Coevolutionary, Distributed Search for Inducing Concept Descriptions

C. Anglano, A. Giordana, G. Lo Bello, L. Saitta
Dipartimento di Informatica, Università di Torino,
C.so Svizzera 185, 10149 Torino, Italy
e-mail:{mino,attilio,lobello,saitta}@di.unito.it

Abstract. This paper presents a highly parallel genetic algorithm, designed for concept induction in propositional and first order logics. The parallel architecture is an adaptation for set covering problems, of the diffusion model developed for optimization.
The algorithm exhibits other two important methodological novelties related to Evolutionary Computation. First, it combines niches and species formation with coevolution, in order to learn multimodal concepts. This is done by integrating the Universal Suffrage selection operator with the coevolution model recently proposed in the literature. Second, it makes use of a new set of genetic operators, which maintain diversity in the population.
The experimental comparison with previous systems, not using coevolution and based on traditional genetic operators, shows a substantial improvement in the effectiveness of the genetic search.

Keywords: Concept Learning, Parallel Genetic Algorithms, Coevolution.

1 Introduction

In the recent literature, Genetic Algorithms (GAs) emerged as valuable search tools in the field of concept induction [2, 8, 6, 3]. The feature that looks particularly attractive for this task is the exploration power, potentially greater than that of traditional search methods.

This paper describes a GA-based inductive learner, oriented to acquire concepts described in First Order Logic. Its architecture relies on a computational model characterized by the absence of global memory, which extends the diffusion model previously developed for GAs. The underlying distributed architecture allows a natural introduction of coevolution. Coevolution has been defined by [7, 10] and refers to the possibility of guiding evolving populations through a global feed-back.

Our starting point is the theory of niches and species formation, which already proved to be effective in learning disjunctive concept definitions. A disjunctive concept definition consists of a set of conjunctive logical formulas, each one capturing a different modality of the target concept. As niches and species formation is a way of addressing multi-modal search problems, disjunctive concept induction naturally fits in this framework. Several recent algorithms, such

as COGIN [6] and REGAL [3], exploit this idea, even though they adopt different methods for promoting species formation.

As discussed in [3], methods only based on species formation may require very large populations when small species are required to survive in the presence of very large ones. For this reason, REGAL adopts a long term control strategy resembling to coevolution, in order to reduce the pressure among the species. Here, we propose a new method which combines the Universal Suffrage selection operator with an explicit coevolutionary strategy similar to the one proposed by [10].

Moreover, this paper presents another substantial novelty, with respect to REGAL and other GAs designed for concept induction tasks, which consists of a new set of genetic operators, which explicitly aim at preserving the diversity in the population. Preserving diversity reduces premature convergence and increases the effectiveness of the genetic search.

As it will be shown in Section 7, the new algorithm, while preserving (or even increasing) the accuracy of REGAL, shows a substantial reduction in the complexity of the genetic search.

2 Learning Concepts with Genetic Algorithms

The task of learning concept definitions from examples can be stated as follows: given a learning set $E = E^+ \cup E^-$, consisting of positive and negative examples of a target concept ω, and a logical language L, the task consists in finding a logical formula $\Phi \in L$, which is true of all the positive examples E^+ and false of all the negative ones E^-. If such a Φ is found, the definition $\Phi \rightarrow \omega$ holds. Depending on the case, the logical language L can be a propositional or a First Order one. Independently of the order of L, the general structure of a concept definition Φ is a disjunction $\Phi = \phi_1 \vee \phi_2 \vee \ldots \vee \phi_n$ of conjunctive definitions $\phi_1, \phi_2, \ldots, \phi_n$, each one representing a different modality of the concept ω. In the following we will assume that L is a VL_{21} language like to the one used by Induce [9] and by REGAL [3].

