GAS, a Concept on Modeling Species in Genetic Algorithms*

Márk Jelasity[1] and József Dombi[2]

[1] RGAI, József Attila University, Szeged, Hungary
jelasity@inf.u-szeged.hu
[2] Department of Applied Informatics, József Attila University, Szeged, Hungary
dombi@inf.u-szeged.hu

Abstract. This paper introduces a niching technique called GAS (S stands for species) which dynamically creates a subpopulation structure (taxonomic chart) using a *radius function* instead of a single radius, and a 'cooling' method similar to simulated annealing. GAS offers a solution to the niche radius problem with the help of these techniques. A method based on the *speed* of species is presented for determining the radius function. Speed functions are given for both real and binary domains. We also discuss the sphere packing problem on binary domains using some tools of coding theory to make it possible to evaluate the output of the system. Finally two problems are examined empirically. The first is a difficult test function with unevenly spread local optima. The second is an NP-complete combinatorial optimization task, where a comparison is presented to the traditional genetic algorithm.

1 Introduction

In recent years much work has been done with the aim of extending genetic algorithms (GAs) to make it possible to find more than one local optimum of a function and so to reduce the probability of missing the global optimum. The techniques developed for this purpose are known as *niching techniques*. Besides the greater probability of the success of the algorithm and a significantly better performance on GA-hard problems (see [13]), niche techniques provide the user with more information on the problem, which is very useful in a wide range of applications (decision making, several designing tasks, etc.).

1.1 Best-Known Approaches

Simple iteration runs the simple GA several times to the same problem, and collects the results of the particular runs. **Fitness sharing** has been introduced by Goldberg and Richardson [6]. The fitness of an individual is reduced if there are many other individuals near it and so the GA is forced to maintain diversity in the population. **Subpopulations** can also be maintained in parallel, usually with the allowance of some kind of communication between them (see, for example, [9]). The GAS method has developed from this approach. The **sequential niche technique** is described in [13]. The GA (or any other optimizing procedure) is run many times on the same problem, but after every run the optimized function is modified (multiplied by a derating function) so that the optimum just found will not be located again.

1.2 Problems

These techniques yield good results from several viewpoints, but mention sholud be made of some of their drawbacks, which do not arise in the case of our method, GAS.

* Appeared as: M. Jelasity and J. Dombi (1998) GAS, a Concept on Modeling Species in Genetic Algorithms, *Artificial Intelligence*, Elsevier, Amsterdam, 99(1):pp1-19. This work was supported by the OTKA grant T020150 and the MKM grant 220

Simple iteration is unintelligent; if the optima are not of the same value relatively bad local optima are found with low probability, while good optima are located several times which is highly unnecessary. **Fitness sharing** needs $O(n^2)$ distance evaluations in every step, besides the evaluation of the fitness function. It cannot distinguish local optima that are much closer to each other than the *niche radius* (a parameter of the method); in other words, it is assumed that the local optima are approximately evenly spread throughout the search space. This latter problem is known as the *niche radius problem.* The **sequential niche technique** also involves the niche radius problem. The complexity of the optimized function increases after every iteration due to the additional derating functions. Since the function is modified many times, "false" optima too are found. The method seems difficult to use for combinatorial problems or structural optimization tasks, which are the most promising fields of GA applications.

GAS offers a solution to these problems including the niche radius problem, which is the most important drawback of all of the methods mentioned earlier.

1.3 Outline of the paper

In section 2 we give a brief description of GAS that is needed for an understanding of the following part of the paper. The reader who is interested in more details should refer to the Appendix on how to obtain more information or GAS itself.

In section 4 we give a possible solution to the niche radius problem with the help of the GAS system. Both real and binary problem domains are discussed.

In section 5 we present experimental results. Two problems are examined. The first demonstrates how GAS handles the uneven distribution of the local optima of the optimized function. The second is an NP-complete combinatorial problem, where a comparison is presented to the traditional GA.

2 Species and GAS

2.1 Basic Ideas and Motivations

The motivation of this work was to tackle the problem of finding *unevenly spread* optima of multimodal optimization problems. For this purpose, a subpopulation approach seemed to be the best choice.

The obvious drawback of subpopulation approaches is that managing subpopulations need special algorithms and the system is relatively difficult to understand and maybe to use as well. There are considerable advantages, however. Every subpopulation may have its own attributes that make it possible for them to adapt to the different regions of the fitness landscape. The subpopulations perform effective local search due to the mating restrictions that usually allow breeding only inside of a subpopulation, and the different subpopulations can even communicate with each other.

In our method GAS, every subpopulation (or species) is intended to occupy a local maximizer of the fitness function. Thus, new species are created when it is likely that the parents are on different hills, and species have to be fused when they are thought to climb the same hill (heuristics will be given later). To shed some light on the way GAS copes with unevenly spread optima, it is natural to use a terminology that is well known from the field of simulated annealing. Thus, when illustrating our definitions and methods, we will talk about the 'temperature' of species, the ability of escaping from local optima. In our system, we made the 'temperature' an explicit attribute of every species (it is the *attraction* of species, see Definition 3). This allowed us to offer an algorithm that 'cools down' the system while species of different 'temperatures' are allowed to exist at the same time. The basic idea of the algorithm is that 'warmer' species are allowed to create 'cooler' species autonomously discovering their own local are of attraction.

