HyGLEAM:
Hybrid GeneraL purpose Evolutionary Algorithm and Method

Wilfried Jakob
Institute for Applied Computer Science, Forschungszentrum Karlsruhe
D-76021 Karlsruhe, Germany, P.O.Box 3640
Email: jakob@iai.fzk.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 expensive fitness calculations often based on run-time-intensive simulations. Incorporating domain-specific knowledge by problem-tailored heuristics is a commonly used solution, but results in a problem-specific tool. This article describes the approach of combining the Evolutionary Algorithm GLEAM with general local search strategies to obtain the best of both procedures by avoiding their drawbacks: HyGLEAM, a robust, but never the less fast "general-purpose" optimization tool. The methods introduced can be applied to other Evolutionary Algorithms with minor modifications being required only. First experiments with test functions and a real world design optimization problem produced promising results.

Keywords: Optimization, planning, Evolutionary Algorithms, simulation-based optimization, "general-purpose" optimization.

1. INTRODUCTION

The Evolutionary Algorithm (EA) GLEAM (General Learning Evolutionary Algorithm and Method, formerly Genetic Learning Algorithm and Method) introduced by Blume [1] was successfully applied to a broad range of real world applications in the last ten years: Collision-free robot path planning and program development [1,2], task planning and learning [3], scheduling [4], resource optimization [5], design optimization [6, 7] or traverse path minimization in a concrete precasting plant [8] to mention only the most important.

One common experience gained from these applications is that the time available for identifying a solution is always 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 [9]. The goal of a reduced number of evaluations is usually achieved by hybridization with problem-specific knowledge or heuristics ([10], most practitioner papers 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 is aimed at overcoming this limitation by improving the performance of the evolutionary search, while maintaining the robustness, the global search, and the general applicability of EAs in the resulting hybrid. In Section 2, GLEAM and the local search strategies selected for hybridization shall be introduced briefly. The third Section shall describe different methods of combining the two classes of algorithms and introduce a new method of estimating good points of time for switching from global to local search or adding local search to the evolution. Experimental results shall be given in Section 4. The conclusions shall be drawn in Section 5, where an outlook shall be given as well.

2. GLEAM AND TWO LOCAL SEARCH ALGORITHMS

GLEAM

GLEAM is an Evolutionary Algorithm of its own. It combines elements from classic Genetic Algorithms (GA) and Evolution Strategy (ES) 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, dependent on the application. This affects mainly the set of applicable mutation operators. If the problem requires a dynamic set of parameters, as the already mentioned robot program development task or some design optimization problems do, the rule for dynamic chromosomes applies: Any number of genes, including zero, of every gene type can occur in a chromosome. Sub-structures of chromosomes within a chromosome are also possible. 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. No artificial solutions are necessary contrary to the binary coding of classical GAs. 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 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 and then, the range of possible alteration 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. Mutation is treated in a more general way as common with GAs or ES: Genes may be substituted by newly generated

This paper was first published in: N. Callaos, S. Esquivel, J. Burge (eds.): World Multiconference on Systematics Cybernetics and Informatics (SCI 2001), IIIS and IEEE, Venezuela, Vol.3, S.187-192, 2001.

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 form the bases for the crossover operators, which are similar to those of traditional GAs: Single and n-point crossover operators work on these segment boundaries.

GLEAM uses the model of structured populations and local selection introduced by Gorges-Schleuter [11]. It can be applied easily to other types of EAs like ES [12]. Every individual of the population is placed on a ring and has its neighborhood of a fixed number of individuals on its right and on its left, called its deme. Reproduction takes place within these demes only. As the demes of nearby individuals are overlapping, the information can spread through the entire population, the velocity of propagation, however being smaller than in case of panmictic populations. Thus, establishing of niches is much more likely and in a later stage of evolution, the niches begin to merge and evolve the best solution. This approach maintains diversity within a population for a longer time and produces good results in preventing premature convergence.

Local Search Algorithms

Suitable local search algorithms must be derivation-free and able to handle restrictions in order to preserve the general applicability of the resulting hybrid. Rosenbrock's algorithm [13] was chosen, because it meets these requirements and is known to be a powerful local search procedure using one start point. Rosenbrock 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.

