Swarm and Evolutionary Computation 31 (2016) 11–23

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Maintaining regularity and generalization in data using the minimum description length principle and genetic algorithm: Case of grammatical inference

Hari Mohan Pandey [a,*], Ankit Chaudhary [b], Deepti Mehrotra [c], Graham Kendall [d]

[a] Department of Computer Science & Engineering, Amity University Uttar Pradesh, Sector 125, Noida, India
[b] Department of Computer Science, Truman State University, USA
[c] Amity School of Engineering & Technology, Amity University, Sector 125, Noida, India
[d] The University of Nottigham Malaysia Campus Jalan Borga, 43500 Semenyih, Selangor Darul Ehsan, Malaysia

ARTICLE INFO

Article history:
Received 23 April 2015
Received in revised form
4 May 2016
Accepted 16 May 2016
Available online 17 May 2016

Keywords:
Bit-masking oriented data structure
Context free grammar
Genetic Algorithm
Grammar induction
Learning algorithm
Minimum description length principle

ABSTRACT

In this paper, a genetic algorithm with minimum description length (GAWMDL) is proposed for grammatical inference. The primary challenge of identifying a language of infinite cardinality from a finite set of examples should know when to generalize and specialize the training data. The minimum description length principle that has been incorporated addresses this issue is discussed in this paper. Previously, the e-GRIDS learning model was proposed, which enjoyed the merits of the minimum description length principle, but it is limited to positive examples only. The proposed GAWMDL, which incorporates a traditional genetic algorithm and has a powerful global exploration capability that can exploit an optimum offspring. This is an effective approach to handle a problem which has a large search space such the grammatical inference problem. The computational capability, the genetic algorithm poses is not questionable, but it still suffers from premature convergence mainly arising due to lack of population diversity. The proposed GAWMDL incorporates a bit mask oriented data structure that performs the reproduction operations, creating the mask, then Boolean based procedure is applied to create an offspring in a generative manner. The Boolean based procedure is capable of introducing diversity into the population, hence alleviating premature convergence. The proposed GAWMDL is applied in the context free as well as regular languages of varying complexities. The computational experiments show that the GAWMDL finds an optimal or close-to-optimal grammar. Two fold performance analysis have been performed. First, the GAWMDL has been evaluated against the elite mating pool genetic algorithm which was proposed to introduce diversity and to address premature convergence. GAWMDL is also tested against the improved tabular representation algorithm. In addition, the authors evaluate the performance of the GAWMDL against a genetic algorithm not using the minimum description length principle. Statistical tests demonstrate the superiority of the proposed algorithm. Overall, the proposed GAWMDL algorithm greatly improves the performance in three main aspects: maintains regularity of the data, alleviates premature convergence and is capable in grammatical inference from both positive and negative corpora.

Abbreviations: ANS, Accepting negative sample; APS, Accepting positive sample; BMODA, Bit masking oriented data structure; BNF, Backus Naur Form; BBP, Boolean based procedure; CFL, Context free language; CFG, Context free grammar; CS, Chromosome size; CM, Crossmask/crossover mask; CR, Crossover rate; DFA, Deterministic finite automata; DL, Description Length; DSL, Domain-Specific Language; EA, Evolutionary algorithm; EMP, Elite Mating Pool Genetic Algorithm; GI, Grammatical inference; GA, Genetic algorithm with minimum description length; GAW, Genetic Algorithm without Minimum Description Length; GP, Genetic Programming; GA, Genetic algorithm; ITBL, Improved Tabular Representation Algorithm; M, Model; MM, Mutmask/mutation mask; MDL, Minimum description length; NN, Neural Network; MA, Memetic Algorithm; MR, Mutation rate; NPR, Maximum number of allowable grammar rules; PAC, Probably Approximately Correct; PRL, Production rule length; PDA, Pushdown automata; PS, Population size; RNN, Recurrent Neural Network; RNS, Rejecting negative sample; RPS, Rejecting positive sample; RL, Regular language; SOM, Self-organizing Map; SNR, Signal to noise ratio; TBLA, Tabular Representation Algorithm

* Corresponding author.
E-mail address: profharimohanpandey@gmail.com (H.M. Pandey).

http://dx.doi.org/10.1016/j.swevo.2016.05.002
2210-6502/

1. Introduction

The problem with inductive and statistical inference systems is to maintain regularity in the data. In other words "*How to take decisions for selecting an appropriate model that should present the competing explanation of the data using limited observations*?" Fig. 1 shows a scenario where a sender who want to transmit some data to the receiver and, is interested in selecting the best model which can maximally compress the observed data and deliver it to the receiver using as few bits as possible.

Formally, the selection of the best model is the process of deciding among the model classes based on the data. The *Principle of Parsimony* (Occam's razor) is the soul of the model selection, states that "*given a choice of theories, the simplest is preferable*" [4,5]. The purpose of implementing the *Parsimony Principle* is to find a model, which can best fit the data. Rissanen extracted the essence of the Occam's theory and presented the *Principle of Minimum Description Length* states that "*choose the model that gives the shortest description of data*" [4,12].

The domain of inquiry in this paper is the GI problem. A grammar can be constructed without using the MDL principle, but does not reflect any regularity in the data (Fig. 2(a)). In addition, it is difficult to know when to generalize and specialize the training data. In such situations, the constructed grammar is considered as a very simple grammar, because it simply provides the validity of any combination of words. Therefore the grammar does not show any regularity, hence a high amount of information is needed to specify them. In contrast, one can construct grammars that can list all possible sentences/corpus, but is not suitable for all sentences (Fig. 2(a)). Although, this type of grammar shows some sort of regularity, it fails to present any generalization, since it contains the information about each observed corpus, therefore it always exhibits poor performance and is assumed to be very complex.

The construction of a grammar using the MDL principle shows regularities in the data and also makes generalizations beyond the observed corpus (Fig. 2(b)). Therefore, the MDL principle behaves as a middle level and fills the gaps presented in Fig. 2(a). Bayes theorem can be used to derive the MDL principle, but the working of the MDL principle is not similar to the Bayes theorem since the MDL principle uses code length rather probabilities [4,12,54]. The MDL principle was used widely in the GI problem [5,13–17,55].

Several approaches have been attempted for the GI (see Section 2). This paper presents a modified GA based approach that utilizes the MDL principle for generating an appropriate number of corpuses (positive and negative) to present the language feature. A GA is a search and optimization algorithm based on natural selection and genetics. The GA is one of the most popular algorithms from the class of EAs. The basic principles of the GA's were initially developed by Holland [1] and further carried by De Jong [17] and Goldberg [2]. Goldberg and Michalewicz have presented a detailed overview of the GA in various fields [2,11]. A GA works with a population of solutions represented by some encoding mechanism. During the implementation of a GA every solution or individual is assigned a fitness value, which is the measure of the quality of the solution. The fitness of an individual is directly related to an objective function of the optimization problem. Then, using the reproduction (crossover and mutation) operators an individual population can be modified to a new one. In GAs, the search for an optimum is iteratively guided by the fitness of the current generation. Whenever, a researcher applies a GA for an optimization problem, it generates thousands of individuals, each representing a solution. The obtained solutions are evaluated and recombined to get an offspring. It has been shown in [1,2,11,55,56] that the previous generations details are only implicitly and partially preserved in the current generation. Hence, the regeneration is hard to manage [30,73]. GAs have gained popularity due to the applicability to a wide range of problems, including multimodal function optimization, machine learning, pattern recognition, image processing, natural language processing and grammar induction [8,23].

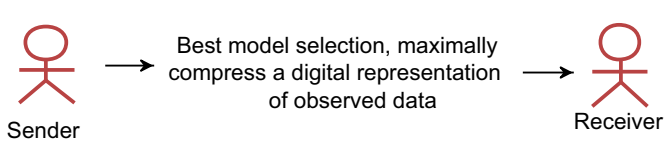

Fig. 1. A scenario showing the rationale of using the MDL principle. The sender wants to transmit some data to the receiver.

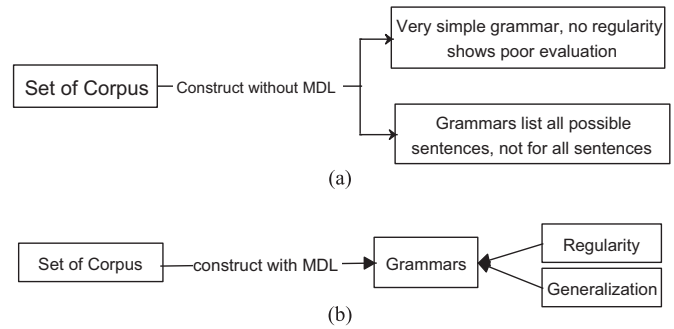

Fig. 2. The MDL principle as a middle level for the grammatical construction.

The domain of inquiry in this paper is the GI problem. Grammar induction poses many theoretical problems, as "*learning of CFGs is much harder than learning DFA*" [57]. As an implication of the work presented in [19], learning algorithms have been developed that exploit knowledge of negative samples, structural information, or restrict grammars to some subclasses such as linear grammars, K-bounded grammars, structurally reversible languages and terminal distinguishable CFLs [57]. Previous research [58–60] shows that few classes of CFLs are polynomial time identifiable in the limit from the positive samples only. Another issue in GI is the immense search space, where an exhaustive approach is not feasible [61].

