# Hierarchical Forecasting with Polynomial Nets

M.S. Lauretto, F. Nakano, C.A.B. Pereira, and J.M. Stern

**Abstract.** This article presents a two level hierarchical forecasting model developed in a consulting project for a Brazilian magazine publishing company. The first level uses a VARMA model and considers econometric variables. The second level takes into account qualitative aspects of each publication issue, and is based on polynomial networks generated by Genetic Programming (GP).

**Keywords:** Genetic programming, Functional trees, Forecasting, Logistics, Metacontrol, Polynomial networks.

## 1 Introduction

This article describes the authors' consulting project for a leading Brazilian magazine publishing company, nicknamed ABC, and its associated distributor company, nicknamed DE. One of the major logistic challenges of this business is the classic newsstand, newsvendor or newsboy problem, asking for optimal inventory levels. The standard operations research models for this problem assume fixed prices and random demand, see Hadley and Whitin (1963) and Denardo (1982). The inventory levels are then optimized in order to minimize the costs of being either over or under stocked.

The cost of overstock is captured by a well known Brazilian proverb stating that "a day-old newspaper is only good for wrapping fish". Unfortunately, old magazines do not even have that use. In most cases, only the cover page of unsold magazines are stripped and sent back some way along the distribution channel for control purposes, while the rest is recycled at the nearest paper factory. The immediate cost of under stock is lost sales. The long term costs of under stock include customer frustration, possibly leading to permanent fidelity or loyalty transfer to another magazine, low visibility, loss of mind and market share, etc.

M.S. Lauretto, F. Nakano, C.A.B. Pereira, and J.M. Stern
University of São Paulo, Brazil
e-mail: {lauretto,nakano}@ime.usp.br

K. Nakamatsu et al. (Eds.): New Advan. in Intel. Decision Techno., SCI 199, pp. 305–315.
springerlink.com 

The distributor company, DE, deals with this problem at several hierarchical levels through the distribution channels, stocking and possibly restocking one or more times from large and small regional depots to individual newsstands. In this article the word newsstand is used as a generic name, encompassing point of sales ranging from street kiosks to supermarket or bookstore shelves.

The first step of the newsstand problem is to decide the print (or press) runs, that is, to determine the number of copies printed at each batch. Usually a magazine issue stays at the newsstand from one week to one month, and there is no time to reprint an issue.

Most magazines are printed in sections, typically of 16 pages, which are assembled within a cover and bounded. Sections containing articles and advertisement planned and written far ahead can be printed in advance, while the cover and other sections, containing articles referring to current events, are printed in a tight schedule. This process allows for substantial savings in the production costs, resulting in a complex operation that requires careful planning.

The optimization aspects of the problem are going to be reported elsewhere; at this article we focus on demand forecasting. The demand is generated by subscribers, newsstands and a small reserve for the back issue service. The number of subscribers is relatively stable over time, posing little challenge for accurate forecasting. In contrast, the newsstands demand is very sensitive to current events, specific aspects of individual issues, and current marketing efforts.

The forecasting tool developed at this consulting project uses a two level hierarchical approach. The first level uses a VARMA (vector auto-regressive moving average) model, see Brockwell and Davis (1991). The VARMA model is based on econometric variables like subscription and newsstand price, minimum, average or typical wage or income of the target populations, seasonal effects, number of days in the newsstand, delay between the release date of an issue and typical payday(s), etc. This first level gives good predictions for average sales, but can be improved to more accurately predict local fluctuations.

The second level of the hierarchical model is based on polynomial networks, an instance of general functional networks briefly described in section 2. This level takes into consideration the qualitative aspects specific to individual issues like the (quality of the) cover story, cover celebrity, cover photo, editorial content, point of sale advertising, national/regional marketing, promotional gifts, etc.

## 2 Functional Trees

This section presents an overview of some theoretical aspects of GP used in the synthesis of functional trees, including a few topics related to the authors' current research.

