Evolutionary Computation

Alex Alves Freitas

PUC-PR, PPGIA-CCET
Rua Imaculada Conceicao, 1155
Curitiba – PR, 80215-901. Brazil
alex@ppgia.pucpr.br
http://www.ppgia.pucpr.br/~alex

Abstract This chapter addresses the integration of knowledge discovery in databases (KDD) and evolutionary algorithms (EAs), particularly genetic algorithms and genetic programming. First we provide a brief overview of EAs. Then the remaining text is divided into three parts. Section 2 discusses the use of EAs for KDD. The emphasis is on the use of EAs in attribute selection and in the optimization of parameters for other kinds of KDD algorithms (such as decision trees and nearest neighbour algorithms). Section 3 discusses three research problems in the design of an EA for KDD, namely: how to discover comprehensible rules with genetic programming, how to discover surprising (interesting) rules, and how to scale up EAs with parallel processing. Finally, section 4 discusses what the added value of KDD is for EAs. This section includes the remark that generalization performance on a separate test set (unseen during training, or EA run) is a basic principle for evaluating the quality of discovered knowledge, and then suggests that this principle should be followed in other EA applications.

1. Introduction

The evolutionary algorithms paradigm consists of stochastic search algorithms that are based on abstractions of the processes of Neo-Darwinian evolution. The basic idea is that each "individual" of an evolving population encodes a candidate solution (e.g. a prediction rule) to a given problem (e.g. classification). Each individual is evaluated by a fitness function (e.g. the predictive accuracy of the rule). Then these individuals evolve towards better and better individuals via operators based on natural selection, i.e. survival and reproduction of the fittest, and genetics, e.g. crossover and mutation operators - Goldberg (1989), Michalewicz (1996), Koza (1992, 1994), Koza et al. (1999), Banzhaf et al. (1998).

The crossover operator essentially swaps genetic material between two individuals. Figure E8.1.1 illustrates a simple form of crossover between two individuals, each represented as a string with four genes. In the context of KDD, each gene could be, say, an attribute-value condition of a rule (see below). Figure E8.1.1(a) shows the individuals before crossover. A crossover point is randomly chosen, represented in the figure by the symbol "|" between the second and third genes. Then the genes to the right of the crossover point are swapped between the two individuals, yielding the new individuals shown in Figure E8.1.1(b).

```
 X1 X2 |  X3 X4          X1 X2|  Y3 Y4
 Y1 Y2 | Y3 Y4           Y1 Y2|  X3 X4

(a) Before crossover    (b) After crossover
```

Figure 1: Simple example of crossover

The mutation operator simply changes the value of a gene to a new random value. Both crossover and mutation are stochastic operators, applied with user-defined probabilities. The probability of mutation is usually much lower than that of crossover. However, mutation is still necessary to increase the genetic diversity of individuals in the population. Note that mutation can yield gene values that are not present in the current population, unlike crossover, which only swaps existing gene values between individuals.

An important characteristic of evolutionary algorithms is that they perform a *global* search. Indeed, evolutionary algorithms work with a population of candidate solutions, rather than working with a single candidate solution at a time. This, together with the fact they use stochastic operators to perform their search, reduce the probability that they will get stuck in local maxima, and increase the probability that they will find the global maximum.

2 Use of Evolutionary Algorithms for KDD

2.1 Evolutionary Algorithms for Rule Discovery

Among the several kinds of evolutionary algorithms used in the literature, genetic algorithms (GA) and genetic programming (GP) have been the most used in rule discovery. These two kinds of algorithms differ mainly with respect to the representation of an individual.

In GA an individual is usually a linear string of rule conditions, where each condition is often an attribute-value pair. The individual can represent a rule, as illustrated in Figure E8.1.2(a), or a rule set, as illustrated in Figure E8.1.2(b). In both illustrations the individual encodes only the conditions of the antecedent (IF part) of a classification rule, and conditions are implicitly connected by a logical AND. In Figure E8.1.2(b) the symbol "||" is used to separate rules within the individual. The predicted class (the THEN part of the rule) can be chosen in a deterministic, sensible way as the majority class among all data instances satisfying the rule antecedent. Supposing that the rules in Figure E8.1.1 refer to a credit data set, the system would choose a predicted class like "*credit = good*" for those rules.