As previously mentioned, an appealing method to exploit GAs in Machine Learning consists in combining species and niches formation [5] with coevolution [10]. Several examples can be found in [10] for learning behavioral strategies. In the following we will consider a two-level architecture, whose lower level is a distributed GA, which searches for conjunctive descriptions by promoting the formation of species in the populations. The upper level applies a coevolutive strategy that performs two tasks: on one hand, it continuously updates a disjunctive description, combining together individuals chosen from the different species evolved at the first level; on the other hand, it interacts with the lower level with the aim of favoring the evolution of those species that better go together in the current disjunctive description. The system REGAL [3] is a first example of how such an architecture can be implemented.

However, the way niching and coevolution are integrated in REGAL and in other systems like Samuel [10] is not very suitable to exploit large network com-

puters. In fact, these systems are still based on the network (or island) model described by Goldberg [4], where niches tend to be identified with single computational nodes. Therefore, the available parallelism is limited by the number of emerging species. In order to overcome this limitation, we designed a different computational model, where the notion of global mating pool has been abandoned. As described in Figure 1, the architecture of the resulting system, G-NET, encompasses three kind of nodes: 1) Genetic nodes (*G-nodes*), where individuals mate and reproduce, 2) Evaluation nodes (*E-nodes*), where individuals are evaluated, and 3) a Supervisor node, which coordinates the computation of the G-nodes according to the coevolutive strategy.

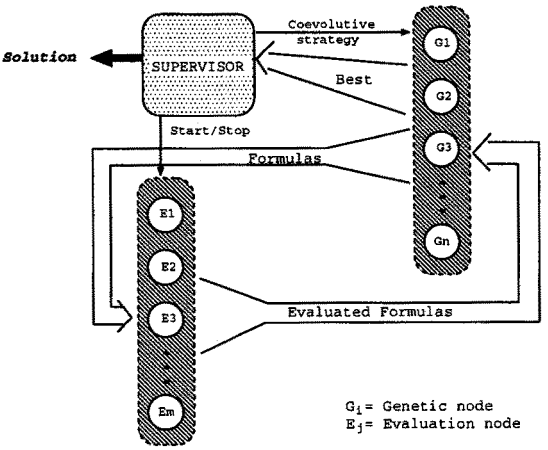

Fig. 1. Parallel Architecture

In G-NET, positive concept instances are considered elementary niches, and each G-node computes an elitist GA, which evolves a micro-population settled on a specific niche. In other words, each G-node searches for the "best" formula covering the positive learning instances associated with it. The same elementary niche can be assigned to many G-nodes at the same time, so that the number of actually active G-nodes depends only upon the available resources and not upon the problem. The Supervisor decides when and how many active copies of a niche must exist.

Upon receiving an individual, an E-node evaluates it on the learning set, computes a fitness value and sends it back (together with the computed information) to the set of G-nodes (broadcast communication), activating thus new genetic cycles. When a G-node finds a solution which is better than the ones it already has, according to a local fitness function f_L, it sends it to the Supervisor as a candidate for assembling the global disjunctive description. Periodically, the Supervisor resumes and actuates the coevolutive strategy: first, it assembles a global disjunctive solution, out of the locally best solutions it received from the

G-nodes, trying to optimize a global fitness function f_G. Afterwards, it gives a feed-back to the G-nodes, in order to guide the genetic search towards solutions that better integrate each other in a global disjunctive description. This coevolutive strategy consists of two components, which act independently. The first component controls the amount of search that has to be done on different niches, and determines the number of G-nodes assigned to each niche. Those niches, which developed solutions "weak" in a global context, are given more processors in order to increase the chance of improving their local solutions. The second component supplies a corrective term to the local fitness function, which allows the genetic search to be explicitly guided towards local solutions which contribute to increase the quality of the global solution. To this aim, the Supervisor broadcasts the current disjunctive description to G-nodes.

The separation of the genetic cycle from the evaluation process simply aims at increasing the explicit parallelism available.

3 The Fitness

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. The function f_G is a combination of three different terms, correspondings to three different features of a concept description, namely: completeness (v), consistency (w) and syntactic simplicity (z), which are three standard criteria used in machine learning since [9].

