Knowledge-Based Evolutionary Search for Inductive Concept Learning

Federico Divina and Elena Marchiori

Vrije Universiteit of Amsterdam {divina,elena}@cs.vu.nl

Summary. This chapter provides a short overview of a GA-based system for inductive concept learning (in a fragment of first-order logic). The described system exploits problem–specific knowledge by means of ad-hoc selection, mutation operators and optimization applied to the single individuals. We focus on the experimental analysis of selection operators incorporating problem knowledge.

1 Introduction

Learning from examples in (a fragment of) first-order logic (FOL), also known as Inductive Logic Programming (ILP) [32], or more in general Inductive Concept Learning (ICL) [29], constitutes a central topic in Machine Learning, with relevant applications to problems in complex domains like natural language and molecular computational biology [31].

The problem can be formulated in the following manner: given a description language used to express possible hypotheses, a background knowledge, a set of positive examples, and a set of negative examples, one has to find a hypothesis which covers all positive examples and none of the negative ones (cf. [26, 30]). The so learned concept can be used to classify new examples.

ICL can be viewed as a search problem in the space of all possible hypotheses. This problem is NP-hard even if the language to represent hypotheses is propositional logic. Many search algorithms for ICL based on different approaches have been introduced, in particular EA-based systems. Systems like GIL [23], GLPS [28, 34] and STEPS [25] use an encoding where a chromosome represents a set of rules. In other GA-based systems like SIA01 [4], REGAL [17, 33], G–NET [3] and DOGMA [22, 21], a chromosome represents one rule. In the latter case a non redundant hypothesis is extracted from the final population at the end of the evolutionary process.

In evolutionary computation 'learning' is a byproduct of the evolutionary process as successful individuals are retained through stochastic trial and error. This learning process can be rather slow, due to the weak strategy used to guide evolution. A way to overcome this drawback is to hybridize the evolutionary process by incorporating knowledge-based operators. In this chapter we describe a hybrid EA for ICL based

on this approach, called Evolutionary Concept Learner (ECL) [12]. ECL evolves a population of if-then rules through the repeated application of selection, mutation and optimization of individuals of the population. Knowledge is used in the mutation and selection operators. We focus on the experimental analysis of selection in ECL.

The chapter is organized as follows. The next section summarizes recent GA-based systems for ILP. In section 3 a description of ECL is given. Section 4 describes three selection operators, which are experimentally analyzed in section 5. Section 6 contains some remarks.

2 GA-based ILP learners

EAs, and in particular GAs and GP, have proved to be successful in solving comparatively hard optimization problems, as well as problems like ICL [19, 24]. Moreover GAs and GP have some features that render them an attractive alternative for tackling ILP problems. First, GAs and GP have an intrinsic parallelism and can therefore exploit parallel machines much more easily than classical search algorithms. Secondly, GAs and GP have the capability of escaping from local optima, while greedy algorithms may not show this ability. Finally EAs tend to cope better than greedy rule induction algorithms when there is interaction among arguments and when arguments are of different type (numerical, discrete) [16].

Depending on the representation used, two major EA-based approaches can be individuated: the *Pittsburgh* and the *Michigan* approach, so called because they were first introduced by research groups at the Pittsburgh's and Michigan's university, respectively. In the former case each individual encodes a whole solution (a set of rules), while in the latter case an individual encodes a part of the solution (typically one rule). Both approaches present advantages and drawbacks. The Pittsburgh approach allows an easier control of the genetic search, but introduces a large redundancy that can lead to hard to manage populations and to individuals of enormous size. The Michigan approach, on the other hand, allows for cooperation and competition between different individuals, hence reduces redundancy, but requires sophisticated strategies, like co-evolution, for coping with the presence in the population of super individuals. Moreover in the Michigan approach it is generally difficult to evaluate the fitness of individuals independently.

A number of systems based on EAs for ICL have been proposed, like REGAL, G–NET, DOGMA, SIA01, GLPS and ECL. All the systems exploit in some way problem–specific knowledge. In the following we will give a brief description of the first five systems, while ECL is described in details in the next section. In particular we focus on how knowledge is incorporated and used.

REGAL exploits the explicit parallelism of GAs. In fact it consists of a network of genetic nodes fully interconnected and exchanging individuals at each generation. Each genetic node performs a GA. A supervisor node is used in order to coordinate these subpopulations. A first way in which REGAL exploits knowledge about the problem is by assigning a set of positive training examples to each genetic node. This set is periodically changed by assigning to each node a different set of positive examples. The sets assigned to each node contains a subset that consists of all positive examples that are not covered by any rule in any node. The purpose of this is to focus the genetic search on regions of the hypothesis space that are less populated. REGAL makes use of a priori knowledge about the problem by means

of an user supplied template. The template consists of a formula belonging to the hypothesis language such that every admissible rule can be obtained by the template. In this way the user can set a language bias on the representable rules, and can direct the search toward desired rules.

