A Restarting Rule Based on the Schnabel Census for Genetic Algorithms [*]

Anton V. Eremeev[†]

January 13, 2019

Sobolev Institute of Mathematics SB RAS, Omsk, Russia,
Institute of Scientific Information on Social Sciences RAS, Moscow, Russia.
Email: eremeev@ofim.oscsbras.ru

Abstract

A new restart rule is proposed for Genetic Algorithms (GAs) with multiple restarts. The rule is based on the Schnabel Census method, transfered from the biometrics, where it was originally developed for the statistical estimation of a size of animal population. In this paper, the Schnabel Census method is applied to estimate the number of different solutions that may be visited with positive probability, given the current distribution of offspring. The rule consists in restarting the GA as soon as the maximum likelihood estimate reaches the number of different solutions observed at the recent iterations.

We demonstrate how the new restart rule can be incorporated into a GA on the example of the Set Cover Problem. Computational experiments on benchmarks from OR-Library show a significant advantage of the GA with the new restarting rule over the original GA. On the unicost instances, the new rule also tends to be superior to the well-known rule, which restarts an algorithm when the current iteration number is twice the iteration number when the best incumbent was found.

Keywords: maximum likelihood, abundance of population, set cover, transfer of methods

1 Introduction

Genetic Algorithms (GAs) are the randomized search heuristics based on the biological analogy of selective breeding in nature, originating from the work of J. Holland [18]. A GA manipulates with a *population* of *individuals*, using the random operators that model mutation and crossover in nature. Suppose that a GA is applied to an optimization problem with the space of solutions D and the objective function $f : D \to \mathbb{R}$ to be maximized. An individual is a pair of *genotype* g and *phenotype* $x(g)$, where g is a fixed

[*]The final authenticated publication is available online at https://doi.org/10.1007/978-3-030-05348-2_29

[†]This research is supported by the Russian Science Foundation grant 17-18-01536.

length string of symbols (called *genes*) from some finite alphabet, and $x(g)$ corresponds to a search point in the space of solutions D. The function $x(g)$ maps g to its phenotype $x(g) \in D$, thus defining a *representation* of solutions in a GA. The search in GAs is guided by the values of the *fitness* function $\Phi(g) = \phi(f(x(g)))$ on the genotypes of the current population Π^t on iteration t. Here $\phi : \mathbb{R} \to \mathbb{R}$ may be an identical mapping or a monotone function, chosen appropriately to intensify the search. The genotypes of the initial population Π^0 are generated according to some a priori defined probability distribution.

Due to randomness of initialization, selection, mutation and crossover operators, their behavior varies from run to run. In order to increase the probability of finding an optimal solution, it is a common practice to use multiple restarts of a GA. The choice of the iteration when the GA is stopped and restarted again (a restart rule) was considered in a number of papers [2, 19, 20, 22]. A stopping criterion is also proposed for the multi-objective evolutionary algorithms in [23].

In this paper, a new restart rule is proposed for the GAs. The rule is based on the Schnabel Census method, transfered from the biometrics, where it was originally developed for the statistical estimation of a size of animal population [34], assuming that one takes repeated samples of size 1 (at suitable intervals of time) and counts the number of distinct animals seen. This method was used in computer science already to estimate the number of local optima on the basis of repeated local search [31]. Experiments [31, 32] showed that the estimates based on this approach are adequate for the landscapes with uniform basin of attraction sizes, but have a negative bias when the basin sizes are significantly unequal.

Here we make a simplifying assumption that during the latest iterations, the GA population was generated according to the same distribution and the Schnabel Census method is applicable to estimate the number of different solutions that may be visited with positive probability in this distribution. The rule consists in restarting the GA as soon as a maximum likelihood estimate reaches the number of different solutions observed at the latest iterations. The rationale of this rule is that it stops the GA when, most likely, there are no more non-visited solutions in the area where the GA population spent the latest iterations. In such a case it would be more appropriate to restart the GA instead of waiting till it leaves the explored area by mutation and crossover.