Finally, let us mention that due to our theoretical results, the large number of parameters of GAS can be reduced to a couple of easy-to-understand ones (see section 4).

2.2 Basic Definitions

Using the notations in the Introduction of [10], let D be the problem domain, $f : D \to \mathbb{R}$ the fitness function and $g : \{0,1\}^m \to D$ for some $m \in \{2,3,\ldots\}$ the coding function. (GAS searches for the *maxima* of f!)

Let us assume that a distance function $d : D \times D \to \mathbb{R}$ and term section (section : $D \times D \to P(D)$, where $P(D)$ is the power set of D) are defined.

Example 1. $D \subseteq \mathbb{R}^m$, D is convex.

$$\text{section}(x,y) = \{z : z = x + t(y-x),\ t \in [0,1]\}$$

Example 2. $D = \{0,1\}^m$ (So if $x \in D$ then $x = (x_1,\ldots,x_m)$)

$$\text{section}(x,y) = \{z : \text{if } x_j = y_j \text{ then } z_j = x_j\}$$

Definition 1. $R : \mathbb{N} \to \mathbb{R}$ *is a* radius function *over D if it is monotonic decreasing, positive,* $R(0) = \max\{\ d(e_1,e_2) : e_1, e_2 \in D\}$ *and*

$$\lim_{n\to\infty} R(n) = 0.$$

Fig. 1a exemplifies these properties. The radius function will be used to control the speed of 'cooling'. In fact, it gives the 'temperature' of the system in a given step (see section 3). This sheds some light on the special requirements we made in Definition 1.

Let us fix a radius function R.

Definition 2. *A* species *s over D is given by the triplet (o,l,S) (notation: $s = (o,l,S)$), where S is a population over D and the members of S are the* individuals *of s; $o(\in S)$ is the* center *of s and is such that $f(o) = \max f(S)$; $l(\in \mathbb{N})$ is the* radius index *or the* level *of s, and so the* radius *of s is $R(l)$. Recall that in GAs a population is a multiset (or bag) of individuals (e.g. $S =< x_1, x_1, x_2 >$).*

Definition 3. *$s = (o,l,S)$ is a species. Let $A(s) = \{a \in D : d(a,o) \le R(l)\}$ be the* attraction *of s.*

Fig. 1b illustrates the terms defined above.

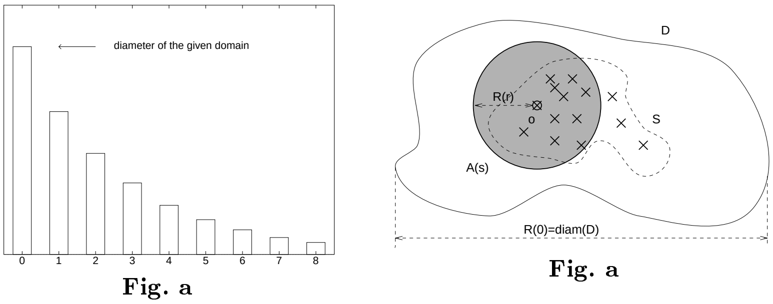

Fig. 1. a: A possible radius function. b: Terms related to species.

Species with small attraction behave as they were 'cooler'; they discover a relatively small area, their motion in the space is slower but they can differentiate between local optima that are relatively close to each other. Note that for a species $s = (o,l,S)$, o is 'almost' determined by S. If the maximal number of different maximizers in a population would be one, Definition 2 would be redundant. Also note that it is not necessary that $S \subseteq A(s)$.

Definition 4. *Let T be a graph with the vertex set $V(T)$, where $V(T)$ is a set of species over D. T is a* taxonomic chart *(t.c.) if T is a tree and there is an $s_r = (o_r, 0, S_r)$ root in T, and if $(s_r =)s_0, s_1, \ldots, s_n$ is a path in T, then for the corresponding levels $l_0 < l_1 < \ldots < l_n$ holds.*

Note that the root s_r of of every t.c. has the level 0 which means that its area of attraction $A(s)$ covers the whole domain D (see Definitions 1 and 3).

2.3 The Algorithm

Let $V(T_0)$ (T_0 is a t.c.) contain only $s_r = (o_r, 0, S_r)$, where S_r is randomly chosen. The algorithm in Fig. 2 shows how GAS creates a T_{n+1} t.c. from a given T_n t.c.

```
procedure activity
begin
   while (population size of T_n < maximum allowed) do begin
      choose two parents
      create two offspring
      place the parents and the offspring back in the population
   end
   dying_off
   fusion
end
```

Fig. 2. The basic algorithm that creates T_{n+1} from T_n.