G–NET is a descendant of REGAL and is a distributed system. Like its predecessor, in G–NET a supervisor node computes the assignment of the positive concept instances to the genetic nodes, but in a slightly different way. The strategy adopted consists of addressing the search on the concept instances that are covered by poor hypotheses, without omitting to continue the refinement of the other hypotheses. G–NET exploits a priori knowledge by adopting a user supplied template, in the same way as REGAL does. Unlike REGAL, G–NET adopts ad-hoc variation operators. The crossover operator used is a combination of the two-point crossover with a variant of the uniform crossover, modified in order to perform either generalization or specialization of individuals. A mutation operator is used in order to generalize an individual and another one is used for the specialization. Both crossover and mutation operators enforce diversity, so that it is assured that in a genetic node there are no equal clauses.

DOGMA follows the same strategy of REGAL and G–NET for incorporating a priori knowledge by means of a user supplied template. DOGMA considers individuals at two distinct levels. On the lower level the Michigan approach is adopted, while on a higher level the Pittsburgh approach is used. On the lowest level the system uses fixed length chromosomes, which are manipulated by crossover and mutation operators. On the higher level chromosomes are combined into genetic families, through some special operators that can merge and break families. DOGMA follows the metaphor of competing families by keeping genetic operators, such as crossover and mutation, working on the lower level, and by building good blocks of chromosomes, while lifting selection and replacement to the family level. A family is determined by which portion of the background knowledge the family is allowed to use.

SIA01 adopts a different encoding. An high level representation is used for encoding clauses. SIA01 directly uses a FOL notation, by using predicates and their arguments as genes. In this way a variety of ILP operations can be directly applied to individuals, e.g., turning a constant into a variable or changing a variable with another variable. These kind of operations are not directly possible with an encoding like the one adopted by REGAL. Moreover, an order of generality among predicates is used by SIA01. In this way, during the evolutionary process, it is possible to generalize individuals also by replacing a predicate with a more general one.

GLPS is a GP system, where an individual encodes a whole logic program. An ad-hoc crossover operator is used, which can operate in various modalities: individuals are just copied unchanged to the next generation, individuals exchange a set of clauses, a number of clauses belonging to a particular rule are exchanged between the individuals, a number of literals belonging to a clause are exchanged.

3 ECL: An Evolutionary Concept Learner

ECL evolves if-then rules (Horn clauses) of the form $p(X, Y) \leftarrow r(X, Z), q(Y, a)$. consisting of atoms whose arguments are either variables (e.g. X, Y, Z) or constants (e.g. a). The atom $p(X, Y)$ is called the head of the clause, while the set of other atoms is called the body of the clause. The head predicate describes the target

concept, and the predicates in the body are in the background knowledge (BK), and represent the conditions that have to be satisfied in order for the clause to be true.

The background knowledge consists of a set of ground facts (i.e. clauses of the form $r(a, b) \leftarrow$ with a, b constants). Positive and negative examples of the target concept are also represented by facts. A clause is said to cover an example if the theory formed by the clause and the background knowledge logically entails the example. A clause has two interpretations: a declarative interpretation (universally quantified FOL implication) $\forall X, Y, Z(r(X, Z), q(Y, a) \rightarrow p(X, Y))$ and a procedural one (in order to solve $p(X, Y)$ solve $r(X, Y)$ and $q(Y, a)$). A set of clauses forms a logic program, which can directly be executed in the programming language Prolog.

The goal of ECL is then finding a Prolog program that models a given target concept, given a set of training examples and a background knowledge.

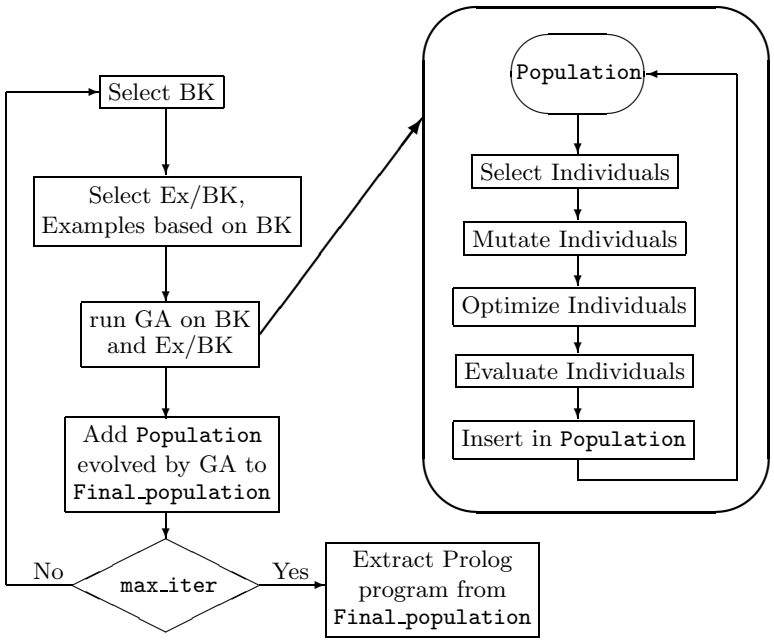

Fig. 1. A schematic description of ECL.

Figure 1 shows a schema of ECL. The fitness of a clause cl is its accuracy, that is, the percentage of correctly classified examples.

ECL uses a high level representation similar to the one used by SIA01 [4]. For instance, the clause $p(X, Y) \leftarrow r(X, Z), q(Y, a).$ is described by the sequence

$$\boxed{p, X, Y}\ ,\ \boxed{r, X, Z}\ ,\ \boxed{q, Y, a}$$

This representation allows the direct application of mutation operators based on standard ILP, and allows to compute directly the fitness using Prolog.