Functional tree methods are used for finding the specification of a complex function. This complex function must be composed recursively from a finite set of primitive functions or operators, $OP = \{op_1, op_2, \ldots op_p\}$, and from a set of atoms, $A = \{a_1, a_2, \ldots\}$. The $k$-th operator, $op_k$, takes a specific number, $r(k)$, of arguments,

also known as the *arity* of $op_k$. We use three representations for (the value returned by) the operator $op_k$ computed on the arguments $x_1, x_2, \ldots x_{r(k)}$ :

$$op_k(x_1, \ldots x_{r(k)}) \ , \qquad \begin{array}{c} op_k \\ / \quad \backslash \\ x_1 \ \ldots \ x_{r(k)} \end{array} \ , \qquad \left(op_k \, x_1 \ \ldots \ x_{r(k)}\right) \ .$$

The first is the usual form of representing a function in mathematics; the second is the tree representation which displays the operator with branches to its arguments; and the third is the prefix, preorder or LISP style representation, which is a compact form of the tree representation.

As a didactical example, let us consider the *Boolean network* specification problem, that is, the specification of a Boolean function of $q$ variables, $f(x_1, \ldots x_q)$, to match a target table, $g(x_1, \ldots x_q)$, see Angeline (1996) and Banzhaf el al. (1998). The primitive set of operators and atoms for this problem are the standard ones used in classical logic:

$$OP = \{\sim, \wedge, \vee, \rightarrow, \odot, \otimes\} \ \text{ and } \ A = \left\{x_1, \ldots x_q, 0, 1\right\} \ .$$

Notice that while the first operator, *not*, is unary, the last five, *and, or, imply, nand, xor*, are binary. Also, this set of Boolean operators is clearly redundant. Notice, for example, that all other operators can be synthesized using only the nand operator. This redundancy may, nevertheless, facilitate the search for the best configuration in the problem's functional space.

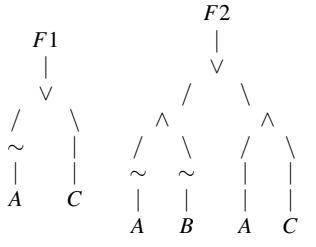
Figure 1 shows a target table, $g(a, b, c)$. As it is usual when the target function is an experimentally observed variable, the target function is *not* completely specified. Unspecified values in the target table are indicated by the don't-care symbol, $*$. The two solutions, $f_1$ and $f_2$, match the table in all specified cases. Solution $f_1$, however, is simpler and for that may be preferred (by some parsimony principle).

Starting from a given random tree, one can start a stochastic search in the problem's (topological) space. In Genetic Programming (GP) terminology, the individual's functional specification is called its *genotype*. the individual's expressed behavior, or computed solutions, is called its *phenotype*. Changing a genotype to a neighboring one is called a *mutation*. The quality of a phenotype, its performance, merit or adaptation, is measured by a *fitness* function. GP does not look at the evolution of a single individual, but rather at the evolution of a population. A time parameter, $t$, indexes the successive generations of the evolving population. In GP, individuals typically have short lives, surviving only a few generations before dying. Meanwhile, populations may evolve for a very long time.

In GP an individual may, during its ephemeral life, share information, that is, swap copies of its (partial) genome, with other individuals. This genomic sharing process is called *sex*. In GP an individual, called a *parent*, may also participate in the creation of a new individual, called its *child*, in a process called *reproduction*. In the reproduction process, an individual gives (partial) copies of its genotype to its offspring. Reproduction involving only one parent is called asexual, otherwise it is called sexual.

Sexual reproduction can be performed by crossover, with parents giving (partial) copies of their genome to their children. Figure 1b shows two parents and a child generated by a single crossover, for the Boolean problem considered in the last example. The tree representation indicates the crossover points by broken edges (=). Notice that in this example the child corresponds to a solution presented earlier. For further details see Stern (2008) and also Banzahf et al. (1998) and Goldberg (1989).

| $A$ | $B$ | $C$ | $G$ | $F1$ | $F2$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | $*$ | 1 | 0 |
| 0 | 1 | 1 | $*$ | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |




$$f_1 = (\sim a) \vee c \ , \ \ f_2 = (\sim a \wedge \sim b) \vee (a \wedge c) \ .$$

$$f_1 = (\vee \ (\sim a) \ c) \ , \ \ f_2 = (\vee \ (\wedge \ (\sim a) \ (\sim b)) \ (a \wedge c)) \ .$$

**Fig. 1a** Two Boolean functional trees for the target $g(a,b,c)$

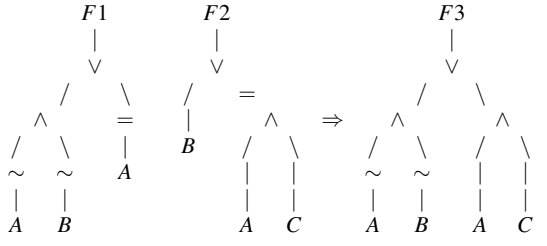



**Fig. 1b** Crossover between Boolean functional trees

Let us now consider the *polynomial network* specification problem in the consulting case at hand. These functional trees use as primitive operators *linear, quadratic* or *cubic* polynomials in one, two or three variables. Two auxiliary operators are defined as follows: A *normalizer* converts its input into an ouput of mean 0 and variance 1, and a *denormalizer* performs the inverse transformation.

