Stopping criteria for a general model of genetic algorithm

Marcin Studniarski[1]

[1] Faculty of Mathematics and Computer Science, University of Łódź, ul. S. Banacha 22, 90-338 Łódź, Poland, email: marstud@math.uni.lodz.pl

Abstract. We consider a general Markov chain model of genetic algorithm described in [3], Chapters 5 and 6. For this model, we establish an upper bound for the number of iterations which must be executed in order to find an optimal (or approximately optimal) solution with a prescribed probability. For the classical genetic algorithm with bitwise mutation, our result reduces to the main theorem of [1] in the case of one optimal solution, and gives some improvement over it in the case of many optimal solutions.

1 Introduction

Obtaining sensible stopping criteria is an important issue in the theory of genetic algorithms. One of the possible approaches to this problem is to obtain upper bounds for the number of iterations necessary to ensure finding an optimal solution with a prescribed probability (see [1] and references therein). In this paper we present some results of this type for a more general model of genetic algorithm, based on the theory developed in [3] and [5].

2 Random Heuristic Search

The RHS (*Random Heuristic Search*) algorithm, described in [5], is defined by an *initial population* $P^{(0)}$ and a *transition rule* τ which, for a given population $P^{(i)}$, determines a new population $P^{(i+1)}$. Iterating τ, we obtain a sequence of populations:

$$P^{(0)} \xrightarrow{\tau} P^{(1)} \xrightarrow{\tau} P^{(2)} \xrightarrow{\tau} ... \tag{1}$$

Each population consists of a finite number of *individuals* which are elements of a given finite set Ω called the *search space*. Populations are *multisets*, which means that the same individual may appear more than once in a given population.

To simplify the notation, it is convenient to identify Ω with a subset of integers: $\Omega = \{0, 1, ..., n-1\}$. The number n is called the *size of search space*. Then a population can be represented as an *incidence vector* (see [3, p. 141]):

$$v = (v_0, v_1, ..., v_{n-1})^T, \tag{2}$$

where v_i is the number of copies of individual $i \in \Omega$ in the population ($v_i = 0$ if the i-th individual does not appear in the population). The *size of population* v is the number

$$r = \sum_{i=0}^{n-1} v_i.$$

We assume that all the populations appearing in sequence (1) have the same size r. Dividing each component of incidence vector (2) by r, we obtain the *population vector*

$$p = (p_0, p_1, ..., p_{n-1})^T,$$

where $p_i = v_i/r$ is the proportion of individual $i \in \Omega$ in the population. In this way, we obtain a more general representation of the population which is independent of population size. It follows that each vector p of this type belongs to the set

$$\Lambda := \left\{ x \in \mathbb{R}^n : x_i \geq 0 \ (\forall i), \ \sum_{i=0}^{n-1} x_i = 1 \right\},$$

which is a simplex in $\mathbb{R}^n$. However, not all points of this simplex correspond to finite populations. For a fixed $r \in \mathbb{N}$, the following subset of Λ consists of all populations of size r (see [5, p. 7]):

$$\Lambda_r := \frac{1}{r} \left\{ x \in \mathbb{R}^n : x_i \in \mathbb{N} \cup \{0\} \ (\forall i), \ \sum_{i=0}^{n-1} x_i = r \right\}.$$

We now define the mapping

$$\mathcal{G} : \Lambda \longrightarrow \Lambda,$$

called *heuristic* [5, p.9] or *generational operator* [3, p. 144], in the following way: for a vector $p \in \Lambda$ representing the current population, $\mathcal{G}(p)$ is the probability distribution that is sampled independently r times (with replacement) to produce the next population after p. For each of these r choices, the probability of selecting an individual $i \in \Omega$ is equal to $\mathcal{G}(p)_i$, the i-th component of $\mathcal{G}(p)$.

