Context–free grammar induction using evolutionary methods

OLGIERD UNOLD
Institute of Engineering Cybernetics
Wroclaw University of Technology
Wyb. Wyspianskiego 27, 50-370 Wroclaw
POLAND

Abstract: The research into the ability of building self-learning natural language parser based on context–free grammar (CFG) was presented. The paper investigates the use of evolutionary methods: a genetic algorithm, a genetic programming and learning classifier systems for inferring CFG based parser. The experiments were conducted on the real set of natural language sentences. The gained results confirm the feasibility of applying evolutionary algorithms for context-free grammatical inference.

Key-words: Grammatical inference, context–free grammars, natural language processing, evolutionary computation

1 Introduction

The large amount of verbal data from common knowledge-elicitation methods suggests using knowledge acquisition by means of natural language processing systems (NLP systems). Additionally, knowledge acquisition via natural language much resembles man-man communication. To make possible wide use of NLP systems, we have to first equip the NLP system with the "smart" parser, which can be designed while working.

The problem of designing parsers (i.e. a correct grammars or equivalent automata) from actual sentences of the language is known as grammatical inference [7]. Grammatical inference is the gradual construction of a parser based on a finite set of sample expressions. In general, the training set may contain both positive and negative examples from the language under study. If only positive examples are available, no language class other than the finite cardinality languages is learnable [7].

It has been proved that deterministic finite automata are the largest class that can be efficiently learned by provable converging algorithms. There is no context-free grammatical inference theory which provable converges, if language defined by a grammar is infinite [32]. Building algorithms that learn context–free grammars is one of the open and crucial problems in the grammatical inference [9].

Many researchers have attacked the problem of evolving of (stochastic) CFG or equivalent pushdown automata [39, 38, 13, 5, 24, 21, 23, 14, 30, 16, 17], but very often for artificial languages (brackets, palindromes). The survey of the non-evolutionary approaches for context-free grammatical inference one can find in [22]. Here, we concentrate on applying three different evolutionary methods, i.e. a genetic algorithm, a genetic programming and learning classifier systems, for context-free grammatical inference of natural language sentences.

Section 2 discusses the evolutionary programming paradigms; section 3 states the motivation behind using evolutionary methods for context – free grammatical inference; the approach based on a genetic algorithm has been given in section 4; section 5 is on the genetic programming approach; section 6 describes the use of learning classifier systems in induction of context-free grammar, section 7 concludes the paper.

2 Evolutionary programming paradigms

Evolutionary algorithms are distinguished by the fact that they act on a population of potential solutions; they adapt an entire population of candidate solutions to the problem. These methods are based on biological populations and include selection operators which increase the number of better solutions in the population and decrease the number of the poorer ones, and other operators which generate new solutions. These methods differ in the standard representation of the problems and in the form and relative importance of the operations which introduce new solutions.

Genetic Algorithms (GA) were first proposed by Holland [10]. It is well known that the elements of genetic algorithms are initial population, crossover (emphasised by Holland), mutation, selection, reproduction and number of generations [8, 26]. In Evolutionary Programming (EP) Fogel [6] used selection and mutation to search over finite-state automata as solutions to a range of problems. Rechenberg [27] and Schwefel [29] developed a method called Evolutionary Strategies (ES) in which solutions to a problem were represented as real numbers. Koza [19] introduced Genetic Programming (GP), in which genetic algorithms are used to search parse trees of s-expressions. Learning

Classifier Systems (LCS) are a machine learning paradigm developed by Holland [11]. CLS use a GA to generate condition/action rules or classifiers which are evaluated during interaction with the problem environment.

These are the five so-called evolutionary programming paradigms.

3 Evolutionary context-free grammatical inference

Context–free grammar is a formal language grammar $G = (V_N, V_P, P, S)$ where the production rules are of the form: $A \rightarrow \alpha$ where $A \in V_N$ and $\alpha \in (V_T \cup V_N)^*$. Most work in inference of CFGs has focused on learning the standardized *Chomsky Normal Form* of CFG in which the rules are of the form $A \rightarrow BC$ or $A \rightarrow a$ where $A, B, C \in V_N$ and $a \in V_T$. Given a CFG G and a string $\alpha \in V_T^*$ we are interested in determining whether or not α is generated by the rules of G and if so, how is derived.