Therefore, a different and more efficient approach to explore the search space is needed, which identifies the regularity in the data and simplifies the representation (handles the huge number of grammar rules). The GI approach implemented in this paper applies a modified GA with the MDL (GAWMDL) principle that combines with BMODS to apply reproduction operators. It utilizes BBP for breeding in the next generation. The key benefit of implementing BBP is that it introduces diversity into the population, which helps to alleviate premature convergence (a situation when the diversity of the population decreases, leading to an unwanted convergence and produces a solution which is far from the best solution). The MDL principle that is incorporated supports two different operations, namely merge and constructs. These two operations, reduce the burden of handling a large number of grammar rules. In addition, the MDL principle allows the system not to overestimate and it generates samples that are sufficient to acquire the basic properties of the language. These features help the proposed GA to converge. The computational experiments have been conducted on a set of corpus (positive and negative) of RLs and CFLs. The robust experimental environment is developed to perform the experiments. The results have been collected and tested against three algorithms are: GAWOMDL, EMPGA [18] and ITBL [51–53]. The primary objective of comparing the proposed GA with EMPGA and ITBL is both of these algorithms were proposed for the CFG induction using the GA. Evidence is available proving that the EMPGA handles the situation of the premature convergence successfully [18]. The computational results demonstrate

that the proposed GA outperforms the other algorithms (GAWOMDL, EMPGA and ITBL). Statistical tests are used to determine the significance of the proposed GAWMDL. The paired *t*-test has been conducted creating three pairs: GAWOMDL-GAWMDL, EMPGA-GAWMDL and ITBL-GAWMDL. The results of the paired *t*-test concludes that the proposed GAWMDL is statistically significant when compared to two algorithms.

The rest of the paper is organized as follows: Section 2 presents the background and related work in the GI with pros and cons of existing approaches. The authors discuss the role of the MDL principle and its connection with the statistical modeling in Section 3. The proposed GAWMDL for the GI is discussed in Section 4. A flow chart of the proposed GAWMDL is presented to demonstrate the overall procedure of the GI and the use of the MDL (role of merging and construct) principle. An example is presented representing the suitability of the MDL principle in the GI and how the GA helps in optimizing the solution. The experimental details, parameters tuning, observations, results, discussion and statistical tests are given in Section 5 followed by the concluding remarks for the paper in Section 6.

2. Background and related work in grammar induction

The GI or grammar learning deals with idealized learning procedures for acquiring grammars on the basis of the evidence about the languages [31,48,49]. It was extensively studied [6,32–37,49] due to its wide fields of application to solve practical problems in a variety of fields, including compilation and translation, human machine interaction, graphic languages, design of programming language, data mining, computational biology, natural language processing, software engineering and machine learning etc.

The first learning model was proposed by Gold [19]. Gold addressed the question "*Is the information sufficient to determine which of the possible languages is the unknown language?*" [19]. It was shown that an inference algorithm can identify an unknown language in the limit from the complete information in a finite number of steps. The key issue with the Gold's approach is that there is not sufficient information present within the inference algorithm to identifies the correct grammar because it is always possible that the next sample may invalidate the previous hypothesis. Angluin [44] has proposed "*tell tales*" (a unique string highlighting the differences between languages) to avoid the drawback of the Gold's model. Although, Gold [19] laid the foundation of the GI, Bunke and Sanfeliu [27] have presented the first usable GI algorithm in the syntactic pattern recognition community with the aim of classify and analyzing the patterns, classifying the biological sequence, and for character recognition, etc. The main drawback of this algorithm is that it only deals with positive data, and is not to deal with noisy data, does not fit exactly into a finite state machine and therefore good formal language theories were lost.

Stevenson and Cordy [28,29] explains theorists and empiricists are the two main groups contributing in the field of GI. Language classes and learning models were considered by the theorists group to set up the boundaries of what is learnable and how efficiently it can be learned. On the other hand, the empiricists group dealt with a practical problem by solving it; finally they have made significant contributions in the GI.

The teacher and query is another learning model, where a teacher, also referred as an oracle knows the target languages and is capable of answering the particular type of questions/queries from the inference algorithm. Six types of queries were described by Angluin [45], two of which are membership and equivalence queries, and having a significant impact on learning. In case of membership queries, the inference algorithm presents either "*yes*" or "*no*" as an answer to the oracle, whereas an oracle receives "*yes*" if the hypothesis is true and "*no*" otherwise by the inference algorithm. Valiant [46] has presented the PAC learning model, which takes the advantages of both the identification of the limit and the *teachers and queries* learning models. The PAC learning model is different from the other two learning models for two reasons: first, it does not guarantee exact identification with certainty; second, compromise between accuracy and certainty. The problem with the PAC model is that the inference algorithm must learn in polynomial time under all distributions, but it is believed to be too strict in reality. These problems occur because many apparently simple classes are either known to be NP-hard or at least not known to be polynomial learnable for all the distributions [29]. To mitigate this issue, Li et al. [47] has proposed an inference algorithm that considers the simple distribution only.

Apart from the above popular learning models, many researchers have explained the suitability of the NN for the GI. The NN has shown the ability to maintain a temporal internal state like a short term memory [29]. In case of the NN, a set of inputs and their corresponding outputs (Yes: string is in the target language, No: otherwise) and a defined function needs to learn, which describes those input–output pairs [20]. Alex et al. [40] has conducted experiments for the handwriting recognition using a NN and the NN has the ability to predict subsequent elements from an input sequence of elements. Cleeremans et al. [39] has implemented a special case of a recurrent network presented by Elman [41], known as a simple RNN, to approximate a DFA. Delgado and Pegalajar [42] have presented a multi-objective GA to analyze the optimal size of a RNN to learn from the positive and negative examples. The merits of the SOM have been used to determine the automation, after the completion of the training process. Although, the NN has been widely used for the GI, as it was found good at simulating an unknown function, but it was found less effective because there is no way to reconstruct the function from the connections in a trained network [29].

A detailed survey of various GI algorithms is presented in [6,29,30,38,39,43,44]. The inductive inference is the process of making a generalization from the input (string). Wyard [3] has presented the impact of the different grammatical representation and the experimental result shows that the EA uses standard CFG in BNF has outperformed the others. Thanaruk and Okumaru [20] have classified the grammar induction methods into three major categories, namely; supervised, semi-supervised and unsupervised on the basis of the type of required data. Javed et al. [21] presented a GP based approach to learning the CFG. The work presented in [2] was an extension of the work conducted in [3] applying the grammar specific heuristic operator. In addition, a better construction of the initial population was suggested. Choubey and Kharat [22] have presented a sequential structuring approach that performs coding and decoding of the binary coded chromosomes into terminal and non-terminals and vice-versa. A CFG induction library was presented using the GA, which contains various Java classes to perform the GI [8,23]. Hrncic and Marjan [61,62] have implemented a MA for the GI that assists the domain experts and software language engineers to develop the DSLs by automatically producing a grammar. Hrncic et al. [63] has proposed an unsupervised incremental learning algorithm using a MA for the DSLs. The authors [74] have proposed a GI approach known as MAGIc (based on the MA), to extract grammars from DSL examples.

Sakakibara and Kondo [51] have proposed a GA for learning the CFG from a finite sample of positive and negative examples. The authors [51] have used a table similar to the parse table that reduces the partitioning problem of non-terminal and then the GA has been applied to solve the partitioning problem. Jaworski and Unold [52] have brought some improvement, which involve:

initial population block size manipulation, block deletes specialized operator and modified fitness function and experimentally proved that the TBLA is not vulnerable to block size and population size, and the ITBL is capable of finding the solutions faster. Bhalse and Gupta [53] have applied the ITBL for the GI.

3. Minimum description length principle

The theory of induction [64,65] says that under the right circumstances learning is "*finding a shorter description of the observed data*". the MDL principle suggests choosing the model, which provides the shortest description of data [4]. it works on coding rather on probability. hence, the focus is about casting a statistical model as a means of generating code, and resulting code lengths. the MDL principle has connections with more traditional frameworks given for the statistical estimation. in classical terms, we intend to estimate the parameter θ of a given model.

$$M = \left\{ f(x^n|\theta) \colon \theta \in \Theta \subseteq \Re^k \right\} \tag{1}$$

Eq. (1) is based on observations $x^n = (x_1 \ldots\ldots x_n)$. The aim is to choose $\hat{\theta}$ to maximize $f_\theta(x^n)$ over $\theta \in \Theta$. According to the maximum likelihood principle $\hat{\theta}$'s asymptotic efficiency in the form of repeated sampling under some regularity and handled by Cramer–Rao information lower bound theory in the finite sample case. From a coding point of view, both sender and receiver know which member f_θ of the parametric family M generated a data string x^n is simply $-\log_2 f_\theta(x^n)$, since on average code based on f_θ, achieve entropy lower bound. The noticeable thing is minimizing $-\log_2 f_\theta(x^n)$ is the same as maximizing, therefore the MDL principle coincides with the maximum likelihood principle in parametric estimation problems. The MDL principle enjoys all the desirable features of the maximum likelihood principle. In case of modeling, one has to transmit θ, as receiver did not know its value in advance. Adding in this case, we get a code length of the data string x^n using Eq. (2).

$$MDL = -\ \log f_\theta(x^n) + L(\theta) \tag{2}$$

Now, if the term $L(\theta)$ is constant, then the MDL principle needs a model, which minimizes $-\log f_\theta(x^n)$ among all the densities in the family. The maximum likelihood principle breaks down when one is forced to choose among nested classes of parametric models. This occurs most noticeably in variable selection for the linear regression.

