Genetic Algorithms

Genetic Algorithm Theory

Applications of Genetic Algorithms

Genetic Programming

Parallel Implementations of Genetic Algorithms

Global Convergence of Genetic Algorithms: a Markov Chain Analysis

A.E. Eiben [1]

E.H.L. Aarts[1, 2]

K.M. Van Hee [1]

[1] Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513 5600 MB Eindhoven, The Netherlands

[2] Philips Research Laboratories
P.O. Box 80000 5600 JA Eindhoven, The Netherlands

ABSTRACT

In this paper we are trying to make a step towards a concise theory of genetic algorithms (GAs) and simulated annealing (SA). First, we set up an abstract stochastic algorithm for treating combinatorial optimization problems. This algorithm generalizes and unifies genetic algorithms and simulated annealing, such that any GA or SA algorithm at hand is an instance of our abstract algorithm. Secondly, we define the evolution belonging to the abstract algorithm as a Markov chain and find conditions implying that the evolution finds an optimum with probability 1. The results obtained can be applied when designing the components of a genetic algorithm.

INTRODUCTION

The underlying idea – and the name – of genetic algorithms Holland (1975), DeJong (1980), Goldberg (1989) and simulated annealing Kirkpatrick, Gelatt & Vecchi (1983), Aarts & Korst (1989) originates from nature. GAs and SA have several applications in learning or classifier systems, Grefenstette (1985,1987), Schaffer (1989), but primarily they are search algorithms to handle combinatorial optimization problems, Papadimitriou & Steiglitz (1982). Since the seventies an

enormous amount of experiments has been carried out but relatively little effort has been made in investigating the foundations of GAs. Our major aim by this paper is to identify the basic components of GAs in general and to determine how these components should be in order to guarantee that a given GA finds an optimum. We are trying to do this as general as possible, therefore we first define an abstract genetic algorithm (AGA) and establish convergence results for the AGA. Notice that following the terminology of approximation algorithms we use the word 'convergence' for 'tending to an optimum'. This should not lead to confusion with another possible use of this word, where 'convergence' stands for 'tending to a uniform population', see Goldberg & Segrest (1987).

1 THE ABSTRACT GENETIC ALGORITHM

In this section we exploit the following raw version of a genetic algorithm.

Make initial population no mutation,crossover just parenting
WHILE NOT *stop* DO
  BEGIN
    Choose parents from the population
    Let the selected parents ***Produce*** children
    Extend the population by adding the children to it
    Select elements of the extended population to survive for the next cycle
  END
Output the optimum of the population

We restrict the application domain to minimization problems, where a minimum of the object function $f \in S \longrightarrow \mathbb{R}$ over the finite search space S is required. According to the GA terminology we refer to the elements of S as individuals, a set of individuals is called a population. The final algorithm is a stochastic, local search method. The stochastic feature is maintained by so called 'random parameters', locality is due to the neighbourhoods on S. In the sequel we give the exact definitions of neighbourhoods on S, parent–lists (groups of individuals capable of producing offspring), choice function, production function, selection function and evaluation function then we precisely define the Abstract Genetic Algorithm (AGA).

Formally, we shall use lists for populations, i.e. if S^* stands for the set of finite lists over S, then a population is an element of S^*. We use the common set operators $\subseteq$, $\cup$, $\cap$ and relations like $\in$ also for lists meaning the trivial definitions behind them.

A *neighbourhood function* is a function that assigns neighbours to each individual in S:

$$N : S \longrightarrow S^*.$$

A *parent-list* is a list of individuals that is capable of producing offspring, $P \subseteq S^*$ denotes the set of all parent–lists.