It is known, that there are very strong negative results for the learnability of CFGs. The main theorem is that it is impossible to identify context-free language in the limit if the data consists only of strings in the language being inferred [7]. Stochastic grammar induction tries to use distributional information to substitute for negative examples. Unfortunately, the "zero-frequency" problem entails assigning a small probability to all possible word patterns, thus ungrammatical n-grams become as probable as unseen grammatical ones. Due to their population-based approach, evolutionary methods are ideally suited for implementation in searching the right grammar, i.e. architecture of natural language parser (NLPa). Use of evolutionary methods can also bring to the grammar induction the ability to balance exploration and exploitation. The encouraging performance of evolutionary algorithms and their properties stimulated the present research.

4 NLPa inference by GA

Genetic algorithms are the search and optimisation techniques based on the "survival of the fittest" principle of natural evolution. Genetic algorithms – the probabilistic search technique – are powerful tools for exploring the multidimensional large search space with multimodality, discontinuity and noise.

The basic construction is to consider a population of individuals (the chromosomes) that each represents a potential solution to the given problem. The relative success of each individual on this problem is considered its fitness, and used to selectively reproduce fitter individual to produce similar but not identical offspring for the next generation. A set of genetic operators is applied to these offspring to make their genetic information different from their parents. By iterating this process, the population efficiently samples the spaces of potential individuals and eventually converges on the most fit.

A context-free grammar form was limited to Greibach normal form. This kind of representation of grammar is encoded as a three-dimensional matrix-chromosome. The left-hand symbols are represented by the first dimension. The options of the right-hand side of the rule are represented by the second dimension. The third dimension delimits the maximum length of the rules. The genetic operators were modified in the hope of better performance and adjusted to the grammar representation. The evaluation function evaluates the fitness value for each production rule proportionally to how often the rule was used, and which examples the rule let analyse.

The modified mutation factor for each rule is calculated on the bases of the classical mutation factor and the fitness value. The proposed selection operator is based on a roulette-wheel algorithm. Three main experiments for the natural languages: Polish, English and German were carried out. About 110 legal and 38 illegal examples of sentences (ratio about of three to one) were used for each language. The examples of sentences were taken from language-course-books for first grade foreign students. All the main experiments were carried out with the same values of genetic algorithm parameters. The probability of crossover and probability of mutation were linearly changed from the initial value until the final value throughout the experiment. Values of genetic algorithm parameters used in experiments: number of nonterminal symbols 14, number of terminal symbols 14, number of rules in each nonterminal 7, maximal length of rule 5 symbols, the size of population 200, generations 8000, the probability of crossover at the beginning of the experiment 0.4, the probability of crossover at the end of the experiment 0.6, the probability of mutation at the beginning of experiment 0.5, the probability of mutation at the end of experiment 0.001.

Figure 1 demonstrates the maximal fitness of a population in selected experiment with natural language sentences. After the fitness reached the level of 0.008 it kept this value for rest of the process for Polish and English language. Apparently the GA was driven to a local optimum from which it could not escape. Only the grammar derived from German sentences is able to find new niche.

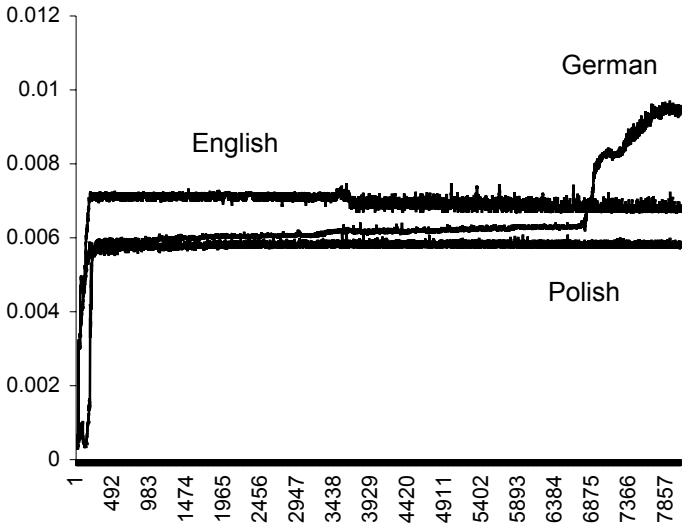


Fig. 1. The maximal fitness in the experiment with natural language sentences

5 NLPa inference by GP

A context-free grammar can be represented by a pushdown automaton [15]. The automaton serves both as an *acceptor* for the language (that is, it can decide whether or not any arbitrary sentence is in the language) and as a *generator* for the language (that is, it can generate any finite sentence in the language in finite time). The recent application of finite-state approach in natural language processing shows the usefulness of automata in this area of AI [18, 28].