As EAs can deliver several results of more or less good quality, a local searcher which can exploit multiple start points can be expected to be useful, too. Thus, the COMPLEX method of Box [14] represents another candidate for hybridization. This method is based on the SIMPLEX strategy of Nelder and Mead [15], which was enhanced by Box such that it can handle constraints (COnstraind 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 resulting in an expansion of 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. Schwefel gives a detailed description of the Rosenbrock algorithm and the COMPLEX method together with experimental results [16].

3. HYBRID GLEAM

Methods of Hybridization

For a generally applicable hybridization of an EA three general alternatives exist:

1. Initialization of the start population
   This provides the evolution with valid solutions of more or less good quality to start with.
2. Post-optimization of the EA results
   EAs are known to converge slowly. Thus, 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. But as Goldberg has pointed out, it is not easy to determine the appropriate switching point [17].
3. Direct integration
   It is also possible to locally optimize every or the best offspring of one mating only and select the best one for possible replacement of the parent. The offspring's genotype can be updated (Lamarckian evolution) or left unchanged (Baldwinian evolution). As both methods usually applied to domain-specific local searchers are controversially discussed in literature [18, 19], this will also be investigated. Orvosh and Davis recommend to update 5% of the accepted offsprings only [20].

Initialization can be combined with the other two methods, while a fusion of direct integration and post-optimization does not appear to be meaningful.

Estimation of Stagnation

Concerning real world problems neither the structure of the fitness landscape nor the optimum or its area of attraction are known in advance. But as computation of the fitness function frequently is time-consuming, it is possible to perform more sophisticated calculations to estimate when to switch from global to local search.

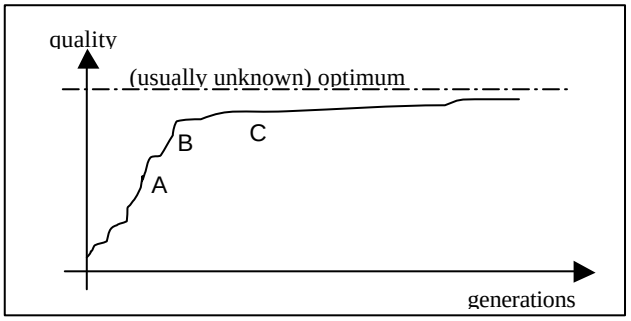

Figure 1: Typical Progress during the Run of an EA

Fig. 1 shows the typical progress of an EA run. Stagnation phases of the overall quality can be identified easily, e.g. A, B or C. But which one shall be selected for terminating the evolution? This cannot be derived from stagnation only. A better measure is the genotypic diversity within the population. If the population consists of a few genotypically different sub-populations (niches) only, which are of minor difference, then a stagnation can be expected, which provides little chance for greater progress. Hence, stagnation phases like in Fig. 1 may be used to trigger a check for niche constitution.

GLEAM with its demes allows for a more sophisticated stagnation indicator than counting generations without global progress. Generations without any local improvements within a deme (gdi) or, stronger, with no acceptance of an offspring in any deme (gda) can be counted instead.

To estimate the genotypic diversity, distance measures for chromosomes must be defined. Considering GLEAM with its universal chromosomes, measures for parameter distance, different gene ordering, and dynamic chromosomes are required. Dis-

tance functions reported in literature are often too much specialized to the problem on hand [21, 22], rather than to serve as a solution here. Measures should be independent of the application in so far as they must not be influenced by actual parameter ranges or the number of gene types and should be within a fixed range, e.g. between 0 and 1.