To incorporate probabilities we introduce parameter sets to randomize important functions of the algorithm. What does it mean to 'randomize' a function $f : X \longrightarrow Y$? Strictly speaking we take $\mathcal{F}$, the set of all functions from X to Y, a probability space $(\Omega,\mathcal{A},\mathbb{P})$ and a random variable $g : \Omega \longrightarrow \mathcal{F}$. Then $g(\omega) : X \longrightarrow Y$ is a function with the desired signature for any $\omega \in \Omega$. Therefore we call ω a *random parameter* and take g as the 'randomized' f denoting $g(\omega)(x)$ by $f(\omega,x)$.

We introduce the sets A, B, C and three parameters $\alpha \in A$, $\beta \in B$, $\gamma \in C$ that are chosen by random drawings.

A *choice function* $f_c : A \times S^* \longrightarrow \mathcal{P}(P)$ is to choose a set of parent–lists from a population such that

$$\forall\, \alpha \in A\ \forall\, x \in S^*\ \forall\, y \in f_c(\alpha,x) : y \subseteq x.$$

A *production function* is to produce the children of a parent–list. Formally it is a function $f_p : B \times P \longrightarrow \mathcal{P}(S)$, such that all the children are from the neighbourhood:

$$\forall\, \beta \in B\ \forall\, x \in P : f_p(\beta,x) \subseteq \bigcup_{s \in x} N(s).$$

A *selection function* is reducing the extended population by selecting individuals that may survive for the next generation. It is a function $f_s : C \times S^* \longrightarrow S^*$, such that

$$\forall\, \gamma \in C\ \forall\, x \in S^* : f_s(\gamma,x) \subseteq x.$$

An *evaluation function* is a function $f_e : S^* \longrightarrow \{true, false\}$ which might depend on some external parameter too (eg. the number of iterations) but here we do not denote this dependence.

Let S be a finite set, f be an object function on S, N be a neighbourhood function on S, f_c, f_p, f_s, f_e be a choice–, a production–, a selection–, and an evaluation function, respectively. The Abstract Genetic Algorithm (AGA) is as follows:

Create an $x \in S^*$
WHILE NOT $f_e(x)$ DO
  BEGIN
    draw α,β and γ
    $q = f_c(\alpha,x)$
    $y = \bigcup_{z \in q} f_p(\beta,z)$
    $x' = x \cup y$
    $x = f_s(\gamma, x')$
  END
Output the actual population

The locality of the search emanates from the production function that is defined in such a way that all the children are from the neighbourhoods of the parents. In the meanwhile, taking the whole set S as the neighbourhood of its elements is a correct definition, which frees us from being restricted to local search.

The drawings that provide the random parameters are decisive. They influence the choice–, the production–, and the selection functions, hence different distributions of these drawings give different characters to an instance of the AGA.

The AGA generalizes GAs as well as SA algorithms as the following examples indicate.

Example 1

We instantiate AGA to a classical genetic algorithm with a finite binary space, crossover of two parents yielding two children plus mutation of a single element to create one child and selection by a pure survival–of–the–ten–fittest mechanism. The appropriate, although partial, instantiation of the AGA is then the following.

$S = \{0,1\}^k, \quad (k \in \mathbb{N})$

$N(s) = S$ for every $s \in S$,

$P = \{ [s] \mid s \in S \} \cup \{ [s,t] \mid s,t \in S, s \neq t \}$,

$$f_p(\beta,x) = \begin{cases} cross(\beta,s,t) & \text{if } x = [s,t] \\ mut(\beta,s) & \text{if } x = [s] \end{cases}.$$

$f_s(\gamma,y) = \{s_1 \in y, \ldots, s_{10} \in y \mid \forall\, 1 \leq i \leq 10\ \forall\, s \in y \setminus \{s_1, \ldots, s_{10}\} : f(s_i) \leq f(s)\}$.