Unold [33, 34, 35] proposed a pushdown automaton-based parser of natural language texts, so called PDAMS (nondeterministic PushDown Automaton with associative Memory accesS), and in [36] presented theoretical bases for the use of two classes of evolutionary computation, that is evolutionary programming and genetic programming, that support inference of automaton-driven parser of natural language.

Genetic programming [19] breeds a population of rooted, point-labelled trees with ordered branches. From here also, to be able to use this class of evolutionary computation in evaluating the architecture of parser, one should first find a suitable method of mapping the transition graph of PDAMS onto a tree structure. There are two encoding techniques that we can apply i.e. cellular encoding and edge encoding. Both methods rely in fact on operating on indirect structures, the so-called grammar-trees, which are subject to genetic programming process. Every structure represents a graph. The grammar-tree is a genotype and the PDMAS constructed in accordance with the tree's instruction is a phenotype.

Experimental assessment for the proposed approach has been done for several grammars, including formal and English grammar. Figure 2 demonstrates the average and maximal fitness of a population in selected experiment with an English adverbial group. The crossover probability of 0.4, mutation probability of 0.9, population size of 30.

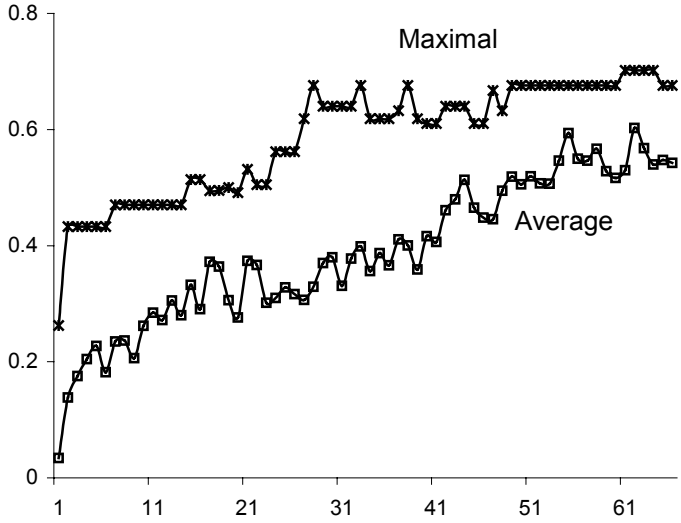


Fig. 2. The average and maximal fitness of a population with an English adverbial group

6 NLPa inference by LCS

Learning Classifier Systems (LCS) exploit evolutionary computation and reinforcement learning to develop a set of condition-rules (the classifiers) which represent a target task that the system has learned from experience. LCS learns by interacting with an environment from which it receives feedback in the form of numerical reward. Learning is achieved by trying to maximize the amount of reward received. There are many models of LCSs and therefore classifier systems (for example Wilson's XCS [37], Stolzmann's ACS [31] or Holmes's EpiCS [12]). All LCSs models comprise four main components: a finite population of condition-action rules (the classifiers); the performance component, which governs the interaction with the environment (the rule and message system); the reinforcement component (the credit assignment component), which distributes the reward received from the environment to the classifiers accountable for the rewards obtained; the discovery component, which is responsible for discovering better rules and improving existing ones through a genetic algorithm.

In the system mentioned, the population of grammars plays the role of system classifiers. Every classifier is of a form: *grammar_production: message*, where *grammar_production* is one of productions of the given

grammar, and *message* is a message sent to the list if the respective classifier reacts. Execution of a rule takes place when a production can parse a part of a sentence or a whole sentence that is currently being analysed.

