Competition and Cooperation in Extended Evolutionary Algorithms

Hartmut Pohlheim
DaimlerChrysler AG, Research and Technology
Alt-Moabit 96a, 10559 Berlin, Germany, +49 30 39982 456
hartmut.pohlheim@daimlerchrysler.com

Abstract

The solution of complex optimization problems is a task often hard to tackle. A promising approach to overcome the difficulties is to combine multiple optimization methods.
In this paper we introduce two extensions to Evolutionary Algorithms: the *application of different strategies* and *competing subpopulations*. Both extensions and their basic properties are discussed in detail. The utility of the extensions is presented using well-known test functions and real-world problems. The real-world problems are taken from the field of Evolutionary Testing – a major research project in automatic software testing at DaimlerChrysler.
This paper concludes with a description of the advantages of the introduced extensions compared to similar work done by other researchers.

1 INTRODUCTION

The optimization of complex systems is a challenging task. We can only master it by combining powerful optimization algorithms and analysis tools. Many of the known algorithms cannot be adjusted to the solution of large problems We have to find new and/or extended methods which are even able to tackle complex problems within huge search domains.

A promising approach for the development of these methods is the intelligent combination of several optimization methods by maintaining the advantageous characteristics of each method (hybridization).

This often results in a rigid combination of two optimization methods. For instance, the first optimization algorithm would work the whole time whereas the second one would only be switched on from time to time to exploit its specific features. Or both methods would be applied alternately according to a fixed schedule.

These rigid schemata lead to several disadvantages:

- The utility of each separate method compared to each other is not evaluated during an optimization run and can therefore not be taken into consideration.
- The methods are applied simultaneously without taking into account an adapted distribution of resources.
- The methods are applied successively.

A large number of problems can be solved successfully using these simple hybridization methods. However, confronted with the optimization of complex systems the listed disadvantages become noticeable. These complex systems often require the application of more than two optimization methods. Furthermore, the combination and chronological application of the methods constitutes a difficult question. We have to find a technique which automates the combination and interaction of several optimization algorithms, especially for real-world applications.

In Section 2 we introduce two extensions to Evolutionary Algorithms (application of different strategies and competing subpopulations) offering an elegant approach mastering the aforementioned problems. Section 3 and 4 describe these extensions and their parameters in detail. Their basic properties and their utility in real-world application is presented in Section 5. Section 6 gives concluding remarks and compares the presented extensions of Evolutionary Algorithms with similar work by other researchers.

2 EXTENDED EVOLUTIONARY ALGORITHMS

Evolutionary Algorithms have proved a powerful optimization algorithm for the successful solution of a large number of optimization problems. Evolutionary Algorithms are especially used in those cases where the characteristics of the system are not or only partly known or where classical mathematical methods are not applicable.

For the solution of real-world applications it is often difficult to decide which of the available Evolutionary Algorithms are best suited and how the operators and parameters should be combined.

We present two extensions to Evolutionary Algorithms. The first one allows the combination of several different Evolutionary Algorithms. It is called *application of different strategies*. The second extension enables the interaction of the different strategies as well as an efficient distribution of the resources. It is called *competing subpopulations*.

Pohlheim, H.: Competition and Cooperation in Extended Evolutionary Algorithms. In *Spector, L. (ed.)*: GECCO'2001 - Proceedings of the Genetic and Evolutionary Computation Conference – Late Breaking Papers. San Francisco, CA: Morgan Kaufmann Publishers, 2001.

Both extensions are based on the regional population model. The regional population model is often referred to as migration model, coarse grained model, or island model. Many papers were published about this population model ([5], [1]). Quite a few focus on the parallel implementation. However, this population model is not only useful for a parallel implementation. Even when used in a serial manner the regional population model proves advantageous compared to a global population model (panmictic population).

The main feature of the regional population model is that parents are selected in a regionally separated pool. The whole population is subdivided into subpopulations which are insulated from each other. An exchange of information (exchange of individuals) between the subpopulations takes place from time to time. This process is called migration.