A transition rule τ is called *admissible* if it is a composition of a heuristic $\mathcal{G}$ with drawing a sample in the way described above. Symbolically,

$$\tau(p) = \text{sample}(\mathcal{G}(p)), \quad \forall p \in \Lambda. \tag{3}$$

Of course, a transition rule defined this way is nondeterministic, i.e., by applying it repeatedly to the same vector p, we can obtain different results. It should also be noted that, although $\mathcal{G}(p)$ may not belong to Λ_r, the result of drawing an r-element sample is always a population of size r; therefore, it follows from (3) that $\tau(p) \in \Lambda_r$.

Theorem 2.1. [5, Thm. 3.4] *Let $p \in \Lambda_r$ be the current population vector. The probability that $q \in \Lambda_r$ is the next population vector is equal to*

$$r! \prod_{j=0}^{n-1} \frac{(\mathcal{G}(p)_j)^{rq_j}}{(rq_j)!}.$$

3 The RHS algorithm as a Markov chain

A sequence of random variables $\{X_t\}_{t\in\mathbb{N}_0}$ (where $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$) defined on the same probabilistic space $(Z, \mathcal{F}, \Pr)$, with values in a countable set S (the state space) is called a *Markov chain* if, for every $t \in \mathbb{N}$ and every sequence $s_0, s_1, ..., s_t \in S$, the following condition is satisfied:

$$\Pr\left(X_t = s_t | \, X_{t-1} = s_{t-1}, ..., X_1 = s_1, X_0 = s_0\right) = \Pr\left(X_t = s_t | \, X_{t-1} = s_{t-1}\right),$$

provided $\Pr(X_{t-1} = s_{t-1}, ..., X_1 = s_1, X_0 = s_0) > 0$.

A matrix is called *stochastic* if all its elements are nonnegative and the sum of every row is equal to 1. A stochastic matrix $\Pi(t) = [\pi_{i,j}(t)]_{i,j\in S}$ is called the *transition matrix of the Markov chain* $\{X_t\}_{t\in\mathbb{N}_0}$ at time t, $t \geq 1$, if $\pi_{i,j}(t) = \Pr\left(X_t = s_j | \, X_{t-1} = s_i\right)$ for all $j \in S$ and i such that $\Pr(X_{t-1} = s_i) > 0$. A Markov chain is called *(temporally) homogeneous* if there exists a matrix $\Pi = [\pi_{i,j}]_{i,j\in S}$ being the transition matrix of this Markov chain at every time t.

Let us now return to the RHS algorithm described in Section 2. It generates a sequence of populations

$$\hat{p},\ \tau(\hat{p}),\ \tau^2(\hat{p}), \dots\ , \tag{4}$$

where $\hat{p}$ is a fixed initial population. The RHS can be regarded as a Markov chain where the state space is Λ_r and the values of successive random vectors X_0, X_1, X_2,... are populations (4). Since $\hat{p}$ is fixed, we may assume that X_0 is a random vector taking on the single value $\hat{p}$ with probability 1.

We denote by $\Pr\left(q\,|\,p\right) = \Pr(\tau(p) = q)$ the probability of obtaining a population q in the current iteration of the RHS algorithm provided the previous population is p. It follows from Theorem 2.1 that

$$\Pr(q\,|\,p) = r! \prod_{j=0}^{n-1} \frac{(\mathcal{G}(p)_j)^{rq_j}}{(rq_j)!}. \tag{5}$$

Since this probability does not depend on t, we deduce that the RHS algorithm is a homogeneous Markov chain with the constant transition matrix $\Pi = [\pi_{p,q}]_{p,q\in S}$, where $S = \Lambda_r$, and the elements

$$\pi_{p,q} = \Pr(q\,|\,p) \tag{6}$$

are given by (5).

4 The transition matrix of a genetic algorithm

In this section we consider a genetic algorithm as a particular case of the RHS. We assume that a single iteration of the genetic algorithm produces the next population form the current population according to the following procedure:

1. Choose two parents from the current population by using a selection method which can be described by some heuristic (e.g. proportional, ranking or tournament selection; see [5, § 4.2]).
2. Crossover the two parents to obtain a child.
3. Mutate the child.