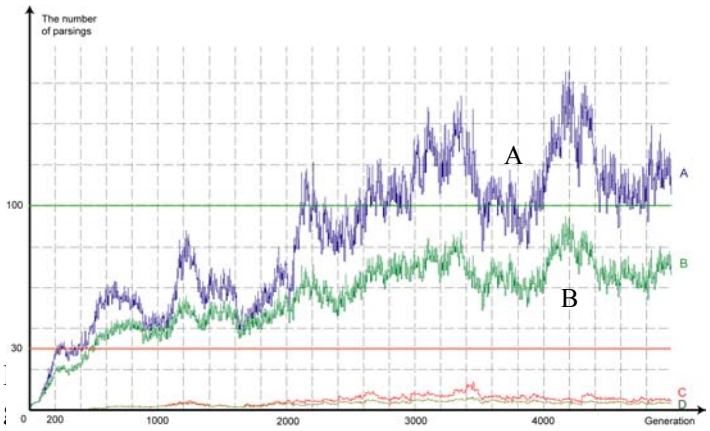
Every classifier has an assigned strength, whose task is to determine the respective classifier's / production's usability for the given grammar. In the beginning all classifiers have the same strength, but in the analysis process they receive positive or negative reward, which is the result of the actions they undertake.

Every sentence of the given natural language is analysed in turns by the whole population. The initial form of the grammars used by the system is randomly generated. The LCS has two aims. The first is checking the adjustment level of the grammars, that is the correctness of analyses of the given sentences in a natural language. The second is paying to the classifiers, or reducing the strength assigned to the classifiers, context-free grammar productions taking part in the analysis. The list of messages is the system memory storing the results of every classifier's action. The first item in the list is always a sentence sent by the system environment. The messages in the list are acted on by the classifiers in turns. These classifiers which are able to match the production to the particular sentence get the right to place their own message in the list. The list is compiled until none of the classifiers can execute its action, that is none of the production matches the sentence part being currently analysed. Every classifier placing its message in the list has to pay a conventional fee, being a part of its strength. The fee is then transferred to a classifier (or distributed among a greater number of classifiers) that placed in the message reinforcing the execution of the current rule. The classifier finishing a sentence analysis receives reward from the system environment. Moreover, all classifiers taking part in the full or partial analysis of a given sentence can be additionally paid a conventional number of strength points. The algorithm described concerns correct sentences. The only modification of the algorithm concerning incorrect sentences is the negative value of the reward.

Genetic algorithm takes the particular grammars as chromosomes, and the grammar productions as genes. Crossing and selection are carried out upon all the grammars (treated as production vectors), whereas mutation modifies (adds, deletes, replaces) singular symbols of the particular productions.

About a hundred of correct and thirty incorrect sentences were used in the experiments. The average adjustment strength for all grammar classifiers / productions, and the difference of the number of the analysed correct and incorrect sentences were taken as the fitness function.

Figure 3 illustrates the results of one of the numerous experiments conducted upon English sentences. Figure shows values of the analysed features averaged for all grammars in the given population. Graph A denotes the number of full parse paths performed in a single analysis cycle. Graph B denotes the number of correct sentences fully analysed by the grammars in a single analysis cycle. The number of sentences analysed correctly by the evolved grammars reaches 90%. The values of certain parameters were as follows: 5000 generations, 14 terminals, 8 nonterminals, maximally 16 rules for one nonterminal, 8 symbols in a rule, fitness function of the type 'number of correct sentences analysed by the grammar minus number of incorrect sentences analysed', size of the population 30, 3-point crossing, crossing probability 90%, mutation probability 1%, genetic algorithm operation every 10 cycles, reward for a full analysis of a correct sentence 40 points, reward for a partial analysis 25 points, negative reward for full analysis of an incorrect sentence 20 points, negative reward for partial analysis 10 points, 102 correct and 30 incorrect sentences in the learning set.



7 Conclusions

The goal of presented research is to build "smart" natural language parser based on CFG. This kind of parser acquires a language as a child – on the basis of the sentences that it encounters during the learning. This paper investigates the use of evolutionary methods: a genetic algorithm, a genetic programming and learning classifier systems for inferring CFG based parser. It is worth noticing, that the experiments were conducted on the real set of natural language sentences.

The first approach, in which a genetic algorithm was applied simply, is not very effective. Grammatical inference is a difficult problem for GA, due to the lack of natural building blocks in matrix-encoded grammars. Moreover, the adapted crossover operators shift the rule productions between the nonterminals. Crossover would take care of improving a grammar, but remember that a

grammar is a complex structure, and the particular parts of grammar (the rule productions) are only good in relation to other parts.

