Towards a Generally Applicable Self-adapting Hybridization of Evolutionary Algorithms

Wilfried Jakob[1], Christian Blume[2], Georg Bretthauer[1]

[1] Forschungszentrum Karlsruhe, Institute for Applied Computer Science, Postfach 3640, 76021 Karlsruhe, Germany
{wilfried.jakob, georg.bretthauer}@iai.fzk.de
[2] University of Applied Sciences, Cologne, Campus Gummersbach, Am Sandberg 1, 51643 Gummersbach, Germany
blume@gm.fh-koeln.de

Abstract. When applied to real-world problems, the powerful optimization tool of Evolutionary Algorithms frequently turns out to be too time-consuming due to elaborate fitness calculations that are often based on run-time-intensive simulations. Incorporating domain-specific knowledge by problem-tailored heuristics or local searchers is a commonly used solution, but turns the generally applicable Evolutionary Algorithm into a problem-specific tool. The new method of hybridization implemented in HyGLEAM is aimed at overcoming this limitation and getting the best of both algorithm classes: A fast, globally searching, and robust procedure that preserves the convergence reliability of evolutionary search. Extensive tests demonstrate the superiority of the approach, but also show a drawback: No common parameterization can be drawn from the experiments. As a solution, a new concept of a self-adapting hybrid is introduced. It is stressed that the methods presented can be applied to Evolutionary Algorithms other than the one used here with no or minor modifications being required only.

1 Introduction

The Evolutionary Algorithm (EA) GLEAM (General Learning Evolutionary Algorithm and Method) introduced by Blume [1] was successfully applied to a broad range of real-world applications in the last fourteen years: Collision-free robot path planning and program development [1,2], scheduling [3], resource optimization [4] or design optimization [5,6,7] to mention only the most important. A common experience gained from these applications is that the time available for identifying a solution always is too short. Often, applications were possible only, because non-optimal solutions obtained in the given time frame were better than the state of the art and, thus, an improvement. But it is not satisfying to win only because common practice is weak enough so that your half-way finished solutions are performing better. Even in the area of design optimization, where it is often no problem to do runs over night, more sophisticated and precise simulation models require so much computing time that again there is a demand for fewer fitness calculations [7]. The goal of a reduced number of evaluations is usually achieved by hybridization with problem-specific knowledge or heuristics ([8], most papers reporting on practical applications in last PPSN, ICGA, GECCO etc. conferences). This partly speeds up the performance drastically, but always results in a more or less problem-specific solution.

HyGLEAM (Hybrid GeneraL-purpose Evolutionary Algorithm and Method) is aimed at overcoming this limitation by improving the performance of the evolutionary search, while maintaining the robustness, global nature of the search, and general applicability of EAs in the resulting hybrid algorithm. It is especially intended for problems where only little or no information is available about the search space. The concept of HyGLEAM [9] described in section 2 was published in 2002 together with a brief overview and comparison of all thirteen kinds of hybridization to which the method was applied [10], see sections 3 and 4. This paper focuses on the work to extract a common parameterization for the best basic kind of hybridization. The results presented in section 5 lead to the consequence of self-adaptation. This new concept is described in section 6 as an outlook for future work.

2 HyGLEAM Concept

The basic idea of the HyGLEAM concept can be summarized in two points:

1. Use of generally applicable local search algorithms instead of the commonly used problem-specific ones for hybridization.
2. Use of a convergence-dependent control mechanism for distributing the computational power between the basic algorithms for suitable kinds of hybridization.

The first point may appear to be simple, but it is a matter of fact that nearly all real-world applications and investigations are based on problem-specific local searchers. Appropriate local search algorithms for parameter optimization must be derivative-free and able to handle restrictions in order to be generally applicable.

For the second point, means are required to measure the convergence of a population. This can be done on the basis of observing the establishment of sub-populations (niches) of similar individuals in the course of generations. To estimate the genotypic diversity, distance measures for chromosomes are required, which quantify the *parameter distance*, the different *gene ordering* for combinatorial problems, and the *common genes* in the case of dynamic chromosomes as required by some applications, see section 4.2 or [2,7]. Distance functions reported in literature [11,12] often are too specialized for the problem on hand to serve as a solution here. The measures used are independent of the application in so far as they are not influenced by actual parameter ranges or the number of genes. Details are omitted here due to the lack of space, see [9,10] and the web page: www.iai.fzk.de/~jakob/HyGLEAM/.

