Evolutionary Computation Framework for Learning from Visual Examples[1]

Krzysztof Krawiec

Institute of Computing Science, Poznan University of Technology,
Piotrowo 3A, 60965 Poznan, Poland
`krawiec@cs.put.poznan.pl`

Abstract. This paper investigates the use of evolutionary programming for the search of hypothesis space in visual learning tasks. The general goal of the project is to elaborate human-competitive procedures for pattern discrimination by means of learning based on the training data (set of images). In particular, the topic addressed here is the comparison between the 'standard' genetic programming (as defined by Koza [13]) and the genetic programming extended by local optimization of solutions, so-called *genetic local search*. The hypothesis formulated in the paper is that genetic local search provides better solutions (i.e. classifiers with higher predictive accuracy) than the genetic search without that extension. This supposition was positively verified in an extensive comparative experiment of visual learning concerning the recognition of handwritten characters.

Keywords: visual learning, genetic programming, genetic local search, learning from examples.

1 Introduction

The search for an appropriate processing and representation of the visual information data is the most complex part of the design of computer vision systems. This task is weekly formalized so far, therefore in most cases the human designer is made responsible for it (see [8], p.657). Requiring significant body of knowledge, experience and even intuition, this work is usually tedious and expensive. The solutions (representations, algorithms, etc.) chosen by the human expert, are subjective and sub-optimal in terms of system performance. For instance, if a machine learning classifier [22,25] is used for the decision making based on given image representation (what is a common approach), the hypothesis space searched during its training is limited by that representation. This phenomenon may disable the training process from discovering bene-

[1] Some results quoted in this paper have been previously reported in [18].

ficial solutions for the considered vision task (e.g. recognition, analysis, or interpretation).

To overcome these difficulties, in our research we aim at synthesizing the complete image analysis and interpretation programs without splitting them explicitly into the stages of feature extraction and classification. As a result, the learning process is no more limited by the representation predefined by the human expert, but encompasses also image processing and analysis. Thus, we follow here the paradigm of *direct* approach to pattern recognition, occupied so far mostly by the artificial neural networks. However, instead of subsymbolic reasoning and gradient-based learning, we employ the paradigm of evolutionary computation for the search of the space of image analysis programs.

This paper is organized as follows. The next section introduces shortly the evolutionary computation paradigm and, in particular, its variety called genetic programming. Section 3 outlines the GPVIS software environment for experimenting with evolutionary search for pattern analysis procedures. Sections 4 and 5, being the main parts of this contribution, describe two novel extensions introduced into GPVIS, which go beyond standard evolutionary computation paradigm: non-scalar evaluation of individuals and their local optimization. Section 6 outlines another extensions, some being currently under development. Section 7 describes selected computational experiments and presents their results. Section 8 groups conclusions and outlines the most promising further research directions.

2 Evolutionary Computation and Genetic Programming

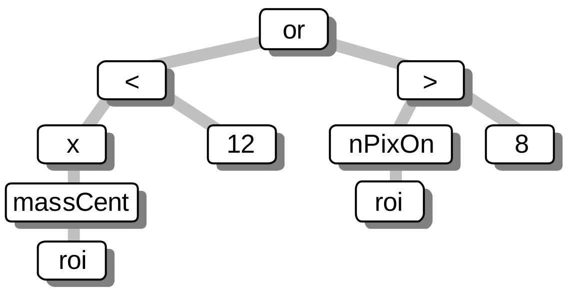
Evolutionary computation (EC) [4,9] has been used in machine learning (ML) and pattern recognition community for quite a long time [24,25]. Now it is widely recognized as a useful metaheuristics or even as one of ML paradigms [22,25]. It is highly appreciated due to its ability to perform global parallel search of the solution space with low probability of getting stuck in local optima of evaluation function. Its most renowned ML-related applications include feature selection [31] and concept induction [5,6]. In this paper, we focus on the latter of the aforementioned tasks, with individual solutions implementing hypotheses considered by the system being trained (from now on, the terms ‘solution’ and ‘hypothesis’ will be used interchangeably). For the sake of brevity we skip the presentation of EC fundamentals, assuming reader’s familiarity with them.