Different measures are defined for the three chromosome types of GLEAM:

- Fixed-length chromosomes with irrelevant gene order
- Fixed-length chromosomes with relevant gene order
- Variable-length chromosomes with relevant gene order

The value of each measure varies between 0 (identity) and 1 (maximal difference). The proofs of compliance with the four metric axioms are omitted here due to lack of space, but can be found on the following web page:

`http://www.iai.fzk.de/~jakob/hy_gleam/`

Fixed-length Chromosomes with Irrelevant Gene Order: This is the simplest chromosome type, for which the calculation of the parameter distance Δ_{par} is sufficient. It is defined as follows:

$$\Delta_{par}(C_1, C_2) = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|par_{i,1} - par_{i,2}\right|}{ub_i - lb_i} \qquad (3.1)$$

where C_j: chromosome j
n: number of all parameters of all genes
$par_{i,j}$: i-th parameter of chromosome j
ub_i: upper bound of the i-th parameter
lb_i: lower bound of the i-th parameter

Fixed-length Chromosomes with Relevant Gene Order: The positional distance $pd_{1,2}(G_i)$ of one gene G_i of two fixed-length chromosomes C_1 and C_2 is defined by their sequential numbering and the calculation of the absolute difference of their indices:

$$pd_{1,2}(G_i) = \left|Idx_1(G_i) - Idx_2(G_i)\right| \qquad (3.2)$$

where $Idx_j(G_i)$: index of gene G_i within chromosome C_j

This leads to the positional distance Δ_{pos}:

$$\Delta_{pos}(C_1, C_2) = \frac{1}{dist_{\max}}\sum_{i=1}^{len} pd_{1,2}(G_i) \qquad (3.3)$$

where len: length of the chromosomes ($len > 1$)
$dist_{\max}$: distance maximum of all genes within one chromosome

The distance maximum $dist_{\max}$ results when genes are shifted by a maximal number of positions. Two cases for odd and even chromosome lengths must be considered, see Eq (3.4). The calculation of this formula is omitted here due to lack of space and can be found on the above-mentioned web site.

$$dist_{\max,\,even} = \frac{len^2}{2} \qquad dist_{\max,\,odd} = \frac{len^2 - 1}{2} \qquad (3.4)$$

The overall distance Δ of fixed-length chromosomes with relevant ordering is defined as the mean of Δ_{par} and Δ_{pos}.

Variable-length Chromosomes: The goal is to determine the precise difference of similar chromosomes, while the exact value of discrepancy of dissimilar ones is of less interest. So the resulting measure may be inexact for more different chromosomes, thus resulting in a less complex formula which reduces the computational effort for the fairly frequent distance calculations.

Let C_1 be the longer chromosome and G_{com} the set of genes common to C_1 and C_2. As genes in chromosomes of variable length may occur several times, they are treated in the sequence of their indexing. If G_{com} is empty, the overall distance is set to the maximum value of 1. Otherwise, Δ_{par} and Δ_{pos} are defined over the set of common genes G_{com}. $dist_{\max}$ is taken from C_2. This may lead to a to large value of Δ_{pos}, which may increase above 1 especially for chromosomes with large differences. Thus, Δ_{pos} is limited to 1 and the error is accepted, as it increases with the chromosome difference.

The difference of the presence of genes Δ_{gp} in two non-empty chromosomes is calculated by dividing the number of genes not common to both chromosomes by twice the length of C_1:

$$\Delta_{gp} = \frac{\left(l(C_1) - card(G_{com})\right) + card(G_{2,diff})}{2 \cdot l(C_1)}$$

$$= 1 - \frac{card(G_{com})}{l(C_1)} \qquad G_{com} \neq \varnothing \qquad (3.9)$$

because of $card(G_{2,diff}) = l(C_i) - card(G_{com})$
where $l(C_i)$: length of chromosome i
$G_{2,diff}$: set of genes of C_2 not belonging to C_1

The overall distance Δ of variable-length chromosomes is defined as the mean of the three distances, with Δ_{gp} being weighted three times, because Δ_{par} and Δ_{pos} are defined over the common set of genes only.

Stop Criterion for the EA

To reduce computational effort, the population is checked for niches only after the end of a generation when threshold values for gdi and gda are reached. For traditional EAs, these threshold values can be replaced by a threshold for generations without improvement. A niche is formed by adjacent individuals with a fitness greater than half of the global best and with Δ smaller than the strategy parameter ε. The center individual of a niche is called its representative. Niches, whose representatives differ by less than ε, are merged. The stop criterion for evolution is the existence of a maximum of N niches, whose representatives do not differ by more than ε_{pop}. This stop criterion is controlled by the five exogenous strategy parameters, gdi, gda, ε, ε_{pop}, and N.

