Proceedings of the 15th International Conference on Machine Learning, Madison (WI), USA - July 24-27, 1998

An Experimental Evaluation of Coevolutive Concept Learning

Cosimo Anglano **Attilio Giordana** **Giuseppe Lo Bello** **Lorenza Saitta**
Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, 10149 Torino, Italy
{mino,attilio,lobello,saitta}@di.unito.it

Abstract

In this paper an extensive experimental evaluation of an evolutionary approach to concept learning is presented. The experimentation, performed with the system G-NET, investigates the effectiveness of the approach along the following dimensions: Robustness with respect to parameter setting, effectiveness of the MDL criterion coupled with a stochastic search bias, impact of coevolution on the quality of the solution and on the computational effort required, and ability to face problems requiring structured representation languages. A discussion of the obtained results and a suggestion on when this type of approach might be useful is also provided.

1 INTRODUCTION

Supervised concept learning has been tackled, so far, with several approaches, including symbolic, connectionist and evolutive ones. Different approaches are better suited to different classes of problems, depending, for instance, on the nature of data or the availability of domain-specific knowledge.

In the hope of making a little step ahead in the direction of matching learning algorithms to problems, in this paper we present an experimental exploration of an evolutionary approach to the task of learning concept descriptions. Our exploration is articulated along three dimensions: The capability of dealing with complex representation languages, such as subsets of predicate logics; the exploitation of distributed architectures, allowing coevolution to be efficiently implemented; the interaction between the stochastic search bias and the Minimum Description Length (MDL) principle (Rissanen, 1978), used as evaluation criterion of the concept description.

The experimentation has been conducted with a new version of G-NET (Version 2.0), a descendant of the system REGAL (Giordana and Neri, 1996). G-NET's architecture relies on a computational model characterized by the absence of global memory, which extends the diffusion model (Manderik and Spiessens, 1989) previously developed for genetic algorithms. With respect to a previous implementation (Anglano et al., 1997), the version described here includes an explicit coevolutionary strategy based on (Potter et al., 1995), a new objective function based on the MDL principle, and an improved set of genetic operators.

A first point emerged from the experimentation, using both G-NET and REGAL, is that evolutionary search techniques can indeed be fruitfully exploited in concept acquisition. On standard benchmarks they showed performances at least comparable with the best ones presented in the literature (Neri and Saitta, 1996).

A second point is that good performance does not come for free: Using a simple genetic algorithm, easy to understand and quick to implement, may not be a solution. The evolutionary inference engine has to be integrated into a possibly complex architecture, allowing sophisticated description languages, flexible heuristic learning strategies, and distributed computation to be accommodated.

A third point is that evolutionary search proved to be quite robust, because it did not require any parameter tuning over a range of different problems. Finally, stochastic search bias proved to be well suited to different evaluation criteria (Anglano et al., 1997), including the MDL (Rissanen, 1978). G-NET is based, as REGAL was, on the theory of niches and species formation, which already proved to be effective in learn-

ing disjunctive concept definitions (Giordana and Neri, 1996). As niches and species formation is a way of addressing multimodal search problems, disjunctive concept induction naturally fits in this framework. However, methods based on species formation may require large populations when weak species must survive in the presence of much stronger ones (Giordana and Neri, 1996).

In order to cope with this problem, G-NET 2.0 uses a new learning method, which combines the Universal Suffrage selection scheme (Giordana and Neri, 1996) with an explicit coevolutionary strategy, similar to the one proposed in (Potter et al., 1995). Finally, G-NET 2.0 uses a new set of genetic operators, which explicitly aims at preserving the diversity in the population, reducing thus premature convergence and increasing the effectiveness of the genetic search.

2 EVOLUTIONARY ALGORITHM

As its ancestor REGAL, G-NET learns concept descriptions in a language similar to VL_{21} (Michalski, 1983). More specifically, a concept is described by a set $\Phi = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ of Horn clauses, in which the construct of internal disjunction is also allowed. In Logic Programming, an internal disjunction is a special term describing a set of constants. By setting a limit on the maximum complexity, Horn clauses can be encoded as fixed length bitstrings. A detailed description of the language used by G-NET can be found in (Giordana and Neri, 1996; Giordana et al., 1997).

