Decreasing the Number of Evaluations in Evolutionary Algorithms by Using a Meta-model of the Fitness Function

Jens Ziegler and Wolfgang Banzhaf

University of Dortmund, Department of Computer Science
D-44221 Dortmund, Germany
{Jens.Ziegler,Wolfgang.Banzhaf}@uni-dortmund.de
http://www.cs.uni-dortmund.de

Abstract. In this paper a method is presented that decreases the necessary number of evaluations in Evolutionary Algorithms. A classifier with confidence information is evolved to replace time consuming evaluations during tournament selection. Experimental analysis of a mathematical example and the application of the method to the problem of evolving walking patterns for quadruped robots show the potential of the presented approach.

1 Introduction

The high number of fitness evaluations in evolutionary algorithms is often expensive, time-consuming or otherwise problematic in many real-world applications. Especially in the following cases, a computationally efficient approximation of the original fitness function reducing either the number or duration of fitness evaluations is necessary: (i) if the evaluation of the fitness function is computationally expensive, (ii) if no mathematical fitness function can be defined, (iii) if additional physical devices must be used. Several approaches have been suggested to reduce the number of fitness evaluations in EA. If the fitness function is a mathematical function, approximations by interpolation between individuals in search space build a meta-model of the fitness function (see e.g. [2, 8]). In more complex cases, the fitness of an individual may be inherited from their ancestors to save evaluations (see e.g. [11]). Another approach tries to minimize the number of evaluations by clustering the individuals around "representatives" which determine the fitness of a subset of the population [9]. Statistical and information theoretical results are used in e.g. [3] to reduce the number of fitness evaluations in GP. An comprehensive collection of works in this field can be found in [7].

The article is organized as follows. The next section introduces the idea of using the result of a classifier to discriminate between better and worse individuals during tournament selection. Section 3 introduces the confidence level for classifications. Section 4 investigates the influence of the meta-model on two different problems. Section 5 describes the evolution of a classifier with GP based on data from a real world experiment and section 6 presents and discusses the results of the evolution of gait patterns for four-legged robots using the meta-model approach. Finally, section 7 gives our conclusion and hints to future works.

C. Ryan et al. (Eds.): EuroGP 2003, LNCS 2610, pp. 264–275, 2003.

2 Tournament Selection by Classification

In this section, a new method will be illustrated which replaces time consuming fitness evaluations of individuals in tournaments by classifications during evolution.

During a tournament, T individuals are compared pairwise on the basis of their fitness values and divided into two sets T_W and T_L. T_W can be considered the class of winners, T_L the class of losers. In a subsequent step, all losers will be replaced by varied copies and recombinations of winners. Whether or not an individual remains in the population can formally be seen as the result of a classification C of the comparisons V:

$$C : V \longrightarrow \{0,1\} \tag{1}$$

The comparisons $V(i,j)$ are divided into two classes, depending on which of the individuals i,j has the better fitness according to a given fitness criterion f.

$$\begin{aligned} C : V = (i,j) &\longrightarrow 0 \Leftrightarrow f(i) > f(j) \\ C : V = (i,j) &\longrightarrow 1 \Leftrightarrow f(j) \geq f(i) \end{aligned} \tag{2}$$

The fitness $f(i)$ of an individual i is usually computed by an evaluation E (see eq. (3)) that assigns a real number as a fitness value to i. A simple comparison using $<,>,\leq,\geq$, separates superior and inferior individuals afterwards. Thus, tournament selection is based on phenotypic information.

$$E : i \longrightarrow I\!R. \tag{3}$$

However, the sets T_W and T_L can be also be obtained as a result of classification (2):

$$\begin{aligned} V = (i,j) \in 0 &\Leftrightarrow i \in T_W, j \in T_L \\ V = (i,j) \in 1 &\Leftrightarrow i \in T_L, j \in T_W \end{aligned} \tag{4}$$

Any classification C that divides the tournament T in two sets T_W, T_L with $T_W \cup T_L = T$ and $|T_W| = |T_L|$ is—from a formal point of view—a valid classification. If a classification C' can be given that has the same characteristics as C, but with a reduced runtime, i.e. with a reduced need for evaluation E, the overall runtime of the algorithm will be reduced accordingly. In other words, a comparison V between two individuals, each requiring the evaluation E, can be replaced by a classification with substantially smaller runtime.

