UEGO, an Abstract Niching Technique for Global Optimization*

Márk Jelasity

Research Group on Artificial Intelligence
MTA-JATE, Szeged, Hungary
jelasity@inf.u-szeged.hu

Abstract. In this paper, UEGO, a new general technique for accelerating and/or parallelizing existing search methods is suggested. UEGO is a generalization and simplification of GAS, a genetic algorithm (GA) with subpopulation support. With these changes, the niching technique of GAS can be applied along with any kind of optimizers. Besides this, UEGO can be effectively parallelized. Empirical results are also presented which include an analysis of the effects of the user-given parameters and a comparison with a hill climber and a GA.

1 Introduction

In this section a short introduction to the history and motivation behind developing UEGO is given, but first let us state what the acronym means. UEGO stands for *Universal Evolutionary Global Optimizer*. However, it must be admitted from the start that this name is not over-informative, and the method is not even evolutionary in the usual sense. In spite of this we have kept the name for historical reasons.

1.1 Roots

The predecessor of UEGO was GAS, a steady-state genetic algorithm with subpopulation support. For more details on GAS the reader should consult [9].

GAS has several attractive features. Perhaps the most important of these is that it offers a solution to the so-called niche radius problem which is a common problem of many simple niching techniques such as *fitness sharing* ([3] or [4]), *simple iteration* or the *sequential niching* [2]. This problem is related to functions that have multiple local optima and whose optima are unevenly spread throughout the search space. With such functions the *niche radius* cannot be set correctly since if it is too small the search becomes ineffective and if it is too large those local optima that are too close to each other cannot be distinguished. The solution of GAS involves a cooling technique which

* M. Jelasity. UEGO, an abstract niching technique for global optimization. In Agoston E. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature - PPSN V*, volume 1498 of Lecture Notes in Computational Science, pages 378–387. Springer-Verlag, 1998. This work was supported by the Hungarian Soros Fundation and FKFP 1354/1997.

enables the search to focus on the promising regions of the space, starting off with a relatively large radius that decreases as the search proceeds.

However, the authors of GAS came in for a number of criticisms, one being that the algorithm was too much complex, and another that parallel implementation turned out to have many pitfalls associated with it.

1.2 Motivations

Although UEGO is based on GAS there are two major differences that were motivated by the need for a better parallel implementation and the requirement of using domain specific knowledge in an effective way.

The structure of the algorithm has been greatly simplified. As a result the parallel implementation is much easier and the basic ideas become more accessible. This is important because, as the results of the paper will show, UEGO performs similarly or better than the GA and the simple stochastic hill climber (SHC) on our test problems, and at the same time it can be parallelized better than these methods.

The new method is more abstract. The common part with GAS is the species creation mechanism and the cooling method. However, the species creation and cooling mechanism has been logically separated from the actual optimization algorithm, so it is possible to implement any kind of optimizers that work inside a species. This allows the adaptation of the method to a large number of possible search domains using existing domain specific optimizers while enjoying the advantages of the old GAS-style subpopulation approach.

In this paper an SHC is implemented as the optimizer algorithm. This choice is supported by results that show that the performance of the SHC is similar to that of the GA in many cases and sometimes may even be better (e.g. [12, 10, 13, 7]). In [5] a GA with very small population size (1) has been suggested for the graph coloring problem, which is in fact an SHC. Our results confirm that the SHC can indeed outperform the GA at least on the problems and parameter settings we considered.

1.3 Outline of the Paper

Section 2 describes UEGO; the basic concepts, the general algorithm and the theoretical tools that are used to set the parameters of the system based on a few user-given parameters. Section 3 discusses the experimental results that describe the effects of these parameters of the algorithm on the quality of the results and compares UEGO with a simple GA and an SHC. Section 4 then provides a short summary.

2 Description of UEGO

In this section the basic concepts, the algorithm, and the setting of the parameters are outlined. In UEGO, a domain specific optimizer has to be implemented. Wherever we refer to 'the optimizer' in the paper we mean this optimizer.

