Benchmarking Hybrid Selection and Adaptive Genetic Operators

Václav Snášel, Pavel Krömer and Jan Platoš

Department of Computer Science,
Faculty of Electrical Engineering and Computer Science,
VŠB - Technical University of Ostrava,
17. listopadu 15, 708 33 Ostrava - Poruba, Czech Republic
{vaclav.snasel, pavel.kromer.fei, jan.platos.fei}@vsb.cz

Abstract. The area of evolutionary computation faces nowadays notable increase of both, research interest and application requirements. Genetic Algorithms are among the most popular evolutionary techniques and they are applied to a variety of real world problems. In all application domains, excellent performance expressed by the means of fast convergence times and precise results are desired. In this paper, we investigate hybrid selection and adaptive genetic operators, a modification to GA aiming to reach better performance and investigate the effect of these improvements on benchmarking problems.

1 Introduction

Genetic Algorithms are formed by wide family of methods adopting the paradigms of evolutionary algorithms and emulated genetic evolution in particular. They usually feature modifications (from minor up to major) that customize the algorithm to particular problem domain. The parameter setup of genetic algorithm is crucial for good performance in given application area and the best performing settings vary among the problem domains. However, several methods aiming to improve the performance of genetic algorithms in general were introduced. We present in this paper hybrid selection schema and genetic operators with adaptable probabilities of application as a contribution to the class of methods improving performance of genetic algorithms. Additionally, we provide initial benchmarking of these methods.

2 Evolutionary Algorithms

Evolutionary Algorithms (EA) are *family of stochastic search and optimization methods based on mimicking successful strategies observed in nature* [3, 5]. The idea behind EA is emulation of Darwinian evolution utilizing Mendelian inheritance concepts for the use in computer science. Together with Fuzzy Theory, Neural Computation, Fractals and Swarm Intelligence, EA are among fundamental ingredients of state-of-the-art Soft Computing methods [7].

EAs operate over a *population* (pool) of artificial individuals (items, chromosomes) encoding possible solutions. Encoded individuals are evaluated using objective function which assigns a *fitness* value to each individual. Fitness value represents the quality (ranking) of each individual as solution of given problem. Competing individuals intensively search the solution space of given problem in more directions simultaneously towards optimal solution [5].

The EA process starts with an initial population of individuals that might be generated randomly or seeded by an operator with potentially good solutions. The search phase of EA is iterative and consists of application of *genetic operators* (selection, crossover, mutation) on current population with the aim to form new population (new generation) that will be in following iteration treated as current. Iterative evolution of better solutions ends after satisfying specified termination criteria, i.e. finding optimal solution or processing specified number of generations.

EAs are successful general adaptable concept with good results in many application areas. The family of evolutionary algorithms consists of *Genetic Algorithms (GA), Evolutionary strategies (ES) and Evolutionary programming (EP)*. Among the most important high-level variants of Genetic Algorithms, *Genetic programming (GP)* by J. Koza attracts attention. GP is used for executing artificial evolution over hierarchical rather than linear chromosomes and therefore is usable to evolve computer programs or other hierarchically structured entities such as search queries. Genetic operators are applied on nodes of tree chromosomes encoding hierarchical individuals. GP has a good ability to produce symbolic output in response to symbolic input [3].

2.1 Genetic Operators

Genetic operators are used for implementation of artificial evolution. Operators, applied during the iterative artificial evolution are [6]:

– *Selection* operator: to select chromosomes from population. Through this operator, selection pressure is applied on the population.

– *Crossover (recombination)* operator: for varying chromosomes from one population to the next by exchanging one or more of their subparts. Mimics sexual reproduction of haploid organisms.
– *Mutation* operator: performs random perturbation in chromosome structure; used for changing chromosomes randomly and introducing new genetic material into the evolving population.

In various EA methods are particular genetic operators used in different manner [6]. Evolutionary programming [5, 1] operates over strings of real numbers and uses only mutation operator, while evolutionary strategies [5, 1] evolve individuals composed of real-numbers string and a group of objective variables used for tuning the mutation operator. Genetic algorithms use recombination operator as most important element of artificial evolution.