G-NET's inductive engine exploits a stochastic algorithm organized in two levels. The lower level, named Genetic layer (G-layer), searches for Horn clauses representing partial definitions φ of the concept to learn. The architecture of the G-layer derives from the diffusion model (Manderik and Spiessens, 1989), and integrates different ideas originated inside the field of evolutionary computation and tabu search (Rayward-Smith et al., 1989). The upper level, namely the Supervisor, builds up a global disjunctive definition Φ, out of the partial definitions φ_i's generated inside the G-layer, using a greedy set covering algorithm. Moreover, the Supervisor interacts with the G-layer according to a coevolutionary strategy (Potter et al., 1995), which aims at increasing the probability of evolving clauses useful to improve the quality of the disjunctive concept description currently in progress.

From a computational point of view, the G-layer consists of a set of elementary searching nodes called G-nodes. Every G-node, G_i, is associated with a single concept instance e^+ and executes a local evolutionary search aimed at constructing an inductive hypothesis covering e^+, and having a fitness value as high as possible. The same instance e^+ can be assigned to more than one G-node.

The association between G-nodes and concept instances is dynamically established by the Supervisor, which decides what regions of the hypothesis space to search, and how much.

Every G-node is provided with a small local memory, where it stores the set of current hypotheses it is working on (local population). Basically, the search algorithm executed by a G-node resembles a simple Genetic Algorithm:

G-node Search Algorithm
repeat

1. Select two clauses φ_1 and φ_2 from the local memory with probabilities proportional to their fitness;
2. Create two new clauses φ_1' and φ_2', both different from φ_1 and φ_2;
3. Evaluate φ_1' and φ_2' on the learning set;
4. Broadcast the new clauses to every G-node associated with some instance e^+ they correctly cover;

until a *halt condition* is reached

The outcome of the evaluation step is a fitness value $f_L(\varphi)$ corresponding to the quality of the clause φ (see below). By generalizing a formula covering the associated instance e^+, a G-node can implicitly generate formulas also covering other instances which are assigned to different G-nodes. The aim of the broadcasting step is to propagate these formulas to the G-nodes, which potentially can benefit from them. When a clause is broadcast to another G-node, it competes for entering the local memory by playing a kind of stochastic tournament (Harik, 1995), based on the fitness value f_L. As the policy we adopt enforces diversity in the local memories, a clause is allowed to play the tournament only if no copy of it is already there. At the beginning, the population of a G-node is initialized with only one individual and can grow up to a maximum predetermined size. The tournament step is performed only after the maximum size has been reached.

This way of propagating inductive hypotheses among G-nodes promotes the formation of families of hypotheses, which cluster the concept instances into

groups, roughly corresponding to different modalities of the target concept. From the point of view of evolutionary computation this process can be seen as a process of niches and species formation (Goldberg and Richardson, 1987).

The emergence of species (i.e. concept modalities) is the baseline for the coevolutionary strategy adopted by the Supervisor. Periodically, the Supervisor (a) collects the best representatives of each species, and works out a global concept description, (b) reassigns the concept instances to the G-nodes in order to increase the search efforts where emerging species still correspond to low quality inductive hypotheses, and (c) supplies a corrective term to be added to the fitness of the inductive hypotheses in the G-layer, helping the species that better contribute to the global solution to develop further.

3 THE FITNESS EVALUATION

In G-NET, two different fitness functions, f_G and f_L, are used in order to evaluate global (disjunctive) and local (conjunctive) concept descriptions, respectively. Both measures are based on the Minimum Description Length Principle (Rissanen, 1978). The function $f_G(\Phi)$ is the sum of three terms:

$$f_G(\Phi) = MDL_{MAX} - MDL(\epsilon^+(\Phi) + \epsilon^-(\Phi)) - MDL(\Phi) \quad (1)$$

being MDL_{MAX} the MDL of the whole learning set, $MDL(\epsilon^+(\Phi) + \epsilon^-(\Phi))$ the MDL of the set ϵ^+ of positive concept instances not covered by Φ and of the set of negative instances ϵ^- covered by Φ, and $MDL(\Phi)$ the minimum description length of the syntactic form of Φ. In turn, $MDL(\Phi)$ is computed as the sum $MDL(\Phi) = \sum_i MDL(\varphi_i)$ of the MDL of the single clauses belonging to Φ. In all cases, the expressions for the MDL of the different terms have been obtained using Stirling's approximation, as in (Oliveira and Sangiovanni-Vincentelli, 1996). The definition of f_G has been chosen in order to have a function which increases when the MDL decreases, because it is easier to transform it into a probability, used to guide the stochastic search.