2.1 Basic Concepts

A key notion in UEGO is that of a *species*. A species can be thought of as a window on the whole search space. This window is defined by its *center* and a *radius*. The center is a solution, and the radius is a positive number. Of course, this definition assumes a *distance* defined over the search space. The role of this window is to localize the optimizer which is always called by a species and can see only its window, so every new sample is taken from there. This means that the largest step made by the optimizer in a given species is no larger than the radius of the given species. If the value of a new solution is better than that of the old center, the new solution becomes the center and the window is moved.

The radius of a species is not arbitrary; it is taken from a list of decreasing radii, the *radius list*. The first element of this list is always the diameter of the search space. If the radius of a species is the ith element of the list, then we say that the *level* of the species is i.

During the process of optimization, a list of species is kept by UEGO. The algorithm is in fact a method for managing this *species-list* (i.e. creating, deleting and optimizing species); it will be described in Section 2.2.

2.2 The Algorithm

Firstly, some parameters of EUGO will be very briefly mentioned more details of which can be found in Section 2.3.

As we mentioned earlier, every species has a fixed level. The maximal value for this level is given by a parameter called `levels`. Every valid level i (i.e. for levels from $[1$,`levels`$]$) has a radius value (r_i) and two fixed numbers of function evaluations. One is used when new species are created at a given level (new_i) while the other is used when optimizing individual species (n_i). To define the algorithm fully, one more parameter is needed: the maximal length of the above-mentioned species list (`max_spec_num`).

The basic algorithm is shown in Figure 1.

```
uego
        init_species_list()
        optimize_species( n[1] )
        for i = 2 to levels
                create_species( new[i]/length(species_list) )
                fuse_species( r[i] )
                shorten_species_list( max_spec_num )
                optimize_species( n[i]/length(species_list) )
                fuse_species( r[i] )
        rof
ogeu
```

Fig. 1. The basic algorithm of UEGO.

Now the procedures called by UEGO will be described.

Init_species_list. Create a new species list consisting of one species with a random center at level 1.

Create_species(evals). For every species in the list create random pairs of solutions in the window of the species, and for every such pair take the middle of the *section* connecting the pair. If the objective function value of the middle is worse than the pair values the members of the pair are inserted in the species list otherwise the species list remains unchanged. Every new species is assigned the actual level value (`i` in Figure 1).

The motivation behind this method is simple: to create species that are on different hills so ensuring that there is a valley between the new species. The parameter of this procedure is an upper bound of the function evaluations. Note that this algorithm needs a definition of section in the search space.

Fuse_species(radius). If the centers of any pair of species from the species list are closer to each other than the given radius, the two species are fused. The center of the new species will be the one with the better function value while the level is the minimum of the levels of the original species.

Shorten_species_list(max_spec_num). Deletes species to reduce the list length to the given value. Higher level species are deleted first.

Optimize_species(evals). Starts the optimizer for every species with the given evaluation number (i.e. every single species in the actual list receives the given number of evaluations). See Section 2.1.

Finally, let us make a remark about a possible parallel implementation. The most time-consuming parts of the basic algorithm is the creation and optimization of the species. Note that these two steps can be done independently for every species, so each species can be assigned a different processor. As our experimental results will clearly show, UEGO performs slightly better than the SHC and the GA even when the number of species is as high as 200.

2.3 Parameters of UEGO

The most important parameters are those that belong to the different levels: the radii and two numbers of function evaluations for species creation and optimalization (see Figure 1). In this section a method is described which sets these parameters using a few easy-to-understand parameters set by the user. In Section 3 further guidelines will be given on the meaning and setting of these remaining user-given parameters.

We will now make use of the notation introduced in Section 2.2. The user-given parameters are listed below. Short notations are also given below that will be used in equations in the subsequent sections.

`evals` (N)**:** The maximal number of function evaluations the user allows for the whole optimization process. Note that the actual number of function evaluations may be less than this value.
`levels` (l)**:** The maximal level value (see Figure 1).
`threshold`: (ν)**:** The meaning of this parameter will be explained later.
`max_spec_num`: (M)**:** The maximal length of the species list.
`min_r`: (r_l)**:** The radius that is associated with the maximal level, i.e. `levels`.

The parameter setting algorithm to be described can use any four of the above five values while the remaining parameters are set automatically.

