SCHOOL OF COMPUTING

Postgraduate Conference 2013

Technical Report

Edited By
Daniel Knox and Jennifer Cooke

University of Kent | Computing

Introduction

It is with great pleasure that we welcome you to the School of Computing's Postgraduate Conference 2013. Students were invited to submit papers or posters of their ongoing research, and we have been delighted with the contributions we received.

The quality of research displayed at this year's conference is a real tribute to the academic excellence of the research community here in the Department of Computer Science at Kent. We are particularly thrilled with the number of paper submissions this year compared with other years, and hope this trend continues in the future. We would like to thank each and every author (of both paper and poster) for taking the time to submit, the level and diversity of work this year is truly outstanding.

At this point, we take the opportunity to thank a few other people who have made this conference possible. Firstly, those students who acted as our review panel, we highly value your efforts and thank you for helping us put together a fantastic array of papers and posters.

Secondly, the academic staff who have supported us throughout this process, namely Sally Fincher, and Richard Jones. Their help and expertise have enabled us to organise this conference to the highest possible standard.

Lastly, but certainly not least, the administration staff who aid us significantly with putting this day together, Mark Wheadon Sandra Shine, Sam McDonagh, and all those within the Course Administration Office. Without your kind support, this conference would just not be possible.

Warm Regards,

Jennifer Cooke and Daniel Knox (Editors.)

May 2013.

Table Of Contents

The design and implementation of a notional machine for teaching introductory programming

Michael Berry
School of Computing
University of Kent
Canterbury, Kent, UK
mjrb5@kent.ac.uk

Michael Kölling
School of Computing
University of Kent
Canterbury, Kent, UK
mik@kent.ac.uk

ABSTRACT

Comprehension of programming and programs is known to be a difficult task for many beginning students, with many computing courses showing significant drop out and failure rates. In this paper, we present a new notional machine design and implementation to help with understanding of programming and its dynamics for beginning learners. The notional machine offers an abstraction of the physical machine designed for comprehension and learning purposes. We introduce the notional machine and a graphical notation for its representation. An evaluation of existing notional machines and their corresponding implementations is given for comparison, and a prototype implementation in BlueJ is described.

1. INTRODUCTION

It is well understood that programming is a fundamental concept of computer science; it is the process by which conceptual ideas are mapped to instructions that can be understood and interpreted by the machine. The teaching of introductory programming within computer science is essential, and mastery of this skill necessary for students to progress.

1.1 Notional Machines

Various studies have shown that learning to program is hard. A summary of some of these is given in Kim & Lerch[6]). Many students fail or drop out of courses that teach introductory programming, with one empirical study quoting a figure of 33%[2]. There have been many reasons proposed as to why this is the case. One popular hypothesis is that students find the concepts of programming too hard to grasp, they do not understand the key properties of their program and how they are controlling them by writing code. This view was presented by Boulay[3], who formalised this into the concept of a *notional machine*. This is a theoretical abstraction designed to provide an easy way of understanding how a particular language or program executes. The notional machine does not need to accurately reflect the exact properties of the real machine, rather it works on a higher conceptual level by providing a metaphorical layer (or indeed several such layers) on top of the real machine that are easier to mentally grasp than the real machine. It is, however, important that the notional machine is consistent and able to explain observed behaviour of the program under observation. The metaphorical layer that the notional machine provides can take on many forms. Visual metaphors are some of the most commonly used, and provide cues representative of the various events that unfold in the actual machine. Visual representations can be replaced or augmented by other media, such as sound.

1.2 The status quo

At present, probably the most common technique for teachers to explain elements of object orientation, and the flow of object-oriented programs, is through the drawing diagrams of objects and classes on a whiteboard. No consistent, complete and widely accepted shared notation exists across classrooms, and it is left to the student to form a mental model based on often ad-hoc diagrams the teacher may use. UML provides a variety of standardised notations, but its dynamic diagrams are not widely used in introductory teaching. Students are often confronted with differing notations to describe the state of a program when switching between teachers, textbooks or tutorials.

One aim of this work is to provide a shared framework and notation in the form of a notional machine that can be used by teachers or lecturers, in textbooks, and in implemented form within a programming tool. This provides learners with a consistent, correct and useful representation to support the formation of a mental model that transfers to a number of contexts and environments. Without the explicit support for the formation of such a model, students form ad-hoc mental models that may be incorrect, incomplete or invalid.

The contribution of this work is two-fold: The first element is the notional machine itself, its abstractions and notation; the second is an implementation of an animated version of this notional machine in a programming tool, BlueJ[7]. The former supports the presentation, discussion and reasoning of program execution, the latter presents an automated software tool that visualises the execution of a Java program in real time. For the purpose of discussion in this paper, we refer to the software implementation of the notional machine

as the BPEV (BlueJ Program Execution Visualiser).

2. RELATED WORK

Several other software systems offer similar functionality to the BPEV. UUhistle[9] is a dynamic, live and programmatic tool which visualises Python programs. Its interface contains a section for the code, the call stack, the current frame, the heap, and separate subsections for classes, functions and operators. A related tool, Jeliot[8], works on a similar conceptual level to UUhistle, but the environment itself is slightly different and works with Java programs rather than Python. It too fulfils the characteristics of being dynamic, live and programmatic. It has an instance area (corresponding to the heap area in UUhistle), a constant area, method area and expression evaluation area. Similarly to UUhistle it also has space for the program code on the left hand side and highlights the subsection of code currently being executed as the execution unfolds. JGrasp[5] and JIVE[4] are two further examples which are particularly interesting in terms of their approach to layout and granularity.

3. RESEARCH QUESTIONS

This work supports two distinct and separate use cases: the *comprehension of programming* and the *comprehension of programs*. The first is most relevant for beginning programmers: the goal here is to understand how a computing system executes program code, the mechanics and details of a programming language and the concepts of the underlying paradigm. Typical questions that the system helps answering in this case are *What does an assignment statement do?* or *How does a method call work?* For experts who have mastered the language this aspect is no longer relevant.

The second use case is to understand and investigate a given program. The goal is to become familiar with a given software system, or to debug a program. Typical questions in this case are *Why does my program behave like this?* or *How many objects are being created when I invoke this method?* This part of the functionality remains relevant even for seasoned programmers.

These use cases lead us to the main aims of the model:

Aim 1 : To provide a shared notation for representing the execution of an object-oriented program within the proposed model.

Aim 2 : To provide a valid mental model for learning and reasoning about object-oriented programming.

Aim 3 : To provide a basis for an implementation in software that can be used to provide a visualisation of the model alongside a running object-oriented program.

These aims further lead us to the two principle research questions:

Research question 1 : Can a consistent, correct and complete model be developed to explain the observed behaviour of the execution of object oriented programs?

Research question 2 : Can an animated form of this model be implemented in software, to provide a visualisation alongside the execution of an object-oriented program?

The term "completeness" in RQ1 is relative to the intended target audience: This work is aimed at beginning learners of programming, and the target group are learners within their first year of study. Therefore, our model is deemed "complete" if it can adequately explain all observed behaviour of programs typically encountered by students in their first year. This scope will have to be defined more precisely at a later stage.

From these research questions, an overall hypothesis is formed for the expected outcome of this work:

Hypothesis : The formation and use of a model as described above is useful for the learning of object-oriented programming and the comprehension of the behaviour of given programs.

This paper presents the design of the notional machine and its prototype implementation, and does not include an evaluation of its effectiveness for learning or teaching. This will be presented separately in a later paper.

4. SCOPE

The BPEV is aimed at first year programming students and therefore focuses on material typically covered within that year. It may also be useful for specific tasks in later years. There are a number of advantages of restricting the scope of the notional machine in this way. One significant advantage is that both BlueJ and the book most commonly used with the system (Objects First With Java[1]) are heavily used by many universities for introductory programming in the first year. By using the material in the book to scope the tool, one can therefore assert that the material covered should be useful and relevant to those same institutions already making use of BlueJ. Other popular introductory programming books are used to set the scope for the tool.

By restricting the intended user group to first year students, several more complex parts of the Java programming language (such as concurrency) can be excluded from the target functionality, leading to a simpler tool design.

5. CURRENT PROTOTYPE

The current prototype is implemented using BlueJ and JavaFX. BlueJ was chosen because it is a popular programming environment, used across many establishments in introductory object oriented programming courses. JavaFX was chosen as an implementation framework since it provides a full Java API, whilst allowing easy access to modern UI features such as custom effects, transitions and animations. JDI is used as the communication library between the virtual machines, since it is the only known existing pure Java framework that performs this task. The prototype is at an interim stage of development that allows us to perform various interesting experiments.

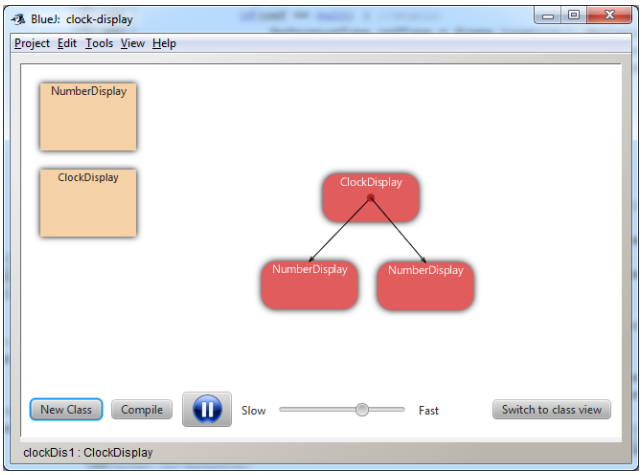

Figure 1: An overview of the current prototype.

5.1 Overview

As a first example, the "clock-display" project from Chapter 3 of the 5th edition of the Objects First textbook[1] has been loaded into the prototype, and a single instance of "ClockDisplay" has been created (Figure 1). The screenshot shows the new main view of the BlueJ environment, with the traditional class view replaced by the BPEV diagram. The diagram presents a unified class and object view, also replacing BlueJ's object bench, which is displayed below the class diagram in the currently available BlueJ system.

The design and layout of the panel containing the controls at the bottom of the screenshot has not yet been finalised. The future placement of the compile button presents a particularly interesting decision, and at this point it is undecided whether it will be integrated into the final implementation or be removed. If it were removed, compilation would be executed as a background process and become invisible to the user, essentially removing this process from the notional machine.

