Diversity-Guided Evolutionary Algorithms

Rasmus K. Ursem

EVALife, Dept. of Computer Science, University of Aarhus, Bldg. 540,
Ny Munkegade, DK-8000 Aarhus C, Denmark
ursem@daimi.au.dk

Abstract. Population diversity is undoubtably a key issue in the performance of evolutionary algorithms. A common hypothesis is that high diversity is important to avoid premature convergence and to escape local optima. Various diversity measures have been used to analyze algorithms, but so far few algorithms have used a measure to *guide the search.*
The diversity-guided evolutionary algorithm (DGEA) uses the well-known distance-to-average-point measure to alternate between phases of exploration (mutation) and phases of exploitation (recombination and selection). The DGEA showed remarkable results on a set of widely used benchmark problems, not only in terms of fitness, but more important: The DGEA saved a substantial amount of fitness evaluations compared to the simple EA, which is a critical factor in many real-world applications.

1 Introduction

A major problem in evolutionary algorithms (EAs) is that simple EAs have a tendency to converge to local optima. This *premature convergence* is caused by several algorithmic features, particularly selection pressure and too high gene flow between population members. First, a high selection pressure will quickly fill the population with clones of the better fit individuals, simply because their survival probability is too high compared to intermediate fit solutions. *Diversity* declines after a short while, and, because the population consists of similar individuals, the algorithm will have difficulties escaping the local optimum represented by the population. However, lowering the selection pressure is rarely an option because this will often lead to an unacceptable slow convergence speed. Second, high gene flow is often determined by the population structure. In simple EAs any individual can mate with any other individual. Consequently, genes spread fast throughout the population and the diversity drops quickly with fitness stagnation as a prevalent outcome.

Several studies have been carried out with the conflicting goals of maintaining a diversity that allows rapid convergence and still avoid premature convergence. Most studies fall in one of the following three categories:

1. Complex population structures to lower gene flow, e.g., the diffusion model [1, C6.3], the island model [1, C6.4], the multinational EA [2], and the religion-based EA [3].

2. Specialized operators to control and assist the selection procedure, e.g., crowding [4], deterministic crowding [5], and sharing [6].
3. Reintroduction of genetic material, e.g., random immigrants [7], mass extinction models [8], [9], and [10].

Diversity is undoubtably closely related to the performance of evolutionary algorithms, especially when attempts are made to overcome the problems of avoiding premature convergence and escaping local optima. Maintaining high diversity is particulary important for optimization of dynamic and multiobjective problems. For dynamic problems high diversity increases the chances of relocating the peak after a change in the landscape, simply because the population covers a larger part of the search space. Algorithms for multiobjective optimization seek to report many tradeoffs between the conflicting objectives. Hence, higher diversity allows the algorithm to discover a larger part of the so-called Pareto front and thus report multiple tradeoffs between the objectives.

Diversity measures are traditionally used to *analyze* evolutionary algorithms rather than *guide* them. However, diversity measures have been used to control EAs in at least three studies. The Diversity-Control-Oriented Genetic Algorithm [11] use a diversity measure based on Hamming distance to calculate a survival probablility for the individuals. A low Hamming distance between the individual and the current best individual is translated into a low survival probability. Hence, diversity is preserved through the selection procedure. Another approach is the Shifting-Balance Genetic Algorithm [12]. The SBGA calculates a so-called containment factor between two subpopulations, which is based on Hamming distances between all members of the two populations. The distance is calculated between each member of population A and all members of population B. The factor determines the ratio between individuals selected on fitness and individuals selected to increase the distance between the two populations. A third, and more distantly related, approach is the Forking GA, which uses specialized diversity measures to turn a subset of the population into a subpopulation [13]. Two variants of the Forking GA exists. The first variant operates on the genotype, whereas the second type base the division on distances in the search space (on the phenotype).

