Journal of Artificial Intelligence and Soft Computing Research

Editor-in-Chief
Leszek Rutkowski
Częstochowa University of Technology
e-mail: Leszek.Rutkowski@iiisi.pcz.pl

Honorary Editor
L.A. Zadeh
Computer Science Division, Department of E.E.C.S., University of California Berkeley, CA 94729, USA

Co-Editors
Marcin Korytkowski
Rafał Scherer
Częstochowa University of Technology
e-mail: Marcin.Korytkowski@iiisi.pcz.pl
e-mail: Rafał.Scherer@iiisi.pcz.pl

Technical Editor
Agnieszka Siwocha
University of Social Sciences in Lodz, Poland

Editorial Board

Bernard De Baets
Gent University, Belgium

James Bezdek
University of West Florida, USA

Rudolf Kruse
Otto von Guericke University of Magdeburg, Germany

Andrzej Cader
University of Social Sciences in Lodz, Poland

Adam Krzyżak
Concordia University, Montreal Canada

Włodzisław Duch
Nicolaus Copernicus University, Poland

Miroslaw Pawlak
University of Manitoba, Canada

Er Meng Joo
Nanyang Technological University, Singapore

Witold Pedrycz
University of Alberta, Canada

Marios Polycarpou
University of Cyprus, Nicosia, Cyprus

Francisco Herrera
University of Granada, Spain

Ryszard Tadeusiewicz
AGH University of Science and Technology Krakow, Poland

Hisao Ishibuchi
Osaka Prefecture University, Osaka, Japan

Jun Wang
The Chinese University of Hong Kong

Janusz Kacprzyk
Systems Research Institute, Poland

Lipo Wang
Nanyang Technological University, Singapore

James M. Keller
University of Missouri-Columbia, USA

Ronald R. Yager
Iona College, New Rochelle, NY, USA

Laszlo T. Koczy
Technical University of Budapest, Hungary

Gary Yen
Oklahoma State University, USA

Józef Korbicz
University of Zielona Góra, Poland

Jacek M. Żurada
University of Louisville, USA
JOURNAL OF ARTIFICIAL INTELLIGENCE AND SOFT COMPUTING RESEARCH (JAISCR) is a quarterly peer-reviewed scientific journal published jointly by the Polish Neural Network Society and the University of Social Sciences in Lodz, Poland.

EDITORIAL OFFICE:
Polish Neural Network Society
Armii Krajowej 36
42-200 Częstochowa
tel: +48 34 3250546
fax: +48 34 3250546
e-mail: jaiscr@iisi.pcz.pl

PUBLISHING OFFICE:
University of Social Sciences (SAN)
Information Technology Institute (ITI)
Sienkiewicza 9
90-113 Lodz
tel.: +48 42 6646654
fax.: +48 42 6366251

Copyright © 2016 University of Social Sciences (SAN), Lodz, Poland. All rights reserved.

AIMS AND SCOPE:
Journal of Artificial Intelligence and Soft Computing Research is a refereed international journal whose focus is on the latest scientific results and methods constituting both traditional artificial intelligence methods and soft computing techniques. Our goal is to bring together scientists representing both approaches and various research communities. The areas of interest include, but are not limited to:

AI in Modelling and Simulation
AI in Scheduling and Optimization
Automated Reasoning and Inference
Bioinformatics
Cognitive Aspects of AI
Computer Vision and Speech Understanding
Data Mining
Distributed Intelligent Processing
Evolutionary Design
Expert Systems
Fuzzy Modelling and Control
Hardware Implementations
Heuristic Search
Hybrid Models
Information Retrieval
Intelligent Database Systems
Knowledge Engineering
Mechatronics
Multi-agent Systems
Natural Language Processing
Neural Network Theory and Architectures
Neuro-informatics and Bio-inspired Models
Pattern Recognition
Robotics and Related Fields
Rough Sets Theory: Foundations and Applications
Supervised and Unsupervised Learning
Swarm Intelligence and Systems
Web Intelligence Applications & Search
Various Applications
Contents

Hui Wei
A BIO-INSPIRED INTEGRATION METHOD
FOR OBJECT SEMANTIC REPRESENTATION .................................................. 137

Zheng Yin, Conall O’Sullivan, and Anthony Brabazon
AN ANALYSIS OF THE PERFORMANCE
OF GENETIC PROGRAMMING FOR
REALISED VOLATILITY FORECASTING .................................................... 155

Vladimir Stanovov, Olga Semenkina, and Eugene Semenkin
SELF-CONFIGURING HYBRID EVOLUTIONARY
ALGORITHM FOR FUZZY IMBALANCED CLASSIFICATION
WITH ADAPTIVE INSTANCE SELECTION ................................................... 173

Shogo Nonaka, Takeshi Tsujimura, and Kiyotaka Izumi
GAIN DESIGN OF QUASI-CONTINUOUS
EXPONENTIAL STABILIZING CONTROLLER
FOR A NONHOLONOMIC MOBILE ROBOT ............................................. 189

Cheng-Hong Yang, Sin-Hua Moi, Yu-Da Lin, and Li-Yeh Chuang
GENETIC ALGORITHM COMBINED
WITH A LOCAL SEARCH METHOD
FOR IDENTIFYING SUSCEPTIBILITY GENES .......................................... 203
A BIO-INSPIRED INTEGRATION METHOD FOR OBJECT SEMANTIC REPRESENTATION

Hui Wei

Laboratory of Cognitive Modeling and Algorithms, Department of Computer Science, Fudan University, Handan Road No.220, 200433 Shanghai, China

Abstract

We have two motivations. Firstly, semantic gap is a tough problem puzzling almost all sub-fields of Artificial Intelligence. We think semantic gap is the conflict between the abstractness of high-level symbolic definition and the details, diversities of low-level stimulus. Secondly, in object recognition, a pre-defined prototype of object is crucial and indispensable for bi-directional perception processing. On the one hand this prototype was learned from perceptional experience, and on the other hand it should be able to guide future downward processing. Human can do this very well, so physiological mechanism is simulated here. We utilize a mechanism of classical and non-classical receptive field (nCRF) to design a hierarchical model and form a multi-layer prototype of an object. This also is a realistic definition of concept, and a representation of denoting semantic. We regard this model as the most fundamental infrastructure that can ground semantics. Here a AND-OR tree is constructed to record prototypes of a concept, in which either raw data at low-level or symbol at high-level is feasible, and explicit production rules are also available. For the sake of pixel processing, knowledge should be represented in a data form; for the sake of scene reasoning, knowledge should be represented in a symbolic form. The physiological mechanism happens to be the bridge that can join them together seamlessly. This provides a possibility for finding a solution to semantic gap problem, and prevents discontinuity in low-order structures.

Keywords: bio-inspired method, object representation, prototype

1 Introduction

1.1 How to ground visual concept semantic?

Figure 1 shows an example of problem solving carried out by kindergarten children. The children were asked, according to the given picture, to match a T-shirt with trousers. To accomplish this task, the children needed to have learned some basic concepts or knowledge about clothes. If this problem is presented to a computer in the form of a picture, to obtain an answer requires image understanding and knowledge-based reasoning. Realizing this in an artificial intelligence system, however, is quite difficult. It is not difficult for us to define some concepts and rules in a knowledge base, such as in a logical expression we define:

Figure 1. The problem solving task here is to match each T-shirt with its appropriate trousers.
\((\exists x)(\exists y) \ Is\_Tshirt(x) \land Is\_trousers(y) \\
\land Same\_size(x, y) \land Same\_color(x, y) \\
\land Same\_pattern(x, y) \rightarrow Match(x, y)\).

But how can these concepts and rules correlate with their occurrences in a picture? If this connection can’t be built, then the computer can’t know what is \(x\) and where is \(x\). So-called high level knowledge defined like this does not settle the problem of operability, i.e. how to apply these concepts on a picture is still unclear. Consequently, these symbolic rules will not be triggered successfully. Almost all applications of computer vision or image understanding concerned with knowledge are seriously suffering from semantic gap problem. In artificial intelligence there is a very famous and conventional hypothesis, which is the discontinuity in low-order structure. In the past history of AI, this hypothesis was a kind of compromise between high computational demand and simple computer. We think this is the reason of leading semantic gap. The reason man does not suffer from semantic problem is because human brain has rich representation layers and rich process layers. We need to find a solution that can connect abstract symbolic concept with pixel-level manipulations, on a condition of this solution submitting to formal paradigm.

In many applications on image retrieval, in advance, all training pictures were manually labeled what there are in a picture [1, 2]. And these symbols are regarded as the semantic of picture. This practice is very popular. For example, a word “car” was tagged to a picture, but this label did not define any detail about this concept. So this method is too simple and too coarse to solve semantic problem. Because (a) the words and syntax used for labeling is highly task-dependend, and many semantic details are neglected; (b) these labels can neither be used as standards to include any other positive case, nor exclude any other negative case; (c) using these labels cannot differentiate an object from its environment.

This solution should have some kind of connectionist infrastructure.

1.2 Two aspects of this solution

Now there are two common senses in computer vision, saying that (a) an animal’s vision system is much cleverer than a machine one, and (b) object recognition needs a help from higher level knowledge. To (a), the advantage of a biological vision system’s that it has a systematic architecture with rich knowledge and rich levels of representing and processing. And some similar strategy has been applied to represent image semantic [3]. To (b), what is knowledge needs to be clarified on the base of a biological representation paradigm. These two aspects outline the main parts of our solution.

1.3 How to fulfill a more detailed definition of an object?

Many concept-representation concerned studies have been done. Some of them were facing image retrieval, thus the bag-of-words algorithm were applied[4; 5]. Another more complicated method is a structural decomposition model. In this model the shape of an object is described in terms of relatively a few generic components which are joined by spatial relationships [6, 7, 8, 9, 10, 11, 12, 13]. Region has been proved to be a kind of effective element to describe image semantic [14]. For example, due to region can provide much more extensive information than pixel, region-based representation can facilitate some advanced processing such as segmentation [15, 16] and tampering detection [17]. Similar to region, patch-based descriptors were also popular in semantic definition [18, 19]. And patch can also represent rich information in an expanded area, so using it can also implement some semantic-concerned task, such as image inpainting [20] and image synthesis [21]. For visual concept application, the representation form [22], topological relations among regions [23], template of an object [24] is very important. All above works were a good start on concept acquisition, explicit representation and top-down effect of visual concept. We also want to contribute on these themes.

We think a prototype representation of an object is one of a crucial types of knowledge for object recognition, especially when neither object nor its background is highly specific or severely restricted. That famous example, recognizing a spotted dog from an environment of swing tree shadow, is a typ-
ical instance of needing prototype. Theory of prototype can be found in the chapters of perception and concept acquisition of Cognitive Psychology [8, 25]. A physiological neuroscience study [26] shows that semantic is processed in the left inferior prefrontal cortex. This hints that a procedure of integration is very necessary, because only higher cortex can collect and process information from extensive area. Here the form of prototype or the materialization of this kind of knowledge is important for the computer vision (CV), because on the one hand it was the result of perceptual experience, and on the other hand it should be able to guide the future practice of downward processing. We can conclude that the prototype of an object is the final destination of learning and also is the source of bi-directional visual processing. So, the formalization schema of prototype representation is the key point. But how to realize this prototype definition is always worth studying. For the sake of defining the prototype of a type of object, two things are necessary. The first is representing the parts of an object, and the second is describing their topographical relationships. This schema should be compatible with upward raw-data input as well as downward pixel-organizing instruction. This has always been the basic goal of CV, but its priority has always been delayed.

The psychological experiments on mental image proof that the prototype of an object resembles the original, and experiments on memory proof that the prototype is somewhat abstract and declarable. From a perspective of AI, these are two conflict requirements: the former is pixel-suited and the latter is symbol-suited. Who can satisfy them simultaneously? The answer is neural vision system, a highly optimal system after a long time of natural selection. The biological vision is made up of many processing loops. The middle layers, from ganglion cell (GC) to V1, are the intersection of bottom-up data and top-down concept. We think these middle layers are very important for producing semantic and grounding semantic, or here are the key locations of semantic emerging. This paper focuses on a kind of non-classical receptive field (nCRF) mechanism, and uses it to form a prototype representation. Perhaps that is the basic tool for brain to bridge low-level stimulus and high-level semantics.

The second section of this paper is about neural mechanism of nCRF. The third section is a bio-inspired model designing as an infrastructure of forming representation. The forth section is about a nCRF-based Delaunary triangulation strategy to obtain many small normal triangles. The fifth section is about how to combine these candidates to some expanded polygons and similarity comparison between two polygons. The sixth section describes an explicit concept representation through a tree structure and also discusses how product rules can reflect direct connections from high-level knowledge to low-level pixel manipulations. The last two sections are about how this bio-inspired infrastructure might ground semantic.

2 Non-classical receptive field mechanism

2.1 Bio-inspired design is a good option

Why brain can protect against the problem of semantic gap should be concerned with its physiological structure. The biological system is worth simulating because it had been evolved for hundreds of thousands of years. We believe that its structure and function had been tested thoroughly and should have been highly optimized. So a bio-inspired principle should be much more rational for algorithm design. Now let’s see a fundamental mechanism.

2.2 The neural mechanism of nCRF

Ganglion cells (GC) are the most important cells in retina. They locate at the rear path of information transmitting in retina. Since 1960s, many researchers found there was a large region outside the classical receptive field (CRF). In this region, light spot stimuli cannot directly cause a reaction of the cells, while they can modulate the reaction caused by the CRF. And this modulation can be facilitory, inhibitory or disinhibitory [27, 28], and this expanded receptive field is called as non-classical receptive field (nCRF). Neurophysiologic researches [29, 30] show a very complex formation of nCRF constructed by receptor cells (RC), horizontal cells (HC), bipolar cells (BC) and amacrine cells (AC), and also by outer and inner plexiform layer. Activities in the region can inhibit the antagonistic effect and compensate the loss of low spatial frequency
caused by the CRF center-periphery antagonism to some extent. nCRF plays an important role in representing contour [31], shape [32], curvature [33, 34]. nCRF can compensate loss of low spatial frequency to some extent by adding the output from extra surround area of CRF. Through its nCRF, a GC expands its information-receiving scope several times as CRF; undoubtedly this neural basis makes GC able to integrate image features in a large scale. Moreover, we think it plays a significant role in separating figures out of background.

2.3 nCRF can self-adapt its size so as to optimize its representational role

From the point of view of an image processing and understanding, GC and its nCRF mechanism are of great significance in a feature detection, and every GC plays a role of feature descriptor. What is surprising is that each GC can adjust its size of nCRF dynamically in order to make the characteristics occurring in its nCRF monotonous. So, a GC can reduce its size of nCRF to represent fine detail occurring in a local area, and also can expand its size to represent a big block with unitary feature. The GC and its nCRF are self-adaptable, localized, with regular shape, autonomous, and parallel. These attributes make it to be an ideal candidate of general descriptor. In traditional image processing, we note that GC was ususally used to extract boundary, filter noise or enhance image. But these are absolutely not the main functions of it, but somewhat wasting its talents.

Figure 2. A whole logic framework of implementing formal representation of prototype

3 A representation schema basing on nCRF mechanism

3.1 Size changeable nCRF can record patch, which can be a sub-component of an object

The research of the neurophysiology has shown, that according to different brightness, color or velocity of stimuli, the size of receptive field can be changed dynamically. This self-adaptability satisfies following two cases. In a dark environment, GCs will enlarge the size of receptive fields by means of reducing the spatial resolution, and accept much light through spatial summation. While distinguishing some fine details, the receptive fields will turn smaller so as to improve the spatial resolution. Each GC can implement this automatically by a local neural circuit. Besides the CRF, there are many rings. We call them sub-regions, and they are made up of nCRF. The maximum size of nCRF is about 3-6 times than the size of CRF. Figure 3(a) is a model of nCRF with multiple sub-regions, and (b) shows several initial RF being covered on an image, and (c) shows they were resized through increasing or decreasing sub-regions according to the stimuli they confronting with.
Moreover, we think it plays a significant role in segmenting figures out of background. GC is able to integrate image features in a large scale. It expands its information-receiving scope several times as CRF; undoubtedly this neural basis makes it feasible to represent a big block with unitary feature. The fore-mentioned multiple sub-regions are for the sake of specialized representation. For simulating this schema, we construct a matrix of computational units and let them change their sizes dynamically to fit the situation they confront. If a unit happens to cover a fine detail, then it will shrink its RF so as to record the detail more accurately. If it happens to cover a piece of homogeneous texture, then it will expand its RF to represent a unitary block. The fore-mentioned multiple sub-regions are for the sake of size changing. If expanding is necessary, then one or more sub-regions will be appended, and if shrinking is necessary, then one or more sub-regions will be deleted. This mechanism guarantees the feasibility of this self-adaptation. Now we can see that the function of GC matrix is to earn a just enough representation.

![A whole logic framework of implementing formal representation of prototype](image)

**Figure 3.** A model of RF and how it changes its size

Then, basing on the structure of neural circuit, we can design a schema to represent image. An image is actually the combination of many fine details and some homogeneous blocks. If we have a matrix of RFs and assign small size RF to record detail and assign big size RF to record block, then we may get an approximation of the original image (Figures 3(b) and (c)). This motivation is very important, because it reveals the basic principle of GC’s working. GC always makes its role of specialized representation. For simulating this schema, we construct a matrix of computational units and let them change their nCRF sizes dynamically to fit the situation they confront. If a unit happens to cover a fine detail, then it will shrink its RF so as to record the detail more accurately. If it happens to cover a piece of homogeneous texture, then it will expand its RF to represent a unitary block. The fore-mentioned multiple sub-regions are for the sake of size changing. If expanding is necessary, then one or more sub-regions will be appended, and if shrinking is necessary, then one or more sub-regions will be deleted. This mechanism guarantees the feasibility of this self-adaptation. Now we can see that the function of GC matrix is to earn a just enough representation.

![A representation schema basing on nCRF mechanism](image)

**Figure 4.** A multi-layer neural architecture and how a GC adjusts its RF dynamically.

(a) A multiple layers architecture. For clearness only several GCs and their RF are drawn. A RF has three parts: positive center, negative surround, and positive extra-surround. (b) This is a neural circuit of RF adjusting dynamically, which is a small functional unit of (a). Dynamic adjustment of RF due to neuron can change its destination of projecting output according to changing stimulus. This can be realized by three relay neurons and three switch neurons. A switch neuron imposes its backward control on three relay neurons, and selectively permits only one relay outputting its signal upwards to GC. This makes a relay neuron may have a chance to join one of three different rings of a RF. In (c)-(d), with the different switch turning on, the same neuron may exclusively participate in forming one of rings of a RF. Then a size-changing RF comes into being. (c) is a big one, (d) is a middle one, and (e) is a small one.

This GC-based image representation is more compact than pixel-based bitmap, because what a RF can represent is usually bigger than a pixel, so it is more efficient. Figure 4 is a hierarchical computational model of a GC and its inferior RF. At the highest level, a GCs array will turn a pixel-bitmap into a block-group. The dimension of block-group will decrease greatly than that of pixel-bitmap. A block-grained representation is more meaningful than pixel-wise representation, and must ease the emergence of semantic.
An ideal self-adaptability of GC is it adjusting its RF to a proper size coinciding with the scale of main component of stimuli occurring in its RF. This is done by a sequence of operations that append or withdraw sub-regions to or from one of three parts of RF. This causes that a GC can summary the attributes of stimuli in different area. Figure 5 is an algorithm to realize this. A more detailed implementation of this model can be seen in [35] and [36].

Figure 5. The Dynamic Adjustment of RF

4 Prototype emerging from integration

In image understanding, a visual concept is activated if and only if thousands of pixels are arranged properly. So semantic is the result of integrating pixels in terms of some statistical pattern of distribution. The algorithm in previous section can facilitate the emergence of this kind of pattern, and one or several stable patterns are right to be prototypes that define a class of object.

4.1 RF mechanism archives a higher efficiency of representation

Here we also use salient object as learning sample [37]. The left of Figure 6 is an original picture, and the right is the result of GC-array running on the picture. The red circles denote the final sizes of RF after dynamic adjustment. For that bird, its back and wing possess same color or similar textures, so we represent them only by a dozen of big size RFs instead of many unorganized pixels. And its eye and beak possess tiny details, so we represent them by some small size RFs. So, a highly efficient representation is achieved. A fact that can’t be ignored is that circle has regular shape and well defined algebraic formula, so it is easy for parameterization, and consequently it is easy to form a symbolic representation. And a parameterized representation also does not prevent original image from being rebuilt accurately.

Figure 6. Size-changeable RFs bring an efficient representation

4.2 Set of regular blocks: a compact representation of object

Many existing algorithms in computer vision use the pixel-grid as the underlying representation. The pixel-grid, however, is not a natural representation of visual scenes. A good representation schema would be more natural, and presumably more efficient, to work with perceptually meaningful entities obtained from a low-level grouping process. Superpixels [38] represent a restricted form of region segmentation. Turbopixels [39] represent an image with a lattice-like structure of compact regions by dilating seeds so as to adapt to local image structure. The superpixel algorithm should partition an image into regions that are approximately uniform in size as: shape (compactness), minimizing region under segmentation, provided that superpixel size is comparable to the size of the smallest target region. Turbopixels achieve this by designing a geometric flow that dilates an initial set of uniformly distributed seeds, where each seed corresponds to one of superpixels. So, it can also be considered as a compact image representation, each of them should represent a simply connected set of pixels. Both of them provided two representation methods in order to achieve computational efficiency, represen-
tational efficiency, perceptual meaningfulness and near-completeness [40]. Our algorithm based on nCRF can also accomplish the same goals through using Inscribed polygon and Delaunay Triangulation from nCRF result. The mechanism to generate them is illustrated in Figure 7.

![Image](image.png)

Every red circle denotes a nCRF and green dots denote intersections of circles. Both Inscribed Polygon (the left) and Delaunay Triangulation (the right) are derived from intersection points on neighboring circles.

**Figure 7.** A superpixel-like effect realized by nCRF-based algorithm

In order to testify the performance between our algorithm and superpixels, we run programs of superpixels, Inscribed Polygon and Delaunay Triangulation schemas on two different image databases. One is CityplaceBerkeley image database (481×321), the other is Microsoft Research image database (640×480). One of results is shown in Figure 8. It can be seen that the effects in (c) and (d) are similar to (a), which indicates that a complete coverage by Inscribed Polygons and Delaunay Triangulations can be calculated from RFs. But there still has a very important difference between algorithms through RFs and superpixels. Superpixel has an irregular shape, and still in a form of pixel-set instead of a form of brief vectors. This makes it difficult to be represented, recorded and operated algebraically. Due to the irregularity of shape, either the boundaries or the vertexes of a superpixel are dot-matrix data but not vectors. If the vectorization is required, it must degrade its time and storage consuming further. While the RF has a completely regular shape, circle or triangular, which can be easily represented, recorded and operated by a symbolic or algebraic means.

### 4.3 Run-time comparison

In order to compare the efficiencies of our nCRF-based Delaunary triangulation algorithm and Super-pixel algorithm, we randomly selected 100 pictures with size of 481×321 pixels from Berkley Image Database. The test computer is with Intel Pentium Dual CPU E2200, 2.20GHz, 2G RAM. Figure 9 shows that our speed of producing small patches is much faster than Superpixel algorithm.