4. FIRST EXPERIMENTS

First experiments were carried out with four test functions and one real world application using GLEAM and the Rosenbrock algorithm. The following configurations are compared: GLEAM (GL) and Rosenbrock (RB) alone, Rosenbrock-initialized start population (RI), directly integrated Rosenbrock from the beginning (GR) or after a certain niching (GRN), and post-optimization with Rosenbrock (PO). For GLEAM the population size is varied. For Rosenbrock the termination criterion controlling the precision, with which the (local) optimum is reached, is varied. Five threshold values for the overall normalized change of 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}, and 10^{-9} are compared and labeled from *s* to *xxl* according to t-shirt sizes. A too great precision (i.e. a too small threshold value) implies the risk of no convergence and *l* is what practitioners start with.

Strategy parameters for GR and GRN runs are the precision, the rule for local offspring optimization (*b* = the best only, *a* = all), the rate of Lamarckian evolution in percent, and *ri* for Rosenbrock pre-optimized start populations. For PO runs gda and gdi are set to 3, if not stated otherwise and there are three settings for ε and ε_{pop}: *e1* = (ε=0.005, ε_{pop}=0.01), *e2*= (ε=0.002, ε_{pop}= 0.005), and *e3*=(ε=0.001, ε_{pop}=0.003). The number of maximum niches N is varied with the population size: N=2 for 5 and 10, N=3 for 20 and 30, N=4 for 50 to 90, and N=5 for greater population sizes. For GRN runs gda and gdi are always set to 1.

Test Cases

Test functions have the advantage of fast calculation and, in most cases, scalable complexity due to a variable amount of parameters and the disadvantage of being somewhat artificial. So every new approach that passes test functions should be checked with some suited real world problems. The test functions taken from GENEsYs [23] are:

- Schwefel's sphere
  Unimodal problem with 30 parameters in the range of [-10^{10}, 10^{10}] and a target value of 0.01 known to be easy for ES, but hard for GA.
- Generalized Rastrigin function
  Multimodal problem with 20 parameters in the range of [-5.12, 5.12] and a target value of 0.0001 known to be hard for ES.
- Fletcher's function
  Multimodal problem with 5 parameters in the range of [-3.14, 3.14] and a target value of 0.00001.
- Fractal function
  Multimodal problem according to Weierstrass and Mandelbrot with 20 parameters in the range of [-5, 5] and an unknown minimum. The target value of –0.5 is taken from [23].

The real world problem is the design optimization of a micro optical communication device considering fabrication tolerances as described in detail in [24]. Despite of its only 3 parameters, it involves some difficulty, because it is of extreme multimodal nature.

Experimental Results

The comparison is done on the basis of the success rate and the required evaluations averaged over 100 runs for each setting of the test functions and over 50 runs for the design problem.

Schwefels's Sphere: GLEAM yields a value of some thousand after 330000 generations and 220 million evaluations. Rosenbrock is successful only for xl (54%) or xxl precision 100%), which is a non-converging setting for all other problems requiring 5404 and 6440 evaluations, respectively. Thus, both alone do not work properly for this problem. Pre-initialized start populations do not help much, but direct integration yields good results, as shown in Fig. 2.

Lowering of the Lamarckian factor always produces worse results. Using higher precisions with the GR runs results in an optimum setting of *l* and a randomly initialized population of size 5 requiring 29600 evaluations only. Post-optimization is successful in about 80% of the runs and the rest is very close to the target value for different strategy parameter settings. The best run needs 14320 evaluations with the setting *e2*, N=2, and a population size of 10.

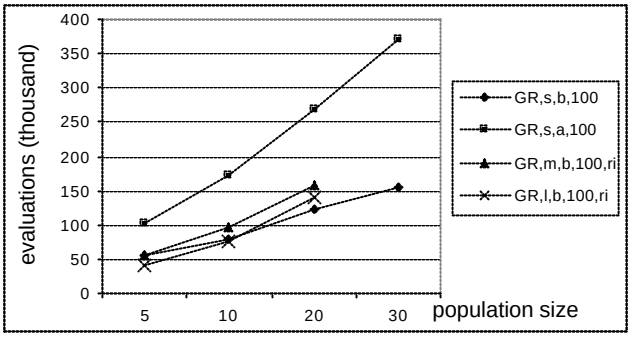

Figure 2: GR Runs with Sphere

Generalized Rastrigin Function: Rosenbrock cannot solve this problem with any precision, and for the precision of *xl* or higher convergence failures occur. GLEAM has no problems, but it even succeeds with extremely small population sizes and evaluation rates dropping simultaneously, as shown in Fig. 3. The author has not yet observed such a behavior with real world applications. Success rates should drop and evaluations should increase, if the population size is lowered below a certain amount. The best result is obtained with a GR run of medium precision and random initialization of the start population.