For the sake of convenience we leave out the further details. □

Example 2

To obtain a simulated annealing algorithm as an instance of AGA let

S be arbitrary

C be the interval $(0,1] \subseteq \mathbb{R}$

$P = \{ [s] \mid s \in S \}$,

$f_c(\alpha,[s]) = \{[s]\}$ for every $\alpha \in A$

$f_p(\beta,[s]) = [t_\beta]$ for every $\beta \in B$ such that $t_\beta \in N(s)$ for a given neighbourhood function N,

$$f_s(\gamma,[s,t]) = \begin{cases} [s] & \text{if } exp\left[\frac{f(s) - f(t)}{c}\right] > \gamma \\ [t] & otherwise \end{cases},$$

where $0 < \gamma \leq 1$ by the definition of C, c is the control parameter. □

These examples also show that simulated annealing can be viewed as a special form of a genetic algorithm characterized by populations of size 1 and the specific elimination mechanism, the above selection function. Adopting the common terminology we can specify *mutation* as a special form of offspring production where the parent set is a singleton. This puts SA algorithms in a still other light: they are GAs where children are produced exclusively by mutation.

To provide a better vision on the nature of AGA we give an informal summary of the differences and the similarities between AGA and the classic genetic algorithms.

Similarities:

1) A final search space is traversed in search for a minimal object function value.
2) The search is an iterative generate–and–test procedure, in each cycle the AGA maintains a set (list) of individuals, a population.

3) New elements of the search space are generated from the old ones, parents are chosen, offspring are produced, the population is extended.
4) There is an elimination mechanism to abort unfit elements to increase the fitness of the population.

Differences:

1) The search space in the AGA is a set in general, the representation of the individuals is not restricted to string –or any other– coding.
2) The number of parents is not restricted to the usual number of two (one for mutation). Creation of children and mutation are unified by generalizing the notion of parent.
3) The usual crossover for making offspring is generalized to a production function.
4) By the unrestricted selection function we allow any arbitrary elimination mechanism.

Observe that by maintaining multiple candidates AGA is very suitable for parallel execution. Choosing the parent sets, computing the children of the parent sets, evaluating a population by evaluating its members all can be done sequentially as well as parallely. Regardless the manner of execution the AGA generates a sequence of populations which we shall call evolution. In the next section we give an exact definition of the evolution and investigate its convergence properties.

2 CONVERGENCE OF THE ABSTRACT GENETIC ALGORITHM

For an exact definition of 'evolution' we introduce the *transition function belonging to an AGA* as a function $f_t : (A \times B \times C) \times S^* \longrightarrow S^*$ to create the next population 'in one go':

$$f_t((\alpha,\beta,\gamma),x) = f_s(\gamma, x \cup \bigcup_{z \in f_c(\alpha,x)} f_p(\beta,z)).$$

Taking $\alpha_n \in A$, $\beta_n \in B$, $\gamma_n \in C$ $(n \in \mathbb{N})$ by independent random drawings we can define the *evolution belonging to an AGA* as $\{ x_n \mid n \geq 0 \}$, where

$x_0 \in S^*$ is the *initial population*,
$x_{n+1} = f_t((\alpha_n,\beta_n,\gamma_n),x_n)$ for $n \geq 0$.

Further on we study only the evolution, without the 'inside' details. Therefore we introduce abstract random variables $Y_n(\omega) \in S^* \longrightarrow S^*$ for the n–th transition function (creating the (n+1)–th population from the n–th one), where $\omega \in \Omega$, $(\Omega,\mathcal{A},\mathbb{P})$ is an arbitrary probability space. Our idea is that for each execution of the AGA an $\omega \in \Omega$ is chosen by a random mechanism. Then the evolution becomes $\{ X_n(\omega) \mid n \in \mathbb{N} \}$, where

$X_0(\omega) = x$, $x \in S^*$ is arbitrarily fixed, that is $\mathbb{P}[X_0(\omega) = x] = 1$,
$X_{n+1}(\omega) = Y_n(\omega)(X_n(\omega))$ for $n \geq 0$.