The local fitness f_L for evaluating a single clause φ in a G-node takes the form:

$$\begin{aligned} f_L(\varphi) \quad = \quad & MDL_{MAX} - MDL(\varphi) + \\ & -MDL(\epsilon^-(\varphi)) + (f_G(\Phi') - f_G(\Phi)) \end{aligned} \quad (2)$$

being Φ the current global description constructed by the Supervisor, and Φ' the formula obtained by adding φ to Φ and eliminating all redundant disjuncts but φ. In other words, the second and third term evaluate how simple and consistent φ is. The fourth term is the bias for enforcing the coevolutionary strategy and evaluates how well φ combines with the other existing partial descriptions in order to form a global solution, covering the instance e^+ associated to the G-node and as much as possible of the other instances.

4 THE COEVOLUTION STRATEGY

The Supervisor enforces coevolution by means of two algorithms, which are executed periodically at the end of a *macro-cycle*. A macro-cycle is measured by counting the number of iterations of the Search Algorithm (μ-cycles) globally performed, in the G-layer, by the G-nodes. The first algorithm computes a global concept description Φ out of the best representatives of the species emerged in the G-layer, and is based on a hill climbing optimization strategy. At first, from every G-node the locally best hypothesis is collected and is then merged into a redundant disjunctive description Φ'. Then, Φ' is optimized by eliminating the disjuncts, which are not necessary. This is done by repeating the following cycle until Φ' reaches a final form Φ, which cannot be optimized further:

1. Search the clause φ such that $f_G(\Phi' - \varphi)$ shows the greatest improvement.
2. Set $\Phi' = \Phi' - \varphi$

The second algorithm computes the assignment of the (positive) concept instances to the G-nodes. The basic strategy consists in focusing the search on the concept instances which are covered by poor inductive hypotheses, without omitting to continue the refinement of the other hypotheses. This is done by balancing the computation among the different emerging species, in such a way that species covering smaller niches will get the same computational power as the ones covering larger niches.

The Supervisor keeps track of the solution state of every positive instance $e^+ \in E^+$ (the set of all positive instances), i.e., the best solution found for it. Moreover, it also records the number c_i of μ-cycles, related to ${e_i}^+$, occurred during the past computation. The kernel of the coevolutionary control strategy is the method used for accounting the μ-cycles related to every ${e_i}^+$. As soon as clauses covering many examples will begin to develop, we will find spontaneously born

clusters of G-nodes that elect the same clause as current best hypothesis in their population. This can be interpreted as a form of implicit cooperation, which leads to the generation of a clause, representative of the work of all of them. Therefore, the Supervisor attributes to an instance ${e_i}^+$ all the μ-cycles executed by the G-nodes whose local memory contains a copy of the best clause attributed to ${e_i}^+$.

At the end of a macro-cycle, the concept instances are reassigned to G-nodes with the goal of balancing the work spent for every ${e_i}^+$, on the basis of the number c_i of μ-cycles. Let C the maximum value for c_i $(1 \leq i \leq |E^+|)$. The Supervisor computes for every ${e_i}^+$ the amount $g_i = C - c_i$ of μ-cycles necessary to balance the computational cost for it. Afterwards, the instances are stochastically assigned to G-nodes with probability proportional to g_i. When a G-node G is assigned to a new instance e^+, it is restarted. If the global description Φ contains a clause φ_e, covering e^+, φ_e is inserted in the population of G. Otherwise, it will be initialized by means of the seeding operator described below.

5 THE GENETIC OPERATORS

In the same way as REGAL, G-NET represents Horn clauses as fixed length bitstrings ((Giordana et al., 1997)); then, search operators can be implemented as in standard Genetic Algorithms (Goldberg, 1989). As a matter of fact, G-NET uses three basic operators: seeding, crossover and mutation. The seeding operator (Giordana and Neri, 1996) is used for initializing the local memory in the G-nodes when it is empty. When called in a G-node G_i, it stochastically generates a clause φ_i, which is guaranteed to cover the instance e^+ currently associated with G_i.