By default, an expanded view of the objects is shown and the classes are "collapsed" down the left hand side of the visualisation. This is the default view while executing a program. When running an object-oriented application, the dynamic relationship between the objects is typically much more useful than the static inheritance relationship between classes, which the classic BlueJ window displays. However, the static class view is still useful, especially when editing the source rather than running the application, and so the user can easily switch to having the classes as the main focus (see Figure 2).

5.2 Notation

In the default diagram for a simple program, classes are represented as peach coloured rectangles and objects are represented using dark red rectangles with rounded corners. This notation has been maintained from the original BlueJ representations. The arrows in this diagram represent references âĂŞ- a single "ClockDisplay" object with references to two separate "NumberDisplay" objects.

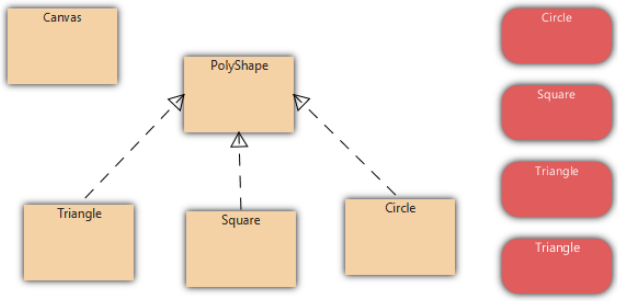

Figure 2: The shapes project with the "classic" expanded classes view. Note the objects have been collapsed down the right hand side of the visualisation, and the classes along with their inheritance relationships take focus.

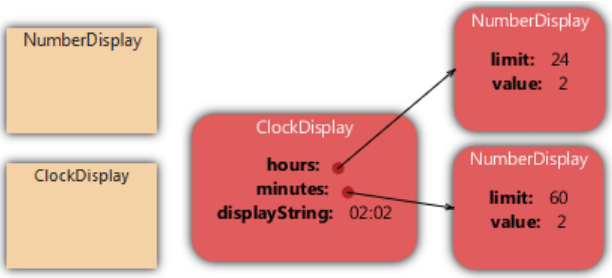

Figure 3: The expanded view of the objects in the notional machine.

One significant difference in representation of objects in this diagram and those in the previous versions of BlueJ on the object bench is the naming scheme. On the object bench, the objects were given unique identifiers (names). This is a conceptual misrepresentation, conflating variables (which have names) with objects (which have not). In fact, objects on the existing BlueJ object bench have partially variable characteristics and partially object characteristics, representing neither concept accurately. This was useful for interacting with these objects, but potentially creates a misconception that objects have a unique name, allowing them to be accessed by any other piece of code which knows this identifier. In the BPEV objects are not named; it is the references which have names. These names are not shown by default, but clicking on any object reveals its "expanded" view, showing the names and state of all its fields, and with it the corresponding references.

5.3 Expanded object view

Figure 3 shows the current expanded view for all the objects currently in the notional machine. The references anchored to the middle of the "ClockDisplay" in the collapsed view now have a clearer focus, they are directly tied to a particular named field (in this case, "hours" and "minutes"). The state of all the fields in the objects are visible. Fields which contain references to other objects are shown as such with a reference arrow; primitive fields are shown by displaying the

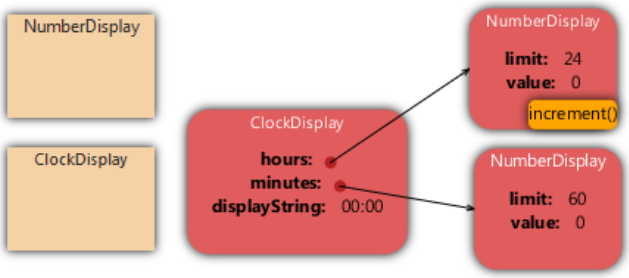

Figure 4: The "increment()" method executing on an instance of "NumberDisplay".

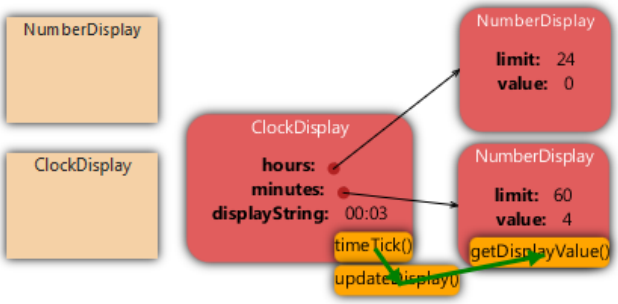

Figure 5: The hybrid object reference and stack trace view as implemented in the current version of the prototype. This particular example has a stack three levels deep, with "getDisplayValue()" being the top-most method in the stack.

corresponding literal. Strings are by default shown inline (as text literals, rather than separate objects), but may be expanded with a change of the conceptual display levels (see section 6.1).

5.4 Methods

As well as displaying the current state of the objects in the notional machine, BPEV also displays the methods currently being active. Methods can be called by right-clicking on the object and then simply selecting the method name from the menu that appears. Figure 4 shows the diagram immediately after the "increment()" method has been called on an instance of "NumberDisplay". Instance methods currently active are displayed as orange rectangles with rounded corners displayed on the bottom right of the object on which they are being called. Static methods are represented similarly, but placed attached to the class representation rather than the object. Arguments, if any are specified, are displayed consistent with the representation of fields on objects; primitives are displayed as literals, and references are displayed with an arrow pointing to the relevant object. The placement of the method invocation display on either object or class visualises the differences between instance methods and static methods. This is a deviation from other systems such as Jeliot, which show instance methods as executing on classes, with a pointer to the relevant object.

5.5 Stack Trace

When a single method is called as in the above example, the method invocation tree is in a simple state, it is thus trivial to deduce from the diagram that a single method has been invoked. However, this increases in complexity when one method calls another, and so on, until a call stack is built up (as in the majority of cases.) When the stack is small it is still often possible to deduce the call sequence by studying the visualisation. However, this quickly becomes impractical or impossible as the size of the stack builds. Other visualisation tools such as JIVE have provided this information by providing a separate "call tree" view, which displays a visualisation of the call sequence at any given moment in time. However, this separation of diagrams requires users to switch between views, remember the state of the previous diagram, and make the relevant connections themselves. The current prototype aims to solve this problem by providing a user-toggled overlay which visualises the stack trace by drawing arrows between the methods active in the current stack (Figure 5).

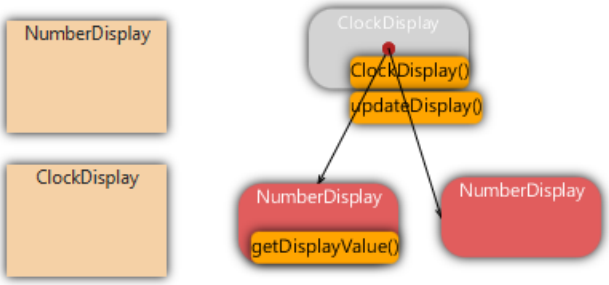

Figure 6: A "ClockDisplay" object whilst being constructed.

This is not as comprehensive as a separate call tree diagram, but allows the user to view the stack trace without switching away from the object view. It is however far from a complete feature and there are a number of challenges with this approach. Perhaps the most obvious is that when the stack trace is large and switches between many methods, the arrow could become difficult to "read", and may cross itself several times. A successful layout algorithm may help to mitigate this problem somewhat, but it is likely the approach will need to be adapted for a large stack trace. The second situation which may prove problematic is the visualisation of recursion – what to display when a method calls itself. For a trivial case where this happens exactly once an arrow "looping back" may well suffice, but the vast majority of recursion cases are more complex, looping over many times. Finally, the direction of the arrow has yet to be decided – whether the stack should point from bottom to top, or top to bottom.

5.6 Constructors

Constructors are currently handled in a similar way to methods, they are displayed in the same format, the only difference being their name is displayed as the name of the class. There is however one notable difference to an object in the notional machine while it is being constructed – while its constructor is still executing, it is displayed in grey rather than its usual deep red colour. This is intended to empha-

sise that the object is not yet fully constructed, and to thus serve as a warning that calling methods or examining the object during this time could result in abnormal behaviour.

6. WORK IN PROGRESS

While some design choices concerning concrete visualisations of the notional machine have been finalised, others have yet to be. These are most notably (but not restricted to) the following two areas:

6.1 Conceptual Levels

It makes sense to show low level operations such as assignments in detail with programs of a few lines, but with programs of a few hundred lines this has far less use. However, that is not to imply useful information cannot be shown with programs on these scales, it is just a different approach that is needed. The proposed notional machine therefore should work across a number of conceptual "zoomed" levels. By zooming in to the most detailed level, the notional machine will describe various low level, atomic operations in detail – so parameters may be shown passing to and from various methods, individual assignments may be represented, and so on. Objects will also likely be represented showing their detail in full – this is already somewhat implemented with the "expanded" view of a particular object, but a *fully* expanded object could show much more detail. The static types of the fields, their visibility level, and any methods could all be shown for an object when viewed in most detail. When this level of detail starts to become too busy from a notional perspective, a conceptual level showing less overall detail can be chosen. This conceptual level, a step down from the most detailed representation, will miss some features – assignments and parameter values may be dropped for instance, and only method invocations may be shown. This "conceptual zooming out" will continue to some form of top level that shows very little detail but may just build up an image of the amount of activity between various objects or object types – a heat-map style view, where the links between the entities could change colour or form depending on the number of times they were utilised. JIVE[4] is interesting with its approach here; it does provide multiple views of granularity. However, they are user-chosen and not integrated on the same diagram.

The details behind the implementation of the conceptual levels in this work have yet to be decided – the choice of how many conceptual levels there are for example and the level of detail that each one should show is still an open question, and one intended to be addressed in the future through experimentation.

6.2 Layout

Existing layout algorithms for similar tools generally position elements within the notional machine either in categories, or in a structured tree-like view. However, this is not necessarily optimal, especially for more complicated use cases where there may be multiple references to a particular object from elsewhere. No existing solutions have been found that aim to combine the object diagram, class diagram and call tree in a single display, further complicating this issue. A desired outcome of this work would be to contribute a new, more optimised layout algorithm specifically for this purpose. A particularly interesting existing tool in this regard is JGrasp[5], which attempts to analyse the underlying code structure and lay its visualisation out according to the purpose of the program.

