The South African Institute of Computer Scientists
and
Information Technologists

Proceedings

of the

1996 National Research and
Development Conference

Industry meets Academia

Interaction Conference Centre, University of Natal,
Durban.
26 & 27 September

Edited by
Vevek Ram
FOREWORD

This book is a collection of papers presented at the National Research and Development Conference of the Institute of Computer Scientists and Information Technologists, held on 26 & 27 September, at the Interaction Conference Centre, University of Natal, Durban. The Conference was organised by the Department of Computer Science and Information Systems of The University of Natal, Pietermaritzburg.

The papers contained herein range from serious technical research to work-in-progress reports of current research to industry and commercial practice and experience. It has been a difficult task maintaining an adequate and representative spread of interests and a high standard of scholarship at the same time. Nevertheless, the conference boasts a wide range of high quality papers. The program committee decided not only to accept papers that are publishable in their present form, but also papers which reflect this potential in order to encourage young researchers and to involve practitioners from commerce and industry.

The organisers would like to thank IBM South Africa for their generous sponsorship and all the members of the organising and program committees, and the referees for making the conference a success. The organisers are indebted to the Computer Society of South Africa (Natal Chapter) for promoting the conference among its members and also to the staff and management of the Interaction Conference Centre for their contribution to the success of the conference.

On behalf of the Organising Committee
Vevek Ram
Editor and Program Chair
Pietermaritzburg, September 1996
Organising Committee

Conference General Chairs
Mr Rob Dempster and Prof Peter Warren (UNP)

Organising Chair
Dr Don Petkov (UNP)

Secretariat
Mrs Jenny Wilson

Program Chair
Prof Vevek Ram (UNP)

Program Committee
Prof Peter Wentworth, Rhodes
Dr Milan Hajek, UDW
Prof Derek Smith, UCT
Prof Anthony Krzesinski, Stellenbosch
Dr Don Petkov, UNP
Mr Rob Dempster, UNP
Prof Peter Warren, UNP
# Table of Contents

Foreword i  
Organising Committee ii  
List of Contributors vi  

**Keynote Speaker**

*The Role of Formalism in Engineering Interactive Systems*  
M D Harrison and D J Duke  

**Plenary**

*Industry-Academic-Government Cooperation to boost Technological Innovation and People Development in South Africa*  
Tjaart J Van Der Walt  

*Checklist support for ISO 9001 audits of Software Quality Management Systems*  
A J Walker  

*The IS Workers, they are a-changin'*  
Derek Smith  

**Research**

*Examination Timetabling*  
E Parkinson and P R Warren  

*Generating Compilers from Formal Semantics*  
H Venter  

*Efficient State-exploration*  
J. Geldenhuys  

*A Validation Model of the VMTP Transport Level Protocol*  
H.N. Roux and P.J.A. de Villiers  

**Intelligent Systems**

*Automated Network Management using Artificial Intelligence*  
M Watzenboeck  

*A framework for executing multiple computational intelligent programs using a computational network*  
H L Viktor and I Cloete  

*A Script-Based prototype for Dynamic Deadlock Avoidance*  
C N Blewett and G J Erwin  

*Parallelism: an effective Genetic Programming implementation on low-powered Mathematica workstations*  
H. Suleman and M. Hajek  

*Feature Extraction Preprocessors in Neural Networks for Image Recognition*  
D Moodley and V Ram
Real-Time Systems

The real-time control system model - an Holistic Approach to System Design
T Considine

Neural networks for process parameter identification and assisted controller tuning for control loops
M McLeod and VB Bajic

Reference Model for the Process Control Domain of Application
N Dhevcharran, A L Steenkamp and V Ram

Database Systems

The Pearl Algorithm as a method to extract information out of a database
J W Kruger

Theory meets Practice: Using Smith's Normalization in Complex Systems
A van der Merwe and W Labuschagne

A Comparison on Transaction Management Schemes in Multidatabase Systems
K Renaud and P Kotze

Education

Computer-based applications for engineering education
A C Hansen and P W L Lyne

Software Engineering Development Methodologies applied to Computer-Aided Instruction
R de Villiers and P Kotze

COBIE: A Cobol Integrated Environment
N Pillay

The Design and Usage of a new Southern African Information Systems Textbook
G J Erwin and C N Blewett

Teaching a first course in Compilers with a simple Compiler Construction Toolkit
G Ganchev

Teaching Turing Machines: Luxury or Necessity?
Y Velinov

Practice and Experience

Lessons learnt from using C++ and the Object Oriented Approach to Software Development
R Mazhindu-Shumba