Therefore, on the one hand, C' has to operate on other criteria than C and, on the other hand, can be seen as a model of the classification C. An ideal classifier C' can now be written as

$$C' : V \longrightarrow C(V), \tag{5}$$

and is nothing else but a function that calculates the outcome of the classification $C(V)$ solely from V, i.e. from i and j. C' thus operates on genotypic information, the information coded in the genome, instead of operating on phenotypic information like C which uses E. The quality of any given classification C' on a set of comparisons V can easily be calculated as the sum of misclassifications:

$$quality(C') = \sum_{V} |C(V) - C'(V)|. \tag{6}$$

$quality(C')$ at the same time is the definition of a fitness function for any machine learning algorithm applied to the problem of creating an adequate substitute C' for C. The ratio of correct classifications and number of comparisons is a quality measure for C' and is known as *hit rate*.

3 Confidence Level

If a classifier C' is the result of a learning process, it is likely that the quality of C' is not optimal. This implies that there will be some misclassifications on the set V. Another observable quantity, the *confidence* k ($k \in [0,1]$) of a classification, is of particular use here. A classification with high confidence level k indicates a more reliable result than a classification with a low k-value. However, a certain error probability p_e ($p_e \in [0,1]$) will always remain. Introducing a confidence level k_V ($k_V \in [0,1]$) for C' yields

$$C' : V \longrightarrow < \{0,1\}, k_V > . \tag{7}$$

This ensures a confidence level k_V for every single classification of C'. Based on k_V a decision can be made whether to accept the result of the classification ($k_V > k$) or not ($k_V \leq k$). If the classification is rejected, two evaluations E are necessary to complete the comparison.

4 Influence on the Performance of the EA

This section is devoted to the investigation of the influence of the introduced method on the overall performance of EA. Therefore it is assumed that a classification C' exists for the two example problems used in this section.

It is difficult to quantify the influence of a certain number or percentage of misclassifications that allow individuals with inferior quality (real fitness) to persist in the population. Evolutionary algorithms are instances of beam search exploring regions of the search space that are defined by the composition of the population. The best route towards the global optimum is a priori unknown due to several reasons like e.g. the initialization of the population, application sequence of variation operators, random number effects, etc. It is therefore impossible to state that persisting individuals with worse fitness values (those misclassified as winners) have only negative influence on the EA's performance.

The following example shows experimentally the influence of certain combinations of confidence level k, error probability p_e and runtime ratio r between evaluation and classification on the performance of a Genetic Algorithm (parameters of the GA are displayed in Table 1) with two different fitness functions. The functions to be maximized are defined in (8) and (9).

$$f_1(i) = \sum_{l=0}^{9} i_l, \tag{8}$$

$$f_2(i) = \begin{cases} 100 \cdot i_9 & \text{,if} \quad i_5 + i_6 \geq 1 \\ (i_0 + 10i_3)^2 - (i_1 + i_2 + i_4)^3 & \text{else.} \end{cases} \tag{9}$$

Table 1. Parameters of the GA used and for the computation of runtime.

Parameter	**Value**
Objective	Maximize eq.(8) (eq. (9))
Number of runs	100
Population size	100
Length of individuals	10 Bit
Prob. of mutation	70%
Prob. of crossover	30%
Initialization	random
Selection scheme	Tournament (size = 4)
Termination criterion	Av. fitness of population $>$ 80% of max. fitness
Runtime of C (t_C)	0.1 s
Speed factor s	$[1,\ldots,1000]$
Runtime of E (t_E)	$t_E = s \cdot t_C$
Minimum confidence level k	$[0.7,\ldots,1.0]$
Error prob. p_e	$\{0.05, 0.1, 0.2, 0.3\}$
Duration of start phase (approx. 25% of standard runtime)	50 (eq. 8), 400 (eq. 9)
Standard runtime (t_S)	200 Tournaments (eq. 8),
	1528 Tournaments (eq. 9)
Learning time t_L for C'	10.0s

The first function just maximizes the number of ones in the genome, whereas the latter is a discontinuous nonlinear function of the elements of the individual i.