The several-rules-per-individual approach has the advantage that the fitness of an individual can be evaluated by considering its rule set as a whole, by taking into account rule interactions. However, this approach makes the individual encoding more complicated and syntactically longer, which in turn may require more complex genetic operators. Some algorithms following this approach are proposed by De Jong et al. (1993), Janikow (1993), Pei et al. (1997).

| Salary = "high" | Age > 18 | . . . (other rule conditions)

(a) GA individual = one rule antecedent

| Employed = "yes" | C/A_balance = "high" || Salary = "high" || . . . (other rules)

(b) GA individual = a set of rule antecedents

Figure 2: Examples of individual encoding in GA for rule discovery.

The single-rule-per-individual approach makes the individual encoding simpler and syntactically shorter. However, it introduces the problem that the fitness of an individual (a single rule) is not necessarily the best indicator of the quality of the discovered rule set. Some algorithms using the one-rule-per-individual encoding are proposed by Greene & Smith (1993), Giordana & Neri (1995), Freitas (1999a), Noda et al. (1999).

In GP an individual is usually represented by a tree, with rule conditions and/or attribute values in the leaf nodes and functions (e.g. logical, relational or mathematical operators) in the internal nodes. An individual's tree can grow in size and shape in a very dynamical way. Figure E8.1.3 illustrates a GP individual representing the rule antecedent: IF (*Employed*=*"yes"*) AND ((*Salary* – *Mortgage_debt*) > 10,000). Assuming again a credit application domain, the rule consequent (THEN part) would be a prediction such as "*credit=good*".

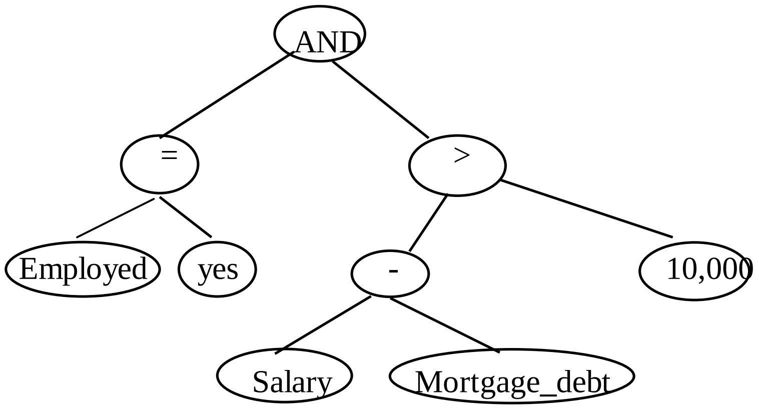

Figure 3: Example of genetic programming individual for rule discovery

We emphasize that encoding rules into a GP individual is a nontrivial problem, due to the closure property of GP. This property requires that the output of a node can be used as the input to any parent node in the tree. This is a problem in the context of KDD. For instance, the operator "<" can be used in a parent node if its children contain "*Age*" and "18", but not if they contain "*Sex*" and "*female*".

Several solutions have been proposed to cope with the requirement of closure in GP. Most of these solutions involve some kind of constrained-syntax GP, often exploiting domain-related semantics - Bhattacharyya et al. (1998), Ngan et al. (1998). A simpler approach is to booleanize all attribute values and use only logical (AND, OR, etc) functions in the internal nodes – Hu (1998), Eggermont et al. (1999), Bojarczuk et al. (1999).

The motivation for using evolutionary algorithms in rule discovery

One of the major contributions of evolutionary algorithms for rule discovery is that they tend to cope well with attribute interactions, as a consequence of their global search. This is in contrast to the local, greedy search performed by often-used rule induction and decision tree algorithms.

Most rule induction algorithms generate (prune) a rule by selecting (removing) *one-rule-condition-at-a-time*. The problem with this approach is illustrated in Figure E8.1.4. The figure contains data instances having either positive ("+") or negative ("-") class, depending on the value of two boolean attributes A_1 and A_2. The goal is to find rule conditions that discriminate between instances with positive and negative classes. The rule condition A_1 = "*false*" (covering instances at the left of the vertical dashed line) is not useful, since it covers as many positive-class examples as negative-class examples. The same holds for its dual condition A_1 = "*true*". The rule condition A_2 = "*false*" (covering instances below the horizontal dashed line), as well as its dual A_2 = "*true*", are not useful either, for the same reason. Hence, an algorithm that selects one rule condition at a time would consider A_1 and A_2 irrelevant attributes and choose another attribute to form a rule condition. However, an algorithm that selects two or more conditions at a time could easily form a rule such as:

IF (A_1 = "*false*") AND (A_2 = "*true*") THEN (*positive-class*).