Parallel hierarchical algorithm for identification of large-scale industrial systems
B Jankovic and VB Bajic
<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Information Technology and Organizational Issues</strong></td>
<td></td>
</tr>
<tr>
<td><em>A cultural perspective on IT/End user relationships</em></td>
<td>243</td>
</tr>
<tr>
<td>A C Leonard</td>
<td></td>
</tr>
<tr>
<td><em>Information Security Management: The Second Generation</em></td>
<td>257</td>
</tr>
<tr>
<td>R Von Solms</td>
<td></td>
</tr>
<tr>
<td><em>Project Management in Practice</em></td>
<td>267</td>
</tr>
<tr>
<td>M le Roux</td>
<td></td>
</tr>
<tr>
<td><em>A Case-Study of Internet Publishing</em></td>
<td>271</td>
</tr>
<tr>
<td>A Morris</td>
<td></td>
</tr>
<tr>
<td><em>The Role of IT in Business Process Reengineering</em></td>
<td>285</td>
</tr>
<tr>
<td>C Blewett, J Cansfield and L Gibson</td>
<td></td>
</tr>
<tr>
<td><strong>Abstracts</strong></td>
<td></td>
</tr>
<tr>
<td><em>On Total Systems Intervention as a Systemic Framework for the...</em></td>
<td>299</td>
</tr>
<tr>
<td>D Petkov and O Petkova</td>
<td></td>
</tr>
<tr>
<td><em>Modular Neural Networks Subroutines for Knowledge Extraction</em></td>
<td>300</td>
</tr>
<tr>
<td>A Vahed and I Cloete</td>
<td></td>
</tr>
<tr>
<td><em>Low-Cost Medical Records System: A Model</em></td>
<td>301</td>
</tr>
<tr>
<td>O A Dami and T Seipone</td>
<td></td>
</tr>
<tr>
<td><em>A Methodology for Integrating Legacy Systems with the Client/Server Environment</em></td>
<td>302</td>
</tr>
<tr>
<td>M Redelinghuys and A L Steenkamp</td>
<td></td>
</tr>
<tr>
<td><em>Information Systems Outsourcing and Organisational Structure</em></td>
<td>303</td>
</tr>
<tr>
<td>M Hart and Kvavatzandis</td>
<td></td>
</tr>
<tr>
<td><em>The relational organisation model</em></td>
<td>304</td>
</tr>
<tr>
<td>B Laauwen</td>
<td></td>
</tr>
<tr>
<td><em>The Practical Application of a New Class of Non-Linear Smoothers for...</em></td>
<td>305</td>
</tr>
<tr>
<td>E Cloete</td>
<td></td>
</tr>
<tr>
<td><em>A Technology Reference Model for Client/Server Software Development</em></td>
<td>306</td>
</tr>
<tr>
<td>R C Nienaber</td>
<td></td>
</tr>
<tr>
<td><em>The Feasibility Problem in the Simplex Algorithm</em></td>
<td>307</td>
</tr>
<tr>
<td>T G Scott, J M Hattingh and T Steyn</td>
<td></td>
</tr>
<tr>
<td><strong>Author Index</strong></td>
<td>309</td>
</tr>
</tbody>
</table>
List of Contributors

Vladimir B Bajic  
Centre for Engineering Research, 
Technikon Natal, 
P O Box 953 
Durban 4000

Ruth de Villiers  
Department of Computer Science and 
Information Systems 
UNISA 
Box 392, Pretoria, 0001

C N Blewett  
Department of Accounting 
University of Natal 
King George V Avenue 
Durban 4001

G J Erwin  
Business Information Systems 
University of Durban-Westville 
Private Bag X54001 
Durban 4000

Justin Cansfield  
Department of Accounting 
University of Natal 
King George V Avenue 
Durban 4001

G Ganchev  
Computer Science Department 
University of Botswana 
PBag 0022 
Gaberone, Botswana

Tom Considine  
Apron Services (Pty) Ltd 
P O Johannesburg 
International Airport 
1600

J Geldenhuys  
Department of Computer Science 
University of Stellenbosch 
Stellenbosch 
7700

Eric Cloete  
School of Electrical Engineering 
Cape Technikon 
Box 652 
Cape Town

Louise Gibson  
BIS, Dept Accounting & Finance 
University of Durban 
Pvt Bag X10 
Dalbridge 4014