The system constructs iteratively a `Final_population` as the union of `max_iter` populations, where `max_iter` is a parameter. At each iteration, a portion BK of the background knowledge is selected by means of a random sampling. The selected background knowledge induces the selection of a subset of the training examples, those that can be covered by at least one fact of BK. The selected BK and relative examples are fed to a GA which evolves a `Population` of clauses by the repeated application of selection, mutation and optimization (right hand side of figure 1).

After `max_iter` runs of the GA, with possibly different portions of background knowledge (and examples) a logic program is extracted from `Final_population` using a procedure that incrementally builds a logic program from the empty set of clauses, by moving each time a most precise clause from `Final_population` to the actual program until the accuracy of the program does not decreases. Each time a clause from `Final_population` is moved, the examples it covers are discarded and the precision of the remaining clauses is recomputed.

For moving in the hypothesis space ECL makes use of standard ILP generalization and specialization operators. A clause is generalized by either deleting an atom from its body or by turning a constant into a variable. Specialization is performed by either adding an atom to the body of a clause or by turning a variable into a constant.

The corresponding four mutation operators act greedily on a clause as follows. When a clause is selected for mutation, one of the four mutations is chosen at random. Suppose, for instance, the mutation using the "delete an atom" operation is chosen. Then the "delete an atom" operation is applied independently to a number (specified by the user) of atoms in the body of the clause and the one yielding the best improvement in clause fitness is selected for application. The number of mutation possibilities considered are determined by the values of four parameters $N_1, \ldots, N_4$. Each mutation operator is equipped with a parameter which determines its degree of greediness. Different values of the parameters determine different search strategies: low values yield weak learning methods, like standard evolutionary algorithms, while high values yield more greedy learning methods, like Inductive Logic Programming systems. Table 1 shows for each mutation operator the associated parameter N_i, $1 \leq i \leq 4$.

Table 1. Mutation operators with associated parameter for controlling the greediness.

Mutation Operator	Associated N_i
Atom Deletion	N_1
Constant into Variable	N_2
Atom Addition	N_3
Variable into Constant	N_4

In the sequel, we perform a detailed experimental analysis of knowledge-based selection operators used in ECL.

4 Selection

Three selection mechanisms are supported by ECL, the first being the US selection operator introduced in [33] and used in REGAL, the other two are variants. The first variant is called Weighted US (WUS) selection operator [12], the second Exponentially Weighted US (EWUS) selection operator [10].

4.1 US Selection operator

The US operator works as follow:

1. the operator randomly selects n positive examples e_i, $1 \leq i \leq n$;
2. for each e_i an individual is selected. For all e_i a roulette wheel tournament is performed among individuals covering e_i. The winners of each tournament are selected for reproduction. The dimension of the sector associated to each individual is proportional to its quality. If an example e_i is not covered then a new individual covering e_i is created using a seed operator.

The US selection operator has various good characteristics. With the US selection operator individuals with a high coverage and with good fitness are favored. This is because individuals with a high coverage participates in more roulette wheel tournaments, and individuals with high fitness have wider sectors in the roulette wheel tournaments in which they participate. In this way a good coverage of positive examples is promoted. Speciation is obtained through the use of the US selection operator. In fact only individuals covering the same examples compete with each other for being selected.

The selection operator is called "universal suffrage" because positive examples can be viewed as voters, chromosomes as candidates to be elected. All examples have the same right (probability) of voting. The US operator does not distinguish between examples that are harder to cover, and this can favor the emergence of so called super–individuals that cover many (easy) examples. Super–individuals will be often selected, and this may lead to a population characterized by a low diversity. This phenomenon can negatively affect the solution, since there are less clauses to choose from, in order to build the final solution.

In REGAL this problem is tackled by using the distributed architecture of nodes where each node evolves a single population acting on a different set of examples, and the nodes co-evolve in order to favor the formation of different species. A supervisor process can shift the focus of the genetic search performed by each genetic node that constitute the system by simple assigning a different set of training example to the nodes. In this way the system promoted diversity.

4.2 WUS Selection Operator

The first variation of the US selection operator is represented by the WUS selection operator. The difference between this operator and the standard US selection operator lies in the first step of the selection. In the WUS operator examples do not have the same voting power, i.e. examples are no longer chosen at random. A weight is associated to each example, where smaller weights are associated to examples harder to cover. More precisely the weight for an example e_i is equal to:

$$w_i = \frac{|Cov(e_i)|}{|Pop|} \tag{1}$$

$1 \leq i \leq P$, being $|Pop|$ the population size and P the number of positive examples and $|Cov(e_i)|$ the number of individuals covering e_i. If the population is empty, then each example has the same weight. Examples are now chosen with a roulette wheel mechanism, where the sector associated to each example is inversely proportional to its weight. So the less individuals cover an example, the more chances that example has of being selected in the first step of the selection.

4.3 EWUS Selection Operator

The WUS selection operator selects with higher probability examples that are harder to cover, by means of the weights assigned to each example. The EWUS selection operator continues this line, with the difference that now the weight w_i of an example e_i, $1 \leq i \leq P$, is given by the following formula:

$$w_i = \frac{e^{-|Cov(e_i)|}}{\sum_{j=1}^{P} e^{-|Cov(e_j)|}} \tag{2}$$

Examples are still selected with a roulette wheel mechanism, where the sector associated to each example is proportional to its weight. In this way the selection pressure toward examples harder to cover will be much higher than in the WUS selection operator.