The fitness is always calculated for each individual. After a starting phase during which no classification takes place, individuals with better fitness values replace inferior ones with a probability p_r (see eq. (10)). With probability p_p (see eq. 11), inferior individuals remain in the population. It is furthermore assumed that the confidence levels k_V of C' are uniformly distributed in [0,1], resulting in a probability of $p_E = k$ that a classification has a confidence level of less than k, meaning that the evaluation E of the individual is necessary. This method simulates the outcome of a classification C' with an error probability p_e.

$$p_r = (1 - p_e)(1 - k) \tag{10}$$

$$p_p = p_e(1 - k) \tag{11}$$

Saving Runtime

To compute the overall runtime needed for the experiment, the number of evaluations and classifications during the evolution are counted. During the starting phase, only evaluations take place and their runtime is added to the total runtime. In the second phase, the runtime for a classification C is added for every comparison. If the confidence level k_V of C is less than k, the runtime for two more evaluations is added. Learning C' is assumed to need a certain time t_L. Every parameter needed for the computation of the total runtime is given in Table 1.

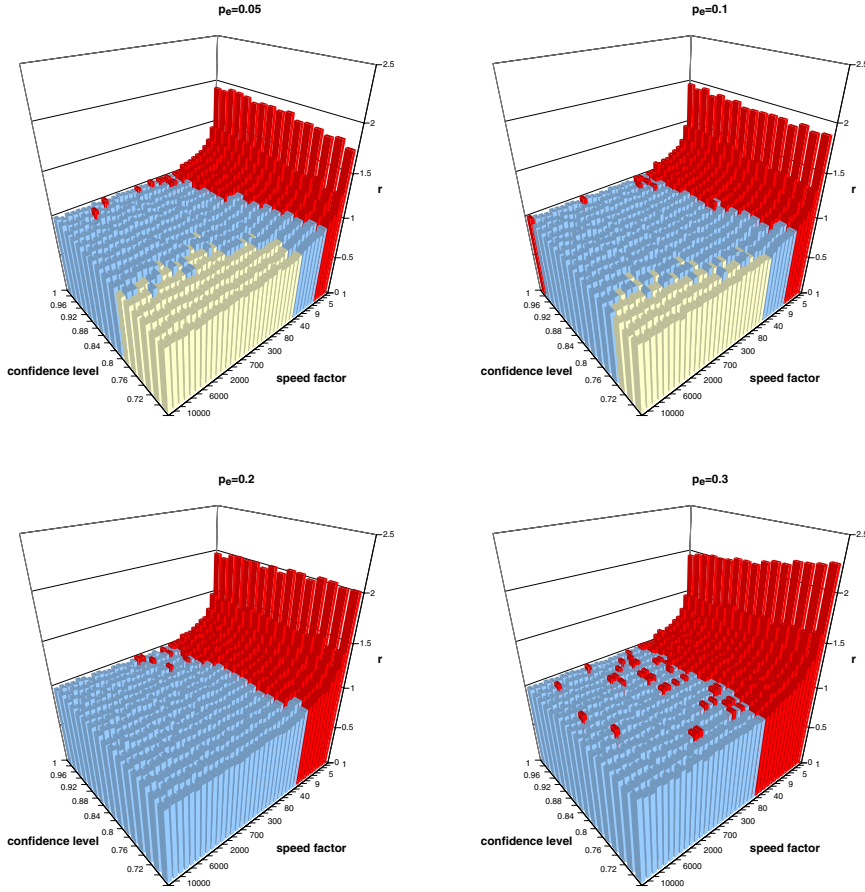

Fig. 1. Comparison of run time ratios (z-axis) of different combinations of confidence level (x-axis) and speed factor (y-axis). The lighter the colors, the smaller is the ratio. Dark regions indicate run time ratios ≥ 1, light regions indicate run time rations ≤ 0.85. Fitness function is eq. (8).

Basis for the experimental analysis is the average time of 100 identical runs (except for the random seed) until the average fitness of the whole population reaches 80% of the a priori known maximum fitness. This value is divided by the standard run time, the average runtime of 100 runs of an unmodified GA. The runtime ratio r is computed as follows:

$$r = \frac{1}{100} \sum_{i=0}^{100} \frac{n_C^i \cdot t_E + n_C^i \cdot t_C + t_L}{t_S \cdot t_E}, \tag{12}$$

with n_C^i, n_E^i the number of classifications and evaluations in run i. A ratio of one or smaller indicates that the parameter combination yields a runtime that is equal to or less than that of the standard algorithm. In Figure 1 the results are shown. It is clearly