4. Put the mutated child into the next population.

5. If the next population contains less than r members, return to step 1.

The process of selection usually uses a *fitness function* $f : \Omega \longrightarrow \mathbb{R}^+$. For example, in proportional selection, the probability of choosing an individual $i \in \Omega$ is proportional to its fitness $f(i)$. The fitness function also defines optimality in the problem we have to solve (see Section 5).

The only difference between the iteration described above and the iteration of the Simple Genetic Algorithm defined in [5, p. 44] is that in our version mutation is done after crossover. This change in order facilitates the probability estimates needed for the proof of Theorem 5.2 below.

To derive our stopping criteria, we will use some properties of mutation which is generally understood as changing one element of the search space to another, with a certain probability. The way of implementing selection and crossover is not important for our model, so we omit the discussion of them (we refer the reader to [3, Chapter 5]). The only requirement is that the composition of the three operations (selection, crossover, mutation) can be described in terms of some heuristic that does not vary in time (i.e., the definitions of the corresponding genetic operators are independent of the generation index t). This implies that algorithms with self-adaptation are excluded from our considerations.

We assume that mutation consists in replacing a given individual from Ω by another individual, with a prescribed probability. Let us denote by $u_{i,j}$ the probability that individual i mutates to j. In this way, we obtain a $n \times n$ matrix $U = [u_{i,j}]_{i,j\in\Omega}$. The probability of generating individual $j \in \Omega$ from population p by successive application of selection, crossover and mutation is equal to (compare with the first equation on p. 120 in [3])

$$\mathcal{G}(p)_j = \Pr([j] \mid p)_{scm} = \sum_{i=0}^{n-1} u_{ij} \Pr([i] \mid p)_{sc}, \tag{7}$$

where the symbol $[i]$ means that we generate a single individual i (not a whole population as in (5)), the subscript *sc* means that we are dealing with the composition of selection and crossover, and the subscript *scm* indicates the composition of selection, crossover and mutation. To get a whole new population, one should draw an r-element sample from probability distribution (7). The probability of generating a population q in this way is equal, by (5) and (7), to

$$\Pr(q \mid p)_{scm} = r! \prod_{j=0}^{n-1} \frac{(\Pr([j] \mid p)_{scm})^{rq_j}}{(rq_j)!}. \tag{8}$$

According to (6), equation (8) gives also a formula for the transition matrix of our algorithm.

5 Stopping criteria for a genetic algorithm

We now consider the problem of maximizing the fitness function f over Ω. For any $\varepsilon \geq 0$, we define the set of *ε-optimal solutions* as follows:

$$\Omega^{+} := \left\{ j \in \Omega : f(j) \geq \max_{i \in \Omega} f(i) - \varepsilon \right\}.$$

In particular, for $\varepsilon = 0$, Ω^{+} is the set of (global) *optimal solutions*. Let $\Omega^{-} := \Omega \backslash \Omega^{+}$.

We assume that the goal of the RHS algorithm is to find some element of Ω^{+}. We say that element i *has been found in iteration* t if population $\tau^{t}(\hat{p})$ contains at least one copy of individual i, which is equivalent to $\tau^{t}(\hat{p})_{i} > 0$.

Let S^{-} denote the subset of the state space $S = \Lambda_{r}$ consisting of all populations which do not contain an element of Ω^{+}:

$$S^{-} := \left\{ p \in S : p_{i} = 0, \ \forall i \in \Omega^{+} \right\},$$

and let S^{+} be the set of all remaining populations: $S^{+} := S \backslash S^{-}$. The populations in S^{+} contain at least one copy of some individual from Ω^{+}. Thus, we can say that *an element of Ω^{+} has been found in iteration* t if the population generated in iteration t belongs to S^{+}. We will denote by A_{t} the event that no element of Ω^{+} has been found in iteration t:

$$A_{t} := \left\{ \tau^{t}(\hat{p}) \in S^{-} \right\}.$$

The following lemma gives an upper bound of the probability that no element of Ω^{+} has been found in the first t iterations. Due to space limitations, we do not give the proof here; it will appear elsewhere.