7. FUTURE WORK

At present the prototype is nearing completion of its first phase, when it contains enough functionality that it can be usefully tested by students. The controls should be refined and the behaviour of the stack trace arrow finalised, then the system should be presented to the target audience and undergo multiple iterations of improvement.

8. SUMMARY

Notional machines exist to provide an easier way for people to understand how a particular program is executing, and thus provide students with a valid way to model the execution of a program. This paper reports on work developing the prototype of a new notional machine in the BlueJ system. This prototype will be used as a platform for ideas and experimentation. At present, the prototype is under heavy development, and detailed decisions on the final notional machine should then be subject to experimentation and feedback from first year students using the prototype.

9. ACKNOWLEDGEMENTS

My supervisor is Michael Kölling at the University of Kent, who has provided great support throughout the course of my PhD thus far.

10. REFERENCES

[1] David Barnes and Michael Kolling. *Objects First with Java: A Practical Introduction Using Bluej.* Pearson Education, 2011.

[2] Jens Bennedsen and Michael E. Caspersen. Failure rates in introductory programming. *SIGCSE Bull.*, 39(2):32âĂŞ36, June 2007.

[3] Du Boulay. Some difficulties of learning to program. *Journal of Educational Computing Research*, 2:57–73, 1986.

[4] Paul V. Gestwicki and Bharat Jayaraman. JIVE: java interactive visualization environment. page 226. ACM Press, 2004.

[5] T. Dean Hendrix and James H. Cross II. jGRASP: an integrated development environment with visualizations for teaching java in CS1, CS2, and beyond. *J. Comput. Sci. Coll.*, 23(2):170âĂŞ172, December 2007.

[6] Jinwoo Kim and F. Javier Lerch. Why is programming (Sometimes) so difficult? programming as scientific discovery in multiple problem spaces. *Information Systems Research*, 8(1):25 –50, March 1997.

[7] Michael Kolling, Bruce Quig, Andrew Patterson, and John Rosenberg. The BlueJ system and its pedagogy. *Computer Science Education*, 13:249–268, December 2003.

[8] AndrÃl's Moreno, Niko Myller, Erkki Sutinen, and Mordechai Ben-Ari. Visualizing programs with jeliot 3. page 373. ACM Press, 2004.

[9] Juha Sorva and Teemu SirkiÃd'. UUhistle. pages 49–54. ACM Press, 2010.

Bayesian Classification via Ant Colony Optimization

Khalid M. Salama
School of Computing
University of Kent
kms39@kent.ac.uk

Alex A. Freitas
School of Computing
University of Ken
A.A.Freitas@kent.ac.uk

ABSTRACT
Bayesian networks (BNs) are powerful tools for knowledge representation and inference that encode (in)dependencies among random variables. A Bayesian network classifier is a special kind of these networks that aims to compute the posterior probability of each class given an instance of the attributes and predicts the class with the highest posterior probability. Since learning the optimal BN structure from a dataset is $\mathcal{NP}$-hard, heuristic search algorithms need to be applied effectively to build high-quality networks. In this paper, we propose a novel algorithm, called ABC-Miner, for learning the structure of BN classifiers using the Ant Colony Optimization (ACO) meta-heuristic. We describe all the elements necessary to tackle our learning problem using ACO, and experimentally compare the performance of our ant-based Bayesian classification algorithm with other algorithms for learning BN classifiers used in the literature.

Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*; H.2.8 [**Database Applications**]: Data mining

General Terms
Algorithms

Keywords
Ant Colony Optimization, Data Mining, Bayesian Network Classifiers

1. INTRODUCTION
Classification is a data mining task where the goal is to build, from labeled cases, a model (classifier) that can be used to predict the class of unlabeled cases. Learning classifiers from datasets is a central problem in data mining and machine learning research fields. While different approaches for tackling this problem exist, such as decision trees, artificial neural networks and rule list [18], our focus in this paper is on the Bayesian approach for classification.

Ant Colony Optimization (ACO) [7] is a meta-heuristic for solving combinatorial optimization problems, inspired by observations of the behavior of ant colonies in nature. ACO has been successful in solving several problems, including classification rule induction [11, 12, 13, 16] and general purpose BN construction [2, 6, 15, 19]. However, as far as we know, it has not been used for learning Bayesian network classifiers.

In this paper, we propose a novel ant-based Bayesian classification algorithm, called ABC-Miner, which learns the structure of a BAN with at most k-dependencies from a dataset using ACO technique for optimization. The rest of the paper is organized as follows. In Section 2 a brief overview on Bayesian networks' basic concepts is given as well as a discussion of various Bayesian network classifiers is shown. Section 3 exhibits the related work on the use of ACO algorithms for building BNs in the literature. In Section 4, we introduce our proposed ABC-Miner algorithm and describe each of the elements necessary to tackle our learning problem using the ACO meta-heuristics. After that, section 5 discusses our experimental methodology and results. Finally, we conclude with some general remarks and provide directions for future research in section 6.

2. BACKGROUND

2.1 Overview on Bayesian Networks
Bayesian networks are knowledge representation tools that aim to model dependence and independence relationships amongst random variables [10]. In essence, BNs are used to describe the joint probability distribution of n random variables $\mathbf{X} = \{X_1, X_2, X_3, ..., X_n\}$. A directed acyclic graph (DAG) is used to represent the variables as nodes and statistical dependencies between the variables as edges between the nodes – child nodes (variables) depend on their parent ones. In addition, a set of conditional probability tables (CPTs), one for each variable, is obtained to represent the parameters Θ of the network. The graphical structure of the network along with its parameters specifies a joint probability distribution over the set of variables $\mathbf{X}$ that is formulated in the product form:

$$p(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} p(X_i | \mathbf{Pa}(X_i), \Theta, G) \qquad (1)$$

where $\mathbf{Pa}(X_i)$ are the parents of variable X_i in G (the DAG that represents the structure of the BN).

Learning a Bayesian network from a dataset $\mathbf{D}$ is decomposed into two phases; learning the network structure, and then learning the parameters of the network. As for parameter learning, it is considered a straightforward process for any given BN structure with specified (in)dependencies between variables. Simply, a conditional probability table (CPT) is computed for each variable with respect to its parent variables. CPT of variable X_i encodes the likelihood of this variable given its parents $\mathbf{Pa}(X_i)$ in the network graph G, and the marginal likelihood of the dataset $\mathbf{D}$ given a structure G is denoted by $P(\mathbf{D}|G)$. The purpose is to find G that maximizes $P(\mathbf{D}|G)$ for a given $\mathbf{D}$, which is the role of BN structure learning phase. The common approach to this problem is to introduce a scoring function, f, that evaluates each G with respect to $\mathbf{D}$, searching for the best network structure according to f. Various scoring metrics are usable for this job [10].

A well-known greedy approach for building BN structure is Algorithm B [1]. It starts with an empty DAG (edgeless structure) and at each step it adds the edge with the maximum increase in the scoring metric f, whilst avoiding the inclusion of directed cycles in the graph. The algorithm stops when adding any valid edge does not increase the value of the scoring metric. K2, a metric based on uniform prior scoring, is one of the most used scoring metrics for building and evaluating Bayesian networks [10].

For further information about Bayesian networks, the reader is referred to [9, 10], which provide a detailed discussion of the subject.

2.2 Bayesian Networks for Classification

Bayesian network classifiers are a special kind of BNs where the class attribute is treated as a unique variable in the network. The purpose is to compute the probability of each value of the class variable given an instance of the predictor attributes and assign this instance to the class that has the highest posterior probability value. Naïve-Bayes consists of a simple BN structure that has the class node as the only parent node of all other nodes. This structure assumes that all attributes are independent of each other given the class. Tree Augmented Naïve-Bayes (TAN) is an extension to Naïve-Bayes that allows a node in a BN to have more than one parent, besides the class variable. This produces a tree-like structure BN. A variation of the Chow-Liu algorithm [3] is the best known method for building TANs. BN Augmented Naïve-Bayes (BAN) is an elaborated version of Naïve-Bayes, in which no restrictions (or at most k-dependencies) are enforced on the number of the parents that a node in the network can depend on. General Bayesian Network (GBN), Unlike the other BN classifier learners, treats the class variable node as an ordinary node. The idea is to build a general purpose Bayesian network, find the *Markov blanket* of the class node, delete all the other nodes outside it and use the resulting network as a Bayesian classifier. Friedman et al. provided an excellent study of these algorithms in [8]. A comprehensive investigation and comparisons of these various Bayesian classifiers by Cheng and Greiner are found in [3, 4] .

3. ACO RELATED WORK

Ant Colony Optimization has an effective contribution in tackling the classification problem. Ant-Miner [13] is the first ant-based classification algorithm. Several extensions on this algorithm have been introduced in the literature, such as AntMiner+ [11], cAnt-Miner [12], and multi-pheromone Ant-Miner [16]. However, the Ant-Miner algorithm as well as its various versions handles the classification problem by building a list of <**IF**_*Antecedent*_**THEN**_*Class*> classification rules. On the other hand, this paper proposes a new ant-based algorithm that handles classification problems, yet with a different approach; learning a Bayesian network to be used as classifier.

As for the use of ACO for building Bayesian networks, to date, there has been only a few research utilizing such a heuristic in learning BN structure, namely: ACO-B [2], MMACO [14, 15], ACO-E [5, 6] and CHAINACO - K2ACO [19]. Moreover, none of them has been used for building BN classifiers. As far as we know, our proposed ABC-Miner is the first algorithm to use ACO, or any evolutionary algorithm, in the task of learning Bayesian networks specific for the classification problem.

Note that the goal of the aforementioned algorithms is to build general purposes BNs. In other words, the selection of the heuristics, quality evaluation metric and other elements of the algorithm are suitable for this aim, but not for building BN classifiers. Hence, in spite of having some similarities, essential aspects of our algorithm are different due to the diversion in the target; our algorithm is only focused on learning BN classifiers. Next we will explore these aspects as we describe our novel Ant-based Bayesian Classifier.

4. A NOVEL ACO ALGORITHM FOR LEARNING BN CLASSIFIERS

4.1 ABC-Miner Algorithm

The overall process of ABC-Miner is illustrated in Algorithm 1. The core element of any ACO-based algorithm is the construction graph that contains the decision components in the search space, with which an ant constructs a candidate solution. As for the problem at hands, the decision components are all the edges $X \rightarrow Y$ where $X \neq Y$ and X, Y belongs to the input attributes of a given training set. These edges represent the variable dependencies in the resulting Bayesian network classifier.