For estimating the convergence of a population, the individuals are assumed to be in linear circular order so that they all have a neighborhood. A niche is formed by adjacent individuals with a greater fitness than half of the global best and with a distance Δ being smaller than the strategy parameter ε. Chromosomes of less quality are ignored, as they do not contribute to niching of individuals of high fitness. Niches, whose center individuals differ by less than ε, are merged regardless of their position. The resulting amount of niches N and the maximum difference of their center individuals $\Delta_{N,max}$ are compared to the strategy parameters N_{max} and ε_{Pop}. The population is considered to be *converged*, if:

$$\Delta_{N,\max} \le \varepsilon_{Pop} \quad \text{and} \qquad N \le N_{\max} \tag{1}$$

It is sufficient to check the population for convergence only, if a certain amount of generations without improvement or, stronger, without acceptance of any offspring occurs. As the formation of niches is common to EAs, this control method can be adapted easily to other EAs.

The following four general hybridization alternatives were investigated:

1. Pre-optimization of the start population
   It can be applied to the entire population or a fraction of it. It provides the evolution with solutions of more or less good quality to start with.
2. Post-optimization of the EA results
   As EAs are known to converge slowly, an improvement can be expected by stopping the evolution after approaching the area of attraction of the (global) optimum and leaving the rest to the local search. The appropriate switching point is determined by the control procedure introduced above.
3. Direct integration
   Optimizing every or the best offspring of one mating only causes the EA to operate over the peaks of the fitness landscape exclusively rather than to treat the valleys and slopes, too. The offspring's genotype can be updated (Lamarckian evolution) or left unchanged (Baldwinian evolution). As both methods which are usually applied to domain-specific local searchers are controversially discussed in literature [14,15], this was also investigated. In accordance with a recommendation, updating 5% of the accepted offspring [16] was examined.
4. Delayed direct integration
   This is a variant of direct integration, where the evolution works on its own until a certain convergence of the population has occurred.
   Pre-optimization can be combined with the other methods, while a fusion of direct integration and post-optimization does not appear to be reasonable.

3 Basic Algorithms of the Test Implementation

For the test implementation that was needed to evaluate the approach introduced, a representative EA and local searchers which meet the above-mentioned requirements must be chosen. As Schwefel [17] gives a detailed description of the selected local procedures together with experimental results, they are explained here only briefly.

3.1 Rosenbrock Algorithm

Rosenbrock [18] modified the well-known coordinate strategy by rotating the coordinate system so that it points in the direction that appears to be most favorable. For this purpose, the experience of failures and successes is gathered in the course of the iterations. The remaining directions are fixed to be normal to the first one and mutually orthogonal. A direct search is done parallel to the axes of the rotated coordinate system. The procedure stops when the rate of changes of the main search direction decreases below a certain value and when the distances covered become too small. The algorithm can be controlled by this strategy parameter. The implementation on hand [19] uses normalized object parameters in the range between zero and one, thus allowing the definition of application independent threshold values for the termination. In the experiments, the five values of 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}, and 10^{-9} are compared.

3.2 Complex Procedure

The Complex method of Box [20] is based on the SIMPLEX strategy of Nelder and Mead [21], which was improved to handle constraints (COnstrained siMPLEX). The idea is to use a polyhedron of n+1 to 2n vertices (n is the number of dimensions), whose worst vertex is reflected at the midpoint of the remaining vertices. The resulting line is lengthened by a factor of 1.3, thus expanding the polyhedron. If this leads to an improvement, the worst vertex is replaced by the new one, otherwise the polyhedron is contracted. The algorithm stops when no improvement is achieved in five consecutive iterations. The stopping criterion of this procedure is left fixed in the implementation used [19]. As this method can exploit multiple start points, it is assumed to be better suited for post-optimization using a set of EA results of more or less good quality for forming one start complex instead of just optimizing the solutions in an isolated manner.