In both the WUS and the EWUS operators, the second step of the selection is the same as the one performed by the US operator. In both the WUS and the EWUS selection operators, at the end of each generation the weights assigned to the positive examples are updated.

Table 2. Weights assigned to each example by the WUS and the EWUS operator.

Operator	$w(e_1)$	$w(e_2)$	$w(e_3)$
WUS	0.4	0.6	0.2
EWUS	0.2477	0.0901	0.6653

Example 4.1 Suppose we have three positive examples e_1, e_2, e_3, and five individuals: $\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5$. Let $Cov(e_1) = \{\varphi_1, \varphi_2\}$, $Cov(e_2) = \{\varphi_2, \varphi_3, \varphi_4\}$, $Cov(e_3) = \{\varphi_5\}$. Then each example has the same probability of being selected in the first step of the US selection operator. The weights assigned to each examples by the WUS and the EWUS operators are given in table 2.

The selection probability relative to each example e_i, $1 \leq i \leq 3$, in the WUS operator is inversely proportional to the weight assigned to e_i. The difference between the WUS and the EWUS operators is that in the former operator the weights are linear, while in the latter the weights are exponential. For this reason, e_2 has

a small chance of being selected by the EWUS operator, while the chance that e_2 has of being selected by the WUS operator is higher. Another observation is that e_3 is the example that will be likely selected by the EWUS operator. Also in the WUS operator e_3 has a probability of being selected higher than the one of other examples. However this probability is much smaller in the WUS operator than in the EWUS operator.

It can be seen e_2 has small chance of being selected by the EWUS operator, while the WUS operator is more in favor of e_2 than EWUS . Moreover e_3 is likely to be selected by the EWUS operator, the WUS operator e_3 is less biasing the selection in favour of e_3.

Table 3. Characteristics of the propositional datasets used in the experiments performed in this chapter. From left to right: number of examples (positive, negative), of continuous attributes, of nominal attributes, and of elements of the BK.

Dataset	Examples (+,-)	Continuous	Nominal	BK Dim.
Accidents	256 (62,194)	3	2	15770
Australian	690 (307,383)	6	8	9660
Breast	699 (458,241)	10	0	6275
Congestions	256 (66,190)	3	2	15770
Crx	690 (307,383)	6	9	10283
Echocardiogram	74 (24,50)	5	1	750
German	1000 (700,300)	24	0	24000
Glass2	163 (87,76)	9	0	1467
Heart	270 (120,150)	13	0	3510
Hepatitis	155 (123,32)	6	13	2778
Ionosphere	351 (225,126)	34	0	11934
Mutagenesis	188 (125,63)	6	4	13125
Pima-Indians	768 (500,268)	8	0	6144
Vote	435 (267,168)	0	16	6568

5 Experiments

In order to analyze the effect of the three selection operators, we performed a number of experiments, using both propositional and relational datasets. In these experiments we focus our attention on the diversity of the final population evolved by ECL and on the number of positive examples that remains uncovered at the end of learning process when using the different selection mechanisms. We also verify if the fact of having more diversity in the population is positively reflected in the quality of the found solutions.

The features of the datasets used are shown in table 3. The mutagenesis dataset originates from [9], while the accidents and the congestions are part of the traffic dataset that originates from [14, 15]. In the original traffic dataset, three classes

Table 4. Parameter settings used in the experiments: *gen* is the number of generations performed by the GA, *sel* is the number of individuals selected per generation, N_i, $i \in [1,4]$, are the greediness parameters of the mutation operators, *lc* is the maximum length of a clause, and *pbk* is the probability of selecting a BK fact.

Dataset	pop size	gen	sel	max_iter	Ni	lc	pbk
Accidents	20	5	5	1	(14,2,2,2)	8	1.0
Australian	50	10	15	1	(4,4,4,4)	6	0.4
Breast	50	5	5	1	(3,3,3,3)	5	1.0
Congestions	30	10	10	1	(10,2,2,2)	8	1.0
Crx	80	20	15	1	(4,4,4,4)	7	1
Ecochardiogram	40	8	10	10	(4,4,4,4)	4	0.7
German	200	10	10	2	(3,4,3,4)	6	0.4
Glass2	150	15	20	3	(2,8,2,9)	5	0.8
Heart	50	10	15	1	(4,4,4,4)	6	1
Hepatitis	50	10	10	5	(4,4,4,4)	7	0.2
Ionosphere	50	10	15	6	(4,8,4,8)	6	0.2
Mutagenesis	50	10	15	2	(4,8,2,8)	3	0.8
Pima-Indians	60	10	7	5	(2,5,3,5)	4	0.2
Vote	80	10	15	2	(3,6,2,5)	4	0.5

are presented: accidents, congestions and noncritical. We analyze here the classes separately, but the results can be united and the class noncritical can be considered as the default class. These datasets are relational datasets. The other eleven datasets are propositional, and are taken from [7]. Table 4 reports the parameter settings used in the experiments. Continuous values are treated by means of inequalities [13, 11], in a way similar to the ones proposed in [2, 18, 1, 27, 5, 6]. A set of boundary points is used in order to initialize and modify inequalities during the evolution process in a local supervised discretization process. In the experiments, we use ten–fold cross validation, where each dataset is divided in ten disjoint sets of similar size, one of these sets forms the test set, and the union of the remaining nine the training set. Each selection operator is then tested by running ECL with the particular operator three times, using different random seeds, on each training set. The output of each run is evaluated on the corresponding test set (so ECL is run 30 times per dataset for each selection operator).