Lemma 5.1. *Suppose that, for some $\alpha \in (0,1)$, we have that*

$$\sum_{q \in S^{-}} \pi_{p,q} \leq \alpha, \quad \forall p \in S. \tag{9}$$

Then

$$\Pr\left(A_{1} \cap A_{2} \cap \ldots \cap A_{t}\right) \leq \alpha^{t} \tag{10}$$

for all $t \in \mathbb{N}$.

Using Lemma 5.1, we can easily obtain a lower bound for the probability that an element of Ω^{+} has been found in the first t iterations. Indeed, let B_{t} denote the event that an element of Ω^{+} has been found in iteration t:

$$B_{t} := Z \backslash A_{t} := \left\{ \tau^{t}(\hat{p}) \in S^{+} \right\}. \tag{11}$$

Then, assuming (9), we get from (11) and (10)

$$\begin{aligned} \Pr\left(B_{1} \cup \ldots \cup B_{t}\right) &= \Pr((Z \backslash A_{1}) \cup \ldots \cup (Z \backslash A_{t})) \\ &= \Pr(Z \backslash (A_{1} \cap \ldots \cap A_{t})) \\ &= 1 - \Pr(A_{1} \cap \ldots \cap A_{t}) \geq 1 - \alpha^{t}. \end{aligned} \tag{12}$$

The following theorem gives a more precise lower bound of the form (12) for the genetic algorithm model considered in the earlier sections.

Theorem 5.2. *We consider the general model of genetic algorithm, described in Section 4, being a special case of the RHS algorithm with the heuristic $\mathcal{G}$ given by (7). Suppose that the set Ω^+ of ε-optimal solutions has the form*

$$\Omega^+ = \{j_1, j_2, ..., j_m\}, \tag{13}$$

where the (possibly unknown) number m of these solutions is bounded from below by some known positive integer $\bar{m}$. Suppose also that there exists a number $\beta \in (0, 1/n]$ satisfying

$$u_{i,j} \geq \beta, \quad \forall i, j \in \Omega. \tag{14}$$

Then the probability of finding an element of Ω^+ in the first t iterations is at least

$$1 - (1 - \bar{m}\beta)^{rt}.$$

Proof. We will show that the assumption of Lemma 5.1 is satisfied with $\alpha = (1 - \bar{m}\beta)^r$. Let $\Pr(S^- \mid p)_{scm}$ denote the probability of generating a population in S^- from population p by first applying heuristic $\mathcal{G}$ given and then drawing an r-element sample from probability distribution $\mathcal{G}(p)$. Further, let $\Pr([\Omega^-] \mid p)_{scm}$ denote the probability of generating an individual in Ω^- from population p by single application of the operations of selection, crossover and mutation (which is equivalent to drawing a one-element sample from $\mathcal{G}(p)$). Then the left-hand side of (9) can be rewritten as follows:

$$\begin{aligned}\sum_{q \in S^-} \pi_{p,q} &= \sum_{q \in S^-} \Pr(q \mid p)_{scm} = \Pr(S^- \mid p)_{scm} \\ &= \big(\Pr([\Omega^-] \mid p)_{scm}\big)^r = \big(1 - \Pr([\Omega^+] \mid p)_{scm}\big)^r, \end{aligned} \tag{15}$$

where the third equality in (15) follows from the independence of r random variables constituting an r-element sample.

Now, using (7) and (14), we deduce that, for any $p \in S$ and $j \in \Omega$,

$$\Pr([j] \mid p)_{scm} \geq \beta \sum_{i=0}^{n-1} \Pr([i] \mid p)_{sc} = \beta, \tag{16}$$

where the final equality follows because we sum up probabilities of disjoint events whose union is the entire sample space Z. Taking into account the representation of Ω^+ given by (13), and using inequality (16), we get

$$\Pr([\Omega^+] \mid p)_{scm} = \sum_{l=1}^{m} \Pr([j_l] \mid p)_{scm} \geq \sum_{l=1}^{m} \beta = m\beta \geq \bar{m}\beta. \tag{17}$$