3.2 GLEAM

GLEAM is an Evolutionary Algorithm of its own that combines elements from Evolution Strategy (ES) and real-coded Genetic Algorithms (GA) with data structuring concepts from computer science. The coding is based on chromosomes consisting of problem-configurable gene types. The definition of a gene type constitutes its set of real, integer or Boolean parameters together with their ranges of values. There are different rules for constructing chromosomes from genes. If the chromosomes are of fixed length, exactly one gene of each gene type will go into the chromosome. Additionally, the order of genes can be defined to be relevant or not, depending on the application. This mainly affects the set of applicable mutation operators. If the problem requires a dynamic set of parameters, as the robot program development task described in section 4.2 or some design optimization problems [7], the rule for dynamic chromosomes applies: Any number of genes, including zero, of every gene type can occur in a chromosome. This provides the user with a flexible mechanism of naturally mapping his problem to the chromosomes and genes, often resulting in genotypes from which phenotypic properties can be derived easily. Contrary to the binary coding of classical GAs, no artificial solutions are needed. Another advantage of this approach is that problem-specific genetic operators can be added easily to the set of general ones, if desired.

The type definition of genes allows for mutation operators that take explicit restrictions into account. The relative mutation operator is inspired by the ES: At first, it is randomly chosen whether to increase or decrease the actual value. Then, the range of possible alteration with respect to the parameter boundaries is calculated. This range is divided into ten equidistant classes, from which one is chosen by chance to define the actual range of mutation, from which the value of modification is selected randomly. This is faster than calculating the normal distribution, as done with ES, but has a similar effect: Greater changes are less likely than smaller ones. In contrast to ES, however, there are no strategy parameters for mutation changed by the evolutionary process. Mutation is treated in a more general way as common with GAs or ES: Genes may be substituted by newly generated ones or deleted, moved or replicated in case of dynamic chromosomes. Additionally, there are evolvable sub-structures within a gene, the so-called segments. They are subject to some sort of macro mutation and they serve as bases of crossover operators which are similar to those of traditional GAs: Single and n-point crossover operators work on these segment boundaries. A more detailed description of GLEAM is given in [13].

4 Test Cases and Results for All Kinds of Hybridization

Appropriate test cases must be representative of real-world applications, their calculation must be comparatively fast for statistical investigations, and the exact or an acceptable solution must be known. Two series of test runs were performed, with the shorter second one being aimed at trying to identify suitable general parameter settings for the best kind of hybridization found in the first one. The first series consisted of five test functions taken from GENEsYs [22] and three real-world problems, which are described very briefly only due to the lack of space. The interested reader is referred to the given literature. The three test functions for the second series were taken from Schwefel [17].

4.1 Benchmark Functions

Table 1 shows characteristic properties of the test functions. The sphere function is known to be easy for ES, but hard for ordinary GA, while Shekel’s foxholes and the Rastrigin Function are hard for ES and local searchers. The test function of Fletcher & Powel is of considerable complexity, while the fractal function with its unknown optimum is really hard. The functions of Bracken and Beale were chosen because of their implicit restrictions. The helical valley is another example of a unimodal function of considerable complexity.

4.2 Real-world Problems

As test functions are just a substitute of real applications, a significant evaluation of a new concept should always include some real-world applications. From the range of applications of GLEAM, three tasks were selected, which fulfill the requirements for use as test benches: Being fast enough for allowing multiple runs and complex enough for leaving room for improvements. As another criterion, they are to cover different areas of applications and problem properties, see Table 2. All tasks were multi-objective optimization using the weighted sum for fitness calculation. Penalty functions were applied if necessary. For the last two problems, the local searchers work on the parameters only and leave the combinatorial aspects to GLEAM.

Table 1. Important properties of the test functions (gray: functions of the 2nd test series)

Test Functions	Number of Parameters	Modality	Restrictions	Range	Target Value
Sphere [22, f1]	30	uni	no	$[-10^{10}, 10^{10}]$	$f_{opt} = 0$ $f_{target} = 0.01$
Shekel's foxholes [22, f5]	2	multi	no	[-500, 500]	$f_{opt} = f_{target} = 0.998004$
Gen. Rastrigin function [22, f7]	20	multi	no	[-5.12, 5.12]	$f_{opt} = 0$ $f_{target} = 0.0001$
Fletcher & Powel [22, f16]	5	multi	no	[-3.14, 3.14]	$f_{opt} = 0$ $f_{target} = 0.00001$
Fractal function [22 , f13]	20	multi	no	[-5, 5]	f_{opt} is unknown $f_{target} = -0.05$
Bracken et al. [17, f2.32]	2	multi	yes, active	$0 \le x_1 \le 1.0$ $0 \le x_2 \le 1.25$	$f_{opt} = -2$ $f_{target} = -1.999999$
Beale, [17, f2.29]	3	uni	yes	$0 \le x_{1,2} \le 3.0$ $0 \le x_3 \le 1.5$	$f_{opt} = 1/9$ $f_{target} = 0.111112$
Helical valley (Fletcher & Powel) [17, f2.34]	3	uni	no	$0.1 \le x_1 \le 100$ $-2.5 \le x_2 \le 100$ $-100 \le x_3 \le 7.5$	$f_{opt} = 0$ $f_{target} = 0.000001$