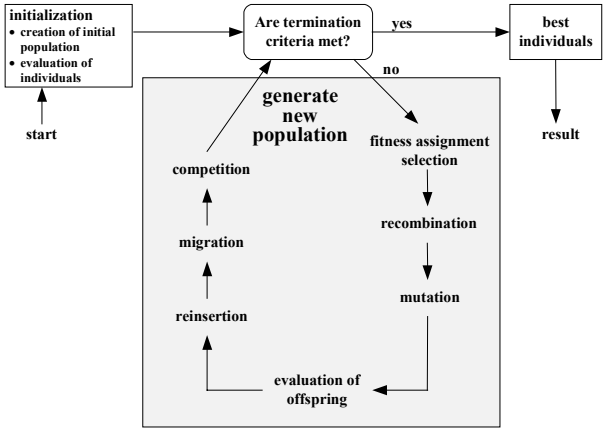

Figure 1. Structure of Extended Evolutionary Algorithms

Both extensions directly adapt themselves to the known structure of Evolutionary Algorithms (see Figure 1). The *application of different strategies* is not detectable in this structure. It is part of the implementation of the high-level operators. For the *competing subpopulations* an extra operator competition is necessary and can directly be added to the structure. This extension is the only change made.

3 APPLICATION OF DIFFERENT STRATEGIES

Evolutionary Algorithms provide a large number of different operators and methods. Thus they can be applied broadly. The search strategies can be adapted to a wide range of problems. On the one hand, there are operators for parameter optimization, sequence optimization, up to the solution of very specific problems. On the other hand, it is possible to vary the search from a globally oriented search to a locally oriented search by setting appropriate parameters. Thus, the user is provided with an extensive toolbox. Now he need only select the appropriate methods.

If the user is well familiar with the type of the problem the choice of a good search strategy is not very difficult. However, in real-world application the necessary system knowledge is seldom available. Another point is that one search strategy is rarely best suited or sufficient. Mostly, the start of a search will have to be done with a different strategy than the end. The *application of different strategies* is an answer to this problem.

The application of different search strategies for each subpopulation is mentioned in a few of the early papers about parallel evolutionary algorithms. In [11] TANESE describes the use of different mutation and recombination probabilities for some or all subpopulations. The results of the experiments indicate that the regional population model with different parameters for each subpopulation was more robust than the use of identical parameters for which the best values are not known.

3.1 DESCRIPTION

The *application of different strategies* is the result of the definition of specific strategy parameters for every subpopulation (e.g. different mutation and recombination operators and/or parameters). In this way it is possible to specify a different behavior for every subpopulation of the Evolutionary Algorithm. To what extent the strategies differ from each other only depends on the application, the aims of the different strategies, as well as the possibilities of the optimization tool used.

On the one hand, strategies might differ with regard to only one parameter (e.g. mutation range which influences the size of the mutation steps). On the other hand, completely different types of Evolutionary Algorithms can be used. The principle of *application of different strategies* in itself does not pose any limitations.

The different strategies, however, do not only work concurrently. Rather, one strategy can be the basis for the success of another strategy. In this case the strategies support each other, i.e. the simultaneous application of the different strategies leads to success. Thus, the *application of different strategies* leads to a cooperation between subpopulations.

3.2 ORDER OF SUBPOPULATIONS

It is necessary to find a measure for the evaluation of the application of different strategies which enables an assessment of the success of each strategy.

The success of a subpopulation is measured by its rank within the order of subpopulations. In this case, a low rank is better than a high rank. This approach is analogous to the ranking during the fitness assignment/selection process. From the rank of a subpopulation we are able to assess the utility of a subpopulation in comparison to the other subpopulations.

To calculate the order of subpopulations they have to be sorted according to one criterion. Since a subpopulation is made up of a number of individuals its quality results from the properties of its individuals.

The calculation of the order of subpopulations is carried out according to the following system:

1. A number of the best individuals is selected from each subpopulation for the evaluation. This can either be the best individual, a number of individuals, or all individuals of the subpopulation.
2. These individuals are ranked according to their quality (i.e. their fitness value). The evaluation of the individuals is identical to the fitness assignment procedure. We suggest the use of linear ranking. This assigns a quality value (fitness value) to each individual compared to the quality of the other individuals in the population ([7], Section 3.1).
3. The fitness of the individuals of each subpopulation is combined to an evaluation of the subpopulation (e.g. calculating the average fitness value of the individuals in a subpopulation).
4. By sorting the evaluation of the subpopulations we obtain their order.

The order of subpopulations offers a temporary picture. The order can fluctuate considerably within several generations, depending on the applied operators and parameters (especially if the strategies are similarly successful). It is therefore of advantage for the assessment and visualization if the order of the subpopulations is filtered. In this way a *position value* is calculated from the *ranking value*, which constitutes a weighted average of the rank of the subpopulation of previous generations.

$$position\ value_{generation} = 0.9 \cdot position\ value_{generation-1} + 0.1 \cdot rank_{generation} \quad (1)$$

The applied parameters of the filter in Equation 1 prevent a strong fluctuation of the position values. The sum of both parameters must be 1. Higher values for the first parameter lead to a higher damping. Similar values are used in automatic control applications for this simple filter type.