I Cloete  
Computer Science Department 
University of Stellenbosch 
Stellenbosch 
7600

Mike Hart  
Department of Information Systems 
University of Cape Town 
Rondebosch 
7700

O A Daini  
Department of Computer Science 
University of Botswana 
Gaborone 
Botswana

M. Hajek  
Department of Computer Science 
University of Durban-Westville 
Pvt Bag X54001 
Durban 4000

Nirvani Devcharan  
Umgeni Water 
Box 9 
Pietermaritzburg 
3200

A C Hansen  
Dept of Agricultural Engineering 
University of Natal 
Private Bag X01 
Scottsville 3209

P J A de Villiers  
Department of Computer Science 
University of Stellenbosch 
Stellenbosch 
7700

J M Hattingh  
Department of Computer Science 
Potchefstroom University for CHE 
Potchefstroom 2520
Boris Jankovic  
Centre for Engineering Research  
Technikon Natal  
P O Box 953,  
Durban 4000  

Paula Kotze  
Department of Computer Science and Information Systems  
UNISA  
Box 392  
Pretoria, 0001  

J W Kruger  
Vista University  
Soweto Campus  
Box 359  
Westhoven 2124  

A C Leonard  
Dept of Informatics  
University of Pretoria  
Pretoria, 2000  

Ben Laauwen  
Laauwen and Associates  
P O Box 13773  
Sinoville  
0129  

Mari Le Roux  
Information technology, development: project leader  
Telkom IT 1015  
Box 2753  
Pretoria 0001  

P W L Lyne  
Dept of Agricultural Engineering  
University of Natal  
Private Bag X01  
Scottsville 3209  

Rose Mazhindu-Shumba  
Computer Science Department  
University of Zimbabwe  
Box MP167  
Harare, Zimbabwe  

Meredith McLeod  
Centre for Engineering Research, Technikon Natal,  
P O Box 953  
Durban 4000  

D Moodley  
Computer Management Systems  
Box 451  
Umhlanga Rocks  
4320  

Andrew Morris  
P O Box 34200  
Rhodes Gift  
7707  

R C Nienaber  
Technikon Pretoria  
Dept of Information Technology  
Private Bag X680  
Pretoria 0001  

E Parkinson  
Department of Computer Science  
University of Port Elizabeth  
Box 1600  
Port Elizabeth 6000  

Don Petkov  
Department of Computer Science and Information Systems  
University of Natal  
PBag x01  
Scottsville 3209  

Olga Petkov  
Technikon Natal  
Box 11078  
Dorpspruit 3206  
Pietermaritzburg  

N Pillay  
Technikon Natal  
Box 11078  
Dorpspruit 3206  
Pietermaritzburg
V Ram
Department of Computer Science and Information Systems
University of Natal
PBag x01
Scottsville 3209

H. Suleman
Department of Computer Science
University of Durban-Westville
Pvt Bag X54001
Durban 4000

Melinda Redelinghuys
Department of Computer Science and Information Systems
UNISA
Box 392
Pretoria, 0001

A Vahed
Department of Computer Science
University of Western Cape
Private Bag X17
Bellville 7530

Karen Renaud
Computer Science and Information Systems
UNISA
Box 392
Pretoria, 0001

A Van der Merwe
Computer science and Information Systems
UNISA
P O Box 392
Pretoria, 0001

H N Roux
Department of Computer Science
University of Stellenbosch
Stellenbosch
7700

Tjaart J Van Der Walt
Foundation for Research and Development
Box 2600
Pretoria, 0001

T G Scott
Department of Computer Science
Potchefstroom University for CHE
Potchefstroom
2520

K Vavatzandis
Department of Information Systems
University of Cape Town
Rondebosch
7700

T Seipone
Department of Computer Science
University of Botswana
Gaborone
Botswana

Y Velinov
Dept Computer Science
University of Natal
Private Bag X01
Scottsville 3209

Derek Smith
Department of Information Systems
University of Cape Town
Rondebosch
7700

H Venter
Department of Computer Science
University of Port Elizabeth
Box 1600
Port Elizabeth 6000

Anette L Steenkamp
Department of Computer Science and Information Systems
UNISA
Box 392
Pretoria, 0001

H L Viktor
Computer Science Department
University of Stellenbosch
Stellenbosch
7600

T Steyn
Department of Computer Science
Potchefstroom University for CHE
Potchefstroom 2520