4. Grammatical inference using GA and the MDL principle

The input for the algorithm is a set of corpus $C_1^L = \{c_1, c_2, \ldots c_i \ldots, c_L\}$. L is the total length of the corpus, c_i indicates the ith string of the corpus set, for each i, $1 \le i \le L$. The proposed GA tries to infer a grammar rule. A partial grammar G is defined that contains a set of CFG rules for the training data. G can be described in a somewhat nonstandard way as a set of classes. For every class g, exactly one corresponding non-terminal g' is present, which is the set of grammar rules with this non-terminal on the left hand side of the production rules. Two basic operations have been performed. First, merge or merge for shorting the production rules. Second, the construction operation, which construct for shorting the production rules. If two production rules are merged, then they have been removed from the G and replaced by a new production rule. The new production rule would be obtained by taking the union of the existing grammar rules. For example, suppose $g_1' = \{g_1' \to g_2' g_4'/g_3'\}$ and $g_8' = \{g_5' \to g_7'\}$ are two production rules belongs to G. Now, if g_1' and g_8' are merged, it produces a new production rule $g_{new} = \{g_1' \cup g_8'\} = \{g_{new}' \to g_2' g_4'/g_3'/g_7'\}$ and we would remove g_1' and g_8' from G. Re-indexing is done at this stage to incorporate g_{new}. Merging of production rules is found effective and yields better result by decreasing the number of classes. On the other hand, if g_l and g_k are two classes, then a new class g_{new} is created, which contains just one production rule $g_{new} = \{g_{new}' \to g_l' g_k'\}$. The working of MDL principle is used for the GI shows these two operations are represented in a separate block in Fig. 3.

In order to define a DL for each $c_i \in C_1^L$, a system generated code is employed, which uses a unique representation for each training data. Dense code is set, i.e., a sequence of code words which defines a training data [65]. The reason of doing this is that we are interested in representing G in the form of code, but the information theory explains that to arrive at an ideal code (shortest description of training data), one need to keep track of the frequencies of occurrence of the training data in classes belongs in G. The two operations (merge and construct) are useful reduces the DL.

4.1. Genetic algorithm adapted

Pandey et al. [8] has presented a GA for CFG induction uses the simple 1-point and 2-point crossover and a bit inversion mutation operator to introduce diversity during the execution of the GA. The authors [7,23] proposed a Java based library for the GI that utilizes the GA. The algorithm implemented in [7,8,23] works successfully for the relatively simple and deterministic CFG induction, but has been found not to work for the complex corpus. In addition, these approaches were not focused towards handling premature convergence in the GA.

In this paper, we have implemented an algorithm, GAWMDL, for the CFG induction. The proposed GAWMDL is different from the other approaches as it uses BMODS to perform the reproduction operations [10]. The breeding process is also different from the previous approaches as the proposed GAWMDL incorporates BBP which uses Boolean based operators (substep-3 in Fig. 3), which not only generates the new offspring, but also alleviates the risk of premature convergence [30] by introducing diversity into the population. The proposed GAWMDL algorithm uses the merit of the MDL principle ad maintains the regularity and generalization in the training data according the DL (Fig. 3).

The e-GRIDS learning model also uses the MDL principle for the generalization and specialization of the training data [50]. The e-GRIDS model is based on a beam search, which starts constructing the initial grammar for each input sentence and then applies the e-GRIDS learning operators, which include *MergeNT*, *CreateNT* and *Create OptionalNT*. The workings of these operators are discussed in [50]. The key drawback of the e-GRIDS learning model are: it is not fit for the negative examples, the beam search has been used in the learning process uses three operators as discussed above, but implementing these operators and collecting the temporary results makes it ineffective.

The proposed GAWMDL algorithm is more powerful as it is able to deal with both positive and negative training data. The MDL principle increases the effectiveness of the proposed algorithm as it supports generalization and specialization of the training data. The training set and test set are required for the learning has been generated by the length L (or DL) (L=0, 1, 2, ……) such that it covers all the possible valid strings of length L until a sufficient number of the valid strings of corpus have been generated. The invalid strings generated during this process are considered as negative strings.

The flow chart of the proposed GAWMDL have BMODS and MDL principle for the CFG induction is presented in Fig. 3. Step 2 demonstrates the process of GI and verification of production rules. The process of the GI begins applying the mapping of the

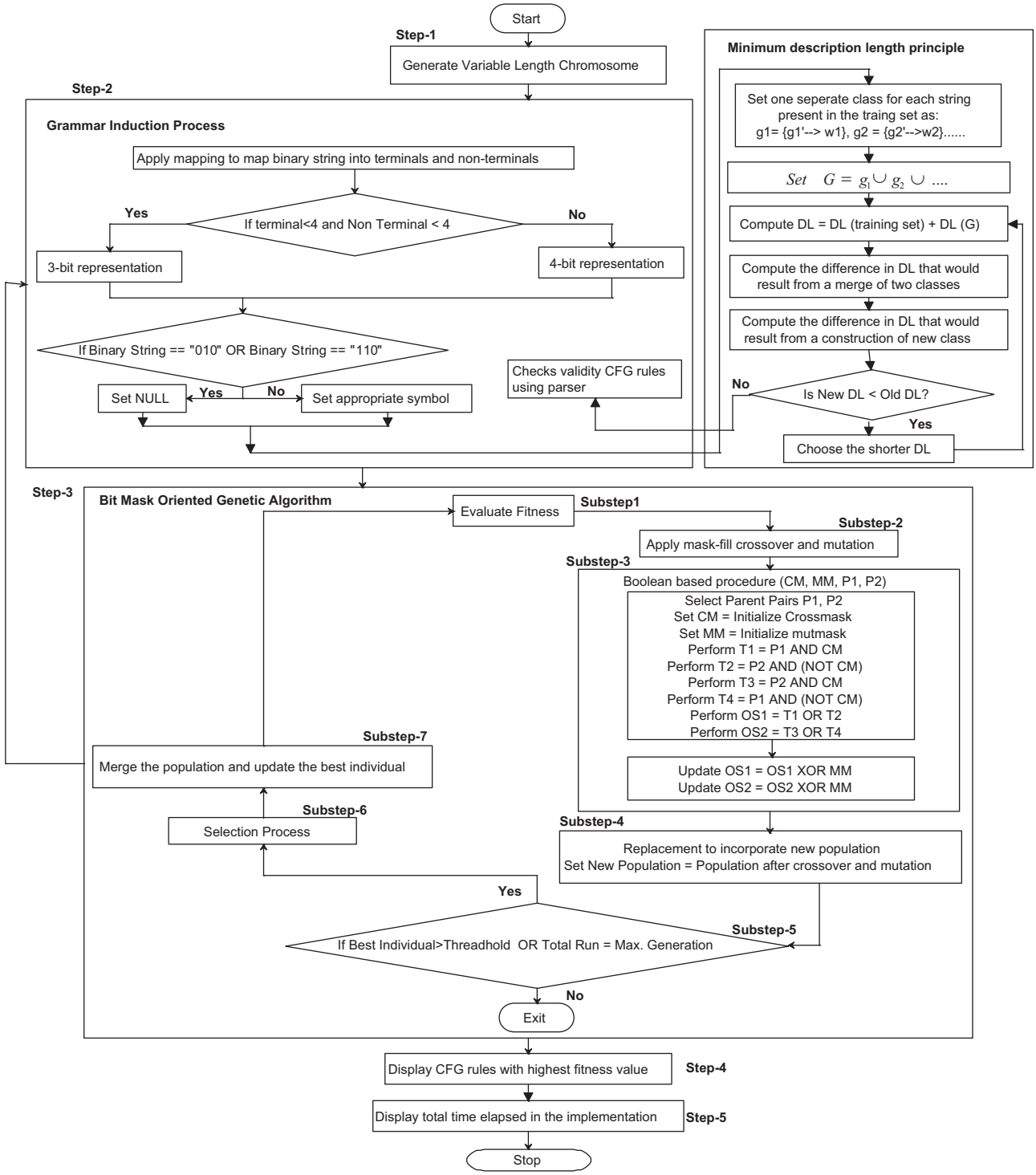

Fig. 3. Grammatical inference using GA and MDL principle. CM: crossmask, MM: mutmask, T1, T2, T3, T4: Temporary variables, OS1, OS2: offspring, DL: Description length, G: Partial grammar set, g: Grammar class, P_1, P_2: Parents.

binary strings into terminals and non-terminals [3,7,8]. We have used 3-bit/4-bit representation of the mapping. This being decided based on the number of symbols present in the input language (3-bit representation has been used in Fig. 4, since two symbols (0 and 1) are used).

During the mapping process, if the string “010” or “110” is encountered, set null (ε). After the completion of the mapping process, the process of the construction of the CFG starts with the start symbol ‘S’ mapped at “000”. The symbolic representation contains the block size of five equal to the PRL (PRL=5).

The symbolic grammar is traced from ‘S’ to terminal to remove useless productions and the remaining production rules are tested for the removal of left recursion, unit production, ambiguity and left factor. During the grammar rule generation, the MDL principle is used in generating the code for the grammar and to perform operations: merging and construct to reduce the complexity (see Section 4).