There are two major GA outlines in use: the *elitist* GAs, which copy a certain number of the "most promising" individuals from the previous population to the next one, and the *non-elitist* GAs, which generate all individuals of a new population with the same probability distribution. The well-known Simple GA [18, 35] is an example of non-elitist GAs. We expect that the new restart rule is sufficiently general to be applicable in both types of GAs, although in the present paper the rule is tested only with an elitist steady-state GA.

We demonstrate how the new restart rule can be incorporated into a GA [9] with Non-Binary Representation (NBGA) for the Set Cover Problem (SCP). Computational experiments on benchmarks from OR-Library show a significant advantage of the GA with the new restarting rule, compared to the original version of the GA [9]. In particular, given equal computational budget, in 35 out of 74 SCP instances the new version of the GA had a higher frequency of finding the optima and only in 5 out of 74 instances the new version showed inferior results. On the unicost SCP instances, the new rule also tends to be superior to the well-known rule [16], which restarts an algorithm when the current iteration number is twice the iteration number when the best incumbent was found.

The idea of using the Schnabel Census method for estimation of the number of unvisited solutions [31], as well as the basic ideas of GA [18], are the examples of transfer of ideas from biology into computer science. In the present paper, both methods are combined together. Schnabel Census, originally developed for estimation of animal population size, is not used for counting individuals here (the population size is a known parameter of a GA, kept constant throughout the run), but for estimation of the number of solutions which may be visited if the distribution of offspring in the GA remains unchanged.

In the next section, we briefly discuss the use of Schnabel Census in biology and computer science. Section 3 gives a motivation and a detailed description of the restart rule based on Schnabel Census. Section 4 briefly describes the GA considered in this paper. Section 5 presents the experimental evaluation of the proposed restart rule. Concluding remarks a given in Section 6.

2 Estimation of Animal Population Size and the Number of Local Optima

Schnabel Census method is developed in biometrics for statistical estimation of the size of animal populations [33, 34]. According to this method, one takes repeated samples of size n_0 from a population and counts the number of *distinct* animals seen. Often it is assumed that the probability of catching any particular animal is the same. The sampled animals are marked, unless they were marked previously, and returned back into the population. Then a statistical estimate for the total number ν of individuals in population is computed on the basis of the total number of animals marked in all the samples.

This method, with the sample size $n_0 = 1$, was adapted in [31] to estimate the number of local optima in combinatorial optimization problems on the basis of repeated local search outcomes with random starting points. A number of other approaches have been proposed for estimation of the number of local optima. One may fit certain type of parametric distribution of the basin of attraction sizes (exponential, gamma, lognormal etc.) [14, 32] to estimate this parameter. Nonparametric estimates, such as the bootstrap or the jackknife, can also be employed [11, 27, 34, 32]. Assuming a particular type of distribution of basin sizes one can obtain the maximal likelihood estimate for the local optima number, or a confidence interval for it.

In what follows, we will apply the Schnabel Census method to estimate the number of values that a discrete random variable may take with non-zero probability. The other methods mentioned above could be applied to this problem as well, but unfortunately this problem does not have a satisfactory solution in the general case (see e.g. [22]), where different values of the random variable may have diferent probabilities.

3 Restart Rule Based on Schnabel Census

One of the theoretical approaches to understanding the Simple GA (and some of its its generalisations) as a dynamical system was suggested in [35, 37]. Suppose that X is a finite genotypes space. In the dynamical system models of GAs, a *population vector* $\mathbf{p}$