Crossover and *mutation* operators can be applied in different modalities, depending upon the clauses they are applied to, and are guaranteed to produce new hypotheses different from the parents (original clauses).

The crossover is a combination of the *two point crossover* with a variant of the *uniform crossover* (Syswerda, 1989), modified in order to perform either generalization or specialization of the hypotheses. More specifically, the crossover operator can be activated in three different modalities: exchanging, specializing and generalizing, which are stochastically selected depending on the consistency and completeness of the selected clauses. Given a pair of clauses φ_1, φ_2, the modality to use is stochastically decided in two steps. In the first step it is decided whether to apply the exchanging modality, with probability p_{ec} (by default $p_{ec} = 0.1$), or to proceed to the second step, with probability $1 - p_{ec}$. Afterwards, if the second step is entered, the system decides whether to apply generalization or specialization to each one of the parent clauses. Let φ_i be one of the parents; the probability $p_{gc}(\varphi_i)$, of using generalization, and $p_{sc}(\varphi_i) = 1 - p_{gc}(\varphi_i)$, of using specialization, are computed according to the rule:

$$p_{gc}(\varphi_i) = (\epsilon^-(\varphi_i)/(m^+(\varphi_i) + \epsilon^-(\varphi_i))) \qquad (3)$$

being m^+ the number of positive instance correctly classified by φ_i and ϵ^- the number of negative instances, as previously defined. Afterwards, if the same modality has been chosen for both operands, the crossover will be applied with this modality. Otherwise, if the modalities are discordant, the exchanging modality will be used.

In this way, the generalizing modality tends to be used when the parents are both consistent, the specializing modality when the parents are both inconsistent, and the exchanging modality when one is consistent and the other is inconsistent. The first decision step guarantees that an assigned percentage of pure information exchange takes place in any case.

In order to guarantee the actual exchange of information, the crossover algorithm first constructs an index $I = \{i_1, i_2, \ldots, i_n\}$ of pointers to the positions in the bitstring where the corresponding bits in the two parents have different values. Afterwards, if generalization/specialization has been chosen, two temporary clauses ψ_1 and ψ_2, identical to φ_1 and φ_2 respectively, are created.

Then, for every element $i_j \in I$ the following procedure is repeated:

- **if** generalizing modality has been chosen **then** with probability p_u replace in ψ_1 and ψ_2 the value of the bit $b(i_j)$ with the logical *or* of the corresponding bits in the operands φ_1 and φ_2.
- **if** specializing modality has been chosen **then** with probability p_u replace in ψ_1 and ψ_2 the value of the bit $b(i_j)$ with the logical *and* of the corresponding bits in φ_1 and φ_2.

If, after applying this stochastic procedure, no bit has been changed, one bit chosen at random in I is generalized/specialized.

When the exchanging modality is chosen, the classical two-point crossover is applied, with the difference that, in order to guarantee an information exchange, the two crossover points are chosen on the index vector I instead of on the whole bitstring.

The *mutation* operator adopts a strategy similar to the one described so far for crossover, and tries to generalize or to specialize an individual, depending on its consistency or inconsistency. Also the mutation operator can have three modalities, namely seeding, generalizing or specializing, which are selected with probability p_{seed} (by default $p_{seed} = 0.1$), p_{gm} and p_{sm}, respectively. The probabilities p_{gm} for generalizing mutation and p_{sm} for specializing mutation are computed with the rule:

$$\begin{aligned} p_{sm} &= (1 - p_{seed})(\epsilon^- / (\epsilon^- + m^+)), \\ p_{gm} &= 1 - p_{seed} - p_{sm}, \end{aligned} \tag{4}$$

where the argument of ϵ^- and m^+ has been omitted for brevity. If the specializing mutation is chosen, the mutation is applied as follows: let n_1 be the number of bits set to "1" in the bitstring; then, the mutation operator turns to "0" a fraction γ of them, which is obtained by randomly selecting a real number in the interval $[0, n_1/10]$. The bits to be set to "1" are selected in an analogous way, when the generalizing mutation is chosen.

It is easy to recognize that generalizing and specializing mutations are nothing else than the dropping condition and adding condition operators defined in (Jong et al., 1993).