Table 5 shows the results regarding the diversity, the average number of positive examples uncovered at the end of the evolution and the average number of individuals covering a positive example, obtained by ECL on the various datasets when the different selection mechanisms are employed. We consider two clauses to be diverse if they do not cover the same set of training examples.

It can be seen that in most of the cases, the EWUS selection operator leads to a population characterized by a higher diversity. Only in two cases (the breast and the pima-indians datasets) the population evolved with the use of the standard US operators has a higher diversity. However, also in these two cases, the diversity of the two populations are comparable. The WUS selection operators was less successful than the EWUS selection operator in promoting diversity. The diversity obtained by the WUS operator is comparable to the diversity obtained by the US operator.

Table 5. Results regarding the average diversity of the evolved final population, the average number of positive examples uncovered at the end of the evolution (Unc) and the average number of individuals covering a positive example (Avg). Standard deviation between brackets.

Dataset	Mechanism	Diversity	Unc	Avg
Accidents	US	12.7 (3.12)	1.4 (0.70)	17.92 (4.00)
	WUS	11.2 (1.55)	1.2 (0.92)	17.73 (3.21)
	EWUS	15.4 (1.78)	0.3 (0.48)	14.45 (3.48)
Australian	US	11.9 (5.74)	10.5 (4.20)	38.54 (5.09)
	WUS	9.7 (3.20)	11.9 (3.84)	41.38 (1.91)
	EWUS	29.7 (4.45)	1.8 (2.20)	24.52 (3.21)
Breast	US	19.1 (1.85)	10.8 (5.71)	16.74 (2.03)
	WUS	18.6 (1.58)	10.5 (4.11)	15.59 (2.24)
	EWUS	17.8 (3.46)	7.2 (5.12)	10.41 (1.88)
Congestions	US	13.1 (2.84)	4.3 (2.17)	17.16 (4.33)
	WUS	12.8 (3.15)	2.0 (1.05)	15.32 (2.63)
	EWUS	13.2 (2.44)	0.4 (0.70)	10.70 (2.52)
Crx	US	3.2 (0.79)	21.5 (2.95)	70.49 (1.47)
	WUS	4.6 (1.51)	19.5 (6.47)	68.70 (2.57)
	EWUS	28.1 (10.70)	4.8 (3.85)	29.24 (8.03)
Echocardiogram	US	142.1 (19.34)	0.87 (0.03)	259.86 (17.40)
	WUS	140.3 (20.16)	0 (0.0)	259.14 (14.28)
	EWUS	154.5 (15.64)	0 (0.0)	216.52 (23.28)
German	US	49.7 (7.27)	1.2 (0.03)	166.22 (6.34)
	WUS	50.1 (8.45)	0 (0.0)	168.43 (7.00)
	EWUS	71.9 (4.65)	0 (0.0)	144.2 (5.98)
Glass2	US	104.3 (24.21)	0.6 (0.04)	403.49 (4.48)
	WUS	123.8 (8.73)	0 (0.0)	399.452 (4.65)
	EWUS	180.4 (30.00)	0 (0.0)	356.142 (19.50)
Heart	US	11.4 (2.01)	7.1 (3.45)	32.85 (1.84)
	WUS	11.6 (4.35)	5.1 (3.07)	31.78 (3.27)
	EWUS	41.7 (3.20)	0.9 (0.57)	28.57 (3.11)
Hepatitis	US	43.7 (5.82)	1.23 (0.02)	229.56 (3.57)
	WUS	45.3 (4.99)	0 (0)	230.2 (3.56)
	EWUS	58.0 (6.50)	0 (0)	221.31 (2.62)
Ionosphere	US	126.6 (8.64)	1.2 (0.18)	160.42 (2.74)
	WUS	122.9 (9.69)	0 (0.0)	260.93 (11.73)
	EWUS	195.0 (6.32)	0 (0.0)	222.07 (9.60)
Mutagenesis	US	29.6 (5.87)	1.2 (0.43)	119.66 (5.13)
	WUS	30.00 (5.696)	0 (0.0)	120.281 (4.07)
	EWUS	41.8 (7.60)	0 (0.0)	109.86 (7.18)
Pima-Indians	US	22.00 (2.36)	0.0 (0.0)	58.71 (2.10)
	WUS	21.6 (1.78)	0.0 (0.0)	57.74 (2.85)
	EWUS	20.3 (1.34)	0.0 (0.0)	51.37 (2.25)
Vote	US	29.4 (3.53)	1.1 (0.32)	129.82 (5.73)
	WUS	30.0 (4.42)	0.2 (0.42)	132.39 (3.87)
	EWUS	48.0 (9.61)	0 (0.0)	127.38 (2.18)

Table 6. Results regarding the average accuracy and the average simplicity obtained with the use of the different selection operators. Standard deviation between brackets.