Table 2. Important properties of the real-world problems

Test Cases	Combinat. Opt.	Parameters	Modality	Restrictions
Design Optimization	no	3 real	multi	no
Scheduling & Resource Opt.	(yes)	87 int	multi	yes
Robot Path Planning	yes	dynamic, mixed	multi	yes

Design Optimization. A micro-optical communication device shall be designed in such a way that despite of fabrication tolerances within a given range, the reject rate is as low as possible. Consideration of fabrication tolerances turns the relatively easy optical problem into an interesting, highly multimodal optimization task. It depends on three parameters and yields four criteria, three technical properties to be fulfilled, and the reject rate to be minimized. The target fitness corresponds to a fulfillment of the technical properties and a reject rate of less than 9.5% [6].

Scheduling and Resource Optimization. This task from chemical industry [4] deals with 87 batches with varying numbers of workers being required during the different phases of each batch. The objective of scheduling these batches means a maximum reduction of the maximum number of workers per shift (human resource) and production time. Restrictions like due dates of batches, necessary pre-products from other batches, and the availability of shared equipment must also be observed. Allocation conflicts are solved by the sequence of the batches within a chromosome. As it can be overwritten by suitable changes of the starting times, however, the combinatorial aspect is limited to solving allocation conflicts (bracketed "yes" in Table 2). The optimum is not known in this case. For the test, a reduction from 12 to 9 workers per shift and to 70% of the original production time given in manual schedules was regarded sufficient. This is close to the best reported solution [4].

Collision-free Robot Path Planning. By controlling the robot axis motors, an industrial robot shall be moved along a line as straightly and fast as possible from a starting to a destination point with collisions with itself and some obstacles being avoided, see [1,2]. The motor parameters are real values. Due to a command that tells the robot control to keep the actual motor settings for a specified amount of control cycles, however, this task has both integer and real parameters. As the number of necessary motor commands is not known in advance, the chromosomes must be of dynamic length and the order of the commands is essential. The chromosomes consist of commands for switching on motors at a given angular velocity and ramp and switching them off using a ramp together with the already mentioned command for keeping the actual motor settings. The two main objectives are to minimize the deviations from the target point and from the straight line. Further criteria are a reduction of the duration of the move and the amount of commands. Collisions are dealt with by a penalty function.

4.3 Results for All Kinds of Hybridization

A basic algorithm or a hybridization together with a setting of its strategy parameters (see Table 3) is called a *job*. For the test functions, the results are based on 100 runs per job. Due to the greater computational effort required for the real-world applications, except of the jobs of the basic procedures and some of the "fast" hybrids, this number was reduced to 50. The comparisons shown in Figures 1 and 2 were made on the basis of the effort measured by the average number of evaluations needed for success. If not stated otherwise, only successful jobs (jobs where all runs reach the given target values) were taken into account for comparison. They were measured by the improvements in effort achieved compared to the best GLEAM job. Table 3 shows the acronyms for the algorithms and their parameterizations.

Fig. 1 gives an overview of the best runs of all thirteen kinds of hybridizations on the basis of the improvements achieved compared to the best GLEAM job. The sphere function is not included, because GLEAM had no success and therefore did not produce any reference value for comparison. It can be stated, however, that post-optimization and (delayed) direct integration yield very good results compared to GLEAM. The robot task was not covered by this comparison, as there was no improvement worth mentioning. This clearly demonstrates the limitation of the approach of locally optimizing parameters: If the combinatorial aspect dominates, the additional effort for local improvements is too costly. For the Foxhole and the Rastrigin functions, no improvement could be achieved. These two functions turned out to be too easy for GLEAM to leave room for any enhancements. Fig. 1 obviously shows the superiority of (delayed) direct integration. This type of hybridization reached the aim of increasing the convergence velocity while preserving, or in case of the sphere function, increasing the convergence reliability. Together with the observation that no suitable common setting for all strategy parameters could be derived from the experiments, this was the essential result of the first test series.