The smaller the position value the greater the success of a subpopulation. If the rank of a subpopulation does not change for a long time the position value becomes equal to the rank. The position value is used for all following calculations and visualizations to illustrate the order of subpopulations.

4 COMPETITION BETWEEN SUBPOPULATIONS

A logical extension of the regional population model with the application of different strategies is the principle of competing subpopulations. During the application of different strategies within the regional model the size of every subpopulation (number of individuals) remains constant. Even when a strategy is not successful it still uses the same resources.

When using competing subpopulations this fixed amount of resources is not kept up. Instead the size of a subpopulation is made dependent on the current success of its strategy. Successful subpopulations receive more resources, less successful ones have to transfer resources to other subpopulations.

Every time a competition is executed between competing subpopulations the following steps have to be carried out:

- calculation of the order of subpopulations (see Section 3.2),
- calculation of the division of resources,
- execution of distribution of resources.

By means of an appropriate algorithm the resources are efficiently redistributed as soon as there is a shift within the success of the subpopulations.

So far little has been published on the application of competing subpopulations. [9] introduces a simple variant of competing subpopulations for strategy adaptation. Here, the order of subpopulations was not calculated. Instead, only the best subpopulation was determined on the basis of the best individual of the whole population. Only the best subpopulation received individuals from all other subpopulations. The competition selection took place uniform at random. In [10] this concept was extended. By introducing the resource consumption parameter it becomes possible to control the relative size of a subpopulation. This version represents the basic model of competing subpopulations. The greatest simplification concerns the division of resources which just depends on the overall best individual. Thus, a subpopulation is always unsuccessful if it does not contain the best individual of the population. This method prefers only one strategy, all others are neglected. Especially during the transition between successful strategies this can lead to an ineffective distribution of resources.

4.1 DIVISION OF RESOURCES

For every competition between subpopulations the available resources are redistributed. This can be done by determining successful and less successful subpopulations, or by a weighted division of the resources.

If we take a closer look at this process of the division of resources we find similarities to the fitness assignment of Evolutionary Algorithms ([7], Section 3.1). Fitness assignment means that every individual is assigned a reproduction probability/fitness (possible number of offspring) depending on its objective value compared to the objective values of the other individuals in the selection pool. The fitness of an individual is converted into resources enabling the individual to produce offspring.

The properties of the fitness assignment can directly be applied to the division of resources of competing subpopulations. This leads to a weighted division of the resources. It is known that procedures based on the ranking of individuals are most robust. The (filtered) position value, see Equation 1, serves as the basis for the division of resources.

In the next step every subpopulation is assigned a part of the available resources according to its rank/position value. Linear and non-linear ranking are well-known methods

from the field of fitness assignment. The parameter of these methods is the selection pressure. In analogy we define the division pressure *DP*. This parameter determines how the resources are distributed. For a low division pressure the differences in the resources of every subpopulation are small. When applying a high division pressure, especially for non-linear ranking, the best subpopulation obtains a much larger share of resources. For the other subpopulations the share of resources is clearly smaller. Nevertheless, the other good subpopulations receive a larger share of resources than the less good subpopulations.

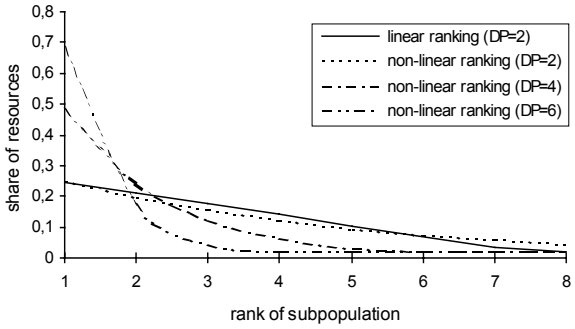

Figure 2. Division of resources for subpopulations: linear and non-linear ranking and different values of division pressure

Figure 2 shows the division of resources of eight subpopulations for different values of the division pressure. A minimum of resources (subpopulation minimum, here 1%) needed for the survival of every subpopulation is taken into account.

When applying linear ranking a maximum division pressure of only 2 is possible. If the best subpopulation(s) is to be given stronger preference we have to use non-linear ranking. In this way values of division pressure up to *NumberofSubpopulations*-2 can be applied.

