Knowledge-Based Genetic Learning

Ursula Rost, Peter Oechtering
Institut für Theoretische Nachrichtentechnik, Universität Hannover
Appelstr. 9a, 30167 Hannover

Abstract. Genetic algorithms have been proven to be a powerful tool within the area of machine learning. However, there are some classes of problems where they seem to be scarcely applicable, e.g. when the solution to a given problem consists of several parts that influence each other. In that case the classic genetic operators cross-over and mutation do not work very well thus preventing a good performance. This paper describes an approach to overcome this problem by using high-level genetic operators and integrating task specific but domain independent knowledge to guide the use of these operators. The advantages of this approach are shown for learning a rule base to adapt the parameters of an image processing operator path within the SOLUTION system.

1 Introduction

Genetic algorithms are a class of adaptive search techniques which use evolutionary principles like inheritance and natural selection. They have been applied successfully to many problems in different areas including machine learning (see for example [1]). However, there are some classes of problems where the standard genetic algorithm which uses bit patterns to represent the knowledge and crossover and mutation to work on it is scarcely applicable. This is especially the case within the field of learning complex behaviour containing multiple dependencies.

The main problems occurring here consist of finding an adequate representation formalism to encode the knowledge to be learned and, in addition to this, genetic operators have to be defined which apply representation-sensitive changes to the knowledge. As problem-specific knowledge has to be incorporated into this process it is sometimes called hybridization [2]. Much research has been dedicated to these problems and many strategies have been proposed. An overview is given for example in [3] and a large set of applications using different kinds of data structures and genetic operators can be found in [4]. While the operators described there are mainly variations of the classic mutation and crossover operators, Janikow suggests in [5] high-level genetic operators as for example specialization or generalization. The genetic programming approach introduced by Koza uses hierarchical structured programs to encode the knowledge thus enabling the individuals of a population to have arbitrary size and shape [6]. A different idea has been developed by Potter et al in [7]. There the population is broken up into several species which are designed to work on different tasks. Each of theses species is treated independently during the genetic phase. For evaluation all subpopulations are combined. As competition is carried out only within each group and the groups collaborate to form a solution this is called a coevolutionary approach.

Something all these approaches share is the stochastic way in which the genetic operators are applied. They only vary to which extent any of the operators is used and to which portion of the population. While much work has been directed towards the investigation of the recombination and mutation parameters (see for example [8]) the stochastic principle used remains the same.

In contrast the approach described in this paper uses a completely different strategy. The selection of the genetic operators is guided by control rules which incorporate feedback received from the environment which is gained during the evaluation process. These control rules evoke high-level genetic operators in a goal directed manner. The task-specific knowledge used is explicit and restricted to the control rules, so that the algorithm can be easily adapted to work on similar problems.

Section 2 will explain the environment the genetic learner presented here was developed for and the main problems that have to be kept in mind when learning in this environment. In section 3 the genetic learning algorithm is described with emphasis on the design principles used. Some results achieved in the chosen environment are presented in section 4 and shortly discussed in section 5.

2 The SOLUTION System

In the past 10–15 years image processing systems have been deployed to a broad range of industrial applications e.g. in the areas of quality surveillance and process control. One major problem however still remaining lies within a lacking flexibility that many of those systems own. The image processing operators they contain are mostly designed and tuned especially to fulfill only one certain task. As changes occur to the environment as for example in the scene illumination or to the task specification, the system fails to produce the expected results. In this case an image processing expert has to analyze the problem and to adapt the parameters of the system to manage the new situation. This is a time-consuming and cost-intensive process that often leads to a long standstill of the production. This is something no company can afford nowadays so that many production managers are extremely hesitating concerning the further deployment of image processing systems.

The basic idea behind SOLUTION which stands for Solution for a Learning Configuration System for Image Processing (and formerly Conny [9]) is to provide a tool to configure the low-level part of an image processing system in a way that always yields results with a specified quality so that subsequent high-level routines work more robustly. Low-level processing thereby means an image to image transformation.