is introduced which has a length $|X|$. A k-th component p_k of this vector is the proportion of the population Π that has a genotype $y_k \in X, k = 0, ..., |X| - 1$ (assuming that the genotypes in X are numbered in some standard order). Given a population vector $\mathbf{p}^t$, of the current population Π^t, a function $G(\mathbf{p}^t)$ produces a vector in $[0,1]^{|X|}$, where the k-th component, $k = 1, \dots, |X|$ equals the probability that an offspring computed for population Π^{t+1} will have the genotype y_k. The result of M.Vose [35] shows that as the population size tends to ∞, the sequence of population vectors of the Simple GA $\mathbf{p}^0, \mathbf{p}^1, ..., \mathbf{p}^t$ converges in probability to the sequence $\mathbf{p}^0, G(\mathbf{p}^0), ..., G^t(\mathbf{p}^0)$ for any finite t. This suggests that it may be helpful to consider the infinite-population GA as an approximation of a finite-population GA because on each iteration of an infinite-population GA the vector $\mathbf{p}^{t+1}$ is a deterministic function $G(\mathbf{p}^t)$ of the current population vector. It was shown in [36, 37] that the fixed points of G (i.e. such population vectors $\mathbf{p}$ where $\mathbf{p} = G(\mathbf{p})$ holds) are crucial in the analysis of the infinite-population GA. The properties of a fixed point $\mathbf{p}$ of an infinite-population GA are determined by the eigenvalues associated with the differential of G at that point (see e.g. [36, 37]). If all these eigenvalues belong to the interior of the unit disk, then the point $\mathbf{p}$ is called *stable*. It was shown in [35] that these stable points are *attractors* in the sense that from almost all initial population vectors $\mathbf{p}^0$ the infinite-population Simple GA converges to one of the stable points. The finite-population GAs demonstrate a similar *metastable* behavior, although in a randomized way: the probability distribution of their population remains "almost" stationary (close to a stable population vector $\mathbf{p}$) for a great number of iterations, until a seldom random event might shift it out into the basin of attraction of some other stable fixed point [24]. The elitist GAs were considered in [37] and shown to have a metastable behavior as well.

In [28], C.Reeves has shown that it is rather unlikely that a Hamming local optimum (w.r.t. Hamming neighborhood of radius 1 in the genotypes space) would not be a GA attractor as well. Experimental [28] and theoretical [36] studies have also revealed that there is a strong connection between attractors and local optima. With this in mind, it should be advisable to restart a GA, once its population has been trapped for a long time in one of the stable points. This idea is implemented in the new restart rule described below.

Let a parameter r define the length of the historical period considered for statistical analysis in the restart rule. Given a value of r, we assume that during the r latest iterations, all new offspring in the GA obeyed the same distribution and their genotypes may be treated as the sampled animals in the Schnabel Census method. Then we apply the Schnabel Census method in order to estimate the number ν of different solutions that may be visited with a positive probability, assuming that the current distribution of offspring remains unchanged.

In what follows, we assume that in the latest r iterations of a GA, the observed sample consists of r independent offspring solutions. Let us define the random variable K as the number of *distinct* solutions in this sample. We will make a simplifying assumption that all solutions, that may be generated in the current distribution, have equal probabilities. Then, as it was noticed in [7], for any fixed ν the value K has the following distribution:

$$\Pr\{K = k\} = \frac{\nu!}{(\nu - k)!} \frac{S(r,k)}{\nu^r},$$

where $S(r,k) = \frac{1}{k!} \sum_{s=0}^{k} (-1)^k \binom{k}{s} (k-s)^r$ is the Stirling number of the second kind. This distribution is also known as the Arfwedson distribution [21]. The maximum likelihood estimate $\hat{\nu}^{\mathrm{ML}}$ for the unknown ν is

$$\hat{\nu}(r,k) = \operatorname{argmax} \left\{ \frac{\nu!}{(\nu-k)!\nu^r} \right\}, \tag{1}$$

where k is the number of different solutions actually generated on the latest r iterations of the GA. The value $\hat{\nu}^{\mathrm{ML}} = \hat{\nu}(r,k)$ may be found from (1) by the standard one-dimensional optimization methods (see e.g. [31]).

The proposed rule restarts the GA as soon as the estimate $\hat{\nu}^{\mathrm{ML}}$ becomes equal to k. The value of r is tuned adaptively during the GA execution. The rationale behind this rule is that once the equality $\hat{\nu}^{\mathrm{ML}} = k$ is satisfied, most likely there are no more non-visited solutions in the area where the GA population spent the latest r iterations. In such a situation, it is more appropriate to restart the GA rather than to wait till the population distribution will significantly change by the evolutionary mechanisms.