4.2 DISTRIBUTION OF RESOURCES

The following parameters/methods have to be defined in order to execute the distribution of resources:

- resource consumption by the individuals,
- competition interval: frequency of a competition for resources,
- competition rate: maximum share of resources which has to be transferred by less successful subpopulations,
- competition selection: type of selection of resources to be transferred,
- subpopulation minimum: minimum size of an unsuccessful subpopulation.

However, all these parameters can be set with default values for nearly all applications. Only for very special problems different parameter values should be defined. Thus, the use of competing subpopulations imposes no additional parameters for the standard user.

4.2.1 Resource Consumption

So far we have only mentioned resources. However, what is missing is the conversion to the number of individuals used by the Evolutionary Algorithm. The resource consumption indicates how many resources are needed per individual.

The simplest variant is that every resource unit corresponds to one individual. This assumes that the individuals of all strategies consume the same amount of resources.

However, it is more realistic for individuals of different strategies to differ in their consumption of resources. This assumption permits a better modeling of the competition between subpopulations/strategies.

For instance, three different strategies with the same amount of resource units can provide for a varying number of individuals depending on the resource consumption parameter. The following example assumes 10 resource units for every strategy:

- resource consumption = 1: 10 individuals,
- resource consumption = 5: 2 individuals,
- resource consumption = 0.1: 100 individuals.

The parameter of resource consumption determines inverse proportionally to what extent a subpopulation will grow when assigned additional resources. At the same time the resource consumption controls whether a strategy will work with a small number of individuals (high consumption of resources), or with a large number of individuals (low consumption of resources).

4.2.2 Competition Interval and Competition Rate

The competition interval defines the points in time for a competition between subpopulations. It also indicates the span of time between competitions in which the subpopulations exist without any change of available resources and it permits the subpopulations to develop for the next competition.

It is easiest to specify a defined number of generations as competition interval. However, it is also possible to define the time of the competition depending on other events. One possibility is to carry out a competition every time a subpopulation has made a clear progress (we are aware that the definition of a “clear progress” is difficult and problem-specific).

The maximum amount of resources transferred by one subpopulation during a competition is determined by the competition rate. The competition rate is specified proportionally to the current resources of the subpopulation (and not as a fixed value). This ensures that subpopulations with few resources transfer a smaller amount of resources than subpopulations with many resources. The competition rate should always involve only a part of the resources of a subpopulation.

Following is a list of reference values for the competition interval (depending on the average or initial number of in-

dividuals per subpopulation) and the competition rate which proved to be suitable as predefined values.

$$\textit{competition interval} = \begin{array}{l} 20\% \cdot \textit{NumIndSubPop}\ [\text{generations}] \\ (10\% - 50\% \Rightarrow 4 - 20\ \text{generations}) \end{array} \quad (2)$$

$$\textit{competition rate} = \begin{array}{l} 10\%\ [\text{resources of subpopulation}] \\ (5\% \text{–} 20\%\ \text{of resources of subpopulation}) \end{array} \quad (3)$$

If the competition interval has a very small value (< 6 generations) the competition rate should also be very small. This avoids a too rapid redistribution of resources between subpopulations. If a higher competition rate (> 10%) is used, a higher competition interval should be selected.

4.2.3 Competition Selection

The competition selection indicates how individuals are selected which are removed from less successful subpopulations.

There are several ways to select the individuals:

- the worst individuals,
- uniform at random selected individuals, and
- the best individuals.

A point in favor for selecting the worst individuals is that unsuccessful subpopulations would not be further weakened during a competition. Consequently, selecting the best individuals means that unsuccessful subpopulations would be further punished than just by the decrease in the number of individuals. The random selection of individuals is in-between the other two options. The selection of the best or worst individuals can be executed by means of one of the known selection methods of Evolutionary Algorithms ([7] Section 3.2).

4.2.4 Subpopulation Minimum

To avoid the complete disappearance of less successful subpopulations it is necessary to specify a minimum subpopulation size and/or a minimum amount of resources retained by the subpopulations. Resources can only be removed until this minimum is reached. Afterwards its size remains constant (until a possible success of the subpopulation).