At the beginning of the algorithm, the pheromone amount is initialized for each decision component with the same value. The initial amount of pheromone on each edge is $1/|TotalEdges|$. In addition, the heuristic value for each edge $X \rightarrow Y$ is set using the conditional mutual information, which is computed as follows:

$$I(X, Y|\mathbf{C}) = \sum_{c \in C} p(c) \sum_{x \in X} \sum_{y \in Y} p(x, y|c) \log \frac{p(x, y|c)}{p(x|c)p(y|c)} \quad (2)$$

where $\mathbf{C}$ is the class variable. $p(x, y|c)$ is the conditional probability of value $x \in X$ and $y \in Y$ given class value c, $p(x|c)$ is the conditional probability of x given c, $p(y|c)$ is the conditional probability of y given c and $p(c)$ is the prior probability of value c in the class variable. Conditional mutual information is a measure of correlation between two

random variables given a third one. In our case, we want to lead the ant during the search process to the edges between correlated variables given the class variable, and so we use such a function as heuristic information associated with the selectable edges. Note that the procedure of heuristic calculation is called only once at the beginning and its calculations used throughout the algorithm.

Algorithm 1 Pseudo-code of ABC-Miner.

```
Begin ABC-Miner
BNC_gbest = φ; Q_gbest = 0
InitializePheromoneAmounts();
InitializeHeuristicValues();
t = 0;
repeat
    BNC_tbest = φ; Q_tbest = 0;
    for i = 0 → colony_size do
        BNC_i = CreateSolution(ant_i);
        Q_i = ComputeQuality(BNC_i);
        if Q_i > Q_tbest then
            BNC_tbest = BNC_i;
            Q_i = Q_tbest;
        end if
    end for
    PerformLocalSearch(BNC_tbest);
    UpdatePheromone(BNC_tbest);
    if Q_tbest > Q_gbest then
        BNC_gbest = BNC_tbest;
    end if
    t = t + 1;
until t = max_itrations or Convergence()
return BNC_gbest;
End
```

The outline of the algorithm is as follows. In essence, each ant_i in the colony creates a candidate solution BNC_i, i. e. a Bayesian network classifier. Then the quality of the constructed solution is evaluated. The best solution BNC_{tbest} produced in the colony is selected to undergo local search before the ant updates the pheromone trail according to the quality of its solution Q_{tbest}. After that, we compare the iteration best solution BNC_{tbest} with the global best solution BNC_{gbest} to keep track of the best solution found so far. This set of steps is considered an iteration of the $repeat - until$ loop and is repeated until the same solution is generated for a number of consecutive trials specified by the `conv_iterations` parameter (indicating convergence) or until `max_iterations` is reached. The values of `conv_iterations`, `max_iterations` and `colony_size` are user-specified thresholds. In our experiments (see section 5), we used 10, 500 and 5 for each of these parameters respectively.

4.2 Solution Creation

Instead of having the user selecting the optimum maximum number of dependencies that a variable in the BN can have (at most k parents for each node), this selection is carried out by the ants in ABC-Miner. Prior to solution creation, the ant selects the maximum number of dependencies (k) as a criterion for the currently constructed BN classifier. This selection of k value is done probabilistically from a list of available numbers. The user only specifies `max_parents` parameter (that we set to 3 in our experiments), and all the integer values from 1 to this parameter are available for the ant to use in the BN classifier construction. The various values of the k are treated as decision components as well. More precisely, the ant updates the pheromone on the value k of the maximum number of parents after solution creation according to the quality of this solution, which used value k as a criterion in the BN classifier construction. This pheromone amount represents the selection probability of this value by subsequent ants, leading to convergence on an optimal value of k dependencies. Algorithm 2 shows the outline of the solution creation procedure.

Algorithm 2 Pseudo-code of Solution Creation Procedure.

```
Begin CreateSolution()
BNC_i ← {Naïve-Bayes structure};
k = ant_i.SelectMaxParents();
while GetValidEdges() <> φ do
    {i → j} = ant_i.SelectEdgeProbablistically();
    BNC_i = BNC_i ∪ {i → j};
    RemoveInvalidEdges(BNC_i, k);
end while
BNC_i.LearnParameters();
return BNC_i;
End
```

Each ant starts with the network structure of the Naïve-Bayes classifier, i. e. a BN in which all the variables have only the class variable as a parent. From that point, it starts to expand this Naïve-Bayes network into a Bayesian Augmented Naïve-Bayes (BAN) by adding edges to the network. The selection of the edges is performed according to the following probabilistic state transition formula:

$$P_{ij} = \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{a}^{I} \sum_{b}^{J} [\tau_{ab}(t)]^{\alpha} \cdot [\eta_{ab}]^{\beta}} \quad (3)$$

In this equation, P_{ij} is the probability of selecting the edge $i \rightarrow j$, $\tau_{ij}(t)$ is the amount of pheromone associated with edge $i \rightarrow j$ at iteration t and η_{ij} is the heuristic information for edge $i \rightarrow j$ computed using conditional mutual information (equation 2). The edge $a \rightarrow b$ represents a valid selection in the available edges. The exponents α and β are used to adjust the relative emphases of the pheromone (τ) and heuristic information (η), respectively. Note that edges available for selection are directed, i. e. $i \rightarrow j \neq j \rightarrow i$.

ABC-Miner adapts the "ants with personality" approach, proposed by the author in [16]. Each ant_i is allowed to have its own *personality* by allowing it to have its own values of the α_i and β_i parameters. In other words, some ants will give more importance to pheromone amount, while others will give more importance to heuristic information. The α_i and β_i parameters are each independently drawn from a Gaussian distribution centered at 2 with a standard deviation of 1. This approach aims to advance exploration and improve search diversity in the colony.

An edge $i \rightarrow j$ is valid to be added in the BN classifier being constructed if the following two criteria are justified: 1) its inclusion does not create a directed cycle, 2) the limit of k parents (chosen by the current ant) for the child variable j is not violated by the inclusion of the edge. After the ant

adds a valid edge to the BN classifier, all the invalid edges are eliminated from the construction graph. The ant keeps adding edges to the current solution until no valid edges are available. When the structure of BNC_i is finished, the parameters Θ are learnt by calculating the CPT for each variable, according to the network structure, producing a complete solution. Afterward, the quality of the BN classifier is evaluated, and all the edges become available again for the next ant to construct another candidate solution.

4.3 Quality Evaluation and Pheromone Update

Unlike the traditional Bayesian networks, the target of our algorithm is to build an effective BN in terms of predictive power with respect to a specific class attribute. In other words, BN learning algorithms aim to maximize a scoring function that seeks a structure that best represents the dependencies between all the attributes of a given dataset. This structure should fit the knowledge representation and inference purposes of a BN, which treats all the variables in the same way, without distinguishing between the predictor and the class attributes. On the other hand, the purpose of learning a BN classifier is to build a structure that can calculate the probability of a class value given an instance of the input predictor variables, and predict the class value with the highest probability to label the instance.

Therefore, using traditional scoring functions to evaluate the quality of a BN classifier should not fit the purpose of building a classifier [8]. According to this reasoning, we evaluate the quality of the constructed network directly as a classifier, where the predictive efficiency is the main concern. We use the *accuracy*, a conventional measure of predictive performance, to evaluate the constructed BN model, computed as follows:

$$Accuracy = \frac{|Correctly\_Classified\_Cases|}{|Validation\_Set|} \quad (4)$$

The best BN classifier BNC_{tbest} constructed amongst the ants in the colony undergoes local search, which aims to improve the predictive accuracy of the classifier. The local search operates as follows. It temporarily removes one edge at a time in a reverse order (removing last the edge that was added to the network first). If the quality of the BN classifier improves, this edge is removed permanently from the network, otherwise it is added once again. Then we proceed to the next edge. This procedure continues until all the edges are tested to be removed from the BN classifier and the BN classifier with the highest quality – with respect to classification accuracy – is obtained.

After BNC_{tbest} is optimized via local search, pheromone levels are increased on decision components (edges) in the construction graph included in the structure of the constructed BN classifier, using the following formula:

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \tau_{ij}(t).Q_{tbest}(t) \quad (5)$$

To simulate pheromone evaporation, normalization is then applied as in [13]; each τ_{ij} is divided over the total pheromone amounts in the construction graph. Note that pheromone update is carried out for the decision components representing the number of dependencies used for building the BN classifier structure as well.

5. EXPERIMENTS AND RESULTS

The performance of ABC-Miner was evaluated using 15 public-domain datasets from the UCI (University of California at Irvine) dataset repository [17]. Datasets containing continuous attributes were discretized in a pre-possessing step, using the C4.5-Disc [18] algorithm. We compare the predictive accuracy of our proposed ant-based algorithm with three other widely used algorithms for learning Bayesian classifiers. In our experiment, we used Weka [18] implementations for these algorithms.

The experiments were carried out using 10-fold cross validation procedure. In essence, a dataset is divided into 10 mutually exclusive partitions, were each time a different partition is used as the test set and the other 9 partitions are used as the training set. The results (accuracy rate on the test set) are then averaged and reported in Table 3 as the accuracy rate of the classifier. Since ABC-Miner is a stochastic algorithm, we run it 10 times – using a different random seed to initialize the search each time – for each cross-validation fold. In the case of the deterministic algorithms, each is run just once for each fold.

Table 3 reports the mean and the standard error of predictive accuracy values obtained by 10-fold cross validation for the 15 datasets, where the highest accuracy for each dataset is shown in bold face. As shown, ABC-Miner has achieved the highest predictive accuracy amongst all algorithms in 12 datasets (with 2 ties), while Naïve-Bayes achieved the highest accuracy in 3 datasets (with 2 ties), TAN in 2 datasets (both are ties) and finally GBN in 4 datasets (with 3 ties).

Ranking the algorithms in descending order of accuracy for each dataset and taking the average ranking for each algorithm across all 15 datasets, ABC-Miner obtained a value of 1.6, which is the best predictive accuracy average rank amongst all algorithms. On the other hand Naïve-Bayes, TAN and GBN have obtained 3.1, 2.5, 2.8 in predictive accuracy average rank respectively. Note that the lower the average rank, the better the performance of the algorithm.

Statistical test according to the non-parametric Friedman test with the Holm's post-hoc test was performed on the average rankings. Comparing to Naïve-Bayes and GBN, ABC-Miner is statistically better with a significance level of 5% as the tests obtained p - values of 0.0018 and 0.013 respectively. Comparing to TAN, ABC-Miner is statistically better with a significance level of 10% as the tests obtained p -value of 0.077.