Figure 2 displays a typical network for sales forecast used in the consulting project described in sections 1 and 3. Variable $x_5$ is the magazine's sales forecast obtained by a VARMA time series model. Variables $x_1$ to $x_4$ are qualitative variables, in a scale corresponding to approximate decile ranks of Bad (0-1), Weak (1-3), Average (3-7), Good (7-9) and Excellent (9-10). This scale is used by experts to assess the appeal or attractiveness of each individual issue of the magazine, according to:

(1) cover impact; (2) editorial content; (3) promotional items; and (4) point of sale marketing. Normalizers at the input edges and a denormalizer at the output edge of the polynomial network are not shown in the figure.

Of course, the optimization of a polynomial network is far more complex than the optimization of a Boolean network: Even having specified the network topology (identification problem), also the parameters $w_0, w_1, \ldots$ of the polynomial function have to be optimized (estimation problem). Parameter optimization can be based on recursive sub-tree regression; gradient, Partan or conjugate-gradient learning rules, etc. Topology optimization is based on GP algorithms.

Some aspects of GP relevant to the forecasting problem at hand and related to the authors' current research are briefly discussed in the following subsections.

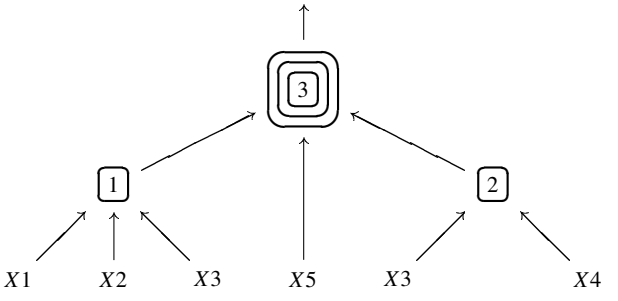



**Fig. 2** Polynomial network. Rings on a node: 1- Linear; 2- (incomplete) Quadratic; 3- (incomplete) Cubic

## *2.1 Meta-control, Building Blocks and Modularity*

P. Angeline (1996) noted that GP generated networks typically contain large segments of *extraneous* code, that is, code segments that, if removed, do not (significantly) alter the solution computed by the network. Trivial examples of extraneous code segments are $(+\, s\, 0)$ and $(*\, s\, 1)$, where $s$ is a sub-expression. By their very definition, extraneous code segments cannot (significantly) contribute to an individual's fitness, and hence to its survival or mating probabilities. However, Angeline noticed that the presence of extraneous code could significantly contribute to the expected fitness of the individual's descendents! Apparently, the role of these (sometimes very large) patches of inert code is to isolate important blocks of working code, and to protect these *modules* or *building blocks* from being broken at recombination (destructive crossover).

In biological organisms, the genetic code of eukaryotes exhibits similar regions of code (DNA) that are or are not expressed in protein synthesis; these regions are called *exons* and *introns*, respectively. Introns do not directly code amino-acid sequences in proteins; nevertheless, they seem to have an important role in the meta-control of the genetic material expression and reproduction.

Iba and Sato (1992, p.548), in a pioneering work, proposed a meta-level strategy for GP based on a self-referential representation, where

> *"[a] self-referential representation maintains a meta-description, or meta-prescription, for crossover. This meta-genetic descriptions are allowed to co-evolve with the gene pool. Hence, genetic and meta-genetic code variations are jointly selected. How well the genetic code is adapted to the environment is translated by the merit or objective function which, in turn, is used for the immediate, short-term or individual selection process. How well the genetic and meta-genetic code are adapted to each other impacts on the system's evolvability, a characteristic of paramount importance in long-run survival of the species."*

Subsequent work of several authors (including these) tried to incorporate meta-control parameters to GP. Functional trees may, for example, incorporate edge annotations, interpreted as probability weights, linkage compatibility or affinity, etc. Such annotations are meta-parameters used to control the recombination of the sub-tree directly bellow a given edge. For example, weights may be used to specify the probability that recombination takes place at that edge, while linkage compatibility or affinity annotations may be used to identify homologous or compatible genes, specifying the possibility or probability of swapping two sub-trees. Other annotations, like context labels, semantic tags, variable type, etc., may provide additional information about the possibility or probability of recombination or crossover, the need of type-cast operations, etc.