Table 3. Acronyms, basic algorithms, hybrids, and strategy parameters

Acronym	Algorithm	Strategy Parameters
G	GLEAM	Population size p =5,10,20,30,50,70,90,120,…
R	Rosenbrock	Termination thresholds, see section 3.1
C	Complex	None
GiR, GiC	GLEAM with Rosenbrock/ Complex pre-optimization	Percentage of pre-optimized individuals for the initial population.
GpR	GLEAM with Rosenbrock post-opt.	The niching is controlled by three parameter settings for ε and ε_{Pop}, P1=(0.005, 0.01), P2=(0.002, 0.005), P3=(0.001, 0.002), and by N_{max} which varies between 2 (p<20) and 5 (p>90). The tests are done after 3 generations without local improvement.
Gp1S	GLEAM with Complex post-opt., each EA-result is 1 start point (1S)	
Gp1C	GLEAM with Complex post-opt., all EA-results: 1 start complex (1C)	
GR, GC	Direct integration of Rosenbrock/Complex	Lamarck-rate (0, 5, 100%), optimization of all or the best of the descendants of one mating.
GdR, GdC	Delayed direct integration of Rosenbrock/Complex	Like post-optimization, but testing for every generation w/o local improvement

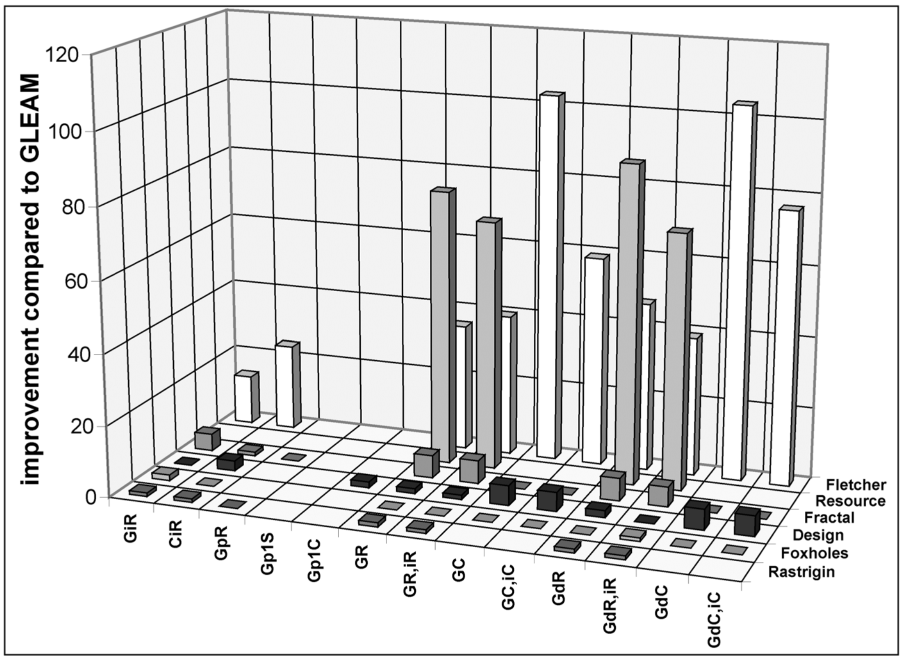

Fig. 1. Comparison of the thirteen kinds of hybridization investigated. Empty fields indicate an insufficient success rate (below 100%), while flat fields denote the fulfilling of the optimization task, but with greater effort than GLEAM. Improvement is calculated on the basis of the best GLEAM job, see section 4.3. Combinations of (delayed) direct integration and pre-optimization are abbreviated by, GR,iR, for example, in case of pre-optimized direct integration using the Rosenbrock procedure. For Gp1S, a 100 % success rate was not reached in any test case

5 Extended Results for the Direct Integration

To find out more about a suitable common parameterization, additional runs were performed for both versions of direct integration. Besides the three last test functions of Table 1, variants of Shekel's foxholes and the Rastrigin function were used, which were rotated by 30°. This turned out to be much harder for GLEAM, because the tasks did no longer have any regularities with respect to the coordinate axes. Evolutionary Algorithms like the ES, GAs or GLEAM work with mutations that tend to alter at once a small amount of parameters only. Consequently, a search is carried out more or less parallel to the coordinate axes frequently resulting in a fast solution of the two original functions.