Before describing the parts of the algorithm, we should make a few remarks.

- It is the flexibility of steady state selection [11] that allows the algorithm to create and manage species, as will be shown later.
- The algorithm can be implemented in parallel on two levels: the level of the `while` cycle and the level of the `procedure`. (However, our implementation is not parallel.)

Let us now examine the parts of the algorithm.

Population Size. The population size of a given T t.c. is $\sum_{s=(o,l,S)\in V(T)} |S|$.

Choose Two Parents. From a given T, we first choose a vertex (a species) with a probability proportional to the number of the elements of the vertices. Then, we choose two parents from this species, using the traditional probability (proportional to the fitnesses of the elements of the species).

Create Two Offspring. From individuals p_1 and p_2, we create p_1' and p_2' by applying onepoint crossover and mutation operators to the parents.

Placing Elements Back in the Population. Since this is the point where new species are created, this is the most important step. We have to decide here whether to separate the given two parents into two different species and we have to find species for the two newly created offspring. If we decide to separate the parents, we must find new existing species for them or create new species for them. The placing-back algorithm is shown in Fig. 3. The notations of the algorithm: p_1, p_2 are the parents, p_1', p_2' are the two offspring, e is a random point on the section that connects p_1

```
if f(e) < f(p1),f(p2) then
   for x=p1,p2,p1',p2' do
      if (there is a child node s_c of s_p such that x is in A(s_c)) then
         move(x,s_c)
      { With the restriction that p1 and p2    }
      { must not be put into the same species. }
   for x=(a parent not put in so far) do
      create a new child s=(x,max{l_p +1,strict},<x>) for s_p
{ else: The parents are left in s_p. }

for x=(an offspring not put in so far) do begin
   s:=s_p; while (x is not in A(s)) do s:=father node of s
   { if s=s_r then A(s)=D! }
   move(p,s)
end
```

Fig. 3. The algorithm that places parents and offspring back in the population.

and p_2 (note that p_1' and p_2' are not on this section in general), and s_p is the original species of the parents. We always mean $s_x = (o_x, l_x, S_x)$ on s_x for any symbol x.

Function `move(p,s)` moves `p` to `S` and updates `o` if necessary. Parameter `strict` determines the precision of the search, i.e. the 'temperature' of the system. Increasing `strict` decreases 'temperature'. The way of using this parameter is described in section 3.

It is clear that for a concave or for a unimodal one-dimensional fitness function GAS will never create a single species.

Dying_off. `Dying_off` deletes as many elements from the population of the t.c. as were inserted in the `while` cycle keeping the population size constant. The method used for selecting elements to die is based on the ranking defined by the transformed fitness function $\hat{f}$:

$$\hat{f}(e) := \frac{f(e) - (\text{a global lower bound of } f \text{ on the whole population})}{|S|}$$

where e is in species $s = (o, l, S)$.

This means that species of small size have more chance to survive (and to grow). The precision of the procedure (i.e. the level of competition) can be varied during the optimization process. In section 3 we discuss how to use this possibility. `Dying_off` has no effect on the species structure (by definition) and does not delete the best individual of a species.

Fusion. The result of `fusion` depends on R and `strict` described earlier. After executing `fusion` for a given T t.c., we get T', for which the following will be true: if $s_1,\ s_2 \in V(T')$, then $d(o_1, o_2) \geq R(\text{strict})$.

`Fusion` simply unites species of T that are too close to each other, and `strict` tells it what is too close. If s_1 and s_2 are united, the result species is $s = (o, \min\{l_1, l_2\}, S_1 \cup S_2)$, where $f(o) = \max\{\ f(o_1),\ f(o_2)\}$ and o is o_1 or o_2. In view of the tree structure, the species with the lower level absorbs the other. If the species have the same level, either of them may absorb the other.

3 Optimization with GAS

For global optimization with GAS, we suggest the algorithm shown in Fig. 4. For determination of the vector of evaluation numbers x and the radius function R, we suggest a method in section 4 based on the *speed* of species with a given radius in a given domain.

```
create a starting t.c. T_0
for strict=1 to ST_m               { 0 < ST_m(=strict_max) < 8 }
   new species
   evolution                       { evolution is the following macro: }
   stabilize                       {      for i=1 to 10 do activity     }
   iterate evolution until reaching {     immigration                   }
   x_strict function evaluations   {      for i=1 to 5  do activity     }
```

Fig. 4. The high-level test algorithm.

The main `for` cycle performs the 'cooling' operation. Increasing `strict` results in new species with smaller radii (see Fig. 3). The basic philosophy is to increase diversity at the beginning of every cycle and then perform optimization of the newly discovered areas. This kind of oscillation can be observed in biological systems as well.

We now describe the species-level genetic operators, used in the algorithm shown in Fig 4.

Immigration. For every species $s = (o, l, S)$ in a given t.c., $|S|/2$ randomly generated new individuals are inserted from $A(s)$. Immigration refreshes the genetic material of the species and makes their motion faster. It has a kind of greasing effect.