In the cycle executed by each G-node, two clauses are selected at each iteration with probability proportional to their fitness f_L. If the population is empty, a new individual will be created using the seeding operator. Otherwise, if the two selected clauses φ_1 and φ_2 are different, crossover is applied. On the contrary, if the same clause is selected two times, two new clauses are created using mutation.

The nice aspect of this strategy is that it automatically adapts to the composition of the population. When the population in a node is dominated by a clause that has a fitness much higher than the others (and, then, it is frequently selected for reproduction with itself), the search turns into a stochastic hill climbing.

6 EXPERIMENTAL EVALUATION

In the following we present an extensive evaluation of G-NET made on a variety of datasets, selected with the aim of testing the system with respect to the dimensions mentioned in Section 1: language bias, robustness to evaluation criteria, and overall performance. The parameters to tune are actually very few (the genetic operators constants are not user tunable) and correspond to the local population size P_s, the macro-cycle size M_c and the number of G-nodes N_g. In all the previous experimentation done, they did not appeared to be critical at all and the following setting has been chosen as a default: $P_s = 10, M_c = 300, N_g = 100$. The results reported in the following have been obtained using the default setting.

Table 1 reports a first group of results on datasets used to test the system Smog (Oliveira and Sangiovanni-Vincentelli, 1996), which exploits the MDL as hypothesis evaluation criterion. Results by C4.5 are used as a baseline. The performance for Smog and C4.5 are those reported in (Oliveira and Sangiovanni-Vincentelli, 1996); Smog used many other datasets, but only some of them are available at the U.C. Irvine repository (Merz et al., 1991).

G-NET has always been run with a set of 100 G-nodes and has been stopped after creating 40000 hypotheses. The specific goal of the test was twofold: to investigate how G-NET is affected by changing its evaluation criterion, all the rest being the same, and whether the MDL could still be effective when coupled with a stochastic search bias, such as the one provided by G-NET, very different from the ones used in Smog and in C4.5. The answer has been positive in both cases.

By considering the results on the *Monk-2* dataset, the effectiveness of G-NET's species formation mechanism is evident: the system always found 26 disjuncts, sometimes the correct ones and sometimes little different; this explains the small error of the acquired knowledge base. The species formation stability is also confirmed by the fact that in all cases G-NET found the same number of disjuncts, differing for small variations.

Table 2 reports results of an experiment aimed at verifying the utility of increasing the computational power of the search when approaching a more large and difficult problem. The dataset used is the *Splice Junctions* dataset (Towell and Shavlik, 1994). The task is that of identifying boundaries between coding (exons) and non-coding (introns) regions of genes occurring in eukaryote DNA.

The Splice Junctions dataset has been previously used to test the system REGAL, which presented the best results so far among the many reported in the literature (Neri and Saitta, 1996). While increasing the

Table 1: Comparison Between G-NET, Smog And C4.5 With Respect To The Average Error Rate Of The Solution, Evaluated With The 10-fold Crossvalidation

Problem	**Dataset size**	**Average Error %**			**Average N. of Disjuncts**
		G-NET	Smog	C4.5	G-NET
monk1	432 10-fold	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	3.0
monk2	432 10-fold	2.80 ± 3.80	0.00 ± 0.00	32.83 ± 10.66	26.0
monk3	432 10-fold	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	3.0
tictactoe	958 10-fold	0.97 ± 0.62	2.82 ± 1.97	7.07 ± 1.82	10.5
credit	690 10-fold	15.8 ± 4.40	19.57 ± 5.08	14.03 ± 3.28	14.0
breast	699 10-fold	5.29 ± 2.89	6.72 ± 2.44	5.85 ± 3.32	2.6
vote	435 10-fold	5.10 ± 3.20	5.29 ± 2.64	4.63 ± 3.05	2.0

Table 2: Comparison Between G-NET And REGAL With Respect To The Average Error Rate And The Complexity Of The Solution

Problem	**Dataset size**	**Average Error %**		**N. of Disjuncts**	
		G-NET	REGAL	G-NET	REGAL
splice-j. (EI)	$2000 + 1190$	3.40	4.40	7	19
splice-j. (IE)	$2000 + 1190$	2.90	4.20	10	26
splice-j. (Neither)	$2000 + 1190$	3.30	5.20	11	21
mushrooms	$4000 + 4124$	0.00	0.00	3	6

system parallelism and decreasing the complexity, G-NET achieved even lower average error rate (error rate is an average over 3 runs). The second best results were achieved by KBANN (Towell and Shavlik, 1994): 7.56% for EI, 8.47% for IE, and 4.62% for Neither. This comparison suggests that genetic search could be better suited to complex problems.