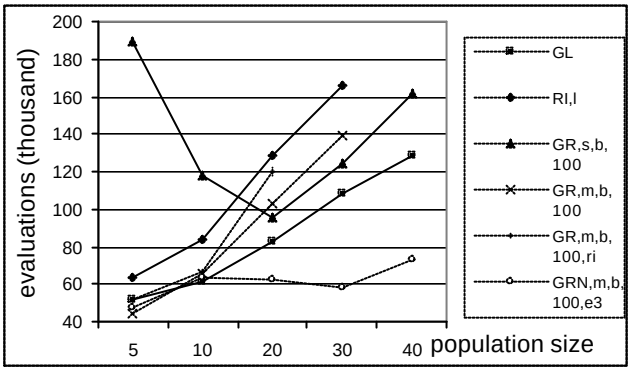

Figure 3: GL, RI, and GR Runs with the Rastrigin Function

It is surprising that both runs with Rosenbrock-initialized start populations do not perform better than randomly initialized runs. Both, lowering of the Lamarckian factor or local optimization all offsprings results in a drastic increase of evaluations (some millions) for all population sizes between 5 and 30. The GRN runs are remarkable, as their computational effort grows more slowly with increasing population sizes than in the other runs. The best GRN run is shown in Fig. 3.

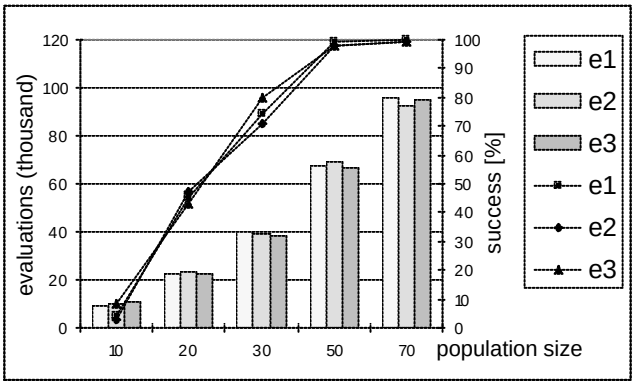

Figure 4: PO Runs with the Rastrigin Function

Post-optimization provides for sufficient success rates from a population size of 50 onwards, as shown in Fig. 4. But the required number of evaluations (columns) is larger than that of the best GR run.

Fletcher's Function: At the precision of *m* Rosenbrock succeeds in solving Fletcher's function in 10 out of 100 runs only, needing 1090 evaluations. Lower precisions do not find any solution at all and higher ones do not converge. GLEAM shows a realistic behavior, see Fig. 5.

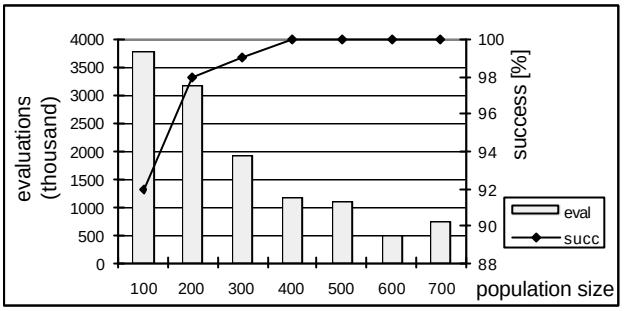

Figure 5: GLEAM with Fletcher's Function

Both algorithms combined in GR, GRN, and RI settings provide for a significant improvement, as shown by their best runs presented in Fig 6. Again, very small population sizes yield best results when the Rosenbrock procedure at precision *m* is directly integrated in GLEAM. As with the Rastrigin function, both, lowering of the Lamarckian factor or local optimization of all offsprings leads to a clear impairment. Post-optimization yields success only with a rate of about 50% and therefore is not suited for this test case.