The string to be tested from the selected sample set is taken as an input with the CFG rules are passed to the finite state controller that verifies the acceptability through proliferation on the PDA. In the EA, an individual chromosome survives based on its fitness

Mapping process for palindrome over (0 + 1)*			
Step-1: Binary Chromosome of size 120 (initial random population)			
000100010000010010000101001111000101000110010000010011101011001000011001001110101010001100000100010110110000001101101110			
Step-2: Symbolic chromosome mapping (3 bit representation)	S1?S??S0ABS0S??S?C0CASCAA?0?A1S1???SA00?		
Generation of CFG: create a block size of five equal (chosen for experiment)			
000|100|010|000|010 010|000|101|001|111 000|101|000|110|010 S1?S? ?S0AB S0S?? 000|010|011|101|011 001|000|011|001|001 110|101|010|001|100 S?C0C ASCAA ?0?A1 000|100|010|110|110 000|001|101|101|110 S1??? SA00?	Maximum 8 grammar rules can be derived Mapping of non-terminals and terminals: Non-terminals: S→ 000 A→001 B→111 C→011 Terminals: 1→100 0→101 ?→010 ?→110 ? represents null ($\mathcal{E}$)		
Final Rules after removing useless productions, left recursion, unit production, ambiguity and left factor	S→1L S→0S L→S L→?		NPR = 4

Fig. 4. Demonstration of step-2 of the algorithm (coding and decoding mechanism adapted).

value [2,9,70,71,72]. In case of the GI problem, the fitness value of an individual chromosome largely depends on the acceptance or rejection of positive and negative sample respectively. A total of four cases are possible that affect the fitness value: an increase in fitness value for APS and RNS and a decrease for ANS and RPS. The NPRs have also shown a considerable impact on the fitness value, hence is considered to determine the fitness value. Eq. (3) has been used to evaluate the fitness of each population.

$$\text{Fitness} = \sum K^*((\text{APS} + \text{RNS}) - (\text{ANS} + \text{RPS})) + (2^*K - \text{NPR}) \tag{3}$$

S.T.
ANS + RNS ≤ Number of positive samples in corpus data
ANS + RPS ≤ Number of negative samples in corpus data
NPR: maximum number of allowable grammar rules
K: constant

Computing Fitness: suppose the CS = 120, which derives a maximum 8 grammar rules (Fig. 4). In the present scenario, 25 positive and 25 negative sample strings are found sufficient to generate the best possible production rules. In an ideal situation, we have assumed that the system is not rejecting any positive strings and not accepting any negative sample strings, then the value of ANS = RPS = 0. In the example that presented in Fig. 4, the value of NPR = 4 is considered. K = 10 is a constant, taken so that the grammar has less production rules with high fitness value can be created.

Putting these values into Eq. (3), we get 516 ((10*(25+25)−(0+0))+(2*10−4)), which is the fitness value in the first generation. At this stage, evolutionary operators (crossover, mutation and selection) are executed. The important thing to note here is, $K = 10$ is considered to conduct the experiment and any increase in K, would lead to high value of fitness by that factor. But with CS = 120, only 8 grammar rules can be extracted. Further, substitution/break for the removal of left recursion and other preprocessing leads to at most an additional 4–5 rules. Therefore, $K = 10$ (i.e. $2^*K = 20$) (from Eq. (3)) is considered that differentiate between various grammar based on the number of rules. As discussed, an increase in K will produce high fitness values, but it will be just for the sake of increasing the fitness value and not for representing the difference between various grammars. Hence, $K = 10$ is sufficient in this process to determine the optimum production rules. If the CS is increased to produce more grammar rule, a higher value of K might be needed.

Step-3 shows the main functions of the proposed GAWMDL. It utilizes BMODS [10] to improve the capability of the crossover and mutation operations, replaces various algorithms and codifies specialized rules of mating, supports a formal separation between searching for a proper bit composition and an effective achievement (using the mask for crossover and mutation) of the offspring. Previous research signifies that a binary code based GA can be grouped into an explicit and implicit binary formulation [11]. On the other hand, in a bit masking scheme, there is no need to use an explicit data structure, since only high level operations, working on integer values are mapped into a discrete representation domain are executed. Iuspa [10] has presented a detailed description about the construction of BMODS. Two integer arrays known as CM and MM are used to perform crossover and mutation.

For the creation of BMODS an integer genome array has been formed, where a set of integer values are linked with the design variables. The binary image has been used to represent the masks and is used to generate the CM and MM. The following convention has been used to represent a binary image for the CM: *high value, i.e. one or true for the current image bit is a pointer to the first parent while low value i.e. zero or false is a pointer to the second parent*. Similarly, for the MM an integer sequence has been used that indicates its binary image using the following convention: "*if the pointed bit of the target string has to be inverted (i.e. high value) or not (i.e. low value)*". In order to create a generic child individual a vector function $f(P_1, P_2, \text{CM}, \text{MM})$ has been used takes four arguments: P_1, P_2, CM and MM.

The implementation of BMODS for any real life problem is a two-step process: first apply crossover and mutation mask-fill operation and then apply mask application on the selected parent strings. Three crossovers (cut crossover, bit-by-bit and local cut) and a mutation (mutation mask-fill: similar to an inverted mutation has been applied based on a specific mutation rate) operations are applied as suggested in [10].

At substep-2 and 3, mask-fill reproduction operators are applied and then BBP. The key challenge in applying a GA is how to handle premature convergence. BBP is able of introduce diversity in the population in a generative manner that helps to avoid premature convergence.

The process of generating a new offspring takes place at substep-3. Two parent strings have been selected using roulette wheel selection technique for the GAWMDL. Two complementary child vectors are generated applying Eq. (4).

$$\begin{aligned} \text{OS}_1 &= f_1(P_1, P_2, \text{CM}, \text{MM}) \\ \text{OS}_2 &= f_2(P_1, P_2, \text{CM}, \text{MM}) \end{aligned} \tag{4}$$

where OS_1, OS_2, P_i and f_i ($i = 1, 2$) are respectively the offspring, parent vectors and a Boolean function that has been used to determine the assembly style of a new chromosome.

The arguments CM and MM are used to determine a suitable crossover operator (cut crossover, bit-by-bit and local cut) and mutation rule (mutation mask-fill). For the sake of simplicity Eq. (4) can be converted into a new form to show both crossover and mutation operations separately. Eq. (5) represents the crossover vector and a binary image that allows P_1 or P_2 to a child bit transfer

P1	1	0	0	1	0	0	0	0	1	0	1	1	1	0	1	1	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0
P2	1	1	0	1	0	1	1	0	1	1	0	1	1	0	0	1	0	1	0	1	0	1	1	1	0	1	0	1	0	0	1	1
CM	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
																		T1 = P1 AND CM														
T1	1	0	0	1	0	0	0	0	1	0	1	1	1	0	1	1	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0
																		T2 = P2 AND (NOT CM)														
T2	1	0	0	1	0	0	0	0	1	0	1	1	1	0	1	1	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0
																		OFFSPRING1= T1 OR T2														
OS1	1	0	0	1	0	0	0	0	1	0	1	1	1	0	1	1	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0
MM	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
																		OFFSPRING AFTER MUTATION = OFFSPRING1 XOR MM														
OS1	1	0	0	1	0	1	0	0	1	0	1	1	1	0	1	1	1	1	0	1	0	1	0	0	0	1	0	0	1	1	1	1
OS2	1	1	0	1	0	0	1	0	1	1	0	1	1	0	0	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0

P1: PARENT1, P2: PARENT2, CM: CROSSMASK, T1, T2: TEMP VARIABLE, OS1: OFFSPRING, MM: MUTMASK

Fig. 5. Demonstration of a new offspring generation after applying genetic reproduction of the GAWMDL.

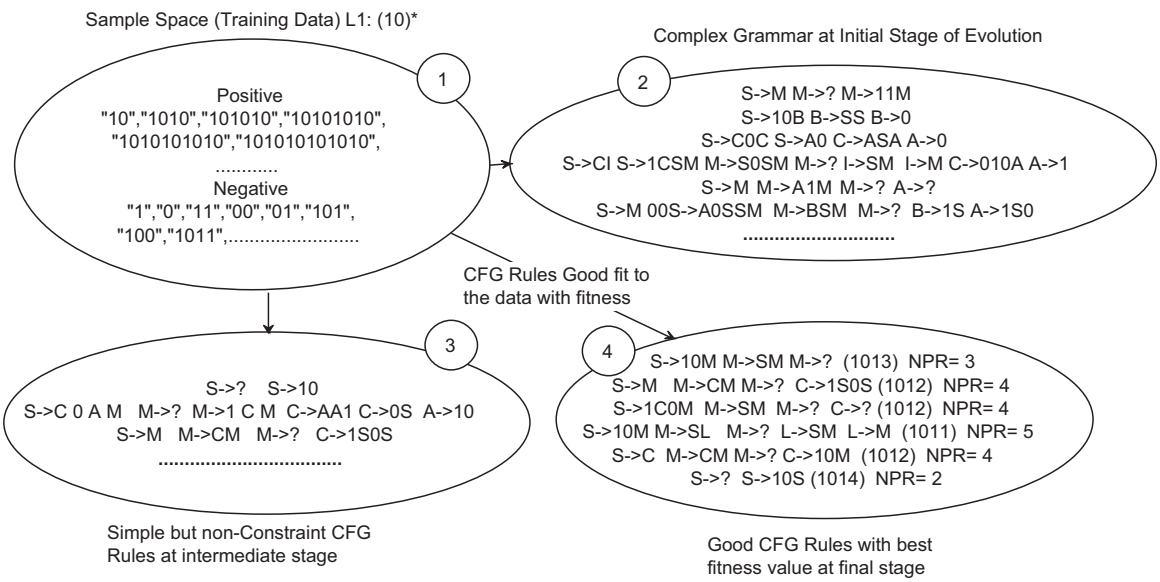

Fig. 6. Demonstration of MDL principle (for L1 = (10)*) which says that "more we are able to compress the data implies that we learned more" (NPR: Number of production rules).

Table 1
Test Languages.