Genetic programming (GP) is a specific paradigm of evolutionary computation proposed by Koza [14] employing more sophisticated representation of solutions than ‘plain’ genetic algorithms, which use strings over binary alphabet. Thus, solutions in GP, usually LISP expressions, implement solutions in a more direct way. GP is reported to be very effective in solving a broad scope of learning and optimization problems, including the impressive achievement of creating human-competitive designs for the controller synthesis problems, some of which have been even patented [15].

3 Genetic Programming for Visual Learning

Evolutionary computation found some applications in image processing and analysis (see, e.g., [2]). However, there are relatively few research projects, which aim at combining image processing and analysis with learning, i.e. the *visual learning*, meant as the search for pattern recognition programs [12,28,26,16]. Only a small fraction of research focuses on inducing the complete image analysis programs based on training examples (images), a paradigm which is very promising and universal in our opinion and which is subject of our research on *genetic programming-based visual learning*. From the machine learning viewpoint [22,25], we focus on the paradigm of supervised learning from examples, employing the genetic programming for the hypothesis representation and for the search of hypothesis space. The candidate programs performing image analysis and recognition are evaluated on a set of training cases (images). In particular, the solution's fitness reflects the accuracy of classification it provides on the training set.

The solutions (pattern recognition procedures) are expressed in GPVIS [20], an image analysis-oriented language encompassing a set of operators implementing simple feature extraction, region-of-interest selection, and arithmetic and logical operations. GPVIS allows formulating the complete pattern recognition program unnecessarily of an external machine learning classifier. This is the opposite of conventional approach, where the processing is divided into the stages of feature extraction and the reasoning.



```
(or
  (<
    (x
      (massCent
        (roi 19 8 2 4)))
    12)
  (>
    (nPixOn
      (roi 7 23 10 18))
```

Fig. 1. Tree-like and textual representations of an exemplary solution formulated in GPVIS language (numerical values omitted in the former representation).

Figure 1 shows a simple example of image analysis program formulated in GPVIS, which has the following interpretation: *if* the x-coordinate of the visual mass center of the rectangular region of interest (*roi*) with coordinates of upper left corner (19,8) and lower right corner (2,4) is less than 12, *or* there are more than 8 pixels in the 'on' state in another region of interest, *then* the return value is *true*. The returned value implies decision in the context of particular task (e.g. classification).

4 Non-scalar evaluation of individuals

Similarly to other metaheuristics, like local search, tabu search, or simulated annealing, the genetic search requires an existence of an evaluation function f. That function guides the search process and is of crucial importance for its final outcome. In inductive learning, f should estimate the predictive ability of the particular hypothesis. In the simplest case, it could be just the accuracy of classification provided by the hypothesis on the training set. However, in practice more sophisticated forms of f are usually applied to prevent the undesired overfitting phenomenon resulting from characteristic for GP overgrowth of solutions. One possible remedy is to apply here the multiple train-and-test approach (so-called *wrapper*) on the training set or to introduce an extra factor penalizing too complex hypotheses, either explicitly or in a more concealed manner (as, for instance, in the minimum description length principle [26]).

The primary claim of this paper is that scalar evaluation reflects well the hypothesis utility, reveals however some shortcomings when used for hypothesis comparison. In particular, scalar evaluation imposes a complete order on the solution space and therefore forces the hypotheses to be always comparable. That seemingly obvious feature may significantly deteriorate the performance of the search, as illustrated in the following example.

Example 1. For a hypothesis h, let $C(h)$ denote the subset of examples from a non-empty training set T that it correctly classifies ($C(h)\subseteq T$). Let the hypotheses be evaluated by means of the scalar evaluation function f being the accuracy of classification of h on T, i.e. $f(h) = |C(h)| / |T|$. Let us consider three hypotheses, a, b, and c, for which $|C(a)| > |C(b)| = |C(c)|$. Thus, with respect to f, hypotheses b and c are of the same quality and are both worse than a.

