



**“Foundations of Genetic Programming”,  
by William B. Langdon and Riccardo Poli  
Published by Springer, 2002.**

Reviewed by *Steve Phelps*,  
Department of Computer Science,  
The University of Liverpool

Is genetic programming (GP) better than random search? Does the GP cross-over operator actually do anything genetically-meaningful, or is it just a fancy mutation operator? What causes the phenomenon of "bloat"- the tendency of GP solutions to grow rapidly in length without improving in fitness?

Anybody who has experimented with GP will have asked themselves these questions at one time or another, and turned to the burgeoning literature on the theoretical foundations of GP only to discover that there are as many conflicting answers as there are conflicting theories. Langdon and Poli take on the ambitious task of giving a unified overview of a field still in its infancy, and the result is an invaluable companion to the literature.

The book assumes very little prerequisites of the reader beyond some high-school mathematics, but nevertheless proceeds to give a comprehensive and illuminating treatment of the most important theorems. The authors never fall into the trap of providing dry proofs without any context and leaving it up to the reader to develop an intuitive understanding on their own; throughout the book the formal side of the theory is developed alongside intuitive explanations and constructive analysis of actual empirical data.

The book starts off gently by setting Genetic Programming in the wider context of search-problems in general, introducing the concept of a fitness-landscape and different strategies for searching it. Just when traditional AI-ers are starting to feel cosy and at home, the authors explain that the fitness-landscape metaphor can be misleading when thinking about how genetic-search operates. This is because it fails to capture the fact that *building-blocks* can be important in solving a problem; when designing a house one does not simply optimise each parameter of the design until one has arrived at the best house in a house search-space, rather the design uses a combination of building-blocks, or components, e.g. bricks, roof, etc., which themselves may have been discovered through evolutionary search.

Much research on the theoretical basis of GP has grown out of John Holland's original schema theorem for genetic algorithms (GA), which attempted to explain the efficacy of genetic search by appealing to this notion of building blocks, or *schemata*, and showing that genetic search will tend to search parts of the search-space that contain promising schemata. Many attempts to analyse the efficacy (or otherwise) of GP have centred around "porting" Holland's schema theorem from the world of fixed-length bit-string genomes to the variable-length tree genomes of GP, and the core of the book is devoted to an analysis of various schema theories, which attempt to model the dynamics of GP.

This treatment starts off with an introduction to Price's theorem, which lies at the heart of any theoretical analysis of genetic populations in both biology and in simulated evolution, and uses this as a basis to explain GA schema theory. GA-schema theory is then generalised to GP with an exposition of the work of Koza, Altenburg, O'Reilly and Whigham, on what the authors refer to as component-based schema theories.

The authors' real interest, however, lies with more modern approaches which view schemata as subsets of the search-space, rather than components of the genome. They briefly review the work of Rosca before proceeding to an overview of their own work, in which they use non-"standard" GP operators (1-point

crossover) designed to make the dynamics more transparent, and are able to provide a lower-bound on the expected number of instances of a given schema in the next generation.

Attention is then turned to so-called "Exact" Schema Theorems, which take into account schema-creation and are able to calculate the probability distribution for the number of matches of a given schema, rather than simply the expected value. The work of Stephens and Waelbroeck on exact schema theories of GAs is reviewed, and the authors show how this can be applied to GP.

The success of a theory can only be measured in its ability to illuminate our understanding of the domain in question, and the authors do not neglect to apply all of this theorising back to real-world questions. An entire chapter is dedicated to applying schema theory in order to understand issues such as effective fitness landscapes (which can be used to explain how attractors can exist in the search-space even with a constant fitness function), operator biases (e.g. does use of a particular operator favour some solutions over others?), whether building-blocks actually exist in GP, and the use of theory to guide the design of new GP strategies.

The remainder of the book deals mainly with an analysis of the performance of genetic-programming compared with other search methods, using a combination of theoretical reasoning and empirical investigation. Chapters seven and eight consider the space of all possible programs, and the distribution of high-fitness solutions within this space as a function of program size. This forms the basis of a comparison between genetic-programming and random search. Chapters nine and ten look at two common benchmark problems for GP: the Santa Fe ant trail and the MAX problem and show how theory can illustrate why GP performs relatively poorly compared to other search techniques on these problems. The book concludes with a section on convergence and bloat and demonstrates that theory offers new insights into both, and how the two might be interrelated.

I came to this book from an engineering perspective as a GP practitioner interested in

practical issues such as which cross-over operator was most applicable for my problem. Whilst this book did not offer any clear-cut answers, this is a reflection of the fact that there are no clear-cut answers, yet. What the book does succeed in doing is providing an illuminating overview of the body of work which will, in time, come to provide a theoretical foundation, and accurate prescriptions, for all of the ad-hoc tweaks and adjustments that we make in practise.

---

**"NETLAB: Algorithms for Pattern Recognition", by Ian T. Nabney, published by Springer in 'Advances in Pattern Recognition' series, 2001**

Reviewed by *Lu Zhang*,  
Department of Computer Science,  
The University of Liverpool

After several decades' research, there have been some widely accepted methods and algorithms developed for typical artificial intelligence problems. However, practitioners have to implement those existing algorithms again and again for their own needs in most cases. The NETLAB software package (accessible at <http://www.ncrg.aston.ac.uk/netlab>) is developed for facilitating the use of these algorithms, especially neural network algorithms. In this book, Ian Nabney provides a well-organised description of NETLAB along with plenty of demonstration programs, worked examples and exercises. The book is divided into ten chapters. The first chapter is an introduction of the whole book. As NETLAB is based on a well-known software tool called MATLAB, there is firstly a brief introduction to MATLAB. After this, the author presents an overview of NETLAB. To comply with the overall style of the book, this chapter also includes a simple worked example.

Before going into central neural network algorithms, the author uses two chapters to illustrate some basic pattern recognition algorithms. These algorithms can be used as means of training neural networks, and as independently applicable routines as well. Chapter 2 provides some general optimisation algorithms. Here, an optimisation problem is

ENGIN MATHS

Autumn 2002  
Vol 5, No 3  
ISSN 1465-4091  
£12.50

# EXPERT UPDATE

The Specialist Group on  
Artificial Intelligence



UCL LIBRARY SERVICES

 **UCL**

UCL Library  
Services

UCL Science  
Library

**ECCAI**

 **BCS**

SPECIAL ISSUE

Intelligent Services for The  
Knowledge Lifecycle

STORE Pers