The subpopulation minimum can be expressed by:

- a fixed amount of individuals,
- a proportion of the average resources/size of the subpopulation, or
- a proportion of all available resources

A better comparison between different-sized subpopulations is possible when the subpopulation minimum is specified proportionally to the subpopulation size rather than stating a fixed amount of individuals/resources. On the other hand, it is often known what minimum size a subpopulation needs in order to work. The most flexible option is the definition as proportion of the overall available resources.

$$\begin{matrix} \textit{subpopulation} \\ \textit{minimum} \end{matrix} = \begin{cases} 6\ (4-10)\ [\text{individuals}] \\ 20\%\ (10\% - 30\%)\ [\%\ \text{of subpopulation}] \\ 4\%\ (1\% - 10\%)\ [\%\ \text{of all resources}] \end{cases} \quad (4)$$

The successful application of most Evolutionary Algorithms can hardly be guaranteed if the subpopulation minimum is set to a very low value (4-6 individuals). A success of such a small subpopulation is only achievable if its strategy is superior to the strategies of the other subpopulations.

5 APPLICATION OF EXTENDED EVOLUTIONARY ALGORITHMS

This chapter illustrates the application of the introduced extensions using a simple and easily comprehensible objective function. In a second part we demonstrate their application to real-world problems taken from our current work.

5.1 APPLICATION OF DIFFERENT STRATEGIES

Figure 3 shows an example of the course and the results of an optimization of DEJONG's Function 1 (hypersphere with 10 dimensions) using the *application of different strategies*. The optimization was executed with 5 subpopulations, with 15 individuals each. Each subpopulation used a different strategy (different parameters of the real-valued mutation operator): 1: large mutation steps; 2: middle-sized mutation steps; 3: small mutation steps; 4: tiny mutation steps; 5: middle-sized and small mutation steps (mutation range: 1e-2, 1e-4, 1e-6, 1e-8, 2e-1; mutation precision: 6, 6, 6, 6, 16). The smaller the mutation steps the more locally oriented the search. All strategies used the same recombination operator (discrete recombination), all other parameters were identical too (e.g. *generation gap*: 0,9). Every 40 generations a migration between subpopulations took place (complete net structure). See [6] for an extensive discussion of these operators.

The left diagram in Figure 3 shows the beginning of the optimization. We can see that strategy 5 is the most successful one at the beginning. This changes after 200 generations. Now strategy 2 becomes better, taking up the leading position after 220 generations (see middle diagram). Strategy 3 becomes the best one after 300 generations, followed by strategy 4 after 500 generations (see right diagram). Strategy 1 shows a good behavior during the first 50 generations (see left diagram), however it then falls behind and will not be able to be successful at any time.

This is not surprising if one knows the properties of the example function. Large mutation steps are especially successful at the beginning. Later they only lead to deterioration. Similarly this applies to each of the strategies 2-4. If larger steps still make progress, small steps are too slow.

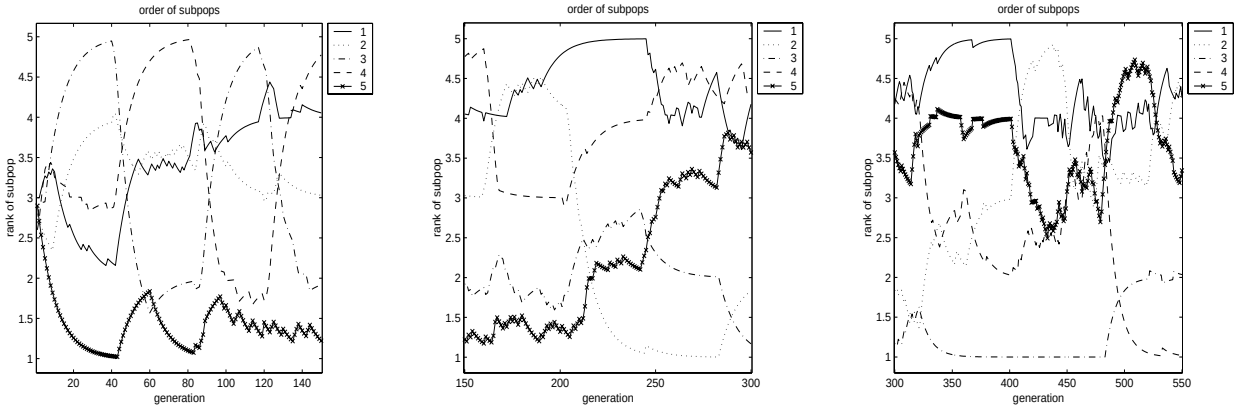

Figure 3. Application of different strategies, order of subpopulations; left: beginning of optimization run, middle: middle phase, right: final phase

Thus, each of the first four strategies is best during a specific period.