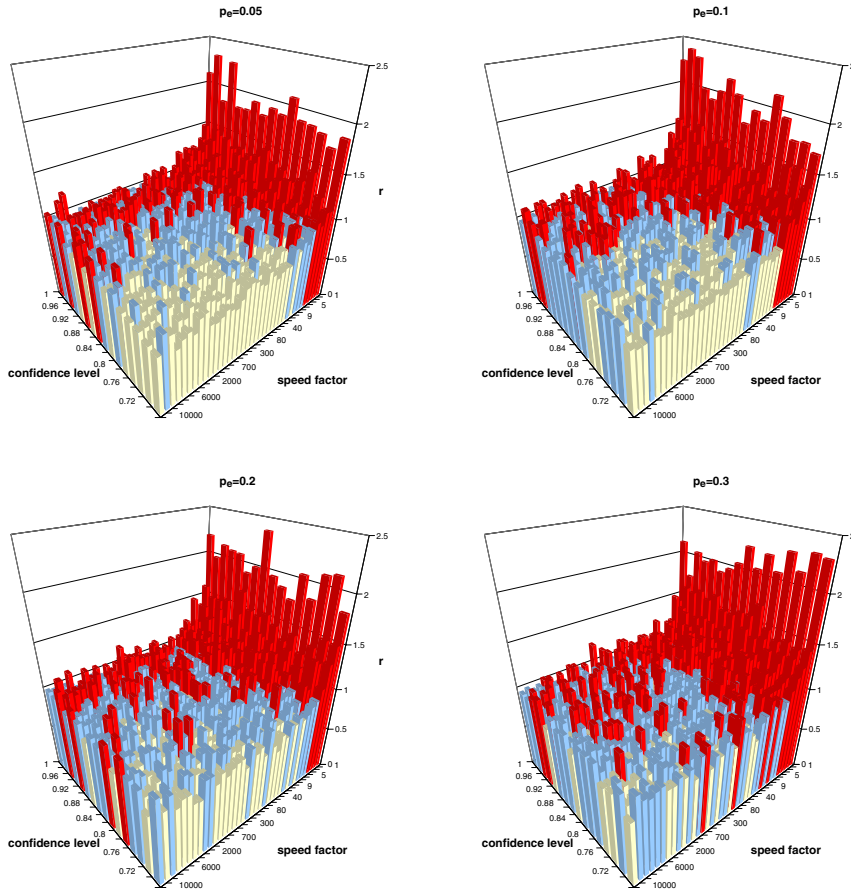

Fig. 2. Comparison of run time ratios using eq. (9).

visible that a smaller confidence level k reduces the runtime of the algorithm, whereas a smaller speed factor between classification and evaluation increases the runtime. On the other hand, increasing the speed factor to values above 300 or more does neither decrease runtime nor compensate for higher confidence levels. Contrariwise, if the speed factor falls below a certain threshold, decreasing the confidence level does not reduce the runtime any more. A common characteristic is the increasing runtime due to increasing error probability p_e. Nevertheless it is astounding that a 30% error of classification still gives run time ratios $r \leq 1$ for a significant set of parameter combinations. If the fitness function is more complex, the picture changes moderately. In Figure 2 the run time ratios are displayed according to the results using eq. (9). A more inhomogeneous behavior with different parameter combinations is visible, but the overall tendency towards lower run time ratios due to higher speed factors and lower confidence levels can be seen. Again, the run time saving reduces with increasing error probability.

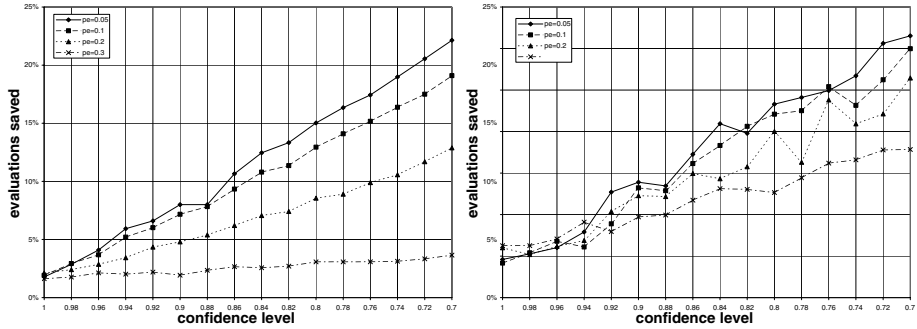

Fig. 3. Average saving of evaluations of all experiments in %. **Left:** Experiments using eq. (8). **Right:** Experiments using eq. (9).