2 The Diversity-Guided EA

The idea behind the DGEA is simple. Unlike most other EAs the DGEA uses a diversity measure to alternate between exploring and exploiting behavior. To use a measure for this purpose it has to be robust with respect to i) the population size, ii) the dimensionality of the problem, and iii) the search range of each of the variables. An immediate measure for N-dimensional numerical problems is the "distance-to-average-point" measure defined as:

$$diversity(P) = \frac{1}{|L| \cdot |P|} \cdot \sum_{i=1}^{|P|} \sqrt{\sum_{j=1}^{N} \left(s_{ij} - \overline{s}_j\right)^2}$$

where $|L|$ is the length of the diagonal[1] in the search space $\mathcal{S} \subseteq \mathbb{R}^N$, P is the population, $|P|$ is the population size, N is the dimensionality of the problem, s_{ij} is the j'th value of the i'th individual, and $\overline{s}_j$ is the j'th value of the average point $\overline{s}$. The pseudocode for the DGEA is listed in Fig. 1.

```
DGEA main
  t = 0
  Initialize population $P(0)$
  Evaluate population $P(0)$
  mode = "Exploit"
  while(not(termination condition)) {
    t = t+1
    if(diversity($P$(t))$< d_{low}$)
        mode = "Explore"
    elseif(diversity($P$(t))$> d_{high}$)
        mode = "Exploit"

    if(mode == "Exploit")
        Select next generation $P$(t) from $P$(t $-$ 1)
        Recombine $P$(t)
    else
        Mutate $P$(t)
    Evaluate population $P$(t)
  }
```

Fig. 1. Pseudocode for the DGEA.

The DGEA applies *diversity-decreasing* operators (selection and recombination) as long as the diversity is above a certain threshold d_{low}. When the diversity drops below d_{low} the DGEA switches to *diversity-increasing* operators (mutation) until a diversity of d_{high} is reached. Hence, phases with exploration and phases with exploitation will occur (see Fig. 2). Theoretically, the DGEA should be able to escape local optima because the operators will force higher diversity regardless of fitness.

If $d_{low} = d_{high}$ the algorithm will maintain a diversity close to the given threshold value, which is particulary useful for dynamic and multiobjective optimization tasks.

An important issue is to apply a mutation operator that rather quickly increases the distance-to-average-point measure. Otherwise, the algorithm will stay in "explore"-mode for a long time. A straightforward idea for a measure-increasing mutation operator is to use the average point of the population to calculate the direction of each individual's mutation. The individual is then mutated with the Gaussian mutation operator, but now with a mean directed away

[1] Assuming that each search variable x_k is in a finite range, i.e., $x_{kmin} \leq x_k \leq x_{kmax}$.

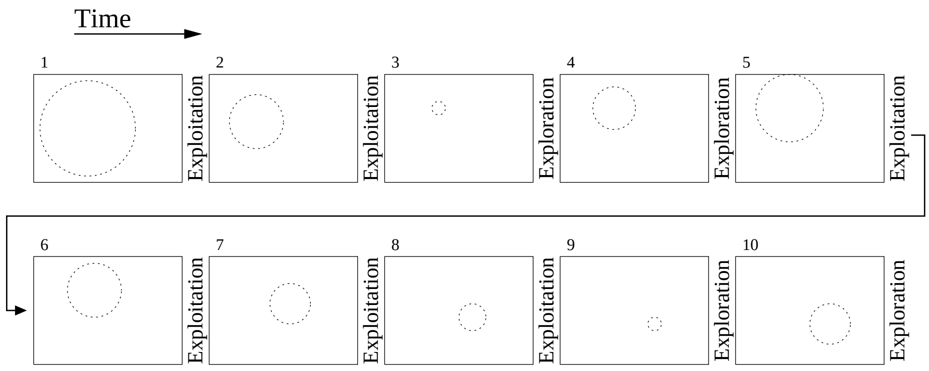

Fig. 2. Phases in the DGEA. The boxes denote the search space, the dotted circles indicate the diversity and position of the population. The mode is shown as the vertical text between the pictures, i.e. exploitation lowers the diversity in picture 1 and transforms it into picture 2.

from the average point (see Fig. 3). The purpose of this mutation operator is to *force* the individuals away from the population center. Preliminary results indicated that scaling the normalized direction vector by 0.001 turned out to give the best results.