New species. This switch alters the state of the system towards managing species creation. It randomizes `dying_off` and relaxes competition by decreasing the lower bound of the fitness function, and so decreases the relative differences between individuals. According to some biologists [3], species are born when the competition decreases; our experiments support this opinion.

Stabilize. The effect of this is the contrary of `new species`. It prohibits the creation of new species and increases competition.

As a summary, we give here some heuristical arguments that support the subpopulation structure approach and use a radius function instead of a single radius.

- The number of distance calculations grows with the size of the t.c. instead of the size of the population.
- Application of species–level operators (e.g. fusion, immigration) becomes possible.
- Lower-level (closer to root) species manage to create new species in their attraction.
- The advantages of the technique based on the radius function and increasing `strict` (see Fig. 4) are similar to those of the 'cooling' technique in the simulated annealing method.

Finally, to make our discussion more rigorous, we give the definitions of stability of species and t.c. These definitions are not really necessary for the present discussion in the sense that will not be used in any strict mathematical environment. However, when stability is mentioned, it is ment in this sense. The impatient reader is free to skip these definitions.

Definition 5. *$W \subseteq D$. Species s is* stable in W *if $o \in W$, and if $o_1, o_2, \ldots$ is a series of new centers inserted by GAS to s during running then it is impossible that for some i $o_i \notin W$.*

Example 3. It is clear that s is stable in $W = \{e \in D : f(e) > f(o)\}$.

Definition 6. *$e_0 \in D$, e_0 is a local optimum (with respect to d) of f. s is* stable around e_0 *if, for every $o_1, o_2, \ldots$ series of new centers inserted by GAS to s during running, $o_n \to e_0$ $(n \to \infty)$ with probability 1.*

Example 4. $W \subseteq D$, $e_0 \in W$. If s is stable in W and e_0 is the global optimum of f in W (i.e. e_0 is a local optimum of D or else s could not be stable in W) and there are no more optima of f in W, then s is stable around e_0. This example would need a proof but we do not give it here because it is marginal from the viewpoint of the paper.

Definition 7. *T is a t.c. T is* stable *if every species of T is stable around distinct local optima of f.*

Definition 8. *T is a t.c. T is* complete *if T is stable and there is exactly one stable species around every local optimum of f.*

4 Theoretical Results

In this section we discuss the theoretical tools and new terms that can be used due to the exact definition of the t.c. data structure and GAS algorithm.

4.1 Speed of Species

We do *not* assume that the optima of the fitness function are evenly spread; we create species instead that "live their own lives" and can *move* in the search space and find the niche on which they are stable. It can be seen that from this point of view determining the radius function R depends more upon the *speed* of the species than on the number of spheres (niches) of a given radius that can be packed into the space. The speed of a given species $s = (o, l, S)$ will depend on its radius $R(l)$. The larger the radius is the faster the species can move towards its stable state so the fewer the number of iterations it needs to become stable. This idea will be used when simultaneously dividing the available number of function evaluations among the species and setting the values of the radius function R.

The solution of the sphere packing problem mentioned above is the base of setting the niche radius parameter of the methods mentioned in the Introduction. This value is useful when evaluating the output of the system since it tells us what percentage of the possible number of optima we have found. In section 4.3 we discuss such packing problems in the case of binary domains.

Real Domains. In real domains we have $D \subseteq \mathbb{R}^n$ for some $n \in \mathbb{N}$. Let us fix a dimension number n and a species $s = (\boldsymbol{o}, l, S)$ and let us denote the radius of s by r (i.e. $r = R(l)$). (Recall, that for a species $s = (\boldsymbol{o}, l, S)$ the center of s, $\boldsymbol{o}$, is an n-dimensional real vector: $\boldsymbol{o} = (o_1, \dots, o_n)$.)

The following suggestion for the definition of *speed* is an approximation. It is assumed that the fitness function f is the projection $f(\boldsymbol{x}) = x_1$ and GAS simply selects new individuals from the attraction of s, $A(s)$, randomly with a uniform distribution instead of generating them using parents and genetic operators and drops them into the species one by one. The speed for a radius r and a dimension n will be the average step size towards the better region.

Definition 9. *The* speed $v(r)$ *of s is $(c_1 - o_1)/2$, where $\boldsymbol{c} \in \mathbb{R}^n$ is the center of gravity of the set*

$$S_{n,r} = A(s) \cap \{\boldsymbol{x} \in \mathbb{R}^n : x_1 > o_1\}$$

In other words, let us choose a random element $\boldsymbol{x}^* = (x_1^*, \dots, x_n^*)$ from $A(s)$ with a uniform distribution. Let $\xi = o_1 - x_1^*$ if $o_1 > x_1^*$, and $\xi = 0$ otherwise. Than $M(\xi)$ (the expected value of ξ) is $v(r)$. This means that $v(r)$ is given by the equation