Dataset	Mechanism	Accuracy	Simplicity
Accidents	US	0.92 (0.01)	3.2 (1.14)
	WUS	0.94 (0.02)	3.3 (0.83)
	EWUS	0.95 (0.02)	3.55 (0.49)
Australian	US	0.85 (0.04)	2.2 (1.23)
	WUS	0.85 (0.02)	2.6 (0.97)
	EWUS	0.85 (0.01)	5.50 (1.43)
Breast	US	0.94 (0.03)	7.0 (1.05)
	WUS	0.94 (0.03)	6.9 (1.59)
	EWUS	0.95 (0.02)	8.6 (1.65)
Congestions	US	0.92 (0.02)	3.30 (0.82)
	WUS	0.93 (0.02)	3.20 (1.03)
	EWUS	0.94 (0.02)	3.55 (0.49)
Crx	US	0.85 (0.04)	1.6 (0.70)
	WUS	0.85 (0.04)	1.9 (0.99)
	EWUS	0.84 (0.01)	5 (1.76)
Echocardiogram	US	0.73 (0.03)	2.8 (0.79)
	WUS	0.76 (0.03)	2.5 (0.71)
	EWUS	0.74 (0.01)	3.00 (0.81)
German	US	0.74 (0.03)	8.2 (2.10)
	WUS	0.73 (0.04)	8.7 (2.21)
	EWUS	0.74 (0.01)	11.7 (0.24)
Glass2	US	0.84 (0.02)	2.7 (0.67)
	WUS	0.83 (0.02)	2.7 (0.95)
	EWUS	0.85 (0.01)	3.7 (1.06)
Heart	US	0.80 (0.05)	3 (0.94)
	WUS	0.76 (0.04)	2.7 (0.67)
	EWUS	0.81 (0.03)	2.9 (0.74)
Hepatitis	US	0.80 (0.03)	5.6 (1.17)
	WUS	0.81 (0.04)	6.7 (1.16)
	EWUS	0.83 (0.02)	6.3 (1.25)
Ionosphere	US	0.87 (0.02)	8.0 (2.36)
	WUS	0.88 (0.03)	8.4 (1.35)
	EWUS	0.89 (0.02)	12.0 (1.76)
Mutagenesis	US	0.87 (0.02)	3.1 (0.74)
	WUS	0.90 (0.02)	2.9 (0.99)
	EWUS	0.88 (0.01)	4.6 (0.84)
Pima-Indians	US	0.75 (0.02)	7.5 (1.65)
	WUS	0.74 (0.05)	8.1 (2.23)
	EWUS	0.77 (0.02)	7.9 (1.29)
Vote	US	0.92 (0.04)	3.1 (1.20)
	WUS	0.93 (0.04)	3.4 (1.51)
	EWUS	0.94 (0.02)	3.7 (1.06)

Generally, there are also less uncovered positive examples at the end of the evolution when the EWUS selection operator is used. Also this results confirms the fact that the population evolved by ECL with the EWUS selection operator is more spread out through the hypothesis space than the populations evolved with the use of the other two selection mechanisms.

The last column of table 5 presents the average number of individuals in the final population covering a positive examples. Also these results confirm the effectiveness of the EWUS selection operator in promoting diversity.

Table 6 reports the average accuracy and the average simplicity obtained by ECL when the different selection mechanisms are used. It can be notice that the EWUS operator generally leads also to a better accuracy. Only on the crx dataset the use of the US selection operator lead to a slightly better accuracy, despite the fact that the population evolved was characterized by a much lower diversity than the population evolved with the use of the EWUS operator. This can be explained by the fact that the population evolved for this dataset with the use of the US operator, contained only few "good" clauses. These clauses were also present in the population evolved by ECL with the use of the EWUS operator. However these clauses do not have a high precision. When the procedure for extracting a final solution from the final population is applied to the population evolved with the use of the EWUS operator, other clauses with higher precision are inserted into the emerging solution before these "good" clauses. This choice had, in some case, negatively affected the quality of the extracted solution.

In almost all the cases the solutions found by ECL with the use of the EWUS operator are less simple that those found with the use of the other selection operators. In some cases, like in the ionosphere, the crx and the australian datasets, the difference is evident, while in other, e.g., in the glass2, the accidents and the congestions dataset, the simplicity obtained by the three methods is comparable. An explanation for this is that the population evolved with the use of the EWUS operators is characterized by an higher diversity. This can be notice for example in the ionosphere, the crx and the australian cases, where the solutions found are more complex and where the population evolved with the EWUS operator is much more diverse than the other two populations. More diversity in the population means that more clauses are taken in consideration for being added to the final solution, thus the logic program extracted can be made of more clauses.

For what regards the computational cost of the three selection mechanisms, the US selection operator is slightly more efficient. This is because the other two operators have to update the weights assigned to positive examples at each generation.

6 Conclusion

In this chapter we have described the hybrid evolutionary system for inductive concept learning ECL. The system exploits knowledge about the particular domain tackled by means of the selection operator and of the variation operators adopted.

We have focused our attention on the selection operator, and particularly on how the selection mechanism can help the system in promoting diversity by means of considering the difficulty of examples. From the experiments, it emerges that taking into account the difficulty of examples is an effective way for achieving diversity in the population.