![Image](image.png)

**Figure 9.** Speed comparison between nCRF-based algorithm and SuperPixel algorithm

5 5 Representing the parts of an object through polygons

#### 5.1 Combining delaunay triangles into a polygon

In Figure 8(d), Delaunay triangulations provide a good start for further processing. So many small triangles can be regarded as components of an object. We can combine them into some polygons, and use these polygons as modules to construct an object. Because polygons are good at topographical and geometrical invariance, so using them to represent object offers the most stability in defining prototype.

During the learning period, we can provide some typical samples, such as clean cows without background disturbance, to computer, and ask it to form a prototype for this concept. Once cow images were uploaded to aforementioned nCRF-based neural computational system, we can obtain so many small triangles which covered a sample completely. Thus the first step of prototype-learning is to combine them appropriately. What we hope is that those combinations can reflect the structural characteristics possessed by a type of objects. A direct solution of combining triangles is to apply polygon, which
is flexible enough to integrate triangles, and polygon’s representation in analytic geometry is simple and compact.

In order to obtain a larger polygon from the result of nCRI algorithm, we design a clockwise spiral coordinate to order those triangles in 2D space. Then we can expand polygon by appending triangle one-by-one. Figure 10 demonstrates this expanding process and an algorithm of producing a polygon through binding some neighboring triangles. Here the principle of searching triangles is keeping the search closest to the periphery of growing-up polygon.

The nCRI-based mechanism provides a basic infrastructure to enable and facilitate patch-grained manipulations on geometrical level greatly. Searching algorithm perhaps is inefficient in pixel-grained space, but it is feasible in block-grained space.

At the top of Figure 11, there are at least hundreds of small triangles, we want them to be allocated to different enlarged polygons, and at same time we hope these polygons happen to be the structural parts of an object. Tab. 1 is an algorithm to draw boundaries of polygons on the Delaunay triangulations. The down of Figure 11 is one of results of this algorithm. And Figure 12 includes two outputs after executing polygon-production algorithm.

**Figure 8.** The partition results on an Example from Berkeley Image Database. (a) is original picture. (b) is superpixel result. (c) is RFs coverage on the original image and (d) is the result by Delaunay Triangulations.

**Figure 11.** A Delaunay triangulations of a flower has many small triangles needing to be fitted by some polygons.
Hui Wei

Figure 8. The partition results on an Example from Berkeley Image Database. (a) is original picture. (b) is superpixel result. (c) is RFs coverage on the original image and (d) is the result by Delaunay Triangulations. It is flexible enough to integrate triangles, and polygon’s representation in analytic geometry is simple and compact.

In order to obtain a larger polygon from the result of nCRF algorithm, we design a clockwise spiral coordinate to order those triangles in 2D space. Then we can expand polygon by appending triangle one-by-one. Figure 10 demonstrates this expanding process and an algorithm of producing a polygon through binding some neighboring triangles. Here the principle of searching triangles is keeping the search closest to the periphery of growing-up polygon.

The nCRF-based mechanism provides a basic infrastructure to enable and facilitate patch-grained manipulations on geometrical level greatly. Searching algorithm perhaps is inefficient in pixel-grained space, but it is feasible in block-grained space. At the top of Figure 11, there are at least hundreds of small triangles, we want them to be allocated to different enlarged polygons, and at same time we hope these polygons happen to be the structural parts of an object. Tab. 1 is an algorithm to draw boundaries of polygons on the Delaunay triangulations. The down of Figure 11 is one of results of this algorithm. And Figure 12 includes two outputs after executing polygon-production algorithm.

Figure 10. Speed comparison between nCRF-based algorithm and SuperPixel algorithm

Table 1. Algorithm for producing multiple polygons on Delaunay triangulations

| Input data: int Small_Tri[id][x1][y1][x2][y2][x3][y3]; //All Delaunay triangles with their vertexes coordinates. |
| Output data: List Polygon[id]; // A list dimension recording all polygons and their children triangles. |
| Temp data: Boolean Neighboring[id][id]; // A matrix of all neighborhoods between triangles. |
| 1. If the array of Small_Tri[id][x1][y1][x2][y2][x3][y3] is EMPTY Then End; |
| 2. Clustering all vertexes by K-means algorithm, store all class center into center[K]; |
| 3. index=0; |
| 4. For each center[k] do |
| Finding Small_Tri[k][x1][y1][x2][y2][x3][y3] whose center of gravity is closest to center[k]; |
| Setting Small_Tri[k][x1][y1][x2][y2][x3][y3] as the seed of growing Polygon[index]; |
| Calling “Algorithm of forming a polygon along a clockwise spiral direction”; // Here Neighboring[id][id] are needed. |
| According to the order of triangles being expanded, storing them into a list named Polygon[index]; index=index+1; |
| 5. For each Polygon[i] |
| For each element in Polygon[i] |
| Marking Small_Tri[element][x1][y1][x2][y2][x3][y3] by BEEN-DELETED; |
| 6. For each Small_Tri[id][x1][y1][x2][y2][x3][y3] |
| If all three neighboring triangles of Small_Tri[id][x1][y1][x2][y2][x3][y3] were marked by BEEN-DELETED Then Deleting all three vertexes of Small_Tri[id][x1][y1][x2][y2][x3][y3]; |
| 7. Iterate since 1. |

Once this algorithm ends, some enlarged polygons come into being. The triangle is the simplest shape with edges, and this causes that searching from one triangle to its neighbors has only two possible choices. And while K is limited, K-means clustering algorithm can guarantee the seeds will not diverge far away from the local centers of topological components of an object. And what we focus on polygon, this further reduces the occasional disturbance of object’s texture.

5.2 Matching between polygons

The core of inductive learning of prototype is to find the common components from different instances of a type of object. This is a hard work to do on pixel-level, or on small triangle-level. But it can be done easily on polygon-level. Once we divide an object into several larger polygons, then we can use famous shape context algorithm [41] to decide which two polygons corresponds well. Figure 13 shows two polygon-groups produced by previous algorithms, and they represent two cows respectively. Between two contours, a dozen of corresponding points can be established by a shape context algorithm. Usually, deciding initial points to
match is a problem to shape context algorithm, because some points not belonging to contour might been selected with high possibility. This will reduce the accuracy of matching. But in a current situation, all polygons are represented algebraically, because they are defined by vertexes and all edges are vectors. Therefore on these edges it is easy to choose initial points. This algorithm is insensitive to position, size and pose, so we can concentrate on shape similarity.

Another strategy that helps us to establish correspondences between polygons is minimum spanning tree algorithm. Firstly, all polygons were named. Secondly, according to their neighboring relationships, a connected graph was formed for each set of polygons in an image. Thirdly, selecting a node as tree root and starting Prim minimum spanning tree algorithm, after that we got a spanning tree growing up from a selected polygon. Figure 14 is the result of this process. We show there a two cow pictures which were made up by many named polygons. We selected three pairs of spanning trees taking root in A15-C15, A8-C7 and A23-C20 respectively. It is obvious that every pairs are very similar, because they really reflect the topological structure of an object. Therefore, applying this strategy we can greatly improve the matching accuracy between different cases.

After this step, we find out which polygons are corresponding in different samples, and they perhaps are the similar parts of a kind of object. Once correspondence relationships among parts or components are established, then an inductive learning procedure can be used to discover those inherent and persistent relationships. Thus, a prototype or some kind of formal semantic description of this type of an object can be defined by these relationships.

6 Representing a prototype by root-tree

6.1 A multi-layer concept-defining tree

In the previous Section, an image representation schema was developed. Basing on it, we can use the combination of multiple polygons to represent an object, and Figure 15 is an example for representing COW. Firstly, several cases of cow, with different appearances, had been watched, and each of them is an instance of the concept “cow”. Secondly, nCRF-based algorithm was applied, and each instance was partitioned by polygons, and topological correlations of these polygons are important. Thirdly, a AND-OR tree was built to record combinational relations at polygon level and at component level. In Figure 15 from root to leaves, they respectively describe: a cow might have multiple instances, and every instance can be divided into several components, and every component includes
Figure 12. Some polygons have been produced on the base of many small triangles. The match is a problem to shape context algorithm, because some points not belonging to contour might be selected with high possibility. This will reduce the accuracy of matching. But in a current situation, all polygons are represented algebraically, because they are defined by vertexes and all edges are vectors. Therefore, on these edges it is easy to choose initial points. This algorithm is insensitive to position, size, and pose, so we can concentrate on shape similarity.

Another strategy that helps us to establish correspondences between polygons is minimum spanning tree algorithm. Firstly, all polygons were named. Secondly, according to their neighboring relationships, a connected graph was formed for each set of polygons in an image. Thirdly, selecting a node as tree root and starting Prim minimum spanning tree algorithm, after that we got a spanning tree growing up from a selected polygon. Figure 14 is the result of this process. We show there a two cow pictures which were made up by many named polygons. We selected three pairs of spanning trees taking root in A15-C15, A8-C7 and A23-C20 respectively. It is obvious that every pairs are very similar, because they really reflect the topological structure of an object. Therefore, applying this strategy we can greatly improve the matching accuracy between different cases.

After this step, we find out which polygons are corresponding in different samples, and they perhaps are the similar parts of a kind of object. Once correspondence relationships among parts or components are established, then an inductive learning procedure can be used to discover those inherent and persistent relationships. Thus, a prototype or some kind of formal semantic description of this type of an object can be defined by these relationships.

6 Representing a prototype by root-tree
6.1 A multi-layer concept-defining tree

In the previous Section, an image representation schema was developed. Basing on it, we can use the combination of multiple polygons to represent an object, and Figure 15 is an example for representing COW. Firstly, several cases of cow, with different appearances, had been watched, and each of them is an instance of the concept “cow”. Secondly, nCRF-based algorithm was applied, and each instance was partitioned by polygons, and topological correlations of these polygons are important. Thirdly, a AND-OR tree was built to record combinational relations at polygon level and at component level. In Figure 15 from root to leaves, they respectively describe: a cow might have multiple instances, and every instance can be divided into several components, and every component includes...
(a) Cow-1 and its labeled polygons

(b) Cow-2 and its labeled polygons

A MST of Cow-1 rooted at A15

A MST of Cow-2 rooted at C15
Figure 14. Spanning tree of connected graph can help establishing correspondence between polygons
multiple blocks, and every block is made up of several polygons, and every polygon has its shape, and each shape can be defined by its vertexes, and any vertex can be obtained by several neighboring RFs. Using a tree-like data structure has been proved to be a practicable choice [42] to organize regions. All these calculations can be executed by algebraic equations of circle, and searching strategy is also workable [43].

The data at the bottom of Figure 13 are some vertexes of polygons, and all coordinates can be relocated according to a new datum point. Each row defines a shape, and no shape here is required to be absolutely fixed. Further more, these shapes can be zoomed in or out, or be rotated easily because they are defined by vectors. Using these tables we can rebuild several prototypes of a concept, and when a new sample occurring, we can compare it with those prototypes and identify which class the new sample belongs to. The concept tree is not absolute too, because the occurrences of Cow are different one to another. Fuzzy inference or probabilistic inference, such as Bayesian reasoning [44], or shape context algorithm [48] is a good tool to use this representation.

6.2 Direct manipulation that done to low-level pixels can be derived from this multi-layer tree

Now let’s go back to the rules we formalized in Section 1. We obtain a concept tree about cow, and at the root of this tree it is symbolic, and at the leaf ends it contains many pixel-concerned coordinates. Moving from root to leaves, the concept definition turns gradually from abstract label to detailed object. This provides us an opportunity to join high level symbols and low-level BMP operations (such as searching pixel) together. Now let’s define some production rules to show this practical method again. For explaining what is a cow:

\[
\begin{align*}
&\text{IF } \text{Is Cow}(x) \text{ THEN } \text{Has Leg1}(x) \land \text{Has Leg2}(x) \land \text{Has Leg3}(x) \land \text{Has Leg4}(x) \land \ldots \land \text{Has Body}(x); \\
&\text{IF } \text{Has Leg1}(x) \text{ THEN } \text{Contain Area Like}(x, \text{Block1}); \\
&\text{IF Verify(\text{Block1}) THEN Search Pixel In Set(\text{Block1})};
\end{align*}
\]

For deciding which pixels combine a cow:

\[
\begin{align*}
&\text{IF } \text{RF Distribution Similar}(x, \text{Block5}) \text{ THEN } \text{Is Leg3}(x); \\
&\text{IF Verify(\text{RF Distribution Similar}(y)) THEN } \\
&\text{Decide a Region Defined by(y);} \\
&\text{IF Verify(\text{Similar}(x, y)) THEN } \\
&\text{Search Pixel In \_ according to y}(x, y);
\end{align*}
\]

The predicates like Search Pixel In( ) and Contain Area Like( ) are totally operable at pixel level. Up to now, we show a feasible method, basing on an elaborate multi-level representation, which can ground semantic and apply them in image understanding.

6.3 Why this tree can work?

Now a new problem arose. We know that tree, as a kind of data structure, is not new, and it was used extensively. For example, some knowledge for animals or plants classification is defined by tree, and some searching spaces are also defined by tree. What, then, makes tree work in grounding semantic this time? The most important reason is that GC-array and their RFs provide several gradual or transitional representation layers to ease the span between symbols and pixels. Another reason is that no information, whatever level or abstractness it is, is neglected, and on the contrary a realistic unit is assigned to represent it. That is to say we prevent our model from the famous and conventional hypothesis of discontinuity in low-order structures.

In Cognitive Psychology, cognitive modeling asks three key steps: (1) the stimulus must be translated into an internal representation, (2) the representation is manipulated by cognitive processes to derive new internal representations, and (3) these are in turn retranslated back into action [45]. Our methods of applying multi-layer representation and keeping intermediate links go along the same way, so a dense definition of semantic can be reached.

7 A semantic-grounding neural infrastructure

Perhaps we may suspect how so many rules can work efficiently. An alternative implementation way is doing this by a neural network. We know that production rules, including fuzzy rules and uncertain rules, can be realized equivalently by a neural network.
An important motivation of this paper is to construct an infrastructure for semantic-grounding. We think GC array provide a powerful base for this goal. Figure 16 is a neural structure [46] which can achieve semantic representation. Layer 2 is GC array, and each of them has a nCRF on photoreceptor layer. These two layers had described in fore-sections. And in layer 3, we design some feature recording units to produce semantic. We divided units in layer 2 and 3 into many small groups (three examples were drawn in each layer), and let them match one-by-one. The famous Bidirectional Associative Memory (BAM) algorithm was applied between each pair of groups. The BAM is unconditionally stable, and it is a heteroassociative network and indeed capable of error correction. Then two groups in a pair can feed each other upwards or downwards. The number of stable states in a BAM network is limited, but there are so many pairs, and the combinational number of states of different BAM networks is very huge. If we define units in layer 3 by some symbols and define units in layer 2 by image features, then layer 2 acts as explainer to ground semantic. When some more complicated network is built in layer 3, then a more flexible representation is possible [47]. This network fulfills the decomposition and integration of semantics with a fine span.

8 Conclusion

When eye receiving an image as input, the retina in it will decide which information is significant and needs to be transmitted to central brain. A good heuristic rule is retaining variance and dis-
carding redundancy. This task turns to be very difficult when scene keeps changing, and processing should be done in real time and the result should be in accordance with diverse upcoming tasks. This means a general and no task-specific schema should be sought. In this paper, we simulated a biological mechanism, using many receptive fields, for image representation, and based on it several important image processing tasks, such as segmentation and integration can be improved through some predefined production rules. Once a neuron-bounded representation is formed, the semantic-grounding turns to be practicable.

Semantic is crucial for computer vision (CV) and natural language understanding (NLU). In CV, possessing semantic means a program knows what a pile of pixels is; and in NLU, means a program knows how to apply rich relationships between concepts flexibly. A hierarchical structure can provide dense, continuous representations and rich linkages between them, from high-level concepts to low-level instances. This will greatly benefit knowledge applying in CV and NLU.

**Acknowledgment**

This work was supported by the 973 Program (Project No. 2010CB327900), the NSFC project (Project No. 61375122).

**References**


Hui Wei received the Ph.D. degree at the department of computer science at Beihang University in 1998. From 1998 to 2000, he was a postdoctoral fellow at the Department of Computer Science and the Institute of Artificial Intelligence at Zhejiang University. Since November 2000, he has joined the Department of Computer Science and Technology at Fudan University. His research interests include artificial intelligence and cognitive science.
AN ANALYSIS OF THE PERFORMANCE OF GENETIC PROGRAMMING FOR REALISED VOLATILITY FORECASTING

Zheng Yin, Conall O’Sullivan and Anthony Brabazon

School of Business, University College Dublin, Ireland

Abstract

Traditionally, the volatility of daily returns in financial markets is modeled autoregressively using a time-series of lagged information. These autoregressive models exploit stylised empirical properties of volatility such as strong persistence, mean reversion and asymmetric dependence on lagged returns. While these methods can produce good forecasts, the approach is in essence atheoretical as it provides no insight into the nature of the causal factors and how they affect volatility. Many plausible explanatory variables relating market conditions and volatility have been identified in various studies but despite the volume of research, we lack a clear theoretical framework that links these factors together. This setting of a theory-weak environment suggests a useful role for powerful model induction methodologies such as Genetic Programming (GP). This study forecasts one-day ahead realised volatility (RV) using a GP methodology that incorporates information on market conditions including trading volume, number of transactions, bid-ask spread, average trading duration (waiting time between trades) and implied volatility. The forecasting performance from the evolved GP models is found to be significantly better than those numbers of benchmark forecasting models drawn from the finance literature, namely, the heterogeneous autoregressive (HAR) model, the generalized autoregressive conditional heteroscedasticity (GARCH) model, and a stepwise linear regression model (SR). Given the practical importance of improved forecasting performance for realised volatility this result is of significance for practitioners in financial markets.

Keywords: Realised Volatility, Genetic Programming, High Frequency Data

1 Introduction

Volatility is an important concept in finance and has different implications depending on the perspective of the user. From an investment perspective, volatility is a measure of the degree to which returns tend to fluctuate. Traders would like to capture the volatility caused by positive returns, whereas in contrast, risk management is more concerned about the volatility caused by negative returns. Volatility is a key element in the pricing of derivatives, is a key input in determining portfolio weights in a portfolio optimisation model and is also a key input in the calculation of regulatory capital requirements under the Basel II accords.1 Hence, many stakeholders have an interest in being able to model and predict volatility.

In a conventional volatility model, volatility is a latent variable that is often estimated parametrically from historical daily returns using discrete time GARCH models [1] or continuous time Stochastic Volatility models [2]. The term realised volatility can be broadly defined as the sum of in-

1Basel II are recommendations on banking laws and regulations issued by the Basel Committee on Banking Supervision.
tradeday squared returns, measured at short intervals [3]. Such a non-parametric volatility estimator has been shown to provide an accurate estimate of the latent process that defines volatility [5] and therefore, through realised volatility estimation, the latent volatility process is theoretically observable from historical intraday returns.

Genetic Programming (GP) is a powerful model induction methodology which has been widely applied for symbolic regression [6, 7]. A number of studies have previously applied GP for volatility modeling [9, 11, 12, 13, 15] but there are still some important questions which have not been addressed.

In particular, market conditions have been documented in the finance literature as having a high correlation with volatility in a variety of settings. A sample of these studies include [16] which examined the relationship between trading volume and volatility, [17] which examined the relationship between the number of transactions and volatility, [18] examined the relationship between price range and volatility, [39] examined the relationship between interest rates and volatility, [20] examined the relationship between implied volatility and volatility and [21] which examined the relationship between the bid-ask spread and volatility. This study extends previous works by identifying a range of metrics on market conditions and allowing GP to use these as inputs in modeling volatility. The calculated realised volatility is modeled directly using GP and the one-day-ahead RV is forecasted. Forecasting results from GP are compared with those from benchmark models drawn from the finance literature.

1.1 Structure of Paper

The remainder of this contribution is organised as follows. Section 2 provides some background on volatility modeling and provides the motivation for applying the GP methodology to RV forecasting. Section 3 describes the data used in this study. The forecasting results are provided in Section 4 and finally, conclusions and opportunities for future work are discussed in Section 5.

2 Overview of Volatility Modelling

In this Section we overview three key items. Initially, we provide an introduction to the concept of realised volatility. Then we briefly introduce current state-of-the-art approaches for the forecasting of realised volatility. Finally, we provide the motivation for the GP methodology adopted in this study.

2.1 Realised Volatility

Under the concept of RV, returns are assumed to be generated by the stochastic differential equation (Equation 1), which is a continuous-time stochastic process over a given time period. The time period is divided into \( t_i \) equally-spaced adjacent intervals and the quadratic variation is defined as the limit of the sum of squared returns over these intervals, as the length of the sampling intervals goes to zero, where \( t_i \) and \( t_{i-1} \) are adjacent intervals (Equation 2). This limit is well-defined in the case of the logarithm price process \( p(t) \), which is a semi-martingale. In the general semi-martingale case, assuming some (mild) restrictions on the types of leverage, the quadratic variation is an unbiased estimator of the integrated variance, \( \int_0^T \sigma^2(t)dt \), and the square root of the quadratic variation is called realised volatility.

\[
dp(t) = \sigma(t)dw(t) \tag{1}
\]

\[
\lim_{i \to \infty} \left( \sum_i (p(t_i) - p(t_{i-1}))^2 \right) \tag{2}
\]

Realised volatility can be used to measure the intraday volatility by summing up the intraday squared returns at short intervals, such as five or fifteen-minute intervals [24]. This concept is very important to volatility modeling. It has been pointed out in [23] that the standard volatility models used for forecasting at the daily level cannot readily accommodate the information in intraday data. The models specified directly for intraday data generally fail to capture the longer interdaily volatility movements sufficiently well. In contrast, using RV allows us to model volatility using relatively high frequency data, and also permits capture of stylised facts concerning interday volatility [23, 24].

In an ideal world, the quadratic variation from shorter intervals (as per Equation 2) is always closer to the integrated volatility than the one calculated using longer intervals. However, returns measured
at intervals shorter than five minutes are plagued by spurious serial correlation caused by various market microstructure effects including asynchronous trading, discrete price observations, and the bid-ask bounce [5].

There are different sampling schemes to estimate the realised volatility as reviewed in [22]. In this study, the RV estimation approach in [25] is followed as we use the same futures index data, FTSE 100 prices. It is also noted that the RV estimated using this method [25] successfully captured the stylised long-memory effect inherent in volatility.

2.2 Conventional RV Forecasting Models

It is well documented in the finance literature that realised volatility is a highly persistent process which has a long memory. Conventional methods used in modeling RV include ARFIMA (Autoregressive Fractionally Integrated Moving Average) [23, 26], HAR (Heterogeneous Autoregressive) proposed by [27], the simple AR (Autoregressive) type model [28, 29], and SV (Stochastic Volatility) with volatility treated as observable [29]. Recently there have also been HAR-type extended models including the HAR-GARCH model proposed by [30], and HAR with a jump process as proposed by [31].

A broad series of empirical work [29, 30, 32] has sought to compare the performance of various RV forecasting models.