$$v(r) = \frac{1}{2} \frac{1}{V(S_{n,r})} \int_{S_{n,r}} x_1 \; dx_1 \dots dx_n \tag{1}$$

where $V(S_{n,r})$ is the volume of $S_{n,r}$. (Recall that if $\xi = 0$ then the center of s o is not changed by GAS.) It can be proved that

$$v(r) = \frac{\binom{n}{\frac{n-1}{2}}}{2^{n+1}} r \tag{2}$$

holds. In the general case (if n is even) (2), is defined with the help of the function $\Gamma(t+1)$, the continous extension of $t!$. $\Gamma(t+1) = \int_0^\infty x^t e^{-x}\, dx$, $\Gamma(1/2+1) = \sqrt{\pi}/2$ and $\Gamma(t+2) = (t+1)\Gamma(t+1)$.

Binary Domains. Let $D = \{0,1\}^n$. Let us fix a dimension number n and a species $s = (o, l, S)$ and let us denote the radius of s by r (i.e. $r = R(l)$). We give a definition of speed similar to Definition 9. Like in the case of real domains, an approximation is used. It is assumed that the fitness of an individual $x \in D$ is given by the number of 1s in it, and GAS works as described in the case of real domains. As in the binary case, the speed for a radius r and a dimension n will be the average size of the first step of o after receiving one random individual. The difference is that in the case of binary domains, the starting center has to be fixed too since the average step sizes change as the center changes. Let $e \in D$ such that the number of 0s is equal to or greater by one than the number of 1s. Let e be the fixed starting center. Let

$$S_{n,r} = \{e' \in D \ : \ d(e', e) \leq r\}$$

where d is the Hamming distance (the sum of the bit differences). Let us choose a random e^* from $S_{n,r}$ with a uniform distribution and let $\xi = d(e^*, e)$ if there are more 1s in e^*, and let $\xi = 0$ otherwise.

Definition 10. *Let $v(r) = M(\xi)$ be the* speed *of species s in D if the radius of s is r.*

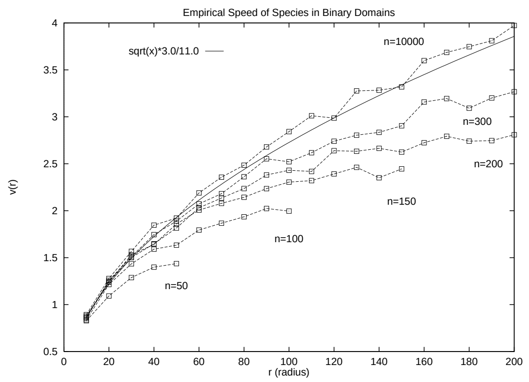

Fig. 5. Speed in binary domains.

We performed experiments to determine $v(r)$ (Fig. 5). It can be seen if $n \gg r$, then the equation

$$v(r) = \frac{3}{11}\sqrt{r} \tag{3}$$

seems to describe the speed. If r approaches n, the growing of the speed becomes slower than (3) would indicate.

4.2 Determining R and x

We use the notations of Fig. 4 here. Recall that ST_m is the number of steps in the 'cooling' procedure, the maximal value of `strict`, and x_i denotes the number of function evaluations at step i. Let us assume that the evaluation number N, the domain type and the corresponding speed function v are given. We know that

$$\sum_{i=1}^{ST_m} x_i = N \tag{4}$$

We suggest a setting for which the system of equations

$$v(R(i))x_i = C \quad (i = 1, \ldots ST_m) \tag{5}$$

holds where C is a constant (independent of i). This simply means that the species of the different levels receive an equal chance to become stabilized. From (4) and (5) it follows that

$$C = \frac{N}{\sum_{i=1}^{ST_m} \frac{1}{v(R(i))}}. \tag{6}$$

We note that C is the distance that a species of level i expectedly crosses during x_i iterations.

In GAS, the upper bound M of the number of species can be set. $M =$ [population size/4] by default. Now we can give the value of C:

$$C = R(0)M\nu \tag{7}$$

Recall that $R(0)$ is the diameter of the domain we examine.

ν is a threshold value. Setting $\nu = 1$ means that every species receives at least sufficient function evaluations for crossing the whole space, which makes the probability of creating a stable t.c. very high. In section 5 we examine the effect of several different settings of ν. Finally let

$$R(i) = R(0)\beta^i \quad (i = 1, \ldots ST_m,\ \beta \in (0, 1)) \tag{8}$$

Then, R is a valid radius function and subproblems defined by the species will be similar in view of the radii.

Using (6), (7) and (8), we can write

$$\frac{N}{R(0)M\nu} = \sum_{i=1}^{ST_m} \frac{1}{v(R(0)\beta^i)} \quad (\beta \in (0, 1)) \tag{9}$$

where everything is given except β.

Since v is monotonous, the right side of (9) monotonically decreases as β increases and so reaches its minimum if $\beta = 1$. Using this fact, the feasibility condition of (9) is

$$\frac{N}{R(0)M\nu} > \frac{ST_m}{v(R(0))} \tag{10}$$

If (10) holds, (9) has exactly one solution. This property allows us to use effective numeric methods for approximating β.