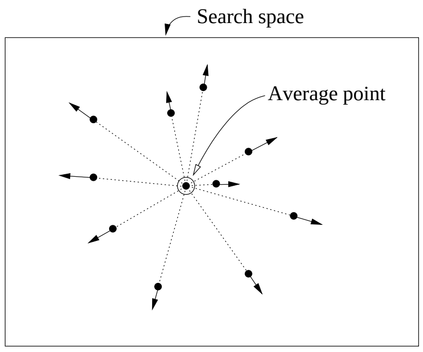

Fig. 3. Directed mutation in the DGEA.

3 Experiments and Results

The experiments were performed using real-valued encoding, a population size of 400 individuals, binary tournament selection, and an elitism of 1 individual. Probability of mutating an entire genome was $p_m = 0.75$ and probability for crossover was $p_c = 0.9$. The compared algorithms all use variants of the standard Gaussian mutation operator (see below for further details). The mutation operator scales the randomly generated numbers by 20% of the length of the search intervals, which is just to make the operator problem independent. The algorithms use an arithmetic crossover with one weight for each variable. All weights except one are randomly assigned to either 0 or 1. The remaining weight is set to a random value between 0 and 1. In preliminary experiments this hybrid between uniform and arithmetic crossover showed better performance than traditional uniform and arithmetic crossover. Two sets of experiments were conducted: i) the traditional comparison between different algorithms and ii) experiments on diversity.

3.1 Traditional Optimization

The algorithms used in the comparison are the "standard EA" (SEA), the self-organized criticality EA (SOCEA), the cellular EA (CEA), and the diversity-guided EA (DGEA). They all use the above parameters. The SEA uses Gaussian mutation with zero mean and variance $\sigma^2 = 1/\sqrt{t+1}$. The SOCEA is a standard EA with non-fixed and non-decreasing variance $\sigma^2 = POW(10)$, where $POW(\alpha)$ is the power-law distribution[2]. The purpose of the SOC-mutation operator is to introduce many small, some mid-sized, and a few large mutations. The effect of this simple extension is quite outstanding considering the effort to implement it (one line of code). The reader is referred to [10] for additional information on the SOCEA. Further, the CEA uses a 20×20 grid with wrapped edges. The grid size corresponds to the 400 individuals used in the other algorithms. The CEA uses Gaussian mutation with variance $\sigma^2 = POW(10)$, which allows comparison between the SOCEA and this version of the CEA. Mating is performed between the individual at a cell and and a random neighbor from the four-neighborhood. The offspring replaces the center individual if it has a better fitness than the center individual. Finally, the DGEA uses the Gaussian mutation operator with variance $\sigma^2 = POW(1)$ and mean calculated as described in Sect. 2. The diversity boundaries were set to $d_{low} = 5 \cdot 10^{-6}$ and $d_{high} = 0.25$, which proved to be good settings in preliminary experiments.

The algorithms were compared using four standard benchmark problems. Each algorithm was tested on three variants of the problems; a 20 dimensional, a 50 dimensional, and a 100 dimensional. The number of generations was set to 50 times the dimensionality of the test problem, i.e., 20D = 1000 generations,

[2] Power-law distributed numbers can be generated by $x = 1/u^{1/\alpha}$, where $u \sim U(0,1)$ is uniformly distributed, and α is a parameter determining the shape of the distribution. Another approach used in [10] is to log the avalanche sizes in the so-called sandpile model [14].