Let us now stress that, due to its aggregating nature, scalar evaluation ignores the more sophisticated mutual relations between a, b and c, for instance the set-theoretical relations between $C(a)$, $C(b)$ and $C(c)$. If, for instance, $C(b) \subset C(a)$, we probably would not doubt the superiority of a over b. But what about the relation between a and c, assuming that $C(c) \not\subset C(a)$ and $|C(c) \cap C(a)| << |C(a)|$? In such a case, although a classifies correctly more examples than c, there is a (potentially large) subset of examples $C(c) \setminus C(a)$, which it does not cope with, while they are successfully classified by c. Thus, superiority of a over c is rather questionable. Moreover, if also $C(a) \not\subset C(c)$, the question concerning mutual relation between a and c should intuitively remain without answer. ■

This example shows us that scalar evaluation applied to hypothesis comparison can be prejudiced against hypotheses that are only slightly worse, but significantly different with respect to the 'behavior' on the (training) data. The primary reason for this shortcoming is the compensatory nature of the summation operator used in accuracy of classification. Such measures may yield similar or even equal values for very different hypotheses. An important implication for the search of hypothesis space (especially evolutionary search) is that some novel and 'interesting' hypotheses, which could initiate useful search directions, may be discarded in the search.

This limitation of scalar aggregating measures is well known in multiple-criteria decision aid, where models alternative to the functional one have been elaborated to overcome that difficulty (see, for instance, [30]). Following those ideas, in [17] we proposed the *relational* method of hypothesis evaluation and selection instead of the *functional* one. In particular, we suggest that when the considered hypotheses 'behave' in a significantly different way on the training set, we should allow them to remain *incomparable*. Allowing incomparability of solutions implies modifying the hypothesis space structure from the *complete* order to the *partial* order. To model such a structure, we propose to use the binary *outranking relation*[2], denoted thereafter by '$\geq$' (see, for instance, chapter 5 of [30]). For a pair (a,b) of solutions, $a \geq b$ means '*a* is *at least as good* as *b*'. Then, exactly one of the four following cases is possible:

? *a* is *indiscernible* with *b* ($a \geq b$ and $b \geq a$), or
? *a* is *strictly better* than *b* ($a \geq b$ and not $b \geq a$), or
? *b* is *strictly better* than *a* ($b \geq a$ and not $a \geq b$), or
? *a* and *b* are *incomparable* (neither $a \geq b$ nor $b \geq a$).

Partial order has a natural graphical representation of directed graph. The nodes of an outranking graph correspond to hypotheses, whereas arcs express the outranking. Particularly, the potentially best solutions should not be outranked, and are therefore represented in such graph by initial (predecessor-free) nodes. Note also that outranking is in general reflexive and non-symmetric.

In the context of inductive learning from examples, the idea is to get rid of the aggregating nature of scalar evaluation measures and to go more into detail by analyzing the behavior of hypotheses on particular instances from the training set. Intuitively, the need of incomparability grows with the dissimilarity between the compared hypotheses and becomes especially important when their scalar evaluations are relatively close. Example 1 showed us that it makes sense to base the comparison of a pair of hypotheses (a,b) on the set difference of the sets of properly classified instances ($C(a)$ and $C(b)$, respectively). In particular, the more examples belong to $C(b) \setminus C(a)$, the less likely should be the outranking $a \geq b$.

To avoid confusions, it is important to stress that the partial order imposed by hypothesis outranking as defined above refers to the 'behavior' of the hypothesis on the training data and therefore it has nothing to do with the orders based on the hypothesis representation, which are also often considered in the literature (e.g. the partial order of decision trees used by top-down decision tree inducers). Therefore, the idea of pairwise hypothesis comparison is universal in the sense that it does not make any assumption about knowledge representation used by the particular induction algorithm.

Hypothesis outranking relation may be reasonably defined in several different ways. In our previous studies on this topic, we based outranking definitions on:

? crisp set inclusion [17],
? inclusion grade measuring the extent of inclusion hypothesis coverings [19].

[2] Formally, an outranking relation induces partial *pre*order, as it permits indiscernibility.

The outranking definitions were designed as to take into account the class distributions in the coverings *C*(*a*) and *C*(*b*). For a more detailed discussion of the topic, see [19].