In [30], ARFIMA, HAR and HAR-GARCH are compared based on tick-by-tick transaction prices from S&P 500 index futures data (1985-2004) with HAR-GARCH producing the best forecasting performance in terms of several metrics including $R^2$, RMSE (Root Mean Squared Error), MAE (Mean Absolute Error) and RMSPE (Root Mean Squared Percentage Error). In [32], AR, ARFIMA and HAR are compared and HAR gives the best result in terms of RMSE, MAE and $R^2$. This conclusion is drawn on a dataset consisting of tick-by-tick series for USDCHF (1989 to 2003), S&P 500 Futures (1990-2007) and 30-year US Treasury Bond Futures (1990-2003). In [29], simple AR, SV and HAR are compared and HAR gives the best forecasting performance in terms of RMSE, MAE and other measures on a dataset of equity market indices of SPX and DJIA (1997-2011) and two exchange rates CADUSD and USDGBP (1998-2011). The ARFIMA has been reported in [30] and [32] to give a similar performance as HAR, however, its estimation procedure is more complex.

2.3 Motivation for Applying GP to RV Modelling

RV transfers intraday return information to an observable volatility, and therefore allows volatility to be modeled directly. While traditional methods of RV modeling rely solely on lagged values of RV (see Section 2.2), it has been documented that trading volume, number of transactions, price range (including the range of open and close, high and low), bid-ask spread and implied volatility have predicative information / explanatory power for volatility.

It has been noted in [33] that different market information is likely to capture distinct subtle aspects of the volatility process, the relative prominence of which may vary over time. Also different market information may suffer to greater or lesser extents from market microstructure biases. A study by [33] indicates that using a combination of the outputs from a series of GARCH models, with different volatility predictors, could reduce the forecast errors in a range of examined stocks.

In prior works, most studies [34, 35, 36, 37, 38, 39, 33, 40] used market information to explain / forecast conditional volatility in a GARCH type framework. The market information was added in the conditional variance equation as an explanatory factor but the underlying model was linear. The nonlinear Granger causality test conducted in [41] shows there is extensive evidence of bidirectional feedback between volume and volatility which such approaches cannot capture.

In summary, while we have some knowledge of the likely set of explanatory variables (based on market conditions) from prior literature, we still lack a clear theoretical framework as to which of these variables are most important and how they should link together to form a quality model for forecasting of RV. This setting of a theory-weak environment suggests a useful role for powerful model induction methodologies such as GP [6, 8].

In this study, GP is used to select from a set of plausible explanatory variables as identified in the finance literature, and then link them to RV by si-
multaneously evolving a suitable functional form. The functional form returned from training is then used to forecast a one-day-ahead RV. The model is re-trained each day using most recent information as no assumption is made in the modeling process that the relative importance of each explanatory variable remains unchanged over time. The performance of the GP models is compared with a HAR model which only uses RV lagged information as inputs, a GARCH model, which models RV and its volatility together and a linear regression model, which uses the same explanatory factors that are used to train the GP model. It should be noted that given the importance of volatility forecasts across a range of investment decisions, even small improvements in forecast accuracy can have significant practical implications.

3 Data and Methodology

3.1 Background

The dataset used in this paper consists of the complete records for all quotes and trades of European-style FTSE 100 index option contracts and FTSE 100 index futures contracts in 2004 from Euronext-Liffe. The London International Financial Futures and Options Exchange (Liffe) was established in 1982 and was taken over by Euronext in January 2002 to form a market called Euronext-Liffe. Since 2000, all trading in financial contracts on Liffe takes place on an electronic limit order book system, called the Liffe Connect platform.

The datasets used in this study are large, ‘ultra high frequency data’ [10], consisting of 75,755,106 records in the case of the index option dataset (41,794,081 records relating to call options and 33,961,025 records relating to put options). The index futures dataset consists of 26,271,084 observations. All of this data is time stamped.

The futures traded price data was used for RV estimation and both the trade and quote information was used to calculate intraday metrics including trading volume, bid-ask information, price range and the number of transactions. FTSE 100 index options data is used for the implied volatility calculation. Interest rate information, specifically, LIBOR rates (overnight, one-week and six-month) for 2004, were collected from Datastream.

The estimated RV is illustrated in Figure 1 (refer to Section 3.1.1 for estimation details for this data). The first six months of the data is used for initial in-sample training with the out-of-sample testing taking place during the final six months (129 trading days) of the year. Each day’s forecast of RV is determined using all data available up to and including the previous day. For the first day’s out-of-sample forecast (commencing on the first day of July), data from January 9th to June 30th is used. For the last day’s forecast (the last day of December), data from January 9th to December 30th is used. The first five trading days in January are excluded as lagged information is required in the modeling process.

![Figure 1: Annualised Daily Realised Volatility](image-url)

### 3.1.1 Realised Volatility Estimation

FTSE 100 index futures traded from 8:00 am to 5:30 pm in 2004 and therefore there are 114 five-minute intraday returns each day in our dataset, which are calculated from the latest prices before each five-minute mark in Equation 3, where \( \ln(p_{t,j}) \) is the log price for the \( j \)th five-minute interval on day \( t \) and \( r_{t,j} \) is the \( j \)th intraday return on day \( t \) with \( j = 1, 2, \ldots, 114 \).

\[
r_{t,j} = \ln(p_{t,j}) - \ln(p_{t,j-1})
\] (3)

Let \( r_{t,j}, 0 \leq j \leq n \), represent a set of \( n+1 \) intraday returns for day \( t \), so that \( j = 0 \) represents the closed-market period from the close on day \( t - 1 \) until the open on day \( t \), \( j = 1 \) represents the first five-minute on day \( t \), \ldots, concluding with \( j = n \) representing the final five-minute period on day \( t \).

The realised volatility is used to measure the intraday return volatility. The realised variance for
trading day $t$, from the close on day $t-1$ to the close on day $t$, is estimated by weighting the intraday squared returns as in Equation 4.

$$\sigma_t^2 = \sum_{j=0}^{n} \omega_j r_t^2$$  \hspace{1cm} (4)

To ensure conditionally unbiased estimates when intraday returns are uncorrelated, so that $E[\sigma_t^2 | \sigma_t^2] = \sigma_t^2$, it is necessary to apply the constraint $\sum_{j=0}^{n} \lambda_j = 1$ [25]. The $\lambda_j$ is the proportion of a trading day’s total return variance that is attributed to period $j$ and is assumed to be the same for all days $t$ in the sample. To satisfy this constraint, $\omega_j$ is estimated as in Equation 5 and $\lambda$ is estimated as in Equation 6.

$$\omega_j = \left\{ \begin{array}{ll} \frac{1}{1-n} \omega_j, & 1 \leq j \leq n \\ 0, & j = 0 \end{array} \right.$$  \hspace{1cm} (5)

$$\lambda_j = \frac{\Sigma r_t^2}{\Sigma \Sigma_{t=0}^{n} r_t^2}$$  \hspace{1cm} (6)

The average annualised volatility from the estimated RV (in Figure 1.) is 10.37 percent, which is very close to the annualised daily return standard deviation, 10.26 percent, therefore, any potential bias caused by autocorrelation among intraday returns is small. The distribution of $\ln(\sigma)$ is almost symmetric and approximately Gaussian. Applying the augmented Dickey-Fuller test indicated that the realised volatility process does not contain a unit root as expected given the high degree of mean reversion in volatility. The low decline in the autocorrelations of the realised volatility series suggests a long memory process, another well documented stylised volatility fact.

### 3.2 Predictive Market Information

The relation between volatility and other exogenous market information has received increasing attention from academic researchers and a selection of the most commonly proposed explanatory variables are summarised in Table 1.

Based on this, the potential explanatory variables related to market conditions used in this study include price range, bid-ask spread, trading volume, number of transactions, corresponding implied volatility and interest rates. The lagged information in Table 2 is also used in the RV modeling process.

#### 3.2.1 Implied Volatility

The implied volatility of at-the-money (ATM) one-month options is calculated from a volatility surface, which is fitted using all available trading information for FTSE 100 Index Options on the previous day. It is shown in Figure 2., and exhibits similar patterns as RV including long memory and persistence. However it should be noted that the implied volatility measure is an annualised estimate of the square root of the integrated variance over the subsequent one-month horizon and is significantly higher than the realised volatility estimates which is a single day measure of volatility even though this realised volatility value is also annualised. The difference between the implied volatility and the realised volatility is often referred to as the volatility risk premium. Thus the one-month ATM implied volatility may contain important information on the future path of the volatility so is a potentially useful explanatory variable to include in the modeling of realised volatility. Therefore lags of one to five periods of the implied volatility are included as explanatory variables in the modeling process.

#### 3.2.2 Price Range

There are two daily information variables created for the price range category including the absolute difference of open price and close price and the absolute difference of the days highest price and the days lowest price. The squared daily return is also used as an explanatory variable. It is a proxy for the absolute price range from the close of adjacent days. The price range information is shown in Figure 5 and Figure 6. The squared daily returns are illustrated in Figure 7. These figures display the typical volatility clustering patterns often observed in financial time series with periods of market turbulence inter-dispersed with periods of tranquillity.

#### 3.2.3 Trading Volume

There are two variables created for trading volume using the available data, namely total trading volume per day and average five-minute trading volume. The number of daily total transactions and the average trading duration (waiting time between
Table 1: Market/Economic Condition Variables Used in Volatility Forecasting

<table>
<thead>
<tr>
<th>Volatility Forecasting Study</th>
<th>Predictive Variables Considered</th>
</tr>
</thead>
<tbody>
<tr>
<td>[43]</td>
<td>Daily high-low range</td>
</tr>
<tr>
<td>[33]</td>
<td>Realised range, realised power variation</td>
</tr>
<tr>
<td></td>
<td>Realised bipower variant and volume</td>
</tr>
<tr>
<td>[44]</td>
<td>Trading volume and implied volatility</td>
</tr>
<tr>
<td>[45]</td>
<td>5 Categories (38 variables) tested</td>
</tr>
<tr>
<td>[50] [49] [48] [47] [20] [46]</td>
<td>Implied volatility</td>
</tr>
<tr>
<td></td>
<td>Volume</td>
</tr>
</tbody>
</table>

This table gives market/economic information variables used in volatility forecasting in different studies besides the lagged volatility and return information.

Figure 2: Implied Volatility of ATM Option Expiry in One Month
trades) are also used. These variables are illustrated in Figures 8 to 11. It is clear that the spikes in trading volume tend to coincide with higher volatility periods as shown in the daily price range and squared return figures, although the trading volume/transaction figures show that large jumps decay away more slowly than the daily price range and squared return values. Furthermore when volatility is high the average trading duration is low as the number of trades tend to increase in these higher volatility periods.

### 3.2.4 Bid-Ask Spread

For bid-ask spread metrics, two variables are created using the available data, namely the daily average bid-ask spread, and the maximum bid-ask spread. These are illustrated in Figure 12, and Figure 13. As expected the maximum bid-ask spread is more volatile than the average bid-ask spread varying between 0.2 and 0.13 index points\(^1\) whereas the average bid-ask spread is approximately 0.06 of an index point. It should be noted that the average index value over this period was approximately 4,523 thus the average bid-ask spread on the futures contracts in a proportionate sense is approximately 0.13 basis points (where one basis point is one hundredth of one percent) emphasising the high liquidity and low trading costs of FTSE 100 futures contracts.

### 3.2.5 Other Explanatory Variables

The squared RV is included as explanatory variable as squared RV is related to the RV’s volatility, that is, the volatility of volatility itself. Overnight Libor, one week and six-month Libor are used as the nominal interest rate proxies and are illustrated in Figures 14-16.

Comparing all the figures, the implied volatility series (in Figure 2), the absolute difference of the daily highest and lowest price (in Figure 6) and daily total transaction number (in Figure 10) seem to be highly associated with the annualised RV (in Figure 1). However, these market variables are all related one to another. In general, when the volatility is high, the price range tends to be high, the transaction volume tends to be high, the trading duration tends to be small, the bid-ask spread tends to be small and the interest rate tends to be low. The relationships invert when volatility is low.

### 3.3 GP Approach

In this study we employ GP for symbolic regression. The target variable is realised volatility and the evaluation of an individual GP tree is therefore an RV forecast. The available GP terminal set is outlined in Table 2 and the available function set is described in Table 3. Potential included variables in the GP trees include, five lagged values of RV, average daily trading (duration how often a trade occurs in a day), and Implied Volatility (IV) (estimated from the FTSE 100 index options data). The factors in Table 2 are from the previous day’s information if no explicit lag indication given.

Each GP individual has a fitness value which indicates how well it performs when tested in-sample on the training dataset. The fitness function in this application is the mean squared error as defined in Equation 7, where \(RV_{\text{target}}\) is the target RV value, \(RV_{\text{ind}}\) is the evaluation of the individual, and \(\text{Number}_{\text{Days}}\) is the number of data points in the training dataset.

\[
\text{Fitness} = \frac{\sqrt{\sum (RV_{\text{target}} - RV_{\text{ind}})^2}}{\text{Number}_{\text{Days}}} \quad (7)
\]

In the experiments, all results are reported averaged across 10 runs and each GP run consists of 50,000 individuals evolved over 50 generations. The operation of the GP system is illustrated in Figure 3. In order to reduce the chance of over-fitting, the maximum tree depth is set to six, based on initial experimentation. The training process is summarized in Figure 4.

### 3.4 Benchmark Models

Below we briefly outline the three benchmark models against which we compare the evolved GP models. The models are drawn from extant studies which forecast RV and include the Heterogeneous Autoregressive model (HAR), a generalised autoregressive conditional heteroscedastic (GARCH) model, and a stepwise regression (SR) model.

\(^1\)Each futures contract is worth 10 times the index when the futures contract expires so a bid-ask spread of 1 point in the futures contract is equivalent to a bid-ask spread of 0.1 point in the underlying index.
Table 2: Potential Explanatory Factors Used in GP

RV Lagged Information (one to five days lag)
Absolute Difference of Day Open and Close Price
Absolute Difference of Day Highest and Lowest Price
Daily Total Trading Volume
Average Five-minute Trading Volume
Daily Number of Transactions
Average Daily Absolute Difference of Bid and Ask Price
Maximum Daily Absolute Difference of Bid and Ask Price
Implied Volatility (IV) of a 1 Month at the Money Option
Average Daily Trading Duration in Seconds
Squared Daily Return
Squared RV
IV Lagged Information (two to five days lag)
Daily Libor
Weekly Libor
Six-month Libor

Figure 3: Flow Chart of GP Process
3.4.1 HAR model

In the Heterogeneous Autoregressive model (HAR) [32], RV is modeled using lagged information, including RV one day before, average RV in the last week and average RV in the last month. This model is provided in Equation 8, where \( c \), \( \alpha \), \( \beta \) and \( \gamma \) are constant coefficients.

\[
RV_t = c + \alpha RV_{t-1} + \beta RV_w + \gamma RV_m
\]

\[
RV_w = \frac{1}{5} \sum_{i=1}^{5} RV_{t-i},
\]

\[
RV_m = \frac{1}{21} \sum_{i=1}^{21} RV_{t-i}
\]  

(8)

The model coefficients are re-trained for each forecasted day.

3.4.2 GARCH Model

In the generalised autoregressive conditional heteroscedastic (GARCH) model, RV and its volatility (\( \epsilon \)) are modeled jointly in Eqs. 9, in which Equation 9a is the mean equation and Equation 9c is the conditional variance equation. \( z_t \) in Equation 9b is a white noise process. \( c \), \( \alpha \), \( \beta \), \( k \), \( \phi \) and \( \eta \) are constant coefficients, and \( \epsilon_{t-i} \) are independent identically distributed random variables sampled from a standard normal distribution.

\[
RV_t = c + \sum_{i=1}^{p} \alpha_i RV_{t-i} + \sum_{i=1}^{q} \beta_i \epsilon_{t-i} + \nu_t
\]  

(9a)

\[
\nu_t = \nu_t z_t
\]  

(9b)

\[
\nu_t^2 = k + \sum_{i=1}^{p} \phi_i \nu_{t-i}^2 + \sum_{i=1}^{q} \eta_i \epsilon_{t-i}^2
\]  

(9c)

3.5 SR Model

Stepwise regression (SR) is a systematic method for adding and removing potential explanatory variables from a multilinear model based on their statistical significance. The method begins with an initial model and then compares the explanatory power of incrementally larger and smaller models. At each step, the \( p \) value of an \( F \)-statistic is computed to test models with and without a potential factor. If a factor is not currently in the model, the null hypothesis is that the factor would have a zero coefficient if added to the model. If there is sufficient evidence to reject the null hypothesis, the factor is added to the model. Conversely, if a factor is currently in the model, the null hypothesis is that the factor has a zero coefficient. If there is insufficient evidence to reject the null hypothesis, the factor is removed from the model.

In this stepwise regression model, all market information variables in Table 2 are considered as potential explanatory variables. The stepwise regression model is fitted for each day’s RV forecast based on the in-sample training dataset, which is from the starting day until one day before the day to be forecasted and this is the same as GP model’s training dataset as explained in Section 3.3. As shown in Equation 10 the factors in each day’s forecasting model may vary in the stepwise selected linear regression model. By default there is always a constant intercept \( c \).
**Table 3:** Function Set Available to GP

<table>
<thead>
<tr>
<th>Function Set Available to GP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
</tr>
<tr>
<td>Subtraction</td>
</tr>
<tr>
<td>Multiplication</td>
</tr>
<tr>
<td>Division</td>
</tr>
<tr>
<td>Cumulative Distribution Function of Normal Distribution</td>
</tr>
<tr>
<td>Exponential Function</td>
</tr>
<tr>
<td>Nature Logarithm Function</td>
</tr>
<tr>
<td>Square Root</td>
</tr>
<tr>
<td>Cube Root</td>
</tr>
<tr>
<td>Sine Function</td>
</tr>
<tr>
<td>Cosine Function</td>
</tr>
</tbody>
</table>

\[
RV_i = c + \alpha_1 Var_1 + \alpha_2 Var_2 \cdots + \alpha_i Var_i \tag{10}
\]

## 4 Results

The out-of-sample results from our experiments are reported in this Section. Initially, we report the forecast errors for each modeling methodology, then we present a statistical analysis of these results. Finally, we report the results from a series of information encompassing tests.

### 4.1 Forecast Errors

The forecast errors are presented in Table 4. In this table, three measures of forecast error (MAE, MAPE and RMSE) are presented for GP and the benchmark models. The final column in the table presents the \( R^2 \) from the linear regression which regresses the actual RV against the predicted values from each method.

The results indicate that using the average of the GP models’ predictions produces a smaller MAE, MAPE and RMSE and also a notably higher correlation in terms of \( R^2 \) than is the case for the three benchmark models.

Comparing the three benchmark models amongst themselves, the SR model performs slightly better than the HAR model and the GARCH model in terms of MAE and MAPE error metrics. Using RMSE as the error metric, we note that the HAR model performs slightly better than the GARCH model and SR models. Considering \( R^2 \), HAR produces an \( R^2 \) of 30.4% as against 25.4% for the SR model, despite the SR model using the same market condition variables as per the GP approach whereas the HAR model only uses lagged volatility information as inputs. The GARCH model produces an \( R^2 \) of 29.71%, close to that of the HAR model.

### 4.2 Statistical Analysis

A series of statistical tests were undertaken to determine the significance of the results including three Diebold-Mariano tests introduced by [4] to test the equality of forecast accuracy between two models. The tests relate prediction error to some very general loss function and analyse the loss differential derived from errors produced by two competing models. The three tests include an asymptotic test that corrects for series correlation and two exact finite sample tests based on the sign test and the Wilcoxon signed-rank test. The last two sign-based tests in particular, works well for small samples.

In this application, the differential loss is defined as the difference of the squared forecast errors from two competing models. The differential loss series \((d_i)\) are calculated in Equation 11, where \(\text{Predicted}_i^a\) is the predicted value by model \(a\) and \(\text{Predicted}_i^b\) is the one from model \(b\) for the \(i^{th}\) RV observation \((i = 1, \ldots, T)\). Under the null hypothesis, the two competing models give equally accurate results. The alternative hypothesis is that two prediction models do not give equally accurate results. Three test statistics are shown in [4] to follow a standard normal distribution. The null hypothesis will be rejected at the 5 percent significant level if \(|S| > 1.96\).
Comparing the three benchmark models

\[ d_i = (RV_i - Predicted_i)^2 - (RV_i - Predicted_i')^2 \]  

(11)

- The Asymptotic Test: According to the Central Limit Theorem, when the sample size is large, the sample mean of the loss differential approximately follows a normal distribution with constant mean and variance. \( \bar{d} \) is the sample mean and \( \bar{\text{var}} \) is the estimate of the asymptotic long-run variance of \( \sqrt{T} \bar{d} \). In this application the forecast is only one step ahead, therefore no correlation adjustment is needed and \( \bar{\text{var}} \) is calculated as the variance of loss differential series. The statistic test is as per Equation 12.

\[ S_1 = \frac{\bar{d}}{\sqrt{\bar{\text{var}}}} \sim_A N(0, 1) \]  

(12)

- The Sign Test: When the sample size is small, a finite sample test such as the sign test can be conducted. The sign test statistic is constructed in Equation 13, where \( I(d_i) \) is one for \( d_i > 0 \) and otherwise zero.

\[ S_2 = \frac{\sum_{i=1}^{T} I(d_i) - 0.5T}{\sqrt{0.25T}} \sim_A N(0, 1) \]  

(13)

- Wilcoxon’s Signed Test: The test statistic is as per Equation 14, where \( \text{rank}(|d_i|) \) is the rank of the absolute values of loss differential series.

\[ S_3 = \frac{\sum_{i=1}^{T} |d_i| \cdot \text{rank}(|d_i|) - 0.5T}{\sqrt{0.25T}} \sim_A N(0, 1) \]  

(14)

Diebold-Mariano tests undertaken on a pairwise basis for the three competing models on the full out-of-sample time period are in Table 4. The resulting statistics are provided in Table 5. The null hypothesis, that two models give equal results in terms of forecasting accuracy, will be rejected at a 5 percent level if the relevant reported test statistic \( |X| > 1.96 \).

### 4.2.1 Summary of Statistical Results

The results from all three statistical tests give consistent results, that the prediction from GP is significantly different, from that produced by the HAR, GARCH and SR model, and as already seen in Table 4, the GP forecasts produced lowest error measures (and highest \( R^2 \)) among the competing models. The null hypothesis of no difference in the predictive accuracy between the methods, is rejected by three Diebold-Mariano tests at a 5 percent level for GP model against the HAR, GARCH and SR model. All three statistical tests also consistently confirm that there is no statistical difference in the predictions from three benchmark models although one model is better than the other in the tested error measures.

### 4.3 Information Encompassing Tests

These tests are used to determine whether one of a pair of forecasts contains all the useful information for a forecast, or conversely, does one forecast contain additional information not captured by the other. In this case, use of a combination of the forecasts can produce a better forecast than either alone.