R Von Solms
Department of Information Technology
Port Elizabeth Technikon
Private Bag X6011
Port Elizabeth 6000
PARALLELISM: AN EFFECTIVE GENETIC PROGRAMMING IMPLEMENTATION ON LOW-POWERED MATHEMATICA WORKSTATIONS

H. Suleman, M. Hajek
Department of Computer Science
University of Durban-Westville
Private Bag X54001
Durban 4000
hsuleman@pixie.udw.ac.za
mhajek@pixie.udw.ac.za

Abstract

Mathematica has proven itself to be a suitable platform on which to develop prototype Genetic Programming applications. However, due to the sheer complexity of genetic programming calculations, non-trivial problems cannot be solved on a single Mathematica workstation. A distributed algorithm is suggested to eliminate such restrictions on the problem domain. A client-server network is utilised to model a system of multiple populations under simultaneous evolution. Regular migration of members, through the medium of the server, results in suitable genetic material being filtered through to other populations. This multi-population model is contrasted with the single-population standard approach in terms of its performance and utility value.

Introduction

Genetic Programming is a generalisation of the standard genetic algorithm. It falls under the broad category of Evolutionary Programming, including both crossover and mutation in its genetic operations set.

Mathematica is proposed as an alternative language of implementation for prototyping Genetic Programming (GP) algorithms. Historically, a large number of GP implementations are done in LISP, with a fewer number in other 3GL languages for greater speed and better data management. Mathematica is an ideal option because of the manner in which data and code can be freely interchanged; this being the case in LISP as well. Since it is essentially a mathematical modelling language, it performs well in problem domains that are mathematically orientated. Such an implementation can also be optimised to take advantage of the specific features of Mathematica, especially its multi-paradigm approach to programming, encompassing all of procedural, declarative and functional techniques.

Evaluation of Implementation

Any implementation of a GP, but especially one in Mathematica, is likely to encounter the problem of excessive execution times. For a non-trivial problem, the parameters cannot be biased towards a known solution, making the lengthy calculations unavoidable.

A time-analysis of a typical execution of the algorithm indicates that the time taken to perform the genetic operations is generally proportionate to the size of the population. However, the time taken to evaluate fitnesses increases as the complexity of the expressions increases. Since the fitness calculations take up a large portion of the time, it may help if these were done in parallel on multiple processors.

Another important factor to be considered is the level of variety in the population. Initially, the size of population is chosen so that sufficient variety is seeded into the first generation. However, as crossover proceeds, many characteristics are lost to subsequent generations. Mutation is introduced to recover these lost attributes. However, mutation is not an ideal solution since it seeks to restore lost genetic material, rather than protect existing potential solutions. Running a GP several times can produce highly contrasting results since the genes considered are different in each run. Thus, repeating a GP run several times is one possible way of ensuring that the final solution found is as near as possible to
the optimal solution. Another option would be to split a large population into sub-populations and execute the GP on these.

Many problem domains have only highly complex solutions. In order to generate such solutions using a GP, a large amount of memory is required. A typical problem to generate a parser for a high-level language requires that the function set spanning a complete language be used as an input parameter. To adequately cater for this variety, the population of solutions needs to be very large. Such problems demand either stringent hardware configurations or innovative approaches to memory management. One innovation would be to split up the population and process each segment separately in sequence, or in parallel if multiple processors are available.

It is clear from the above discussion that many problems and shortcomings of the standard GP can be overcome by using multiple populations evaluated in parallel on multiple processors.

**Parallel Processing Model**

Many models exist to transform the standard genetic programming algorithm into a parallel one. Some models consider the population to be singular and break up the work to be done at the lower level of the individual operations or calculations. This is obviously more suited where a parallel or distributed operating system is available. In other cases, the population needs to be split up into sub-populations to facilitate parallelism, as was done for this study.

The most important consideration after dividing the population is whether to utilise discrete or continuous population evaluations. Continuous evaluations provide a better model of reality but incur the added expense of greater inter-population communication. Discrete evaluations are an attempt to cut down on inter-population communication.

Where a single population was evaluated for the standard GP, multiple populations will have to be evaluated for the parallel algorithm. These populations each evolve through one generation in a single cycle of the algorithm. Thereafter, a global solution can be sought, and a new cycle may begin. Just searching for global solutions among the various sub-populations would reduce the problem to a multiple-run equivalent. Instead a mechanism to bind the sub-populations together must be introduced. This would imply members of one sub-population interacting with members of another sub-population through either reproduction or crossover. This solution has the disadvantage that genetic operations have to be performed on multiple sub-populations simultaneously, thereby reducing the ability to process the populations in parallel.