To provide an easier reading of the formulae we often leave out the symbol ω from the notation. In such cases $\mathbb{P}[property(X_n)]$ means $\mathbb{P}[\ \{\ \omega \in \Omega \mid property(X_n(\omega))\}\ ]$.

Notice that we obtain different evolutions for different initial populations. Therefore we use a notation that indicates the dependence on the initial population:

$\{X_n \mid n \in \mathbb{N}\}_x$ denotes the evolution with $\mathbb{P}[X_0 = x] = 1$ and
$\mathbb{P}_x[\ldots X_n \ldots]$ stands for $\mathbb{P}[\ldots X_n \ldots \omega\, X_0 = x]$.

Lemma

If Y_n's are independent then $\{X_n \mid n \in \mathbb{N}\}_x$ is a Markov chain for every $x \in S^*$ and if the Y_n's have the same distribution then the chain is homogeneous. □

To establish convergence we have to express formally that the algorithm tends to an optimum. Observe that according to our general approach the search space is simply a set without any norm or distance measure. Therefore we can not expect convergence criteria saying that X_n $(n \longrightarrow \infty)$ is 'getting close' to an optimum. Instead, we require that the chance of containing an optimum in X_n is growing to 1.

Let $S_0 = \{s \in S \mid s \text{ is a minimum of } f\}$.

An $s \in S$ is *accessible* by the evolution $\{X_n \mid n \in \mathbb{N}\}_x$ if $\mathbb{P}_x[\exists\, n \in \mathbb{N} : s \in X_n] > 0$.

The evolution is *monotone* if $\forall\, n \in \mathbb{N} : min\{f(s) \mid s \in X_{n+1}\} \le min\{f(s) \mid s \in X_n\}$.

The set of *possible successors* of $x \in S^*$ is $succ(x) = \{y \in S^* \mid \exists\, n \in \mathbb{N} : \mathbb{P}_x[X_n = y] > 0\}$.

The following theorem is the technical basis of our general convergence results.

Theorem 1

Let $x \in S^*$ and let the following hold

a) $\{X_n : n \in \mathbb{N}\}_x$ is monotone,

b) $n_k \in \mathbb{N}$ and $\varepsilon_k \in (0,1]$ $(k \in \mathbb{N})$ are such that $n_k \to \infty$ $(k \to \infty)$ and $\prod_{k=0}^{\infty} \varepsilon_k = 0$ and

$\forall\, y \in succ(x) : \mathbb{P}[X_{n_{k+1}} \cap S_0 = \emptyset \mid X_{n_k} = y] \le \varepsilon_k$

holds for every $k \in \mathbb{N}$.

Then $\{X_n \mid n \in \mathbb{N}\}_x$ almost surely reaches an optimum, that is $\mathbb{P}_x[\lim_{n\to\infty} X_n \cap S_0 \neq \emptyset] = 1$.

Proof

We omit the proof only remarking that its main idea is to have upper bounds on the probability of taking the wrong way, i.e. transitions in S that do not reach any optimum. For the details see Aarts, Eiben and van Hee (1989). □

Theorem 2

Let $x \in S^*$ be such that the following conditions are satisfied:

a) $\{X_n \mid n \in \mathbb{N}\}_x$ is monotone, and

b) $\{X_n \mid n \in \mathbb{N}\}_x$ is homogeneous, and

c) for every $y \in succ(x)$ there exists at least one accessible optimum.

Then $\{X_n \mid n \in \mathbb{N}\}_x$ almost surely reaches an optimum, i.e. $\mathbb{P}_x[\lim_{n\to\infty} X_n \cap S_O \neq \emptyset] = 1$.

Proof

We apply Theorem 1 and construct a sequence $n_0, n_1, \ldots$, and a sequence $\varepsilon_0, \varepsilon_1, \ldots$ so that they satisfy its condition (b). □

Next we come to our original purpose to find restrictions on the functions of the AGA such that together they imply convergence.