SOLUTION is able to do the following: First of all it lets the user specify an image processing task in high-level domain specific terms, which can be transformed into a list of properties the result image should have. Depending on this task specification it chooses some appropriate image processing operators out of a given set and configures an image processing path out of them, that means a chain of operators which are processed consecutively. Finally it adapts the parameters of the operators in the path so that the result image generated when processing the path has the desired properties.

As the system should be highly adaptable to varying tasks and to extensions with further image processing operators a knowledge-based approach was chosen for SOLUTION. The knowledge used is encoded as forward-chaining rules.

Those who have ever build up an expert system know that it is a complex and time consuming task to create a consistent and complete knowledge base. For this reason SOLUTION was extended to contain some learning units to support the knowledge acquisition process. Up to now there are two learning components: a module that

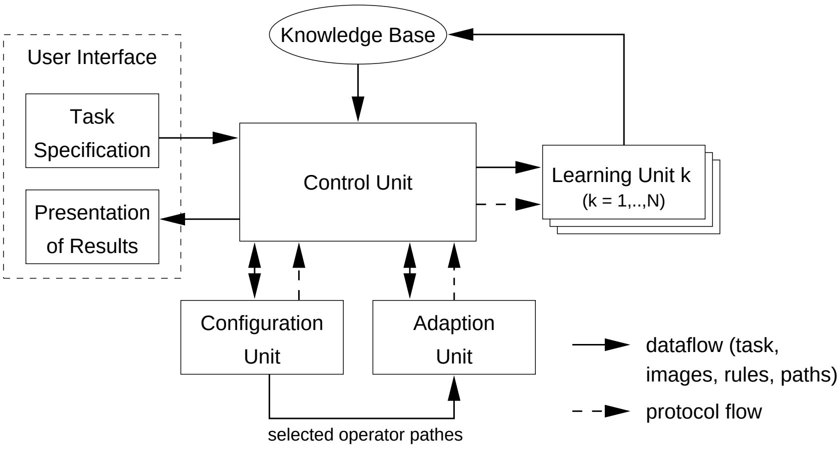

Figure 1: Structure of SOLUTION

learns by experimentation and generalizes the knowledge by a simple inductive inference mechanism and the knowledge-based genetic algorithm described in this paper.

Figure 1 shows the general structure of SOLUTION. The control unit in the center is responsible for the coordination of all system activities. It provides all other modules with the information they need and controls the data and protocol flow.

The task the user has to specify consists of three major parts. First of all there are some general requirements as for example a short processing time or the accuracy of edge positions needed. The user can set a priority value for these requirements by telling whether it is "unnecessary", "unimportant", "important" or "necessary" to fulfill these requirements.

Secondly, as the selection of appropriate image operators depends, among other things, on some properties of the input image such as noise and disturbances, the user can either specify a value or can interactively define a ROI (Region of Interest) in which the property is automatically measured and quantized to one of the values "not", "weak", "average" and "high".

The third part of the task consists of the specification of several properties which the result image is expected to possess. These properties are collected and transformed to a so-called global goal. This is the most crucial part because at this point domain specific information has to be included. Depending on the contents of the image to be processed some of the properties play an extremely important role while others don't care at all. Table 1 shows an example referring to the general task of creating a "good" contour image. The three selected properties specified in the example define what "good" means in a chosen context, e.g. that it is important to achieve that the thickness of the lines in the result image is at most "weak".

After the task has been defined the configuration process is started. It consists mainly of a rule inference process that checks which operators can be used to build an image processing path that fulfills the given constraints. The rules used here are built up by a precondition referring to the first two parts of the task specification and an action that eliminates one image processing operator from the list of defined operators. Examples of such rules are:

$<$if a high exactness of the vertex positions is necessary$>$
$\Rightarrow$ $<$don't use a Pratt operator$>$
$<$if there are no disturbances in the image$>$
$\Rightarrow$ $<$don't use a Median operator$>$

Table 1: Example of a global goal

property	value to reach	relation	importance
thickness of lines	weak	at most	important
closeness of lines	strong	identical	necessary
parallelism of lines	not	identical	important

The remaining operators are used to create a set of image processing paths which are considered to be possible solutions. The parameters of the operators contained in these paths are given initial values and are then transferred to the adaptation unit.