L-id	Language description	Standard sets
L1	All strings not containing '000' over $(0+1)^*$.	Tomita [25]/Dupont set [26]
L2	0^*1 over $\{0+1\}^*$.	Dupont set [26]
L3	$(00)^*(111)^*$ over $\{0+1\}^*$.	–
L4	Any String with even 0 and odd 1 over $\{0+1\}^*$.	–
L5	$0(00)^*1$ over $\{0+1\}^*$.	–
L6	All strings with even number of 0 over $\{0+1\}^*$.	–
L7	$(00)^*10^*$ over $\{0+1\}^*$.	–
L8	Balanced Parentheses Problem.	Huijsen [24]/Keller and Lutz set [5]
L9	$\{0^n1^n, n \geq 0\}$ over $\{0+1\}^*$.	Keller and Lutz set [5]
L10	$\{0^n1^{2n}\ n \geq 0\}$ over $\{0+1\}^*$.	Dupont set [26]
L11	Even Length Palindrome over $\{a, b\}^*$	Huijsen [24]/Keller and Lutz set [5]
L12	$(10)^*$ over $(0 + 1)^*$	Tomita [25]/Dupont set [26]
L13	Odd binary number ending with 1	Dupont set [26]

according to the correlated CM value.

$$OS_1 = (P_1 \text{AND CM}) \text{OR} (P_2 \text{AND} (\text{NOT CM}))$$
$$OS_2 = (P_2 \text{AND CM}) \text{ OR} (P_1 \text{ AND} (\text{NOT CM})) \quad (5)$$

Table 2
Resultant grammar rules with fitness value and number of production rules.

L-id	Fitness	Grammar $<V, \sum, P, S>$	NPR
L1	1011	$<\{S,C,M\}, \{0, 1\}, \{S \to CCM, M \to ?, M \to 1SM, C \to ?, C \to 0\}, S>$	5
L2	1014	$<\{S\}, \{0, 1\}, \{S \to 1, S \to 0S\}, S>$	2
L3	1013	$<\{S\}, \{0, 1\}, \{S \to ?, S \to 11S1, S \to 00S\}, S>$	3
L4	1011	$<\{S, M\}, \{0, 1\}, \{S \to 1M, S \to 0SM, M \to SSM, M \to ?, M \to 0M\}, S>$	5
L5	1013	$<\{S, C\}, \{0, 1\}, \{S \to C, S \to 00S, C \to 01\}, S>$	3
L6	1012	$<\{S, C\}, \{0, 1\}, \{S \to C, S \to 1S, S \to 0S, C \to 0\}, S>$	4
L7	1012	$<\{S, M\}, \{0, 1\}, \{S \to 1M, S \to 00SM, M \to ?\}, M \to 0M$	4
L8	1014	$<\{S\}, \{(,)\}, \{S \to ?, S \to (S)S\}, S>$	2
L9	1014	$<\{S\}, \{0, 1\}, \{S \to ?, S \to 0S1\}, S>$	2
L10	1012	$<\{S, A\}, \{0, 1\}, \{S \to A11, S \to 1, S \to 011, A \to 0S\}, S>$	4
L11	1013	$<\{S\}, \{a, b\}, \{S \to bSb, S \to aSa, S \to ?\}, S>$	3
L12	1014	$<\{S\}, \{0, 1\}, \{S \to ?, S \to 10S\}, S>$	2
L13	1012	$<\{S, M\}, \{0, 1\}, \{S \to 1M, S \to 0SM, M \to SM, M \to ?\}, S>$	4

NPR: number of production rules.

Eq. (6) expresses the mutation operation and has been derived from the Eq. (4), under the situation that a single MM vector of both child strings is set.

$$OS_j = OS_i \text{XOR MM} \quad (6)$$

The step-by-step mechanism of generating a new offspring is

Table 3
Comparative analysis of GA with and without MDL.

L-id	GAWOMDL				GAWMDL				ITBL				EMPGA			
	Th	GR	μ	σ	Th	GR	μ	σ	Th	GR	μ	σ	Th	GR	μ	σ
L1	30	21 ± 10	22.6	5.7	27	15 ± 11	15.4	4.5	28	18 ± 8	20.7	4.3	31	24 ± 9	24.8	6.2
L2	16	9 ± 7	8.3	3.85	12	6 ± 4	5.3	4.3	19	10 ± 7	6.2	3.4	18	13 ± 5	11.6	4.89
L3	21	26 ± 16	26.3	8.95	17	24 ± 15	23.2	6.78	18	28 ± 15	27.5	8.24	25	30 ± 12	30.4	9.5
L4	33	21 ± 11	18.7	6.3	30	19 ± 10	16.6	5.8	29	19 ± 12	16.4	5.8	37	26 ± 14	21.8	7.41
L5	44	12 ± 9	10.45	5.46	39	9 ± 7	8.53	4.8	47	13 ± 11	10.9	5.62	51	15 ± 8	11.9	12.02
L6	18	14 ± 9	14.9	4.8	13	12 ± 7	12.83	3.4	13	12 ± 9	12.5	3.9	23	18 ± 8	17.5	5.86
L7	19	18 ± 13	21.3	8.91	16	15 ± 8	18.8	6.24	16	19 ± 8	22.8	7.3	26	21 ± 7	20.2	10.61
L8	16	8 ± 7	8.2	3.64	9	6 ± 4	6.7	3.2	18	7 ± 5	6.6	3.2	19	13 ± 10	9.7	5.9
L9	15	7 ± 4	3.6	1.24	11	5 ± 3	3.46	1.03	14	8 ± 5	5.6	2.3	21	10 ± 6	5.3	3.54
L10	22	33 ± 24	21.63	14.83	17	30 ± 22	19.8	12.6	26	37 ± 25	20.2	15.9	27	38 ± 26	27.4	16.2
L11	16	30 ± 19	32.4	10.08	12	29 ± 15	29.23	8.6	19	27 ± 21	30.3	27.8	22	42 ± 21	35.4	18.3
L12	10	7 ± 4	4.8	1.235	8	5 ± 3	3.8	1.12	7	9 ± 5	3.2	2.7	16	11 ± 8	4.8	3.5
L13	24	14 ± 8	12.3	5.3	12	12 ± 6	10.9	4.6	21	13 ± 9	11.2	6.7	31	18 ± 9	13.5	7.6

Th: Threshold, GR: Generation range, μ: Mean, σ: Standard deviation.

depicted at Substep-3 (Fig. 3), whilst Fig. 5 demonstrates the process of offspring creation using an example.

The interesting thing to note at this stage is as the CM and MM vectors have been considered as an argument to the function (f1 and f2), a new individual has no strict correlation with the specific type of the crossover scheme or parent pairs as happens in case of an explicit binary formulation. In some cases, if the evolutionary process is needed for some couples for an identical crossover such as bit-by-bit crossover with a constant seed, then only that operation is performed and fill the mask properly, then apply Eq. (5) multiple times, changing the selected parent pairs only.

An individual population is updated with its fitness value (substep-4) and then merge the populations. This process has been repeated until the termination condition (maximum number of generations or threshold (threshold indicates the highest rank solution's fitness)) is reached. This stopping criterion is common for each language input. Finally, we display the best production rules and the processing time.

4.2. The MDL principle in the GI: an example

An example of L1 = (10)* demonstrates the applicability of the MDL principle in maintaining the regularity of the data (Fig. 6).

1) The first ellipse indicates the sample space of the positive and negative training data for L1 = (10)*.
2) Initially, we get very complex CFG rules with a low fitness value which can be refined by applying the reproduction operator in each generation, where the MDL principle helps in compressing the grammar rules and to generate positive and negative string set required during the execution.
3) After a few generations, simple grammar, but non-constraint CFG rules have been received.
4) When the proposed GAWMDL search reaches the threshold/termination condition, it produces grammar's rule and maximum fitness value. Such grammars are assumed as a good grammar with best fitness value.
5) In the fourth ellipse six CFG rules are provided: first CFG rules have NPR = 3, fitness value = 1013. In second, third and fifth CFG, NPR = 4, fitness value = 1012 but the noticeable thing is the rules generated are different from the same language. At fourth CFG, NPR = 5, fitness value = 1011. In case of sixth CFG, NPR = 2, fitness value = 1014, indicates that the MDL principle has compressed the data more in the case of sixth CFG rules with a maximum fitness value and therefore the system has learned more.

In the present scenario, for selecting the corpus, strings of terminals are generated for the length L for the given language. Initially, $L = 0$ is chosen, which gradually increases up to the required length to represent the language features. Here, a corpus of 25 positive and 25 negative strings are found to be sufficient to represent the selected languages L1–L13 for the CFG induction.

5. Simulation model

The computational experiments have been conducted on a set of RLs and CFLs using L1 through L13 as listed in Table 1. The Java programming Net Beans IDE 7.0.1, Intel Core™ 2 processor (2.8 GHz) with 2 GB RAM have been used.

5.1. Parameter tuning

An extensive control parameter tuning is performed. The orthogonal array with Taguchi SNR [66–69] is utilized for the tuning process. The Taguchi SNR is a log function of the desired output serves as an objective function for the optimization helps in data analysis and prediction of an optimum result. Eq. (7) has been used to evaluate the SNR.

$$SNR_i = -10 \log\left(\sum_{u=1}^{N_u} \frac{y_u^2}{N_i}\right) \tag{7}$$

where i = experiment number, u = trial number, N_i = number of trials for the experiment, and y_u = number generations taken in each trial to reach to the solution.

The GA's performance largely depends on PS, CS, CR and MR. During the tuning process four control factors with three levels PS = [120, 180, 360], CS = [120, 240, 280], CR = [0.3, 0.7, 0.9], and MR = [0.2, 0.5, 0.8] have been used, where following setting gave the best results PS: CS: CR: MR = [120: 120: 0.9: 0.8]. The maximum number of generations = 500 is taken for the experimentations.