Speed of the optimizer. Before presenting the parameter setting method, the notion of the *speed* of the optimizer must be introduced. As explained earlier, the optimizer cannot make a larger step in the search space than the radius of the species it is working in. Given a certain number of evaluations, it is possible to measure the distance the given species moves during the optimization process. This distance can be approximated as a function of the radius and evaluations for certain optimizers using mathematical models or experimental results. This naturally leads to a notion of speed that will depend on the species radius and will be denoted by $v(r)$. As we will not give any actual approximations here, the reader should refer to [9].

The parameter-setting method is based on intuitive and reasonable *principles* which are based on personal experience with GAS. Though the parameters are still ad hoc since the principles are ad hoc as well, this method has advantages since these principles are much easier to understand, they can be expressed in human language, and the number of parameters are larger than the number of principles. These principles are now described below.

Principle of equal chance. At a level, every species moves a certain distance from its original center due to optimization. This principle ensures that every species will receive the number of evaluations that is enough to make at least a fixed distance at every level. This common distance is defined by $r_1\nu$. The meaning of `threshold` is now clear: it directly controls the distance a species is allowed to cover, so it actually controls the stability of the resulting species (i.e. the probability that they represent a local optimum). Recall that r_1 is always the diameter of the search space. Now the principle can be formalized:

$$\frac{v(r_i)n_i}{M} = r_1\nu \quad (i = 2, \ldots, l) \tag{1}$$

Principle of exponential radius decreasing. This principle is quite straightforward; given the smallest radius and the largest one (r_l and r_1) the remaining radii are expressed by the exponential function

$$r_i = r_1(\frac{r_l}{r_1})^{\frac{i-1}{l-1}} \quad (i = 2, \ldots, l). \tag{2}$$

Principle of constant species creation chance. This principle ensures that even if the length of species list is maximal, there is a chance of creating at least two more species for each old species. It also makes a strong simplification, that all the evaluations should be set to the same constant value.

$$new_i = 3M \quad (i = 2, \ldots, l) \tag{3}$$

Decomposition of N. Let us define $new_1 = 0$ for the sake of simplicity since new_1 is never used by UEGO. The decomposition of N results in the trivial equation

$$\sum_{i=1}^{l} n_i + new_i = (l-1)3M + \sum_{i=1}^{l} n_i = N \tag{4}$$

making use of (3) in the process. One more simplification is possible too; set $n_1 = 0$ whenever $l > 1$. Note that if $l = 1$ then UEGO reduces to the optimizer it uses for optimizing the species.

Expressing n_i from (1) and substituting it into (4) we can write

$$(l-1)3M + \sum_{i=2}^{l} \frac{Mr_1\nu}{v(r_i)} = N. \tag{5}$$

Using (2) as well, it is quite evident that the unknown parameters in (5) are just the user given parameters and due to the monotonity of this equation in every variable, any of the parameters can be given using effective numeric methods provided the other parameters are known. Using the above principles the remaining important parameters (n_i, new_i and r_i) can be evaluated as well. Note however that some of the configurations set by the user may be infeasible.

3 Experiments

In this section we will discuss the performance of UEGO on an NP-complete combinatorial optimization problem: the subset sum problem. A comparison with a simple GA, GENESIS [6] will be presented. As another result of the experiment the behavior of the parameters of UEGO will be illustrated.

3.1 Problem and Coding

In the case of the subset sum problem we are given a set $W = \{w_1, w_2, \ldots, w_n\}$ of n integers and a large integer M. We would like to find a $V \subseteq W$ such that the sum of the elements in V is closest to, without exceeding, M. This problem is NP-complete. Let us denote the sum of the elements in W by SW.

We created our problem instances in a similar way to the method used in [11]. The size of W was set to 50 and the elements of W were drawn randomly with a uniform distribution from the interval $[0, 10^{12}]$ instead of $[0, 10^3]$ (as was done in [11]) to obtain

larger variance. According to the preliminary experiments, the larger variance of W results in harder problem instances which is important since comparing methods on almost trivial problems makes little sense. The problem instance used here turned out to be so tough that none of the methods employed could find an optimal solution. Based on the results of [8], M was set to $SW/2$. As was shown in [8], this is the most GA-friendly setting, so there is no bias against GENESIS introduced by the problem instance.