5 Extending Genetic Programming by Local Search

The evolutionary computation carries out random parallel search and therefore does not require any neighborhood definition in the solution space. As a result, solution having high value of evaluation function may be overlooked even being very similar to the solution visited in the search. Starting from the late 80's, several attempts have been made to prevent that negative phenomenon. One of the considered remedies was to combine the genetic search with other, more systematic, metaheuristics. In particular, hybridizing GA with various types of local search was reported to improve significantly the effectiveness of the search in combinatorial optimization [1]. Such approaches, known in the literature as Genetic Local Search (GLS), Memetic Algorithms or Hybrid Genetic Algorithms are now intensively studied in single- and multi-objective metaheuristic combinatorial optimization (for review, see [10]).

Our research reported in [18] follows this direction in the context of GP and focuses on intertwining the actions of genetic operators with iterative improvement of solutions. Our expectation is that such a local improvement can make the search more biased toward good (optimal or suboptimal) solutions. The particular implementation of this idea may be different as far as the following features are concerned: neighborhood definition, local search algorithm, and the extent of the local optimization.

Neighborhood definition. A natural way is to base the neighborhood definition on the representation of solutions in the GPVIS language. As the neighbor solutions should be similar, we generate neighbors of a given solution by introducing minor changes in leaves of its expression tree, which, according to GPVIS syntax, contain exclusively numerical values. In particular, we obtain a single neighbor by increasing (or decreasing) the value at a selected leave of current solution by a small constant.

Local search algorithm. In experiments reported in the Section 7 we used the *hill-climbing* algorithm for the local improvement of solutions. This metaheuristics evaluates all the solutions in the neighborhood and moves to the best one, provided it is better than the current solution. Otherwise no move is performed and the algorithm gets stuck in (usually local) optimum.

The extent of the local optimization. The most appealing setting would be to apply an extensive local improvement to all individuals in the population. However, that would increase the computation time several times. Thus, we control the local search by means of two parameters:

? the percentage of best (fittest) individuals that undergo the local improvement,
? the limit of hill-climbing steps.

In particular, in this study we chose the minimal option, improving locally only *one best solution* in each generation and allowing only *one step* of the hill-climbing algorithm. Seemingly, such a setting limits seriously the local search extent, as it stops prematurely the hill-climbing for solutions, which could continue with improvement. In fact, many solutions that *could* be further optimized actually *will* continue with local search in next generations of evolution, as only a part of the population is being modified in the recombination process.

6 Other extensions of GP paradigm

In this section we briefly describe some other major extensions of the standard genetic programming paradigm that have been introduced in the GPVIS system. A more detailed description of these changes improvements may be found in [20].

Windowing. The GPVIS environment implements the learning from examples paradigm of machine learning. An evaluation of individual consists in its execution on the set of training examples. In real-world machine learning tasks, a great part of the training set represents the ‘easy’ part of the problem (‘core’), whereas the remaining ones constitute the ‘hard’ part of the problem (‘exceptions’). The performance of learning algorithm depends mainly on its ability to deal with that latter part of the training set.

In iterative learning (e.g. evolutionary learning) the training algorithm reaches very fast the stage, when most (or even all) of the training examples coming from the ‘core’ of the problem are correctly classified. Thus, the evaluation of considered hypotheses on such examples is rather of secondary importance, and, from the viewpoint of time complexity, it is rather waste of time.

The *windowing* technique, proposed initially by Quinlan in his well-known C4.5 decision tree induction algorithm [27], is an answer to the problems discussed above. It focuses the attention of the learner on the exceptions by incrementally increasing their share in the training set. The training algorithm starts the learning with an initial working set of examples (the ‘window’) being a small sample drawn randomly from the original training set. After inducing a hypothesis from the working set, the working set is extended by those examples from the remaining part of the training set, which are incorrectly classified by the that hypothesis. That process is repeated until all examples from outside window are correctly classified.

This procedure in a simplified form has been implemented in GPVIS. In particular, it is controlled by two parameters. The first of them is the fitness threshold. The working set of training examples is extended when the best individual in the population reaches that threshold. The second parameter is the window increment, i.e. the number of new examples that should be included into the training set when the windowing procedure is being launched.