Finally, Table 3 reports the results of experiments aimed at confirming that G-NET (as its predecessor REGAL) is able to effectively deal with more complex languages, such as predicate logic based ones.

The first row in Table 3 refers to the *mutagenesis* dataset, a challenging problem widely used in the ILP community for testing induction algorithms in First Order Logic (King et al., 1995). The problem consists in learning rules for discriminating substances having cancerogenetic properties on the basis of their chemical structure. The difficulty lies mainly in the complexity of matching formulas in First Order Logic, which limits the exploration capabilities of any induction system. To our knowledge, the best results with this database have been obtained by STILL (Sebag, 1997) a stochastic induction algorithm that easily reaches error rates below 10%, and, with a careful setting of the control parameters, made the best hit at 6.4%. Many other systems, going from Linear Regression to PROGOL and FOIL, reported error rates ranging between 11% and 14%. G-NET, using only the predicates used in (Sebag, 1997), obtained an error rate of 8.8%.

The second case study is a classification problem (Esposito et al., 1992) of documents acquired through a scanner, and processed by an image processing program that produces a structured description of the layout. The dataset contains structured data described with 5 symbolic and 3 numeric attributes, and has been used to test learners with the capability of dealing with numerical features in FOL (Esposito et al., 1992; Botta and Giordana, 1993). G-NET does not have, at the moment, any specific strategy for dealing with numerical features, and so we transformed the problem into a symbolic one by discretizing the numeric features. Each numeric feature has been discretized by subdividing the range into 16 equal length intervals. G-NET easily reached an error rate below the 1%, approximately the same as SMART+ which has specific strategies for dealing with numerical features.

Finally, the last case study (*Train Checkout-3*) is a

Table 3: Experiments With First Order Problems. Error Rate For The *Train Check-out 3* Is An Average Of 3 Runs

Problem	**Dataset size**	**Average Error %**			**N. of Disjuncts**
		G-NET	STILL	SMART + FONNs	G-NET
mutagenesis	230 10-fold	8.80 ± 7.90	6.4 ± 4.5	n.a.	3
office-doc	$210 + 160$	0.89 ± 0.72	n.a.	0.80	11
train check-out 3	$500 + 6000$	11.3 ± 0.47	n.a.	16.8	2

difficult artificial dataset generated for testing FONNs (Botta et al., 1997), a kind of neural network recently proposed for refining numerical terms in Horn Clauses. The dataset contains the description of a set of trains, similar to the one proposed by Michalski, where each coach is described by means of a set of 5 symbolic and 4 numerical attributes. In (Botta et al., 1997) three different learning problems of increasing difficulty are presented, related to this dataset. The problems consist in learning sets of rules for assessing when a train meets the safety conditions required for travelling on a given line. The one we considered here is the most difficult among them and the challenge is to discover the rule used for classifying the concept instances:

> **a train cannot go if it contains two near cars, both without brakes and heavier than a threshold we_3 or if it contains two near cars carrying an unstable load (special material) and heavier than a threshold $we_4 < we_3$.**

FONNs could easily reach an error rate below 2% on a test set of 6000 instances starting from a handcrafted knowledge base, which correctly described the structure of the rule hidden in the data, but only reached an error rate of about 17% starting from a set of rules learned by SMART+ from 500 learning instances. Reshaping the problem in propositional calculus, C4.5 and CART could not go below an error rate of 27%, and neural networks such as multi-layer perceptron and cascade correlation where performing even more poorly (Botta et al., 1997).

G-NET has been run by discretizing every numeric attribute into a range of 30 intervals. As it appears from the last row in Table 3, it was able to find two clauses which show an error rate around 11%.

7 DISCUSSION

As it appears from the results reported above, G-NET is a very flexible system, able to deal with many different problems, producing good results. Moreover, as already stated, the results have been obtained without performing any specific tuning, so that the system proved to be quite robust and easy to use. This looks surprising considering that a major complain against GAs is the difficulty of tuning parameters.