We used the same coding and objective function as suggested in [11]. For a solution $(x = (x_1, x_2, \ldots, x_{50}))$,

$$f(x) = -(a(M - P(x)) + (1 - a)P(x))$$

where $P(x) = \sum_{i=1}^{50} x_i w_i$, and $a = 1$ when x is feasible (i.e. $M - P(x) \geq 0$) and $a = 0$ otherwise. Note that the problem is defined as a *maximization* problem.

3.2 The Optimizer and GA Settings

In UEGO, the optimizer was chosen to be a simple SHC as was discussed in the Introduction. In our implementation the SHC works as follows: mutate every bit of the solution with a given probability (but mutating one bit at least), evaluate the new solution and if it is better than or equal to the actual solution, it becomes the new actual solution. This type of SHC worked best in [12], as well. The mutation probability was set at $4/n$ where n is the chromosome length. This value was the same in all the experiments carried out including those with GENESIS. The other GA parameters were a population size of 50, 1-point crossover with probability 1, and elitist selection.

3.3 The Experiments

One of the two main goals of these experiments was to analyze the effects of the user-given UEGO parameters described in Section 2.3. To perform this analysis, several values were chosen for each parameter (see Table 1) then UEGO was run 50 times for every possible combination of these values. This meant that $5 \cdot 4 \cdot 6 \cdot 50 = 6000$ experiments

evals	levels	max_spec_num	treshold	min_r
3000, 10000, 30000, 100000, 300000	2, 3, 5, 10	5, 10, 20, 40, 100, 200	automatically set	fixed to 1

Table 1. The values of the UEGO parameters. Experiments were performed for all combinations.

were performed for one problem instance. Three problem instances were examined but since the results were similar in each case, only one problem instance is discussed below.

Figure 2 shows the effects of the different parameter settings. As the plots are tipical it was inferred that the parameters of UEGO must be fairly robust for this particular

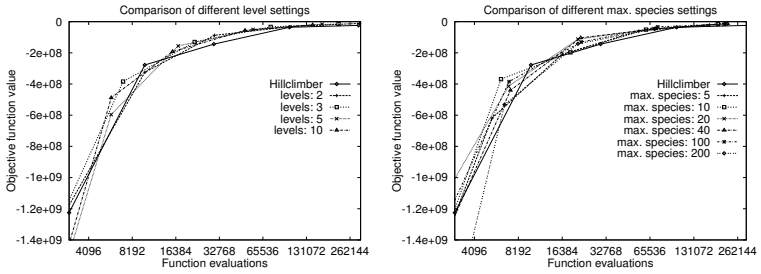

Fig. 2. With the various level settings, `max_spec_num` is 100 and for the differents max. species settings `levels` is 3.

problem class. Some interesting implications of this fact will be touched on in Section 3.4.

The other goal of the experiments was to make a comparision. Figure 3 shows the relevant results. Note that it was difficult to select the best and the worst performance

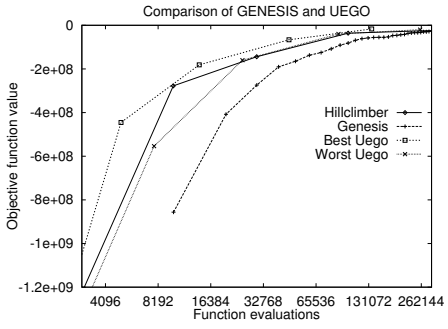

Fig. 3. The parameters for the best UEGO were `max_spec_num`=20 & `levels`=10, and for the worst `max_spec_num`=5 & `levels`=2.

because the curves cross, but the plots give a good approximation. Here SHC is simply UEGO with the setting of `levels`=1.

3.4 Discussion

UEGO *parameters.* As we saw in Figure 2, the parameters seem to be quite robust in this problem, a fact which has rather interesting implications. As was mentioned in Section 2.2, the interaction between the species is minimal so an effective parallel implemetation is possible. This point of view sheds new light on the robustness of the parameters: the larger the number of species the faster the parallel algorithm can be, provided enough processors are available. Good robustness here simply means that we

can increase the speed by as much as a hundred times since every species can be handled by a different processor while the performance remains the same.