Saving Evaluations

If the focus is on the amount of saved evaluations and not on the absolute amount of saved runtime, the saving is independent from the speed factor. The development of the number of saved evaluations depending on confidence level and error probability is shown in Figure 3. Each data point represents the average saving of evaluations of 3,700 experiments. A direct consequence of a high confidence level combined with the assumption of uniform distributed k_V values is that only a small percentage of classifications reaches this level resulting in nearly no saving. Decreasing the level of confidence increases the possibility of a successful classification and reduces the number of evaluations needed. A higher error probability reduces the savings. This observations are independent from the fitness function.

It is astounding that even with high error probabilities and complex fitness functions a 10% reduction of the number of evaluations is possible (see figure 3, right). Another interesting observation is that with a complex fitness function higher error probabilities reduce the number of evaluations more than with a simple fitness function. The reasons for this phenomenon are not yet entirely clear and need to be investigated in the future. The given example does not have enough explanatory power to justify more general conclusions.

Whether or not a classifier is able to reduce either the runtime of the algorithm or the number of evaluations can be reduced to the question if such a classifier *exists*. That the answer is positive, even for a complex genotype-phenotype-relation, will be demonstrated in the following section.

5 Evolving a Classifier with Genetic Programming

The data used for the evolution of C' have been saved during the manual evolution of control programs for a quadruped robot (see section 6). In this section a retroactive analysis of the experimental data was carried out to investigate the possibilities of online evolution of classifiers with GP [1]. The result of every comparison of every tournament was saved in the form

Table 2. Performance of classifier variants with $k = 0.65$. Lower error rates are achieved by using only data from recent generations.

		all	**last 5**	**last 2**
eq. (14)	above conf. level	63,8%	70,0%	60,0%
	variance	0.02	0.02	0.02
	misclassifications	51,7%	39,0%	32,7%
	variance	0.02	0.01	0.03
eq. (15)	above conf. level	90,3%	69,0%	78,5%
	variance	0.01	0.03	0.03
	misclassifications	54,1%	35,0%	34,7%
	variance	0.02	0.02	0.02

$$V' =< i_1, i_2, c > \tag{13}$$

with $c \in \{0,1,2\}$. A new class, class 2, was introduced to discriminate between clear (one individual has a significantly better fitness: class 0 or 1) and undetermined (indistinguishable fitness: class 2) comparisons. A classifier for each of the three classes was evolved using DISCIPULUS[1], a fast machine code GP system with linear representation. The three classifiers are combined using eq. (14), resulting in that class c_i which has the highest confidence level k_V.

$$C'(V') =< c, k_V > \text{ with } c = c_i | k_V = max\left(k_V(c_i)\right), i = 0,1,2 \tag{14}$$

Taking into consideration that a low confidence level k_V in favor of a certain class at the same time stands for a high confidence level $1 - k_V$ in favor of the other classes we use eq. (15) to combine the results alternatively.

$$C'(V') =< c, p > \text{ with } c = c_i | k'_V(c_i) = max\left(k'_V(c_i)\right), i = 0,1,2 \tag{15}$$

$$k'_V(c_i) = max\left(k_V(c_i), \frac{\sum\limits_{c_j \neq c_i} (1 - k_V(c_j))}{2}\right), i, j = 0,1,2 \tag{16}$$

Training and validation set are composed in three different variants. (i) All data are used, 50% training, 50% validation. (ii) Data from the last five generations are used (50%-50%) (iii) Only data of the last two generations are used (50%-50%). The evolved classifiers are then used to predict the outcome of the comparisons in the actual generation.

In Table 2, the results of a series of experiments are shown. The three variants of training/validation set formation are each evaluated with both versions of result combination. If the overall classification result follows eq. (14), the average saving of evaluations is slightly higher with simultaneously similar classification rates. Obviously, using only the results of recent generations for the evolution of classifiers leads to better classification results. It seems that ignoring earlier data sharpens the classification

[1] DISCIPULUS is a trademark of AIM Learning Inc. The free academic version was used with standard GP settings for classification problems, no changes were made by the authors [5].