5.2. Performance comparison

The authors have compared the performance of the proposed GAWMDL with the GAWOMDL, ITBL and EMPGA. The ITBL and EMPGA have been considered for the comparison purpose as both algorithms were applied to the CFG induction. The EMPGA was proposed to alleviate premature convergence [18]. As the authors have made the claim that the proposed GAWMDL is capable of handling the premature convergence (as the mask-fill reproduction operators and the BBP introduces diversity in the offspring's)

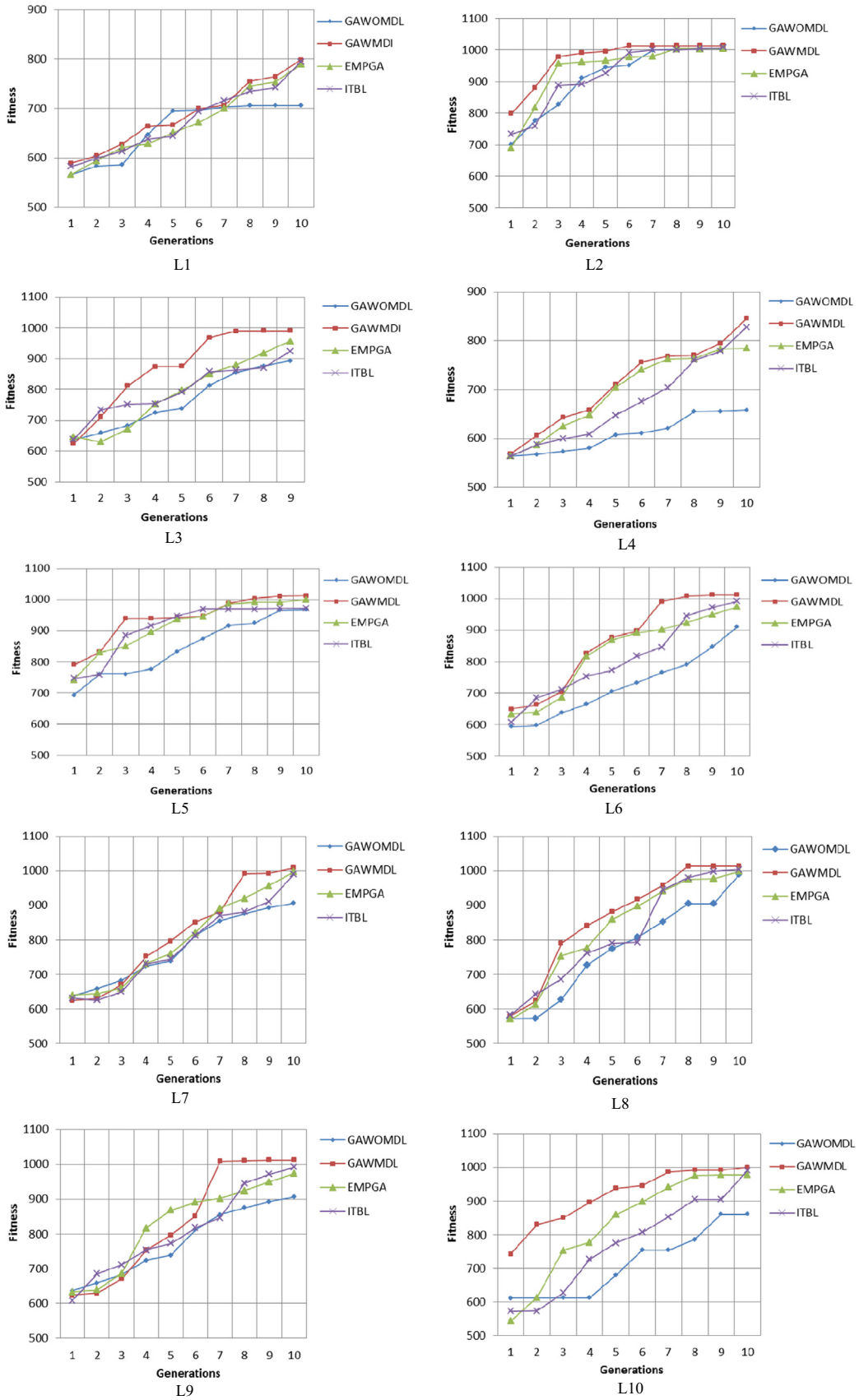

Fig. 7. Fitness vs. generation charts w.r.t. proposed approaches for each algorithm implemented.

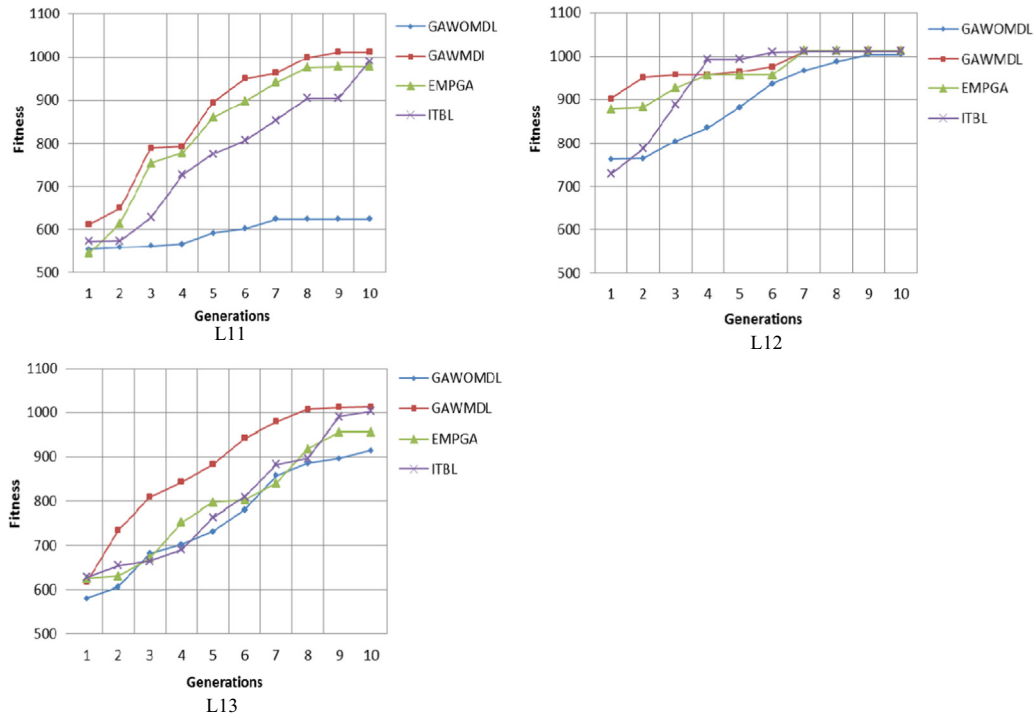

Fig. 7. (*continued*)

Table 4
Paired sample statistics for Pair-1, Pair-2 and Pair-3.

Algorithm's pair		Mean	*N*	Std. deviation	Std. error mean
Pair 1	GAWOMDL	825.4000	15	133.89718	34.57210
	GAWMDL	926.2800	15	124.15734	32.05729
Pair 2	EMPGA	860.1867	15	139.40202	35.99345
	GAWMDL	926.2800	15	124.15734	32.05729
Pair 3	ITBL	866.6200	15	150.62443	38.89106
	GAWMDL	926.2800	15	124.15734	32.05729

leads to compare the performance of the proposed GAWMDL against an algorithm (in our case EMPGA) that introduces diversity in the offspring. The same computational environment has been set up for each algorithm.

5.3. Results and discussion

The experimental results show that the GAWMDL is effective in CFG induction. The MDL principle is able to identify the correct sample string from the corpus with a minimum DL (Fig. 6). The GA is a stochastic search technique; therefore results are collected at an average of ten runs. The resultant grammar rule is validated against the best known available grammar rules are represented via the standard representation $<V, \sum, P, S>$. Table 2 represents the grammar rules generated, fitness value and NPRs.

In order to evaluate the performance of the proposed GAWMDL, a comparative analysis has been conducted as depicted in Table 3. The show that the performance has vastly improved in the case of the GAWMDL. Table 3 shows generation range, threshold value, mean and standard deviation for each language L1–L13. As discussed, the results are collected at an average of the first successful ten runs. The number of generations over ten runs varies, therefore generation range is given. The phenomenon involved with generation range can be understood with the help of an example: the generation range for L1 in case of "GAWO MDL" is 21 ± 10 indicates that generations taken over ten runs varies between 11 $(21-10)$ and 31 $(21+10)$, similarly for others. The mean

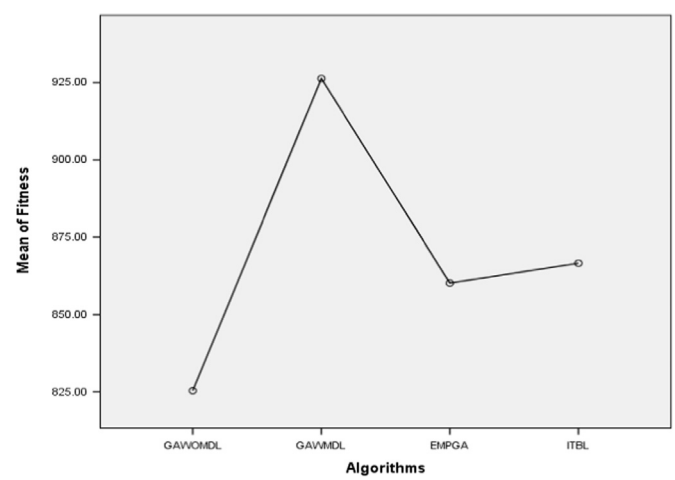

Fig. 8. Profile plot for estimated marginal means of fitness for each approach.

and standard deviation for the GAWMDL concludes that the convergence rate is faster than other algorithms.

Also, the convergence rate of the ITBL and EMPGA is considerably good, whilst the convergence rate of the GAWOMDL is worst.

The comparison chart for the best average fitness value with respect to the generations are shown in Fig. 7 for the first ten iterations for each algorithm. We conclude that the proposed GAWMDL has outperformed the other approaches. The performance of the EMPGA is almost identical to the GAWMDL, whereas the performance of the GAWOMDL is worst.