2.2 Genetic Algorithms

Genetic Algorithms, introduced by Holland in 60s and extended by Goldberg at the end of 80s, were defined when exploring the possibilities of computer emulated evolution [6]. Now, they are widely applied, rapidly growing and highly successful EA variant.

In contrast to ES and EP, genetic algorithms were originally designed as a general model of adaptive processes. Similarly to ES and EP, they have found most of its applications in the domain of optimization. Basic workflow of Holland's originally proposed *generational GA* is as follows:

1. Encode initial population of possible solutions and evaluate chromosomes (assign fitness value)
2. Create new population (evolutionary search for better solutions):
    a. Select *suitable chromosomes for reproduction (parents)*
    b. *Apply* crossover *operator on parents with respect to crossover probability to produce new chromosomes (offspring)*
    c. *Apply* mutation *operator on offspring chromosomes with respect to mutation probability. Add newly constituted chromosomes to new population*
    d. *Until the size of new population is smaller than size of current population go back to* a.
    e. *Replace current population by new population*
3. Evaluate current population; assign fitness values to chromosomes
4. Check termination criteria; if not satisfied go back to 2.

There are several high-level modifications of genetic algorithms. Among the most important are: *generational GA*, where the replacement of chromosomes in population between two generations is driven according to a *generation gap* parameter. If the generation gap value is 1, whole parent population is replaced by newly created offspring

chromosomes [6]. This approach corresponds to Holland's original proposal of GA. In contrast, *steady state GA* has never replaced whole population when migrating from one generation to another. Only few weakest individuals are replaced by fittest offspring chromosomes. This is more exact emulation of long time living species evolution, where parents and children compete in one population. Therefore, more promising individuals can be intensively investigated at the time of their creation. There is no proof whether first of these two approaches is fundamentally better than latter; both are used according to application demands [6].

2.3 Optimization of genetic parameters

Termination criteria, probability of crossover, probability of mutation, size of population, maximum number of processed generations and migration strategy are among the most important parameters of each GA implementation. Moreover, the particular operator implementation affects the efficiency of the algorithm fundamentally. All the mentioned parameters must be tuned to fit the particular problem and its domain in order to provide best performance. The tailoring of proper genetic parameters is non-trivial task and employs sophisticated techniques. The methods can be divided on domain knowledge based approaches, exploiting certain property of problem domain to choose proper genetic parameters and soft approaches seeing the genetic parameter optimization task as general optimization problem. Among others soft approaches, genetic algorithms itself were used for optimization of GA parameters introducing meta-optimization to GA parameters harmonization studies. In this paper, we investigate two GA improvement techniques aiming to offer performance gain when exploiting GA as soft optimization method. The methods are introduced and their performance results evaluated using selection of GA benchmarking problems.

3 Hybrid selection

GA selection schemes are crucial for the successful run of the optimization algorithm. The selection operator implementation introduces selection pressure to the population when selecting chromosomes for mating to form new offspring and therefore contribute on the new population. Natural requirements on selection operator are quick convergence to global optima while avoiding local optima. The scenario when algorithm jams in local optima is known as *premature* convergence. Also, the *diversity* (variety of different chromosomes) in population is desired to stay high in order to avoid local optima. Genetic Algorithm using some sort of common selection operator preferring blindly higher fitness chromosomes such as proportionate selection, truncation selection,

tournament selection and so on usually offer good diversity in population leading to globally optimal solutions for the price of notably longer convergence time.

GA exploiting elitary selection means chose parents among the best ranked individuals in population only. This supports quick convergence under the threat of premature convergence. In this paper, we combine common selection with elitary selection in order to obtain quick and robust genetic algorithm providing both, improved convergence time and local optima awareness.

The hybrid selection scenario is defined as follows: first parent is selected by elitary means and the second by common means. It means that the selection method always picks one member of the class of bets ranked chromosomes from the population, being the best solutions found so far, and second parent is chosen from whole population (possibly but not necessarily also belonging to the class of best ranked individuals).