In contrast to the configuration process the adaptation is an iterative process. The strategy used here is a stepwise improvement of the operator's parameter values with respect to the desired properties of the result image. In each step the actual path is processed and the properties of the output image are computed by special routines called sensors. The values are then compared to the values defined in the global goal. From every difference that occurs a so-called local goal is made up. If for example the global goal contains the condition that the "thickness of the lines should be at most weak" and the thickness of the lines is actually "strong" a local goal is created that states that the "thickness of the lines should be decreased strong". The computation of the local goals is followed by a rule inference process in which each rule consists of one or more local goals that are to be reached in the condition part and one parameter change in the action part. The following example shows a rule with two local goals.

<if the thickness of the lines should be decreased strong> and
<if the closeness of the lines should be increased weak>
⇒ <increase the threshold parameter of the Binarize operator strong>

The adaptation process is hence a goal-directed process which tries to minimize the difference between the desired properties defined in the global goal and the measured properties of the result image of a given image processing path.

3 A Knowledge-Based Genetic Algorithm

The configuration process can be regarded as a single-level process because each of the rules eliminates exactly one operator which will not be reconsidered. In contrast to this, the adaptation causes a chain of rule firings. Each rule changes a parameter value and thus creates a new situation concerning the properties of the result image. Replacing one rule in the chain by another rule can lead to a completely new situation at that point of the adaptation process and thus to strong differences within the rest of the chain. The rule base needed for the adaptation therefore has to cover a much larger set of situations than the rule base used for the configuration. As the rules mutually influence each other the task of building up a consistent and complete rule base is very complex and should be supported by a learning algorithm.

The applicability of genetic algorithms to this kind of problem in principle has been shown in earlier studies. The genetic algorithm described in [10] already uses a coding scheme reflecting the structure of the rules. An individual is thereby represented by a single rule. The genetic operators crossover and mutation are adapted to work on this structure but their application is directed by the same stochastic means as in the classic approach, neglecting the mutual influence the rules have on each other. It was shown that a rule base can be learned successfully in this way but with a very poor performance. The algorithm needs up to 20 000 generations until convergence. This behaviour mainly results from the way the genetic operators are applied. Therefore it seems quite natural to incorporate some general task-specific knowledge to reduce the degrees of freedom of the algorithm and thus to speed up convergence. On the one side the genetic algorithm developed in consequence uses several methodologies well known in the genetic community such as a non-randomly chosen initial population and niching while the latter is implemented by a restricted mating of rules. In addition to this most of the genetic operators represent logic or heuristic operators which are applied in reaction to the different situations which occurred during the preceding adaptation. Such kind of relationship of a learning operator to its environment is sometimes called Lamarckian. In [11] Grefenstette describes Lamarckian Learning in the domain of sequential decision making in an multi-agent environment. In contrast to his approach the selection of all genetic operators presented here is guided by some explicit formulated control rules which include knowledge about the errors made during the adaptation process and how to improve it. The learning process is performed as an inference process and thus the genetic algorithm can be considered to be knowledge-based. In the remainder of this section the algorithm will be explained in more detail.