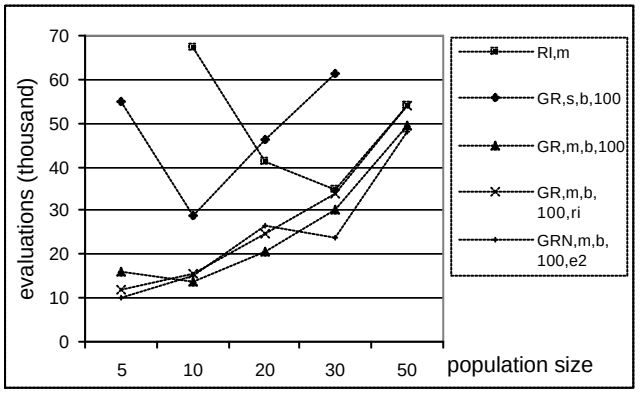

Figure 6: RI, GR, and GRN Runs with Fletcher's Function

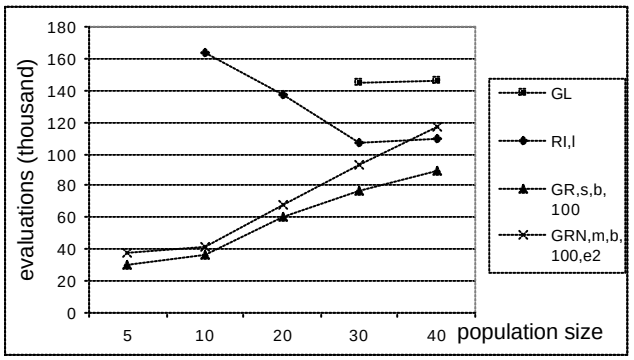

Figure 7: Best GL, RI, GR, and GRN Runs with the Fractal Function

Fractal Function: With the fractal function GLEAM shows a similar realistic behavior as with Fletcher's function. Rosenbrock provides for no success at all. Fig. 7 compares the best settings for GL, RI, GR, and GRN. The advantages of adding the local search are not as big as with Fletcher's function, which can be attributed to the fact that there is no well-suited smooth surface for the local search. Post-optimization does not work well with this test function.

Micro Optical Design Problem: Rosenbrock has a success rate of 15% with high precision, but fails to converge with higher ones. As shown in Fig. 8, GLEAM yields good success rates starting from a population size of 150 and works reliably from 240 onwards. As the average number of required generations varies between 56 and 7, there is not as much room for drastic improvements as with the test functions. The typical behavior of GLEAM with too low populations sizes can also be observed in this case.

Pre-initialization works well for population sizes of 20 or higher, but more than twice as many evaluations are needed than without. Post-optimization works best (4633 evaluations) with the *e2* setting, precision *l*, and a population size of 50, but success can be achieved in 84% of the runs only.

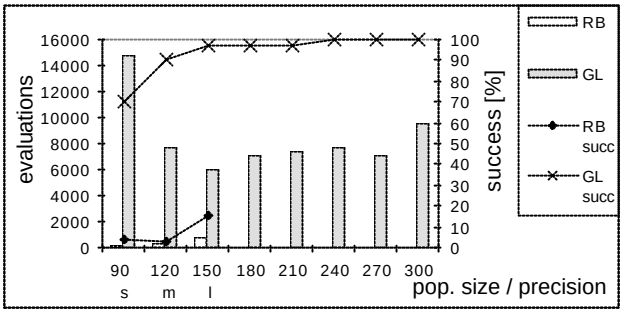

Figure 8: GLEAM vs. Rosenbrock (Design Optimization)

Better results than with GLEAM alone can be obtained from GR and GRN runs, as shown in Fig. 9. It is remarkable that this is the only case where Baldwinian evolution works comparably well or even a little better than the Lamarckian one (Lamarckian rates 5% and 0%).