Conditions (15) and (17) imply

$$\sum_{q \in S^-} \pi_{p,q} \leq (1 - \bar{m}\beta)^r.$$

We may assume without loss of generality that $\Omega^+ \neq \Omega$; then $\bar{m} \leq m < n$, and $(1 - \bar{m}\beta)^r \in (0, 1)$. Hence, we may apply Lemma 5.1 with $\alpha = (1 - \bar{m}\beta)^r$ to obtain

$$\Pr(A_1 \cap ... \cap A_t) \leq (1 - \bar{m}\beta)^{rt},$$

for all $t \in \mathbb{N}$. By using (12), we can estimate the probability of finding an element of Ω^+ in the first t iterations as follows:

$$\Pr\left(B_1 \cup ... \cup B_t\right) = 1 - \Pr(A_1 \cap ... \cap A_t) \geq 1 - (1 - \bar{m}\beta)^{rt},$$

which concludes the proof of the theorem. □

Corollary 5.3. *For any $\delta \in (0,1)$, we denote by $t_{\min}(\delta)$ the smallest number of iterations required to guarantee that an element of Ω^+ has been found with probability δ. Then*

$$t_{\min}(\delta) \leq \left\lceil \frac{\ln(1-\delta)}{r\ln(1-\bar{m}\beta)} \right\rceil, \tag{18}$$

where $\lceil x \rceil$ is the smallest integer greater than or equal to x.

Proof. By choosing the number of iterations t satisfying the inequality

$$1 - (1 - \bar{m}\beta)^{rt} \geq \delta, \tag{19}$$

we have guaranteed that an element of Ω^+ has been found with probability at least δ. Inequality (19) is equivalent to the following one:

$$t \geq \frac{\ln(1-\delta)}{r\ln(1-\bar{m}\beta)}. \tag{20}$$

For each positive integer t satisfying (19) (or equivalently, (20)), we have that $t_{\min}(\delta) \leq t$. Hence, by taking t equal to the right-hand side of (18), we get the desired inequality for $t_{\min}(\delta)$. □

Corollary 5.3 suggests the following criterion of termination of a genetic algorithm. First, we choose the probability δ (guarantee level) with which we want to find an ε-optimal solution of our optimization problem. Next, we compute the number T of required iterations, where T is the right-hand side of formula (18). We run T iterations of the algorithm, and at each iteration we store in memory and update the best individual (in the sense of the value of f) found so far. Then the best individual found in T iterations is an ε-optimal solution with probability δ.

6 The case of strings over an arbitrary alphabet

We now consider the model of genetic algorithm described in [2], where individuals are represented as strings of symbols drawn from an arbitrary alphabet of cardinality c. Without loss of generality we may identify this alphabet with the set of *integers modulo* c:

$$\mathbb{Z}_c := \{0, 1, ..., c-1\},$$

where $c \geq 2$. For any $a, b \in \mathbb{Z}_c$, we define the operator of addition modulo c as follows:

$$a \oplus b := (a + b) \bmod c.$$

Each element $a \in \mathbb{Z}_c$ has exactly one inverse element with respect to the operation $\oplus$, denoted by $-a$. For simplicity, we use the notation

$$a \ominus b := a \oplus (-b).$$

We assume that the search space Ω of a genetic algorithm is the set of all strings of length ℓ composed of elements of $\mathbb{Z}_c$ (a string is a finite sequence of elements which are called *digits* here). More precisely, we have

$$\Omega = \underbrace{\mathbb{Z}_c \times \ldots \times \mathbb{Z}_c}_{\ell \text{ times}}.$$

We can easily extend the operations $\oplus$ and $\ominus$ onto Ω by performing them componentwise.