Migration of members is an attempt to achieve both these goals without the need for distinct operations on the data. Analogous to migration of people, members of a population may migrate from one population to another. This migration is fitness-proportionate to ensure that the important genetic material of a sub-population has a higher probability to migrate. Since migration does not involve any genetic operations, it can be done in between the discrete evolution of populations.

Migration introduces the added problem of selecting criteria for the movement of members. If all members were allowed to migrate to any sub-population, the problem of losing genetic material would not be solved, as sub-populations with very fit individuals would very soon influence all other sub-populations. In order to preserve the genetic material of any sub-population, migration must be restricted to a subset of the set of sub-populations.

To restrict migration, it is useful to map the set of sub-populations onto a mathematical surface, introducing a specific spatial distribution into the algorithm's parameters. If all the sub-populations were arranged in a straight line, each sub-population would only allow migration with its immediate neighbours on either side. Restrictions could also be set to dictate the probability with which migration will occur between further sub-populations. In this study, migrations were limited to immediate neighbours to reduce the inter-population communication during a migration operation.

If the surface used was a circle, migration would also be possible between the first and last sub-populations, which would result in all sub-populations having similar behaviour. Such linear arrangements will result in clusters of sub-populations with common sub-expressions occurring at various points along the surface.

Generalisations of this technique may involve adding more dimensions to the surface. The experiments in this study used a two-dimensional surface (Figure 1). Sub-populations were distributed to a square grid on the surface, with those at the ends and corners wrapping around to the opposite end. Thus, each sub-population can effectively allow migration between itself and eight other sub-populations. This would give rise to regions with common characteristics in the plane of the surface.

This entire sub-population system is analogous to reality as it depicts the manner in which people live in separate communities and then move among these communities.
Distributed Implementation

The parallel GP algorithm, as outlined above, was implemented in Mathematica for MS-DOS using a Windows for WorkGroups Network Operating System (NOS). Mathematica was retained as the language of choice since it allowed painless modelling of new problems. The NOS chosen was used simply because of availability and not because of any assumed superiority.

Instead of using programs that communicate directly over the network, precedence was given to code that communicated using primitive file-based data exchanges. Since all communication was through the opening, closing and deleting of files, the system would be independent of any particular NOS.

A common dilemma when splitting up the population, is whether the population data should reside on a single computer or on multiple computers. If the population data resides on a single computer, there will be extra communication along the network. If the population data resides on multiple machines, those machines will dictate the number of sub-populations and those machines will have to be present for the full duration of the calculations to prevent drastic scenario changes. The single machine option was chosen to keep the number of processors flexible throughout the lifespan of the calculations. All processors would mount the shared directory on the "server" in order to use the common population store.

Many initial attempts at parallelizing a single-computer GP were abandoned due to problems with the particular platform chosen. These techniques may nonetheless be used in future implementations.

The first algorithm used file sharing and locking to ensure that two processors did not attempt to process the same population. Initially, a set of dummy "flag" files were created, one corresponding to each population. Processors would then lock a flag file, process the relevant population and delete the flag file. The file locking would pre-empt the possibility of any other processor corrupting or duplicating the calculation. After all the flag files were deleted, one of the processors would perform migration, using a migration flag, and then generate a set of new flag files. Each processor selected the population it would process by searching through the flag files for one that wasn't locked; the process was dynamic and depended solely on the processors. This method did not work because the OS locking mechanism failed when heavily loaded.

As an alternative, deleting files and searching for files was used instead of file locking. This also failed since the indivisibility of such operations was not guaranteed.

Eventually, it was decided that the processors could not themselves decide which populations to process. A static scheduler was introduced to submit jobs to the individual processors. This scheduler would then take over the role of creating the flag files in pre-specified directories. The individual processors would search their specific directories for flags. When one was found they would process the relevant population and then delete the flag file. The scheduler would constantly check for such deletions and immediately schedule another job to that processor. After all the population evolution is done, the scheduler submits a "Migrate" job to one of the processors and thereafter begins the cycle over again.
Analysis of Implementation