3.1 Representation

The rules that are to be learned can contain a varying number of local goals. Moreover each local goal and also the action consist of several parts which have to be treated by the genetic operators in different ways. Therefore the standard fixed-length binary pattern representation is not adequate here and a symbolic model is used instead which includes a separate condition and action part. In addition to this each rule can be identified by an id and has a priority value P which represents the strength of a rule as needed within the genetic algorithm. Simultaneously P is used to establish preferences among rules of the same specificity when choosing the best rule out of a conflict set. The general structure of a rule can be represented as follows:

$R_{id,P}$: <if local-goal-1> and <if local-goal-2> and ... and <if local-goal-n>
$\Rightarrow$ <increase/decrease parameter-i of operator-j strong/weak>

Each local goal thereby consists of a list of three elements containing the property the local goal refers to, the direction in which a change is desired and the strength of this change.

3.2 Initial Population

A careful choice of the initial population of a genetic algorithm can improve its efficiency in a substantial way as various studies have shown (e.g. [12, 13]). Although the algorithm described here is able to start from scratch, i.e. with an empty rule base, it turns out to reach a better performance when given an initial rule base that includes at least some useful rules. Such a rule base can be generated by a simple experimentation process: The effect a rule should take consists of a property change of the result image. This should be reached by a parameter change in one of the image processing operators. As all properties can be measured by sensors it is possible to go the other way around to create some rules with a high reliability. This is done by taking an operator path with arbitrarily chosen parameter values and changing each of the values separately in different directions. The changes that are thereby caused in the result image of the path are collected and used to generate simple rules containing one condition generated from the property change and an action that contains the parameter change. For example the experiment might consist of

A_1: <increase the threshold of the Binarize operator strong>

That would perhaps cause the following two property changes:

C_1: <the thickness of the lines decreased weak> and

C_2: <the closeness of the lines increased weak>

As consequence the following two rules would be created:

R_1: <if the thickness of the lines should be decreased weak>
$\Rightarrow$ <increase the threshold parameter of the Binarize operator strong>

R_2: <if the closeness of the lines should be increased weak>
$\Rightarrow$ <increase the threshold parameter of the Binarize operator strong>

The restriction to rules which contain only one local goal is performed because a parameter change can cause up to 15 property changes which is the overall number of sensors used. Including them all in one rule would lead to overspecialization. Using only one local goal in each condition part ensures that there is nearly always one applicable rule when trying to reach any global goal during the adaptation. Clearly the rule base created in this way doesn't ensure a successful adaptation and it also covers irrelevant rules but the important thing is that it represents a good starting point for the genetic algorithm.

3.3 Genetic Operators

The genetic operators described here are developed to be used in an environment where the effects of a firing rule can be judged at least partially so that logic or heuristic implications can be made about a reasonable change of the rules. The main portion of the operators created perform this kind of changes. In addition to this some genetic operators are used that preserve the "genetic part" by making guesses about new possibly interesting areas of the search space. The principles they work on are described below. The first three operators belong to the group of operators that perform goal-directed changes to the rule base whereas the latter three are responsible for the exploration of new areas.

Invert: This operator is a special mutation operator that works on two-valued ranges, as for example represented by the direction ("up", "down") or strength ("weak", "strong") part of the local goals and the action. It replaces one value by the other.

Generalize: This operator changes the condition part of a rule by removing one local goal from it and thus makes it available to fire in more situations.

Specialize: In contrast to the previous operator this one adds a local goal to the condition part of a rule so that it covers less situations.

Explore: This operator takes up to three goals that appeared in a single adaptation step and combines them in a new rule together with a randomly chosen action. By this means search is directed into new areas.

Join: This operator is the only one including mating. It takes two rules that fired in a single adaptation step, combines their condition parts and chooses an action part from one of the rules.

Mutate: The mutation operator replaces the action of a rule by another, randomly chosen one.

In fact no operator really changes an existing rule or deletes it from the knowledge base. It always creates a copy to work on. To ensure that the new rule will fire in the next adaptation cycle instead of the old rule, the rule is made inactive by setting a flag. This strategy allows to observe the impact of the control rules on the learning process and thus provides a tool to judge and adapt the learning process itself.

3.4 Credit Assignment and Selection

Due to the multiple dependencies between the rules in the rule base there is no easy way to assign a fitness value to each individual rule. To take this problem into account three partially competing criteria are used.