4 Adaptive genetic operator probability

The disadvantage of GA as soft universal optimization method is the need to choose right parameters for the genetic algorithm in order to obtain good performance in every particular application area. There is no common general parameter-less GA (although several investigations are in progress) or common general GA parameter setup (although some GA parameter values are being suggested or widely used in particular application areas). Moreover, the static assignment of GA operator probability does not reflect the changing state of the genetic algorithm and its population during the search. The maturity of whole population at certain generation (at certain optimization step) can be described by the means of *fitness landscape*. The fitness landscape (FL), inspired similarly as genetic algorithms in biological science, is used to describe the dynamics of artificial evolution [2]. FL is a N+1 dimensional hyper-surface for which the N corresponds with number of genes in chromosome and the final dimension is fitness value for the chromosome. According to the fitness value, FL can be divided into regions of several kinds:

1. *Flat* or *neutral*, where the offspring population have no or insufficient improvement over parent populations
2. *Multimodal*, where the convergence varies rapidly
3. *Unimodal*, which is the state in which the population approaches optimum state

Intuitively, GA is desired to leave the flat and multimodal areas and FL aware probability of application of GA parameters can contribute to this. It has been shown, that properly defined dynamically changing fitness sensitive probability of mutation and crossover can improve the performance of GA over static parameter setup.

Our genetic algorithm uses the functions $P_M : [0,1] \rightarrow R$ and $P_c : [0,1] \rightarrow R$ defined as follows:

$$P_C(f) = f^2 \tag{1}$$

$$P_M(f) = 1 - \frac{1}{(1 + e^{(\frac{f + c \cdot \ln(2^{\frac{1}{d}} - 1) - b}{c})})^d} \tag{2}$$

Asymmetric reverse sigmoid (2) is for our purposes defined with following parameters: b denoting centre of the transition is equal to 0.75, c and d used for enumeration of transition width is equal to -0.05 and 0.3 respectively. It assigns high mutation rate to low valued individuals and low mutation rate to high fitness chromosomes (see figure 2). The fitness values are expected to be normalized to the interval [0, 1] and the algorithm is supposed to perform maximization of fitness value.

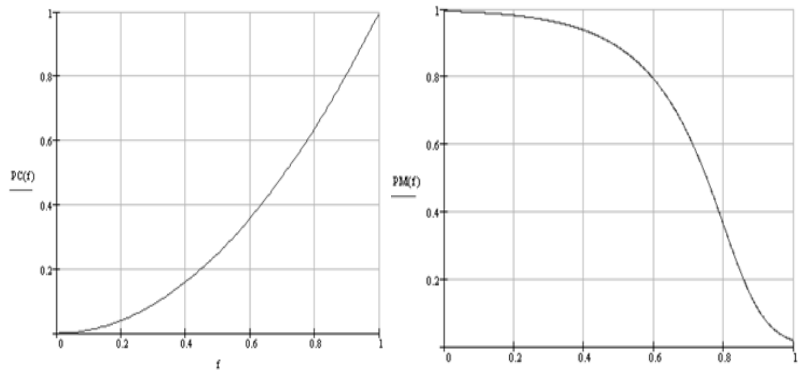

Fig. 1. Adaptive probability of crossover P_C and adaptive probability of mutation P_M

Adaptive genetic operators defined as above prioritize for lower fitness chromosomes mutation and for higher fitness chromosomes crossover. It means that for low fitness areas the algorithm performs rather random search jumps over large distances in FL in order to change the bad genome and for high fitness chromosomes is performed careful search exploiting good genome of high ranked chromosomes.

5 Experiments and results

We have tested proposed improvements with several GA benchmarking functions in R^2 selected mainly from [4]. The performance of improved GA was compared to performance of classical GA.