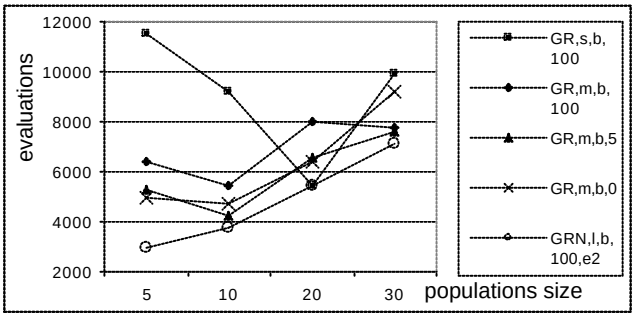

Figure 9: GR and GRN Runs (Design Optimization)

5. CONCLUSIONS AND OUTLOOK

The five test cases show that significant improvements can be achieved when local search is added to the evolutionary one. Direct integration of the local searcher in the reproduction phase of GLEAM always works well, but post-optimization turns out to be not reliable enough to yield a robust solution. The niche detection procedure introduced was used successfully for de-

layed adding of the local search within the direct integration approach. The following improvements can be stated:

- Schwefel's sphere
  This unimodal test function cannot be solved reliably by either procedure alone, but very well with direct integration.
- Generalized Rastrigin function
  Direct integration works about as well as GLEAM alone. But GLEAM still works well with untypically small population sizes, which does not only explain this observation, but reduces the value of this test function working as a proper benchmark.
- Fletcher's function
  A drastic improvement could be achieved: Delayed direct integration works about 47 times faster than the evolution alone.
- Fractal function
  Good improvements could be observed again with direct integration: A speed-up factor of 4.5 is reached.
- Design optimization example
  Delayed direct integration is 2.4 faster than evolution alone.

The above figures are the result of a comparison of the best runs with 100% success rates between GLEAM and the hybridization by (delayed) direct integration. However, it is never known in advance which is the best hybridization and its parameterization for a problem on hand. From the complete data material, it can be concluded that delayed direct integration with a population size of 5, or safer 10, and a setting of *e2* may be a good choice. In Fig. 10, the improvements of the best settings are compared with this conclusion.

Further work is presently being performed in order to include the COMPLEX algorithm in HyGLEAM and to investigate the effect of hybridization on other suited real world problems.

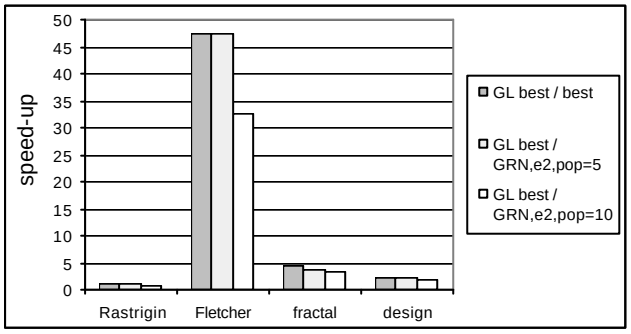

Figure 10: Comparison of the Speed-up of the Best Runs and 2 Recommended Settings

6. REFERENCES

[1] C. Blume: *GLEAM - A System for Intuitive Learning*. In: H.P. Schwefel, R. Männer (eds.): Proc. of PPSN I, pp.48-54, LNCS 496, Berlin: Springer, 1990.

[2] C. Blume: *Optimized Collision Free Robot Move Statement Generation by the Evolutionary Software GLEAM.* In: S. Cagnoni et al. (eds.): Real-World Applications of Evolutionary Computing, pp.327-338, Proc. of EvoWorkshops 2000, Springer, 2000.

[3] W. Jakob, M. Gorges-Schleuter, C. Blume: *Application of Genetic Algorithms to Task Planning and Learning.* In: R. Männer, B. Manderick (eds.): Proc. PPSN II, pp.291-300, Amsterdam: North-Holland, 1992.

[4] C. Blume, W. Jakob: *Cutting Down Production Costs by a New Optimization Method.* Proc. of the Japan - U.S.A. Symposium on Flexible Automation, ASME, 1994.

[5] C. Blume, M. Gerbe: *Deutliche Senkung der Produktionskosten durch Optimierung des Ressourceneinsatzes.* atp 36, 5/94, pp.25-29, München: Oldenbourg (in German), 1994.