Adaptive operators. Before explaining this particular extension in more detail, let us have a closer look on a part of the training process in evolutionary learning from examples. An individual’s evaluation yields a scalar, usually normalized value (unless the non-scalar evaluation described afore is used). That evaluation influences the chance for that individual to pass through the selection and to participate in the breeding process of the next generation, as well as the chance of being subject to random modifications during mutation process. However, all these potential changes are **not directly driven by the value of evaluation function**, and, in particular, do not depend on the response of the individual hypothesis to particular examples from the training set.

In other words, the inherent feature of evolutionary learning is that the translation of the training signal (performance on particular training cases or accuracy of classification) into hypothesis modification is very indirect and random. Thus, a chance that an individual will really undergo some changes, which would essentially **adapt** it to the fitness set environment is extremely low.

That observation is the main premise for constructing **hybrid evolutionary learning systems,** most of which are essentially feature selection tools. In that setting, the evolutionary learning searches the feature space and each individual corresponds to a set of features, which are used by machine learning algorithm. Then, the performance of that algorithm constitutes the value of evaluation function. Such an approach, referred to as *wrapper* in the literature, became very popular in the recent years [13] and proved its usefulness in many real-world case studies (see, e.g. [21]).

The extension implemented in GPVIS follows that idea, however on a much lower conceptual level. For the sake of simplicity and elegance, we try to avoid splitting the problem into two separate parts: evolutionary feature construction and conventional hypothesis induction. Instead of that, the adaptive learning is implemented as a part of genetic programming methodology. The idea is to **introduce into the hypothesis language new terms equipped with some elementary adaptation ability**. Such terms, called *adaptive elements* have been implemented in GPVIS. When evaluating an individual, the adaptive element collects the values returned by its child subexpressions for consecutive examples form the training set. Then, the means of returned values are computed separately for the positive and negative class. Finally, the threshold value is computed as the average of those two means. Comparison of the value obtained for particular example to that threshold yields the final (binary) outcome.

Such learning procedure is much more flexible and gives much more chance for picking up valuable and novel individuals. An advantage of this solution over the hybrid techniques described before is low computational cost, which may be practically neglected when compared to the actual cost of individual's evaluation.

Detection of useful subexpressions. There is nothing new in stating that individuals evolved in genetic programming are redundant, i.e. they contain a lot of stuff that is either useless (e.g. subexpressions which are 'ingored' by their parent expressions) or overcomplicated. However, already an early research in evolutionary computation recognized the positive influence of that redundancy. The 'dead code' fragments, so-called *introns*, protect the useful parts of an individual from being destroyed by genetic operators.

There is however another face of this issue. Usually, when evaluating an individual, we test exclusively the response of its root nonterminal. However, some of the individual's subexpressions may actually perform better than the entire individual's expression tree! In the standard evaluation mode this fact will be never discovered and the solution may be lost in the search process due to its moderate evaluation as a whole.

Therefore, the extension introduced in GPVIS consists in **testing (evaluating) all the nonterminals in the individual's expression** (except for those that return constant values for all examples from the training set) subject to being potential roots of the expression tree. When an expression providing better evaluation than the root is found, it replaces the entire individual. The primary advantage is here more chance for finding good solutions obtained at extremely low computational cost (the return values for subexpressions have to be computed for all training examples anyway).

7 The Computational Experiment

The primary goal of the computational experiment described in this section was to compare the search effectiveness of the 'plain' genetic programming (GP) and genetic programming extended by the local search (GPLS), as described in Section 5.

7.1 Off-Line Handwritten Character Data

As a test bed for the method, we chose the problem of off-line handwritten character recognition, often referred to in the literature due to the wide scope of real-world applications (see [23] for review of methods). The data source was the MNIST database of handwritten digits provided by LeCun et al. [23]. MNIST consists of two subsets, training and test, containing together images of 70,000 digits written by approx. 250 persons, each represented by a 28×28 halftone image (Fig. 2). Characters are centered and scaled with respect to their horizontal and vertical dimensions, however, not 'deskewed'.