The forecast information encompassing tests are performed using regression analysis on the full out-of-sample time period and the results are displayed in Table 6.

\[ RV = \alpha + \beta \text{Predicted} \]  

(15)
Table 5: Diebold-Mariano Tests

<table>
<thead>
<tr>
<th></th>
<th>HAR</th>
<th>GARCH</th>
<th>SR</th>
<th>GP-avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>-0.4731</td>
<td>-0.0880</td>
<td>-0.3303</td>
<td>3.3858</td>
</tr>
<tr>
<td>X2</td>
<td>-0.0559</td>
<td>0.7924</td>
<td>0.2809</td>
<td>3.4519</td>
</tr>
<tr>
<td>X3</td>
<td>-0.1054</td>
<td>0.7924</td>
<td>0.8545</td>
<td>3.4055</td>
</tr>
<tr>
<td>X1</td>
<td>-0.3303</td>
<td>0.7924</td>
<td>0.8545</td>
<td>4.1235</td>
</tr>
<tr>
<td>X2</td>
<td>0.7924</td>
<td>0.7924</td>
<td>2.7294</td>
<td>2.7294</td>
</tr>
<tr>
<td>X3</td>
<td>0.2809</td>
<td>0.8545</td>
<td>3.3744</td>
<td>3.3744</td>
</tr>
</tbody>
</table>

This table gives D-M tests including the asymptotic test (X1), sign test (X2) and Wilcoxon’s signed test (X3) on the full out-of-sample period. The null hypothesis that two models give equal accuracy results will be rejected at 5 percent significant level if $|X| > 1.96$.

Initially, a single factor analysis is performed for each model, where RV is the dependent variable and the prediction from each model is the explanatory variable as in Equation 15. The results from this are reported in Table 6. In evaluating these results it is important to distinguish between bias and predictive accuracy. In this single factor analysis, the prediction is unbiased only if $\alpha = 0$ and $\beta = 1$. The predictive power is indicated by $R^2$. A higher $R^2$ means higher predictive power. Ideally, we seek a forecast with low residual error and high $R^2$ [5]. While it might appear that bias is always undesirable, a biased forecast can still have predictive utility and conversely an unbiased forecast is of little use if the forecast errors produced by it are large.

The coefficients fitted in the single factor regression analysis in Table 6 shows that forecast results from the benchmark models (HAR, GARCH and SR) are closer to an unbiased prediction than those produced by GP. The intercept $\alpha$ are very close to zero and the coefficient for the model prediction, $\beta$ are closer to one in the HAR model. In the case of GP, $\alpha$ is significant as it is not zero at the 5 percent level and the $\beta$ value of 1.2813 is significantly higher than one. However, indicated by $R^2$ the predictive power from GP is much higher than that of the other models and hence it has significant utility despite its bias.

The second group of Information Encompassing Tests adds an extra prediction result from another model to the right-hand side of Equation 15 as a second regressor. An increased adjusted $R^2$ indicates that the first model can not subsume the second model and therefore that the second model provides extra predictive power. In other words, we are testing whether adding the prediction result from a second model as an extra explanatory factor can further improve the prediction result.

The adjusted $R^2$ from the regression of RV against the prediction from GP is found to be 38.84%, higher than the values for HAR, GARCH and SR. The adjusted $R^2$ values for the regression when both GP and one of the benchmark models, HAR / GARCH / SR forecasting results are used as regressors for RV increase to 40.20%, 39.24%, and 44.38% respectively. This indicates that the prediction from GP does not fully subsume the prediction from the benchmark models and suggests that a joint forecast from a hybrid of GP and the benchmark HAR, GARCH and SR models could potentially produce a higher predictive power.

From the empirical results, GP produces better forecasts than the benchmark models. There are two plausible reasons for this. First, GP takes account of market conditions (as inputs) in forecasting RV. Second, GP permits the use of non-linear functional forms between the RV and market conditions.
Forecasting daily returns volatility is crucial in finance. Traditionally, volatility is modeled using a time-series of lagged information only, an approach which is in essence atheoretical. Although the relationship of market conditions and volatility has been studied for decades, we still lack a clear theoretical framework to allow us to forecast volatility, despite having many plausible explanatory variables. This setting of a theory-weak environment suggests a useful role for powerful model induction methodologies such as Genetic Programming. This study forecasts one-day ahead realised volatility (RV) using a GP methodology that incorporates information on market conditions including trading volume, number of transactions, bid-ask spread, average trade duration and implied volatility. The forecasting result from GP is significantly better than that produced by the heterogeneous autoregressive model (HAR), the generalized autoregressive conditional heteroscedasticity (GARCH) and a linear stepwise regression (SR) model. Further, the regression-based Information Encompassing Tests show that the forecasts from benchmark models contain information not captured by GP, which indicates that a combination forecast from GP and conventional models could potentially improve forecast performance further. This is left for future work.

5 Conclusions

The above two tables provides forecast Information Encompassing Test results on the out-of-sample time period. The first table gives a regression fitting result when regressing the RV on the predicted value from each of five models as shown in Equation 15. The second table gives the adjusted $R^2$ when extra predictive results from another model is added in to the right-hand side of the Equation 15 as a second regressor. An increased adjusted $R^2$ indicates that the first model can not subsume the second model and the second model provides extra predictive power.

**Table 6: Out-of-sample Forecast Information Encompassing Test**

<table>
<thead>
<tr>
<th></th>
<th>HAR</th>
<th>GARCH</th>
<th>SR</th>
<th>GP-avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>-0.0048</td>
<td>-0.0132</td>
<td>0.0028</td>
<td>-0.0302</td>
</tr>
<tr>
<td>p-Value</td>
<td>0.7302</td>
<td>0.3863</td>
<td>0.8471</td>
<td>0.0346</td>
</tr>
<tr>
<td>$\beta$</td>
<td>1.0043</td>
<td>1.0876</td>
<td>0.9545</td>
<td>1.2813</td>
</tr>
<tr>
<td>p-Value</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>$R^2$</td>
<td>30.42%</td>
<td>29.71%</td>
<td>25.39%</td>
<td>39.32%</td>
</tr>
<tr>
<td>Adj-$R^2$</td>
<td>29.87%</td>
<td>29.16%</td>
<td>24.80%</td>
<td>38.84%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>HAR</th>
<th>GARCH</th>
<th>SR</th>
<th>GP-avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adj-$R^2$ 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adj-$R^2$ 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

It is found from analysis of the form of GP generated solutions that some variables including lagged values of RV, and average trading duration occurred frequently. The relationship between these factors and RV seem robust over time.

**Acknowledgment**

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant Number 08/SRC/FM1389.
Figure 5: Absolute Difference of Open to Close

Figure 6: Absolute Difference of High to Low

Figure 7: Daily Squared Return

Figure 8: Total of Trading Volume

Figure 9: Average 5-minute Trading Volume

Figure 10: Transaction Number
Figure 11: Average Trading Duration

Figure 12: Average Bid-Ask Spread

Figure 13: Max Bid-Ask Spread

Figure 14: Daily Overnight LIBOR

Figure 15: Daily 1-week LIBOR

Figure 16: Daily Six-month LIBOR
References


Dr. Yin completed the undergraduate study of Economics in Shaanxi University of Finance & Economics, P.R. China. She received MSc of Quantitative Finance and Ph.D degree from the Michael Smurfit Graduate Business School, University College Dublin, Ireland. She is currently a researcher in the Graduate Business School, University College Dublin and a senior risk analyst in Allied Irish Bank. Her main research interests include applying evolutionary computation in financial modelling, financial market microstructure, financial derivatives hedging, credit risk modelling. She has presented papers at international conferences including GECCO (2007 Genetic and Evolutionary Computation Conference) and CEC (2015 IEEE Congress on Evolutionary Computation).

Dr. O’Sullivan is a lecturer in finance and director of the MSc in Quantitative Finance at the Michael Smurfit Graduate Business School, University College Dublin. His primary research interests are numerical methods for derivatives pricing, derivatives markets, interest rate models and the application of natural computing algorithms in finance. He has published works in journals such as Quantitative Finance, the International Journal of Theoretical and Applied Finance and the Journal of Computational Finance. He presented papers at international conferences including the European Finance Association, Quantitative Methods for Finance and Computational Financial Econometrics.

Prof. Brabazon’s research interests concern the development of natural computing algorithms and their application to real-world problems. He is a co-founder and co-director of the Natural Computing Research & Applications Group at UCD (see http://ncra.ucd.ie). He has published in excess of 200 peer-reviewed journal and conference papers. He has also authored or edited fourteen books including ‘Biologically Inspired Algorithms for Financial Modelling’, ‘Foundations in Grammatical Evolution for Dynamic Environments’ and ‘Natural Computing in Computational Finance’ (Volumes I, II, III and IV), and ‘Natural Computing Algorithms’. Prof. Brabazon completed his undergraduate studies at UCD and subsequently undertook graduate degree studies at a number of universities including Trinity College Dublin, Heriot-Watt, Kingston and Stanford.
SELF-CONFIGURING HYBRID EVOLUTIONARY ALGORITHM FOR FUZZY IMBALANCED CLASSIFICATION WITH ADAPTIVE INSTANCE SELECTION

Vladimir Stanovov, Eugene Semenkin, Olga Semenkina

Institute of Computer Science and Telecommunications, Siberian State Aerospace University Krasnoyarskii rabochii ave. 31, 660014, Krasnoyarsk, Russian Federation

Abstract

A novel approach for instance selection in classification problems is presented. This adaptive instance selection is designed to simultaneously decrease the amount of computation resources required and increase the classification quality achieved. The approach generates new training samples during the evolutionary process and changes the training set for the algorithm. The instance selection is guided by means of changing probabilities, so that the algorithm concentrates on problematic examples which are difficult to classify. The hybrid fuzzy classification algorithm with a self-configuration procedure is used as a problem solver. The classification quality is tested upon 9 problem data sets from the KEEL repository. A special balancing strategy is used in the instance selection approach to improve the classification quality on imbalanced datasets. The results prove the usefulness of the proposed approach as compared with other classification methods.

Keywords: Fuzzy classification, instance selection, genetic fuzzy system, self-configuration

1 Introduction

Today the area of machine learning (ML) techniques is quickly developing due to recent advances in computer and internet technologies. These advances led to the need to process, analyse and understand massive amounts of data. Modern machine learning methods often use evolutionary algorithms (EAs) as design techniques to adjust the weights or define the structure of the solution. The classical evolutionary algorithm – the genetic algorithm (GA) is a powerful method, although there is a set of specialized approaches for various problems, such as neural network structure design and weight adjustment, neuro-fuzzy inference system design, fuzzy rule base design, etc. The specialized evolutionary algorithms applied to solve machine learning problems are called genetics-based machine learning methods (GBML). A typical application of such methods is solving of classification problems [1-3].

In this paper we focus on fuzzy classification methods, i.e. fuzzy rule bases. There is a set of GBML approaches used to solve the problem of a fuzzy rule base design for classification. These methods differ in several ways: some of them optimize the positions and shape of fuzzy terms, while others optimize the rules and their combinations. The so-called Pittsburg approach, when an individual in the EA is a rule base, is used more often than the Michigan algorithm, when an individual is a single rule. However, the methods which combine both the Michigan and Pittsburgh approaches seem to be the most promising.
In the case of solving real-world classification problems, researchers may face different issues related to the data available. These issues are: too large or too small amount of data available, large numbers of classes, irrelevant features and instances, errors in data measurements and missing data, imbalances in the amount of data per class and so on. These issues may cause serious difficulties in learning the actual background of the real-world process and representing it in a classification model. In this paper we focus on two main problems: a large amount of data and an imbalance in the number of instances.

The problem of a large amount of data can be solved with data reduction methods (DR). There are several groups of methods, including training set selection (TSS), active learning, instance selection (IS) and feature selection. In our work we concentrate on instance selection methods, which are used for supervised learning problems, such as classification. Instance selection and feature selection methods are divided into two groups: filter and wrapper approaches.

Instance selection is strongly connected to the problem of irrelevant instances in the training sample. Removing instances from the training set does not necessarily lead to the loss of information, especially in the case of a large number of training examples. So, the instance selection method can be not only a way to decrease the computational complexity of an algorithm, but also a method to increase the overall accuracy of the resulting model. The idea behind this study is the development of a method for selecting the instances in such a way in order to increase the learning capabilities of a GBML algorithm.

As a GBML method for our experiments, we used our modification of the hybrid fuzzy evolutionary algorithm, originally proposed by the H. Ishibuchi group. Our modifications include self-configuration, parameter tuning and some adjustments for imbalanced datasets. Unlike our previous work, in this paper we perform instance selection testing for different parameters and also consider the influence of instance selection on various classification measures. Special attention is paid to the class imbalance problem.

The rest of the paper is organized as follows: Section 2 describes the classification method, Section 3 contains the description of an instance selection approach, Section 4 contains the experimental results, and Section 5 concludes the paper.

2 Hybrid fuzzy GBML algorithm

The original hybrid fuzzy evolutionary algorithm was introduced by H. Ishibuchi et. al. in [4]. However, as we developed our method from scratch, we provide a short description of our implementation.

The main loop of the evolutionary algorithm implements the Pittsburg approach, i.e. each individual is a rule base. The number of rules in the rule base is not fixed and may change during the evolutionary process for every individual.

There were four different fuzzy partitions used, namely partitions into 2, 3, 4 and 5 fuzzy terms, as well as the “don’t care” condition, resulting in 15 fuzzy sets $A_0 - A_{14}$. The fuzzy sets are presented in Figure 1.

![Figure 1. Fuzzy partitions](image)

Each rule was represented as an integer string number from 0 to 14. The Pittsburg-type algorithm consists of the following steps:

1. Initialization using instances from the training set
2. Fitness calculation
3. Selection, Crossover, Mutation
4. Apply the Michigan-style part to each individual
5. If stopping criteria are not satisfied, go to step 2.
The Michigan part contain the following steps:

1. Define each rule in a rule base as an individual and calculate its fitness.
2. Remove or add new rules to the rule base with genetic or heuristic approach.
3. Return the modified rule base to the population.

Let us describe each step in detail. The initialization procedure uses instances from the training set to generate new rules. This step is important as in the case of a large number of features it is difficult to randomly generate a rule which would describe at least one instance correctly. To generate a rule, a random instance is taken from the sample, and the membership values $\mu$ are calculated for each variable for all fuzzy sets. After this, the probability of a fuzzy set to be selected is determined as

$$P(A_j) = \frac{\mu_{A_j}(x_{pi})}{\sum_{k=1}^{14} \mu_{A_k}(x_{pi})}.$$  

The same procedure is repeated for all variables. Next, for every variable the fuzzy set number is changed to “Don’t care” condition with 0.9 probability. After generating a rule, the confidence value is calculated using the available:

$$Conf(A_q \rightarrow Classk) = \frac{\sum_{k=1}^{14} \mu_{A_k}(x_{pi})}{\sum_{k=1}^{14} \mu_{A_k}(x_{pi})}.$$  

If the confidence value is larger than 0.5, then the fuzzy rule is added into the rule base.

The class number associated with a rule is not coded in the chromosome, and is determined heuristic using confidence values. For this purpose, the confidence values are calculated for all classes for every rule, and the class number corresponding to the largest confidence value is set.

The rule weight [5] is also calculated using confidence value:

$$CF_q = Conf(A_q \rightarrow Classk) - \sum_{k=1, k \neq C_q}^{M} Conf(A_q \rightarrow Classk).$$

The classification is performed by determining the winner-rule, i.e. the rule that has the largest $\mu_{A_q}(x_{pi})$ $CF_q$ value. The instance is classified into a class, corresponding to the winner-rule.

We generated rules 20 times for each individual, and the maximum number of rules in the algorithm was limited to 40. If all the rules received confidence values lower than 0.5, i.e. the rule base was empty, than the generation procedure was repeated.

The fitness value for all individuals was calculated as a combination of three main criteria, the training sample error $f_1(i)$, the number of rules $f_3(i)$, and the total length of all rules $f_2(i)$. The training sample error was the percentage with weight coefficient $w_1 = 1$, the two other criteria were used with weights $w_2 = 1$ and $w_3 = 1$.

For the selection step we applied three classical selection schemes, i.e. fitness proportional, rank and tournament selection with tournament size of 2.

There was only one specialized crossover operator used, which combines two rule bases to produce one offspring. For this purpose, the number of rules for the offspring is determined as a random number from 1 to $|S_1| + |S_2|$, where $|S_i|$ is the number of rules for individual $i$. If $|S_1| + |S_2|$ exceeds the maximum number of rules (i.e. 40 in our computational experiments) then the number of rules is set to be equal to this maximum number. After this, the rule base is filled with new rules in a random way from a general rule pool created from parents’ rules.

The mutation step is similar to the one used in GAs. The probability of a gen (fuzzy set) to be changed is defined as $1/(n \cdot |S_1|)$ for average mutation, $3/(n \cdot |S_1|)$ for strong mutation and $1/(3 \cdot n \cdot |S_1|)$ for weak mutation, where $n$ is the number of variables. Each of the fuzzy sets in a rule base could mutate, including “Don’t care” conditions. If the confidence value of a fuzzy rule after the use of the mutation operator becomes lower than 0.5, the rule is excluded from the rule base. If there are no rules left after mutation then several rules are generated using the same procedure as at the initialization step.

The Michigan-style part is applied to every rule base after the mutation operator. At the first step, the fitness values are calculated for every rule. The rule fitness is equal to the number of instances correctly classified with this rule. If there are two identical rules, only one of them gets the non-zero fitness value. There were three types of the Michigan
part: adding new rules, deleting rules and replacing rules, i.e. first deleting, then adding. In the case of deleting rules, the number of rules $k$ to be deleted is defined as $5 \cdot (k - 1) < |S| < 5 \cdot k$. The rules with the lowest fitness values are deleted first. In the case of adding new rules, the number of rules to be added is defined in the same way as for deleting, but if the number of rules exceeds the maximum number of rules then no rule is added.

New rules are added with the use of two different methods, heuristic and genetic ones. The heuristic method uses incorrectly classified instances to generate new rules using the same procedure as at the initialization step. The genetic method uses rules from the rule base to produce new rules with the tournament selection, uniform crossover and average mutation as in the genetic algorithm.

One of the modifications of the original algorithm was the self-configuration procedure which was implemented for selection, mutation, the Michigan part and adding rules in the Michigan part. The self-configuration method was first introduced in [6, 7] and was successfully applied to a similar problem in [8]. The main idea of the method is the assigning to genetic operators of the probabilities to be used in the future based on their success in the past. The method uses averaged fitness values of offspring generated by a certain operator to select a winner-operator at each generation. The winner’s probability increases, while all other operators get their probabilities decreased.

We will describe this method in detail. Let $z$ be the number of different operators of the $i-th$ type. The starting probability values are set to $p_i = 1/z$. The success estimation for every type of operator is performed based on the averaged fitness values:

$$AvgFit_i = \frac{1}{2} \sum_{j=1}^{n_i} f_{ij}, l = 1, 2, \ldots, z,$$

where $n_i$ is the number of offspring formed with the $i-th$ operator, $f_{ij}$ is the fitness value of the $j-th$ offspring, produced with the $i-th$ operator. $AvgFit_i$ is the average fitness of the solutions, produced with the $i-th$ operator. Then the probability of applying the operator, whose $AvgFit_i$ value is the highest among all the operators of this type, is increased by $(z \cdot K - K)/(z \cdot N)$, and the probabilities of applying other operators are decreased by $K/(z \cdot N)$, where $N$ is the number of evolutionary algorithm generations, $K$ is a constant value usually equal to 0.5.

### 3 Adaptive instance selection algorithm for imbalanced classification problems

As we mentioned in the Introduction, data reduction does not necessary lead to lower classification quality. In some cases excluding instances from the training set may lead to classification quality improvement, because the deleted instances were noisy, repeated many times and so on.

So, most of the instance selection methods are focused on both data reduction and classification quality improvement [9, 10]. However, these methods are mainly designed to create a subset only once, and not to train an accurate classifier.

The proposed instance selection algorithm is designed for learning algorithms which use a lot of iterations during the learning process. This method does not require any data preprocessing, it is not based on the $k-NN$ method and does not require the distance to be calculated between instances. Instead, instances are selected based on classification quality, i.e. it implements the wrapper approach.

Let us describe the idea in detail. At the first stage, a subsample of the training sample is created, having a fixed size (set by user). At this step, the instances are selected with equal probabilities. After this, the learning process starts for a number of generations (iterations), called the adaptation period. Only the training subset is used.

Here every instance receives a counter value $U_i$, which means the number of successful uses of the $i-th$ instance. At the beginning, all $U_i = 1, i=1 \ldots n$, and then are changed for every instance. At the end of the adaptation period, the best current solution, i.e. best classifier, for the subsample is used to update the counter values. Only the counters of the instances which are in the subsample are updated. If an instance $j$ was classified correctly, then $U_j = U_j + 1$, otherwise $U_j = 1$.

So, the sample instances which were classified correctly get their counters updated, while the counters for incorrectly classified measurements are reset. After the update of counters, a new subsample
is created, using new counters. The probability for an instance \( i \) to be selected is calculated using the equation:

\[
p_i = \frac{1/U_i}{\sum_{j=1}^{n} 1/U_j}.
\]

According to this equation, increasing the counter leads to a decrease in the probability of an instance being included into the subsample. The denominator in this case is needed for normalization.

Thus, the adaptive instance selection algorithm assigns lower probabilities for instances which are easier to classify. At the same time, instances which are difficult to classify or those which have not been used before get larger probabilities to be selected in the new training sample. This procedure implements two main principles: the exploration of areas of the feature space unknown before, and using information about classification quality to build a better separation between classes.

During the learning process, the best solution for every subsample is changed after every adaptation period, as it depends on the instances selected. Because of this, the probabilities are also changed, as the best solution for the subsample may present different results for the whole training set. As the sample constantly changes, different solutions can be received, improving the search process. In the case of an evolutionary algorithm being used as a learning method, during every new adaptation period, a population of solutions from the previous step is saved. Individuals in this population are able to classify instances of the new training set, but on most occasions with lower accuracy.

At each generation of the evolutionary algorithm, the best solution found for the subsample is checked on the whole sample. This step is required to exclude losing the best found solution. Moreover, the best solution for the whole training set is included into population together with the best solution for the subsample. At the end of the adaptation period, all individuals of the current population are checked on the whole training sample. This step is required as the population may contain other solutions which could have even better generalization than the best solution for the current subsample.

One more important issue during creating a subsample is setting the amount of instances to be selected for every class. If the described procedure will be used without taking the original class distribution into account, the distribution in the subsample may significantly differ. Because of this, the number of instances available for every class has to influence the subsample creation procedure. The problem of imbalanced classification with fuzzy rule bases has been previously studied in [11, 12].

One of the methods for taking this distribution into consideration is a stratified approach, which is commonly used in cross-validation. This approach is important for balanced problems, as it saves the class distribution.

However in the case of solving imbalanced classification problems, the stratified approach may be not the best way. As the described adaptive instance selection approach may choose different groups of instances, for imbalanced problems it is possible to sample more balanced subsets, than the original set. This idea is implemented in the balancing approach, which creates the subset so that the number of instances would be as balanced as possible. In this case, the minority classes may be entirely included into the training set.

![Figure 2. Example of instance selection](image)

In Figure 2, we show a graphical example of which of the instances are chosen by the algorithm and how the active instance selection procedure leads to an improvement in the classification quality. The example is a two-class problem with 31 measurements, with different class instances shown with crosses and squares. On the left side, the start of the algorithm is shown, where all instances have the same probability of being selected, shown by the size of the points. The measurements in circles
are the instances that have been selected into the current subsample. The separating surface obtained by the end of the adaptation period does not classify all patterns correctly: instances 16 and 18, which are in the subsample, are misclassified. On the right side, the next adaptation period is shown. Instances 16 and 18 did not change their size (i.e. $U_{16} = U_{18} = 1$) compared to unused instances, while all the rest (1, 3, 7, 12, 14, 19, 20, 28, 29), used and correctly classified, received a lower probability and thus a smaller size.