5.4. Statistical tests

A statistical test has been conducted to evaluate the significance of the proposed GAWMDL with the GAWOMDL, ITBL and EMPGA. The paired *t*-test is conducted on the collected sample considering the hypothesis: "there is no significant difference in the mean of samples at the 5% level of confidence" i.e.

Table 5
Paired sample *t*-test.

Algorithm's pair		Paired differences					*t*	*df*	Sig. (2-tailed)
		Mean	Std. deviation	Std. error mean	95% Confidence interval of the difference				
					Lower	Upper			
Pair 1	GAWOMDL - GAWMDL	−100.88000	41.02952	10.59378	−123.60139	−78.15861	−9.523	14	.000
Pair 2	EMPGA - GAWMDL	−66.09333	50.57572	13.05859	−94.10123	−38.08543	−5.061	14	.000
Pair 3	ITBL - GAWMDL	−59.66000	60.91191	15.72739	−93.39189	−25.92811	−3.793	14	.002

H_0: $\mu_1 = \mu_2 = \mu_3 = \mu_4$

H_A: $\mu_1 \neq \mu_2 \neq \mu_3 \neq \mu_4$

A paired *t*-test is applied to compare the two sample means. Three pairs: pair-1 (GAWOMDL-GAWMDL), pair-2 (EMPGA-GAWMDL) and pair-3 (ITBL-GAWMDL) have been formed to conduct the paired *t*-test. Table 4 represents the paired sample statistics for Pair-1, 2 and 3 respectively. Total 15 ($N=15$) samples have been drawn from each algorithm. The average fitness value for the proposed GAWMDL is 926.2800 higher than the others 825.4000, 860.1867 and 866.6200 have been received respectively for the GAWOMDL, EMPGA and ITBL. The main result of the paired *t*-test is presented in Table 5.

The mean difference for Pair-1 is −100.88000 (825.4000–926.2800), similarly for the other pairs. The p-value represented by "Sig. (2-tailed)" is 0.000, 0.000 and 0.002 for the Pair-1, 2, and 3 respectively. Since the obtained *p*-value is less than 0.05 for each pair, so we could reject the null hypothesis and conclude that the performance of the proposed GAWMDL is statistically significantly different than the other algorithms (GAWOMDL, EMPGA and ITBL). Fig. 8 shows the mean fitness value for each algorithm. The *X*-axis and *Y*-axis are represented respectively the algorithms and estimated marginal mean fitness value. From Fig. 8, it can also be seen that the proposed GAWMDL has shown the highest average fitness value as compared to the other algorithms.

6. Conclusions

In this paper, we have developed a GAWMDL for the CFG induction using BMODS to perform the crossover and mutation operations creating CM and MM. BBP has been used to create an offspring in the next generation. The proposed GA uses the MDL principle to generate a corpus of positive and negative strings up to an appropriate length. A more robust experimental environment has been designed using an orthogonal array and the Taguchi SNR method.

The authors have used 3-levels and four factors during the robust experimental design process. The computational experiments have been performed in various languages of varying complexities (Table 1). The results reported have demonstrated the capability of the proposed algorithm for the GI. Also, it is important to note that the Boolean based operators introduce the diversity in the population in a generative manner that helps the proposed GAWMDL to alleviate the premature convergence. The performance of the proposed GAWMDL has been evaluated against three algorithms: GAWOMDL, EMPGA and ITBL. The EMPGA has been considered in the comparison, mainly because it was proposed to alleviate the premature convergence within the GA and has been applied for the GI. On the other hand, the ITBL focusses on the CFG induction. The comparative results have demonstrated the superiority of the proposed GAWMDL over the other algorithms (GAWOMDL, EMPGA and ITBL). The statistical test (paired *t*-test) has been conducted. The pairs (Pair-1, 2, and 3) have been formed to conduct the tests conclude that the proposed GAWMDL is statistically significantly different than the other methods. One thing more to note at this stage is: the performance of the EMPGA and ITBL is almost similar, whilst the GAWOMDL has shown the worst performance. Overall, a GA based GI system has been proposed using the MDL principles for the generalization and specialization of the training data.

References

[1] John H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, MIT Press, United States, 1992.
[2] David E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learnig, 1989.
[3] P. Wyard, Representational issues for context-free grammar induction using 431 genetic algorithm, in: Proceedings of the 2nd International Colloquium on 432 Grammatical Inference and Applications, Lecture Notes in Artificial Intelli- 433 gence, vol. 862, 1994, pp. 222–235 434.
[4] Mark H. Hansen, Bin Yu, Model selection and the principle of minimum description length, J. Am. Stat. Assoc. 96 (454) (2001) 746–774.
[5] Bill Keller, Rudi Lutz, Evolving stochastic context-free grammars from examples using a minimum description length principle, in: Proceedings of the Workshop on Automata Induction Grammatical Inference and Language Acquisition, 1997.
[6] Yasubumi Sakakibara, Recent advances of grammatical inference, Theor. Comput. Sci. 185 (1) (1997) 15–45.
[7] Nitin Choubey, Hari Mohan Surajkishor, Pandey, M.U. Kharat, Developing Genetic Algorithm Library Using Java for CFG Induction, Int. J. Adv. Technol. 2 (1) (2011) 117–128.
[8] Hari Mohan Pandey, Anurag Dixit, Deepti Mehrotra. Genetic algorithms: concepts, issues and a case study of grammar induction, in: Proceedings of the CUBE International Information Technology Conference ACM, 2012.
[9] R. Sivaraj, T. Ravichandran, A review of selection methods in genetic algorithm, Int. J. Eng. Sci. Technol. 3 (2011) 5.
[10] Luigi Iuspa, Francesco Scaramuzzino, A bit-masking oriented data structure for evolutionary operator's implementation in genetic algorithms, Soft Comput. 5 (1) (2001) 58–68.
[11] Zbigniew Michalewicz, Genetic Algorithms+data Structures=evolution Programs, Springer, Berlin, Germany, 1996.
[12] Jorma Rissanen, Modeling by shortest data description, Automatica 14 (5) (1978) 465–471.
[13] Hlynsson, Höskuldur. Transfer Learning using the Minimum Description Length Principle with a Decision Tree Application, 2007.
[14] Istvan Jonyer, B. Holder Lawrence, J. Cook Diane, MDL-based context-free graph grammar induction and applications, Int. J. Artif. Intell. Tools 13 (01) (2004) 65–79.
[15] Markus Saers, Karteek Addanki, Dekai Wu, "Iterative Rule Segmentation under Minimum Description Length for Unsupervised Transduction Grammar Induction." Statistical Language and Speech Processing, Springer, Berlin Heidelberg 2013, pp. 224–235.
[16] Lee, Kyuhwa, Tae-Kyun Kim, Yiannis Demiris. Learning action symbols for hierarchical grammar induction, in: Proceedings of the IEEE 21st International Conference on Pattern Recognition (ICPR), 2012.
[17] De Jong, Kenneth Alan. Analysis of the Behavior of a Class of Genetic Adaptive Systems, 1975.
[18] Nitin Choubey, Madan Kharat, Approaches for handling premature convergence in cfg induction using GA, Soft Comput. Ind. Appl. (2011) 55–66.
[19] E. Mark Gold, Language identification in the limit, Inf. Control. 10 (5) (1967) 447–474.
[20] Thanaruk Theeramunkongy, Manabu Okumuray, Grammar acquisition and statistical parsing by exploiting Local Contextual Information, J. Nat. Lang. Process. 2 (3) (1995).
[21] Javed, Faizan, et al. Context-free grammar induction using genetic programming, in: Proceedings of the 42nd Annual Southeast Regional Conference.