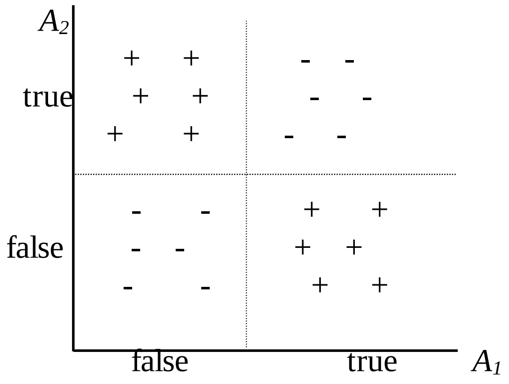

Figure 4: Attribute interaction in classification.

Evolutionary algorithms usually evaluate a rule as a whole via the fitness function, rather than evaluating the impact of adding/removing one condition to/from a rule. In addition, crossover usually swaps several-rule-conditions-at-a-time between two individuals. Therefore, evolutionary algorithms tend to cope well with attribute interaction.

This is *not* to say that evolutionary algorithms are inherently superior to rule induction algorithms. No rule discovery algorithm is superior in all cases - Domingos (1998), Michie et al. (1994).

2.2 Use of Evolutionary Algorithms in Attribute Selection

Evolutionary algorithms have been quite successful in attribute selection [link to Section C3.2] - Bala et al. (1995a, 1995b), Vafaie & De Jong (1993), Guerra-Salcedo & Whitley (1998, 1999), Martin-Batista & Vila (1999), Hsu et al. (1999). The reason is that the core problem of attribute selection is to cope with attribute interaction, since the original attributes can be redundant and correlated in a highly nonlinear manner. This seems to be a kind of problem where the global search performed by evolutionary algorithms tends to present better results than a local search-based approach selecting one attribute at a time.

Most evolutionary attribute selection methods are based on a simple GA, where an individual directly represents a candidate attribute subset. In essence, an individual is a binay string with m genes, where m is the number of attributes. Each gene can take on the value 1 or 0, indicating whether or not the corresponding attribute is selected. For instance, the individual 0 1 1 0 1 0 0 0 represents a candidate solution where the second, third, and fifth attributes are selected.

A more elaborate individual encoding has been proposed by Cherkauer & Shavlik (1996). In their approach, each gene of an individual contains either an attribute name or no attribute, denoted by 0. All the attributes occurring in any of the genes of an individual are the attributes selected by the individual. For instance, the individual 0 0 A_7

A_7 A_2 0 A_7 A_5 0 0 represents a candidate solution where the attributes A_7, A_2 and A_5 are selected. This unconventional individual encoding has some advantages, e.g. the fact that occurrences of very relevant attributes may be replicated across the genome.

Regardless of the internal details of individual encoding, the attribute subset selected by an individual is given to a KDD algorithm. The fitness of that individual depends on the result (e.g. predictive accuracy) achieved by that algorithm using only those selected attributes. Hence, the GA acts as a wrapper around the KDD algorithm.

2.3 Optimization of Parameters for Other KDD Algorithms

Genetic Algorithms (GA) have also been successfully used as a wrapper to optimize parameters of several other kinds of KDD algorithms. Some examples are as follows.

Kelly & Davis (1991) and Punch et al. (1993) used a GA to optimize the attribute weights of a k-nn classifier [link to Section C5.1.6]. Each individual consists of m real-valued weights, where m is the number of attributes. The fitness of an individual is measured by the predictive accuracy of a k-nn classifier, by using the attribute weights contained in the individual. Raymer et al. (1996) extended this approach by using a genetic programming system to construct new attributes.

Turney (1995) proposed a hybrid GA / decision tree system [link to Section C5.1.3] where each individual of the GA consists essentially of attribute costs to be given as input to a cost-sensitive decision tree algorithm. The fitness of an individual is measured by the misclassification cost of the decision tree generated by the cost-sensitive decision tree algorithm.

Janikow (1995) proposed a GA to optimize the fuzzy sets [link to Section B7] used to build a decision tree. The decision tree algorithm is given fuzzy attributes taking on linguistic values such as “low”, “medium”, “high” - each of which is a fuzzy set with an associated membership function. The GA optimizes the shape of the membership function of each fuzzy set.