First of all each rule owns a reliability value that is composed of the average impact the rule has had in the past. Every time a rule fires, the resulting changes of the properties covered by the local goals are compared to what has been claimed in the conditions. Each correspondence increases the reliability value, each difference decreases it according to its strength. In addition to this the group of rules that fired within a single adaptation step are given a common quality-improvement value that depends on the way the quality of the operator path was changed in this step, i.e. how much the distance to the global goal was shortened (or lengthened). Any improvement of the path with respect to the global goal will result in a higher value, any deterioration in a lower one. Finally the overall adaptation-quality, that means the minimum distance to the global goal reached in the whole adaptation process is judged and spread evenly among all rules which have fired during the adaptation.

As the development of the criteria viewed over several generations is not necessarily the same for a rule, Pareto optimality is used to provide a single numeric quality value for every rule. In this connection a maximum value means that a rule is non-dominated by any other rule in the sense that no other rule has a higher value in all three criteria. A more detailed description of the methodology of Pareto optimality can be found for example in [1]. The Pareto rank of each rule is converted into its priority value and thus influences whether it is selected during the conflict resolution process of the rule inference.

The selection process is completely driven by the control rules described in the next section. That means it only depends on the experiences made during the adaptation.

3.5 Learning Control

As mentioned above the genetic operators described here are designed to apply changes to the rule base mainly in a logic or heuristic manner and not stochastically as in the classic approach. Therefore the adaptation process is observed and a protocol is generated in which information about the progress made is gathered. Among other things the information contains the impact of the rules which fired, the local goals which were created and which occured in the firing rules and which not, the final quality of the adaptation process and the reason of failure if the adaptation process wasn't successful.

The information is evaluated and used by a couple of control rules which incorporate the results of the evaluation in their condition part and invoke an appropriate genetic operator within their action part. The rule inference process is continued until no more control rules can fire.

The following descriptions of some selected control rules, grouped by their aim, should give an impression on how the genetic learning process is controlled.

Improve Reliability I: If there is a rule which only contains one local goal in the condition part and the achieved direction of change was inverse to that what was claimed, the *Invert* genetic operator is called accordingly. If the direction was correct but the strength of change was too harsh or too weak the Invert operator is applied to the strength part of the action. In both cases the rule must have fired more than once to cover fluctuations in the development of the property values.

Improve Reliability II: If a rule contains several local goals that were fulfilled up to different degrees the goal that achieved the worst result is deleted from the rule by the *Generalize* operator. On the one hand this leads to a more general rule that can fire in more situations on the other hand the reliability of this rule should be greater and the missing local goal is free to be fulfilled by applying another rule.

Specify Exception: The parameter values that are changed by the action parts of the rules all belong to ordered numeric ranges that are limited on both sides. The window size of a low-pass operator for example can not be made less than 3×3 and it makes no sense to enlarge it further than to a given portion of the image size. It may occur that a local goal can only be nearly fulfilled by a certain parameter change but not totally. This leads sometimes to the situation that a rule fires several times with a high reliability but at a certain point in the adaptation process the parameter change evoked by the action part is in fact not performed. In this case the *Specialize* operator is called to create a copy of the rule, to add any local goal not used in one of the steps and to replace the action by a randomly chosen new action. Hereby an exception rule is formulated which will fire in the next generation with high probability as more specialized rules are always preferred in the rule inference. If no unused local goal exists, the *Mutate* operator is applied and the new rule gets a higher priority value to ensure that it will fire in the next generation.

Follow New Directions: Sometimes the adaptation is stopped because there is no rule that matches the given situation. This is particularly the case when the learning process is started with an empty rule base. It may also occur that some of the local goals genereated in one adaptation step were not covered by any rule. Then the *Explore* operator is called to create new rules that avoid the failure.