In section 5.1 we discuss the parameters that have to be set in GAS.

4.3 Evaluating the Output

We based the setting of the parameters of GAS on the speed function. However, it is important to know the maximal possible size of a t.c. for a given radius function R (assuming an arbitrary large evaluation number and population size) since it tells us what percentage of the maximal possible number of optima we have found.

The problem leads to the general sphere packing problem and this has been solved neither for binary nor for real sets in the general case.

Real Case

In n-dimensional real domains Deb's method [5] can be used.

$$p = (\frac{\sqrt{n}}{2r})^n$$

where r is the species radius, the domain is $[0,1]^n$ and p is the number of optima, assuming that they are evenly spread. We note that this is only an approximation.

Binary Case

Results of coding theory can be used to solve the packing problem in binary domains since it is one of the central problems of this field. We will need the definition of binary codes.

Definition 11. *$d, n \in \mathbb{N}$, $d \leq n$. $C \subseteq \{0,1\}^n$ is a $(n, |C|, d)$* binary code *if $\forall c_1, c_2 \in C$: $dist(c_1, c_2) \geq d$ (The function "dist" is the Hamming distance, the sum of the bit differences.)*

Definition 12. *$d, n \in \mathbb{N}$. $A(n, d) := \max\{|C| \ : \ C$ is a $(n, |C|, d)$ binary code$\}$.*

$A(n, d)$ has not yet been calculated in the general case; only lower and upper bounds are known. Such bounds can be found for example in [12], [2] or [1]. One of these is the Plotkin bound:

Theorem 1. *((Plotkin bound)) For $d, n \in \mathbb{N}$, we have*

$$A(n, d) \leq \frac{d}{d - \frac{1}{2}n} \quad \text{if} \ \ d \geq \frac{1}{2}n$$

Proof. [12].

In a special case, the exact value is also known:

Theorem 2. *For binary codes and $m \in \mathbb{N}$, we have*

$$A(2^{m+1}, 2^m) = 2^{m+2}.$$

In Table 1 we show the Plotkin upper bounds for $2n = 32, 128$ and 1024. The values have been calculated according to the following formulas:

$$\begin{array}{l} A(2n, n+a) \leq \frac{n+a}{n+a-2n/2} = \frac{1}{a}(n+a) \\ A(2^{m+1}, 2^m) = 2^{m+2} \\ A(2n, n-a) \leq 2^{2a+1} 2(n-a) \end{array}$$

5 Experimental Results

In this section we examine two problems. The first demonstrates how GAS handles the uneven distribution of the local optima of the optimized function. The second is an NP-complete combinatorial problem, where a comparison is presented to the traditional GA.

d 2n	32	128	1024
n-3	3 328	15 616	130 304
n-2	896	3 968	32 640
n-1	240	1 008	8 176
n	64*	256*	2048*
n+1	17	65	513
n+2	9	33	257
n+6	3	11	86
n+16	2	5	33
n+40	-	2	13

Table 1. Plotkin upper bounds for $A(2n, d)$. The indicated values are exact.

5.1 Setting of GA and GAS Parameters

In the following experiments, the settings of the traditional GA parameters are P_m (mutation probability) = 0.03 (see e.g. [7]) and P_c (crossover probability) = 1, while the population size = 100. In the `while` cycle of the basic algorithm (shown in Fig. 2), the maximum allowed population size is 110. For continuous domains, we used Gray coding as suggested in [4].

The settings of the specific GAS parameters are the following:

- R (radius function) and x (evaluation numbers) can be determined using the method described in section 4.
- M (maximal number of species in the t.c.) is set to M = (pop. size)/4. Setting a larger value is not recommended since too many small species could be created.
- N ($\sum_{i=1}^{ST_m} x_i$) depends on the available time and computational resources. We used $N = 10^4$.
- ν (treshold) and ST_m (maximal strict level) are the parameters we tested so we used several values (see the descriptions of the experiments).

For simplicity, we run `evolution` only once after `new species` (see Fig. 4) but we note that increasing that number can significantly improve the performance in some cases. The cost of one `evolution` is 275 evaluations after `new species`, and 200 after `stabilize` at the above settings.

5.2 A Function with Unevenly Spread Optimas

The problem domain D is $[0, 10]$. The fitness function $f : D \to \mathbb{R}$.

$$\mathbf{a} = \begin{pmatrix} 3.040 \\ 1.098 \\ 0.674 \\ 3.537 \\ 6.173 \\ 8.679 \\ 4.503 \\ 3.328 \\ 6.937 \\ 0.700 \end{pmatrix} \quad \mathbf{k} = \begin{pmatrix} 2.983 \\ 2.378 \\ 2.439 \\ 1.168 \\ 2.406 \\ 1.236 \\ 2.868 \\ 1.378 \\ 2.348 \\ 2.268 \end{pmatrix} \quad \mathbf{c} = \begin{pmatrix} 0.192 \\ 0.140 \\ 0.127 \\ 0.132 \\ 0.125 \\ 0.189 \\ 0.187 \\ 0.171 \\ 0.188 \\ 0.176 \end{pmatrix} \qquad f(x) = \sum_{i=1}^{10} \frac{1}{(k_i(x - a_i))^2 + c_i}$$

f (shown in Fig. 6) is a test function for global optimization procedures suggested in [8].