Further, we assume that the mutation operation in the considered genetic algorithm is defined by a mutation rate (see [2, Remark 1]). Such a mutation acts as follows: first a *mutation mask* $m \in \Omega$ is randomly selected, and then an individual $y \in \Omega$ (also randomly selected) is replaced by $y \oplus m$. A number $\mu \in [0, 1/2)$ is called a *mutation rate* if it specifies a probability distribution $\hat{\mu} \in \Lambda$ by the formula

$$\hat{\mu}_m := \left(\frac{\mu}{c-1}\right)^{n(m)} (1-\mu)^{\ell - n(m)}, \quad \forall m \in \Omega,$$

where

- $\hat{\mu}_m$ is the probability of selecting m as a mutation mask;
- $n(m)$ is the number of nonzero digits in m; it is also the number of digits in y that will be mutated (i.e., changed by adding modulo c the respective nonzero digit of m);
- $\ell - n(m)$ is the number of digits in y that do not get mutated.

It should be remembered that, in the process of mutation described above, two probability distributions are used: $\hat{\mu}$ for selecting the mutation mask, and another distribution $\hat{\nu} \in \Lambda$ for selecting the individual y to be mutated. Since mutation follows after selection and crossover, we have $\hat{\nu}_k := \Pr([k] \,|\, p)_{sc}$ (assuming that the previous population is p).

The probability that $i \in \Omega$ mutates to j is equal to (cf [4, str. 474])

$$u_{i,j} = \hat{\mu}_{j \ominus i} = \left(\frac{\mu}{c-1}\right)^{n(j \ominus i)} (1-\mu)^{\ell - n(j \ominus i)}. \tag{21}$$

Corollary 6.1. *Under the assumptions of Theorem 5.2 and Corollary 5.3, if the elements of the matrix U are given by (21), then*

$$t_{\min}(\delta) \leq \left\lceil \frac{\ln(1-\delta)}{r \ln\left(1 - \bar{m}\left(\frac{\mu}{c-1}\right)^{\ell}\right)} \right\rceil. \tag{22}$$

Proof. Since $\mu \in [0, 1/2)$, we have $\mu \leq \frac{c-1}{c}$, which, after simple calculations, gives

$$1 - \mu \geq \frac{\mu}{c-1}. \tag{23}$$

Conditions (21) and (23) imply

$$u_{i,j} \geq \left(\frac{\mu}{c-1}\right)^{\ell}.$$

By defining $\beta := \min\left\{\frac{1}{n}, \left(\frac{\mu}{c-1}\right)^{\ell}\right\}$, we ensure that the assumptions of Theorem 5.2 are satisfied. Inequality (22) then follows from (18). $\square$

Comparing Corollary 6.1 with Theorem 1 of [1], we see that, for $\bar{m} = 1$, these two results give the same estimate under the assumption (23). However, for $\bar{m} > 1$, the result obtained here is strictly better.

7 Final remarks

The stopping criteria presented in this paper use only some properties of mutation. The properties of selection and crossover have no influence on them: because the probabilities $\Pr([i] \,|\, p)_{sc}$ in (16) sum up to 1, their values are not essential for our estimate. This observation suggests that it should be possible to strengthen the results by making some use of the fact that, in the process of selection, *better* individuals are chosen with *greater* probabilities. Another possible way of improvement is to incorporate the case where crossover is done after mutation. This could possibly be accomplished by using the result of Exercise 5 on p. 35 in [5] which gives a sufficient condition for the mutation scheme and the crossover scheme to commute. Further research will be devoted to these two questions as well as to obtaining similar results for multiobjective optimization problems.

Acknowledgment. The author wishes to thank Prof. Robert Schaefer for his helpful comments.

Bibliography

[1] D. Greenhalgh, S. Marshall, *Convergence criteria for genetic algorithms*, SIAM Journal on Computing **30** (2000), 269-282.

[2] G.J. Koehler, S. Bhattacharya, M.D. Vose, *General cardinality genetic algorithms*, Evolutionary Computation **5** (1998), 439-459.

[3] C.R. Reeves, J.E. Rowe, *Genetic Algorithms — Principles and Perspectives: A Guide to GA Theory*, Kluwer, Boston, 2003.

[4] J.E. Rowe, M.D. Vose, A.H. Wright, *Structural search spaces and genetic operators*, Evolutionary Computation **12** (2004), 461-493.

[5] M.D. Vose, *The Simple Genetic Algorithm: Foundations and Theory*, MIT Press, Cambridge, Massachusetts, 1999.