Avoid Dead-End: One reason of failure is caused by a loop within the development of the parameter values. It may occur that the effect of one rule neutralizes the effect of another rule which fires in the same adaptation step so that in fact no parameter change is made. In this case the *Join* operator is used to replace those two rules by a new rule which includes both condition parts. The action part is chosen from the rule owning the better reliability value.

Avoid Loop: A loop can also span several steps and different rules. One way to break it is achieved by letting the *Mutation* operator change the action part of that rule involved in the loop that got the worst priority value.

Table 2: Example of the application of a genetic operator called by a control rule

adaptation rule R (generation i)	<if the closeness of the lines should be increased weak> ⇒ <increase the threshold parameter of the Binarize operator strong>
quality change evoked by rule R	-0.4
reason of failure of the adaptation after generation i	loop encountered over several steps (parameter values in last step are the same as 5 steps before)
control rule	<if reason of failure was "loop encountered over several steps"> and <if in each adaptation step only one rule fired> and <of all rules that have fired rule X caused the worst change of quality> ⇒ <apply the genetic operator Mutate to rule X>
genetic operator	*Mutation*: replace the action of a rule by a different one, which is chosen randomly out of the set of possible actions
adaptation rule R' (generation $i+1$)	<if the closeness of the lines should be increased weak> ⇒ <decrease the threshold parameter of the Binarize operator weak>

Table 2 shows an example of how a new rule is generated out of an existing one by applying the genetic operator *Mutation*. It is assumed that the adaptation that used the rule base generated in generation i failed because a loop in the parameter values encountered. It is further assumed that only one rule fired in each step of the adaptation and the rule presented in the first row lead to the worst quality change when compared to the quality changes evoked by the other rules which fired. In this case the control rule *Avoid Loop*, shown in the third row, takes this rule and applies the *Mutation* operator to it. The resulting rule is added to the population of the generation $i + 1$.

4 Results

To test the knowledge-based genetic algorithm under realistic circumstances a task was specified according to the needs of a real world problem. This problem consists of extracting the contours in an x-ray image that shows the projection of a human knee in which a knee joint endoprosthesis was implied. The algorithm used to compute the spatial position of the implant [14] needs for example highly accurate contour points to produce good results. Furthermore, the contours in the result image should not be frayed out and a certain portion of small artefacts should not be exceeded. Referring to this and other demands a global goal was created that includes 12 specifications referring to different properties a result image should possess. Concerning the mentioned properties of the result image one part of this goal demands for example that the value of the property frayness should be equal to "not" and another part says that the value of the property granulation should be less or equal to "average". In addition to this an image processing path which contains two free parameters was selected manually. Computing the total fitness landscape it could be verified that only two valuable solutions existed.

The learning process was performed by taking three different parameter combinations and two different images to create six learning paths used as starting points for the adaptation process. These paths were used in a cyclic way by the genetic algorithm. Every time the rule base learned in a certain generation lead to a successful adaptation the next learning path was taken. This process continued until the rule base enabled the successful adaptation of all learning paths. As the paths were initialized in