The best results were obtained in LCS approach. First of all, we have restricted the movement of the rule production during the crossover. Next, the generalization of LCS was exploited with success. Better results can be obtained by employing a greater population of grammars, than by increasing sizes of individuals, that is by increasing excessively the number of productions and their lengths.

The genetic programming approach based on the pushdown automaton shows great promise, although it still has few weak points. Proper inferring of the PDMAS parser relays in fact mainly on the proper definition of the edge encoding operators. For the simpler languages, as regular ones, the method and representation has been proven to be effective.

In summary, the results obtained confirm the feasibility of applying evolutionary algorithms for context–free grammatical inference, where CFG represents the natural language grammar.

References:

[1] D. Andre, F.H. Bennet III, J. Koza, M. Keane, On the Theory of Designing Circuits using Genetic Programming and a Minimum of Domain Knowledge, Proc. of the 1998 IEEE Congress on Computional Intelligence WCCI'98, Anchorge, Alaska, 1998, pp. 130-135.

[2] M. Chrobak, O. Unold, Natural Language Grammar Inference by Genetic Search, [in:] J.A. Meech at al. (eds.) Proceedings of the Third International Conference on Intelligent Processing and Manufacturing of Materials. IPMM - 2001, Canada, 2001, July 29 - August 3.

[3] G. Dąbrowski, Use of Classifier Systems in Natural Language Processing, M.Sc. - thesis, Wroclaw University of Technology, 2001 (in Polish).

[4] G. Dulewicz, O.Unold, Evolving Natural Language Parser with Genetic Programming, [in:] A. Abraham, M. Koppen (eds.) Advances in Soft Computing. Hybrid Information Systems, Physica-Verlag, Springer-Verlag Company, Germany, 2002, pp. 361-377.

[5] P. Dupont, Regular Grammatical Inference from Positive and Negative Samples by Genetic Search, Grammatical Inference and Application, Second International Colloquium ICG-94, Berlin, Springer, 1994, pp. 236-245.

[6] L.J. Fogel, A.J. Owens, M.J. Walsh, Artificial Intelligence through Simulated Evolution. John Wiley, New York, 1966.

[7] E. Gold, Language Identification in the Limit, Information Control, 10, 1967, pp. 447-474.

[8] D.E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, Massachusetts, 1989.

[9] C. de la Higuera. Current Trends in Grammatical Inference, [in:] F. J. Ferri et al. (eds.) Advances in Pattern Recognition, Joint IAPR International Workshops SSPR+SPR'2000, volume 1876 of LNCS, Springer, 2000, pp.28-31.

[10] J. Holland, Adaptation in Natural and Artificial Systems, MIT Press, 1975.

[11] J. Holland. Escaping brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems, [in:] R. S. Michalski, J.G. Carbonell, T.M. Mitchell (eds.) Machine Learning Vol.II, chapter 20, Morgan Kaufmann Publishers, Inc., 1986, pp. 593--623.

[12] J.H. Holmes, Evolution-Assisted Discovery of Sentinel Features in Epidemiologic Surveillance, Ph.D. – thesis, Drexel University, 1996.

[13] W. Huijsen, Genetic Grammatical Inference: Induction of Pushdown Automata and Context-Free Grammars from Examples Using Genetic Algorithms. M.Sc.- thesis, Dept. of Computer Science, University of Twente, Enschede, The Netherlands, 1993.

[14] T.E. Kammeyer, R.K. Belew, Stochastic Context-Free Grammar Induction with a Genetic Algorithm Using Local Search. Technical Report CS96-476, Cognitive Computer Science Rsearch Group, Computer Science and Engineering Department, University of California at San Diego, 1996.

[15] A. Kandel, S.C. Lee, Fuzzy Switching and Automata: Theory and Applications, Crane Russak, New York, 1979.

[16] B. Keller, R. Lutz, Learning Stochastic Context-Free Grammars from Corpora Using a Genetic Algorithm, Proceedings International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA-97), 1997.

[17] E.E. Korkmaz, G. Ucoluk, Genetic Programming for Grammar Induction, Proc. Of the Genetic and Evolutionary Conference GECCO-2001, San Francisco Ca, Morgan Kaufmann Publishers, 2001, pp. 180.