ACM, 2004.
[22] N.S. Choubey, M.U. Kharat, Sequential structuring element for CFG induction using genetic algorithm, Int. J. Futur. Comput. Appl. 1 (2010).
[23] Hari Mohan Pandey, Context free grammar induction library using Genetic Algorithms, in: Proceedings of the IEEE International Conference on Computer and Communication Technology (ICCCT), 2010.
[24] Huijsen, Willem-Olaf. Genetic grammatical inference. CLIN IV: Papers from the Fourth CLIN Meeting, 1993.
[25] Tomita, Masaru. Dynamic construction of finite-state automata from examples using hill-climbing, in: Proceedings of the Fourth Annual Cognitive Science Conference, 1982.
[26] Pierre Dupont, Regular grammatical inference from positive and negative samples by genetic search: the GIG method, Gramm. Inference Appl. (1994) 236–245.
[27] Bunke Horst, Alberto Sanfeliu (Eds.), Syntactic and Structural Pattern Recognition: Theory and Applications, 7, World Scientific, Singapore, 1990.
[28] Andrew Stevenson, R. Cordy James, Grammatical inference in software engineering: An overview of the state of the art, Softw. Lang. Eng. (2013) 204–223.
[29] Andrew Stevenson, R. Cordy James, A survey of grammatical inference in software engineering, Sci. Comput. Program. (2014).
[30] Hari Pandey, Ankit Mohan, Choudhary, Deepti Mehrotra, A comparative review of approaches to prevent premature convergence in GA, Appl. Soft Comput. (2014).
[31] Geoffrey K. Pullum, Learnability, hyperlearning, and the poverty of the stimulus, in: Proceedings of the Annual Meeting of the Berkeley Linguistics Society vol. 22(1), 2012.
[32] Dana Angluin, H. Smith Carl, Inductive inference: theory and methods, ACM Comput. Surv. (CSUR) 15 (3) (1983) 237–269.
[33] King Sun Fu, Syntactic Pattern Recognition and Applications, Prentice-Hall, Englewood Cliffs, 1982.
[34] Michael A. Harrison, Introduction to Formal Language Theory, Addison-Wesley Longman Publishing Co., Inc., 1978.
[35] Kevin J. Lang, Random DFA's can be approximately learned from sparse uniform examples, in: Proceedings of the Fifth Annual Workshop on Computational Learning Theory. ACM, 1992.
[36] Arlindo L. Oliveira (Ed.), Grammatical Inference: Algorithms and Applications: 5th International Colloquium, ICGI 2000, Springer, Lisbon, Portugal, 2000.
[37] Alexander Clark François Coste, Laurent Miclet. Grammatical Inference: Algorithms and Applications, 2008.
[38] Y.Sakakibara, et al. Grammatical Inference: Algorithms and Applications, in: Proceedings of 2006.
[39] Axel Cleeremans, David Servan-Schreiber, James L. McClelland, Finite state automata and simple recurrent networks, Neural Comput. 1 (3) (1989) 372–381.
[40] Alex Graves, et al., A novel connectionist system for unconstrained handwriting recognition, Pattern Anal. Mach. Intell. IEEE Trans. 31 (5) (2009) 855–868.
[41] Jeffrey L. Elman, Finding structure in time, Cognit. Sci. 14 (2) (1990) 179–211.
[42] Miguel Delgado, M.C. Pegalajar, A multiobjective genetic algorithm for obtaining the optimal size of a recurrent neural network for grammatical inference, Pattern Recognit. 38 (9) (2005) 1444–1456.
[43] Arianna D'Ulizia, Fernando Ferri, Patrizia Grifoni, A survey of grammatical inference methods for natural language learning, Artif. Intell. Rev. 36 (1) (2011) 1–27.
[44] Dana Angluin, Inductive inference of formal languages from positive data, Inf. Control. 45 (2) (1980) 117–135.
[45] Dana Angluin, Queries and concept learning, Mach. Learn. 2 (4) (1988) 319–342.
[46] Leslie G. Valiant, A theory of the learnable, Commun. ACM 27 (11) (1984) 1134–1142.
[47] Ming Li, M.B. Vitányi Paul, Learning simple concepts under simple distributions, SIAM J. Comput. 20 (5) (1991) 911–935.
[48] Colin De La Higuera, A bibliographical study of grammatical inference, Pattern Recognit. 38 (9) (2005) 1332–1348.
[49] Colin de la Higuera, Grammatical Inference: Learning Automata and Grammars, Cambridge University Press, New York, NY, USA, 2010.
[50] Petasis, et al., e-GRIDS: computationally efficient grammatical inference from positive examples, Grammars 7 (69–110) (2004) 2004.
[51] Y. Sakakibara, M. Kondo, GA-based learning of context-free grammars using tabular representations, ICML 99 (1999) 354–360.
[52] M. Jaworski, O. Unold, Improved TBL algorithm for learning context-free grammar, in: Proceedings of the International Multiconference on ISSN, vol. 1896, 2007, p. 7094.
[53] N. Bhalse, V. Gupta, Learning CFG using Improved TBL algorithm, Comput. Sci. Eng. 2 (1) (2012) 25.
[54] Peter Grünwald, A minimum description length approach to grammar inference. Connectionist, statistical and symbolic approaches to learning for natural language processing, Springer, Berlin Heidelberg 1996, pp. 203–216.
[55] Heni Ben Amor, Achim Rettinger. Intelligent exploration for genetic algorithms: using self-organizing maps in evolutionary computation, in: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation. ACM, 2005.
[56] Colin Higuera, Ten open problems in grammatical inference, Gramm. Inference: Algorithms Appl. (2006) 32–44.
[57] Ryo Yoshinaka, Identification in the limit of k, l-substitutable context-free languages, Gramm. Inference: Algorithms Appl. (2008) 266–279.
[58] Alexander Clark, R.émi Eyraud, Amaury Habrard, A polynomial algorithm for the inference of context free languages, Gramm. Inference: Algorithms Appl. (2008) 29–42.
[59] Alexander Clark, Distributional learning of some context-free languages with a minimally adequate teacher, Gramm. Inference: Theor. Results Appl. (2010) 24–37.
[60] Matej Črepinšek, Marjan Mernik, Viljem Žumer, Extracting grammar from programs: brute force approach, ACM Sigplan Not. 40 (4) (2005) 29–38.
[61] Dejan Hrncic, Marjan Mernik. Memetic grammatical inference approach for DSL embedding, in: Proceedings of the IEEE 34th International Convention, MIPRO, 2011.
[62] Dejan Hrncic, Marjan Mernik, Barrett R. Bryant. Improving Grammar Inference by a Memetic Algorithm, in: Proceedings of the IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 42.5, 2012, pp.692–703.
[63] Dejan Hrnčič, et al., A memetic grammar inference algorithm for language learning, Appl. Soft Comput. 12 (3) (2012) 1006–1020.
[64] Ray J. Solomonoff, A formal theory of inductive inference Part I, Inf. Control. 7 (1) (1964) 1–22.
[65] Robert G. Gallager, Information Theory and Reliable Communication, Wiley, New York, 1968.
[66] Tapan P. Bagchi, Kalyanmoy Deb, Calibration of GA parameters: the design of experiments approach, Comput. Sci. Inform. 26 (1996) 46–56.
[67] W.H.P. Yang, Y.S. Tarng., Design optimization of cutting parameters for turning operations based on the Taguchi method, J. Mater. Process. Technol. 84 (1) (1998) 122–129.
[68] Unal, Resit, Edwin B. Dean, Taguchi Approach To Design Optimization For Quality And Cost: An Overview, 1990.
[69] Ranjit K. Roy, Design of Experiments Using the Taguchi approach: 16 Steps to Product and Process Improvement, John Wiley & Sons, United States, 2001.
[70] Hari Mohan Pandey, et al., Evaluation of Genetic Algorithm's Selection Methods." Information Systems Design and Intelligent Applications, Springer, India 2016, pp. 731–738.
[71] Anupriya Shukla, Hari Mohan Pandey, Deepti Mehrotra. Comparative review of selection techniques in genetic algorithm, in: Proceedings of the IEEE 2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), 2015.
[72] Hari Mohan Pandey, Performance evaluation of selection methods of genetic algorithm and network security concerns, Procedia Comput. Sci. 78 (2016) 13–18.
[73] Matej Črepinšek, Shih-Hsi Liu, Marjan Mernik, Exploration and exploitation in evolutionary algorithms: a survey, ACM Comput. Surv. (CSUR) 45 (3) (2013) 35.
[74] Dejan Hrnčič, Marjan Mernik, Barrett R. Bryant, Embedding DSLS into GPLS: A Grammatical Inference Approach*, Inf. Technol. Control. 40 (4) (2011) 307–315.

Hari Mohan Pandey is major in Computer Science and Engineering and pursuing Ph.D. in Formal Language Theory, Grammatical Inference and Evolutionary Algorithms. He has served in industry and in many academic institutions. Previously, He was associated with the Middle East College, Coventry University, U.K. Presently, he is working in the department of computer science and engineering at Amity University Uttar Pradesh, India. He has published research papers in various International conferences and journals. He has received the global award for the best computer science faculty of the year 2015. He is the author of several books of Computer Science & Engineering for McGraw-Hill, Pearson Education, University Science Press, and Scholar Press. He is associated with various International Journals as a reviewer and editorial board member. He has served as a leading guest editor for several International journals. He has organized special sessions at International conferences, served as chair and delivered keynotes.

Ankit Chaudhary is Assistant Professor at Department of Computer Science, Truman State University, MO, USA. He received his Ph.D. in Computer Engineering and his areas of research interest are computer vision, artificial intelligence and graph algorithms. He has authored 75 research papers and an Associate Editor of Computer & Electrical Engg. Journal, Elsevier. He is member of IEEE and also serves in the editorial board at many international journals.

Deepti Mehrotra did Ph.D. from Lucknow University and currently she is working as Professor in Amity school of Engineering and Technology, Amity University, Noida, earlier, she worked as Director of Amity School of Computer Science, Noida, India. She has more than 20 years of research, teaching and content writing experience. She had published more than 60 papers in international refereed Journals and conference Proceedings. She is editor and reviewer of many books, referred journal and conferences. She is regularly invited as resource persons for FDPs and invited talks at national and international conference. She guided Ph.D. and M.Tech students.

Graham Kendall received the B.S. in computation (first class, honors) from the Institute of Science and Technology, University of Manchester, Manchester, U.K., in 1997 and the Ph.D. degree in computer science from the University of Nottingham, Nottingham, U.K., in 2001. His previous experience includes almost 20 years in the information technology industry where he held both technical and managerial positions. He is a Professor of Computer Science at the University of Nottingham and is currently based at their Malaysia Campus where he holds the position of Vice-Provost (Research and Knowledge Transfer). He is a Director of two companies (EventMAP Ltd., Nottingham, U.K.; Aptia Solutions Ltd., Nottingham, U.K.) and CEO of two companies (MyRIAD Solutions Sdn Bhd, Malaysia and MyResearch Sdn Bhd, Malaysia). He is a Fellow of the Operational Research Society. He is an Associate Editor of nine international journals, including two IEEE journals: the IEEE Transactions on Evolutionary Computation and the IEEE Transactions on Computational Intelligence and AI In Games. He chaired the Multidisciplinary International Conference on Scheduling: Theory and Applications in 2003, 2005, 2007, 2009, and 2011, and has chaired several other international conferences, which has included establishing the IEEE Symposium on Computational Intelligence and Games. He has been awarded externally funded grants worth over 6 million from a variety of sources, including the Engineering and Physical Sciences Research Council (EPSRC) and commercial organizations.