The introduction of the two weighted variants of the US selection operator was motivated by the following aims:

1. Good coverage of positive examples.
2. Diversity in the population.

One would usually like to have the final hypothesis found by the GA to cover as many positive examples as possible and as few negative examples as possible. The fitness function deals with the coverage of positive examples. The variants of the US selection operator deal also with the second aspect. By having the learning system focusing more and more on difficult example to cover, it is easier to have each positive example covered by at least one individual, as well as to have more diversity in the population.

Maintaining a good diversity in the population would also lead to a good coverage of examples. Generally it is a good idea to have the population spread across the hypothesis space, so that all the areas of the hypothesis space can be searched. Different methods for achieving species and niches formations, as well as for maintaining diversity in the population, have been proposed. Among these crowding [8] and sharing function [20] are two popular methods. These methods imply the use of some distance measure in order to establish the similarity between individuals. Establishing a distance measure is not an easy task when the rules are expressed in FOL. The US selection operator promote species and niches formation without the need of any distance measure.

The US selection operator considers all the examples as equally important. In this way the search can focus mainly in the areas where the concentration of positive examples is higher, and seldom touches less inhabited areas of the hypothesis space. With the weighted variants of the US operator, examples difficult to cover will have higher chance of being selected. In this way, if we have many individuals covering the same subset of training examples, those individuals will be seldom selected for reproduction, because that area of the hypothesis space is already crowded. Instead we will select, or create, individuals that occupy, or will occupy, less exploited regions.

Formally, the probability that an individual φ has of being selected can be written as:

$$P(\varphi) = \sum_{e_i \in p_\varphi} P(\varphi|e_i) \cdot P(e_i) \tag{3}$$

where $P(\varphi|e_i)$, the probability of φ being selected conditioned to the selection of example e_i in p_φ, is equal to

$$\frac{f(\varphi)}{\sum_{\varphi \in Cov(e_i)} f(\varphi)} \tag{4}$$

with $f(\varphi)$ the fitness of individual φ. The three US based selection operators induce different probability $P(e_i)$ of selection of example e_i.

In the US selection operator $P(e_i) = \frac{1}{P}$. No distinction is made among positive examples, so individuals with a high recall are favored in this scheme, because they cover many positive examples. A possible effect of this could be the presence of super individuals, that will dominate the population and will be often selected, leading to a low diversity in the population. Also some examples could stay uncovered, because not selected and difficult to cover.

In the WUS selection operator $P(e_i)$ is proportional to $\frac{1}{w_i+1}$, with w_i defined in equation 1. In this way WUS tries to favor not only examples that are uncovered, but in general it favors examples that are difficult to cover. Individuals with a high recall will still have more changes of being selected. However the system will focus more and more on examples harder to cover. In this way it is more probable that at the end of the evolution process all positive examples are covered. The weights given to examples by the WUS operator increase linearly with each individual that cover the example.

This means, for instance, that an example covered by one individual has almost the same probability of being selected as it has an example that is not covered by any individual. Instead, in the EWUS operator, the probability that an example has of being selected is $P(e_i) = w_i$, with w_i defined in equation 2. In this case the probability of selection of each example change exponentially with each individual covering the example. Thus, in the EWUS operator, examples covered by few individuals have higher probability of being selected.

In this way a good coverage of positive examples is encouraged, leading also to a good diversity in the population. Diversity is promoted because individuals that cover many examples are selected only if there are not multiple copies of them. This is because if there are multiple copies of an individual, the examples covered by these superindividuals will be considered as easy, since they are covered by many individuals. Therefore the examples covered by the superindividuals will have little chances of being selected in the first step of the selection.

References

1. J. Aguilar-Ruiz, J. Riquelme, and M. Toro, *Evolutionary learning of hierarchical decision rules*, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 33(2) (2003), pp. 324–331.
2. J. S. Aguilar-Ruiz, J. C. Riquelme, and C. D. Valle, *Improving the evolutionary coding for machine learning tasks*, in European Concefence on Artificial Intelligence, ECAI'02, Lyon, France, 2002, IOS Press, pp. 173–177.
3. C. Anglano, A. Giordana, G. L. Bello, and L. Saitta, *An experimental evaluation of coevolutive concept learning*, in Proc. 15th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA, 1998, pp. 19–27.
4. S. Augier, G. Venturini, and Y. Kodratoff, *Learning first order logic rules with a genetic algorithm*, in The First International Conference on Knowledge Discovery and Data Mining, U. M. Fayyad and R. Uthurusamy, eds., Montreal, Canada, 20-21 1995, AAAI Press, pp. 21–26.
5. J. Bacardit and J. M. Garrell, *Evolution of multi-adaptive discretization intervals for A rule-based genetic learning system*, in Proceedings of the VIII Iberoamerican Conference on Artificial Intelligence (IBERAMIA'2002), LNAI vol. 2527, Springer, 2002, pp. 350–360.
6. J. Bacardit and J. M. Garrell, *Evolving multiple discretizations with adaptive intervals for a Pittsburgh rule-based learning classifier system*, in Genetic and Evolutionary Computation – GECCO-2003, vol. 2724 of LNCS, Chicago, 12-16 July 2003, Springer-Verlag, pp. 1818–1831.
7. C. Blake and C. Merz, *UCI repository of machine learning databases*, 1998.