Fig. 4. The four-legged Sony Aibo robot used in the experiments. The robot has 20 degrees of freedom.

results. This might might be caused by the fact that the fitness of a classifier is based on a uniform classification rate on all elements of the training set, a fact that neglects the changing structure of the population during an evolutionary process.

This example demonstrates that it is possible to generate classifiers with GP based on genotypic information alone that are able to replace the time consuming evaluations normally necessary for tournament selection in Evolutionary Algorithms. The next section shows an online application of this technique.

6 Application

The method introduced above is now tested with a real world problem, the evolution of gaits for walking robots. The used robot is a Sony Aibo, a four-legged robot (figure 4) which has successfully been used for similar experiments before [4]. Here, a set of 16 parameters of an inverse kinematic transformation [10] has to be evolved describing gait patterns for four-legged walking robots (Sony Aibo robots). The parameters of the GA can be found in Table 3.

The evaluation of the individual is done manually due to reasons inherent in the problem of evolving control programs for real walking robots: it is on the one hand difficult to mathematically formulate a sufficiently precise fitness function, on the other hand, once you have such a function, it is difficult to set up all necessary measuring devices to get detailed information about the actual state of the experiment. Using a simulated robot here reduces the wear out but entails other problems instead [13, 6]. Therefore, a interactive evolution[2] with tournament selection was started, during which the experimenter just had to decide which of two individuals had better fitness (this

[2] Interactive evolution embedds human intuition, preference, subjectivity, cognition, perception, and sensation into an EA. An introduction into the field of interactive evolution can be found e.g. in [12].

Table 3. Parameter setting of the GA for the Evolution of gait patterns.

Parameter	**Value**
Objective	maximize forward walking speed
Population size	26
individual size	16
Terminal set	$\mathbb{R}$
Prob. of crossover	0.5
Prob of mutations	0.2
Prob. of reproduction	0.3
Selection scheme	Tournament (size = 4)
Initialization	Standard parameter set (GT2002) with added noise

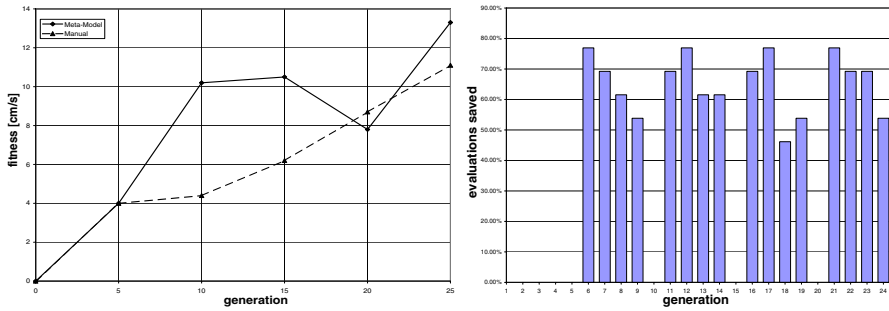

Fig. 5. Left: The fitness of the best individual of the population with manual and meta-model evolution. **Right:** Percentage of classifications per generations. The meta-model evolution reaches a similar fitness with only approx. 53% of evaluations needed in the manual evolution. The whole population is evaluated manually every five generations to extract reliable information about convergence (therefore $p_{1,2,3,4,5,10,15,20,25} = 0.$)

is, in fact, a realization of eq. (2) with subsequent mapping (4)). In order to get an unbiased overview over the performance of the individuals, the whole population is evaluated manually every five generations and the forward speed of each individual is measured. This first experiment was carried out to have a basis to compare the results of the meta-model evolution with.

To evolve 25 generations takes 650 manual evaluations and yields the results shown in Figure 5. The best-of-run individual, when executed on the robot, reaches a speed of 11 cm/s (see Figure 5).

In a second evolutionary experiment, starting from generation five of the manual evolution, a meta-model for every generation was created by evolving classifiers with DISCIPULUS as presented in section 5. The manually evaluated tournaments of the past five generations together formed the training and validation set (75%-25% here, because of the small amount of reliable fitness cases, i.e. manually evaluated ones) for the evolution. The evolved three classifiers (for every class 0,1, and 2) were used to compute the outcome of each of the 13 comparisons V (using eq. (15)) to combinate the particular results. If the confidence level k_V of the overall classification C' was smaller than the

minimal value k, which was set to 0.65 here, both individuals of the comparison were executed on the real robot and the result was manually determined. This comparison was afterwards added to the training set for the next generation. The better individuals, either determined by classification or manually, formed the set T_W. After the variation step, Individuals of T_W replace the worse individuals in T_L.