The results of all suited test cases for all types of direct integration are compared in Fig. 2. Integration of the Rosenbrock procedure always works, while the Complex algorithm yields better results in many cases, but only if it works at all. Thus, integration of the Rosenbrock procedure is more reliable, but less promising compared to the Complex algorithm. The delayed version often yields better results. Table 4 shows the parameters for the best runs and the evaluations required together with the evaluations of the best GLEAM job. The second best job often is only a little worse as shown in both figures. From the entire material, only some parameters can be narrowed down or fixed: The population size should not be too small to avoid premature convergence, so a size between 10 and 50 can be recommended depending on the estimated complexity of the problem on hand. Optimization of the best offspring of one mating instead of all and Lamarckian evolution can be fixed, while the termination threshold of the Rosenbrock algorithm and the niching parameterization are still an open question.

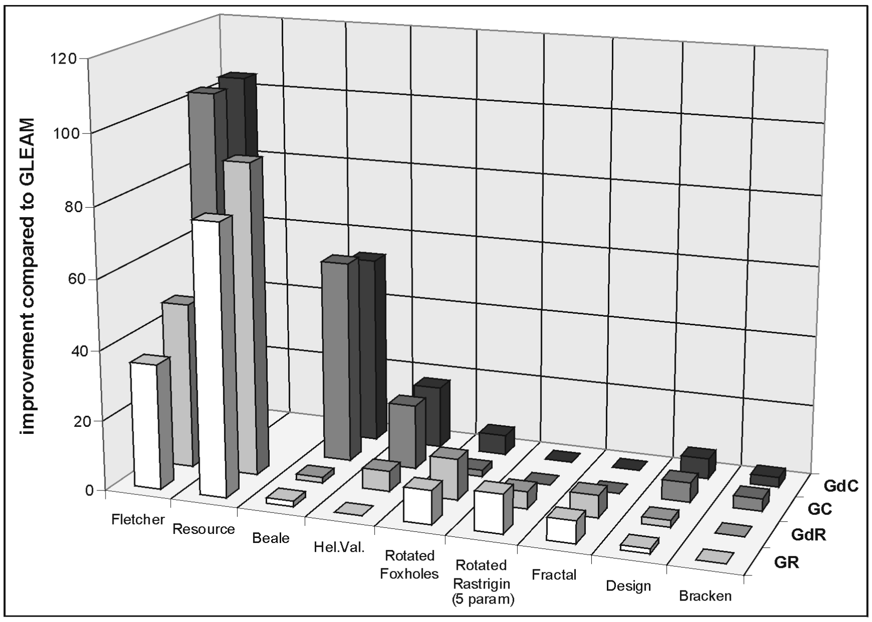

Fig. 2. Overall comparison of all types of direct integration. Explanations see Fig. 1

Table 4. Evaluations required and strategy parameters (see also Table 3) of the runs of Fig. 2

Test case	GLEAM evaluations	Hybrid	Evaluations	Population size	Rosenbrock threshold	Opt. of best or all offspring	Lamarck rate	Niching parameter
Fletcher	483,566	GdC	4,635	5		best	100 %	P3
Resource	5,376,334	GdR	60,225	5	10^{-2}	best	100 %	P1
Beale	76,229	GC	1,319	5		best	100 %	
Hel. Valey	47,288	GC	2,576	5		best	100 %	
Rot. Fox.	103,195	GC	8,831	20		best	0 %	
Rot. Rast.	3,518,702	GR	315,715	70	10^{-2}	all	100 %	
Fractal	195,129	GdR	30,473	5	10^{-2}	best	100 %	P2
Design	5,773	GdC	996	5		best	100 %	P3
Bracken	5,000	GC	1,433	5		best	100 %	

6 A Concept of Adaptive Direct Integration

As pointed out in the previous section, (delayed) direct integration of the local searchers in the offspring production (memetic part of HyGLEAM) yielded superior results compared to GLEAM. The delayed variant was an attempt to reduce the effort by a better balance between global and local search. For a further step in this direction, it is assumed that local search may be inaccurate to some extent in the beginning and the effort spent for the exact determination of local optima rejected later on can be saved. Thus, local search shall start coarsely and become more precise as the search goes on. As neither the optimum nor the path to it is known in advance, the process of becoming more precise should be adjusted adaptively instead of using fixed schedules as proposed by Zitzler et al. [23]. This also holds for the choice between Rosenbrock and Complex algorithm.