3 Research problems in evolutionary algorithms relevant for KDD

3.1 Discovering comprehensible rules with Genetic Programming (GP)

The usually-large size and complexity of GP trees makes it difficult to understand them. A partial solution for this problem is to include in the fitness function a penalty term that penalizes complex (syntactically long) rules - Bojarczuk et al. (2000).

More elaborate approaches for improving discovered-rule comprehensibility include the use of hybrid GP / decision trees systems [link to Section C5.1.3]. For instance, Ryan & Rayward-Smith (1998) proposed a hybrid system where a decision tree algorithm is called not only to generate each individual of the initial population but also to modify individual trees during the GP run. The fitness function favors the discovery of small, accurate trees.

Marmelstein & Lamont (1998) proposed a system to construct a decision tree using GP to implement the decision nodes. Each node of the decision tree evolves a GP to separate the data into two classes. Hence, each GP node in the decision tree is relatively small (due to the divide-and-conquer approach of decision trees) and can be separately analyzed.

3.2 Discovering surprising rules

Overall, evolutionary algorithms seem to have a good potential to discover truly surprising rules, due to their ability to cope well with attribute interaction. The rationale for this argument is as follows.

Most users have a reasonably good idea of the relationship between a single predicting attribute and the goal (class) attribute. For instance, in credit data sets the higher the salary of an employee the better his/her credit.

What users do not usually know is the more complex relationship between several-attributes-at-a-time and the goal attribute. For instance, credit quality depends on the interaction between salary, number of dependents, mortgage debt, etc.

Recently there has been a growing interest in rule surprisingness measures in the rule induction literature - Liu et al. (1997), Suzuki & Kodratoff (1998), Padmanabhan & Tuzhilin (1998), Freitas (1998a, 1999b). An interesting research direction is to adapt these measures or design new ones to evaluate the rules produced by evolutionary algorithms - Noda et al. (1999).

3.3 Scaling up Evolutionary Algorithms with Parallel Processing

In the context of mining very large databases, the vast majority of the processing time of an evolutionary algorithm is spent on evaluating an individual's fitness. This processing time can be significantly reduced by using parallel processing techniques - Freitas & Lavington (1998), Anglano et al. (1997), Neri & Giordana (1995), Araujo et al. (1999), Flockhart & Radcliffe (1995), Braud & Vrain (1999), Freitas (1998b).

The are two broad ways of parallelizing the computation of the fitness of individuals. The first approach is a kind of *inter*-individual parallelization, distributing the population individuals across the available processors and computing their fitness in parallel. At a given time different processors compute the fitness of different individuals, but each individual's fitness is computed by a single processor. It is common to replicate all the data being mined in all the processors, so that the fitness of an individual can be computed without accessing data in other processors. However, this strategy reduces scalability for large databases.

The second approach is a kind of *intra*-individual parallelization, where the fitness of each individual is computed in parallel by all processors. This is a data-parallel approach, where the data being mined is partitioned across the processors. At a given time, each processor is using its local data to compute a partial quality measure for the individual. These partial measures are then combined to compute the individual's fitness. Note that these two parallelization approaches can be combined into a hybrid approach.

4 Added Value of KDD for Evolutionary Algorithms

KDD offers several research opportunities for new methodological developments in evolutionary algorithms. We briefly draw attention here to two possibilities.

First, KDD has a very interdisciplinary nature and uses many different paradigms of knowledge discovery algorithms. This motivates the integration of evolutionary algorithms with other knowledge discovery paradigms, as discussed in section E8.1.3. There has also been research on the integration between genetic programming and database systems - Martin et al. (1998), Freitas (1997), Ryu and Eick (1996).

Second, several KDD tasks involve some kind of prediction, where generalization performance on a separate test set is much more important than the performance on a training set. This is a basic principle for evaluating the quality of a solution in several KDD tasks, and it should be followed in other kinds of problems where evolutionary algorithms are being applied.

For instance, consider the problem of simulating the navigation behavior of an ant aiming at collecting all the food lying along an irregular trail. Having implemented a GP system to solve this problem in a particular trail, Koza (1992) states (pp. 150) that: “As we will see this one fitness case is sufficiently representative for this particular problem to allow the ant to learn to navigate this trail and reasonable generalizations of this trail.” However, Kuscu (1998) has shown that training the GP on a single trail does not lead to a good performance in other similar trails. Kuscu has also shown that it is possible to achieve a good generalization performance on test trails by using more training trails.