4 The Genetic Algorithm for Set Cover Problem

The Set Cover Problem may be formulated as follows. *Consider a set* $\mathcal{M} = \{1,\dots,m\}$ *and the subsets* $\mathcal{M}_j \subseteq \mathcal{M}$, *where* $j \in \mathcal{N} = \{1,...,n\}$. *A subset* $\mathcal{J} \subseteq \mathcal{N}$ *is a cover of* $\mathcal{M}$ *if* $\bigcup_{j\in\mathcal{J}} \mathcal{M}_j = \mathcal{M}$. *For each* $\mathcal{M}_j$, *a positive cost* c_j *is assigned. The SCP is to find a cover of minimum summary cost.*

The SCP is a well-known NP-hard problem [13]. A number of heuristic algorithms are developed for approximate solving the SCP within relatively short running time: Lagrangian relaxation heuristics [6], neural networks [15], local search [38], GAs [4, 9], ant colony algorithms [1] etc.

Here we will use the GA which was proposed in our earlier work without any restart rule [9]. This GA is based on the elitist steady-state population management. It is denoted as NBGA because of the non-binary representation of solutions [4, 9], involving an alphabet with up to n symbols. The NBGA uses a problem-specific crossover operator based on the linear programming, the proportional selection operator and a mutation operator that makes random changes in every gene with a given probability p_{m}. The offsprings are improved by the means of different greedy procedures before they are added into the population.

5 Computational Experiments

The NBGA was implemented in Borland Delphi 5 and tested on Pentium-IV with 3 GHz CPU and 2 GB RAM, using the OR-Library [3] benchmark problem sets *4-6*, *A-H*, and two sets of combinatorial problems *CLR* and *Stein*. The sets *4-6* and *A-H* consist of randomly generated problems with costs c_j from 1,...,100, while *CLR* and *Stein* consist of combinatorial unicost problems, where $c_j = 1$ for all j. Dimensions of the problems and the number of instances in each randomly generated series are given in Table 1.

We compared three modes of GA execution with equal computational budget:

- Mode A. Single run with no restarts.

Table 1: Parameters of randomly generated instances

Series Name	*4*	*5*	*6*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*
Rows (m)	200	200	200	300	300	400	400	500	500	1000	1000
Columns (n)	1000	2000	1000	3000	3000	4000	4000	5000	5000	10000	10000
Num. of probl.	10	10	5	5	5	5	5	5	5	5	5

- Mode B. Restart the GA as soon as the current iteration number becomes twice the iteration number when the best incumbent was found. This rule was used successfully by different authors to restart random hill-climbing method [16] and GAs [2, 10]. To avoid early restarts, this rule is applied only after a certain number of iterations, denoted by $t_{\min}$. We use $t_{\min}$ equal to the population size.
- Mode C. Restart the GA using the new rule proposed in Section 3. The value of parameter r is chosen adaptively as follows: *Whenever the best found solution is improved, r is set to be the population size. If the best incumbent was not improved during the latest $2r$ iterations, then the value of r is doubled.* We reset r to the population size when the best found solution is improved, assuming that whenever the best incumbent is improved, this means that the population has reached a new unexplored area and the length of the historic period for analysis should be reduced. To reduce the CPU cost, the termination condition is checked only when the value of r is updated.

A single experiment with a GA, given a certain computational budget we will call *a trial*. In the experiments, $N = 30$ independent trials of the GA in each of the three modes were carried out. The GA population size was set to 100. Let the statistic σ be the average relative error $\sum_{k=1}^{30} \frac{f_k - f^*}{30 f^*} \cdot 100\%$, where f_k is the cost of solution found in the k-th trial and f^* is the optimal cost. In what follows, F_{bst} will denote the frequency of obtaining a solution with the best known cost from the literature, estimated by 30 trials.