In order to introduce the analytic form of f_G we need to introduce some basic definitions. The symbol M^+ shall denote the cardinality of E^+, i.e the number of positive training instances, and M^- shall denote the cardinality of E^-, i.e the number of negative training instances. Let φ be a inductive (disjunctive or conjunctive) hypothesis; $m^+(\varphi)$ and $m^-(\varphi)$ shall denote the number of positive and negative instances covered by φ, respectively. For the sake of simplicity we will write m^+, m^-, and so on, being the argument φ evident from the context.

The completeness is evaluated as $v = m^+/M^+$, whereas the consistency is evaluated as an exponential function $w = e^{-m^-}$ of the covered negative examples. The syntactic simplicity is evaluated as the ratio $z = m^+/(N_a + m^+)$, being N_a the number of conditions occurring in a formula. In practice z tries to capture the information compression represented by the syntactic form with respect to its extension on the learning set. As long as N_a decreases, z increases approaching to 1.

The analytic form for the global fitness f_G is then defined by the expression:

$$f_G(\Phi) = (1 + Av(\Phi) + Bz(\Phi))w(\Phi)^C \qquad (1)$$

where A, B and C are user tunable constants, which allow the different components to be weighted.

The local fitness f_L for an hypothesis ϕ adds a corrective term to expression (1), in order to account for how much ϕ contributes to improve (worsen) the current global concept description. Let Φ be the disjunctive concept description

currently elaborated by the Supervisor and broadcast to the G-nodes. Let, moreover, ϕ be a conjunctive hypothesis in a G-node. The fitness $f_L(\phi)$ is evaluated as:

$$f_L(\phi) = (1 + Av(\phi) + Bz(\phi))w(\phi)^C + (f_G(\Phi') - f_G(\Phi)) \quad (2)$$

being Φ' the formula obtained by adding ϕ to Φ and eliminating all redundant disjuncts but ϕ.

4 The Genetic Nodes

Conjunctive solutions (individuals) are represented as bitstrings of fixed length as in REGAL [3]. In the bitstring every bit represents a logical condition; if the bit is "1", the condition occurs in the formula, if it is "0", it does not. This correspondence can always be established, provided that a limit is imposed on the maximum complexity of a conjunctive formula.

Every G-node executes a genetic algorithm aimed at finding the formula that better covers the concept instance it has been assigned to. As described in Figure 2, the basic architecture is quite simple, in order to have a light weight computational object, which can easily be dynamically allocated on a network computer; the architecture comprises the program encoding the genetic cycle, a small memory containing the local population, an input port receiving the individuals sent by the E-nodes, and an output port sending the new individuals to the E-nodes. As the local population is kept small (from 5 to 40 individuals) a non-standard replacement policy (similar to that described in [1]) is used, in order to enforce diversity: all the individuals in the same node are required to be different, so that a richer genetic information is present in the population.

The individuals arriving at the input gate of a G-node are all those evaluated by the E-nodes; then, they may or may not cover the specific instance e assigned to it. Even though the goal of the G-node is to find formulas covering e, individuals not covering e can nevertheless carry information useful to find better generalizations for the individuals which actually cover e. Therefore, individuals both covering and not covering e are allowed to enter the population, but a policy is adopted which does not allow the individuals not covering e take over the other ones.

A G-node repeats the following cyclic procedure, until stopped by the supervisor:

Cellular Genetic Algorithm

1. Select two individuals from the local population with probability proportional to their fitness
2. Generate two new individuals φ_1 and φ_2 by applying the genetic operators
3. Broadcast copies of φ_1 and φ_2
4. **While** the network is ready **do**
   (a) Receive some individuals ψ from the network according to rule (3) (see below)

(b) Replace ψ in the local population by playing a tournament step for each one of them

5. Go to step 1

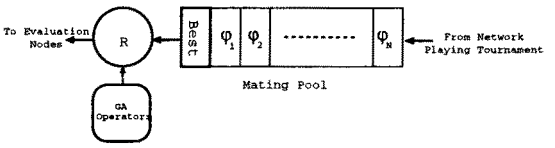

Fig. 2. G-node architecture.