For the open parameters, an adaptive mechanism based on the fitness gain obtained by local search and the effort spent is proposed. It is described for the case of scheduling the two local searchers (LS). Initially, both LS have the same probability of being selected. The fitness gain *fg* and the required evaluations *eval* are summed up, and the applications of each LS are counted. It was one result of the experiments that between 50 and 200 matings were needed for the (delayed) direct integration with fixed iteration limits. This value is supposed to increase for a varying precision of ocal search. Adjustment of the application probabilities of the LS is done, if either each LS was used three times at minimum or there were 15 matings in total. The new relation between Complex and Rosenbrock is calculated as follows:

$$\frac{\sum fg_{i,compl}}{\sum eval_{i,compl}} : \frac{\sum fg_{j,rosen}}{\sum eval_{j,rosen}} \qquad (2)$$

After the adjustment, the sums are reset to zero so that the adaptation is faster. If the relation for one LS is worse than 1:10 for 3 consecutive alterations, it is ignored from that on. This approach can be extended easily for the adjustment of the following parameters as well as for more local searchers, if desired. Both LS can be controlled by the permissible number of evaluations per run and the Rosenbrock procedure additionally by the termination threshold. The five values from the experiments will be extended to nine values ranging from 10^{-1} to 10^{-9}. Based on the experiences gained from the experiments, the number of permissible evaluations shall range from 100 to 1500 using a step size of 100 in the beginning and 200 in the end, resulting in about 10 levels. Only three of these possible levels are active at the same time and the first three lowest ones initially have the same probability. If the lowest or the highest active level receives a probability of more than 50%, the next lower or higher level is added. The one at the opposite end is dropped and its likeliness is added to its neighbor. The new level receives 20% from the two others. As a result the three consecutive levels are moved along the scale of possible ones according to their performance determined by the achieved fitness gain and the required evaluations. To raise the quality of local search at the end of the run, the algorithms for niching control introduced in section 2 can be used to readjust the threshold values for the level movements: The threshold for a level decrease can be raised to 60% for example and the one for an increase to 40%.

This concept allows for an adaptive control of the schedule of the local searchers as well as of the intensity of their search, which is based on fitness gain and effort. Cumbersome parameterization of the hybrid and its adjustment to the problem on hand, which will not be possible in many practical applications due to time and effort restrictions, are no longer required.

7 Conclusion and Outlook

The HyGLEAM concept of hybridization between an EA and local searchers for parameter optimization has been presented. Extensive tests with a total of eight benchmark functions and three real-world applications show the superiority of the approach: About 90 and 100 times less evaluations were required for the two best test cases (scheduling with resource optimization, Fletcher's function). The number of evaluations needed was reduced by about 6 times in the "worst" test application (design optimization). Convergence reliability of the EA was preserved and in one case even improved. As no common setting for all parameters of the best hybridization could be derived, the concept of an *adaptive direct integration* was introduced, which will be subject of further work.

References

1. Blume, C.: GLEAM - A System for Intuitive Learning. In: Schwefel, H.P., Männer, R. (eds): Conf. Proc. of PPSN I. LNCS 496, Springer Verlag, Berlin (1990) 48-54
2. Blume, C.: Optimized Collision Free Robot Move Statement Generation by the Evolutionary Software GLEAM. In: Cagnoni, S. et al. (eds): Real-World Applications of Evolutionary Computing. Proc. of EvoWorkshops 2000. Springer Verlag, Berlin (2000) 327-338