A statistical analysis of experimental data is carried out using the significance test from [5] (see Ch. 8, §2), which is used to compare two algorithms in terms of probability of finding an optimal or a best-known solution. Let P_1 and P_2 denote the probabilities of success for some Algorithm 1 and Algorithm 2, respectively. The null hypothesis is that $P_1 = P_2$. Under the null hypothesis, the estimate of common success rate is $F = (F_1 + F_2)/2$, where F_1 denotes the frequency of success in N trials of Algorithm 1, and F_2 is the frequency of success in N trials of Algorithm 2. In our case $N = 30$. The difference $F_1 - F_2$ can be expressed in units of the standard deviation as the statistic $A = |F_1 - F_2|/\hat{SD}$, where $\hat{SD} = \sqrt{2F(1-F)/N}$ denotes the estimate of the standard deviation. It is appropriate to assume that A is normally distributed if the number of trials is sufficiently large. The null hypothesis may be tested at a confidence level p by comparison of the value of A to the quantile of the standard normal distribution $z_{p/2}$ (e.g. with $p = 0.05$, we have $z_{0.025} \approx 1.96$). If $A > z_{p/2}$, then the null hypothesis is rejected. In Tables 2–4 and 6, the statistically significant difference (at level $p \le 0.05$) between the frequencies of finding optimal solutions is indicated by "$*$" when mode C is compared to mode A and by "+" when modes C and A are compared.

Table 2: Relative errors and frequencies of finding optima in series *4*,*5* and *6*.

Instance	Single run mode A		Restart mode B			New restart mode C		
	σ	F_{bst}	σ	F_{bst}	t_{avg}	σ	F_{bst}	t_{avg}
401	0.117	0.967	0	**1**	537.6	0	**1**	760.8
402	0.703	0.833	0	**1**	537.6	0	**1***	817.6
403	0.039	0.967	0	**1**	768.0	0	**1**	845.4
404	1.154	**0.767***	0.445	0.3	717.1	0.364	0.4	748.3
405	0.156	0.867	0	**1**	455.4	0.039	0.967	828.0
406	0.554	0.5	0	**1**	642.4	0	**1***	770.0
407	0.186	0.867	0	**1**	488.9	0	**1***	784.4
408	0.589	0.933	0	**1**	497.6	0	**1**	798.8
409	1.451	0.533	0.125	0.867	738.7	0.062	**0.933***	796.6
410	0.117	0.933	0	**1**	657.0	0	**1**	803.8
501	1.976	0.433	0	**1**	752.6	0.079	0.967*	803.8
502	1.987	0.667	0.066	**0.933**$^+$	691.2	0.397	0.633	793.8
503	0	**1**	0	**1**	413.8	0	**1**	777.7
504	1.033	0.333	0.041	0.967	586.0	0	**1***	839.0
505	4.171	0.333	2.559	0.267	373.4	2.417	**0.367**	684.9
506	0	**1**	0	**1**	249.1	0	**1**	759.7
507	0.785	0.233	0.307	0.7	889.4	0.273	**0.733***	792.7
508	0.486	0.767	0	**1**	518.0	0	**1***	766.3
509	0.251	0.767	0	**1**	317.3	0.036	0.967*	770.4
510	1.245	0.567	1.132	**0.6**	457.6	1.396	0.5	765.3
601	0.87	0.867	0.145	**0.967**	174.9	0.29	0.933	711.0
602	0.89	0.667	0	**1**	165.9	0	**1***	667.5
603	0	**1**	1.034	0.833	145.0	0	**1**$^+$	661.9
604	0	**1**	0	**1**	127.8	0	**1**	1014.9
605	1.677	0.7	0	**1**	189.1	0	**1***	664.5
average	0.817	0.74	0.231	0.901	485.2	0.217	0.892	773.9

5.1 Experiments with Randomly Generated Problems

In the experiments described in this subsection, the total budget, counting all restarts during the GA trial, was set to 10000 tentative solutions. For all randomly generated problems, the mutation probability p_{m} is 0.1. Tables 2, 3 and 4 show the results of experiments with series *4-6*, *a,c,d* and *e,f,g,h.* The optimal solution values are known for all instances of these series (see e.g. [38]). The highest frequency of obtaining optima is indicated in bold. In series *b,* all runs yielded optimal solutions regardless of a restart rule, therefore we skip series *b* in Table 3. Tables 2, 3 and 4 also show the average number of iterations t_{avg} (over 30 runs) that were made until the restart rule terminated a GA. The symbol "–" indicates the cases where no restarts were made, until an optimum was found, or the total budget of 10000 iterations was reached.