Proportionate selection is used, whereas replacement is made using the tournament policy, adapted to the local need, as it will be explained in the following. First of all, every individual arriving from the network is subject to a probabilistic filter which can accept or refuse it.

$$\begin{aligned} &\textbf{if } \varphi \textit{ belongs to the local population } \textbf{then } \textit{reject } \varphi \\ &\quad \textbf{else if } \varphi \textit{ matches the learning event } e \\ &\qquad \textbf{then } \textit{accept } \varphi \textit{ with probability } \mathbf{p_{cv}} \\ &\qquad \textbf{else } \textit{accept } \varphi \textit{ with probability } \mathbf{p_{uc}} \end{aligned} \tag{3}$$

In (3), $p_{cv} \in [0, 1]$ is a user defined parameter, whereas, p_{uc} is computed as $p_{cv}(1-\nu)^6$ being ν the proportion of individuals non covering e in the population. In this way the pressure of the individuals not covering e is automatically limited. Each individual that goes through the filter competes for entering the population by playing a tournament: an opponent is randomly selected and the victory is assigned with probability proportional to the respective fitnesses.

An elitist strategy is used so that in each G-node the currently best solution cannot be replaced by a worse individual. This is obtained by avoiding to choose the best solution as opponent in a tournament. Periodically, all G-nodes send a copy of the best found individuals to the supervisor, which reacts reassigning the examples to the G-nodes.

5 The Genetic Operators

Being conjunctive formulas encoded into fixed length bitstrings, the reproduction operators are straightforward to implement, and borrow many ideas from the Genetic Algorithm theory [4]. In particular, the reproduction operators are: a task specific crossover, a task specific mutation, and the seeding operator [3], used for initializing the population when this last is empty. All operators are implemented in such a way that they always produce offsprings different from the parents.

The *crossover* is a combination of the *two point crossover* with a variant of the *uniform crossover* [11], 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 chromosomes. Given a pair of chromosomes φ_1, φ_2, the modality to use is decided in two steps. At the first step it is decided whether to apply the exchanging modality, with probability p_{ec}, or to proceed through the second step, with probability $1 - p_{ec}$, being p_{ec} a user definable parameter. Afterwards, if the second step is entered, the system independently decides whether to apply generalization or specialization to each one of the parents. 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:

$$p_{gc}(\varphi_i) = (m^-(\varphi_i)/(m^+(\varphi) + m^-(\varphi)))^\alpha \tag{4}$$

being α a user defined parameter. Afterwards, if the same modality has been chosen for both parents, 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 inconsistents, 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 construct 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 value. Afterwards, if the generalization/specialization has been chosen, two temporary offsprings 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 parents.
- **if** specializing modality has been chosen **then** with probability p_u (a-priori assigned) replace in ψ_1 and ψ_2 the value of the bit $b(i_j)$ with the logical *and* of the corresponding bits in the parents.

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 chromosome.

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 may have three

modalities, namely seeding, generalizing and specializing, which are selected with probability p_{seed}, p_{gm} and p_{sm}, respectively. The seeding probability p_{seed} is given a priori, whereas the probabilities p_{gm} for generalizing mutation and p_{sm} for specializing mutation are computed with the rule:

$$p_{sm} = (1 - p_{seed})(m^- / (m^- + m^+))^\alpha, \quad p_{gm} = 1 - p_{seed} - p_{sm} \tag{5}$$

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 $[0, \beta_m n_1]$, being $\beta_m \in [0, 1]$ a user defined parameter. The bits to be set at "1" are selected in an analogous way, when the generalizing mutation is chosen.