7.2 Experiment Design and Presentation of Results

To ensure comparability of results, the runs of GP and GPLS algorithms were paired, i.e. they used the same initial population, training and test sets as well as the values of parameters: population size: 50; probability of mutation: .05; tournament selection scheme [7] with tournament size equal to 5. In each generation, half of the population was retained unchanged, whereas the other fifty percent underwent recombination.

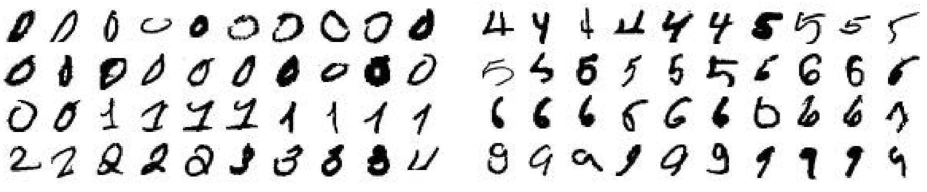


Fig. 2. Selected difficult examples from the MNIST database.

An evolutionary experiment started from a randomly generated initial population. Each solution was evaluated by computing its accuracy of classification on the training (fitness) set, containing 50 images from each of considered decision (digit) classes. In the recombination process, the offspring solutions were created by means of the crossover operator, which randomly selects subexpressions (corresponding to subtrees in Fig. 1) in the two parent solutions and exchanges them. Then, for a small (.05) fraction of the population, the mutation operator randomly selected an individual's subexpression and replaced it by an expression generated at random. The operators obeyed the so-called *strong typing* principle [14], i.e. they yielded correct expressions w.r.t. to GPVIS syntax.

As the evolution proceeds, the GP solutions tend to grow, because large expressions are more resistant to the performance deterioration, which often results from recombination. This phenomenon is conducive to inconvenient overfitting of hypotheses to the training data. To prevent that tendency, the fitness function was extended by an additional penalty term implementing the so-called *parsimony pressure*. Solutions exceeding 100 terms (nodes of expression tree) were linearly penalized with the evaluation decreasing to 0 when the threshold of 200 terms was reached.

To simplify the task, the problem was decomposed into 10×9/2=45 binary problems, each for a unique pair of decision classes. The evolution was carried out for each binary subproblem separately, based on the training set limited to the examples representing the two appropriate decision classes. The triangular matrix of 45 independently induced classifiers form the so-called *metaclassifier*, in particular the n^2 (or *pairwise coupling*) type of it [11]. The metaclassifier was then verified on an independent test set, containing 2000 cases, i.e. 200 images for each of 10 decision classes, acquired from a different group of people [23].

Table 1. Comparison of the pattern recognition programs evolved in GP and GPLS runs (accuracy of classification expressed in percents, superior results in bold).

Max # of generations	Training set				Test set		Classifier size (# of terms)	
	GPLS better	Average inc. of accuracy	Metaclassifier accuracy		Metaclassifier accuracy			
			GP	GPLS	GP	GPLS	GP	GPLS
20	34/45	3.64	57.4	**64.3**	52.0	**57.4**	2353	2203
40	33/45	3.58	60.6	**66.8**	55.1	**61.8**	2669	2719
60	31/45	3.02	62.4	**66.9**	58.3	**62.5**	2627	2927
80	25/45	2.16	64.7	**68.3**	62.5	**62.8**	2844	3139
100	24/45	1.47	67.7	**68.4**	**64.5**	62.6	2969	3131

Table 1 presents the comparison of the best solutions obtained in GP and GPLS runs. Each row summarizes the results of 45 pairs of genetic runs for binary subproblems. Each run consisted of GP and GPLS training starting from the same initial population. Consecutive rows describe experiments with different maximal number of generations. To provide better insight, apart from the complete 10-class classification problem, the 2nd and 3rd columns of the table contain also results for binary subproblems. In particular, the table includes:

– the maximal number of generations (iterations) for the search process (run length), as set for a single binary learning task (‘*Max # of generations*’),
– the number of pairs of GP and GPLS runs (per total of 45) for which the best solution evolved in GPLS yielded strictly better accuracy of classification on the training set than the best one obtained from ‘plain’ GP (‘*GPLS better*’),
– the average increase (GPLS minus GP) of accuracy of classification for binary subproblems, obtained on the training set (‘*Average inc. of acc.*’),
– the accuracy of classification of the compound n^2 metaclassifier on the training set and test set for GP and GPLS (‘*Metaclassifier accuracy*’),
– the size of the metaclassifier, measured as the number of terms of GPVIS expression (‘*Classifier size*’).