The lower graph in Figure 2 demonstrates the situation after several adaptation periods, where those instances that are close to the separating plane between classes have higher probabilities of being chosen (and they are actually chosen at this iteration), while the remaining instances that lie further away have lower probabilities.

4 Experimental setup and results

We performed a set of computational experiments with the presented adaptive instance selection algorithm and hybrid fuzzy GMBL algorithm to evaluate their effectiveness. The experiments were performed on a 4-core Intel Core-i7 2600K@4400MHz processor, the program system was implemented in C++ with GCC 4.8.1 compiler, only standard C++ libraries used.

The parameters of the hybrid evolutionary fuzzy classification algorithm were set in the same way for all experiments: the population size was equal to 100, the number of generations to 10000 and the maximum number of rules to 40. The parameters for the instance selection, i.e. the size of the subsample and the length of the adaptation period varied. The subsample size was set to 5%, 10%, 15%, 20%, 25% and 30% of the original training sample, the length of the adaptation period was set to 50, 100, 200 and 400 generations. Also two sampling strategies were tested – a stratified and a balancing strategy for the subsample. For each combination of parameters a 10-fold cross-validation procedure was performed twice, so that the classification quality measures were averaged over 20 runs. The standard algorithm without instance selection has also been tested.

The classification problems for testing were taken from the UCI [13] and KEEL [14] repositories. The selected problems have a large number of instances, variables and classes, some of which are highly imbalanced. The parameters of the datasets are presented in Table 1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of instances</th>
<th>Number of features</th>
<th>Number of classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magic</td>
<td>19020</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>Page-blocks</td>
<td>5472</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Penbased</td>
<td>10992</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>Phoneme</td>
<td>5404</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Ring</td>
<td>7400</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>Satimage</td>
<td>6435</td>
<td>36</td>
<td>6</td>
</tr>
<tr>
<td>Segment</td>
<td>2310</td>
<td>19</td>
<td>7</td>
</tr>
<tr>
<td>Texture</td>
<td>5500</td>
<td>40</td>
<td>11</td>
</tr>
<tr>
<td>Twonorm</td>
<td>7400</td>
<td>20</td>
<td>2</td>
</tr>
</tbody>
</table>

The error values in Table 2 are the error rates in percentages. The next table contains the best results with instance selection and the stratified strategy.

The best instance selection configurations for the stratified strategy were the following, for Magic: 30% of the training sample and adaptation period length of 400 generations; for Page-blocks: 30%, 200; for Penbased: 15%, 50; Phoneme: 25%, 200; Ring: 25%, 50; Satimage: 30% , 50; Segment: 30%, 50; Texture: 25%, 50; Twonorm: 30%, 100. So, for most of the problems the best results were obtained with the maximum subsample size.

The difference between the standard method and instance selection with the stratified strategy is presented in Table 4.

The time ratio was calculated as a ratio of the modified algorithm to the standard algorithm in percentages. The maximum difference in accuracy was 3.59% (Penbased problem). Only for the Phoneme problem was the effect of instance selection negative (i.e. 1.26% lower for the training sample). The time spent by the modified algorithm was from 18% to 32% of the original, depending mainly on the size of the subsample. Results for the balancing strategy are presented in Table 5.

Table 6 contains the comparison of Tables 2 and 5.
Table 2. Results for the standard algorithm

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Training error</th>
<th>Test error</th>
<th>Number of rules</th>
<th>Rule length</th>
<th>Time(minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magic</td>
<td>15.06</td>
<td>15.73</td>
<td>12.6</td>
<td>3.82</td>
<td>370.62</td>
</tr>
<tr>
<td>Page-blocks</td>
<td>3.52</td>
<td>3.96</td>
<td>10.1</td>
<td>3.49</td>
<td>94.48</td>
</tr>
<tr>
<td>Penbased</td>
<td>7.06</td>
<td>7.46</td>
<td>30.5</td>
<td>6.23</td>
<td>385.2</td>
</tr>
<tr>
<td>Phoneme</td>
<td>15.03</td>
<td>16.48</td>
<td>18.3</td>
<td>3.01</td>
<td>84.95</td>
</tr>
<tr>
<td>Ring</td>
<td>4.64</td>
<td>5.82</td>
<td>26.6</td>
<td>3.83</td>
<td>226.70</td>
</tr>
<tr>
<td>Satimage</td>
<td>12.22</td>
<td>14.22</td>
<td>20.4</td>
<td>11.06</td>
<td>345.82</td>
</tr>
<tr>
<td>Segment</td>
<td>4.55</td>
<td>6.45</td>
<td>22.2</td>
<td>6.69</td>
<td>146.18</td>
</tr>
<tr>
<td>Texture</td>
<td>6.50</td>
<td>7.75</td>
<td>25.8</td>
<td>14.90</td>
<td>352.26</td>
</tr>
<tr>
<td>Twonorm</td>
<td>4.42</td>
<td>6.06</td>
<td>17.4</td>
<td>7.40</td>
<td>254.88</td>
</tr>
</tbody>
</table>

Table 3. Results for instance selection with the stratified strategy

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Training error</th>
<th>Test error</th>
<th>Number of rules</th>
<th>Rule length</th>
<th>Time(minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magic</td>
<td>14.98</td>
<td>15.23</td>
<td>10.7</td>
<td>4.77</td>
<td>121.19</td>
</tr>
<tr>
<td>Page-blocks</td>
<td>3.59</td>
<td>3.83</td>
<td>7.8</td>
<td>4.94</td>
<td>20.73</td>
</tr>
<tr>
<td>Penbased</td>
<td>3.18</td>
<td>3.87</td>
<td>31.3</td>
<td>6.42</td>
<td>65.36</td>
</tr>
<tr>
<td>Phoneme</td>
<td>16.29</td>
<td>16.77</td>
<td>12.5</td>
<td>3.15</td>
<td>15.45</td>
</tr>
<tr>
<td>Ring</td>
<td>3.38</td>
<td>4.86</td>
<td>30.0</td>
<td>4.12</td>
<td>54.08</td>
</tr>
<tr>
<td>Satimage</td>
<td>11.28</td>
<td>13.05</td>
<td>27.9</td>
<td>6.72</td>
<td>96.31</td>
</tr>
<tr>
<td>Segment</td>
<td>3.01</td>
<td>5.13</td>
<td>25.1</td>
<td>6.32</td>
<td>39.21</td>
</tr>
<tr>
<td>Texture</td>
<td>3.34</td>
<td>4.31</td>
<td>28.1</td>
<td>11.76</td>
<td>97.47</td>
</tr>
<tr>
<td>Twonorm</td>
<td>3.07</td>
<td>4.66</td>
<td>18.7</td>
<td>7.45</td>
<td>58.92</td>
</tr>
</tbody>
</table>

Table 4. Difference between Table 2 and Table 3, positive is better

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Training error</th>
<th>Test error</th>
<th>Number of rules</th>
<th>Rule length</th>
<th>Time ratio, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magic</td>
<td>0.08</td>
<td>0.5</td>
<td>1.9</td>
<td>-0.95</td>
<td>32.70</td>
</tr>
<tr>
<td>Page-blocks</td>
<td>-0.07</td>
<td>0.13</td>
<td>2.3</td>
<td>-1.45</td>
<td>21.94</td>
</tr>
<tr>
<td>Penbased</td>
<td>3.88</td>
<td>3.59</td>
<td>-0.8</td>
<td>-0.19</td>
<td>16.97</td>
</tr>
<tr>
<td>Phoneme</td>
<td>-1.26</td>
<td>-0.29</td>
<td>5.8</td>
<td>-0.14</td>
<td>18.19</td>
</tr>
<tr>
<td>Ring</td>
<td>1.26</td>
<td>0.96</td>
<td>-3.4</td>
<td>-0.29</td>
<td>23.86</td>
</tr>
<tr>
<td>Satimage</td>
<td>0.94</td>
<td>1.17</td>
<td>-7.45</td>
<td>4.34</td>
<td>27.85</td>
</tr>
<tr>
<td>Segment</td>
<td>1.54</td>
<td>1.32</td>
<td>-2.9</td>
<td>0.37</td>
<td>26.82</td>
</tr>
<tr>
<td>Texture</td>
<td>3.16</td>
<td>3.44</td>
<td>-2.3</td>
<td>3.14</td>
<td>27.67</td>
</tr>
<tr>
<td>Twonorm</td>
<td>1.35</td>
<td>1.4</td>
<td>-1.3</td>
<td>-0.05</td>
<td>23.12</td>
</tr>
</tbody>
</table>
Table 5. Results for instance selection with the balancing strategy

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Training error</th>
<th>Test error</th>
<th>Number of rules</th>
<th>Rule length</th>
<th>Time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magic</td>
<td>14.62</td>
<td>15.08</td>
<td>17.3</td>
<td>3.63</td>
<td>129.65</td>
</tr>
<tr>
<td>Page-blocks</td>
<td>2.71</td>
<td>3.25</td>
<td>18.9</td>
<td>4.82</td>
<td>18.53</td>
</tr>
<tr>
<td>Penbased</td>
<td>3.27</td>
<td>3.81</td>
<td>30.8</td>
<td>6.11</td>
<td>91.42</td>
</tr>
<tr>
<td>Phoneme</td>
<td>15.63</td>
<td>16.88</td>
<td>24.0</td>
<td>2.84</td>
<td>19.62</td>
</tr>
<tr>
<td>Ring</td>
<td>3.23</td>
<td>5.08</td>
<td>30.2</td>
<td>3.85</td>
<td>68.23</td>
</tr>
<tr>
<td>Satimage</td>
<td>10.57</td>
<td>12.93</td>
<td>27.2</td>
<td>5.84</td>
<td>85.12</td>
</tr>
<tr>
<td>Segment</td>
<td>3.55</td>
<td>5.19</td>
<td>25.1</td>
<td>6.24</td>
<td>32.40</td>
</tr>
<tr>
<td>Texture</td>
<td>3.37</td>
<td>4.45</td>
<td>27.0</td>
<td>12.81</td>
<td>114.79</td>
</tr>
<tr>
<td>Twonorm</td>
<td>4.03</td>
<td>4.81</td>
<td>15.0</td>
<td>7.74</td>
<td>38.11</td>
</tr>
</tbody>
</table>

Table 6. Difference between Table 2 and Table 5, positive is better

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Training error</th>
<th>Test error</th>
<th>Number of rules</th>
<th>Rule length</th>
<th>Time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magic</td>
<td>0.44</td>
<td>0.65</td>
<td>-4.7</td>
<td>0.19</td>
<td>34.98</td>
</tr>
<tr>
<td>Page-blocks</td>
<td>0.81</td>
<td>0.71</td>
<td>-8.8</td>
<td>-1.33</td>
<td>19.61</td>
</tr>
<tr>
<td>Penbased</td>
<td>3.79</td>
<td>3.65</td>
<td>-0.3</td>
<td>0.12</td>
<td>23.73</td>
</tr>
<tr>
<td>Phoneme</td>
<td>-0.60</td>
<td>-0.40</td>
<td>-5.75</td>
<td>0.17</td>
<td>23.10</td>
</tr>
<tr>
<td>Ring</td>
<td>1.41</td>
<td>0.74</td>
<td>-3.6</td>
<td>-0.02</td>
<td>30.10</td>
</tr>
<tr>
<td>Satimage</td>
<td>1.65</td>
<td>1.29</td>
<td>-6.8</td>
<td>5.22</td>
<td>24.61</td>
</tr>
<tr>
<td>Segment</td>
<td>1.00</td>
<td>1.26</td>
<td>-2.9</td>
<td>0.45</td>
<td>22.16</td>
</tr>
<tr>
<td>Texture</td>
<td>3.13</td>
<td>3.30</td>
<td>-1.2</td>
<td>2.09</td>
<td>32.59</td>
</tr>
<tr>
<td>Twonorm</td>
<td>0.39</td>
<td>1.25</td>
<td>2.4</td>
<td>-0.34</td>
<td>14.95</td>
</tr>
</tbody>
</table>

Applying the balancing strategy changed the behaviour of the algorithm for most of the datasets. For the Phoneme dataset the difference changed from -1.29% to -0.4%. The best improvement was for the Penbased problem, 3.65%. Although the average number of rules increased for 8 datasets out of 9, the average rule length decreased for 6 datasets out of 9, which means that the algorithm has designed more rules, but they are less complex.

The best configurations for the balancing strategy were: Magic: 30%, 200; Page-blocks: 20%, 50; Penbased: 25%, 100; Phoneme: 30%, 50; Ring: 30%, 50; Satimage: 30%, 50; Segment: 25%, 50; Texture: 30%, 50; Twonorm: 20%, 200.

However, the effect of the balanced strategy was mainly demonstrated not in terms of overall accuracy, but in terms of a more balanced classification. To compare how successfully the algorithm recognizes both minority and majority classes, we used the Recall_M measure from [15]. Table 7 contains the comparison of (1-Recall_M)*100 values for the original algorithm, stratified strategy and balanced strategy.

The presented results prove that using the balanced strategy not only increases the overall classification accuracy, but also allows different classes to be recognized more precisely. However, because for most cases the best results were obtained when using 30% of the sample, applying the balancing strategy to the Page-blocks problem, for example, still resulted in a highly imbalanced subsample. Nevertheless, the improvement for this problem is about 13% compared with original algorithm.

In Tables 8 and 9 we provide the (1-Recall_M)*100 values and accuracy values for the Page-blocks problem for all subsample sizes and all adaptation period lengths.

The lowest (1-Recall_M)*100 value was obtained when the size of the subsample was minimal, i.e. when the subsample is as balanced as possible. We should mention that even for 5% for the Page-blocks datasets it was impossible to create a completely balanced subsample, as the minority class contains only 28 measurements, while all the other classes in this case contained 54 measurements. The bias towards the majority classes
has been almost eliminated thanks to the balancing strategy, as the accuracy value was equal to 7.95.

For comparison, we provide the accuracy values for the Page-blocks problem.

Table 8. (1- Recall_M)*100 values for Page-blocks

<table>
<thead>
<tr>
<th>Adaptation period length</th>
<th>50</th>
<th>100</th>
<th>200</th>
<th>400</th>
</tr>
</thead>
<tbody>
<tr>
<td>5%</td>
<td>10.18</td>
<td>12.92</td>
<td>15.18</td>
<td>12.06</td>
</tr>
<tr>
<td>10%</td>
<td>12.92</td>
<td>13.71</td>
<td>14.59</td>
<td>17.51</td>
</tr>
<tr>
<td>15%</td>
<td>20.51</td>
<td>19.63</td>
<td>18.35</td>
<td>19.26</td>
</tr>
<tr>
<td>20%</td>
<td>23.05</td>
<td>21.23</td>
<td>22.72</td>
<td>22.36</td>
</tr>
<tr>
<td>25%</td>
<td>28.35</td>
<td>26.14</td>
<td>23.05</td>
<td>22.24</td>
</tr>
<tr>
<td>30%</td>
<td>28.04</td>
<td>27.73</td>
<td>28.42</td>
<td>29.51</td>
</tr>
</tbody>
</table>

Table 9. Accuracy values for Page-blocks

<table>
<thead>
<tr>
<th>Adaptation period length</th>
<th>50</th>
<th>100</th>
<th>200</th>
<th>400</th>
</tr>
</thead>
<tbody>
<tr>
<td>5%</td>
<td>7.95</td>
<td>8.44</td>
<td>9.25</td>
<td>8.48</td>
</tr>
<tr>
<td>10%</td>
<td>5.81</td>
<td>6.03</td>
<td>6.67</td>
<td>7.27</td>
</tr>
<tr>
<td>15%</td>
<td>4.00</td>
<td>4.50</td>
<td>4.90</td>
<td>5.77</td>
</tr>
<tr>
<td>20%</td>
<td>3.25</td>
<td>3.40</td>
<td>3.93</td>
<td>4.40</td>
</tr>
<tr>
<td>25%</td>
<td>3.78</td>
<td>3.78</td>
<td>3.69</td>
<td>4.15</td>
</tr>
<tr>
<td>30%</td>
<td>3.38</td>
<td>3.69</td>
<td>3.78</td>
<td>4.06</td>
</tr>
</tbody>
</table>

For 5% of the training set in the subset the accuracy decreases by a factor of 2 compared to values obtained when using 30% of the sample. However, the classifiers obtained with 5% of the sample can be more suitable, as they are capable to successfully classify all classes with the same accuracy.

In Tables 10 and 11 we also provide the confusion matrixes for the Page-blocks problem to show the effect of the balancing strategy.

In every row of tables 10 and 11 the predicted class having the largest number of instances classified was marked in bold. The original algorithm correctly classifies the first, majority class, but the other classes are also classified into the first class, except the second one. This means that actually the algorithm correctly classifies only the first and second classes. When using the balancing strategy, the situation changes, and most of the instances are correctly classified in their class, however, this leads to a lower general classification quality.

For the remaining problems similar results were obtained. The main trend is that more balanced classifiers can be trained with a smaller size of the sample. If the sample becomes balanced at some particular percentage, there is no sense in decreasing its size any more.

In Figures 3-10 we show the surface plot as a graphical representation of the dependence of accuracy on the size of the subsample and the length of the adaptation period.

![Figure 3. Segment](image-url)
Table 10. Confusion matrix for the Page-blocks problem, original algorithm

<table>
<thead>
<tr>
<th>Class</th>
<th>Pred. 1</th>
<th>Pred. 2</th>
<th>Pred. 3</th>
<th>Pred. 4</th>
<th>Pred. 5</th>
<th>Unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>True 1</td>
<td>487.1</td>
<td>2.90</td>
<td>0</td>
<td>0.36</td>
<td>0.90</td>
<td>0</td>
</tr>
<tr>
<td>True 2</td>
<td>3.09</td>
<td>29.0</td>
<td>0.18</td>
<td>0.18</td>
<td>0.45</td>
<td>0</td>
</tr>
<tr>
<td>True 3</td>
<td>1.72</td>
<td>0</td>
<td>1.09</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>True 4</td>
<td>2.45</td>
<td>0.09</td>
<td>0</td>
<td>6</td>
<td>0.18</td>
<td>0</td>
</tr>
<tr>
<td>True 5</td>
<td>8.73</td>
<td>0</td>
<td>0.27</td>
<td>0</td>
<td>2.54</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 11. Confusion matrix for the Page-blocks problem, balanced strategy

<table>
<thead>
<tr>
<th>Class</th>
<th>Pred. 1</th>
<th>Pred. 2</th>
<th>Pred. 3</th>
<th>Pred. 4</th>
<th>Pred. 5</th>
<th>Unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>True 1</td>
<td>452.0</td>
<td>19.36</td>
<td>3.72</td>
<td>6.54</td>
<td>9.64</td>
<td>0.09</td>
</tr>
<tr>
<td>True 2</td>
<td>1.72</td>
<td>30.27</td>
<td>0.09</td>
<td>0.36</td>
<td>0.36</td>
<td>0.09</td>
</tr>
<tr>
<td>True 3</td>
<td>0.18</td>
<td>0</td>
<td>2.54</td>
<td>0.09</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>True 4</td>
<td>0</td>
<td>0.27</td>
<td>0</td>
<td>8.27</td>
<td>0.18</td>
<td>0</td>
</tr>
<tr>
<td>True 5</td>
<td>1.27</td>
<td>0.36</td>
<td>0.18</td>
<td>0.36</td>
<td>9.36</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 4. Phoneme

Figure 5. Page-blocks

Figure 6. Satimage

Figure 7. Twonorm
In most of the cases the increasing of the subsample size leads to better classification accuracy. The dependence on the adaptation period length is not so straightforward; however for most of the problems shorter adaptation periods are more preferable. This happens because for most of the problems the population is capable of adjusting to the new subsample within 50 generations. Moreover, a shorter adaptation period gives the possibility to update the $U_i$ values more often and better distinguish the problematic areas of the feature space.

To compare the time required for computation by the original algorithm and the instance selection method, the computation time was averaged over all adaptation period lengths for every subsample size. The resulting time values were compared to the ones for the original algorithm. Next, the acceleration rate equal to $T_{orig}/T_{IS}$ was calculated. Table 12 contains the acceleration values.

The maximal acceleration value achieved is equal to 21.5, and the minimal to 3.09. For most of the problems the algorithm with adaptive instance selection had results not worse than the original method already at the point of 10-15% of the training sample being used, which means that at the same level of classification quality the instance selection allows the algorithm to work 6-12 times faster.

For better understanding of the training process with instance selection, we provide an error graph for the best individual for the overall sample and the subsample. In Figure 11 the graph for the Penbased problem with an adaptation period of 50 and subsample size of 15% is presented. In Figure 12 the graph for the Penbased problem with an adaptation period of 400 generations and subsample size of 15% is presented.

One of the characteristics of the proposed adaptive instance selection algorithm is that the subsample error rate appears to be larger than the overall training error. Nevertheless, this does not prevent a successful training process. The reason for such behaviour is that the instance selection focuses on instances which are difficult to classify (most of the time they are instances on the border between classes) and includes them into the training subset. In the first adaptation periods, the subsample error and the overall error are very similar, which means that the measurements are selected almost uniformly. But later the situation changes, because instances which are easy to classify get larger
counter values and lower probabilities to be chosen. As these instances are still present in the training sample and classified correctly, the overall training error becomes lower than the subsample error.

When increasing the adaptation period length, the training process becomes slower. In Figure 12 it is seen that at the first adaptation period the training subsample error becomes lower than the overall training set error. This happens because the population overfits to the subsample, i.e. the best rule base for the subsample describes the trends which are presented only in the subsample. During the several next adaptation periods the error on the subsample and training set is almost identical, but then the probabilities are changed so that the algorithm focuses on problematic areas of the search space, which leads to a larger subsample error. In the case of a longer adaptation period, the difference between the training set and subset errors is smaller than for the shorter adaptation period.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>Orig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magic</td>
<td>13.51</td>
<td>9.11</td>
<td>6.15</td>
<td>4.83</td>
<td>3.47</td>
<td>3.09</td>
<td>1</td>
</tr>
<tr>
<td>Page-blocks</td>
<td>13.51</td>
<td>8.90</td>
<td>6.70</td>
<td>5.41</td>
<td>4.41</td>
<td>3.60</td>
<td>1</td>
</tr>
<tr>
<td>Penbased</td>
<td>14.04</td>
<td>9.47</td>
<td>6.39</td>
<td>5.02</td>
<td>3.61</td>
<td>3.20</td>
<td>1</td>
</tr>
<tr>
<td>Phoneme</td>
<td>21.50</td>
<td>11.39</td>
<td>8.72</td>
<td>6.05</td>
<td>4.96</td>
<td>4.16</td>
<td>1</td>
</tr>
<tr>
<td>Ring</td>
<td>16.13</td>
<td>10.47</td>
<td>8.83</td>
<td>5.57</td>
<td>4.22</td>
<td>3.46</td>
<td>1</td>
</tr>
<tr>
<td>Satimage</td>
<td>17.25</td>
<td>11.62</td>
<td>8.10</td>
<td>5.73</td>
<td>4.34</td>
<td>3.50</td>
<td>1</td>
</tr>
<tr>
<td>Segment</td>
<td>19.65</td>
<td>10.65</td>
<td>7.37</td>
<td>5.56</td>
<td>4.42</td>
<td>3.72</td>
<td>1</td>
</tr>
<tr>
<td>Texture</td>
<td>17.70</td>
<td>11.09</td>
<td>7.15</td>
<td>4.82</td>
<td>3.76</td>
<td>3.15</td>
<td>1</td>
</tr>
<tr>
<td>Twonorm</td>
<td>18.54</td>
<td>12.41</td>
<td>8.86</td>
<td>6.72</td>
<td>5.13</td>
<td>4.39</td>
<td>1</td>
</tr>
</tbody>
</table>

![Figure 12](image12.png)

**Figure 12.** Accuracy change during the training process, Penbased problem, 50 generations

Let us consider the training process in the sense of different classification quality measures. The Page-blocks problem will be considered, as it is the most imbalanced problem. To calculate the quality measures such as Accuracy (overall classification accuracy), Precision, Recall, Fscore, Precision\(_M\), Recall\(_M\), Fscore\(_M\) taken from [15], the confusion matrix was calculated for every generation. In Figure 13 we provide the averaged graphs for all measures averaged over all algorithm runs.