It is easy to recognize that specializing and generalizing mutations are nothing else than the dropping condition and adding condition operators defined in [2].

In the genetic loop executed by each G-node, two individuals 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 individuals φ_1 and φ_2 are genetically different, crossover is applied. On the contrary, if the same individual is selected two times, two new offsprings will be 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 an individual that has a fitness much higher than the others (and, then, it is frequently selected for reproduction with itself), the genetic search turns into a stochastic hill climbing.

6 The Coevolutive Strategy

The medium-term control strategy actuated by the Supervisor node is based on a coevolutive approach. It basically goes through a cycle, which has a period measured in terms of the number of iterations performed by the G-nodes, in which it receives the best solution found by each one of the G-nodes, elaborates the current disjunctive concept description Φ, and then gives a feedback to the G-nodes which enforces a coevolutionary model.

At the moment, the algorithm used for working out a disjunctive description is very simple and is based on a hill climbing optimization strategy. At first, all conjunctive solutions collected from the G-nodes are included 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 disjunct ϕ such that $f_G(\Phi' - \phi)$ shows the greatest improvement.
2. Set $\Phi' = \Phi' - \phi$

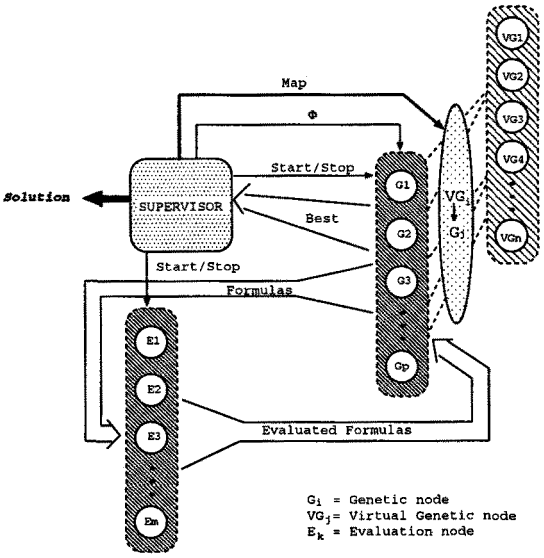

Fig. 3. The medium term coevolutive strategy

The first component of the co-evolutive control strategy on the G-nodes' activity is reminiscent of the techniques used in Operating Systems for multiplexing the Cpu among the processes in execution. According to the model in Section 2, every positive example in the learning set is associated to a local learning task performed by one or more processes executed by the G-nodes. For distinguishing G-nodes from the elementary learning tasks, we will denote the latter as VG-nodes (Virtual Genetic nodes); more specifically, VG_i $(1 \leq i \leq |E_i^+|)$ will denote the learning task associated to the example $e_i \in E^+$.

The Supervisor keeps track of the solution state of every VG-node VG_i, i.e. the best solution found for it, a selection (possibly empty) of the individuals found in the populations evolved by the G-nodes associated to it, and other information, such as the number c_i of computational events, related to task VG_i, occurred during the past computation. The kernel of the co-evolutive control strategy is the method used for accounting the events related to every task. As soon as formulas covering many examples will begin to develop, we will find spontaneously born clusters of G-nodes which elect the same formula as current best individual in their population. This can be interpreted as a form of implicit cooperation which led to the generation of a formula representative of the work of all of them. Therefore, the Supervisor attributes to a VG-node VG_i all the events produced by the G-nodes sharing the best formula attributed to VG_i.

When, at the end of a cycle, the control strategy is actuated, the VG-nodes are reassigned to G-nodes. The criterion for the reassignment is that of balancing the work spent for every task VG_i on the basis of the number c_i of computational events. Let C be the maximum value for c_i $(1 \leq i \leq |E^+|)$; the Supervisor

computes for every VG_i the amount $g_i = C - c_i$ of computational events necessary to balance the computational cost for it. Afterwards, the VG-nodes are stochastically assigned to G-nodes with probability proportional to g_i divided by the fitness $f_{L_i}^{(best)}$ of the currently best formula covering VG_i. Then the G-nodes are initialized with the information saved in the corresponding VG-nodes and restart.