We point out that, in spite of its architecture strongly resembling a Genetic Algorithm, G-NET cannot be considered a classical GA, because the principles which control the evolution are substantially different. In our opinion, two aspects determine the success of G-NET: the enforcement of diversity in the local populations and the coevolution.

In their basic formulation GAs use genetic pressure, i.e. the capability of the most fit individuals to reproduce more quickly, so that the weakest ones are eliminated from the population. This mechanism has the positive effect of focusing the search on the most fit individuals, so that, in the best case, the algorithm will climb up a maximum of the fitness function. Unfortunately, the mechanism is unstable and a too quick convergence prevents reaching optimal solutions. Another drawback is that, in this way, many identical individuals will be present in the population, so that the search can become ineffective because the major search operator (crossover) reproduces again and again the same individuals.

A trend in the GA literature, which at least partially relieves this problem, is related to the theory of species and niches formation. Species formation can be promoted in many ways by limiting the genetic pressure between species (Goldberg and Richardson, 1987). Species formation offers some benefits, such as the possibility of restricting crossover to the individuals of the same species (crossover among dif-

ferent species is essentially deceptive), increasing the search effectiveness and allowing the discovery of multiple modalities. For instance, in G-NET, as well as in REGAL, this has been exploited for learning disjunctive concept descriptions. However, even in this framework, genetic pressure continues to be used inside a same species as a mechanism of focus of attention. Requiring that a population (the local memory of G-nodes) contains individuals (clauses) all different, is a definite departure from this mechanism, and drastically limits any form of genetic pressure. Therefore, the algorithm becomes much more stable and less sensitive of crossover type, and of crossover and mutation rates. Furthermore, in the place of genetic pressure other strategies, tailored to the specific task, can be used for guiding the search. In our case, the coevolution is now the major strategy that focuses the search where it is necessary instead of letting it follow the stream enforced by the genetic pressure. A second component is represented by the local search operators, which are context sensitive and make the best effort in order to increase the exploration capability of the algorithm.

Both the idea of maintaining the population diversity and the one of coevolution originated before G-NET, whose originality consists in the adaptation to the specific task and to the integration of these ideas into a unique framework. On the one hand, diversity in GAs has been already proposed by several authors (Augier et al., 1995), although no one speculates on the reasons why a GA should benefit from it. On the other hand, diversity could be related to tabu search. The local memory of a G-nodes works as an elementary tabu list which prevents the algorithm from reprocessing already generated instances without an explicit will to do so.

Coevolution appeared inside the GA community several years ago (Husbands and Mill, 1991), and has been considered by few others in the following. The coevolution model, described here, conforms to the one proposed by (Potter et al., 1995), properly re-interpreted in the framework of concept learning, which naturally conforms to it.

Finally, the reassignment of the examples to be covered to G-nodes, performed by the Supervisor, can be considered a kind of boosting (Shapire, 1990): in subsequent runs, the search efforts shall be concentrated on those parts of the hypothesis space not yet adequately covered. Currently, the series of found hypotheses are combined into a unique formula, which differentiate this approach from a genuine boosting. However, nothing hinders the Supervisor from keeping apart the hypotheses and using them according to a majority voting classification strategy, instead of combining them. This possibility has not been explored yet.

8 CONCLUSIONS

In this paper we presented a new induction system based on an evolutionary approach, which is the outcome of several years of investigation in this direction.

Given the good results obtained across a variety of datasets, languages, and evaluation criteria, it should be evident that a system like G-NET can be profitably used to explore the structure of new learning problems, when little a priori information, clearly pointing to another approach, is available.

Moreover, thanks to its computational model, G-NET is able to effectively exploit parallel computing systems, allowing to deal with large and complex datasets. As a matter of fact, in addition to the possibility of distributing the search among many G-nodes, G-NET offers also the possibility of distributing the hypotheses evaluation on several processors. Although this aspect has not been described here, because it is outside the scope of the paper, the current implementation of G-NET runs on a cluster of workstations (Anglano et al., 1997). This facility has been extensively exploited for the experiments on *Mutagenesis* and *Splice Junctions* datasets, so that the results for every run have been obtained in a few hours.

The conclusion is that G-NET seems to be very well-suited to learning structured concepts, such as the ones typically learned by ILP methods, and, in addition, to face learning problems on large databases.