A series of tests was run on the parallel Mathematica implementation to determine its effectiveness compared to the standard single-population approach. All tests were run on a network of 486-DX33 computers with one acting as the server and the other being clients. The server contained the population data and ran the scheduler while the clients ran the various Mathematica population processing programs. The parameters (Table 1) for all calculations was identical except for a variance in the number of clients used.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crossover Probability</td>
<td>90%</td>
</tr>
<tr>
<td>Mutation Probability</td>
<td>10%</td>
</tr>
<tr>
<td>No Of Sub-populations</td>
<td>9</td>
</tr>
<tr>
<td>Total Population Size</td>
<td>450</td>
</tr>
<tr>
<td>Minimum Solution Fitness</td>
<td>100%</td>
</tr>
<tr>
<td>Maximum Initial Size</td>
<td>5</td>
</tr>
<tr>
<td>Maximum Size</td>
<td>17</td>
</tr>
<tr>
<td>Maximum Complexity</td>
<td>50</td>
</tr>
<tr>
<td>Migration Probability</td>
<td>25%</td>
</tr>
<tr>
<td>Migration Percentage</td>
<td>10%</td>
</tr>
<tr>
<td>Migration Std. Deviation</td>
<td>5%</td>
</tr>
</tbody>
</table>

"Total Population Size" refers to the sum of the sizes of the sub-populations. "Maximum Initial Size" is the maximum depth that an expression can take on at generation 0. "Maximum Size" is the maximum depth that an expression can reach as a result of genetic operations. "Maximum Complexity" is the maximum number of nodes that an expression may contain. "Migration Probability" is the probability with which migration occurs after between two sub-populations. "Migration Percentage" is the average number of individuals that are exchanged during a migration operation. "Migration Std. Deviation" controls the amount by which the Migration Percentage may vary.

The problem was a simple symbolic regression one, where a curve of unspecified form was to be fitted through a set of 10 sample points. The equation used to generate the sample data was:

\[ f(x) = x^4 + x^3 + x^2 + x \]

The function set used was \{+(3), +(2), *(3), *(2), -(1), /2\} with the associated arity in brackets, to speed up convergence. Using only functions with arity 2 in prior experiments, the algorithm was shown to converge but at a much slower rate. The terminal set contained only the unknown variable. Every run of the algorithm produced a perfect solution. The rather large population size, relative to this problem, was chosen so that the efficiency of the algorithm could be measured instead of the ability to find a solution; previous experiments on a single population had confirmed that GP could resolve this problem for smaller populations.

The time taken was as follows:

<table>
<thead>
<tr>
<th>Table 2 : Benchmark times</th>
</tr>
</thead>
<tbody>
<tr>
<td>sec.</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>1 client</td>
</tr>
<tr>
<td>3 client</td>
</tr>
<tr>
<td>9 client</td>
</tr>
</tbody>
</table>

Each experiment was repeated five times and a mean of these was calculated. The second column refers to the average time taken for the algorithm to reach the perfect solution. The third column is the average time taken to produce a new generation by evolution, followed by the minimum and maximum times to produce a new generation.

The experiment was also repeated using a single population to check the net effect of using the parallel approach.
It is apparent that the time taken to generate a new full population decreases as the number of clients increases. However, this increase is not proportional to the increase rate of the number of clients. This is due to the fact that the system is affected by many other factors than the number of clients.

As the number of clients increase, so does the load on the server and the amount of network communication. Thus, more clients do not necessarily increase the efficiency of the network. Migration was done on a single computer. Thus the entire system would come to a halt while the migration operation was being carried out. The rationale behind this was that migration would take far less time than evaluation of fitnesses so the latter had greater priority to be distributed. However, when a large number of clients were used, the time taken to generate new populations became less than that to migrate. Thus, migration affected the algorithm more as the number of processors was increased.

Since Mathematica eventually ran out of memory when doing long sequences of calculations, it was restarted before each population evolution cycle, adding to the overhead of the algorithm. When compared to the single-population model, all versions of the multi-population still take longer to find the solution. However, these are all attributable to the specifics of the implementation. As a result of parallelism, the algorithm can now be applied to problems of arbitrary size and complexity, without fear of overrunning the resources of the computer. Also, one iteration of the algorithm would now find solutions that previously required multiple runs to maintain variety of genetic material.

Conclusion

Genetic Programming is a powerful technique to solve problems with no explicit solution methodology. Two of its shortcomings are execution time and loss of genetic material, which can be addressed by using a parallel algorithm. Parallel algorithms, although incurring the overheads of inter-process communications, allow large problems to be solved irrespective of the size and complexity of the solution or the computational power of the machines.

The particular implementation on Mathematica can be improved by parallelizing the migration operation. In addition, the server/network can be changed to enable faster communication. With just these two improvements, the parallel algorithm will definitely surpass the standard approach in even the category of evaluation time.

References