![Figure 13](image13.png)

**Figure 13.** Accuracy change during the training process, Penbased problem, 400 generations
Vladimir Stanovov, Eugene Semenkin, Olga Semenkina

Table 12. Time acceleration values

<table>
<thead>
<tr>
<th>Dataset</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>Orig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magic</td>
<td>13.51</td>
<td>9.11</td>
<td>6.15</td>
<td>4.83</td>
<td>3.47</td>
<td>3.09</td>
</tr>
<tr>
<td>Page-blocks</td>
<td>13.51</td>
<td>8.90</td>
<td>6.70</td>
<td>5.41</td>
<td>4.41</td>
<td>3.60</td>
</tr>
<tr>
<td>Penbased</td>
<td>14.04</td>
<td>9.47</td>
<td>6.39</td>
<td>5.02</td>
<td>3.61</td>
<td>3.20</td>
</tr>
<tr>
<td>Phoneme</td>
<td>21.50</td>
<td>11.39</td>
<td>8.72</td>
<td>6.05</td>
<td>4.96</td>
<td>4.16</td>
</tr>
<tr>
<td>Ring</td>
<td>16.13</td>
<td>10.47</td>
<td>6.83</td>
<td>5.57</td>
<td>4.22</td>
<td>3.46</td>
</tr>
<tr>
<td>Satimage</td>
<td>17.25</td>
<td>11.62</td>
<td>8.10</td>
<td>5.73</td>
<td>4.34</td>
<td>3.50</td>
</tr>
<tr>
<td>Segment</td>
<td>19.65</td>
<td>10.65</td>
<td>7.37</td>
<td>5.56</td>
<td>4.42</td>
<td>3.72</td>
</tr>
<tr>
<td>Texture</td>
<td>17.70</td>
<td>11.09</td>
<td>7.15</td>
<td>4.82</td>
<td>3.76</td>
<td>3.15</td>
</tr>
<tr>
<td>Twonorm</td>
<td>18.54</td>
<td>12.41</td>
<td>8.86</td>
<td>6.72</td>
<td>5.13</td>
<td>4.39</td>
</tr>
</tbody>
</table>

For this problem the Precision<sub>M</sub>, Recall and Fscore measures are almost identical, but Precision<sub>M</sub>, Recall<sub>M</sub>, Fscore<sub>M</sub> are different. This happens due to the fact that the last measures are macro-measures and are calculated for each class separately and then averaged. In the case of imbalanced classification, the Precision<sub>M</sub> becomes larger than Accuracy, but Recall<sub>M</sub> becomes smaller. The errors for each of the five classes are presented in Figure 14.

Figure 14. The change in classification quality, Page-blocks problem, original algorithm

Figure 15. Classification errors for five classes, Page-blocks problem, original algorithm

Only the first, most represented class can be recognized correctly by the original algorithm – the error values for the remaining classes are high or very high. The next two Figures, 15 and 16, show the change in classification quality measures and classification errors for the instance selection algorithm with the balancing strategy, with the following parameters: 5% of the training sample in the subsample and an adaptation period of 50 generations.

Figure 16. The change in classification quality, Page-blocks problem, balanced instance selection

Figure 17. Classification errors for five classes, Page-blocks problem, balanced instance selection

Compared to the previous case, the solutions are now more balanced, which is especially noticeable in Figure 16. Here all classes have error rates of about 0.1 except for the fifth class, which has 0.2. Also, the Fscore<sub>M</sub> values are larger than for the original algorithm. The presented results prove that the instance selection method allows the classification quality to be increased when using the balanced strategy for imbalanced datasets.

For comparison with other methods, an additional computation experiment was performed on the same set of problems. The algorithm was tested without instance selection, but with larger resources, i.e. the population size was 210, the number of generations was 50000, and the number of rules did not change and was equal to 40. Several classification algorithms were chosen for comparison from [16, 17]. Among them there was also the prototype algorithm developed by the H. Ishibuchi
The presented approach with the balancing strategy, noted as HEFCA IS (Hybrid Evolutionary Fuzzy Classification Algorithm with Instance Selection) appeared to be the best in the sense of accuracy for 3 problems out of 9. Moreover, this algorithm has overcome not only the original method (HEFCA Orig.) on 8 problems out of 9, but also the method with 10.5 times larger resource (HEFCA Res.) on 7 problems. The prototype algorithm and its parallel implementation were outperformed on 4 problems out of 9. On the Satimage problem the results are almost identical. The parallel implementation with subsamples rotation used 10.5 times more computational resource and 7 threads.

Table 14 contains the comparison of the time required for training by the algorithm with instance selection and the Parallel fuzzy GBML [4].

On average, the presented approach is 4.6 times slower, but the parallel algorithm used 7 threads for calculation, allowing it to train up to 7 times faster. One should mention that the presented time values for instance selection are for the best configurations, and for most of the problems the best results were obtained when using 30% of the training sample. This means that the time required for training can be decreased even more without a significant loss in classification accuracy.
5 Conclusion

The proposed adaptive instance selection algorithm for imbalanced datasets creates samples of a fixed size out of the original training sample to guide the training process by taking the information about classification results into consideration. The goal of this method is to decrease the time required for computation along while increasing the quality of classification. The testing results prove that the proposed method allows a significant improvement in classification quality, especially for imbalanced datasets because of the use of the balancing strategy.

The resulting classifiers are capable of recognizing all classes with similar accuracy values, which is important for many real-world problems. The time required for training decreases significantly, and can be adjusted by changing the size of the subsample. The resulting accuracy is at the same level as the best algorithms for the presented set of problems, or even better.

The proposed adaptive instance selection method is universal and can be applied to any other iteration-based machine learning technique used for classification, including non-evolutionary algorithms.

References


**Vladimir Stanovov** received the Bachelor of Science and Master of Science degrees in systems analysis and control from Reshetnev Siberian State Aerospace University, Krasnoyarsk, Russia, in 2012 and 2014 respectively. His research interests include genetic fuzzy systems, self-configured evolutionary algorithms and machine learning. He is currently a PhD student at the Siberian State Aerospace University. Mr. Stanovov received the Best Student Paper Award from the 4th International Congress on Advanced Applied Informatics in 2015.

**Eugene Semenkin** received his Master in Applied Mathematics degree from Kemerovo State University (Kemerovo, USSR) in 1982, his PhD in Computer Science from Siberian Aerospace Academy (Krasnoyarsk, Russia) in 1995 and his DSc in Engineering from the Siberian State Aerospace University (Krasnoyarsk, Russia) in 1997. Since 1997, he has been a professor of systems analysis at the Institute of Computer Science and Telecommunications of the Siberian State Aerospace University. His areas of research include the modelling and optimization of complex systems, computational intelligence and data mining. He has been awarded the Tsiolkovsky Badge of Honour by the Russian Federal Space Agency and the Reshetnev medal by the Russian Federation.

**Olga Semenkina** received her Master in Mathematics degree from Kemerovo State University (Kemerovo, USSR) in 1982, her PhD in Computer Science from Siberian Aerospace Academy (Krasnoyarsk, Russia) in 1995 and her DSc in Engineering from the Siberian State Aerospace University (Krasnoyarsk, Russia) in 2002. Since 2003, she has been a professor of Higher Mathematics of the Siberian State Aerospace University. Her areas of research include the modelling of complex systems and discrete optimization. She has been awarded the Badge of Honoured Worker of Higher Education of the Russian Federation.
GAIN DESIGN OF QUASI-CONTINUOUS EXPONENTIAL STABILIZING CONTROLLER FOR A NONHOLONOMIC MOBILE ROBOT

Shogo Nonaka\(^1\), Takeshi Tsujimura\(^2\) and Kiyotaka Izumi\(^2\)

\(^1\)Department of Science and Advanced Technology, Saga University, Saga, Japan
e-mail: 12634014@edu.cc.saga-u.ac.jp

\(^2\)Department of Mechanical Engineering, Saga University, Saga, Japan
e-mail: \{tujimura, izumi\}@me.saga-u.ac.jp

Abstract

The control of nonholonomic canonical form using an invariant manifold is investigated to apply to a mobile robot steered by two independent driving wheels. A quasi-continuous exponential stabilizing controller is designed by using another input pattern. Additionally, the control gain designing method is proposed for this controller. Modified error system of nonholonomic double integrator model is used as nonholonomic canonical form. Generally, the gain cannot be calculated due to the non-linear transform of system. Owing to complicated relation of several parameters, the controller behavior is inconstant by gain pattern. We propose a method of designing gain which uses desired settling time. An approximate equation to obtain designed gains is derived based on the evaluation function. The design method to determine gains of the assumed actual system is simulated. The effectiveness of the proposed method is confirmed by these simulations.

Keywords: gain design, nonholonomic canonical form, exponential stabilized control, mobile robot

Wheel type robots are widely researched as nonholonomic systems \([1–3]\). Among these robots, car with two independent driving wheels is easier to deal owing to fewer states and inputs. Hence, this robot is actively researched to expect wide applications. However, these robots have the restriction of non-sideslip. The restriction makes it difficult to control nonholonomic robots with simple control technique.

Nonholonomic systems cannot be stabilized by continuous feedback \([4]\). Nonholonomic systems are transformed into canonical form of bilinear systems. The systems enable continuous feedback control by discontinuous models or by invariant manifolds \([5–9]\). Using an invariant manifold enables the system to be controlled with number of input less than state one. The methods of invariant manifolds deal with phased switching control, such as attractive to an invariant manifold based on Lyapunov stability theory and continuous feedback control \([10, 11]\), or discrete switching control after setting energy area based on an invariant manifold \([12, 13]\). In contrast, Khennouf proposes quasi-continuous exponential stabilizing control that integrates two phase control \([14, 15]\). They indicate several conditions for control design. Canonical form involves chained-form, power-form and double integrator model. We derived the unknown modified error system of the nonholonomic double integrator model, and inspected the steady-state characteristics \([16]\).
In this paper, another pattern of controller about quasi-continuous exponential stabilizing control is proposed. The controller includes a control in part of attracting state variables to an invariant manifold. Additionally, we propose the method of designing gain. The gain cannot be calculated simply owing to complicated relation of several parameter, the controller behavior is inconstant by gain pattern. We propose a method of designing gain by using desired settling time. First, eight patterns including desired values are set to remove influence of different desired values. Next, an evaluation function is set to design gain. The eight pattern simulations are evaluated by using the function with respective gain combinations. The relation of the evaluation value is verified by given initial value and gain combinations. The eight pattern simulations are formulated to derive the approximate equation for obtaining designed gain. Finally, designed gain is simulated with the proposed method. The proposed method verifies effectiveness by comparing evaluation value.

1 Nonholonomic Double Integrator Model

1.1 Kinematics of Mobile Robot

The model of a mobile robot is shown in Figure 1 as a car with two independent driving wheels. The absolute coordinate system O is set as X and Y. Position P of the robot is set the center of the axle. The distance between the left wheel and the right wheel is assigned to 2R. The wheel radius is assigned to r. The robot position is assigned to x and y. The robot rotating angle is assigned to θ. The translational velocity and angular velocity are assigned to v and ω, respectively.

![Figure 1. A nonholonomic robot platform](image)

The state variable of the mobile robot is defined as 

\[
x(t) = [x(t) y(t) \theta(t)]^T.\]

The kinematic model is formulated as

\[
\dot{x}(t) = \begin{bmatrix}
\dot{x}(t) \\
\dot{y}(t) \\
\dot{\theta}(t)
\end{bmatrix} = \begin{bmatrix}
\cos \theta(t) & 0 \\
\sin \theta(t) & 0 \\
0 & 1
\end{bmatrix}\begin{bmatrix}
v(t) \\
\omega(t)
\end{bmatrix}.
\]

The robot cannot produce speed to the axle direction. From this condition, a nonholonomic constraint is derived as follows;

\[
\dot{x}(t) \sin \theta(t) - \dot{y}(t) \cos \theta(t) = 0. \tag{2}
\]

The velocity of the robot is expressed as

\[
\dot{x}(t) \cos \theta(t) + \dot{y}(t) \sin \theta(t) = v(t). \tag{3}
\]

1.2 Nonholonomic Double Integrator Model

The conversion of non-linear state variable for x(t) is

\[
z(t) = \begin{bmatrix}
z_1(t) \\
z_2(t) \\
z_3(t)
\end{bmatrix} = \begin{bmatrix}
\theta(t) \\
x(t) \cos \theta(t) + y(t) \sin \theta(t) \\
x(t) \sin \theta(t) - y(t) \cos \theta(t)
\end{bmatrix}. \tag{4}
\]

The conversion of the state variable z(t) to double integrator model is

\[
\eta(t) = \begin{bmatrix}
\eta_1(t) \\
\eta_2(t) \\
\eta_3(t)
\end{bmatrix} = \begin{bmatrix}
z_1(t) \\
z_2(t) \\
-2z_3(t) + z_1(t)z_2(t)
\end{bmatrix}, \tag{5}
\]

where, the state variable is η(t). After η(t) is differentiated at time, robot’s input conversion is defined as

\[
u(t) = \begin{bmatrix}
u_1(t) \\
u_2(t)
\end{bmatrix} = \begin{bmatrix}
\omega(t) \\
v(t) - z_3(t)\omega(t)
\end{bmatrix}. \tag{6}
\]
From Eq. (1) to (6) we obtain
\[
\hat{\eta}(t) = \begin{bmatrix}
\hat{\eta}_1(t) \\
\hat{\eta}_2(t) \\
\hat{\eta}_3(t)
\end{bmatrix} = \begin{bmatrix}
u_1(t) \\
u_2(t) \\
\eta_1(t)u_2(t) - \eta_2(t)u_1(t)
\end{bmatrix}.
\]  

(7)

1.3 Modified Error System

The modified error system of closed-loop system is derived. If the error system is simply set, the system does not satisfy nonholonomic double integrator model. The modified desired value must be able to derive closed-loop system that satisfies nonholonomic double integrator model. Where \(x_d, y_d, \theta_d\)^T. The transformation of non-linear state variable for \(x_d\) is defined as
\[
z_d = \begin{bmatrix}
z_{d1} \\
z_{d2} \\
z_{d3}
\end{bmatrix} = \begin{bmatrix}
\theta_d \\
x_d \cos \theta_d + y_d \sin \theta_d \\
x_d \sin \theta_d - y_d \cos \theta_d
\end{bmatrix}.
\]  

(8)

The conversion of the state variable \(z_d\) to double integrator model is defined as
\[
\eta_d = \begin{bmatrix}
\eta_{d1} \\
\eta_{d2} \\
\eta_{d3}
\end{bmatrix} = \begin{bmatrix}
z_{d1} \\
z_{d2} \\
-2z_{d3} + z_{d1}z_{d2}
\end{bmatrix}.
\]  

(9)

Here, \(\eta_d\) is the state variable. Modified desired value \(\hat{\eta}_d(t)\) is defined as
\[
\hat{\eta}_d(t) = \begin{bmatrix}
\hat{\eta}_{d1} \\
\hat{\eta}_{d2} \\
\hat{\eta}_{d3}(t)
\end{bmatrix} = \begin{bmatrix}
\eta_{d1} \\
\eta_{d2} \\
\eta_{d3} + \eta_{d1}(\eta_2(t) - \eta_d) \\
\eta_{d2}(\eta_1(t) - \eta_{d1})
\end{bmatrix}.
\]  

(10)

A closed-loop system satisfying nonholonomic double integrator model is constructed. The system uses the modified target value and conversion variable \(\eta(t)\) of Eq. (5). The modified error \(e(t)\) is expressed as
\[
e(t) = \begin{bmatrix}
e_1(t) \\
\eta_1(t) - \hat{\eta}_{d1} \\
e_2(t) \\
\eta_2(t) - \hat{\eta}_{d2} \\
e_3(t) \\
\eta_3(t) - \hat{\eta}_{d3}(t)
\end{bmatrix}.
\]  

(11)

After \(e(t)\) is differentiated by time, robot’s input conversion is defined as
\[
u(t) = \begin{bmatrix}
u_1(t) \\
u_2(t) \\
\omega(t) \\
v(t) - z_3(t)\omega(t)
\end{bmatrix}.
\]  

(12)

\[
\dot{e}(t) = \begin{bmatrix}
\dot{e}_1(t) \\
\dot{e}_2(t) \\
\dot{e}_3(t)
\end{bmatrix} = \begin{bmatrix}
u_1(t) \\
u_2(t) \\
e_1(t)u_2(t) - e_2(t)u_1(t)
\end{bmatrix}.
\]  

(13)

Modified error system of nonholonomic double integrator model is derived. This satisfies canonical form.

2 Quasi-continuous Exponential Stabilizing Control

2.1 Invariant Manifold

The control scheme using invariant manifold is explained as follows. We divide three variables into two parts. An invariant manifold consists of two variables as the first group. The second group consists of remaining one variable. The variables of two groups are controlled using two inputs. The control is divided into two phases per group. The first phase forms an invariant manifold using two inputs. The second phase stabilizes using two inputs while securing an invariant manifold. The flow of the control is shown in Figure 2. The invariant manifold is derived basing on a double integrator model. A control object is \(e\) of Eq. (13). The issue of stabilization is handled to settle 0 in \(t \to \infty\) of \(e(t)\). To derive the invariant manifold, a state feedback rule of
\[
u_1(t) = -ke_1(t),
\]
\[
u_2(t) = -ke_2(t),
\]  

(14)

is applied to Eq. (13). Time response of closed-loop system is demanded as follows;
\[
e_1(t) = e_1(0)e^{-kt},
\]
\[
e_2(t) = e_2(0)e^{-kt}.
\]  

(15)

One invariant manifold candidate can be derived as follows, using constant term \(e_3(t)\).
\[
s(e) = e_3(t).
\]  

(16)

When feedback rule of Eq. (14) is applied to Eq. (13) in \(k > 0\), the time derivative of \(s(e)\) becomes as follows;
\[
\dot{s}(e) = \dot{e}_3(t) = 0.
\]  

(17)

In the feedback rule of Eq. (14), the condition of \(s(e)\) is realized as
\[
s(e) = \text{Const.},
\]  

(18)
2.2 Stabilizing Control

Quasi-continuous exponential stabilizing control is the integrated control. It consists of both controls of attraction to an invariant manifold, and control of convergence over an invariant manifold. In the control of two steps, controller that attracts to an invariant manifold can be set several control methods. However, there are several conditions to integrate these two controls. Design of the input must satisfy several conditions for the control. In the proposed method, single input is considered for attracting control to the invariant manifold instead of two inputs. Thus, input of $u_1(t)$ is set to 0. First, the control of attraction to the invariant manifold is considered. Control rule of the first step is demanded to realize $s(e) = 0$ as condition $e_1(0) \neq 0$ for the control of attraction to the invariant manifold. Lyapunov function of $s(e)$ is considered to be as follows;

$$V(e) = \frac{1}{2}s^2(e).$$

(19)

The control inputs are given as follows;

$$u_1(t) = 0,$$

$$u_2(t) = -fs(e)/e_1(t).$$

(20)

Using $f > 0$, it becomes as

$$\dot{V}(e) = s(e)s'(e)$$

$$= s(e)[e_1(t)u_2(t) - e_2(t)u_1(t)]$$

$$= -s(e)[fs(e)]$$

$$= -fs^2(e)$$

$$\leq 0.\quad (21)$$

$\dot{V}(e)$ always becomes the negative constant at any place other than $s(e) \equiv 0$. $s(e) \to 0$ is provided asymptotically at time of $t \to \infty$. Therefore, input of the controller becomes as follows;

$$u_1(t) = -ke_1(t),$$

$$u_2(t) = -fs(e)/e_1(t) - ke_2.$$

(22)

Then,

$$s(e) = \dot{e}_3(t) = e_1(t)u_2(t) - e_2(t)u_1(t)$$

$$= e_1(t)(-fs(e)/e_1(t) - ke_2(t))$$

$$- e_2(t)(-ke_1(t))$$

$$= -f s(e).\quad (23)$$
The time response is defined as
\[ s(t) = s(e(0))e^{-ft}. \] (24)

It is found that, \( s(t) \rightarrow 0 \) in \( t \rightarrow \infty \), and \( e_3(t) \) convergences asymptotic to the origin. On the other hand, it must be as
\[ \frac{|s(e)|}{e_1(t)} < \infty, \] (25)
to keep the input of Eq. (22) to finite. \( e_1(t) \) is asymptotically stable from Eq. (13) using Eq. (22). In addition, it becomes as
\[ \frac{|s(e)|}{e_1(t)} = \frac{s(e(0))e^{-ft}}{e_1(0)e^{-kt}} \leq \frac{s(e(0))}{e_1(0)} e^{-(f-k)t}, \] (26)
from Eq. (24). If it is satisfied as \( f-k > 0, \) \( s(e)/e_1(t) \) is damping exponentially in coefficient to \( f-k \). That is, \( e_3(t) \) can be converged to zero by the exponential factor of \( k \), if \( f \geq k \).

3 Gain Design

3.1 Simulation Results Based on Evaluation Function

In the derived controller, relation of gain should satisfy \( f > k \). However, the controller behavior is inconstant by the given gain. Hence, the gain must be adequately designed for desired control. Non-linearity of the system prevents us from determining proper gain easily. Thus, we investigated on the way to reduce the influence of difference of the desired values. Simulations are conducted with respect to various desired values, gains and evaluation functions. A representative evaluation value is decided regarding one combination of gains \( k \) and \( f \). The optimum gain combination concerning each evaluation function is determined so that the minimum evaluation-value can be provided by applying various gain combinations. Finally, an approximate equation has been derived from the selected gains and evaluation function.