50D = 2500 generations, and 100D = 5000 generations. The four (minimization) problems are:

$$\text{Ackley F1}(x) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi \cdot x_i)\right) \quad \text{where } -30 \le x_i \le 30$$

$$\text{Griewank F1}(x) = \frac{1}{4000}\sum_{i=1}^{n}(x_i - 100)^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i - 100}{\sqrt{i}}\right) + 1 \quad \text{where } -600 \le x_i \le 600$$

$$\text{Rastrigin F1}(x) = \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10) \quad \text{where } -5.12 \le x_i \le 5.12$$

$$\text{Rosenbrock F1}(x) = \sum_{i=1}^{n-1}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \quad \text{where } -100 \le x_i \le 100$$

The results listed in Table 1 clearly show that the DGEA outperforms the other algorithms by several magnitudes. In preliminary tests the DGEA continued to improve the fitness at the end of the fixed number of generations. Hence, better fitness was obtained if the DGEA ran until it stagnated, which is here defined to be 500 generations without fitness improvement. The results for this optimization are listed in the column denoted DGEA*.

From the pseudocode listed in Fig. 1 it seems that the DGEA should be somewhat faster than other algorithms, because the evolutionary operators are used less frequently (although the diversity calculation might be more expensive). Table 2 lists the runtime for the tested algorithms. According to these experiments the DGEA uses less time than even the simple standard EA.

3.2 Investigations on Diversity

Low diversity is often blamed for being the main reason for premature convergence. The principal argument is that a population of clones is likely to get trapped in a local optimum. However, low diversity might increase the chances of producing fitter offspring because the population will be located in the vicinity of an optimum. Table 3 shows the average diversity and mode for each test problem. The data points are recorded at each fitness improvement *after* the first period of fitness stagnation, which is defined to be 20 consecutive generations with no fitness improvement[3].

[3] This is to eliminate the noise in the beginning of a run.

Table 1. Average fitness of the SEA, the SOCEA, the CEA, and the DGEA. 20D problems were optimized for 1000 generations, 50D for 2500, and 100D for 5000 generations. The results in the column DGEA* were obtained by running the DGEA until the best fitness was constant for 500 generations.

Problem	SEA	SOCEA	CEA	DGEA	DGEA*
Ackley 20D	2.49431	0.63380	0.23972	8.05E-4	3.36E-5
Ackley 50D	2.87039	1.52580	0.65169	4.61E-3	2.52E-4
Ackley 100D	2.89336	2.22080	1.14013	0.01329	9.80E-4
Griewank 20D	1.17195	0.93078	0.64280	7.02E-4	7.88E-8
Griewank 50D	1.61642	1.14741	1.03284	4.40E-3	1.19E-3
Griewank 100D	2.25001	1.62948	1.17907	0.01238	3.24E-3
Rastrigin 20D	11.12678	2.87524	1.25016	2.21E-5	3.37E-8
Rastrigin 50D	44.67488	22.46045	14.22400	0.01664	1.97E-6
Rastrigin 100D	106.21298	86.36449	58.38013	0.15665	6.56E-5
Rosenbrock 20D	8292.320	406.490	149.056	96.007	8.127
Rosenbrock 50D	41425.674	4783.246	1160.078	315.395	59.789
Rosenbrock 100D	91250.300	30427.636	6053.870	1161.550	880.324

Table 2. Milliseconds used by the SEA, the SOCEA, the CEA, and the DGEA. Average of 100 runs with 5000 generations on each of the four 100D problems.