3. Blume, C., Jakob, W.: Cutting Down Production Costs by a New Optimization Method. In: Proc. of the Japan - U.S.A. Symposium on Flexible Automation. ASME (1994)
4. Blume, C., Gerbe, M.: Deutliche Senkung der Produktionskosten durch Optimierung des Ressourceneinsatzes. atp 36, 5/94, Oldenbourg Verlag, München (in German) (1994) 25-29
5. Jakob, W., Meinzer, S., Quinte, A., Süß, W., Gorges-Schleuter, M., Eggert, H.: Partial Automated Design Optimization Based on Adaptive Search Techniques. In: Parmee, I.C. (ed): Adapt. Comp. in Engineering Design and Control '96. Univ. of Plymouth (1996) 236-241
6. Sieber, I. Eggert, H., Guth, H., Jakob, W.: Design Simulation and Optimization of Microoptical Components. In: Bell, K.D. et al. (eds): Proceedings of Novel Optical Systems and Large-Aperture Imaging. SPIE Vol.3430 (1998) 138-149
7. Quinte, A., Jakob, W., Scherer, K.-P., Eggert, H: Optimization of a Micro Actuator Plate Using Evolutionary Algorithms and Simulation Based on Discrete Element Methods. In: Laudon, M.. Romanowicz, B. (eds): Proc. of the 5th Int. Conf. on Modeling and Simulation of Microsystems. Computational Publications, Boston, Massachusetts (2002) 194-197
8. Davis, L. (ed): Handbook of Genetic Algorithms. Van Nostrand Reinhold, New Y. (1991)
9. Jakob, W.: Eine neue Methodik zur Erhöhung der Leistungsfähigkeit Evolutionärer Algorithmen durch die Integration lokaler Suchverfahren. Doctoral thesis, FZKA 6965, University of Karlsruhe (in German) (2004), see also: www.iai.fzk.de/~jakob/HyGLEAM/
10. Jakob, W.: HyGLEAM – An Approach to Generally Applicable Hybridization of Evolutionary Algorithms. In: Merelo, J.J., et al. (eds): Conf. Proc. PPSN VII. LNCS 2439, Springer Verlag, Berlin (2002) 527-536
11. Ronald, S.: Distance Functions for Order Based Encodings. In: Fogel, D. (ed): Conf. Proc. CEC 1997. IEEE Press, New York (1997) 46-54
12. Tagawa, K., Kanzaki, Y., Okada, D., Inoue, K., Haneda, H.. A New Metric Function and Harmonic Crossover for Symmetric and Asymmetric Traveling Salesman Problems. In: Conf. Proc. CEC 1998. IEEE Press, New York (1998) 822-827
13. Blume, C., Jakob, W.: GLEAM – An Evolutionary Algorithm for Planning and Control Based on Evolution Strategy. In: Cantú-Paz, E. (ed): GECCO – 2002, Vol. Late Breaking Papers (2002) 31-38
14. Gruau, F., Whitley, D.: Adding Learning to the Cellular Development of Neural Networks: Evolution and the Baldwin Effect. Evolutionary Computation, Vol.1 (1993) 213-233
15. Whitley, D., Gordon, V., Mathias, K.: Lamarckian Evolution, The Baldwin Effect and Function Optimization. In: Davidor, Y. et al. (eds): Conf. Proc. PPSN III, LNCS 866, Springer Verlag, Berlin (1994) 6-14
16. Orvosh, D., Davis, L.: Shall We Repair? Genetic Algorithms, Combinatorial Optimization, and Feasibility Constraints. In: Forrest, S. (ed): Conf. Proc. Fifth ICGA, Morgan Kaufmann, San Mateo, California (1993) 650
17. Schwefel, H.-P.: Evolution and Optimum Seeking. John Wiley & Sons, New York (1995)
18. Rosenbrock, H.H.: An Automatic Method for Finding the Greatest or Least Value of a Function. The Computer Journal, 3 (1960) 175-184
19. Peters, D., Bolte, H., Böhnke, R., Bischoff, L., Laur, R.: MODOS - Model Based Design Optimization System. In: IEEE Proceedings of the 43rd Midwest Symposium on Circuits and Systems, Lansing, Michigan (2000)
20. Box, M.J.: A New Method of Constrained Optimization and a Comparison with Other Methods. The Computer Journal, 8 (1965) 42-52
21. Nelder, J.A., Mead, R.: A Simplex Method for Function Minimization. The Computer Journal, 7 (1965) 308-313
22. Bäck, T.: GENEsYs 1.0, University of Dortmund (1992) ftp://lumpi.informatik.uni-dortmund.de/pub/GA/
23. Zitzler, E., Teich, J., Bhattacharyya, S.S.: Optimizing the Efficiency of Parameterized Local Search within Global Search: A Preliminary Study. In: Conf. Proc. CEC 2000, IEEE press, Piscataway, N.J. (2000) 365-372