[6] W. Jakob, S. Meinzer, A. Quinte, W. Süß, M. Gorges-Schleuter, H. Eggert: *Partial Automated Design Optimization Based on Adaptive Search Techniques.* In: I. C. Parmee (ed.): Adaptive Computing in Engineering Design and Control '96, pp.236-241, PEDC, Univ. of Plymouth, 1996.

[7] W. Jakob, M. Gorges-Schleuter, I. Sieber: *Comparison of Evolutionary Algorithms for Design Optimization.* In: A.Eiben, et al. (eds.): Proc. PPSN V, LNCS 1498, pp.917-926, Springer, 1998.

[8] C. Blume: *Optimization in Concrete Precasting Plants by Evolutionary Computation.* In: D. Whitley et al. (eds.): Proc. GECCO 2000, Vol. Late Papers, pp.43-50, Morgan Kaufmann, 2000.

9] W. Jakob, A. Quinte, K.-P. Scherer, H. Eggert: *Optimization of a Micro Fluidic Component Using a Parallel Evolutionary Algorithm and Simulation Based on Discrete Element Methods*. Submitted to OPTI'2001, 2001.

[10] L. Davis (ed.): *Handbook of Genetic Algorithms.* New York: Van Nostrand Reinhold, 1991.

[11] M. Gorges-Schleuter: *Genetic Algorithms and Population Structures - A Massively Parallel Algorithm*. Doctoral thesis, University of Dortmund, Germany, 1990.

[12] M. Gorges-Schleuter: *A Comparative Study of Global and Local Selection in Evolution Strategies.* In: A. Eiben et al. (eds.): Proc. PPSN V, LNCS 1498, pp.367-377, Berlin: Springer, 1998.

[13] H. H. Rosenbrock: *An Automatic Method for Finding the Greatest or Least Value of a Function*. Comp. Journ. 3, pp.175-184, 1960.

[14] M. J. Box: *A New Method of Constrained Optimization and a Comparison with Other Methods*. Comp. Journal 8, pp.42-52, 1965.

[15] J. A. Nelder, R. Mead: *A Simplex Method for Function Minimization*. Comp. Journal 7, pp.308-313, 1965.

[16] H.-P. Schwefel: *Evolution and Optimum Seeking*. John Wiley & Sons, Chichester, 1995.

[17] D. E. Goldberg, S. Voessner: *Optimizing Global-Local Search Hybrids*. In: W. Banzhaf et al.: Proc. GECCO'99, pp.220-228, San Mateo, CA: Morgan Kaufmann, 1999.

[18] D. Whitley, V. Gordon, K. Mathias: *Lamarckian Evolution, The Baldwin Effect and Function Optimization*. In: Y. Davidor et al.: Proc. PPSN III, LNCS 866, pp.6-14, Berlin: Springer, 1994.

[19] F. Gruau, D. Whitley: *Adding Learning to the Cellular Development of Neural Networks: Evolution and the Baldwin Effect*. Evol. Comp. 1, vol.3, pp.213-233, 1993.

[20] D. Orvosh, L. Davis: *Shall We Repair? Genetic Algorithms, Comb. Opt., and Feasibility Constraints*. In: S.Forrest (ed): 5th ICGA, pp.650, M. Kaufmann, 1993.

[21] S. Ronald: *Distance Functions for Order Based Encodings.* In: ICEC'97, IEEE, pp.46-54, 1997.

[22] K. Tagawa, Y. Kanzaki, D. Okada, K. Inoue, H. Haneda: *A New Metric Function and Harmonic Crossover for Symmetric and Asymmetric Traveling Salesman Problems.* In: ICEC'98, IEEE, pp.822-827, 1998.

[23] T. Bäck: *GENEsYs 1.0*, ftp://lumpi.informatik.uni-dortmund.de/pub/GA

[24] I. Sieber, H. Eggert, H. Guth, W. Jakob: *Design Simulation and Optimization of Microoptical Components*. In: Novel Optical Systems and Large-Aperture Imaging, SPIE´s 43rd Annual Meeting, SPIE Vol.3430, pp.138-149, 1998.