The effect of this strategy is that of focusing the computational resources on the VG-nodes covered by formulas having a low fitness or a low ratio m^+/M^+. In fact, the VG-nodes falling in these cases will have either a low value for $f_i^{(best)}$ or a high value for g_i, because they will accumulate the computational events with few other VG-nodes.

The second component of the co-evolutive strategy reduces to broadcasting the current disjunctive description Φ to all G-nodes, so that they can update the local fitness f_L for the individuals in the local populations.

7 Experimental Evaluation

In order to have a comparison with REGAL, which can be considered the source of inspiration of the current system, we will report a benchmark on the *Mushrooms* dataset and on *Thousand Trains* dataset, used in [3].

A first comparison shows the benefits of using the co-evolutive control strategy illustrated in Section 6 and the new reproduction operators described in Section 5. A second comparison is with REGAL and shows how our system can obtain solutions of at least the same quality as REGAL with a smaller computational cost.

For the Mushrooms dataset the task consists in discriminating between *edible* and *poisonous* mushrooms. The dataset consists of 8124 instances, 1208 of edible mushrooms and 3196 of poisonous ones. Each instance is described by a vector of 22 discrete, multi-valued attributes. By defining a condition for each one of the attribute values, a global set of 126 constraints is obtained, which leads to bitstrings of 126 bits for encoding the individuals. Randomly selected sets of 4000 instances (2000 edible + 2000 poisonous) have been used as learning sets, while the remaining 4124 instances have been used for testing. The system can always found a perfect definition for both classes, covering all the examples and no counterexamples on the test set, as also REGAL and other induction algorithms can do.

The Thousand Trains dataset represents an artificial learning problem in First Order Logic obtained by extending the well known *Trains Going East or Going West* problem proposed by Michalski in order to illustrate Induce' learning capabilities [9]. In this case 500 examples of trains going east and 500 trains going west have been generated using a stochastic generator. A detailed description can be found in [3].

The parameter, which has been used by the previous Genetic Algorithms to evaluate the quality of the found solution, has been the complexity of the final disjunctive solution, measured as the total number of conditions present in

it. In the machine learning literature, simple solutions are considered preferable among a set with the same performances.

In Figure 4 we report the evolution of the complexity for the Mushrooms dataset, in terms of the number of genetic cycles executed by all the G-nodes, globally. The curves labeled as "cv", "ncv", and "rough" correspond respectively to using the complete operator set and the coevolutive strategy, disactivating the co-evolutive strategy, and using random assignment and using REGAL' reproduction operators.

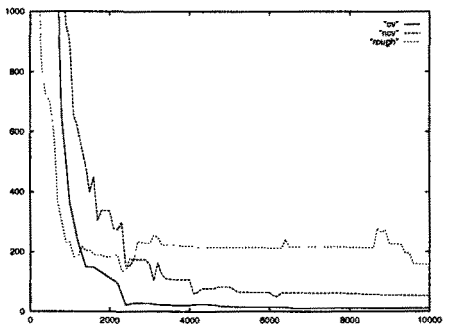

Fig. 4. Mushroom dataset: Complexity of the solution vs. the genetic cycles.

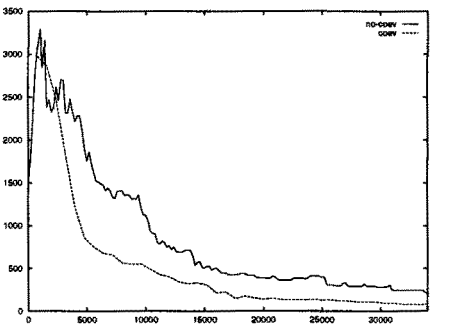

Fig. 5. Thousand Trains dataset: complexity of the solution vs. the genetic cycles.