References

Anglano, C.; Giordana, A.; Lo Bello, G.; Saitta, L. (1997) A network genetic algorithm for concept learning. Back, T. (Ed.) *Proc. 7th Int. Conf. Genetic Algorithms (ICGA-97)*, 434-441. San Mateo, Calif.: Morgan Kaufmann.

Araujo, D.L.A.; Lopes, H.S.; Freitas, A.A. (1999) A parallel genetic algorithm for rule discovery in large databases. *To appear in Proc. 1999 IEEE Systems, Man and Cybernetics Conf.* Tokyo, Japan, Oct. 1999. IEEE Piscataway, NJ.

Bala, J.; DeJong, K.; Huang, J.; Vafaie, H.; Wechsler, H. (1995a) Using learning to facilitate the evolution of features for recognizing visual concepts. *Evolutionary Computation 30:* 441-451. Special issue on evolution, learning, and instinct: 100 years of Baldwin effect.

Bala, J.; Huang, J.; Vafaie, H.; DeJong, K.; Wechsler, H. (1995b) Hybrid learning using genetic algorithms and decision trees for pattern classification. C.S. Mellish. (Ed.) *Proc. 14th Int. Joint Conf. on Artificial Intelligence (IJCAI-95)*, 719-724. San Mateo, Calif.: Morgan Kaufmann.

Banzhaf, W.; Nordin, P.; Keller, R.E.; Francone, F.D. (1998) *Genetic Programming ~ an Introduction: On the Automatic Evolution of Computer Programs and Its Applications.* San Mateo, Calif.: Morgan Kaufmann.

Battacharyya, S.; Zumbach, G.; Olivier, P. Representational semantics for genetic programming based learning in high-frequency financial data. Koza, J.R.; Banzhaf, W.; Chellapilla, K.; Deb, K.; Dorigo, M.; Fogel, D.B.; Garzon, M.H.; Goldberg, D.E.; Iba, H.; Riolo, R.L. (Eds.) *Genetic Programming 1998: Proc. 3rd Annual Conf.*, 11-16. San Mateo, Calif.: Morgan Kaufmann.

Bojarczuk, C.C.; Lopes, H.S.; Freitas, A.A. (1999) Discovering comprehensible classification rules using genetic programming: a case study in a medical domain. *To appear in Proc. 1999 Genetic and Evolutionary Computation Conf. (GECCO-99).* Orlando, FL, USA. July 1999. Morgan Kaufmann, San Mateo, CA.

Bojarczuk, C.C.; Lopes, H.S.; Freitas, A.A. (2000) Genetic programming for knowledge discovery in chest pain diagnosis. *IEEE Engineering in Medicine and Biology Magazine* 19(4), pp. 38-44. July 2000.

Braud, A. & Vrain, C. (1999) Parallelisation d’Algorithmes Genetiques fondee sur le modele BSP. (in French) *Rapport de Recherche No. 99-06.* Universite d’Orleans, LIFO. France.

Cherkauer, K.J. and Shavlik, J.W. (1996) Growing simpler decision trees to facilitate knowledge discovery. Simoudis, E.; Han, J.; Fayyad, U. (Eds.) *Proc. 2nd Int. Conf. Knowledge Discovery & Data Mining (KDD-96)*, 315-318. Menlo Park, Calif: AAAI.

DeJong, K.A.; Spears, W.M.; Gordon, D.F. (1993) Using genetic algorithms for concept learning. *Machine Learning* 13, 161-188.

Domingos, P. (1998) Occam's two razors: the sharp and the blunt. Agrawal, R.; Stolorz, P.; Piatetsky-Shapiro, G. (Eds.) *Proc. 4th Int. Conf. Knowledge Discovery & Data Mining (KDD-98)*, 37-43. Menlo Park, Calif.: AAAI.

Eggermont, J.; Eiben, A.E.; vanHemert, J.I. (1999) A comparison of genetic programming variants for data classification. *To appear in Proc. Third International Symposium on Intelligent Data Analysis (IDA-99)*. Amsterdam, The Netherlands. August 1999. Springer-Verlag, Berlin.

Flockhart, I.W. and Radcliffe, N.J. (1995) GA-MINER: parallel data mining with hierarchical genetic algorithms - final report. *EPCC-AIKMS-GA-MINER-Report 1.0*. University of Edinburgh, UK.

Freitas, A.A. (1997) A genetic programming framework for two data mining tasks: classification and generalized rule induction. Koza, J.R.; Deb, K; Dorigo, M; Fogel, D.B.; Garzon, M; Iba, H; Riolo, R.L. (Eds.) *Genetic Programming 1997: Proc. 2nd Annual Conf.*, 96-101. San Mateo, Calif.: Morgan Kaufmann.