[18] A. Kornai (ed), Extended Finite State Models of Language, Cambridge University Press, Cambridge, 1999.

[19] J. Koza, Genetic Programming, MIT Press, Cambridge, MA, 1992.

[20] M.M. Lankhorst, Grammatical Inference with a Genetic Algorithm, [in:] Dekke L., Smit W., Zuidervaart J.C. (eds.) Proc. of the 1994

EUROSIM Conf. on Massively Parallel Procesing Applications and Development, Elsevier, Amsterdam, 1994, pp. 423-430.

[21] M.M. Lankhorst, A Genetic Algorithm for the Induction of Nondeterministic Pushdown Automata. Computing Science Reports CS-R 9502, Department of Computing Science, University of Groningen, 1995.

[22] L. Lee, Learning of Context-Free Languages: A Survey of the Literature, Harvard University Technical Report TR-12-96, 1996.

[23] R.M. Losee, Learning Syntactic Rules and Tags with Genetic Algorithms for Information Retrieval and Filtering: An Empirical Basis for Grammatical Rules, in Information Processing & Management, 1995.

[24] S. Lucas, Context-Free Grammar Evolution, [in:] First International Conference on Evolutionary Computing, 1994, pp. 130-135.

[25] S. Luke, L. Spector, Evolving Graphs and Networks with Edge Encoding: Preliminary Report, [in:] Koza J (ed.) Late-Breaking Papers of Genetic Programming 96, Stanford Bookstore, 1996, pp. 117-124.

[26] K.E. Man, K.S. Tang, S. Kwong, Genetic Algorithms: Concept and Design, Springer, 1999.

[27] I. Rechenberg, Evolutionsstrategie: Optimierung Technischer System nach Prinzipien der biologischen Evolution, Fromman-Holzboog, Stuttgart, 1973.

[28] E. Roche, Y. Schabes, Finite-State Language Processing, A Bradford Book, The MIT Press, Cambridge, Massachusetts, 1997.

[29] H. Schwefel, Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie, [in:] Interdisciplinary Systems Research vol. 26, Birkhauser, Basel, 1997, pp. 319-354.

[30] T.C. Smith, I.H. Witten, Learning Language Using Genetic Algorithms, [in:] S. Wermter, E.Rilo, G. Scheler (eds.) Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing, volume 1040 of LNAI, 1996.

[31] W. Stolzmann, An Introduction to Anticipatory Classifier Systems, [in:] Lanzi et al (eds.) Learning Classifier Systems: From Foundation to Application, Vol. 1813 of LNAI, Springer-Verlag, Berlin, 2000, pp. 175-194

[32] E. Tanaka, Theoretical Aspects of Syntactic Pattern Recognition. Pattern Recognition, 28(7), pp. 1053-1061, 1995.

[33] O. Unold, Automatic Analysis of Natural Language Texts in Man-Machine Communication [in:] Wojtkowski G. at al (eds.), Systems Development Methods for the Next Century, Plenum Publishing Corp., New York, 1997, pp. 185-193.

[34] O. Unold, A Fuzzy Automaton Approach to Dialog Systems, Proc. of the IASTED International Conference-ASC'98, Cancun, Mexico, May 1998, pp. 215-218.

[35] O. Unold, Toward Fuzziness in Natural Language Processing, [in:] Roy R at al [eds.] Advances in Soft Computing – Engineering Design and Manufacturing, Springer Verlag, London, 1999, pp. 554-567.

[36] O. Unold, An Evolutionary Approach for the Design of Natural Language Parser, [in:] Suzuki Y at al [eds.] Soft Computing in Industrial Applications, Springer Verlag, London, 2000, pp. 293-297.

[37] S.W. Wilson, Classifier Systems and the Animat Problem, Machine Learning 2, 1987, pp.199-228.

[38] P. Wyard, Context Free Grammar Induction Using Genetic Algorithms, [in:] R.K. Belew, and L.B. Booker (eds.) Proceedings of the Fourth International Conference on Genetic Algorithms, San Diego, CA. Morgan Kaufmann, 1991, pp. 514—518.

[39] H. Zhou, J. J. Grefenstette, Induction of Finite Automata by Genetic Algorithms, Proceedings of the 1986 International Conference on Systems, Man and Cybernetics, 1986, pp. 170-174.