Comparing the GA results in modes A and C reported in Tables 2–4, one can see that among 37 instances, where these two modes yield different frequencies F_{bst}, mode C has a higher value F_{bst} in 31 cases and in 16 out of these 31 cases the difference is statistically significant. Mode A has a statistically significant advantage to mode C only on a single

Table 3: Relative errors and frequencies of finding optima in series *a,c* and *d*.

Instance	Single run mode A		Restart mode B			New restart mode C		
	σ	F_{bst}	σ	F_{bst}	t_{avg}	σ	F_{bst}	t_{avg}
a1	1.344	**0.133**	1.067	0.1	501.9	1.146	0.033	773.1
a2	0.476	0.867	0.159	**0.933**	845.2	0.278	0.833	810.8
a3	1.207	0.2	0.862	0.333	541.8	0.819	**0.367**	756.3
a4	2.051	0.567	0.256	**0.8**	672.3	0.427	0.667	797.4
a5	0.89	0.3	0.466	0.633	465.9	0.381	**0.7***	752.9
c1	0.661	0.8	0.088	**0.933**	566.9	0.264	0.8	803.8
c2	0.868	0.467	0	**1**	523.9	0	**1***	838.0
c3	1.811	0.533	0.288	**0.767**	763.7	0.37	0.7	889.0
c4	0.091	0.933	0	**1**	392.4	0	**1**	1017.0
c5	0.419	0.8	0	**1**	440.7	0	**1***	839.4
d1	0	**1**	0	**1**	–	0	**1**	–
d2	0	**1**	0	**1**	120.9	0	**1**	–
d3	0.417	0.9	3.333	0.333	127.8	0	**1**$^+$	709.1
d4	0	**1**	0.806	0.833	108.3	0	**1**$^+$	932.4
d5	0	**1**	0	**1**	120.1	0	**1**	–
average	0.682	0.7	0.488	0.778	–	0.246	0.807	–

Table 4: Relative errors and frequencies of finding optima in series *e,f* and *h*.

Instance	Single run mode A		Restart mode B			New restart mode C		
	σ	F_{bst}	σ	F_{bst}	t_{avg}	σ	F_{bst}	t_{avg}
e1	0	**1**	0	**1**	–	0	**1**	–
e2	0	**1**	0.667	0.933	114.5	0	**1**	662.7
e3	0	**1**	0	**1**	104.4	0	**1**	–
e4	0	**1**	0	**1**	102.3	0	**1**	–
e5	0	**1**	0	**1**	203.3	0	**1**	–
f1	0	**1**	0	**1**	101.0	0	**1**	–
f2	0	**1**	0	**1**	–	0	**1**	–
f3	0	**1**	0	**1**	101.7	0	**1**	473.1
f4	0	**1**	0	**1**	102.8	0	**1**	–
f5	22.308	**0.033**	23.077	0	110.9	22.308	**0.033**	579.9
g1	0.852	0.733	0	**1**	198.0	0	**1***	816.0
g2	2.078	0.4	1.234	0.367	296.0	1.104	**0.433**	874.6
g3	2.831	0.033	2.53	0	268.6	1.747	**0.067**	892.3
g4	2.262	0.4	0.595	**0.833**	228.1	1.071	0.633	819.3
g5	0.952	0.667	0	**1**	271.6	0	**1***	905.8
h1	4.762	0	4.762	0	136.3	4.762	0	777.0
h2	4.762	0	4.762	0	124.1	4.762	0	406.8
h3	0.169	**0.967**	3.898	0.233	142.0	0.508	0.9$^+$	825.6
h4	0	**1**	2.759	0.467	139.9	0	**1**$^+$	829.2
h5	0	**1**	0	**1**	111.0	0	**1**	–
average	2.049	0.712	2.214	0.692	–	1.813	0.753	–