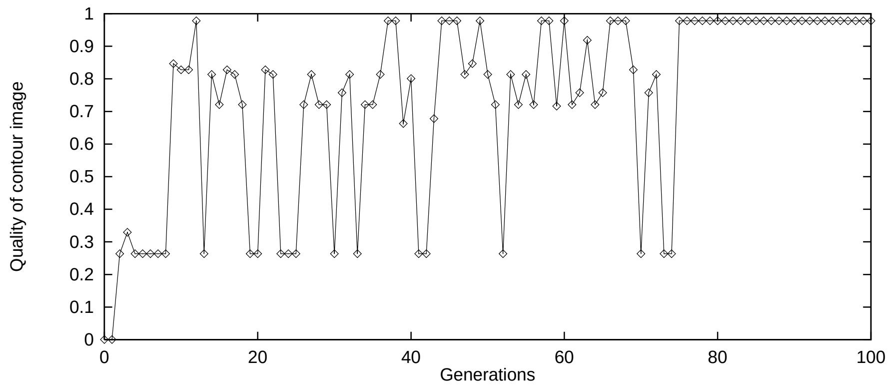

Figure 2: Learning curve starting with empty rule base

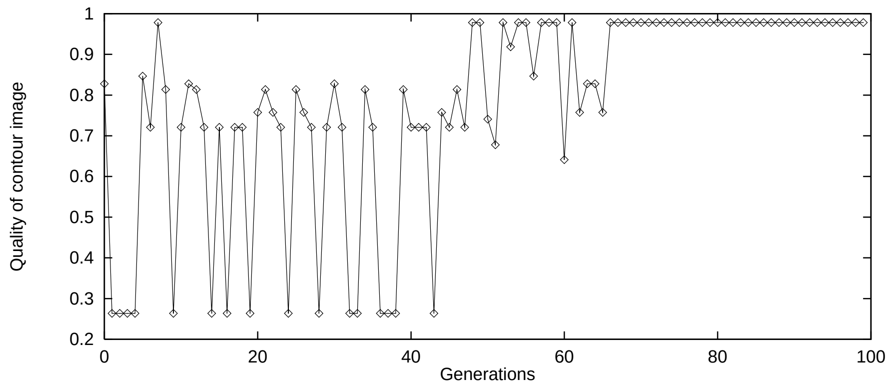

Figure 3: Learning curve starting with rule base generated by experimentation

quite different ways this strategy should provide that the learning algorithm traversed a sufficiently large set of different situations.

To sum up, one can say that about 90 percent of all learning processes converged within 200 generations. Starting with an empty rule base 28 – 170 generations were needed, using an initial population of rules created by experimentation resulted in 48 – 78 generations showing thus a more stable behaviour. This is clearly caused by the smaller number of stochastically generated rules starting instead with simple but more reliable ones.

Figure 2 shows a typical result achieved when starting with an empty rule base while figure 3 shows a typical result when using an initial rule base created by experimentation. Each mark denotes the final quality value reached when performing an adaptation process with the rule base created in the actual generation. As a total conformance with the global goal (value 1.0) could not be reached with the selected path a value of 0.93 was considered to be sufficient. The interpretation of this value means that at most one weak difference is accepted. The slopes that can be seen after each peak are due to the change of the learning path at this point. The oscillating behaviour of the curve reveals that some of the genetic operators apply crucial changes to the rule base. These changes sometimes lead to completely different rule chains and therefore to a quite different result quality of the adaptation process.

The average number of rules added to the rule base was about 40 in both cases.

After the learning phase the result rule bases were used to adapt paths with different parameter initializations not appearing in any of the adaptation steps of the learning

Figure 4: Results of the adaptation process: the x-ray image to be processed (left), the result contour image at the beginning of the adaptation (middle), the result contour image after the adaptation with the learned rule base (right)

paths to test their universality. The results achieved show a high stability. More than 87 percent of the paths could be adapted successfully in the first case. Again the use of rule bases learned starting with a good initial population ended up in an even better result of 96 percent successful adaptations.

Figure 4 depicts a typical result of an adaptation process when using a rule base that has been learned by the knowledge-based method described here. On the left side the input x-ray image is presented. In the middle the resulting output image of the path can be seen before the parameters have been adapted. On the right side the final output image after 7 adaptation steps is shown. All but one of the desired properties defined in the global goal have been reached. The value of the frayness is in fact "not" and the value of the granulation is "average".

5 Discussion and Future Work

The knowledge-based genetic algorithm presented in this paper produces a performance increase up to the factor 100 when compared to a standard genetic algorithm applied to the same problem of learning a complex rule base for parameter adaptation (see [10]). This success is mainly due to the knowledge-based control of applying the high-level genetic operators used in this approach. This control strategy guides the learning process towards promising areas of the search space while at the same time evolutionary principles are still kept to prevent the algorithm from premature convergence that hill-climbing approaches suffer from.