6. CONCLUDING REMARKS

In this paper, we introduced a novel ant-based algorithm for learning Bayesian network classifiers. Empirical results showed that our proposed ABC-Miner significantly out performs the well-known Naïve-Bayes, TAN, and GBN algorithms in term predictive accuracy. Moreover, the automatic selection of the maximum number of k-parents value makes ABC-Miner more adaptive and autonomous than conventional algorithms for learning BN classifiers. As a future work, we would like to explore the effect of using different scoring functions for computing the heuristic value used by ABC-Miner, as well as other scoring functions to evaluate the quality of a constructed BN classifier. Another direc-

Table 1: Predictive Accuracy % (*mean* ± *standard error*) Results.

Dataset	Naïve-Bayes	TAN	GBN	ABC-Miner
bcw	92.1 ± 0.9	**95.4 ± 0.9**	93.8 ± 0.9	**95.4 ± 0.6**
car	85.3 ± 0.9	93.6 ± 0.6	86.2 ± 0.9	**97.2 ± 0.3**
cmc	52.2 ± 1.2	49.8 ± 1.2	49.8 ± 1.2	**67.3 ± 0.6**
crd-a	77.5 ± 1.2	85.1 ± 0.9	85.7 ± 0.9	**87.3 ± 0.6**
crd-g	**75.6 ± 0.9**	73.7 ± 1.2	**75.6 ± 1.2**	69.5 ± 0.9
drm	96.2 ± 0.6	97.8 ± 0.9	97.2 ± 0.6	**99.1 ± 0.3**
hay	80.0 ± 2.8	67.9 ± 3.1	**83.1 ± 2.5**	80.0 ± 3.1
hrt-c	56.7 ± 2.2	58.8 ± 2.5	56.7 ± 2.2	**73.3 ± 0.9**
iris	**96.2 ± 1.5**	94.2 ± 1.8	92.9 ± 1.8	**96.2 ± 0.9**
monk	**61.6 ± 0.6**	58.8 ± 0.6	**61.6 ± 0.9**	51.9 ± 0.9
nurs	90.1 ± 0.9	94.3 ± 0.9	90.1 ± 0.9	**97.0 ± 0.9**
park	84.5 ± 2.5	91.7 ± 2.2	84.5 ± 2.5	**94.2 ± 2.8**
pima	75.4 ± 1.2	**77.8 ± 1.5**	**77.8 ± 1.5**	**77.8 ± 1.5**
ttt	70.3 ± 0.3	76.6 ± 0.6	70.3 ± 0.3	**86.4 ± 0.6**
vot	90.3 ± 0.6	92.1 ± 0.4	90.3 ± 0.6	**94.6 ± 0.9**

tion is to explore different methods of choosing the value of k parents for building a network structure for the Bayesian classifier.

7. REFERENCES

[1] Buntine, W. : Theory refinement on Bayesian networks. *17th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann*, pp. 52–60 (1991).

[2] De Campos, L.M., Gámez, J.A., Puerta, J.M. : Learning Bayesian network by ant colony optimisation. *Mathware and Soft Computing*, pp. 251–268 (2002).

[3] Cheng, J. and Greiner, R. : Comparing Bayesian network classifiers. *15th Annual Conference on Uncertainty in Artificial Intelligence*, pp. 101–108 (1999).

[4] Cheng, J. and Greiner, R. : Learning Bayesian Belief Network Classifiers: Algorithms and System. *14th Biennial Conference: Advances in Artificial Intelligence*, pp. 141–151 (2001).

[5] Daly, R., Shen, Q., Aitken, S. : Using ant colony optimization in learning Bayesian network equivalence classes. *Proceedings of UKCI*, pp. 111–118 (2006).

[6] Daly, R. and Shen, Q. : Learning Bayesian network equivalence classes with ant colony optimization. *Journal of Artificial Intelligence Research*, pp. 391–447 (2009).

[7] Dorigo, M. and Stützle, T. : Ant Colony Optimization. *MIT Press*, (2004).

[8] Friedman, N., Geiger, D., Goldszmidt, M. : Bayesian Network Classifiers. *Machine Learning Journal*, pp. 131–161 (1997).

[9] Friedman, N. and Goldszmidt, M. : Learning Bayesian networks with local structure. *Learning in Graphical Models, Norwell, MA: Kluwer*, pp. 421–460 (1998).

[10] Heckerman, D., Geiger, D., Chickering, D.M. : Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learning Journal*, pp. 197–244 (1995).

[11] Martens, D., Backer, M.D., Haesen, R., Vanthienen, J., Snoeck, M., Baesens, B. : Classification with ant colony optimization. *IEEE TEC*, pp. 651–665 (2007).

[12] Otero, F., Freitas, A., Johnson, C.G. : *c*Ant-Miner: an ant colony classification algorithm to cope with continuous attributes. *Ant Colony Optimization and Swarm Intelligence*, pp. 48–59 (2008).

[13] Parpinelli, R.S., Lopes, H.S., Freitas, A. : Data mining with an ant colony optimization algorithm. *IEEE TEC*, pp. 321–332 (2002).

[14] Pinto, P.C., Nägele, A., Dejori, M., Runkler, T.A., Costa, J.M. : Learning of Bayesian networks by a local discovery ant colony algorithm. *IEEE World Congress on Computational Intelligence*, pp. 2741–2748 (2008).

[15] Pinto, P.C., Nägele, A., Dejori, M., Runkler, T.A., Costa, J.M. : Using a Local Discovery Ant Algorithm for Bayesian Network Structure Learning. *IEEE TEC*, pp. 767–779 (2009).

[16] Salama, K.M., Abdelbar, A.M., Freitas A.A. : Multiple pheromone types and other extensions to the Ant-Miner classification rule discovery algorithm. *Swarm Intelligence Journal*, pp. 149–182 (2011).

[17] UCI Repository of Machine Learning Databases. Retrieved Oct 2011 from, URL:http://archive.ics.uci.edu/ml/index.html

[18] Witten, H. and Frank, E. Data Mining: Practical Machine Learning Tools and Techniques. *second edition, Morgan Kauffman*, (2005).

[19] Yanghui, Wu., McCall, J., Corne, D. : Two novel Ant Colony Optimization approaches for Bayesian network structure learning. *IEEE World Congress on Evolutionary Computation*, pp. 1–7 (2010).

Centrality Measures and Location of Structures

Akanmu, A. A. G
School of Computing
Medway Campus,
University of Kent
(aaga2@kent.ac.uk

ABSTRACT
The basis of centrality measures lies in the formal theory of social network mechanism, and this is largely implied here in this work on the mergers of centrality measures in the study of weights' combination to evaluate network topologies, in the physical, social and cybernetic aspects of cloud computing. This work aims to study the basic principles of centrality and evolve new methods that would be applied to solve problem of resource allocation or citing of a distribution centre to augment the existing ones whenever the need arises for a supply chain network.

Categories and Subject Descriptors
J. Computer Applications

J.1 Administrative Data Processing

Subject Descriptor - Business

General Terms
Management

Keywords
Centrality, Geodesic Paths, Degree of Nodes, Weights of Edges, Topology

1. INTRODUCTION

Degree Centrality

The degree centrality has as part of its advantage that only local structure round the node could be known, for ease of calculation this is acceptable, but it becomes a concern if a node is central and not easily accessible to other nodes for one reason or another. Zhuge (2010) described the degree centrality as shown in (i) below:

$$C_D(s) = \frac{\deg ree(s)}{n-1} \qquad (i)$$

where s = node and n = total number of nodes in the graph.

Closeness Centrality

Closeness centrality takes the distance from the focal node to other nodes in a graph into consideration but it has the setback of not taking disconnected nodes (those who are not in the same graph) into consideration. Zhuge (2010) formally expresses closeness centrality as

$$C_C(s) = \frac{n-1}{\sum_{t \in S \setminus s} \mathrm{d_G(s,t)}} \qquad (ii)$$

where n = number of nodes, $d_G(s,t)$ = geodesic distance between s and t. Zhuge (2010) formally expresses betweenness centrality as

$$C_B(v) = \sum_{\substack{s \neq v \neq t \in v \\ s \neq t}} \frac{\sigma_{st}(v)/\sigma_{st}}{(n-1)(n-2)} \qquad (iii)$$

where σ_{st} is the number of the shortest geodesic paths from s to t, and σ_{st} (v) is the number of the shortest geodesic paths from s to t that pass through node v. Since weights are of importance, the equations in *(i)* to *(iii)* above can be extended to include weights, therefore, the weighted degree centrality of node *s* is hereby represented by

$$C_D^W(s) = \frac{\sum^{n} w_{st}}{n-1} \qquad (iv)$$

where w_{st} is the sum of the weights of edges connected to the particular source node *s* and , *t* represents a particular target node. In the same vein, the weighted closeness centrality, $C_c^w(s)$ is also represented by

$$C_C^W(s) = \frac{n-1}{\sum d(s,t)} \qquad (v)$$

which is the weight of geodesic paths between s and t, while the weighted betweenness centrality is

$$C_B^W(v) = \sum_{\substack{s \neq v \neq t \in v \\ s \neq t}} \frac{\sigma_{st}^w(v)/\sigma_{st}^w}{(n-1)(n-2)} \qquad (vi)$$

where σ_{st} is the number of the shortest geodesic paths from s to t, σ_{st} (v) is the number of the shortest geodesic paths from s to t that pass through node v and w is the assigned weights to the ties.

The relative importance of the number of ties compared to the weights on the ties is referred to as a tuning parameter α (Opsahl, 2010).

2. GRAPH OF INTEREST

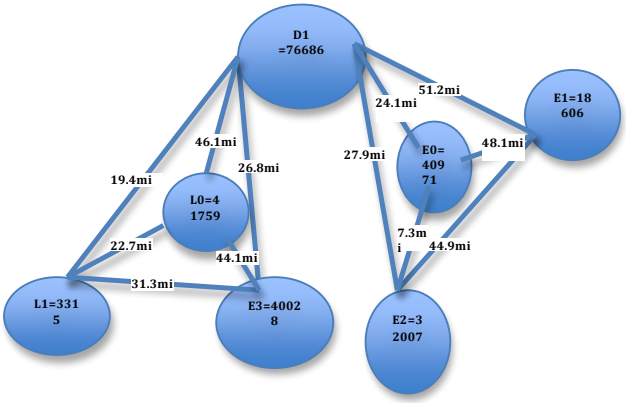

Fig.1 Figure showing locations of some sales outlets and the closest distribution centre D1, sales values are inscribed per node and the ties represent distances between nodes.