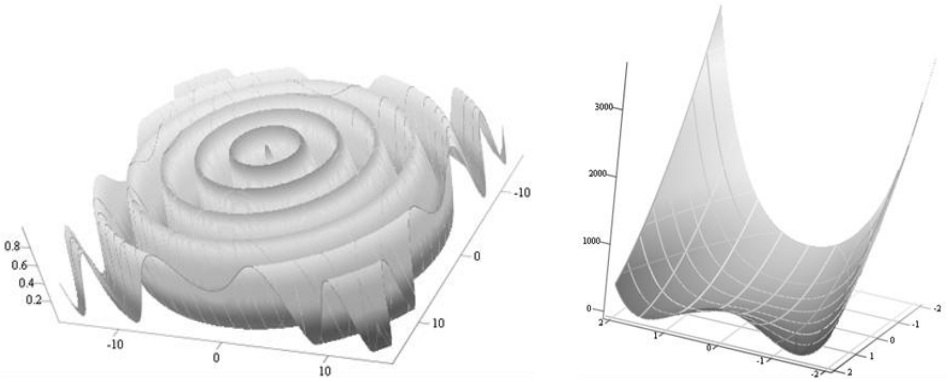

Fig. 2. Function f_1 and Rosenbrock valley function f_2

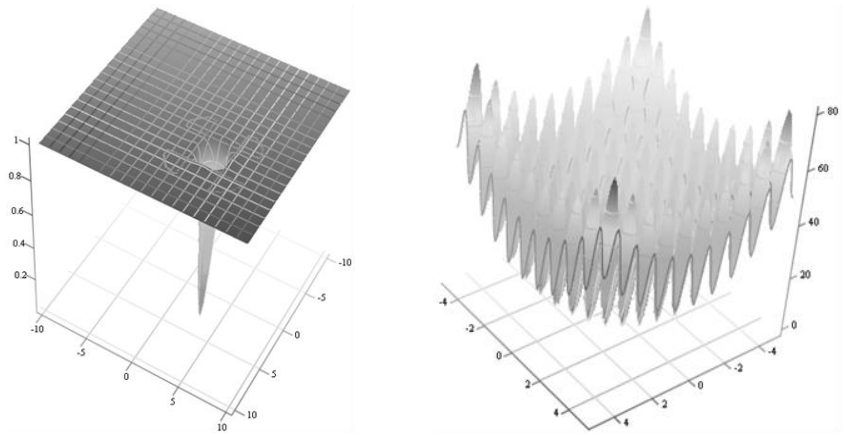

Fig. 3. Easom function f_3 and Rastrigin function f_4

$$f_1(x,y) = \frac{1-(\sin(\sqrt{x^2+y^2}))^2}{1+\frac{x^2+y^2}{1000}} \tag{3}$$

$$f_2(x,y) = 1 - x^2 + 100 \cdot (y - x^2)^2 \tag{4}$$

$$f_3(x,y) = 1 - \cos(x) \cdot \cos(y) \cdot e^{-(x-\pi)^2 - (y-\pi)^2} \tag{5}$$

$$f_4(x,y) = 20 + x^2 + y^2 - 10 \cdot \cos(2\pi x) + \cos(2\pi y) \tag{6}$$

Let us briefly mention the properties of benchmarking functions [4]. Function f_1 has global optimum at [0, 0] and features lots of global minima and maxima in circles around [0, 0]. Rosenbrock valley function f_2 defined in (4) features parabola shaped narrow ridge with global minima 0 at [1, 1]. Both functions are shown in Figure 2. Easom function f_3 defined in (5) is difficult due to the steep and relatively small filler area with global minima 0 at [π, π] while the majority of its surface is flat. Ratsrigin function f_4 of (6) contains dozens of local optima and one global minimum 0 at [0, 0]. These functions are illustrated in Figure 3.