References

Anglano, C., Giordana, A., Lo Bello, G., and Saitta, L. (1997). A network genetic algorithm for concept learning. In *Int. Conf. on Genetic Algorithms*, pages 434–441, Lansing, MI. Morgan Kaufmann.

Augier, S., Venturini, G., and Kodratoff, Y. (1995). Learning first order rules with a genetic algorithm. In *Proc. of the First International Conference On Knowledge Discovery and Data mining*, pages 21–26, Montreal, Quebec, CA. AAAI Press.

Botta, M. and Giordana, A. (1993). SMART+: A multi-strategy learning tool. In *IJCAI-93,*

Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, volume 2, Chambéry, France.

Botta, M., Giordana, A., and Piola, R. (1997). Fonn: Combining first order logic with connectionist learning. In *14-th International Conference on Machine learning ICML-97*, pages 157–166, Nashville, TN. Morgan Kauffman.

Esposito, F., Malerba, D., and Semeraro, G. (1992). Classification in noisy environments using a distance measure between structural symbolic descriptions. *IEEE trans. on Pattern Analisys and Machine Intelligence*, 14(3):390–402.

Giordana, A. and Neri, F. (1996). Search-intensive concept induction. *Evolutionary Computation*, 3:375–416.

Giordana, A., Neri, F., Saitta, L., and Botta, M. (1997). Integrating multiple learning strategies in first order logics. *Machine Learning*, pages 209–240.

Goldberg, D. (1989). *Genetic Algorithms*. Addison-Wesley.

Goldberg, D. and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In *Int. Conf. on Genetic Algorithms*, pages 41–49, Cambridge, MA. Morgan Kaufmann.

Harik, G. (1995). Finding multimodal solutions using restricted tournament selection. In *Int. Conf. on Genetic Algorithms*, pages 24–31, Pittsburgh, PA. Morgan Kaufmann.

Husbands, P. and Mill, F. (1991). Co-evolving parasites improve simulated evolution as an optimization procedure. In *4th Int. Conf. on Genetic Algorithms*, pages 264–270, Fairfax, VA. Morgan Kaufmann.

Jong, K. D., Spears, W., and Gordon, F. (1993). Using genetic algorithms for concept learning. *Machine Learning Journal, 13*, pages 161–188.

King, R., Srinivasan, A., and Stenberg, M. (1995). Relating chemical activity to structure: an examination of ilp successes. *New Generation Computing*, 13.

Manderik, B. and Spiessens, P. (1989). Fine-grained parallel genetic algorithms. In *Int. Conf. on Genetic Algorithms*, pages 428–433, Fairfax, VA. Morgan Kaufmann.

Merz, C., Murphy, P., and Aha, D. (1991). *Repository of Machine Learning Databases*. University of California, Irvine, CA.

Michalski, R. (1983). A theory and methodology of inductive learning. In Michalski, R., Carbonell, J., and Mitchell, T., editors, *Machine Learning: An Artificial Intelligence Approach*, pages 83–134, Los Altos, CA. Morgan Kaufmann.

Neri, F. and Saitta, L. (1996). Genetic algorithms for pattern recognition. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, PAMI-18:1135–1142.

Oliveira, A. and Sangiovanni-Vincentelli, A. (1996). Using the minimum description length principle to infer reduced ordered decision graphs. *Machine Learning*, pages 23–50.

Potter, M., DeJong, K., and Grefenstette, J. (1995). A coevolutionary approach to learning sequential decision rules. In *Int. Conf. on Genetic Algorithms*, pages 366–372, Pittsburgh, PA. Morgan Kaufmann.

Rayward-Smith, V., Osman, I., Reeves, C., and Smith, G. (1989). *Modern Heuristic Search Methods*. Wiley, New York, NY.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14:465–471.

Sebag, M. (1997). Tractable induction and classification in first order logic. In *15th International Joint Conference on Artificial Intelligence*, Tokyo, Japan.

Shapire, R. (1990). The strength of weak learnability. *Machine Learning*, 5:197–227.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. In *3th Int. Conf. on Genetic Algorithms*, pages 2–9, Fairfax, VA. Morgan Kaufmann.

Towell, G. and Shavlik, J. (1994). Knowledge based artificial neural networks. *Artficial Intelligence*, 70(4):119–166.