3. Mixed-Mean Centrality

Mixed-Mean Centrality =

$$\frac{1}{3n}\left\{\sum_{n=1}^{k}(CBW_n + CDW_n + CCW_n)\right\} \qquad \text{(iv)}$$

where, CBWn, CDWn & CCWn are the weighted Betweenness, Degree and Closeness Centralities of the graph n, where n > 0 respectively. The equation (v) below models the graph in fig.1

$$c_d^{w\alpha\beta}(i) = k_i \times \left(\frac{s_i}{k_i}\right)^{\alpha} = k_i^{(1-\alpha)} \times s_i^{\alpha} \times z_i^{\beta} \qquad \text{(v)}$$

where k_i = degree of nodes ; $s_i = C_D^W$ (s) as defined in (iii) above

z_i = weight of nodes, where $\alpha \geq 0$; $\{\beta \in Z : -1 \leq \beta \leq 1\}$

The same argument goes for $CC^{w\alpha_\beta}$ and $CB^{w\alpha_\beta}$ The table below shows the output result of (v) applied in (iv):

Table1. Table showing the results of the averages for the weighted degree centralities at the weight of nodes z1 and z2

				β=1	FOR Z1	
	No of degrees	WEIGHTS ON THE NODES OF GRAPH 1	MIXED-MEAN CENTRALITY @ DIFF α FOR Z1 AND BETA=1			
NODE		Z1	α=0	α=½	α=1	α=1½
D1	6	176686	1995570	1994150	1988871	1985740
L0	3	41759	271201	272004	272132	274264
L1	3	3315	14254.5	14166.3	14052.3	13953.2
E0	3	40971	190060	189013	188392	188581
E1	3	18606	153189	153951	154980	155718
E2	3	32007	149544	148913	148540	148213
E3	3	40028	236165	228146	235411	235710

4. CONCLUSION

As reflected in the table1 above, the node D1 which is the distribution centre has the highest centrality of all the nodes irrespective of the value of alpha, this is not unexpected as it serves all the outlets, so it has the highest, that is , the combined values of Z1 for all other nodes. However among all the other nodes, despite the fact that they all have same number of degrees, node L0 was consistently the highest ranking followed by E3 with Z1=40028, even though node E0 has a higher value of Z1 = 40971. So, in terms of decision relating to siting of new distribution centre, node L0 or an area closest to it will be the most appropriate to cite the new centre.

Graph was considered and effects of the combined weights on edges (distances connecting the nodes) and nodes (sales values and/or unit of goods sold) were evaluated taking the betweenness, closeness and degree centrality into cognisance.

The resulting Mixed-Mean centrality made use of tuning parameters α and *β*. This new centrality was later applied to a supply chain, precisely a subset consisting of the sales outlets of an enterprise - TESCO.

The scenario presented here can be applied to cloud computing by using the idea of mixed-mean centrality to discover the most central and/or most viable nodes in terms of resource allocation thus minimising costs and maximizing profits.

It can be used in locating the performance level of a particular node or edge and thus aiding in decision on which node or edge deserves attention, e.g. for security and fault-tolerance purposes.

For the purpose of green initiative, the most central node becomes the one that will be focused upon as much energy is expected to be consumed within and around such nodes.

In disease control, it can be used to detect the most central region where epidemic diseases are proned to spread easily or to find the most vulnerable group in the society to an epidemic disease.

5. REFERENCES

[1] Akanmu, A.A.G ., Wang, F.Z & Huankai, C.(2012). Introducing Weighted Nodes To Evaluate The Cloud Computing Topology. Special Issue (Vol. 5 No. 11A, 2012) of JSEA. www.scirp.org/journal/jsea

[2] Freeman, L.C. (1978) Centrality in Social Networks: Conceptual Clarification. Social Networks 1, 215-239. Accessed from http://moreno.ss.uci.edu/27.pdf on 06 June, 2012

[3] Freeman, S.C., Freeman, L.C (1979). The Networkers Network: A study of the impact of a new communications medium on sociometric structure. Social Science Research Reports S46. University of California, Irvine, CA.

[4] Opsahl, T.; Agneessens, F. & Skvoretz, J. (2010). Node Centrality in Weighted Networks: Generalizing Degree Shortest Paths. Social Networks 32 (2010) 245-251.

[5] Zhuge, H. & Zhang, Junsheng (2010). Topological Centrality and It's e-Science Applications. Wiley Interscience.

[6] TESCO Depot Finder: accessed on 27 Mar 2013 from http://www.tesco-careers.com/home/you/depots/picking-team/pickers

A New Design and Evaluation of Memristor-CAM (Content Addressable Memory) Cell

Wanlong Chen
School of Computing
University of Kent
UK
E-mail: W.Chen@kent.ac.uk

ABSTRACT
Content addressable memory (CAM) is one kind of the associative memory; it can be used in several high-speed searching applications. We propose a new CAM cell that employs memristors (memory resistor) as the storage element and the main part of the comparator. This presented new CAM cell involves two memristors; one works as a storage element and the other one is used as an important part of the comparator instead of the conventional logic gate. Based on the unique characters of the memristor, we explore the new Memristor-CAM cell design to analyze the possibility to build memristor-based storage structures with lower power consumption to achieve the purpose of saving energy and decreasing the searching time.

General Terms
Design

Keywords
Content Addressable memory, CAM, memristor, memory.

1. MEMRISTOR-CAM CELL
The conceptual circuitry (Fig. 1) shows a new memristor CAM cell that stores data using single memristor which can be stored bit '0' or '1' [1]. Apparently, from Fig. 1, data can be read from the cell through the line ($V_{c/r}$); data can be written or deleted into the cell through the second line ($V_{w/d}$); the line of (V_{search}) is responsible for the transmission of the data that the user would like to search. The comparator can analyze and return the result of "match" or "mismatch" once the searched data and stored data have been compared.

2. CONCLUSION
In this novel design, one memristor is used to replace the conventional storage element; with memristors' intrinsic feature of resistance switching, the other one is utilized as a conventional logic gate of the comparator . With the benefits of memristors, this new CAM cell design reduces the power consumption, storage element size and design complexity compared with traditional storage elements. The proposed CAM cell design is the initial step to develop a general and large scale CAM memory based on memristors.

3. REFERENCES
[1] XiaoYang , Wanlong Chen, Frank Zhigang Wang. A Memristor-CAM (Content Addressable Memory) Cell: New Design and Evaluation. International Conference on Computer Science and Information Technology, (2013): 1045-1048.

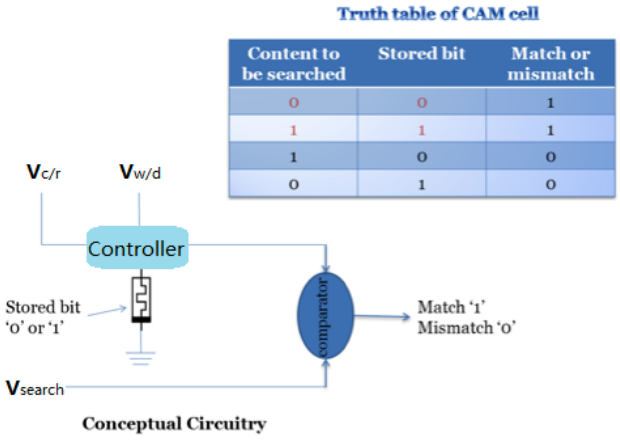

Fig. 1 The Memristor-CAM cell conceptual circuitry and its truth table. A table shown at top right corner is the truth table of the comparison process of the memristor-CAM cell. $V_{c/r}$ indicates the voltage for controlling the controller and reading the memristor. $V_{w/d}$ indicates the voltage for writing the deleting (or resetting) the memristor. Apparently, the different signals will give 'mismatch', otherwise 'match' will be given

Interactions in Consciousness:

Can conscious processing be inferred from neuropsychological measurements?

Jennifer Cooke
School of Computing
University of Kent
Canterbury.
J.Cooke@kent.ac.uk

Amirali Shirazibeheshti
School of Computing
University of Kent
Canterbury
A.Shirazibeheshti@kent.ac.uk

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

ABSTRACT

The human conscious experience has been a topic of debate for centuries. What was once a philosophical discussion of abstract concepts, is now a scientific exploration into the human brain. In an attempt to better understand the most fundamental part of the human psyche, neuropsychology and philosophy have evolved to focus on patterns of brain activity as a means of establishing the source, or at least a marker, of human consciousness. In this short paper we will discuss an auditory paradigm, devised by Bekinschtein et al. (2009), and designed to characterize conscious awareness. A potential flaw with using this method is highlighted.

Keywords

consciousness, EEG, global-local paradigm, mismatch negativity, P300 component, disorders of consciousness.

1. INTRODUCTION

So what is consciousness? What does it mean to be conscious? In the context of the current research we refer to consciousness as conscious awareness, that is being immediately aware of what one perceives. Though this is an intuitive concept, Neuroscience has yet to capture it empirically. Being able to diagnose states of consciousness has a specific clinical relevance to those with Disorders of Consciousness (DOC). A DOC can be characterized as any medical condition that inhibits consciousness, most commonly a coma state as the result of a traumatic accident. Patients with a DOC are diagnosed into one of two main categories: vegetative state (VS), that is awake but not aware, or the more recently classified minimally conscious state (MCS), awake and transiently aware. Practitioners rely heavily on behavioral observations to make this diagnosis, which can be deeply flawed, as behavior will vary substantially between patients. Therefore, successful diagnosis and treatment is extremely difficult to achieve. At present, around 40% of patients are misdiagnosed as VS when they are actually minimally conscious [2]. This means that practitioners (by no fault of their own) are misdiagnosing the state of consciousness in these patients, and as such they are failing to meet their needs, particularly regarding pain management. With advancing technology, it is now possible to use Electrocephalography (EEG) and neuroimaging techniques (such as Magnetic Resonance Imaging [MRI]), to record brain activity, or lack thereof, in people within low awareness states. Using such techniques it may be possible to detect a marker, or neural correlate, of consciousness in these patients. Something clinicians can use as a diagnostic aid. If this is possible, then it seems plausible that we may use this information to find ways of communicating with these patients; some already have (see [3]). Thus, refining a diagnostic criteria for conscious awareness is pivotal for improving the quality of life for these patients.