8 Conclusions and Future Research Directions

The main qualitative result obtained in the experiment is that evolutionary search involving local improvement of solutions (GPLS) outperforms on average the ‘plain’ genetic programming (GP).

As far as the binary classification problems are concerned (2nd and 3rd columns of Table 1), GPLS provided positive increment in terms of accuracy of classification on the training set, for all settings of maximal number of iterations. Each increase shown in 3rd column of the table is statistically significant at 0.05 level with respect to the Wilcoxon’s matched pairs signed rank test, computed for the results obtained by particular binary classifiers. These improvements seem to be impressive, bearing in mind the complexity of the visual learning task in the direct approach (see Section 3) and the fact, that the accuracy provided by both the methods for binary problems was at the end of runs close to 100%.

The results for the binary classification tasks influence those of the metaclassifiers (columns 4-7 of Table 1). For all run lengths set in the experiment, GPLS metaclassifiers are superior on the training set. On the test set, that superiority is also observable, but it significantly decreases with time and when the maximal number of iteration reaches 100, the metaclassifier evolved by ‘plain’ GP becomes better. This observation indicates that some of the GPLS solutions probably get stuck in the local optima of the fitness function due to the local optimization.

It should be stressed that these improvements have been obtained by means of very limited extent of local optimization (one step of local optimization applied to the best individual in each GP generation). Note also that the obtained GPLS solutions have similar size to those reached by GP (last two columns of Table 1).

Further work on this topic may concern different aspects of the approach. In particular, it seems to be interesting to consider other local search metaheuristics as tools for local improvement of solutions or to look for the compromise between the extent of the local optimization and the computation time. We plan to also work on other related research directions, with a special stress on detection of useful subexpressions (Section 6) and discovering structure of the task. The reader is invited to track further advances in this research on the Web: http://www-idss.cs.put.poznan.pl/gpvis/.

Acknowledgements

The author wishes to acknowledge financial support of the KBN research grant no. 8T11F 006 19 from the State Committee for Scientific Research, and of the subsidy no. 11/2001 from the Foundation for Polish Science.

References

1. Ackley, D.H., *A connectionist machine for genetic hillclimbing*. Kluwer Academic Press, Boston, (1987).
2. Bala, J.W., De Jong, K.A., Pachowicz, P.W. Multistrategy learning from engineering data by integrating inductive generalization and genetic algorithms. In R.S. Michalski, G. Tecuci, *Machine learning. A multistrategy approach. Volume IV*. San Francisco: Morgan Kaufmann, (1994) 471-487.
3. Chan, P.K., Stolfo, S.J. Experiments on multistrategy learning by meta-learning. *Proceedings of the Second International Conference on Information and Knowledge Management* (1993).
4. De Jong, K.A. An analysis of the behavior of a class of genetic adaptive systems. Doctoral dissertation, University of Michigan, Ann Arbor (1975).
5. De Jong, K.A., Spears, W.M., Gordon, D.F. Using genetic algorithms for concept learning. *Machine Learning*, *13* (1993) 161-188.
6. Goldberg, D. *Genetic algorithms in search, optimization and machine learning*. Reading: Addison-Wesley (1989).
7. Goldberg, D.E., Deb, K., Korb, B. Do not worry, be messy. *Proceedings of the Fourth International Conference on Genetic Algorithms*. San Mateo: Morgan Kaufmann (1991) 24-30.
8. Gonzalez, R.C., Woods, R.E. *Digital image processing*. Reading: Addison-Wesley, 1992.
9. Holland, J.H. *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press (1975).
10. Jaszkiewicz, A. *On the performance of multiple objective genetic local search on the 0/1 knapsack problem*. A comparative experiment. Research report, Institute of Computing Science, Poznan University of Technology, RA-002 (2000).
11. Jelonek, J., Stefanowski, J. Experiments on solving multiclass learning problems by n^2-classifier. In C. Nedellec, C. Rouveirol (eds.) *Lecture Notes in Artificial Intelligence 1398,* Berlin: Springer Verlag (1998) 172-177.
12. Johnson, M.P. *Evolving visual routines*. Master’s Thesis, Massachusetts Institute of Technology, 1995.