Strategy 5 takes up a special position. The left diagram shows clearly that strategy 5 is the most successful one at the beginning: larger steps now and then, and mostly smaller steps is a good strategy. After 200 generations strategy 5 runs out of breath. It rarely produces good individuals anymore. Now one of the more specialized strategies is advantageous.

5.2 COMPETING SUBPOPULATIONS

The following example demonstrates the application of competing subpopulation. It uses the same objective function and the same strategies as the example above with the addition of competition between subpopulations.

The competition parameters were set as follows:

- only the best subpopulation receives resources,
- the resource consumption for each individual is 1,
- the competition interval is 4 generations,
- the worst individuals are removed from less successful subpopulations,
- the subpopulation minimum is set to 5 individuals.

The example presents a powerful but still comprehensible variant of competing subpopulations. Figure 4 shows selected sections of the course and the results of the optimization run.

The left and middle diagram in Figure 4 show the course of the competition between the subpopulations clearly. The left diagram displays the absolute size of the single subpopulations, and the middle one their relative size using a stacked line plot. The number of individuals in the middle diagram results from the distance between two contiguous lines, i.e. the size of each subpopulation can be identified without a legend.

At the beginning of the optimization run the size of subpopulation 1 (large mutation steps) increases for a short time. After about 50 generations subpopulation 5 grows (large and middle-sized mutation steps). Subpopulation 2 is only successful after 320 generations (middle-sized mutation steps). Subpopulation 3 obtains more individuals after 400 generations (small mutation steps). For about the last 200 generations subpopulation 4 (tiny mutation steps) is the most successful one, having the most individuals. At all times the less successful subpopulations/strategies use only a small proportion of the overall population.

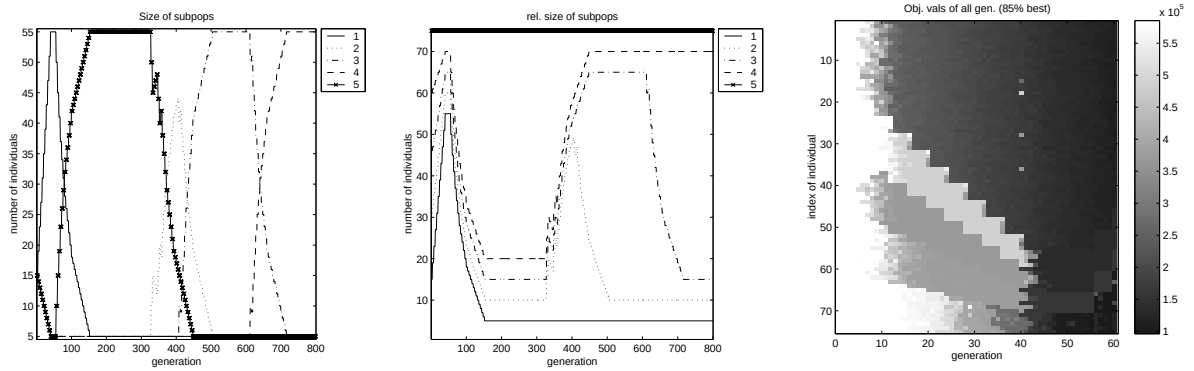

Figure 4. Competing subpopulations; left: size of subpopulations, middle: relative size of subpopulations, right: objective values of all individuals at the beginning of the optimization run

An additional insight into the course of the optimization is given by the right diagram in Figure 4. It shows the objective values of all individuals over several generations (the individuals with the lowest index numbers form subpopulation 1, the individuals with the highest index numbers subpopulation 5, the other subpopulations are in-between). The division between the subpopulations is clearly visible and enables the user to indirectly recognize the size of every subpopulation. The diagram shows the beginning of the run where subpopulation 1 was successful and increased in size. The reason for this can also be seen: the greatest improvement of the objective values takes place in subpopulation 1. Subpopulations 2, 3, and 4, however, do not show much of an improvement. Only subpopulation 5 demonstrates a clear progress during the first 40 generations. The migration in generation 40 distributes the best individuals of the subpopulations. Thus, the less successful subpopulations are improved towards the level of quality of the best subpopulation. For further diagrams depicting the transitions between successful strategies please refer to [7], Section 4.6.

5.3 REAL-WORLD APPLICATIONS

This section demonstrates the use of competing subpopulations in real-world applications. The examples are drawn from the field of Evolutionary Testing:

- structure-oriented testing of software modules,
- temporal testing of real-time software modules.