Banzahf (1998, p.164), gives a simple example of functional tree annotation:

> *"Recently, we introduced the explicitly defined introns (EDI) into GP. An integer value is stored between every two nodes in the GP individual. This integer value is referred as the EDI value (EDIV). The crossover operator is changed so that the probability that crossover occurs between any two nodes in the GP program is proportional to the integer value between the nodes. That is, the EDIV integer value strongly influences the crossover sites chosen by the modified GP algorithm, Nordin et al. (1996).*
>
> *The idea behind EDIVs was to allow the EDIV vector to evolve during the GP run to identify the building blocks in the individual as an* emergent *phenomenon. Nature may have managed to identify genes and to protect them against crossover in a similar manner. Perhaps if we gave the GP algorithm the tools to do the same thing, GP, too, would learn how to identify and protect the building blocks. If so, we would predict that the EDIV values within a good building block should become low and, outside the good block, high."*

Let us finish this section presenting two interpretations for the role of modularity in genetic evolutionary processes. This interpretations are common in biology, computer science and engineering, an indication that they provide powerful insights. These two metaphors are commonly referred to as:

- New *technology dissemination* or *component design substitution*, and
- *Damage control* or *repair mechanism*.

The first interpretation is perhaps the more evident. In a modular system, a new design for an old component can be easily incorporated and, if successful, be rapidly disseminated. A classical example is the replacement of mechanical carburetors by

electronic injection as the standard technology for this component of gasoline automotive engines. The large assortment of *upgrade kits* available in any automotive or computer store gives a strong evidence of how much these industries rely on modular design.

The second interpretation explains the possibility for the "continued evolution of germlines otherwise destined to extinction", see Michod and Levin (1988). A classic illustration related to the damage control and repair mechanisms offered by modular organization is given by the Hora and Tempus parable of Simon (1996), see also Growney (1998). The lessons learned from this parable may be captured by the following dicta of Herbert Simon:

> *"The time required for the evolution of a complex form from simple elements depends critically on the number and distribution of potential intermediate stable subassemblies."* Simon (1996, p.190).
> *"Hierarchy, I shall argue, is one of the central structural schemes that the architect of complexity uses."* Simon (1996, p.184).

## 2.2 Schemata and Parallelism

The *intrinsic parallelism* argument, first presented in Holland (1975), provides alternative insights into the concepts of building blocks and modularity. For a mathematical analysis of this argument, see Reeves (1993, Ch.4) or Stern (2008, H.2). According to Reeves,

> *"The underlying concept Holland used to develop a theoretical analysis of his GA [GP] was that of* schema*. The word comes from the past tense of the Greek verb* $\varepsilon\chi\omega$*,* echo*, to have, whence it came to mean shape or form; its plural is* schemata*."* (p.154)

Schemata are partially specified patterns in a program, like partially specified segments of prefix expressions, or partial code for functional sub-trees. The *length* and *order* of a schema are the *distance* between the first and last defined position on the schema, and the number of defined positions, respectively. The intrinsic parallelism theorem states that the number of schemata (of order $l$ and length $2l$, in binary coded programs, in individuals of size $n$) present in a population of size $m$, is proportional to $m^3$. The crossover operator enriches the neighborhood of an individual with the schemata present in other individuals of the population. If, as suggested by the implicit parallelism theorem, the number of such schemata is large, GP is likely to be an effective strategy. Schaffer (1987, p.89), celebrates this theorem stating that:

> *"this [intrinsic parallelism] constitutes the only known example of combinatorial explosion working to advantage instead of disadvantage."*

Indeed, Schaffer has ample reasons to praise Holland's result. Nevertheless, we must analyze this important theorem carefully, in order to understand its consequences correctly. In particular, we should pay close attention to the unit, $u$, used to measure the population size, $m$. It so happens that this unit, $u = 2^l$, is itself exponential in the schemata order. Therefore, the combinatorial explosion works to our

advantage as long as we use short schemata, relative to the log-size of the population. This situation is described by Reeves as:

> *"Thus the ideal situation for a GA [GP] are those where short, low-order schemata combine with each other to form better and better solutions. The assumption that this will work is called by Goldberg (1989) the* building-block hypothesis*. Empirical evidence is strong that this is a reasonable assumption in many problems."* (p.158)

One key question we must face in order to design a successful GP application is, therefore: How then can we organize our working space so that our programming effort can rely on short schemata?