2. EEG AND EVENT RELATED POTENTIALS (ERPs)

EEG involves placing electrodes on the scalp, which detect electrical dipoles in the brain, generated by neural activity. Much of the activation captured by this method comes from the cortex and outer layers of the brain. In studies of consciousness one is particularly interested in Event Related Potentials (ERPs). ERPs are quite simply the average EEG signal across many trials observed in response to an external stimulus. In order to capture this response, the EEG signal is time-locked to the stimulus onset, thus you capture the signal evoked by a certain event.

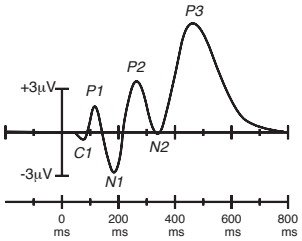

Figure 1. Averaged ERP waveform

Two ERP components that have been linked with consciousness are the P300 (P3) and Mismatch Negativity (MMN). MMN is thought to signify the brain's awareness of incongruency, for example the presentation of two unrelated stimuli. MMN occurs at frontal electrodes around 200ms after stimulus presentation, and is believed to be the fast and automatic response to detecting incongruence (see N1 on Figure 1). The P300 (P3), a positive peak at central electrodes around 300ms after stimulus presentation, it has been associated with the updating and regulation of working memory [4]. Therefore, it is considered to involve higher-level cognitive processes, and thus is more closely linked to the presence of conscious perception and awareness (see Figure 1).

3. THE GLOBAL-LOCAL PARADIGM

Bekinschtein et al. (2009) devised an auditory irregularity paradigm they call the global-local task, to investigate the relationship between consciousness, MMN, and the P300 [5]. The experimental design is comprised of sequences of tones either consistent in frequency or containing one deviant tone of a higher frequency. The deviant tone (if present) is always the last tone in a

sequence. Tones must be presented in sequences of five (four standard tones and a fifth tone, either standard or deviant in nature). Five sequences, each containing five tones, creates a sub-block. Around twenty-four sub-blocks create a block (see Figure 2). Each block lasts around three minutes.

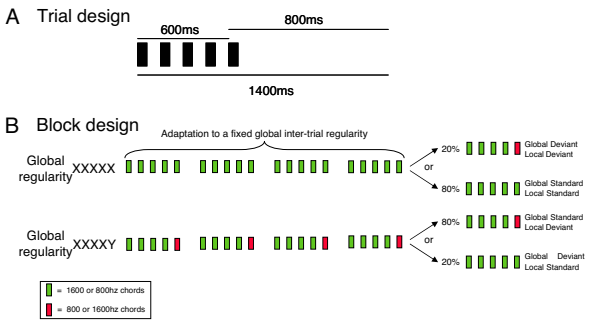

Figure 2. The global-local auditory paradigm

Bekinschtein et al. (2009) propose that this paradigm contains both a local violation of auditory regularity (that is the fifth tone of a sequence can be deviant), and a global violation of auditory regularity (that is, the fifth sequence of a sub-block is deviant when it violates the context of the four preceding sequences). It is theorized that a local violation of auditory regularity generates MMN, because the detection of this incongruency only requires a very basic amount of cognitive processing, that is the realization that the tone is different to the preceding four [5]. On the other hand, Bekinschtein et al. (2009) suggest that a global violation of auditory regularity requires a higher-level of cognitive processing, and therefore is associated with a P3 response. This is because the participant has to recognize the global context in which this violation occurs. Thus, Bekinschtein et al. (2009) are able to examine local effects (standard vs. deviant), and global effects (standard vs. deviant) separately with this paradigm.

4. INTERACTIONS

Bekinschtein et al. (2009) assume that their local (MMN) and global (P3) effects are independent of one another, that is they have no baring on eachother. However, the current investigation found that the size of the global effect depends on the level of the local effect; meaning the two components actually interact. This interaction could suggest that in actual fact the global effect is facilitated by the local effect. In essence, when there is both a local violation of auditory regularity, and a global violation of auditory regularity (global deviant local deviant), the amplitude of MMN (see Figure 3), and the P3 (see Figure 4) is much higher, compared with when there is no local violation of auditory regularity (global deviant local standard).

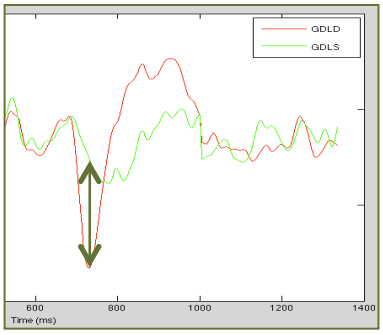

Figure 3. Difference in size of MMN at electrode Fz (mV)

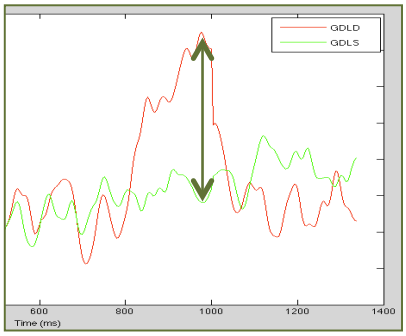

Figure 4. Difference in size of P3 at electrode Pz (mV)

Statistical analysis revealed that the interaction was significant between local and global effects around 300ms after the stimulus was presented at 600ms (see Figure 5).

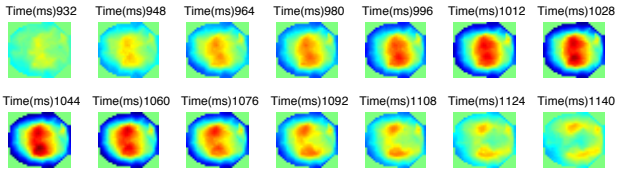

Figure 5. Statistical T-maps ($p < .05$)

5. CONCLUSION

ERP analysis revealed the size of the global effect was dependent upon the level of the local effect. This is evidence the two effects interact. Statistical analysis then confirmed that this interaction was significant ($p<.05$) around 300ms after stimulus presentation. Therefore, it is proposed that the global (P3) and local (MMN) effects are not mutually exclusive, as postulated by Bekinschtein et al. (2009), and consequently, the global effect (P3) cannot reliably be used as a diagnostic marker for conscious awareness.

6. REFERENCES

[1] Treisman, A. 2003. Consciousness and perceptual binding. *The unity of consciousness: Binding, integration, and dissociation*, 95-103.

[2] Giacino, J. T., et al. 2002. The Minimally Conscious State – Definition and Diagnostic Criteria. *Neurology*. 58. 3. 349-353.

[3] Cruse, D., et al. 2011. Bedside detection of awareness in the vegetative state: a cohort study. *The Lancet*. 378. 9809. 2088-2094.

[4] Morgan, H., et al. 2008. Working memory for faces modulates P300, N170, and N250r. *Journal of Cognitive Neuroscience*. 20. 6. 989-1002.

[5] Bekinschtein, T., et al. 2009. Neural signature of the conscious processing of auditory regularities. *PNAS*. 106. 5. 1672-1677.

Computing Neuronal Saliency by Relevance Assessment

Keith A. Greenhow
University of Kent
Canterbury, Kent
kg246@kent.ac.uk

ABSTRACT

The objective of Neuronal Saliency is to determine the relevance of a node to the output of a Artificial Neural Network. This information could be used to interpret and analysis the network or for optimisation purposes, such as modifying its topology to reduce the number of nodes.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Connectionism and neural nets*

General Terms

Artificial Neural Networks, Neural Saliency, Relevance Assessment

Keywords

Relevance, Saliency

1. INTRODUCTION

The majority of the research for determining the relevance of a neurons is to either for the pruning of low ranked neurons and inputs to reduce complexity or for the understanding and modelling of the final solution learned by the network. For my work, I require a method that runs alongside learning/training of the ANN such that I can modify the interpolation of the input data in real-time.

1. **Which Neurons in a ANN can be said to be of low relevance to the networks function?**
2. **Can this be generalised to all nodes of an ANN, thus including inputs?**
3. **Could it be performed during learning to aid input interpolation?**

2. BACKGROUND

An Artificial Neural Network (ANN) is an interconnected network of processing units, based upon connectionist models of natural neural networks, such as the brains and spinal cords of animals. Each of the processing units applies a simple set of rules to its input data along with unique weights the can stimulate (via excitation) or suppress (via inhibition) the processing units own output. This level to which the node is stimulated is then propagated along to the next layer of processing units. Each layer can contain an arbitrary number of processing units. The input layer is the only layer in an ANN that does not contain processing units, rather it contains *input nodes* that act as a preparation and buffering units for the data input into the network. These input nodes and the subsequent processing units are collectively known as *nodes*, a term taken from graph theory due to their typical depiction. The last layer in the network is the output layer. This layer provides the output of the network in the form of a numerical vector. Any remaining layers between the input and output layers are known as hidden layers. This term comes from the behaviourists model of the brain, due to the hidden nature of the functioning of the brain. Techniques used to prune a neural network during or after learning most have some method to measure or rank a processing units relevance to the function of the ANN. Early works, such as [Sejnowski and Rosenberg, 1987], demonstrated methods where this could done by clustering nodes and ranking the relevance of each cluster; or the work [Mozer and Smolensky, 1989], introducing a method for computing each neuron and inputs saliency. This shows that first two questions ask can be answered *yes*. Some of the problems faced when attempting to determine the saliency/relevance of a node are shared by models of attentions, specifically in the area of selective routing [Rothenstein and Tsotsos, 2011, Tsotsos, 1990, Tsotsos et al., 1995].

3. HOW CAN YOU DETERMINE A NODES SALIENCY?

There are potentially numerous was to define and measure the saliency of a node. The method discussed here is intended to allow computing of a nodes saliency based on the networks current solution to the task. This is intended to allow for a quick computation of an approximate saliency of each node in a single (reverse) pass of the network. This is the model currently being studied and has yet to be verified statistically.

3.1 Relevance Assessment

This model used to generate the saliency of each node is inspired by Mozar's work on skeletonisation of a neural network [Mozer and Smolensky, 1989], itself inspired by the method of propagating error back through a network in supervised back-propagation learning rule for training of ANNs. It generates a saliency measure for each non-output node in the network ranging from 0 to 1 based on the effect the node has on subsequent processing units and their saliency.