13. Kohavi, R., John, G.H. Wrappers for feature subset selection. *Artificial Intelligence Journal*, 1--2, 1997, pp. 273-324.
14. Koza, J.R. *Genetic programming - 2*. Cambridge: MIT Press (1994).
15. Koza, J.R., Keane, M., Yu, J., Forrest, H.B., Mydlowiec, W. Automatic Creation of Human-Competetive Programs and Controllers by Means of Genetic Programming. *Genetic Programming and Evolvable Machines* 1 (2000) 121-164.
16. Krawiec, K. *Constructive induction in picture-based decision support*. Doctoral dissertation, Institute of Computing Science, Poznan University of Technology , Poznan (2000).
17. Krawiec, K., Pairwise Comparison of Hypotheses in Evolutionary Learning, In: Brodley, C.E., Pohoreckyj Danyluk, A. (eds.) Machine Learning. Proceedings of the Eighteenth International Conference (ICML 2001), Morgan Kaufmann Publishers, San Francisco 2001, pp. 266-273.
18. Krawiec, K., Genetic Programming with Local Improvement for Visual Learning from Examples.In: Skarbek, W. (ed.) Computer Analysis of Images and Patterns. Lecture Notes in Computer Science (LNCS) 2124, Springer Verlag, Berlin 2001, pp. 209-216.
19. Krawiec, K., On the Use of Pairwise Comparison of Hypotheses in Evolutionary Learning Applied to Learning from Visual Examples, In: Perner, P. (ed.) Machine Learning and Data Mining in Pattern Recognition, Lecture Notes in Artificial Intelligence (LNAI) 2123, Springer Verlag, Berlin 2001, pp. 307-321.
20. Krawiec, K., Pairwise Comparison of Hypotheses Coverings as a Natural Mean Agains Undesirable Niching in Evolutionary Inductive Learning . Research Report RA-005/2001, Institute of Computing Science, Poznan University of Technology, Poznan 2001.
21. Komosinski, M., Krawiec, K. Evolutionary weighting of image features for diagnosing of CNS tumors. Artificial Intelligence in Medicine, (19)1, 2000, pp. 25-38.
22. Langley, P. *Elements of machine learning*. San Francisco: Morgan Kaufmann (1996).
23. LeCun, Y., Jackel, L. D., Bottou, L., Brunot, A., et al. Comparison of learning algorithms for handwritten digit recognition. *International Conference on Artificial Neural Networks* (1995) 53-60.
24. Mitchell, T.M. *An introduction to genetic algorithms*. Cambridge, MA: MIT Press (1996).
25. Mitchell, T.M. *Machine learning*. New York: McGraw-Hill (1997).
26. Poli, R. *Genetic programming for image analysis*, (Technical Report CSRP-96-1). The University of Birmingham (1996).
27. Quinlan, J.R. *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers, Inc., San Mateo, 1992.
28. Teller, A., Veloso, M. A controlled experiment: evolution for learning difficult image classification. *Lecture Notes in Computer Science, Vol. 990* (1995) 165-185.
29. Vafaie, H., Imam, I.F. Feature selection methods: genetic algorithms vs. greedy-like search. *Proceedings of International Conference on Fuzzy and Intelligent Control Systems* (1994).
30. Vincke, P.: Multicriteria decision-aid. John Wiley & Sons, New York (1992)
31. Yang, J., Honavar, V. Feature subset selection using a genetic algorithm. In H. Motoda, H. Liu (Eds.), *Feature extraction, construction, and subset selection: A data mining perspective*. New York: Kluwer Academic (1998).