Freitas, A.A. (1998a) On objective measures of rule surprisingness. Zytkow, J.M. & Quafafou, M. (Eds.) *Principles of Data Mining & Knowledge Discovery (Proc. 2nd European Symp., PKDD-98). Lecture Notes in Artificial Intelligence 1510*, 1-9. Berlin: Springer-Verlag.

Freitas, A.A. (1998b) A survey of parallel data mining. Arner, H.F. & Mackin, N. (Eds.) *Proc. 2nd Int. Conf. Practical Applications of Knowledge Discovery & Data Mining (PADD-98),* 287-300. London: The Practical Application Company.

Freitas, A.A. (1999a). A genetic algorithm for generalized rule induction. Roy, R; Furuhashi, T.; Chawdhry, P.K. (Eds.) *Advances in Soft Computing - Engineering Design and Manufacturing (Proc. WSC3, 3rd on-line world conf., hosted on the internet, 1998),* 340-353. Berlin: Springer-Verlag.

Freitas, A.A. (1999b) On rule interestingness measures. *Knowledge-Based Systems Journal*, 12(5-6), pp. 309-315. Oct. 1999.

Freitas, A.A. and Lavington, S.H. (1998) *Mining Very Large Databases with Parallel Processing*. Boston: Kluwer.

Giordana, A.; and Neri, F. (1995) Search-intensive concept induction. *Evolutionary Computation 3(4)*: 375-416, Winter 1995.

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

Greene, D.P. and Smith, S.F. (1993) Competition-based induction of decision models from examples. *Machine Learning 13*, 229-257.

Guerra-Salcedo, C. and Whitley, D. (1998) Genetic search for feature subset selection: a comparison between CHC and GENESIS. Koza, J.R.; Banzhaf, W.; Chellapilla, K.; Deb, K.; Dorigo, M.; Fogel, D.B.; Garzon, M.H.; Goldberg, D.E.; Iba, H.; Riolo, R.L. (Eds.) *Genetic Programming 1998: Proc. 3rd Annual Conf.*, 504-509. San Mateo, Calif.: Morgan Kaufmann.

Guerra-Salcedo, C. and Whitley, D. (1999) Feature selection mechanisms for ensemble creation: a genetic search perspective. *To appear in: Freitas, A.A. (Ed.) Proc. AAAI-99 & GECCO-99 Workshop on Data Mining with Evolutionary Algorithms: Research Directions. (AAAI Technical Report, in press.)* Menlo Park, Calif.: AAAI.

Hsu, W.H; Pottenger, W.M.; Welge, M.; Wu, J.; Yand, T.-H. Genetic algorithms for selection and partitioning of attributes in large-scale data mining problems. *To appear in: Freitas, A.A. (Ed.) Proc. AAAI-99 & GECCO-99 Workshop on Data Mining with Evolutionary Algorithms: Research Directions. (AAAI Technical Report, in press.)* Menlo Park, Calif.: AAAI.

Hu, Y.-J. A genetic programming approach to constructive induction. Koza, J.R.; Banzhaf, W.; Chellapilla, K.; Deb, K.; Dorigo, M.; Fogel, D.B.; Garzon, M.H.; Goldberg, D.E.; Iba, H.; Riolo, R.L. (Eds.) *Genetic Programming 1998: Proc. 3rd Annual Conf.*, 146-151. San Mateo, Calif.: Morgan Kaufmann.

Janikow, C.Z. (1993) A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning* 13, 189-228.

Janikow, C.Z. (1995) A genetic algorithm for optimizing fuzzy decision trees. Eshelman, L.J. (Ed.) *Proc. 6th Int. Conf. Genetic Algorithms (ICGA-95)*, 421-428. San Mateo, Calif.: Morgan Kaufmann.

Kelly, J.D. and Davies, L. (1991) A hybrid genetic algorithm for classification. *Proc. 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, 645-650. Sidney, Australia. Morgan Kaufmann, San Mateo, CA.

Koza, J.R. (1992) *Genetic Programming: on the programming of computers by means of natural selection.* Cambridge, Mass.: MIT.

Koza, J.R. (1994) *Genetic Programming II: automatic discovery of reusable programs.* Cambridge, Mass.: MIT.