We have determined R and x for $\nu = 1/4, 1/2, 3/4$ and 1 (see Table 2). ST_m is 8 in every case. Recall that according to the algorithm in Fig. 4 the elements of x must be divisible by 200 (the cost of `evolution` after `stabilize`) and the sum of them must be $10^4 - ST_m \cdot 275$.

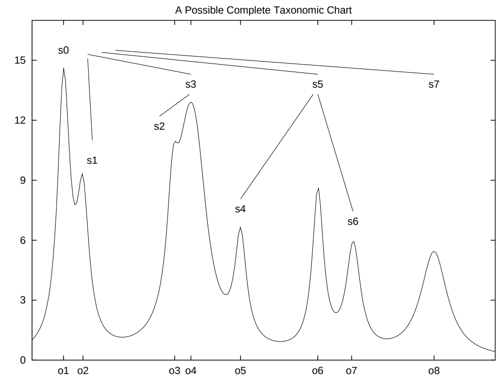

Fig. 6. The function with unevenly spread optima.

	$\nu = 1$		$\nu = 3/4$		$\nu = 1/2$		$\nu = 1/4$	
	R	x	R	x	R	x	R	x
1	6.61	200	6.333	200	5.978	0	5.334	0
2	4.369	200	4.011	200	3.573	200	2.845	0
3	2.888	400	2.54	200	2.136	200	1.517	200
4	1.909	600	1.609	400	1.277	400	0.809	200
5	1.261	800	1.019	800	0.763	600	0.432	600
6	0.834	1200	0.645	1200	0.456	1200	0.23	1000
7	0.551	1800	0.409	1800	0.273	2000	0.123	2000
8	0.364	2800	0.259	3000	0.163	3200	0.066	3600

Table 2. Radius and evaluation numbers for $\nu = 1/4, 1/2, 3/4$ and 1.

We run the corresponding algorithms 100 times. The numbers of stable species that converged to one of the local optima are shown in Table 3. The most important result is that *no* unstable species appeared in the output even for $\nu = 1/4$.

The best results are observed in the case of $\nu = 1/4$. Here, even o_3 was found 2 times in spite of its very small attraction.

	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8
$\nu = 1$	100	0	0	100	60	97	48	94
$\nu = 3/4$	100	1	0	100	65	87	72	94
$\nu = 1/2$	100	34	0	100	74	99	58	98
$\nu = 1/4$	100	25	2	100	85	100	90	100

Table 3. Number of stable species around the local optima.

Fig. 7 shows the average number of species detected before increasing `strict` (after stabilizing for the old `strict`). From these values, we can gain information on the structure of the optima of the fitness function. For example, for radii greater than 3, very few species were created, which means that the optima are probably closer to each other than 3.

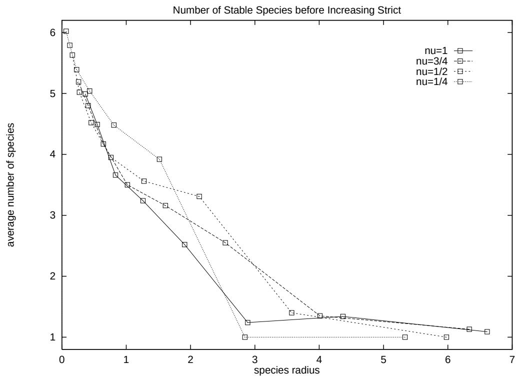

Fig. 7. The species number increasing history (average of 100 runs).

5.3 An NP–complete Combinatorial Problem

We study the subset sum problem here. We are given a set $W = \{w_1, w_2, \ldots, w_n\}$ of n integers and a large integer C. We would like to find an $S \subseteq W$ such that the sum of the elements in S is closest to, without exceeding, C. This problem is NP-complete.

We used the same coding and fitness function as suggested in [14]: $D = \{0,\ 1\}^{128}$. If $e \in D$ ($e = (e_1, e_2, \ldots, e_{128})$), then let $P(e) = \sum_{i=1}^{128} e_i w_i$, and then

$$-f(e) = a(C - P(e)) + (1 - a)P(e)$$

where $a = 1$ when e is feasible ($(C - P(e)) \geq 0$), and $a = 0$ otherwise.

Here, $\forall w \in W\ 1 \leq w \leq 1000$ and C is the sum of a randomly chosen subset of W (every element is chosen with a probability 0.5).

We do not need a coding function here since D is the code itself.