Initially the saliency of all the output processing units is

set to an arbitrary value, depending on what you intend to measure. To find the saliency of the nodes for a particular task, set the outputs that are responsible for that result high and others low. For an overall view of the nodes saliency to all possible responses, set all the output nodes saliency high. By sequentially computing the weighted average (using the magnitude of the weight connecting the two nodes) off all the nodes output to it by the nodes in the subsequent layer, you can generate an approximate measure of the strength of the effect that the node has on the output of the ANN.

4. REFERENCES

[Mozer and Smolensky, 1989] Mozer, M. C. and Smolensky, P. (1989). Skeletonization: a technique for trimming the fat from a network via relevance assessment. In Touretzky, D. S., editor, *Advances in neural information processing systems*, pages 107–115. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Rothenstein and Tsotsos, 2011] Rothenstein, J. K. and Tsotsos, A. (2011). Computational models of visual attention. *Scholarpedia*, 6(1):6201.

[Sejnowski and Rosenberg, 1987] Sejnowski, T. J. and Rosenberg, C. R. (1987). Parallel networks that learn to pronounce english text.

[Tsotsos, 1990] Tsotsos, J. K. (1990). A complexity level analysis of vision. *Behavioral and Brain Sciences*, 13(3):p423–455.

[Tsotsos et al., 1995] Tsotsos, J. K., Culhane, S. M., Kei Wai, W. Y., Lai, Y., Davis, N., and Nuflo, F. (1995). Modeling visual attention via selective tuning. *Artificial intelligence*, 78(1):507–545.

Multi-Label Feature Selection Methods for the Classification of Microarray Data

Suwimol Jungjit
School of Computing
University of Kent, Canterbury, CT2 7NF, UK
sj290@kent.ac.uk

Alex A. Freitas
School of Computing
University of Kent, Canterbury, CT2 7NF, UK
A.A.Freitas@kent.ac.uk

M. Michaelis
School of Biosciences
University of Kent, Canterbury, CT2 7NF, UK
M.Michaelis@kent.ac.uk

J. Cinatl
Institut fuer Medizinische Virologie, Klinikum der
Goethe-Universitaet, Paul Ehrlich-Str. 40
60596 Frankfurt am Main, Germany
cinatl@em.uni-frankfurt.de

ABSTRACT

This short paper presents an overview of three different multi-label feature selection methods. The first one is the multi-label correlation based feature selection method which is adapted for multi-label classification problems. The method is used to select features (genes) in a neuroblastoma microarray dataset. We also propose other two methods: (1) the multi-label correlation based feature selection method using biological knowledge and (2) Genetic Algorithms (GA) and biological knowledge for Multi-label feature selection. Finally, designing of the last two approaches and future works will be discussed.

Keywords

Feature Selection, Microarray Data, Multi-label Classification

1. INTRODUCTION

Feature selection is one of the most important research topics in the analysis of gene expression microarray data, where the system has to select the most relevant features (genes) to classify each instance into predefined classes. The vast majority of feature selection studies have addressed just conventional single-label classification problems, where each instance (a cell line, in this work) is associated with only one class label. However, in some cases, like in this work, the classification problem is inherently multi-label, i.e. each instance can be associated with a set of class labels [1]. In this context, we propose:

- The Multi-Label Correlation Based Feature Selection Method (ML-CFS)
- The Multi-Label Correlation Based Feature Selection Method using Biological knowledge
- The Genetic Algorithm for Multi-Label Correlation Based Feature Selection Method (GA for ML-CFS)

2. THE TARGET BIOLOGICAL PROBLEM

Microarray technology was developed for measuring the gene expression levels of tens of thousands of gene simultaneously. Hence, the main challenge for classification algorithms is the dimensionality of the data (the number of genes, or features), which is very large, while the number of instances is small [2], [3]. In essence, our two microarray datasets were established from neuroblastoma patients. Our datasets consist of 22,060 and 22058 features (genes), and 24 instances (cell lines). There are two or three class attributes for each dataset, those attributes stand for drugs which are used to treat neuroblastoma, each drug can take two class labels (sensitive and resistant for each cell line).

3. SINGLE-LABEL CORRELATION BASED FEATURE SELECTION

We now review the basic idea of the Correlation-based Feature Selection (CFS) method, which is proposed by Hall [4] CFS is a well-known filter method for single-label classification. That work claimed this method is simple and fast to execute and suitable for both nominal class and continuous class problems. Moreover, Hall stated that a good feature subset should have two main properties: (1) the correlation between a feature and other features in that subset should be low, and (2) the correlation between a feature and the class attribute should be high. Hence, the merit of a feature subset is evaluated by Equation (1).

$$Merit = \frac{k\,\overline{r_{FL}}}{\sqrt{k+k(k-1)\overline{r_{FF}}}} \qquad (1)$$

Where $\overline{r_{FL}}$ is the average feature-label correlation over all feature-label pairs for all features in the current feature subset, $\overline{r_{FF}}$ is an average feature-feature correlation over all pairs of features in the current feature subset, and where k is the number of features in the current feature subset.

4. THE FIRST PROPOSED MULTI-LABEL FEATURE SELECTION METHOD

The difference between the conventional single-label CFS approach and our multi-label approach is in the way that the term $\overline{r_{FL}}$ is estimated. The basic idea is that we use equations (2) and (3) to calculate the average feature-label correlation using the arithmetic mean (i.e., the average value of the correlation coefficient between each feature in a candidate subset *F* and each label in the set of all class labels). In order to implement the proposed multi-label correlation-based feature selection method, we currently use a hill climbing search strategy, using equation (1) to evaluate each candidate feature subset, where the term $\overline{r_{FL}}$

in equation (1) is computed by the proposed equations (2) and (3).

$$r_{f\bar{L}} = \frac{\sum_{i=1}^{|L|} r_{fL_i}}{|L|} \quad (2)$$

$$\overline{r_{FL}} = \frac{\sum_{f=1}^{|F|} r_{f\bar{L}}}{|F|} \quad (3)$$

Where f = a feature, F = a feature subset and L_i = *i*th class label, *i*=1..|L|

5. COMPUTATIONAL RESULTS

To evaluate the effectiveness of our proposed multi-label CFS method, we compare the Hamming Loss values obtained by ML-KNN [5] in each of the three feature selection approaches: (1) our proposed multi-label CFS method (2) Intersection of the two gene sets selected by single-label CFS and (3) Union of the two gene sets selected by single-label CFS. Note that, the Hamming Loss is a well-known measure of predictive accuracy for multi-label classification problems.

Table 1. Hamming Loss obtained by ML-KNN – with standard error between brackets – and number of selected genes for each feature selection approach

Approach used to select genes used by ML-KNN	Hamming Loss	Number of selected genes
Genes selected by multi-label CFS, considering both drug types together	0.278 (0.061)	3
Intersection of the two gene sets selected by single-label CFS considering each drug type separately	0.431 (0.038)	4.3
Union of the two gene sets selected by single-label CFS considering each drug type separately	0.500 (0.032)	1,357.6

6. OTHER PROPOSED MULTI-LABEL FEATURE SELECTION METHODS

6.1 Multi-Label CFS using Biological Knowledge

The idea behinds this approach is using some biological knowledge about cancer-related pathways for improving the predictive performance of the multi-label correlation based feature selection method. We employ knowledge about KEGG pathways, which is a well known pathway database, and use it as part of the evaluation function that evaluates the feature subset. (In this approach, we use hill climbing search as a search strategy). There are two types of relationship between genes in feature subset and cancer-related pathways: (1) the average relative frequency of pathways per gene and (2) the average pathway coverage associated with the subset of features selected. The fitness function for the second approach show below.

$$Fitness\,(FSS_i) = \alpha * Merit_{FSS_i} + \beta * AvgRFP_{FSS_i} + \gamma * AvgPC_{FSS_i} \quad (3)$$

$$AvgRFP_{FSS_i} = \frac{\sum_{f=1}^{k} RFP_f}{k} \quad (4)$$

$$AvgPC_{FSS_i} = \frac{\sum_{p=1}^{m} PC_p}{m} \quad (5)$$

Where $AvgRFP_{FSS_i}$ is the average relative frequency of pathways per gene, $AvgPC_{FSS_i}$ is the average pathway coverage associated with the subset of features, k is the total number of selected features (genes) in the current feature subset and m is the number of user-specified cancer-related pathways.

6.2 The Genetic Algorithm for Multi-Label CFS (GA for ML-CFS)

For the third proposed method, we use Genetic Algorithms as a search strategy and incorporate biological knowledge in fitness function (merit function is same as in the second approach). Each individual (candidate solution) in the population is a candidate feature (gene) subset. Individuals iteratively evolve towards better and better individuals, guided by the fitness (evaluation) function.

7. CONCLUSIONS

Our first method (Multi-Label Correlation-Based Feature Selection with Hill-Climbing search) obtained a smaller Hamming Loss than using the conventional single-label CFS method and obtained the smallest number of selected genes. Moreover, some selected genes are known to be involved in cellular processes that may be relevant in the context of drug resistance.

As ongoing research**,** we have been implementing the use of biological knowledge in the merit function and started experiments with the GA for multi-label feature selection.

8. REFERENCES

[1] Tsoumakas, G., Katakis, I., Vlahavas, I. 2010. Mining Multi-label data. In *Data Mining and Knowledge Discovery Handbook*, Maimon, O., Rokach, L. Ed., Springer, Heidelberg, 667-685.

[2] George, G. V. S. and Raj, V. C., 2011. Review on Feature Selection Techniques and the Impact of SVM for Cancer Classification Using Gene Expression Profile. *Computer Science & Engineering Survey,* 2, 3 (Aug. 2011), 16-26.

[3] Wang, Y., Tetko, I. V., Hall, M. A., Frank, E. and Facius, A.. 2005. Gene selection from microarray data for cancer classification—a machine learning approach. *Computational Biology and Chemistry,* 29, 1, 37-46.

[4] Hall, M. A. 2000. Correlation-based Feature Selection for Discrete and Numeric Class machine Learning. *In Proceedings of the 17th International Conference on Machine Learning (ICML-2000)*, Morkan Kaufmann, San Francisco, 359-366.

[5] Zhang, M. L., Zhou, Z. H. 2007. ML-KNN: a lazy learning approach to multi-label learning. *Pattern Recognition*, 40,7, 2038-2048.