8. K. De Jong, *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*, PhD thesis, Dept. of Computer and Communication Sciences, University of Michigan, Ann Arbor, MI, 1975.
9. A. Debnath, R. L. de Compadre, G. Debnath, A. Schusterman, and C. Hansch, *Structure-Activity Relationship of Mutagenic Aromatic and Heteroaromatic Nitro Compounds. Correlation with molecular orbital energies and hydrophobicity*, Journal of Medical Chemistry, 34(2) (1991), pp. 786–797.
10. F. Divina, M. Keijzer, and E. Marchiori, *Non-universal suffrage selection operators favor population diversity in genetic algorithms*, in Benelearn 2002: Proceedings of the 12th Belgian-Dutch Conference on Machine Learning (Technical report UU-CS-2002-046), Utrecht, The Netherlands, 4 December 2002, pp. 23–30.
11. ———, *Evolutionary concept learning with constraints for numerical attributes*, in BNAIC 2003: Proceedings of the Belgian-Dutch Conference on Artificial Intelligence, Nijmegen, The Netherlands, 23-24 October 2003, pp. 107–114.
12. F. Divina and E. Marchiori, *Evolutionary concept learning*, in GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, 9-13 July 2002, Morgan Kaufmann Publishers, pp. 343–350.
13. ———, *Handling continuous attributes in an evolutionary inductive learner*, Tech. Rep. IR-AI-005, Vrije Universiteit Amsterdam, 2003.
14. S. Džeroski, N. Jacobs, M. Molina, and C. Moure, *ILP experiments in detecting traffic problems*, in European Conference on Machine Learning, 1998, pp. 61–66.
15. S. Dzeroski, N. Jacobs, M. Molina, C. Moure, S. Muggleton, and W. V. Laer, *Detecting traffic problems with ILP*, in International Workshop on Inductive Logic Programming, 1998, pp. 281–290.
16. A. Freitas, *Data Mining and Knowledge Discovery with Evolutionary Algorithms*, Spinger-Verlag, Berlin, unknown 2002.
17. A. Giordana and F. Neri, *Search-intensive concept induction*, Evolutionary Computation, 3 (1996), pp. 375–416.
18. R. Giráldez, J. S. Aguilar-Ruiz, and J. C. Riquelme, *Natural coding: A more efficient representation for evolutionary learning*, in Genetic and Evolutionary Computation – GECCO-2003, vol. 2723 of LNCS, Chicago, 12-16 July 2003, Springer-Verlag, pp. 979–990.
19. D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, 1989.
20. D. E. Goldberg and J. Richardson, *Genetic algorithms with sharing for multimodal function optimization*, in Proc. of 2nd Int'l Conf. on Genetic Algorithms, Morgan Kaufmann Publishers, 1987, pp. 41–49.
21. J. Hekanaho, *Background knowledge in GA-based concept learning*, in International Conference on Machine Learning, 1996, pp. 234–242.
22. J. Hekanaho, *DOGMA: a GA based relational learner*, in Proceedings of the 8th International Conference on Inductive Logic Programming, D. Page, ed., LNAI 1446, Springer Verlag, 1998, pp. 205–214.
23. C. Janikow, *A knowledge intensive genetic algorithm for supervised learning*, Machine Learning, 13 (1993), pp. 198–228.
24. K. D. Jong, W. Spears, and D. Gordon, *Using Genetic Algorithms for Concept Learning*, Machine Learning, 13(1/2) (1993), pp. 155–188.

25. C. J. Kennedy and C. Giraud-Carrier, *A depth controlling strategy for strongly typed evolutionary programming*, in GECCO 1999: Proceedings of the First Annual Conference, Morgan Kauffman, July 1999, pp. 1–6.
26. M. Kubat, I. Bratko, and R. Michalski, *A review of Machine Learning Methods*, in Machine Learning and Data Mining, R. Michalski, I. Bratko, and M. Kubat, eds., John Wiley and Sons Ltd., Chichester, 1998.
27. W. Kwedlo and M. Kretowski, *An evolutionary algorithm using multivariate discretization for decision rule induction*, in Principles of Data Mining and Knowledge Discovery, 1999, pp. 392–397.
28. K. Leung and M. Wong, *Genetic logic programming and applications*, IEEE Expert, 10(5) (1995), pp. 68–76.
29. T. Mitchell, *Generalization as search*, Artificial Intelligence, 18 (1982), pp. 203–226.
30. ———, *Machine Learning*, Series in Computer Science, McGraw-Hill, 1997.
31. S. Muggleton, *Inductive logic programming: issues, results and the challenge of learning language in logic*, Artificial Intelligence, 114 (1999), pp. 283–296.
32. S. Muggleton and L. D. Raedt, *Inductive logic programming: Theory and methods*, Journal of Logic Programming, 19-20 (1994), pp. 669–679.
33. F. Neri and L. Saitta, *Analysis of genetic algorithms evolution under pure selection*, in Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA, 1995, pp. 32–39.
34. M. L. Wong and K. S. Leung, *Inducing logic programs with genetic algorithms: The genetic logic programming system*, in IEEE Expert 10(5), 1995, pp. 68–76.