In Figure 5, the best-of-generation individuals are displayed, showing a better performance (the best-of-run individual reaches a maximum speed of approx. 13 cm/s). It is remarkable that the evolution just uses 306 manual evaluations, summing up to a total saving of **approx. 53%**. The number of classifications per generation are shown in Figure 5, too.

Carrying out the manual evolution took about 8 hours, the meta-model evolution took about twice the time, due to the time consuming 48 evolutions of classifiers for every single class (see eq. 14) and every generation. The next step will be to reduce this time, probably by adapting recent classifiers to the only slightly changed training and validation sets.

7 Discussion

Using a meta-model of the fitness function in form of a classifier for tournament selection is able to reduce the number of evaluations during evolution and the total runtime of experiments significantly. The method presented here seems to be resilient against misclassifications permitting worse individuals to persist in the population. However, the classifiers have to be generated by another machine learning algorithm, raising another computationally intensive problem to be solved. The number and nature of experiments shown here do not give sufficient background for more general statements, but the method seems to be powerful enough to be investigated further in the future.

Acknowledgements

This project is supported by the Deutsche Forschungsgemeinschaft (DFG), under grant Ba 1042/6-2. The authors wish to thank Walter Nowak and Philipp Limbourg for their technical support.

References

1. W. Banzhaf, P. Nordin, R Keller, and F. Francone. *Genetic Programming — An Introduction*. dpunkt/Morgan Kaufmann, Heidelberg/San Francisco, 1998.
2. M. Emmerich, A. Giotis, M. Özdenir, T. Bäck, and K. Giannakoglou. Metamodel-assisted evolution strategies. In J.J. Merelo Guervos, P. Adamidis, H.-G. Beyer, J.-L. Fernandez-Villacanas, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature VII (PPSN)*, Lecture Notes in Computer Science, pages 361–370. Springer, Berlin, 2002.
3. M. Giacobini, M. Tomassini, and L. Vanneschi. Limiting the number fitness cases in genetic programming using statistics. In J.J. Merelo et al., editor, *Proccedings of the Seventh International Conference on Parallel Problem Solving from Nature (PPSN)*, Lecture Notes in Computer Science, LNCS 2439, pages 371–380. Springer Verlag, 2002.

4. G. S. Hornby, M. Fujita, S. Takamura, T. Yamamoto, and O. Hanagata. Autonomous evolution of gaits with the sony quadruped robot. In W. Banzhaf et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1297–1304. Morgan Kaufmann, 1999.
5. AIM Learning Inc. *Discipulus Owner's Manual*. http://www.aimlearning.com/Technology Overview.htm. WWW-Document.
6. N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In F. Moran et al., editor, *Proceedings of the 3rd European Conference on Artificial Life*, Lecture Notes in Artificial Intelligence, pages 704–720. Springer New York, Berlin, Heidelberg, 1995.
7. Y. Jin. *Fitness Approximation in Evolutionary Computation - Bibliography*. http://www.soft-computing.de/amec.html. WWW-Document.
8. Y. Jin, M. Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6(5):481–494, 2002.
9. H.-S. Kim and S.-B. Cho. An efficient genetic algorithms with less fitness evaluation by clustering. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 887–894. IEEE Press, 2001.
10. M. Risler. *InvKinWalkingEngine - Kurze Beschreibung der Parameter*. Technische Universität Darmstadt, GermanTeam 2002, 2002.
11. K. Sastry, D.E. Goldberg, and M. Pelikan. Don't evaluate, inherit. In L. Spector et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 551–558. Morgan Kaufmann, 2001.
12. H. Tagaki. Interactive evolutionary computation: System optimization based on human subjective evaluation. In *IEEE International Conference on Intelligent Engineering Systems (INES)*, pages 1–6. IEEE Press, 1998.
13. J. Ziegler, J. Barnholt, J. Busch, and W. Banzhaf. Automatic evolution of control programs for a small humanoid walking robot. In P. Bidaud and F. Ben Amar, editors, *Proceedings of the 5th International Conference on Climbing and Walking Robots (CLAWAR)*, pages 109–116. Professional Engineering Publishing, 2002.