Proceedings of the Institution of Civil Engineers
Maritime Engineering 163
September 2010 Issue MA3
Pages 135–136
doi: 10.1680/maen.2010.163.3.135

Paper 900003

Paper published: *Proc. Instn Civ. Engrs – Marit. Engng*, 2007, **160**, Sept., 113–119

Discussion: Neural network – genetic programming for sediment transport

A. K. Singh, M. C. Deo and V. Sanil Kumar

A. H. Gandomi, *Tafresh University, Tafresh, Iran*

Descriptions given in Section 2 and Figures 2, 3 and 4 of the paper studied by the authors clearly indicate that the method utilised for estimating the sediment rate is a tree-based genetic programming (TGP) approach. TGP was introduced by Koza (1992) as an extension of the genetic algorithms (GAs), in which programs are represented as tree structures and expressed in the functional programming language, LISP (Koza, 1992).

According to Section 5 of the paper, the software package Discipulus, which was developed by Conrads *et al.* (1998), was applied to the sediment rate forecasting problem.

Utilisation of Discipulus software indicates that the approach employed by the authors is machine-code-based, linear genetic programming (LGP) (Deschaine and Francone, 2002; Francone and Deschaine, 2004; Francone *et al.*, 2005; Langdon and Banzhaf, 2005) not TGP. LGP (Brameier and Banzhaf, 2007) is a subset of GP that has recently emerged. Comparing LGP with the traditional Koza's TGP, there are some main differences. LGPs have graph-based functional structures and evolve in an imperative programming language (such as C/C++) (Brameier *et al.*, 1998) and machine code (Nordin, 1994) rather than in expressions of a functional programming language such as LISP (see Figure 10). Unlike TGP, structurally non-effective codes coexist with effective codes in LGPs (Brameier and Banzhaf, 2007). Owing to the imperative program structure in LGP, the non-effective instructions can be identified efficiently. As only effective programs are executed, evaluation can be accelerated significantly. In addition, the machine-code-based, LGP approach searches for the computer program and the constants at the same time (Nordin, 1994). Further information on LGP can be found in Brameier and Banzhaf (2007).

In the paper discussed here, another important task on LGP application is not considered. A popular modularisation concept in LGP is the evolution of program teams (Brameier and Banzhaf, 2001). A team solution is formed by an uneven number of programs, of which every program has one vote. Whereas in general a team solution performs better than a single solution (Brameier and Banzhaf, 2007; Conrads *et al.*, 1998; Gandomi *et al.*, 2009), the prediction qualities of team solutions were not investigated by the authors.

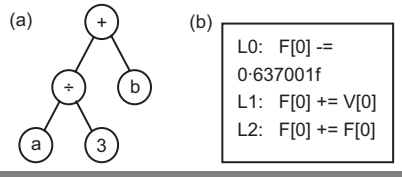

Figure 10. Comparison of GP program structures: (a) TGP (b) LGP

Considering the above arguments, the scientific value of the paper would have increased had the authors investigated the detailed aspects of the utilised method.

Authors' reply

Our objective in describing the GP that way was to introduce it as an extension of the familiar genetic algorithm in the most simple terms, although the code used might have applied GP in a different manner.

Considering the scope of our paper it is felt that subsequent investigators can take the lead from our work further and show how prediction quality could be improved by analysing team solutions, etc.

REFERENCES

Brameier M and Banzhaf W (2001) Evolving teams of predictors with linear genetic programming. *Genetic Programming and Evolvable Machines* **2(4)**: 381–407.

Brameier M and Banzhaf W (2007) *Linear Genetic Programming*. Springer Science + Business Media, New York, NY, USA.

Brameier M, Kantschik W, Dittrich P and Banzhaf W (1998) *SYSGP – A C++ Library of Different GP Variants*. Collaborative Research Center 531, University of Dortmund, Germany, Technical Report No. CI-98/48.

Conrads M, Dolezal O, Francone FD and Nordin P (1998) *Discipulus – Fast Genetic Programming Based on AIM Learning Technology*. Register Machine Learning Technologies, Littleton, CO, USA.

Deschaine LM and Francone FD (2002) *Comparison of Discipulus™ Linear Genetic Programming Software with Support Vector Machines, Classification Trees, Neural Networks and Human Experts*. Register Machine Learning Technologies, Littleton, CO, USA, White Paper. See

http://www.rmltech.com/Comparison.White.Paper.pdf (accessed 25/01/2009).

Francone FD and Deschaine LM (2004) Extending the boundaries of design optimization by integrating fast optimization techniques with machine-code-based linear genetic programming. *Information Sciences* **161(3–4)**: 99–120.

Francone FD, Deschaine LD, Battenhouse T and Warren JJ (2005) Discrimination of unexploded ordnance from clutter using linear genetic programming. In *Genetic Programming Theory and Practice III*. Springer Science + Business Media, New York, NY, USA, Ch. 4.

Gandomi AH, Alavi AH, Kazemi S and Alinia MM (2009) Behavior appraisal of steel semi-rigid joints using linear genetic programming. *Journal of Constructional Steel Research* **65(8–9)**:1738-1750.

Koza JR (1992) *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA.

Langdon WB and Banzhaf W (2005) Repeated sequences in linear genetic programming genomes. *Complex Systems* **15(4)**: 285–306.

Nordin PJ (1994) A compiling genetic programming system that directly manipulates the machine code. *Proceedings of International Conference on Advances in Genetic Programming* (Kenneth E and Kinnear J (eds)). MIT Press, Cambridge, MA, USA, pp. 311–331.