Table 5: Average time (sec.) and number of covers generated till finding optima

Instance	Single run mode A		Restart mode B		New restart mode C	
	opt. time	tent. sol.	opt. time	tent. sol.	opt. time	tent. sol.
503	0.53	1364.0	1.09	1996.9	**0.39**	1256.0
603	0.09	2264.5	0.11	2962.6	**0.08**	1869.5
d1	0.04	129.5	0.04	136.1	**0.03**	127.9
d2	0.34	325.6	0.66	731.8	**0.23**	320.5
d5	0.05	174.7	0.05	166.5	**0.04**	144.4
e1	**0.01**	101.8	0.02	101.2	0.02	101.6
e3	**0.19**	249.0	0.35	558.1	0.24	293.5
e4	**0.18**	245.5	0.38	690.4	0.31	257.6
e5	**0.05**	137.7	**0.05**	137.3	**0.05**	132.6
f1	0.11	177.1	0.19	231.6	**0.06**	152.5
f2	**0.03**	107.3	**0.03**	107.4	**0.03**	107.6
f3	1.04	509.6	1.1	1596.8	**0.64**	594.3
f4	0.34	243.5	0.5	592.1	**0.20**	281.9
h5	0.31	235.0	0.45	423.4	**0.25**	211.3
average	0.236	447.486	0.359	745.157	0.184	417.943

instance 404.

In these tables, modes B and C show different frequencies F_{bst} on 28 instances. On 16 of these 28 instances, mode C has a higher value F_{bst} than mode B and in 5 out of these 16 cases the difference is statistically significant. Mode B has a statistically significant advantage to mode C only on a single instance 502. In terms of percentage of deviation σ, averaged over all instances of series *4-6*, *a,c,d* and *e,f,g,h*, mode C gives the least error (see the row "average").

The restart mode C terminated the GA later than mode B in most of the cases. This seems to be natural because the restart rule of mode B is based solely on the values of objective function of the obtained solutions, while the new restart rule uses the information about the generated covers and their frequencies. In particular, in those cases where mode C yielded statistically greater frequency of finding the optima, compared to mode B, the average number of iterations t_{avg} in mode C was 4-8 times larger than in mode B.

To evaluate the overall CPU cost of NBGA execution with the new restart rule (including the CPU time for Schnambel Census estimate $\hat{\nu}^{\text{ML}}$), we considered those instances, where the frequency of finding optima was equal to 100% (see Table 5). On average, in terms of the CPU time and in terms of the total number of tentative solutions made until first visiting an optimum, mode C tends to be more efficient than the other two. Here the CPU time is not proportional to the number of tentative solutions because the individuals of initial populations are computed faster than the offspring in the main loop of NBGA.

5.2 Experiments on Combinatorial Unicost Problems

The three modes of running the GA were also tested on two series of combinatorial unicost SCP instances. In the experiments described in this subsection, the total budget, for each GA trial, was set to 10000 tentative solutions as in the previous subsetion. The set of

Table 6: Frequencies of finding optimal solutions in series *clr* and *Stein*.

Instance	Rows m	Columns n	Best known solution	Single run mode A	Restart mode B		New restart mode C	
				F_{bst}	F_{bst}	t_{avg}	F_{bst}	t_{avg}
clr10	511	210	25	0.2	0.033	119.8	**0.433**$^{*+}$	746.2
clr11	1023	330	23	**0.433**	0	104.7	**0.433**$^{+}$	724.0
clr12	2047	495	23	0.467	0.267	124.1	**0.567**	775.4
clr13	4095	715	23	**0.033**	0	113.9	0	899.7
Stein27	117	27	18	**1**	**1**	–	**1**	–
Stein45	330	45	30	0.867	0.967	101.0	**1***	636.5
Stein81	1080	81	61	**1**	**1**	101.6	**1**	1328.5
Stein135	3015	135	103	0	0	111.6	0	684.8
Stein243	9801	243	198	0.833	0.667	111.7	**1**$^{*+}$	680.0

problems *clr* is derived from one of the well-known questions of P. Erdös [8], stated as a unicost SCP (see e.g [15]) which turns out to be very hard for exact solvers. The series of problems *Stein* arise from Steiner triple systems, and it was proposed in [12] as a set of examples of hard problems that can be used for evaluation of different algorithms. Table 6 shows the dimensions of the unicost SCP instances considered in this paper. Columns F_{bst} and t_{avg} have the same meaning as in tables above. To the best of our knowledge, optimality of the best-known solutions indicated in this table is proven only for *Stein* series [25]. Again we use the population size of 100 individuals and the total budget of GA iterations, equal to 10000 in each trial. The mutation probability p_{m} is set to 0.01 for all instances.