Testing is the most significant analytic quality assurance measure of software. The systematic design of test cases is crucial for the test quality. Structure-oriented test methods, which define the test cases on the basis of the internal program structure, are widely used. For the automation of structure-oriented test case design we use Evolutionary Testing. A test environment was developed, that realizes a fully automatic test data generation for most structure-oriented test methods.

Real-time systems must also be checked for their temporal correctness. The temporal behavior of real-time systems is defective when the computation of input situations violate specified timing constraints. In general, this means that outputs are produced too early or their computation takes too long. The tester's task, therefore, is to find the input situations with the shortest or longest execution times to check whether they produce a temporal error.

Both test systems apply the GEATbx as optimization tool [6]. This toolbox is a powerful tool supporting different representations of the individuals (binary, integer, real). It is possible to implement genetic algorithms, evolution strategies as well as many hybrid forms of evolutionary algorithms. The GEATbx offers a large number of different operators and possesses extensive visualization functions for displaying the optimization process.

The GEATbx contains a complete implementation of competing subpopulations (including the application of different strategies). This allows the application of these extensions of Evolutionary Algorithms in large and complex real-world applications.

The application of different strategies and competing subpopulations serves multiple aims at the same time:

- During the initial evaluation of the systems we could easily test a number of parameter settings to determine promising strategies for the respective problem class (structure-oriented testing or temporal testing of real-time software). Simultaneously we excluded unsuccessful strategies (competition between strategies).
- During the productive use of Evolutionary Testing all promising strategies were used at the same time. This is particularly important. We could use one configuration for a large class of software modules. The user of the test tool need not bother with the configuration of the optimization engine. Nevertheless, the optimization algorithm is still powerful.
- Not all of the applied strategies were successful for each single software module/optimization run. However, the division of resources ensured only a small amount of resources being used for currently not successful strategies.
- Very often more than one of the applied strategies is successful during an optimization run. Each of these strategies is particularly successful at a specific point in time.

Table 1: Maximum execution times of engine control software modules determined by Evolutionary and systematic testing

	max. exec. time in µs			
module name	Evolutionary Testing	developer test	lines of code	parameters
zr2	69,6 µs	67,2 µs	41	18
t1	120,8 µs	108,4 µs	119	18
mc1	112,0 µs	108,4 µs	98	17
mr1	68,8 µs	64,0 µs	81	32
k1	59,6 µs	57,6 µs	39	14
zk1	58,4 µs	54,0 µs	56	9

The results of both application domains demonstrate that the developed test systems are able to perform automated software testing. In [8] we compared the results of black-box temporal testing to the results of the developer of the software module (white-box testing). The automated test found always the same or better execution times, see Table 1.

In [12] results of structure-oriented testing are reported. A number of typical test objects were examined. For all reported modules maximum branch coverage was achieved.

An extensive discussion of both real-world systems can be found in separate publications (temporal testing [8], structure-oriented testing [12]).

6 CONCLUDING REMARKS

This paper introduces two extensions to Evolutionary Algorithms: the *application of different strategies* and *competing subpopulations*.

The *application of different strategies* permits the simultaneous application of different parameter settings. This accelerates and facilitates an optimization in contrast to the execution of multiple independent experiments. Results are presented more clearly and in a compact manner. It is very easy to identify successful and unsuccessful strategies as well as the varying success of strategies during an optimization run. Furthermore, the cooperation between strategies during a run can lead to their achieving better results than separated strategies. The success of a strategy at the beginning of a run often is the basis for the subsequent success of another strategy.

The use of different strategies enables a higher level of the application of Evolutionary Algorithms, especially within the following two areas:

- testing a number of parameter settings to determine successful strategies and simultaneously exclude unsuccessful ones (competition between strategies),
- the simultaneous application of different strategies where each strategy is particularly successful at a specific point in time, i.e. the strategies supplement each other (cooperation between strategies).

The extension *competing subpopulations* is based on the application of different strategies, however, it goes one step further. The different strategies are not only applied simultaneously but the successful ones receive more resources than the less successful ones. This leads to a dynamic distribution of resources.

The resource distribution leads to an indirect adaptation of the strategy parameters during an optimization run. This means the user does not have to specify the parameters beforehand because they are indirectly set depending on the success of the subpopulations. This opens up new dimensions to the application of Evolutionary Algorithms.

By applying competing subpopulations those strategies less suitable for solving the problem will receive less resources. Thus, only a small amount of resources are used for testing unsuccessful strategies which in turn enables the testing of additional promising strategies for very complex problems.