	$ST_m = 1$		$ST_m = 2$		$ST_m = 3$		$ST_m = 4$		$ST_m = 5$		$ST_m = 6$	
	R	x	R	x	R	x	R	x	R	x	R	x
1	2	9600	20	2600	47	1800	72	1400	93	1200	109	1200
2			3	6800	17	2800	41	1800	67	1400	92	1200
3					6	4600	23	2400	49	1600	79	1400
4							13	3200	35	2000	67	1400
5									26	2400	57	1600
6											48	1600

Table 4. Radii and evaluation numbers for $ST_m = 1, 2, \ldots, 6$.

We tested several values of ST_m. Table 4 shows R and x for $ST_m = 1, 2, \ldots, 6$. The value 8 is not feasible and 7 is also very close to that bound. $\nu = 1$ in every case. We run the corresponding algorithms 50 times.

For comparison, in experiments on the same problem with two times more (i.e. $2 \cdot 10^4$ instead of 10^4) evaluation numbers in [14], 0.93 optimal solutions were found per run. Here, this value is at least one for every ST_m, and for $ST_m = 2$ it is 2.62 (see Table 5).

Besides this, many near-optimal solutions were found (as shown in Fig. 8) so we received much more information with only 10^4 function evaluations.

Fig. 8. Number of near-optimal solutions found during the 50 runs.

ST_m	opt. found/ run	avg. fitness of all spec.	number of species
1	1.56	-3.194	201
2	2.62	-15.616	1250
3	2.1	-10.866	1250
4	2.12	-12.568	1238
5	2.08	-12.84	846
6	1.0	-6.943	211

Table 5. Result of the experiment (50 runs).

6 Summary

In this paper we have introduced a method called GAS for multimodal function optimization (or multimodal heuristic search). GAS dynamically creates a subpopulation structure called a taxonomic chart, using a radius function instead of a single radius, and a 'cooling' method similar to simulated annealing.

We based setting of the parameters of the method on the speed of species instead of their relative size to the search space and we gave speed functions for both real and binary domains.

We performed experiments for a difficult test function with unevenly spread local optima and for an NP-complete combinatorial problem.

In both cases our results are encouraging though much work will have to be done to examine the effects of the parameters of the method more thoroughly.

7 Acknowledgements

I would like to thank my reviewers for their constructive criticism; it certainly made the paper more correct and much easier to follow. Of course, the remaining errors are ours.

Appendix

GAS and a detailed description of the system is available via anonymous ftp at the following URL:

`ftp://ftp.jate.u-szeged.hu/pub/math/optimization/GAS`

This is a directory. Please read the file `readme`.

The authors would highly appritiate it if you informed them about any problems regarding GAS (compilling, using, etc.).

References

[1] F.J. MacWilliams, N.J.A. Sloan (1977) *The Theory of Error-correcting Codes.* North Holland.

[2] R.J. McEliece (1977) *The Theory of Information and Coding.*, Addison-Wesley.

[3] V. Csányi (1982) *General Theory of Evolution.* Publ. House of the Hung. Acad. Sci. Budapest.

[4] R.A. Caruana, J.D. Schaffer (1988), Representation and hidden bias: Gray vs. binary coding for genetic algorithms, in *Proceedings of the fifth international conference on machine learning* (editor: John Laird), pp153-161.

[5] K. Deb (1989) *Genetic algorithms in multimodal function optimization.* Masters thesis, TCGA Report No. 89002. The University of Alabama, Dep. of Engineering mechanics.

[6] K. Deb, D.E. Goldberg (1989) An investegation of niche and species formation in genetic function optimization. In *Proceedings of the Third ICGA* (pp42-50). Morgan Kaufmann.

[7] D. E. Goldberg (1989), *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley, `ISBN 0-201-15767-5`.

[8] A. Törn, A. Žilinskas (1989), *Global Optimization*, in Lecture Notes in Computational Science 350. Springer-Verlag.

[9] Y. Davidor (1991) A naturally occuring niche and species phenomenon: the model and first results. In *Proceedings of the Fourth ICGA* (pp257-263). Morgan Kaufmann.

[10] G.J.E. Rawlins (ed.) (1991) *Foundations of Genetic Algorithms.*, Morgan Kaufmann.

[11] G. Syswerda (1991) A Study of Reproduction in Generational and Steady State Genetic Algorithms. in [10].

[12] J.H. van Lint (1992) *Introduction to Coding Theory.*, Springer-Verlag.

[13] D. Beasley, D.R. Bull, R.R. Martin (1993) A Sequential Niche Technique for Multimodal Function Optimization. *Evolutionary Computation* **1**(2), pp101-125, MIT Press.

[14] S. Khuri, T. Bäck, J. Heitkötter (1993), An Evolutionary Approach to Combinatorial Optimization Problems, in *The Proceedings of CSC'94*.

[15] M. Jelasity, J. Dombi (1995) GAS, an Approach to a Solution of the Niche Radius Problem. In *The proceedings of GALESIA'95.*