Problem	SEA	SOCEA	CEA	DGEA
Ackley 100D	1128405	1528864	2951963	864316
Griewank 100D	1171301	1562931	3656724	969683
Rastrigin 100D	1124925	1513691	2897793	819691
Rosenbrock 100D	1087615	1496164	2183283	883811
Total	4512246	6101650	11689763	3537501
Percentage	100%	135.2%	259.1%	78.4%

Two interesting conclusions can be drawn from the results in Table 3. First, the diversity appears to be surprisingly low when improvement occurs[4]. Second, almost no improvement occurs during the exploration phases. This is particulary interesting because it can help save a substantial amount of fitness evaluations during the exploration periods – an important feature for time-consuming evaluation of real-world problems. A variant (DGEA2) without evalution during exploration was investigated to see if it was possible to save fitness evaluations. The optimization results and the percentage of evaluations used by DGEA2 compared to DGEA and SEA are listed in Table 4. It should be noted that the DGEA2 uses a special kind of elitism during the explorative phases. In this elitism operator the best individual from the previous exploit-phase overwrites a random individual in the population, whereas the worst individual is over-

[4] The range of the diversity measure is 0 to 0.5.

Table 3. Average diversity and current mode for the DGEA after first stagnation period. Data points are recorded when fitness improvement is detected, i.e., 100% in the Exploit column means that all improvements occured in Exploit mode. Each row is the average of 100 runs.

Problem	Diversity	Exploit	Explore
Ackley 20D	0.000388	100.00%	0.00%
Ackley 50D	0.000764	100.00%	0.00%
Ackley 100D	0.001082	100.00%	0.00%
Griewank 20D	0.000253	100.00%	0.00%
Griewank 50D	0.000662	100.00%	0.00%
Griewank 100D	0.000932	100.00%	0.00%
Rastrigin 20D	0.002056	100.00%	0.00%
Rastrigin 50D	0.002379	100.00%	0.00%
Rastrigin 100D	0.002817	100.00%	0.00%
Rosenbrock 20D	0.000601	99.99%	0.01%
Rosenbrock 50D	0.001134	99.91%	0.09%
Rosenbrock 100D	0.001562	99.91%	0.09%

written during the exploitation phases. This special scheme is used to avoid the evaluation of individuals during the explorative phases.

Table 4. Average fitness of DGEA, average fitness of DGEA2, and number of fitness evaluations in percentage used by DGEA2 compared to DGEA and SEA. Number of generations are: 20D = 1000 generations, 50D = 2500, and 100D = 5000 generations (same as Table 1).

	Fitness		Evaluations in DGEA2	
Problem	DGEA	DGEA2	vs. DGEA	vs. SEA
Ackley 20D	8.05E-4	1.01E-3	70.1%	64.0%
Ackley 50D	4.61E-3	4.36E-3	74.3%	67.9%
Ackley 100D	0.01329	0.01311	77.0%	70.5%
Griewank 20D	7.02E-4	1.11E-3	94.2%	86.7%
Griewank 50D	4.40E-3	3.96E-3	94.8%	87.3%
Griewank 100D	0.01238	9.94E-3	95.4%	87.9%
Rastrigin 20D	2.21E-5	6.88E-4	58.6%	53.2%
Rastrigin 50D	0.01664	0.03699	61.9%	56.3%
Rastrigin 100D	0.15665	0.15613	64.1%	58.4%
Rosenbrock 20D	96.007	86.891	89.9%	82.7%
Rosenbrock 50D	315.395	295.680	90.3%	83.0%
Rosenbrock 100D	1161.550	758.040	90.5%	83.2%
Average			80.1%	73.4%

4 Conclusions

The experiments revealed a number of interesting features of the DGEA in relation to optimization tasks. First, the DGEA outperformed the other algorithms by several magnitudes on all test problems – it is clearly capable of escaping local optima. Second, the EA part (crossover, selection, and mutation) of the DGEA has lower running time than the standard EA, which again has lower running time than most other EAs. Third, the number of fitness evaluations may be reduced by approximately 25% compared with the standard EA, because the fitness is constant during the explorative phases. Reducing fitness evaluations is highly desireable for real-world applications, because the evaluation is often the time-critical factor in such applications. However, the results showed some variation in the reduction percentages, which indicates that this could be problem dependent.