Comparison The parameters of GENESIS were not finely tuned; however, the author has some experience with the GA on this problem [8], and it is clear from Figure 3 at least that the GA is more sensitive to parameter setting. However, even if the parameter setting had been badly done, the parallel implementations of GAs cannot provide the speedup that UEGO can. So, after all, it is probably true in this case that the performance of UEGO is superior to that of a simple GA. In the case of SHC similar conclusions can be drawn: the application of the general UEGO technique to the SHC results in an increased quality of the solutions found.

4 Summary

In this paper, UEGO, a general technique for accelerating and/or parallelizing existing search methods was discussed. As was shown, most of the parameters of the system are hidden from the user due to an algorithm for calculating those parameters from a couple of simple parameters. This algorithm is based on *principles* stated in section 2.3 and the *speed* of the applied optimizer. It was also shown that the user-given parameters are robust, at least in the case of the subset sum problem.

Other experimental results were also given, such as the comparison of the technique with a GA and an SHC. It was shown, that UEGO is slightly better than both, and, due to the relative isolation of the species, UEGO should run much better than both of them on a parallel machine since the robustness of the parameters ensures that increasing the number of species does not result in decreasing performance.

5 Acknowledgements and a Note

I would like to thank my reviewers for their criticism, especially the one who — among very useful comments — clearly wrote:

> The paper . . . fails to show any benefit achieved by the new UEGO.

Though the situation is not that simple, there is some truth behind this statement. The coin has another side, however.

UEGO is a result of improving GAS. Every improvement made GAS less and less like a GA just like in [5]. Even the most sceptical readers have to admit that UEGO is not *worse* than the GA so they can interpret this paper as one more call for more comparisons between simple or multi-start stochastic hill climbers and GAs to find and describe the advantages of the later.

References

1. Thomas Bäck, editor. *Proceedings of the Seventh International Conference on Genetic Algorithms*, San Francisco, California, 1997. Morgan Kaufmann.

2. D. Beasley, D. R. Bull, and R. R. Martin. A sequential niche technique for multimodal function optimization. *Evolutionary Computation*, 1(2):101–125, 1993.
3. K. Deb. Genetic algorithms in multimodal function optimization. TCGA report no. 89002, The University of Alabama, Dept. of Engineering mechanics, 1989.
4. K. Deb and David E. Goldberg. An investegation of niche and species formation in genetic function optimization. In J. D. Schaffer, editor, *The Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann, 1989.
5. A. E. Eiben and J. K. van der Hauw. Graph coloring with adaptive genetic algorithms. *Journal of Heuristics*, 4(1), 1998.
6. J. J. Grefenstette. Genesis: A system for using genetic search procedures. In *Proceedings of the 1984 Conference on Intelligent Systems and Machines*, pages 161–165, 1984.
7. Hisao Ishibuchi, Tadahiko Murata, and Shigemitsu Tomioka. Effectiveness of genetic local search algorithms. In Bäck [1], pages 505–512.
8. Márk Jelasity. A wave analysis of the subset sum problem. In Bäck [1], pages 89–96.
9. Márk Jelasity and József Dombi. GAS, a concept on modeling species in genetic algorithms. *Artificial Intelligence*, 99(1):1–19, 1998.
10. A. Juels and M. Wattenberg. Stochastic hillclimbing as a baseline method for evaluating genetic algorithms. Technical report, UC Berkeley, 1994.
11. S. Khuri, T. Bäck, and J. Heitkötter. An evolutionary approach to combinatorial optimization problems. In *The Proceedings of CSC'94*, 1993.
12. M. Mitchell, J. H. Holland, and S. Forrest. When will a genetic algorithm outperform hill-climbing? In J. D. Cowan et al., editors, *Advances in Neural Information Processing Systems 6*. Morgan Kaufmann, 1994.
13. Mutsunori Yagiura and Toshihide Ibaraki. Genetic and local search algorithms as robust and simple optimization tools. In Ibrahim H. Osman and James P. Kelly, editors, *Meta-Heuristics: Theory and Application*, pages 63–82. Kluwer Academic Publishers, 1996.