However, there are still some open issues. As mentioned in the last section not all learning processes converge within 200 generations and not all adaptation processes end up successfully. The influence of some additional image properties not covered by the used sensors may be the reason for this. To make the system work more robustly an alternative strategy for the adaptation has to be provided in these situations. Therefore it is planned to include an optimal search technique into SOLUTION that is able to replace the inference process if necessary.

One major problem remaining concerns the varying influence of different parameters of the image processing operators on the sensor values. Changing the window size of the median operator has for example a much smaller impact than changing the threshold of the Binarize operator. Therefore it is often neglected by the learning process.

Taking a closer look at the behaviour of the sensor values whithin the function domain another important point can be revealed. Some of the sensors don't behave monotonically. When for example the threshold of the Binarize operator is increased the value of the frayness of the contours usually increases first up to a certain point and then decreases. This becomes quite clear when regarding the result images. At first there are many long contours in the image that break up into several pieces when the threshold is increased but finally disappear so that only the strongest contours remain. Therefore the rule structure will have to be extended to cover situation-dependent behaviour in order to be able to use local goals which refer to image properties which show this kind of behaviour.

Future work will be dedicated to solve these problems. In addition to this further genetic operators and control rules should be developed that include learning paradigms allowing to optimize an already working rule base such as represented by explanation based learning.

6 Acknowledgements

The project has been supported by a grant of the Deutsche Forschungsgemeinschaft.

References

[1] D.E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley, 1989.

[2] L. D. Davis (Ed.), "Handbook of Genetic Algorithms",Van Nostrand Reinhold, 1991.

[3] K. De Jong, "Learning with Genetic Algorithms", Machine Learning, Vol.3, pp. 121–138, 1988.

[4] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs", third, revised and extended edition, Springer, 1996.

[5] C.Z. Janikow, "Inductive Learning of Decision Rules from Attribute-Based Examples: A Knowledge-Intensive Genetic Algorithm Approach", TR91-030, July 1991.

[6] J.R. Koza, "Genetic Programming", MIT Press, 1992.

[7] M.A. Potter, K.A. De Jong, J.F. Grefenstette, "A Coevolutionary Approach to Learning Sequential Decision Rules", Proc. of the 6th International Conference on Genetic Algorithms, Morgan Kaufmann, pp. 366–372, July 1995.

[8] W.M. Spears, "Recombination parameters", Handbook of Evolutionary Computation, IOP Publishing Ltd. and Oxford University Press, 1995.

[9] C.-E. Liedtke, A. Blömer, T. Gahm, "Knowledge-Based Configuration of Image Segmentation Processes", International Journal of Imaging Systems and Technology, Vol.2, pp. 285–295, 1990.

[10] C.-E. Liedtke, T. Schnier, A. Blömer, "Automated Learning of Rules Using Genetic Operators", Proceedings of the 5th International Conference on Computer Analysis of Images and Patterns, pp. 327–334, Budapest, Sept. 1993.

[11] J.J. Grefenstette, "Lamarckian learning in multi-agent environments", Proc. of the 4th International Conference on Genetic Algorithms, Morgan Kaufmann, pp. 303–310, 1991.

[12] C.L. Ramsey, J.F. Grefenstette, "Case-Based Initialization of Genetic Algorithms", Proc. of the 5th Int. Conference on Genetic Algorithms, Morgan Kaufmann, pp. 84–91, 1993.

[13] Lingyan Shu, Jonathan Schaeffer, "Improving the Performance of Genetic Algorithm Learning by Choosing a Good Initial Population", Technical Report TR 90-22, Department of Computing Science, University of Alberta, Canada, July 1990.

[14] U. Rost, C. Lehr, J. Wingbermühle, C. Kramer, M. Fleute: "Lageerkennung in der Knieendoprothetik", "Der Orthopäde", pp. 237–245, Springer-Verlag, 3/97.