Koza, J.R.; Bennett III, F.H.; Andre, D.; Keane, M.A. (1999) *Genetic Programming III: Darwinian Invention and Problem Solving.* San Mateo, Calif.: Morgan Kaufmann.

Kuscu, I. (1998) Evolving a generalized behaviour: artificial ant problem revisited. Porto, V.W.; Saravanan, N.; Waagen, D.; and Eiben, A.E. (Eds.) *Proc. 7th Annual Conf. on Evolutionary Programming. Lecture Notes in Computer Science 1447.* Berlin: Springer-Verlag.

Liu, B.; Hsu, W.; Chen, S. (1997) Using general impressions to analyze discovered classification rules. Heckerman, D.; Mannila, H.; Pregibon, D.; Uthurusamy, R. (Eds.) *Proc. 3rd Int. Conf. Knowledge Discovery & Data Mining (KDD-97)*, 31-36. Menlo Park, Calif.: AAAI.

Marmelstein, R.E. and Lamont, G.B. (1998) Pattern classification using a hybrid genetic programming - decision tree approach. Koza, J.R.; Banzhaf, W.; Chellapilla, K.; Deb, K.; Dorigo, M.; Fogel, D.B.; Garzon, M.H.; Goldberg, D.E.; Iba, H.; Riolo, R.L. (Eds.) *Genetic Programming 1998: Proc. 3rd Annual Conf.*, 223-231. San Mateo, Calif.: Morgan Kaufmann.

Martin, L.; Moal, F.; Vrain, C. (1998) A relational data mining tool based on genetic programming. Zytkow, J.M. & Quafafou, M. (Eds.) *Principles of Data Mining & Knowledge Discovery: Proc. 2nd European Symp. (PKDD-98). Lecture Notes in Artificial Intelligence 1510*, 130-138. Berlin: Springer-Verlag.

Martin-Bautista, M.J. and Vila, M.A. (1999) A survey of genetic feature selection in mining issues. *To appear in Proc. Congress on Evolutionary Computation (CEC-99).* Washington D.C., USA. July 1999. IEEE, Piscataway, NJ.

Michalewicz, Z. (1996) *Genetic Algorithms + Data Structures = Evolution Programs.* 3rd Ed. Berlin: Springer-Verlag.

Michie, D.; Spiegelhalter, D.J.; Taylor, C.C. (1994) *Machine Learning, Neural and Statistical Classification.* New York: Ellis Horwood.

Neri, F. and Giordana, A. (1995) A parallel genetic algorithm for concept learning. Eshelman, L.E. (Ed.) *Proc. 6th Int. Conf. Genetic Algorithms (ICGA-95)*, 436-443. San Mateo, Calif.: Morgan Kaufmann.

Ngan, P.S., Leung, K.S.; Wong, M.L.; Cheng, J.C.Y. (1998) Using grammar based genetic programming for data mining of medical knowledge. Koza, J.R.; Banzhaf, W.; Chellapilla, K.; Deb, K.; Dorigo, M.; Fogel, D.B.; Garzon, M.H.; Goldberg, D.E.;

Iba, H.; Riolo, R.L. (Eds.) *Genetic Programming 1998: Proc. 3rd Annual Conf.*, 254-259. San Mateo, Calif.: Morgan Kaufmann.

Noda, E.; Freitas, A.A.; Lopes, H.S. (1999) Discovering interesting prediction rules with a genetic algorithm. *To appear in Proc. Congress on Evolutionary Computation (CEC-99)*. Washington D.C., USA, July 1999. IEEE, Piscataway, NJ.

Padmanabhan, B. and Tuzhilin, A. (1998) A belief-driven method for discovering unexpected patterns. Agrawal, R.; Stolorz, P.; Piatetsky-Shapiro, G. (Eds.) *Proc. 4th Int. Conf. Knowledge Discovery & Data Mining (KDD-98),* 94-100. Menlo Park, Calif.: AAAI.

Pei, M.; Goodman, E.D.; Punch, W.F. (1997) Pattern discovery from data using genetic algorithms. *Proc. 1st Pacific-Asia Conf. Knowledge Discovery & Data Mining (PAKDD-97)*. Feb. 1997. World Scientific.

Punch, W.F.; Goodman, E.D.; Pei, M.; Chia-Sun, L.; Hovland, P.; Enbody, R. (1993) Further research on feature selection and classification using genetic algorithms. Eshelman, L.E. (Ed.) *Proc. 5th Int. Conf. Genetic Algorithms (ICGA-93),* 557-564. San Mateo, Calif.: Morgan Kaufmann.