In a more general context this study show the importance of *both* high and low diversity in optimization. High diversity allows the algorithm to escape local optima whereas low diversity ensures progress when fine-tuning the solutions.

Future work includes testing various variants of the algorithm. For instance, annealing the diversity thresholds d_{low} and d_{high} could lead to improvements, because it may be an advantage to decrease d_{low} near the end of the optimization. Further, investigations on a set of real-world problems are necessary to verify the results in a more realistic context. A number of system identification problems from control engineering are currently being investigated. Preliminary results from these studies are very encouraging.

References

1. Bäck, T., Fogel, D.B., Michalewicz, Z., and others, (eds.): Handbook on Evolutionary Computation. IOP Publishing Ltd and Oxford University Press, (1997)
2. Ursem, R.K.: Multinational Evolutionary Algorithms. In: Proceedings of the Congress of Evolutionary Computation (CEC-99), Angeline, P.J., Michalewicz, Z., Schoenauer, M., Yao, X., and Zalzala, A. (eds.), Vol. 3. 1633–1640 (1999)
3. Thomsen, R., Rickers, P., and Krink, T.: A Religion-Based Spatial Model For Evolutionary Algorithms. In: Parallel Problem Solving from Nature – PPSN VI, Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J.J., and Schwefel, H.P. (eds.), Vol. 1. 817–826 (2000)
4. De Jong, K.A.: An Analysis of the Behavior of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan, Ann Arbor, MI, (1975). Dissertation Abstracts International 36(10), 5140B, University Microfilms Number 76-9381
5. Mahfoud, S.: Crowding and preselection revisited. Technical Report 92004, Illinois Genetic Algorithms Laboratory (IlliGAL), (1992)
6. Goldberg, D.E. and Richardson, J.: Genetic Algorithms with Sharing for Multimodal Function Optimization. In: Genetic Algorithms and their Applications (ICGA'87), Grefenstette, J.J. (ed.), 41–49. Lawrence Erlbaum Associates, Publishers, (1987)

7. Cobb, H.G. and Grefenstette, J.F.: Genetic Algorithms for Tracking Changing Environments. In: Proceedings of the 5th International Conference on Genetic Algorithms, Forrest, S. (ed.), 523–530 (1993)
8. Thomsen, R. and Rickers, P.: Introducing Spatial Agent-Based Models and Self-Organised Criticality to Evolutionary Algorithms. Master's thesis, University of Aarhus, Denmark, (2000)
9. Greenwood, G.W., Fogel, G.B., and Ciobanu, M.: Emphasizing Extinction in Evolutionary Programming. In: Proceedings of the Congress of Evolutionary Computation, Angeline, P.J., Michalewicz, Z., Schoenauer, M., Yao, X., and Zalzala, A. (eds.), Vol. 1. 666–671 (1999)
10. Krink, T., Thomsen, R., and Rickers, P.: Applying Self-Organised Criticality to Evolutionary Algorithms. In: Parallel Problem Solving from Nature – PPSN VI, Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J.J., and Schwefel, H.P. (eds.), Vol. 1. 375–384 (2000)
11. Shimodaira, H.: A Diversity Control Oriented Genetic Algorithm (DCGA): Development and Experimental Results. In: Proceedings of the Genetic and Evolutionary Computation Conference, Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., and Smith, R.E. (eds.), Vol. 1. 603–611 (1999)
12. Oppacher, F. and Wineberg, M.: The Shifting Balance Genetic Algorithm: Improving the GA in a Dynamic Environment. In: Proceedings of the Genetic and Evolutionary Computation Conference, Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., and Smith, R.E. (eds.), Vol. 1. 504–510 (1999)
13. Tsutsui, S., Fujimoto, Y., and Ghosh, A.: Forking Genetic Algorithms: GAs with Search Space Division Schemes. Evolutionary Computation **5**, 61–80 (1997)
14. Bak, P.: How Nature Works. Copernicus, Springer-Verlag, New York, 1st edition, (1996)