The solution to this question is well known to computer scientists and software engineers: Organize the programs hierarchically (recursively) as self-contained (encapsulated) building-blocks (modules, functions, objects, sub-routines). As shown in the previous subsection, meta-contol is a key mechanism for modular organization in GP, promoting the spontaneous emergence of complex hierarchical systems.

For further details about the implementation of these concepts in the optimization of polynomial networks, see Nikolaev and Iba (2001, 2003, 2006). For further implications of these ideas in artificial intelligence and statistical modeling, see Stern (2008, ch.5) and the authors' forthcoming articles.

## 3 Example of Forecasting

In this section we return to our consulting project, and present a case study based on time series for a magazine published by ABC. Due to confidentiality and disclosure agreements, the time series given in this example has been de-trend and is presented in a relative percentage scale. Forecasts are always made three month ahead with the current past data. From the total of 39 months comprising the time series, the first 27 were used as training data, and the remaining 12 months as test data.

The first level of the hierarchical model consists of a VARMA model, implemented in order to capture trends, seasonal effects and market elasticities, built using automated variable selection procedures, see Brockwell and Davis (1991). The available explanatory variables include the dates of distribution and recall of each issue, its price for subscription and at the newsstand, minimum, average or typical wage or income of the target populations, typical payday schedules, etc.

Figure 3 presents the actual sales time series and the sales forecasts provided by the VARMA econometric model (top), as well as the models improved by qualitative data, using linear regression (center) and the polynomial network in Figure 2 (bottom). Table 1 shows the average error rates for VARMA, linear regression and polynomial network models. Notice that error rates provided by polynomial networks are smaller than in VARMA and linear regression models. The optimal polynomial network selection is guided by a regularization parameter, $\rho$, controlling the network complexity vs. training error. Its default value is $\rho = 1.0$. As expected, if $\rho$ is too large, the network becomes too simple, resulting in an under-fitted model. On the other hand, if $\rho$ is too small, the network becomes too complex, resulting in an

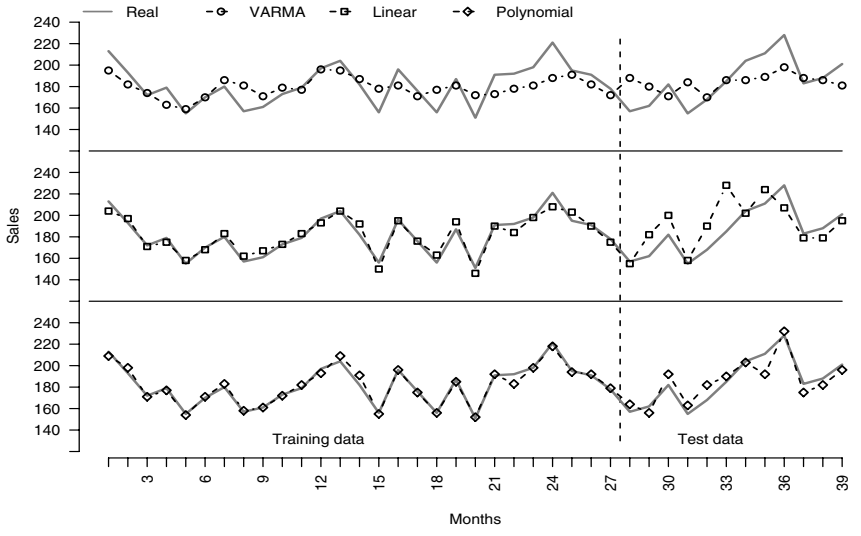



**Fig. 3** Monthly sales and corresponding forecasts with VARMA models (top), linear regression (center) and polynomial network (bottom)

**Table 1** Error averages using VARMA, Linear reg. and Polynomial networks

| Model | Training dataset | Test dataset |
|---|---|---|
| VARMA alone | 6.3% | 8.6% |
| + Linear regression | 4.1% | 7.4% |
| + Polynomial network, $\rho = 0.5$ | 1.1% | 5.1% |
| + Polynomial network, $\rho = 1.0$ | 2.5% | 4.2% |
| + Polynomial network, $\rho = 2.0$ | 3.4% | 4.6% |

over-fitted model (over adjusted to the peculiarities of the training data). In either case, the network has low predictive (generalization) power.