First, an evaluation function is considered for one of desired value. Using desired settling time \( T_d \), real settling time \( T_i \) and movement distance \( L_i \), the evaluation function for each desired values is defined by following relation.
\[ d_i = L_i + |T_i - T_d|. \] (27)

The evaluation of respective desired values is \( d_i \), \( i \) is defined as desired value number. The lower the evaluation value becomes, the more desirable the control system responds. Eight sets of desired values are defined as in Table 1 and satisfied for all relations of plus and minus with three state variables. The distance between initial position and desired position is regulated as same length because evaluation criterion contains the distance. To inspect evaluation of respective combination gain, the function evaluation \( J \) is defined as follows;
\[ J = \sum_{i=1}^{8} d_i. \] (28)

Eq. (27) shows evaluation variable of desired settling time \( T_d \). Desired \( T_d \) changes to verify gain combination for minimum evaluation value of respective \( T_d \). An approximate equation that decides designed gain combination is derived using gain combination of respective \( T_d \). Gain combination is inspected. Eq (26) defined gain \( k \) and \( f \). The initial convergence of control to the invariant manifold, and it is affected by gain \( f \). Feedback control that is affected by gain \( k \), must become subsequently convergence. Hence, the gain that relates the settling time is gain \( k \) of finally convergence input. In this research, gain \( k \) is set from 0.05 to 0.85, in steps of 0.05.

Next, the combination of \( f \) which relates the respective combination \( k \) is considered. As shown in Eq. (15) and Eq. (24), the time response exponentially relates to the respective gain. The influence of \( f \) gradually becomes smaller according to the increase of \( f \). On the other hand, high gain makes instability with systems. For this reason, the upper limit value of \( f \) sets 90, that is inverse of sampling time 0.01. These conditions and prerequisite if \( f > k \) are satisfied in following relation.
\[ f = a \times 10^n, \] (29)
where, parameter \( a \) is changed from 1 to 9, then \( n \) is changed from \(-1 \) to \( 1 \). The gain, \( f \) is assigned by combining with these respective values. Table 2 shows all gain \( f \) in this case. Initial values is set to \([x \ y \ \theta] = [1.0 \ 2.0 \ \pi/2] \).

Decision of convergence satisfies all conditions, which are \(|e_1| < 0.000872, \ |e_2| < 0.05\ and |e_3| < 0.05. 0.000872[rad] is 0.05[deg]. Five desired settling times \( T_d \) are defined as \( T_d[s] = (10, 20, 30, 40, 50) \). Respective simulation time
Table 2. Gain $f$

<table>
<thead>
<tr>
<th>$a$</th>
<th>$n = -1$</th>
<th>$n = 0$</th>
<th>$n = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a = 1$</td>
<td>0.1</td>
<td>1.0</td>
<td>10.0</td>
</tr>
<tr>
<td>$a = 2$</td>
<td>0.2</td>
<td>2.0</td>
<td>20.0</td>
</tr>
<tr>
<td>$a = 3$</td>
<td>0.3</td>
<td>3.0</td>
<td>30.0</td>
</tr>
<tr>
<td>$a = 4$</td>
<td>0.4</td>
<td>4.0</td>
<td>40.0</td>
</tr>
<tr>
<td>$a = 5$</td>
<td>0.5</td>
<td>5.0</td>
<td>50.0</td>
</tr>
<tr>
<td>$a = 6$</td>
<td>0.6</td>
<td>6.0</td>
<td>60.0</td>
</tr>
<tr>
<td>$a = 7$</td>
<td>0.7</td>
<td>7.0</td>
<td>70.0</td>
</tr>
<tr>
<td>$a = 8$</td>
<td>0.8</td>
<td>8.0</td>
<td>80.0</td>
</tr>
<tr>
<td>$a = 9$</td>
<td>0.9</td>
<td>9.0</td>
<td>90.0</td>
</tr>
</tbody>
</table>

Table 3. Simulation parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Radius of wheel</th>
<th>Distance between wheel</th>
<th>Sampling interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>0.035 [m]</td>
<td>0.233 [m]</td>
<td>0.01 [s]</td>
</tr>
</tbody>
</table>

$T$, is defined as $T = 2T_d [s]$. Simulation parameters are shown in Table 3.

Figs. 3 to 7 show simulation results in 3D graphs about the relation of $k$, $f$, and evaluation value $J$ from Eq. (28). The range of $k$ obtains the range of $\pm 0.10$ from $k$ in minimum evaluation value of $J$. The axis of $f$ makes log scale, the range of $J$ is assigned per each $T_d$. Sequentially, Figure 3 shows the graph of simulation for $T_d = 10$, Figure 4 shows for $T_d = 20$, Figure 5 shows for $T_d = 30$, Figure 6 shows for $T_d = 40$ and Figure 7 shows for $T_d = 50$.

Figure 3. Evaluation value with simulation ($T_d = 10$)

Figure 4. Evaluation value with simulation ($T_d = 20$)

Figure 5. Evaluation value with simulation ($T_d = 30$)
Table 2. Gain

\[ g = 1 \quad a = 2 \quad a = 3 \quad a = 4 \quad a = 5 \quad a = 6 \quad a = 7 \quad a = 8 \quad a = 9 \quad a = 10 \]

Table 3. Simulation parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radius of wheel</td>
<td>0.035 [m]</td>
</tr>
<tr>
<td>Distance between wheel</td>
<td>0.233 [m]</td>
</tr>
<tr>
<td>Sampling interval</td>
<td>0.01 [s]</td>
</tr>
</tbody>
</table>

\( T_d \), is defined as \( T_d = 2 T_d \) [s]. Simulation parameters are shown in Table 3.

Figs. 3 to 7 show simulation results in 3D graphs about the relation of \( k, f, \) and evaluation value \( J \) from Eq. (28). The range of \( k \) obtains the range of \( \pm 0.1 \) from \( k \) in minimum evaluation value of \( J \). The axis of \( f \) makes log scale, the range of \( J \) is assigned per each \( T_d \). Sequentially, Figure 3 shows the graph of simulation for \( T_d = 10 \), Figure 4 shows for \( T_d = 20 \), Figure 5 shows for \( T_d = 30 \), Figure 6 shows for \( T_d = 40 \) and Figure 7 shows for \( T_d = 50 \).

These figures show \( k \) of each \( T_d \). The evaluation value becomes low along a certain \( k \). The evaluation value gradually becomes high with leaving the certain \( k \). On the \( k - J \) plane, the relation is shown such as a V-shaped of which bottom is located on a certain \( k \). The relationship is easier to see in Figure 5, Figure 6 and Figure 7 in a relatively large \( T_d \). Similarly, Figure 3 and Figure 4 also show the relation, however These variations are smaller than the other variations. As mentioned previously, the reason for it is believed that time response exponentially is related to gain. It should be noted that \( k \) and \( T_d \) are in inverse relationships, because, input size relates proportional to gain size as shown in Eq. (22). The gain \( k \) increases with increasing \( T_d \). In the relation of \( f, \) at minimum of \( J, \) it is about \( (f \approx 2k) \) in all \( T_d \) on the \( f-J \) plane. The gain \( f \) increases with increasing the \( J \) after \( f \approx 2k \). The value of \( J \) at \( f < 2k \) is larger.

Figures 6, 7, 8, 9, and 10. Approximate equation with desired settling time and gain \( k \) by linear equation, quadratic equation, and linear equation.

Figure 6. Evaluation value with simulation \((T_d = 40)\)

Figure 7. Evaluation value with simulation \((T_d = 50)\)

Figure 8. Approximate equation with desired settling time and gain \( k \) by linear equation.

Figure 9. Approximate equation with desired settling time and gain \( k \) by quadratic equation.

Figure 10. Approximate equation with the reciprocal of desired settling time and gain \( k \) by linear equation.
Figure 11. Approximate equation with the reciprocal of desired settling time and gain $k$ by quadratic equation.

3.2 Design of Gain $k$

The determination of the gain is considered by using the evaluation value of $J$ as shown in Figs. 3 - 7. The value of $k$ for minimum $J$ has been found to be a strong relationship with $T_d$. Therefore, a relation equation for deriving $k$ in minimum $J$ of respective $T_d$ is obtained. The relation equation of $k$ and $T_d$ is derived by using approximation of least-squares method. The approximate model uses a linear equation and a quadratic equation. The applied model is decided by referring to a standard deviation $\sigma$ and a correlation coefficient $r$. The standard deviation is indexed for dispersion of data, and can indicate the irrelevant degree from approximation. The lower the value of the standard deviation becomes, the higher the accuracy of the approximate model is. The coefficient correlation is shown as interdependence degree. The correlation is higher when the value of $r$ is closer to 1.

First, the $T_d$-$k$ graph is approximated by using linear equation as shown in Figure 8, and by using quadratic equation as shown in Figure 9. Next, the $1/T_d$-$k$ graph is approximated by using linear equation as shown in Figure 10, and by using quadratic equation as shown in Figure 11. Each approximate equation is defined as $k = aT_d^2 + bT_d + c$. These coefficients $a$, $b$, $c$ and the standard deviation $\sigma$ and the coefficient correlation $r$ are shown in Table 4. The significant figure is 4 digits. $T_d$-$k$ graphs in Figure 8 and in Figure 9 show large deviation of the data and the approximate line. In contrast, the approximate line of $1/T_d$-$k$ graph passes over the data point or passes near of data point as shown in Figure 10 and Figure 11. According to Table 4, the $\sigma$ and the $r$ prove that Figure 10 and Figure 11 are shown with higher accuracy. The difference of $\sigma$ is nearly 10 times. Next, the linear equation and the quadratic equation are compared. The $\sigma$ of the quadratic equation is smaller than the linear equation owing to high degree. However, the difference of $\sigma$ between Figure 10 and Figure 11 is small. Both correlations, $|r|$ are as high as around 1. These values are the same to four significant figures. Hence, linear equation is sufficient to use the approximate model. Accordingly, the $k$ approximate equation is defined by Figure 10 as follows;

$$k = 7.483 \frac{1}{T_d} + 3.040 \times 10^{-4} \quad (30)$$

3.3 Design of Gain $f$

Here, an approximate equation for deciding $f$ by using $k$ is determined. The approximate equation of $f$ is derived by the relationships of $k$ and $f$ in minimum $J$. $f$ is converted to $\log_{10}f$ owing the exponential relationship. The $k$-$\log_{10}f$ graphs are shown in Figure 12 in which approximation is based on a linear equation, and also in Figure 13 in which approximation is based on a quadratic equation.

Figure 12. Approximate equation with gain $f$ and gain $k$ by linear equation.
Table 4. Coefficient of approximate equation and approximate evaluation for gain $k$

<table>
<thead>
<tr>
<th>Figure name</th>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$\sigma$</th>
<th>$r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 8</td>
<td>$-1.380 \times 10^{-2}$</td>
<td>$7.560 \times 10^{-1}$</td>
<td>$3.084 \times 10^{-1}$</td>
<td>$8.989 \times 10^{-1}$</td>
<td></td>
</tr>
<tr>
<td>Figure 9</td>
<td>$5.286 \times 10^{-4}$</td>
<td>$-4.551 \times 10^{-2}$</td>
<td>$1.126$</td>
<td>$1.871 \times 10^{-1}$</td>
<td>$9.869 \times 10^{-1}$</td>
</tr>
<tr>
<td>Figure 10</td>
<td>$7.483$</td>
<td>$3.040 \times 10^{-4}$</td>
<td>$4.896 \times 10^{-2}$</td>
<td>$9.999 \times 10^{-1}$</td>
<td></td>
</tr>
<tr>
<td>Figure 11</td>
<td>$3.330$</td>
<td>$7.073$</td>
<td>$9.246 \times 10^{-3}$</td>
<td>$3.834 \times 10^{-2}$</td>
<td>$9.999 \times 10^{-1}$</td>
</tr>
</tbody>
</table>

Figure 13. Approximate equation with gain $f$ and gain $k$ by quadratic equation.

Each approximate equation is defined as $k = aT_d^2 + bT_d + c$. These coefficients $a$, $b$, $c$ and the standard deviation $\sigma$ and the correlation coefficient $r$ are shown in Table 5. The significant figure is 4 digits. In a comparison between Figure 12 and Figure 13, the approximate line of Figure 13 passes nearer than of Figure 12. The standard deviation $\sigma$ of Figure 13 is also smaller than the other one. The correlation coefficient, $r$ of Figure 13 is 0.994 which is higher than the other. Accordingly, the approximate equation of gain $f$ is defined by Figure 13 as follows:

$$\log_{10} f = -1.093k^2 + 2.131k - 6.824 \times 10^{-1}. \quad (31)$$

4 Verification of Proposed Gain Design

4.1 Gain Design

The gain is actually designed by using Eq. (30) and (31). First, $T_d$ is set to design. Each parameter is defined to decide $T_d$. In this simulation, a model of the mobile robot is made as in Table 3. The maximum velocity of the mobile robot is 0.50 [m/s]. Each distance between initial position and eight desired positions is same and about 7.2 [m]. However, actual orbit rarely becomes to the shortest distance. In this case, an orbit is assumed about 10.0 [m]. Assuming that the constant velocity is 0.35 [m/s] in distance of 10.0 [m], we have decided the settling time as 35 [s]. The reason of considering the constant velocity to 0.35 [m/s] is having the maximum velocity of the mobile robot to 0.50 [m/s]. Next, gain $f$ and $k$ are designed by using $T_d$. In this case, the significant figure is 3 digits. $k$ is calculated by using $T_d$ and Eq. (30) as

$$k = \frac{7.483}{T_d} + 3.040 \times 10^{-4}$$

$$= 0.214104$$

$$\simeq 0.214,$$

where, $f$ is calculated by using $k$ and Eq. (31) as follows;

$$\log_{10} f = -1.093k^2 + 2.131k - 6.824 \times 10^{-1}$$

$$= -0.050055028 + 0.456034 - 0.6824$$

$$= -0.276421028$$

$$f \simeq 0.529.$$

Therefore, the gains of $k$ and $f$ are decided at $T_d = 35$ [s] to 0.214 and 0.529, respectively.
Table 5. Coefficient of approximate equation and approximate evaluation for gain $f$

<table>
<thead>
<tr>
<th>Figure name</th>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$\sigma$</th>
<th>$r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 12</td>
<td>1.121</td>
<td>$-5.164 \times 10^{-1}$</td>
<td>$1.887 \times 10^{-1}$</td>
<td>$9.895 \times 10^{-1}$</td>
<td></td>
</tr>
<tr>
<td>Figure 13</td>
<td>-1.093</td>
<td>2.131</td>
<td>$-6.824 \times 10^{-1}$</td>
<td>$9.374 \times 10^{-2}$</td>
<td>$9.994 \times 10^{-1}$</td>
</tr>
</tbody>
</table>

4.2 Simulation with Designed Gain

The calculated gain is simulated. The simulation parameter is used by Table 3. The evaluation value consists of Eq. (27) and Eq. (28). The initial value is $[x\ y\ \theta] = [1.0\ 2.0\ \pi/2]$. The eight desired values are defined by Table 1. In this case, Table 6 shows the evaluation value.

Table 6. Evaluation value of simulation with the designed gain

<table>
<thead>
<tr>
<th>$T_d$</th>
<th>$k$</th>
<th>$f$</th>
<th>$J$</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>0.214</td>
<td>0.529</td>
<td>87.417</td>
</tr>
</tbody>
</table>

To verify effectiveness of the gain designing method, Table 7 shows gain combinations in which $J$ becomes the minimum value at each $T_d$.

Table 7. Evaluation minimum value of simulation each desired settling time

<table>
<thead>
<tr>
<th>$T_d$</th>
<th>$k$</th>
<th>$f$</th>
<th>$J$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.75</td>
<td>2.0</td>
<td>88.175</td>
</tr>
<tr>
<td>20</td>
<td>0.37</td>
<td>0.9</td>
<td>89.175</td>
</tr>
<tr>
<td>30</td>
<td>0.25</td>
<td>0.6</td>
<td>87.664</td>
</tr>
<tr>
<td>40</td>
<td>0.19</td>
<td>0.5</td>
<td>92.403</td>
</tr>
<tr>
<td>50</td>
<td>0.15</td>
<td>0.4</td>
<td>88.319</td>
</tr>
</tbody>
</table>

The minimum $J$ fluctuates to $92.4 \lesssim J \lesssim 87.6$ as shown in Table 7. The average is $\bar{J} \simeq 89.1$. When the values $J$ of Table 7 and Table 6 are compared, the value $J$ that is simulated by using designed gain is the lowest in the values of Table 7. Hence, the gain that is designed by this method is confirmed. As a result, the simulation is conducted under the condition that initial value $[x\ y\ \theta]$ is $[1.0\ 2.0\ \pi/2]$ and desired value $[x\ y\ \theta]$ is $[7.0\ 6.0\ \pi]$, is shown. Figure 14 shows a response of modified error $e$. Figure 15 shows the orbit of the mobile robot. Figure 16 shows a response of input $u$. Figure 17 shows the robot’s speed of translation $v$ and angular velocity $\omega$. Figure 14 shows a convergence to 0 in all the state $e$. Robot’s state is set to desired state as shown in Figure 15. The orbit is changed by desired value. In this simulation, movement distance becomes about 13.9 [m] by occurrence of turning over. In another pattern, movement distance becomes about 7.9 [m]. The average movement distance of the eight desired values is 10.9 [m]. Therefore, it is enough to estimate that the movement distance is 1.5 times more than the shortest distance. Figure 16 shows convergence of input to 0 according to the state. Figure 17 shows the outputs that are the translational velocity and the angular velocity according to the input.
Table 7 shows gain combinations in which the minimum value is the lowest in the values of Table 7. Hence, the gain that is designed by this method is confirmed.

Figure 14. Error of state with canonical form

Figure 15. Orbit of mobile robot

Figure 16. Inputs

Figure 17. Relationship between time and velocity
The maximum velocity is larger than about 0.50 [m/s] owing to the initial state value that is larger than the later state. The solution of the problem needs input saturated measures. The average velocity of the simulation is about 0.40 [m/s]. The velocity is larger than the assumed velocity owing to the movement distance that becomes longer by behavior of controller. For example, there is another simulation in which maximum velocity is about 0.80 [m/s], and in which average velocity is about 0.23 [m/s]. The average velocity of eight simulations becomes about 0.31 [m/s]. This is close to the assumed velocity 0.35 [m/s] because it is 88% of the velocity. These results prove the effectiveness of gain that is obtained by using proposed designing method.

5 Conclusions

In this paper, our approach is verified by a quasi-continuous exponential stabilizing controller using another pattern of input that is the control of attractive to the invariant manifold. A quasi-continuous exponential stabilizing controller consists of a continuous feedback control and a control of attractive an invariant manifold. We focused on the control method to attract an invariant manifold. In accordance with the method of Khennouf, the control of attractive to the invariant manifold consists of single input. The modified error system of a nonholonomic double integrator model is used in this research as a nonholonomic canonical forms. This method is expected to be applied to other nonholonomic canonical forms.

We verified the stabilization of the derived quasi-continuous exponential stabilizing controller. Furthermore, we proposed the designing method of control gain by the controller. In order to design the control gain, evaluation function for simulation at each gain is defined. The evaluation function is defined by desired settling time, real settling time and movement distance. Additionally, the eight patterns of desired values are constituted for satisfying all relations of plus and minus with three state variables to consider the influence from initial value. The approximate equation is derived from the evaluation value. Derivation of gain $k$ has been approximated by linear equation using a relationship between $1/T_d$ which is the inverse of desired settling time $T_d$. Derivation of gain $f$ is approximated by quadratic equation using a relationship between $\log_{10} f$ and $k$. Accordingly, it is possible to determine the gains $k$ and $f$ by giving the desired settling time $T_d$ with the method.

This design method that determines the gain of the assumed actual system has been simulated. The simulation was verified by using the evaluation function. The evaluation value of the simulation using designed gain with the method has become the better value than several evaluation values of simulation without the proposed method. The average movement distance of the eight desired values is 10.9 [m]. The average velocity of eight simulations becomes about 0.31 [m/s]. By contrast, assumed movement distance is 10 [m]. Assumed average velocity is 0.35 [m/s]. These relations are obtained high correlative as nearly to 90%. Therefore, the effectiveness of the proposed method has been verified by these results.

References


Shogo Nonaka, he received his B.E. and M.E. degrees from SAGA University, Japan, in 2006 and 2012, respectively. He is currently a Ph.D. student at SAGA University. His research interests include control systems design.

Takeshi Tsujimura, he received the B.S. and M.S. and the Ph.D. degrees in precision mechanical engineering from Tokyo University in 1981, 1983 and 1995, respectively. He was a senior research engineer of Access Network Service Systems Laboratories, Nippon Telegraph and Telephone Corporation, and worked as a professor of University of Tsukuba, Japan. Now, he serves at SAGA University, Japan as a professor of Department of Mechanical Engineering. He received Young Engineer Award of the Japan Society of Mechanical Engineers in 1989, Best Paper Award of SICE-ICCAS in 2006 and Best Paper Award of OPTICS/ICETE in 2014.


Kiyotaka Izumi received B.E. degree in Electronic Engineering from the Nagasaki Institute of Applied Science in 1991, M.E. degree in Electrical Engineering from the Saga University in 1993, and D.E. degree in Faculty of Engineering Systems and Technology from the Saga University in 1996. From 1996 to 2004, he was a Research Associate in the Department of Mechanical Engineering at Saga University. From 2004, he was an Associate Professor of Saga University. His research interests are: nonlinear control, softcomputing, man-machine interface, intelligent control.
GENETIC ALGORITHM COMBINED WITH A LOCAL SEARCH METHOD FOR IDENTIFYING SUSCEPTIBILITY GENES

Cheng-Hong Yang¹, Sin-Hua Moi¹, Yu-Da Lin¹, and Li-Yeh Chuang²

¹Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan
  e-mail: chyang@cc.kuas.edu.tw, moi9009@gmail.com, e0955767257@yahoo.com.tw

²Department of Chemical Engineering & Institute of Biotechnology and Chemical Engineering, I-Shou University, Kaohsiung, Taiwan
  e-mail: chuang@isu.edu.tw

Abstract

Detecting genetic association models between single nucleotide polymorphisms (SNPs) in various disease-related genes can help to understand susceptibility to disease. Statistical tools have been widely used to detect significant genetic association models, according to their related statistical values, including odds ratio (OR), chi-square test ($\chi^2$), $p$-value, etc. However, the high number of computations entailed in such operations may limit the capacity of such statistical tools to detect high-order genetic associations. In this study, we propose lsGA algorithm, a genetic algorithm based on local search method, to detect significant genetic association models amongst large numbers of SNP combinations. We used two disease models to simulate the large data sets considering the minor allele frequency (MAF), number of SNPs, and number of samples. The three-order epistasis models were evaluated by chi-square test ($\chi^2$) to evaluate the significance ($P$-value < 0.05). Analysis results showed that lsGA provided higher chi-square test values than that of GA. Simple linear regression indicated that lsGA provides a significant advantage over GA, providing the highest $\beta$ values and significant $p$-value.

Keywords: Genetic algorithms, identifying susceptibility genes, local search algorithm

1 Introduction

Single nucleotide polymorphisms (SNPs) are important biomarkers in genomes [1], and gene expression may be influenced by the SNP alone or by interaction between SNPs [2]. Thus, improved understanding of associations between SNPs contributes to the analysis of diseases and cancers [3-5]. Genetic associations indicate that the effect of any single genetic variation (e.g., SNPs) will likely be dependent on other genetic variations (interaction between SNPs) [6]. Genetic association studies focus on which SNP combinations may be associated with high risk in genes related to diseases and cancers. Thus, epistasis identification can be regarded as a feature selection problem, and genetic association detection remains a challenge in bioinformatics [7].