Raymer, M.L.; Punch, W.F.; Goodman, E.D.; Kuhn, L.A. (1996) Genetic programming for improved data mining -- application to the biochemistry of protein interactions. Koza, J.R.; Goldberg, D.E.; Fogel, D.B.; Riolo, R.L. *Genetic Programming 1996: Proc. 1st Annual Conf.*, 375-380. Cambridge, Mass.: MIT.

Ryan, M.D. and Rayward-Smith, V.J. (1998) The evolution of decision trees. Koza, J.R.; Banzhaf, W.; Chellapilla, K.; Deb, K.; Dorigo, M.; Fogel, D.B.; Garzon, M.H.; Goldberg, D.E.; Iba, H.; Riolo, R.L. (Eds.) *Genetic Programming 1998: Proc. 3rd Annual Conf.,* 350-358. San Mateo, Calif.: Morgan Kaufmann.

Ryu, T.-W. and Eick, C.F. Deriving queries from results using genetic programming. Simoudis, E.; Han, J.; Fayyad, U. (Eds.) *Proc. 2nd Int. Conf. Knowledge Discovery & Data Mining (KDD-96),* 303-306. Menlo Park, Calif.: AAAI.

Suzuki, E. and Kodratoff, Y. (1998) Discovery of surprising exception rules based on intensity of implication. Zytkow, J.M. & Quafafou, M. (Eds.) *Principles of Data Mining and Knowledge Discovery (Proc. 2nd European Symp, PKDD-98). Lecture Notes in Artificial Intelligence 1510,* 10-18. Berlin: Springer-Verlag.

Turney. P.D. (1995) Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligent Research*, 2, Mar. 1995, 369-409.

Vafaie, H. and DeJong, K. (1993) Robust feature selection algorithms. *Proc. 1993 IEEE Int’l Conf. on Tools with Artificial Intelligence,* 356-363. Boston, MA, USA, Nov. 1993. IEEE, Piscataway, NJ.

Further Reading.

Banzhaf, W.; Nordin, P.; Keller, R.E.; Francone, F.D. (1998) *Genetic Programming ~ an Introduction: On the Automatic Evolution of Computer Programs and Its Applications.* San Mateo, Calif.: Morgan Kaufmann. A comprehensible book on genetic programming, discussing both fundamentals and advanced topics in the area.

CEC-99. (1999) *Proc. 1999 Congress on Evolutionary Computation (CEC-99). (In press)* Washington D.C., USA. July 1999. These proceedings include recent research papers presented at a special session on data mining with evolutionary algorithms, chaired by Jan Zytkow.

Dhar, V. and Stein, R. (1997) *Seven Methods for Transforming Corporate Data into Business Intelligence.* Upper Saddle River, NJ: Prentice Hall. A pedagogical book discussing several data mining methods. One of the discussed methods is genetic algorithms.

Freitas, A.A. (1999) (Ed.) *Proc. AAAI-99 & GECCO-99 Workshop on Data Mining with Evolutionary Algorithms: Research Directions. (AAAI Technical Report, in press.)* Orlando, FL, USA. July 1999. Menlo Park, Calif.: AAAI. A collection of recent research papers on data mining with evolutionary algorithms.

Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, Mass.: Addison-Wesley. Although some parts of this book are out of date (it was written in 1989), it is a very pedagogical book and it is still a highly recommended reading.

Koza, J.R. (1992) *Genetic Programming: on the programming of computers by means of natural selection.* MIT Press. The pionering book on genetic programming, showing how to apply genetic programming in a wide range of problems. Recommended reading, as well as the second and third volumes, mentioned in the reference list - Koza (1994) and Koza et al. (1999).

Koza, J.R.; Banzhaf, W.; Chellapilla, K.; Deb, K.; Dorigo, M.; Fogel, D.B.; Garzon, M.H.; Goldberg, D.E.; Iba, H.; Riolo, R.L. (1998) (Eds.) *Genetic Programming 1998: Proc. 3rd Annual Conf.* San Mateo, Calif.: Morgan Kaufmann. These proceedings include several research papers on genetic programming for knowledge discovery and data mining.

Michalewicz, Z. (1996) *Genetic Algorithms + Data Structures = Evolution Programs.* 3rd Ed. Berlin: Springer-Verlag. Good introduction to genetic algorithms in general. It also includes one chapter on genetic algorithms for machine learning and rule discovery.