The selection function $f_s : C \times S^* \to S^*$ is *conservative* if it always keeps one of the strongest individuals of any population, that is if for every $x \in S^*$ and $\gamma \in C$

$$M_x \cap f_s(\gamma,x) \neq \emptyset$$

where

$$M_x = \{ s \in x \mid \forall t \in x : f(s) \leq f(t) \}$$

contains the minima of x. It is easy to see that if the selection function is conservative then any evolution created by AGA is monotone.

The neighbourhood structure is *connective* if stepping from neighbour to neighbour we can get from any given individual to any other individual, i.e. if

$$\forall s \in S \, \forall t \in S : s \mapsto t,$$

where $\mapsto$ stands for the transitive closure of the relation $\{ (s,t) \in S \times S \mid t \in N(s) \}$.

In the following three definitions we define the same predicate for the choice –, the production – and the selection function. We call them *generous* if they give a positive chance to everybody, to become a parent, to be born and to survive, respectively.

The *choice function is generous* if

$$\{ [s] \mid s \in S \} \subseteq P \quad \text{and} \quad \forall x \in S^* \, \forall s \in x : \mathbb{P}[[s] \in f_c(\alpha,x)] > 0.$$

The *production function is generous* if

$$\forall s \in S \, \forall t \in N(s) : \mathbb{P}[t \in f_p(\beta,[s])] > 0.$$

The *selection function is generous* if

$$\forall x \in S^* \, \forall s \in x : \mathbb{P}[s \in f_s(\gamma,x)] > 0.$$

Observe that the generousness of the choice and the production function implies that mutation is used.

Let Z_n denote the stochast that determines the parameters α_n, β_n and γ_n for the n–th transition function. Formally, let $(\Omega,\mathcal{A},\mathbb{P})$ be a probability space, and $Z_n \in \Omega \to A \times B \times C$ $(n \in \mathbb{N})$ be a sequence of independent random variables to be used in $f_t(Z_n(\omega),x)$. Denoting the projections of $Z_n(\omega)$ as $\alpha_n = Z_n(\omega).1$, $\beta_n = Z_n(\omega).2$ and $\gamma_n = Z_n(\omega).3$ we get back the former notation. We also make a little technical assumption, in the sequel we assume that A, B and C are countable sets, with positive probability for all their members,

$$\forall \alpha \in A : \mathbb{P}[\{\omega \mid Z(\omega).1 = \alpha\}] > 0, \text{ etc.}$$

Theorem 3

Let Z_n $(n \in \mathbb{N})$ have the same distribution, the choice –, the production–, and the selection function be generous, the selection function be conservative and the neighbourhood structure be connective.

Then for any initial population the AGA finds an optimum with probability 1, that is

$\mathbb{P}_x[\lim_{n\to\infty} X_n \cap S_O \neq \emptyset] = 1.$

Proof

The proof goes via Theorem 2, we show that its conditions (a), (b) and (c) hold for any $x \in S^*$. For the details cf. Aarts, Eiben & van Hee (1989). □

The requirements on the neighbourhood structure can be relaxed at the cost of a further restriction of the selection function. The *neighbourhood structure is quasi connected* if

$$\exists\, s \in S\ \forall\, t \in S : s \mapsto t.$$

In such a case let σ stand for an element with $\forall\, t \in S : \sigma \mapsto t$.

If the neighbourhood structure is quasi connected and $\sigma \in S$ is as given above then the selection function is called σ-*preserving* if

$$\forall\, x \in S^*\ \forall\, \gamma \in C : \sigma \in x \implies \sigma \in f_s(\gamma,x).$$

Theorem 4

Let us assume that the drawings Z_n's have the same distribution. Let the choice –, the production – and the selection function be generous and the selection function be conservative. Furthermore let the neighbourhood structure be quasi connected and the selection function be σ–preserving. Then for any initial population the AGA finds an optimum with probability 1, i.e. $\mathbb{P}_x[\lim_{n\to\infty} X_n \cap S_O \neq \emptyset] = 1.$

Proof

It is similar to that of Theorem 3. □

The latter theorems can be applied to any instance of AGA, thus to any genetic algorithm. Applying them to simulated annealing, however, requires carefulness. Observe that for the classic SA (Example 2) conservativity does not hold due to the special selection (acceptance) mechanism. To obtain monotonicity without changing this mechanism we extend the standard SA algorithm in such a way that the we maintain a second element s_b in the population, that is always 'a best seen so far'.