Here it is possible to appreciate the effect of the co-evolution and of the new reproduction operators, separately. When the co-evolution was disactivated, the G-nodes were assigned to VG-nodes using an equal probability.

An analogous comparison which illustrates the benefit of co-evolution on the Thousand Trains dataset is reported in Figure 5, where the curves labelled as "coev" and "no-coev" correspond to using the co-evolution strategy and disactivating it, respectively. In both experiments the same default setting for $A = B = C = \alpha = 1$ in the fitness function and the same number (400) of G-nodes have been used. It is worth noting that these parameters are not critical, and very similar results have been obtained with quite different settings.

A comparison between REGAL and G-NET on the task of finding a definition for the concept of *poisonous mushroom* is reported in Table 1. We notice that G-NET can reach a complexity smaller than REGAL with a smaller computational cost. We believe that this improvement is substantially due to the use of the mutation operator.

8 Conclusion

A new distributed model for a Genetic Algorithm designed for concept induction from examples has been presented. The algorithm presents several important novelties. A first novelty is represented by the fine-grained distributed mating

Table 1. Comparison between G-Net and REGAL

System	G-Net		REGAL	
	avg	min	avg	min
Complexity	11	11	21.40	13
Cost	12423	11738	31130	23900

pool, which eliminates every notion of common memory in the system so that it is easy to exploit distributed computational resources such as Connection Machines and Network Computers. Other novelties are represented by the coevolutive strategy inspired by the model proposed by [10], and by the new set of genetic operators.

The new algorithm has been evaluated on non trivial induction tasks obtaining good results. Therefore, the new architecture seems to be a promising one for facing computing intensive induction problems emerging in data mining applications.

References

1. AUGIER, S., VENTURINI, G., AND KODRATOFF, Y. Learning First Order Logic Rules with a Genetic Algorithm. In *Proc. of the First International Conference on Knowledge Discovery and Data Mining* (1995).
2. DEJONG, K., SPEARS, W., AND GORDON, F. Using genetic algorithms for concept learning. *Machine Learning Journal, 13* (1993), 161–188.
3. GIORDANA, A., AND NERI, F. Search-intensive concept induction. *Evolutionary Computation Journal* (1996), Winter, 1995.
4. GOLDBERG, D. *Genetic Algorithms.* Addison-Wesley, Reading, MA, 1989.
5. GOLDBERG, D., AND RICHARDSON, J. Genetic algorithms with sharing for multimodal function optimization. In *Int. Conf. on Genetic Algorithms* (Cambridge, MA, 1987), Morgan Kaufmann, pp. 41–49.
6. GREENE, D., AND SMITH, S. Competition-based induction of decision models from examples. *Machine Learning Journal, 13* (1993), 229–258.
7. HUSBANDS, P., AND MILL, F. Co-evolving parasites improve simulated evolution as an optimization procedure. In *4th Int. Conf. on Genetic Algorithms* (Fairfax, VA, 1991), Morgan Kaufmann, pp. 264–270.
8. JANIKOW, C. A knowledge intensive genetic algorithm for supervised learning. *Machine Learning Journal, 13* (1993), 198–228.
9. MICHALSKI, R. A theory and methodology of inductive learning. In *Machine Learning: An AI Approach* (Los Altos, CA, 1983), J. C. R. Michalski and T. Mitchell, Eds., Morgan Kaufmann, pp. 83–134.
10. POTTER, M., DEJONG, K., AND GREFENSTETTE, J. A coevolutionary approach to learning sequential decision rules. In *Int. Conf. on Genetic Algorithms* (Pittsburgh, PA, 1995), Morgan Kaufmann, pp. 366–372.
11. SYSWERDA, G. Uniform crossover in genetic algorithms. In *3th Int. Conf. on Genetic Algorithms* (Fairfax, VA, 1989), Morgan Kaufmann, pp. 2–9.