The approach introduced in this paper eliminates restrictions of earlier publication ([9], [10]). The inclusion of the best, multiple, or all individuals of the subpopulations enables the calculation of a weighted order of the subpopulations. This leads to a multilevel assessment of the success of the strategies/subpopulations. The weighted division of resources permits a distribution to all subpopulations and not only to the best subpopulation. Especially when employing similar or supplementary strategies this leads to a more equitable division of the available resources.

The presented extensions of Evolutionary Algorithms, *application of different strategies* and *competing subpopulations*, are very powerful, especially in real-world applications. At DaimlerChrysler these extensions are an integral part of the optimization engine of two test tools used for structure-oriented testing of software and temporal testing of real-time software. The extensions allow the setup of powerful optimization algorithms applicable to a large class of real-world software test problems.

Both extensions constitute another step towards the development of powerful Evolutionary Algorithms for the solution of large and complex systems.

References

[1] Cantú-Paz, E.: A Summary of Research on Parallel Genetic Algorithms. Technical Report IlliGAL No. 95007, July 1995, University of Illinois at Urbana-Champaign, 1995.

[2] *Corno, F., Prinetto, P., Rebaudengo, M. and Reorda, M. S.*: Exploiting Competing Subpopulations for Automatic Generation of Test Sequences for Digital Circuits. in Voigt, H.-M. (ed.): Parallel Problem Solving from Nature - PPSN IV: International Conference on Evolutionary Computation. volume 1141 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, pp. 792-800, 1996.

[3] *Grochtmann, M., and Wegener, J.*: Test Case Design Using Classification Trees and the Classification-Tree Editor CTE. Proceedings of Quality Week '95, San Francisco, USA, 1995.

[4] *Mathworks, The*: Matlab - UserGuide. Natick, Mass.: The Mathworks, Inc., 1994-1999. http://www.mathworks.com/

[5] *Mühlenbein, H., Schomisch, M. and Born, J.*: The parallel genetic algorithm as function optimizer. Parallel Computing, 17, pp.619-632, 1991.

[6] *Pohlheim, H.*: GEATbx - Genetic and Evolutionary Algorithm Toolbox for Matlab. http://www.geatbx.com/, 1994-2001.

[7] *Pohlheim, H.*: Evolutionäre Algorithmen - Verfahren, Operatoren, Hinweise aus der Praxis. Berlin, Heidelberg: Springer-Verlag, 1999. http://www.pohlheim.com/eavoh/

[8] *Pohlheim, H, Wegener, J., Sthamer, H.*: Testing the Temporal Behavior of Real-Time Engine Control Software Modules using Extended Evolutionary Algorithms. in Computational Intelligence im industriellen Einsatz, VDI-Berichte 1526, Düsseldorf: VDI-Verlag, pp. 61-66, 2000.

[9] *Schlierkamp-Voosen, D., Mühlenbein, H.*: Strategy adaptation by competing subpopulations. In *Davidor, Y., Schwefel, H.-P. and Männer, R.*: Parallel Problem Solving from Nature – PPSN III: International Conference on Evolutionary Computation. Vol. 866 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, pp. 199-208, 1994.

[10] *Schlierkamp-Voosen, D., Mühlenbein, H.*: Adaptation of Population Sizes by Competing Subpopulations. In Proceedings of the 1996 IEEE Conference on Evolutionary Computation, Piscataway, New Jersey, USA: IEEE Press, pp. 330-335, 1996.

[11] Tanese, R.: Parallel Genetic Algorithm for a Hypercube. In Grefenstette, J. J. (ed.): Proceedings of the Second International Conference on Genetic Algorithms and their Application, Hillsdale, New Jersey, USA: Lawrence Erlbaum Associates, pp. 177-183, 1987.

[12] *Wegener, J., Baresel, A., Sthamer, H.*: Evolutionary Test Environment for Automatic Structural Testing. Special Issue of Information and Software Technology devoted to the Application of Metaheuristic Algorithms to Problems in Software Engineering, 2001.

[13] *Wegener, J., and Grochtmann, M.*: Verifying Timing Constraints of Real-Time Systems by Means of Evolutionary Testing. Real-Time Systems, 15, pp. 275-298, 1998.

Pohlheim, H.: Competition and Cooperation in Extended Evolutionary Algorithms. In *Spector, L. (ed.)*: GECCO'2001 - Proceedings of the Genetic and Evolutionary Computation Conference – Late Breaking Papers. San Francisco, CA: Morgan Kaufmann Publishers, 2001.