Table 1. Average number of generations (out of 25000)

Function / Algorithm	f_1	f_2	f_3	f_4
plain (tuned parameters)	15592	25000	25000	5690,1
plain (wrong parameters)	25000	25000	25000	25000
Elitism	13899	25000	25000	6471,6
Hybrid	10696	25000	25000	2414,8
Adjusted	25000	25000	25000	7864,9
hybrid and adjusted	19658	22823	25000	5326,7
hybrid and elitary	25000	25000	25000	5706,4

The experimental results expressed by the means of number of generations processed and fitness at final generation are shown in Tables 1 and 2 respectively. The experiments were performed multiple times and presented results are average values of the experimental runs. Genetic algorithm was terminated when the global optimum was reached or 25000 generations passed. Genetic operators were 1-point crossover and bit mutation. The chromosome was implemented as a pair of two coordinates with at most two decimals (so for instance the best approximation of π was 3.14). The classic implementation of GA is in the tables referred as plain and adjusted denotes the experiments with adaptive genetic operators.

Table 2. Average number of generations (out of 25000)

Function / Algorithm	f_1	f_2	f_3	f_4
plain (tuned parameters)	1	0,8131	0,8661	1
plain (wrong parameters)	0,9936	0,8992	0,9843	0,503
elitism	1	0,8126	0,8602	1
hybrid	0,999	0,7595	0,8311	1
adjusted	0,9938	0,8609	0,6997	0,9112
hybrid and adjusted	0,997	0,976	0,7526	0,9201
hybrid and elitary	0,9942	0,7553	0,6662	0,9201

The results indicate the following: it has been confirmed, that the right parameter setup affects GA performance rapidly. The wrong parameters avoided GA to reach optimal solution in less than 25000 generations while it has been achieved when using tuned parameters (row 1). Surprisingly, the parameter setup we expected as wrong caused increase in final fitness for f_2 and f_3. Elitism decreased the number of generations needed for finding optima for functions f_1 and f_4 while not affecting eminently results for f_2 and f_3. Hybrid selection increased the convergence speed at most while producing only slightly lower average fitness. It means that the algorithm with hybrid selection converged to global optimum quickly several times but also sometimes missed the optimum significantly. Adjusted operator probability did not speeded up the convergence over wrong parameterized GA but caused better average final fitness. The combination of hybrid selection and adaptive genetic operators increased the convergence speed over the level of GA with adaptive operators only and over-performs clearly wrong parameter GA and also the combination of elitism with adaptable operators. Also, the variant with adaptable operators and hybrid selection as the only found the global optima of Rosenbrocks function f_2.

6 Conclusion

This paper summarizes hybrid selection and genetic operators with adaptable probabilities and provides benchmark of those methods. We show that hybrid selection and adjustable operators can contribute to both, convergence speed and result precision, especially comparing to classic GA with wrong parameter setup. Good results obtained for Rosenbrock valley function suggest that the modified algorithm is good at finding the right direction in the problem space.

Clearly, the results obtained for adaptable genetic operator probabilities applied without the support of hybrid selection are rather unsatisfactory and the adaptable operators are still under investigation, focusing on the function $P_C(f)$, since there is not as firm underlying semantics as for $P_M(f)$. Moreover, $P_M(f)$ as defined in (2) features number of parameters that deserve further investigation.

References

1 Thomas Bäck, U. Hammel, and H.-P. Schwefel. Evolutionary computation: comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):pp. 3–17, April 1997.

2 S. Blum, R. Puisa, J. Riedel, and M. Wintermantel. Adaptive mutation strategies for evolutionary algorithms. In *The annual conference: EVEN at Weimarer Optimierungs- und Stochastiktage 2.0*, 2001.

3 Mehrdad Dianati, Insop Song, and Mark Treiber. An introduction to genetic algorithms and evolution strategies. Technical report, University of Waterloo, Ontario, N2L 3G1, Canada, July 2002.

4 Jason Digalakis and Konstantinos Margaritis. An experimental study of benchmarking functions for evolutionary algorithms. *International Journal of Computer Mathemathics*, 79(4):403–416, April 2002.

5 Gareth Jones. Genetic and evolutionary algorithms. In Paul von Rague, editor, *Encyclopedia of Computational Chemistry*. John Wiley and Sons, 1998.

6 Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.

7 Vitorino Ramos and Ajith Abraham. Evolving a stigmergic self-organized data-mining. *CoRR*, cs.AI/0403001, 2004.