Genetic associations can be identified by identifying significant differences between pathological (case) and normal (control) state. Many statistical methods have been proposed to identify significant genetic associations, such as PLINK [8] and BOOST [9]. However, these methods only identify two-order genetic associations. Identification of high order genetic associations is a NP-hard prob-
lem, especially for high-dimensional SNP combinations and large SNPs [10]. Traditional statistical methods, e.g., chi-square test ($\chi^2$), are suitable for computationally intensive operations. Thus, evolutionary computations have been applied to improve statistical methods for identifying significant genetic associations. Particle swarm optimization (PSO) has been applied to identify significant genetic associations for facial emotion perception [11] and hypertension [12]. Genetic algorithm (GA) has been successfully used to identify significant genetic associations for chronic dialysis [13] and breast cancer [14]. These previous studies showed that the limitations imposed by large statistical evaluations can be overcome by evolutionary computation. Moreover, significant genetic associations with SNP combinations show that a marginal SNP may be associated with disease when combined with other SNPs [13]. However, the search abilities of these methods are insufficiently robust for large numbers of SNPs.

In this study, we used local search to improve on GA to enhance population diversity. Local search can reduce the probability of the same vector being identified between two selected chromosomes to create a crossover operation. A high-dimensional data set was simulated using the biological parameters of SNPs. The results of the improved GA outperform those of the traditional GA.

2 Method

2.1 Problem definition

An SNP represents three types of genotypes, including ‘AA’ (homozygous reference genotype), ‘Aa’ (heterozygous genotype), and ‘aa’ (homozygous variant genotype). In this study, the genotype at SNP $i$ is defined as a set $G_i = \{1, 2, 3 \mid 1=‘AA’, 2=‘Aa’, 3=‘aa’\}$, where $i$ is the $i^{th}$ SNP in $n$ SNPs ($n=$total number of SNPs) which is related to disease. Genetic association identification aims to select the $m$ SNP ($m \geq 2$) by determining whether their combination has significant associations with disease or not. A genetic association can be regarded as a set $E = \{s_1, s_2, s_3, ..., s_m\}$, where $s_i = \{SNP_i, G_i\}$ and the problem space consists of the $m$-dimensional SNP selection. The objective function $f(E)(f : \delta \subseteq RmR)$ is defined by chi-square test ($\chi^2$) and the objective $E^*$ is the set $E$ with highest $\chi^2$ value, i.e., $f(E^*) > f(E)$ for all $E \in \delta$, where $\delta$ is a non-empty large finite set serving as the problem space.

2.2 Genetic algorithm

Genetic algorithm (GA) was proposed by Holland [15] and has been applied to research in artificial intelligence, such as gene expression in biology problems. Thus, GA has been applied to the problems of classification [16] and primer design [17]. In GA, a chromosome is represented as an available solution in the search space, i.e., a genetic association set $E$. Each chromosome is evaluated by the objective function, and the good chromosomes have a higher probability to precede the evolutionary operation. Furthermore, bad chromosomes will be eliminated from the population, leaving the promising elements in the good chromosomes for the next generation. The evolutionary strategy in GA includes six operations: (1) chromosome initialization, (2) population estimation, (3) selection operation, (4) crossover operation, (5) mutation operation, and (6) replacement operation. Algorithm 1 shows the GA process.

2.3 Genetic algorithm based on the local search algorithm

The local search algorithm searches the $k$-exchange neighborhood to improve the chromosome from the current solution by exchanging at most $k$ elements [18]. Various studies have successfully applied the local search algorithm to improve evolutionary algorithms, such as multi-objective flexible job-shop scheduling problem [19], multi-modal optimization [20] and best-offspring hybrid genetic algorithm [21]. This study use the local search algorithm to enhance the population diversity after the mutation operation in GA process (lsGA). Algorithm 2 shows the lsGA pseudo-code. The detailed operations are explained in the following sections.

Chromosomal representation

The chromosomes are defined by the definition of genetic association and are shown below:

$$C_l = \{SNP_l, G_l\},$$

where $SNP_l$ is a set included the selected $m$ SNPs, where $l$ is the $l^{th}$ chromosome in the population, in
which each SNP cannot be selected repeatedly. $G_j$ is a set including the genotypes which correspond to SNP$_j$. Let $C_l = \{10, 17, 1, 2\}$, which indicates that the $l^{th}$ chromosome consists of the 'AA' genotype of SNP$_{10}$ and 'Aa' genotype of SNP$_{17}$, in which the number of the SNP is its order in the dataset.

**Objective function**

In the GA process, the objective function is used to estimate the values of the chromosomes, referred to as fitness values. The chi-square test ($\chi^2$) aims to identify the significant epistasis. The objective function can be written as:

$$F(C_l) = \frac{(a+b+c+d)(a+d-b+c)^2}{(a+b)(c+d)(a+c)(b+d)}$$

where $a$, $b$, $c$, and $d$ are respectively the four cells in the contingency table (see Table 1). The $a$ is the total number of matched $C_l$ in the cases, $b$ is the total number of matched $C_l$ in the controls, $c$ is the total number of unmatched $C_l$ in the cases, and $d$ is the total number of unmatched $C_l$ in the control. In this study, a high objective function indicates a better chromosome.

**Selection operation**

In GA, genetic operations require two parents ($P_1$ and $P_2$) to produce two children ($P'_1$ and $P'_2$), and the parents are selected by the selection operation. We used rank-based tournament selection which ranks the chromosomes according to their fitness values and selects the two top chromosomes as the parents.

**Crossover operation**

The crossover operation performed a one-point crossover that randomly generated the $D$ binary strings ($D$ is the dimension of the parent). The first string indicates that the elements of two parents $P_1$ and $P_2$ need to be exchanged, while remaining strings are unchanged. Let binary string = {1, 0, 0, 1}, $P_1 = \{1, 4, 2, 1\}$, and $P_2 = \{2, 4, 1, 3\}$, the two offsprings $P'_1$ and $P'_2$ are $\{2, 4, 2, 3\}$ and $\{1, 4, 1, 1\}$, respectively.

**Mutation operation**

The mutation operation performed the binary string mutation in which each bit in the binary string randomly generated a probability. If the probability is smaller than mutation threshold, this point in the offspring randomly generates a possible element. If the binary string remains unchanged after the mutation operation, this operation is repeatedly performed until a single bit is mutated.

**Local search algorithm**

The local search algorithm was used to find the better solution in the offspring neighborhood, and it could enhance the population diversity, especially when the production of offsprings is similar in the population. Algorithm 3 shows the pseudo-code of the local search algorithm. $P'$ indicates the offsprings and $C'$ is the neighboring offspring. $d$ is the increased distance value between $P'$ and $C'$. If the fitness value of $C'$ is better than the fitness value of $P'$, then $C'$ replaces $P'$.

**Replacement operation**

The replacement operation aims to keep the good chromosomes for genetic operations in the following generation. The two producing offspring are added into the population and the least two chromosomes with low fitness values are deleted from the population.

**Algorithm 1 – GA pseudo-code**

```
01: begin
02: Initial population
03: while (generation ≠ termination)
04: Evaluate population
05: Selection
06: Crossover
07: Mutation
08: Replacement
09: Output best chromosome
10: end
```

**Algorithm 2 – lsGA pseudo-code**

```
01: begin
02: Initial population
03: while (generation ≠ termination)
04: Evaluate population
05: Selection
06: Crossover
07: Mutation
08: Local search
09: Replacement
10: Output best chromosome
11: end
```
Algorithm 3 – Local search algorithm pseudo-code

01: begin
02: for \((i=1; i < \text{total number of offsprings}; i++)\)
03: \(\text{Copy } P'_i \text{ into } C'_i;\)
04: for \((j=1; j < \text{the dimension of } C'_i; j++)\)
05: \(d_j = l_j \times \text{Rand} [0:1]; l_j \in \mathbb{N}: l_j < \max(C_j)\)
06: \(C'_{ij} = C'_{ij} + d_j\)
07: if fitness \((P'_i) > \text{fitness } (C'_{ij})\)
08: \(\text{Replace } P'_i \text{ by } C'_{ij}\)
09: end

3 Result and Discussion

3.1 Data set

In the performance comparison, two epistasis models, ZZ model [22, 23] and XOR model [24], were selected to test all methods. The XOR model is the nonlinear epistasis, and high risk of disease is dependent on inheriting a heterozygous genotype from one locus or a heterozygous genotype from another locus, but not all loci. In the ZZ model, high risk of disease is dependent upon inheriting exactly two high-risk alleles from two loci. MAFs of disease-associated SNPs were set at 0.1 and 0.2, and MAFs of unassociated SNPs were set from [0.05, 0.5]. Total numbers of SNPs were 50 and 100, and total numbers of samples were 400 (cases = 200 and controls = 200) and 1000 (cases = 500 and controls = 500). GAMETES was used to generate the SNP dataset using the above parameters [25]. Each parameter combination generated 100 data sets in each disease model. The objective is to identify the significant genetic association models.

3.2 Parameter settings

In this study, all methods used the same parameters and the same initial population to test statistical ability to identify genetic associations. The exchange probability for the one-point selection operation is 1.0 and the exchange probability for the one-point mutation operation is 0.1. The population size is 50 and the total number of generations is 1000.

3.3 Evaluation of identified genetic association models in 12 XOR models and 12 ZZ models

In this study, each initial population between GA and lsGA is the same and the random seed in the program is also the same. Figures 1 and 2 show the results of three-order genetic association models in GA and lsGA. The symbols, upper side and lower side, in each point indicate the mean ± standard deviation (SD), and each point saves all fitness values of the population for every 50 generations over 100 data sets.

In Figure 1, the results for all generations showed that the mean best fitness values from lsGA outperform those of the traditional GA in 12 XOR models. The difference of chi-square test \(\chi^2\) values (fitness values) between GA and lsGA is very large, indicating that lsGA outperforms GA in identifying the most significant genetic association model, and the increased \(\chi^2\) values indicate that the \(p\)-value is decreased, i.e., \(p\)-value \(<< 0.05\). Both total number of SNPs and samples can influence the \(\chi^2\) values, in which the \(\chi^2\) values of large samples are higher than small samples because the \(a\) and \(d\) in Table 1 increases when the total number of samples are increased. The \(\chi^2\) values of SNP = 500 are lower than other XOR models with SNP = 50 and 100. This clearly shows that a high total number of SNPs can increase the degree of difficulty in processing the evolutionary algorithm. However, lsGA can enhance the \(\chi^2\) values, especially in XOR model with SNP = 500 and sample = 1000. This shows that the local search algorithm facilitates the finding of better solutions.

<table>
<thead>
<tr>
<th></th>
<th>Case</th>
<th>Control</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C_l)</td>
<td>a</td>
<td>b</td>
<td>(a+b)</td>
</tr>
<tr>
<td>(C_l)</td>
<td>c</td>
<td>d</td>
<td>(c+d)</td>
</tr>
<tr>
<td>Total</td>
<td>(a+c)</td>
<td>(b+d)</td>
<td>(a+b+c+d)</td>
</tr>
</tbody>
</table>

\(C_l\) indicates unmatching \(C_l\)

In Figure 2, results for all generations show that the mean of best fitness values from lsGA outperform those of GA in 12 ZZ models, and the high \(\chi^2\) values indicate that significant genetic association models are identified by GA and lsGA. However, lsGA identified more significant genetic asso-
one-point mutation operation is 0.1. The population change probability for the one-point selection operation is 0.9. Thus, the local search algorithm can avoid the population being trapped in a local optima. However, lsGA can continually enhance the \( \chi^2 \) values, especially in ZZ models with SNP = 500. This shows that the local search algorithm can avoid the population being trapped in a local optima.

Table 2 shows the average results of a simple linear regression for best fitness values and the average amount of fitness values for the population using GA and lsGA methods in the XOR and ZZ models. The positive \( \beta \) values indicate that the lsGA is superior to GA, and the high value indicates the greater improvement. The p-value \( (P > t) \) is used to determine whether lsGA significantly improves on GA in XOR and ZZ models. The lsGA shows a significant advantage as compared to the GA, providing the highest \( \beta \) values and significant p-value.

### 3.4 Comparison of GA and lsGA for population

Figures 3 and 4 show the mean sum of fitness values of the population in the form of a log10 value over the number of generation in GA and lsGA. The symbols, upper side and lower side, are the mean ± standard deviation (SD). Each point is the mean sum of the fitness values in the population over 100 data sets.

The distribution curves in Figures 3 and 4 are respectively similar with those in Figure 1 and 2, indicating the values of chromosomes are improved by the genetic operations of GA. In addition, the improvement trend in GA is relatively slow, while that in lsGA is more obvious. This indicates that the local search algorithm can provide better offspring to advance the population for finding better epistasis models in the XOR and ZZ models.

### 3.5 Effectiveness comparison of GA and lsGA

The effectiveness of the proposed lsGA is shown by computer simulations on genetic association models consisting of 12 XOR models and 12 ZZ models. The results clearly showed that lsGA can effectively escape from the local optima. Thus, the more significant genetic association models could be identified by lsGA, and these genetic association models with high risk included several SNPs which can help improve understanding of the associations between genes and disease. Several local search algorithms have been proposed to improve evolutionary algorithms in various problems, including multi-objective optimization [26], location-routing problem [27], and so on. Therefore, our proposed lsGA may be able to solve other problems. Furthermore, these local search algorithms may be more effectively in improving the search ability of GA for identifying better genetic association models.

### 3.6 Runtime comparison of GA and lsGA

The computational running time of lsGA was similar to that of GA. The local search algorithm evaluates the \( D \)-dimensional vectors in the two offsprings after the mutation operation. The computational complexity of GA can be represented as big-O(NM), where \( N \) is the total number of generations and \( M \) is the total population size. lsGA is big-O(N(M+2D)), where \( D \) is the chromosome dimension. The \( D \) is very small in the problem of identifying genetic associations, e.g., a 3-order genetic association only uses a 6-dimensional vector. However, lsGA is superior to GA in terms of finding better genetic association models with higher \( \chi^2 \) values.

### 4 Conclusions

The local search algorithm is used to improve the GA (named as lsGA) to detect genetic associations amongst disease-related genes. Two disease models are used to evaluate the ability of lsGA to detect significant genetic association models regarding the marks of SNPs located in susceptibility genes. Our results show that lsGA can detect more significant models than GA, and continued to effectively enhance the \( \chi^2 \) values for finding better mod-
Table 2. Comparison of GA and IsGA in the mean of best fitness values and in mean sum of fitness values of population in XOR and ZZ model by simple linear regression

<table>
<thead>
<tr>
<th></th>
<th>MAF = 0.1, Sample size = 400</th>
<th>MAF = 0.1, Sample size = 1000</th>
<th>MAF = 0.2, Sample size = 400</th>
<th>MAF = 0.2, Sample size = 1000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean of best fitness values</td>
<td>Mean of best fitness values</td>
<td>Mean of best fitness values</td>
<td>Mean of best fitness values</td>
</tr>
<tr>
<td></td>
<td>β</td>
<td>P &gt; t</td>
<td>β</td>
<td>P &gt; t</td>
</tr>
<tr>
<td>XOR model SNPs = 50</td>
<td>0.23 1.10E-98 0.21</td>
<td>4.58E-81 0.39</td>
<td>4.12E-154 0.16</td>
<td>4.94E-24</td>
</tr>
<tr>
<td>SNPs = 100</td>
<td>0.25 5.49E-122 0.21</td>
<td>6.04E-82 0.41</td>
<td>7.81E-171 0.33</td>
<td>7.84E-106</td>
</tr>
<tr>
<td>SNPs = 500</td>
<td>0.33 2.83E-219 0.30</td>
<td>8.80E-170 0.37</td>
<td>3.41E-137 0.34</td>
<td>6.28E-115</td>
</tr>
<tr>
<td>ZZ model SNPs = 50</td>
<td>0.29 1.79E-158 0.34</td>
<td>2.04E-223 0.35</td>
<td>5.57E-124 0.30</td>
<td>1.40E-123</td>
</tr>
<tr>
<td>SNPs = 100</td>
<td>0.26 6.21E-126 0.35</td>
<td>3.00E-245 0.27</td>
<td>1.51E-69 0.34</td>
<td>1.22E-84</td>
</tr>
<tr>
<td>SNPs = 500</td>
<td>0.13 4.06E-31 0.11</td>
<td>9.65E-24 0.19</td>
<td>1.69E-34 0.06</td>
<td>5.18E-61</td>
</tr>
<tr>
<td></td>
<td>Mean sum of fitness values of population XOR model SNPs = 50</td>
<td>Mean sum of fitness values of population XOR model SNPs = 100</td>
<td>Mean sum of fitness values of population XOR model SNPs = 500</td>
<td>Mean sum of fitness values of population XOR model SNPs = 1000</td>
</tr>
<tr>
<td></td>
<td>β</td>
<td>P &gt; t</td>
<td>β</td>
<td>P &gt; t</td>
</tr>
<tr>
<td>SNPs = 50</td>
<td>0.26 9.20E-134 0.26</td>
<td>2.32E-125 0.37</td>
<td>1.62E-138 0.24</td>
<td>9.64E-58</td>
</tr>
<tr>
<td>SNPs = 100</td>
<td>0.28 8.96E-153 0.22</td>
<td>3.48E-95 0.39</td>
<td>2.15E-149 0.35</td>
<td>5.20E-120</td>
</tr>
<tr>
<td>SNPs = 500</td>
<td>0.34 1.68E-232 0.32</td>
<td>1.08E-199 0.37</td>
<td>1.14E-134 0.34</td>
<td>8.40E-116</td>
</tr>
<tr>
<td>ZZ model SNPs = 50</td>
<td>0.32 2.69E-197 0.36</td>
<td>8.52E-261 0.35</td>
<td>8.05E-91 0.36</td>
<td>4.73E-131</td>
</tr>
<tr>
<td>SNPs = 100</td>
<td>0.27 1.40E-138 0.36</td>
<td>5.06E-260 0.29</td>
<td>9.41E-117 0.36</td>
<td>2.83E-129</td>
</tr>
<tr>
<td>SNPs = 500</td>
<td>0.18 4.93E-59 0.10</td>
<td>2.47E-18 0.25</td>
<td>2.76E-05 0.08</td>
<td>5.02E-08</td>
</tr>
</tbody>
</table>

Figure 1. Mean best fitness values in the form of a log10 value over the number of generations for GA and IsGA in the 12 XOR models. The error bar is evaluated by the standard deviation in each point.
Table 2. Comparison of GA and lsGA in the mean of best fitness values and in mean sum of fitness values of population in XOR and ZZ model by simple linear regression.

Figure 1. Mean best fitness values in the form of a log10 value over the number of generations for GA and lsGA in the 12 XOR models. The error bar is evaluated by the standard deviation in each point.

Figure 2. Mean best fitness values in the form of a log10 value over the number of generations for GA and lsGA in the 12 ZZ models. The error bar is evaluated by the standard deviation in each point.

Figure 3. Mean sum fitness values of population in the form of a log10 value over the number of generations for GA and lsGA in the 12 XOR models. The error bar is evaluated by the standard deviation in each point.
Figure 4. Mean sum fitness values of population in the form of a log$_{10}$ value over the number of generations for GA and IsGA in the 12 ZZ models. The error bar is evaluated by the standard deviation in each point.
els, indicating that lsGA can be applied to identify complex genetic association models in large data sets.

**Acknowledgment**

This study was partly supported by the National Science Council of Taiwan for Grant NSC 102-2221-E-151-024-MY3, 102-2622-E-151-003-CC3, and 102-2221-E-214-039.

**References**


Cheng-Hong Yang is a professor of the Department of Electronic Engineering at National Kaohsiung University of Applied Sciences, Taiwan. He received his M.S. and Ph.D. degrees in computer engineering from North Dakota State University in 1988 and 1992, respectively. His main areas of research are evolutionary computation, bioinformatics, and assistive tool implementation.

Sin-Hua Moi received her M.S. degree from the Department of Oral Hygiene, Kaohsiung Medical University, Taiwan, in 2014. She is currently working towards the Ph.D. degree in the Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Taiwan. She received diploma with specialization in biomedicine statistical analysis, database management and analysis. Her main areas of research are bioinformatics and biostatistics.

Yu-Da Lin received her M.S. and Ph.D. degree in the Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Taiwan, in 2009 and 2015, respectively. He has rich experiences in computer programming, database design and management, and systems programming and design. His main areas of research are bioinformatics and computational biology.

Li-Yeh Chuang is a professor and director of the Department of Chemical Engineering & Institute of Biotechnology and Chemical Engineering at I-Shou University, Kaohsiung, Taiwan. She received her M.S. degree from the Department of Chemistry at the University of North Carolina in 1989 and her Ph.D. degree from the Department of Biochemistry at North Dakota State University in 1994. Her main areas of research are bioinformatics, biochemical engineering and genetic engineering.
Journal of Artificial Intelligence and Soft Computing Research is a refereed international journal whose focus is on the latest scientific results and methods constituting artificial intelligence and soft computing, published quarterly by the Polish Neural Network Society and the University of Social Sciences in Lodz, Poland.

The recommended format is LaTeX. To ensure reasonable quality and consistency of preparation, it is important that authors adhere closely to the instructions. Any manuscript submitted in technically unsuitable form will be returned to the author for retyping. Initial submissions must be FULL papers and not merely abstracts. All submissions should be in a .zip compressed archive including: the source (input) files, i.e. TEX files for the text and jpg, PS or EPS files for the figures, style files, templates, and special fonts the authors may have used, the final DVI file (for papers prepared using LATEX or TEX) the final Adobe Acrobat (.pdf) file, the scanned completed and signed copyright form. By submitting the paper to the journal, the author(s) agree that they have not submitted this paper to any other journal or conference.

Authors are encouraged to submit complete manuscripts electronically through the journal web page.

http://jaiscr.eu

The site contains instructions on how to prepare the paper and how to proceed with submission.
<table>
<thead>
<tr>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hui Wei: A Bio-Inspired Integration Method for Object Semantic</td>
<td>137</td>
</tr>
<tr>
<td>Representation</td>
<td></td>
</tr>
<tr>
<td>Zheng Yin, Conall O'Sullivan, and Anthony Brabazon: An Analysis of</td>
<td>155</td>
</tr>
<tr>
<td>The Performance of Genetic Programming for Realised Volatility</td>
<td></td>
</tr>
<tr>
<td>Forecasting</td>
<td></td>
</tr>
<tr>
<td>Vladimir Stanovov, Olga Semenkina, and Eugene Semenkin: Self-</td>
<td>173</td>
</tr>
<tr>
<td>configuring Hybrid Evolutionary Algorithm for Fuzzy Imbalanced</td>
<td></td>
</tr>
<tr>
<td>Classification with Adaptive Instance Selection</td>
<td></td>
</tr>
<tr>
<td>Shogo Nonaka, Takeshi Tsujimura, and Kiyotaka Izumi: Gain Design of</td>
<td>189</td>
</tr>
<tr>
<td>Quasi-continuous Exponential Stabilizing Controller for a Nonholo-</td>
<td></td>
</tr>
<tr>
<td>nomic Mobile Robot</td>
<td></td>
</tr>
<tr>
<td>Cheng-Hong Yang, Sin-Hua Moi, Yu-Da Lin, and Li-Yeh Chuang: Genetic</td>
<td>203</td>
</tr>
<tr>
<td>Algorithm Combined with a Local Search Method for Identifying</td>
<td></td>
</tr>
<tr>
<td>Susceptibility Genes</td>
<td></td>
</tr>
</tbody>
</table>