Example 3 Extended Simulated Annealing

S, C, P and f_p are the same as in Example 2.

$f_c(\alpha,[s,s_b]) = [s]$ for every $\alpha \in A$

$$f_s(\gamma,[s,s_b,t]) = \begin{cases} [t,s_b'] & \text{if } \exp\left[\dfrac{f(s) - f(t)}{c}\right] > \gamma \\ [s,s_b'] & \text{otherwise} \end{cases}$$

where

$$s_b' = \begin{cases} s_b & \text{if } f(t) \geq f(s_b) \\ t & \text{if } f(t) < f(s_b) \end{cases} \quad \text{and}$$

$0 < \gamma \leq 1$ by the definition of C, $c \in \mathbb{R}$ is the usual control parameter for SA. □

It is easy to see the the above reduction function is conservative, hence the evolution belonging to an Extended Simulated Annealing (ESA) algorithm is always monotone.

SUMMARY

In this paper we identified genetic algorithms as stochastic search methods to handle combinatorial optimization problems. We specified an Abstract Genetic Algorithm that possesses the main features of GAs. As Example 2 displayed also SA algorithms can be obtained as an instance of AGA, thus it generalizes GAs and SA as well. This AGA was then taken as the subject of convergence investigations, our the question was: which conditions should hold for the components of AGA to guarantee that it finds an optimum. We defined the evolution belonging to an AGA as a Markov chain and modified the original question to: which conditions imply that the evolution reaches an optimum. The answers are encapsulated in Theorems $1 - 4$, they can be applied to any instance of AGA, thus any genetic algorithm. As for simulated annealing we observed that in their classic form they do not satisfy the obtained conditions, hence we slightly modified them and presented an extended version.

LITERATURE

Aarts, E.H.L., Eiben, A.E., and van Hee, K.H. (1989), A General Theory of Genetic Algorithms, Computing Science Notes, Eindhoven University of Technology.

Aarts, E.H.L. and Korst, J. (1989), Simulated Annealing and Boltzmann Machines, J. Wiley and Sons.

DeJong, K.A. (1980), Adaptive System Design: A genetic Approach, IEEE Transactions on Sys. Man and Cybernetics, SMC–10,9 566–574.

Garey, R.M. and Johnson, D.S. (1979), Computers and Intractability: A Guide to the Theory of NP–Completeness, Freeman and Co.

Goldberg, D.E. (1989), Genetic Algorithms in Search, Optimization and Machine Learning, Addison–Wesley, Reading MA.

Goldberg, D.E. and Segrest, P. (1987), Finite Markov Chain Analysis of Genetic Algorithms, in Grefenstette, J.J. ed., Proceedings of the 2nd International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ.

Grefenstette, J.J., ed. (1985), Proceedings of the International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ.

Grefenstette, J.J. ed. (1987), Proceedings of the 2nd International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ.

Holland, J.H. (1975), Adaptation in Natural and Artificial Systems, Univ. of Michigan Press, Ann Arbor.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983), Optimization by simulated annealing, Science 220, 671–680.

Papadimitriou, C.H. and Steiglitz, K. (1982), Combinatorial Optimization: Algorithms and Complexity, Prentice –Hall, Englewood Cliffs, N.J.

Schaffer, J.D. (1989), ed., Proceedings of the 3rd International Conference on Genetic Algorithms, Morgan Kaufmann.