## 4 Enterprise Integration

As in so many Operations Research projects, solving the mathematical and algorithmic aspects of the optimization and statistical models, and its computational implementation in a user friendly decision support tool, is just part of the entire consulting project. Training the corporate decision makers to use the tools and carefully explaining the concepts involved is also an essential part of the project. All those are prerequisites to the vital goal of integrating the new OR tools into the everyday life of the enterprise. Otherwise, the full benefits of the project are never achieved or, even worst, the new fancy tools are soon condemned to oblivion.

ABC is organized in business units according to major target populations, for example: children, including comic books; male teens; female teens; women, including arts, house and garden, gossip, etc.; men, including cars, computers, sports, swim suite, etc; business and economy; and general news.

ABC's business units were often evaluated by their total sales, market share, and other performance indices that do not take into account production and distribution costs. Meanwhile, DE and the printing plants were often evaluated by their operating costs, regardless of the global company performance. Needless to say, such evaluation metrics generated conflict and misunderstanding inside the company.

The primary objective of the statistical and optimization tools developed in this consulting project was to improve the quantitative fine tuning of the operation, and the project successfully accomplished this goal. However, the project could also make significant contributions to a secondary objective, namely, to improve the cooperation, integration, rational dialogue and mutual understanding concerning the different roles played by the several agents in such a complex operation. We hope that, in the future, it will also contribute for the development of more encompassive performance metrics, capable of harmonizing and integrating locally conflicting goals into global multi-objective functions.

## References

Angeline, P.: Two Self-Adaptive Crossover Operators for Genetic Programming. In: Angeline, Kinnear (eds.) Advances in Genetic Programming, ch. 5, vol. 2, pp. 89–110. MIT, Cambridge (1996)

Banzahf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic Algorithms (1998)

Brockwell, P.J., Davis, R.A.: Time Series: Theory and Methods, 2nd edn. Springer, Heidelberg (1991)

Denardo, E.: Dynamic Programming. Prentice-Hall, Englewood Cliffs (1982)

Farlow, S.J.: Self-Organizing Methods in Modeling. Marcel Dekker, New York (1984)

Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)

Growney, J.S.: Planning for Interruptions. Mathematics Magazine 55(4), 213–219 (1998)

Hadley, G., Whitin, H.M.: Analysis of Inventory Systems. Prentice-Hall, Englewood Cliffs (1963)

Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)

Iba, H., Sato, T.: Meta-Level Strategy for Genetic Algorithms Based on Structured Representation. In: Proc. of the Second Pacific Rim Int. Conf. on Artificial Intelligence, pp. 548–554 (1992)

Madala, H.R., Ivakhnenko, A.G.: Inductive Learning Algorithms for Complex Systems Modeling. CRC, Boca Raton (1994)

Michod, R.E., Levin, B.R.: The Evolution of Sex. An Examination of Current Ideas. Sinauer (1988)

Nikolaev, N.Y., Iba, H.: Regularization Approach to Inductive Genetic Programming. IEEE Transactions on Evolutionary Computation 5(4), 359–375 (2001)

Nikolaev, N.Y., Iba, H.: Learning Polynomial Fedforward Neural Networks by Genetic Programming and Backpropagation. IEEE Transactions on Neural Networks 14(2), 337–350 (2003)

Nikolaev, N.Y., Iba, H.: Adaptive Learning of Polynomial Networks. In: Genetic and Evolutionary Computation. Springer, Heidelberg (2006)

Reeves, C.R.: Modern Heuristics for Combinatorial Problems. Blackwell Scientific, Malden (1993)

Schaffer, J.D.: Some Effects of Selection Procedures on Hyperplane Sampling by Genetic Algorithms. In: Davis, L. (ed.) Genetic Algorithms and Simulated Annealing, Pittman, pp. 89–103 (1987)

Simon, H.A.: The Sciences of the Artificial. MIT Press, Cambridge (1996)

Stern, J.M.: Cognitive Constructivism and the Epistemic Significance of Sharp Statistical Hypotheses. In: MaxEnt 2008, The 28th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering, Boracéia, São Paulo, Brazil, July 6-11 (2008)