On the combinatorial unicost instances, restart mode C shows better or equal results compared to the other two modes, except for a single instance clr13. Column t_{avg} indicates that the restart rule of mode B triggers too early, precluding the GA from finding good solutions. This is more evident than in the case of randomly generated instances because the combinatorial unicost instances tend to have large plateaus of solutions with equal objective function values. On clr13, the best known solution was found only in mode A and it took more than 4000 iterations. Mode C was irrelevant on this instance, probably due to a negative bias of the maximum likelihood estimate $\hat{\nu}^{\text{ML}}$ from formula (1) (see [11, 31, 32]).

5.3 Reduced Iterations Budget

We repeated the experiments of Subsections 5.1 and 5.2 with the total iterations budget reduced to 5000 and obtained the following results.

Comparing the GA results on randomly generated problems in modes A and C, among 41 instances, where these two modes yielded different frequencies F_{bst}, mode C had a higher value F_{bst} in 29 cases and in 15 out of these 29 cases the difference was statistically significant. Mode A had a statistically significant advantage to mode C only on instances 404 and a4.

Modes B and C showed different frequencies F_{bst} on 35 randomly generated instances. On 21 of these instances, mode C had a higher value F_{bst} than mode B and in 5 out of these 21 cases the difference was statistically significant. Mode B had a statistically significant advantage to mode C only on two instances 502 and a4. In terms of percentage of deviation σ, averaged over all instances of series *4-6*, *a,c,d* and *e,f,g,h*, mode C gave the least error.

On the combinatorial unicost instances, none of the modes significantly outperformed the other ones, presumably because 5000 iterations is a relatively short budget for these series and the restarts do not improve the results.

5.4 Increased Iterations Budget

We also repeated the experiments of Subsections 5.1 and 5.2 with the total iterations budget increased to 15000 and obtained the following results.

Comparing the GA results on randomly generated problems in modes A and C, among 38 instances, where these two modes yield different frequencies F_{bst}, mode C has a higher value F_{bst} in 35 cases and in 20 out of these 35 cases the difference is statistically significant. Mode A has a statistically significant advantage to mode C only on three instances 404, 502 and a1.

Modes B and C showed different frequencies F_{bst} on 22 randomly generated instances. On 14 of these instances mode C has a higher value F_{bst} than mode B and in 3 out of these 14 cases the difference is statistically significant. Mode B has a statistically significant advantage to mode C on two instances 502 and 507. In terms of percentage of deviation σ, averaged over all instances of series *4-6*, *a,c,d* and *e,f,g,h*, mode C gives the least error.

On the combinatorial instances, mode C shows better or equal results compared to the other two modes, except for a single instance Stein45, where it was outperformed by mode B but the difference in this case was not statistically significant. At the same time, mode C had a statistically significant advantage to mode B in 4 cases. Mode B performed poorly on these instances, showing a relative error on average by a factor of 1.5 greater than modes A and C.

Experiments with randomly generated SCPs are summarized in Figures 1, 2.

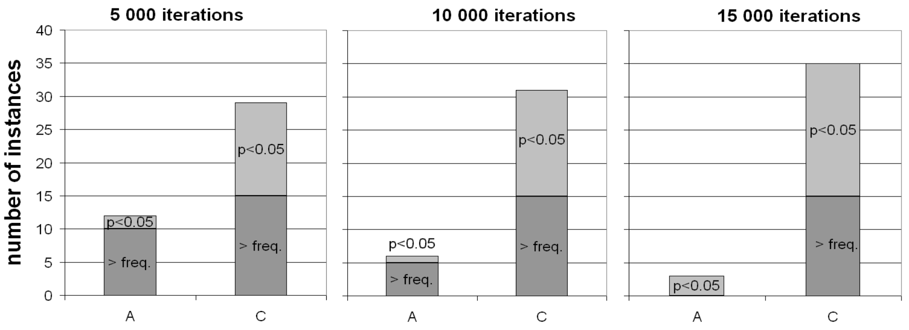

Figure 1: Number of instances where modes A or C have a greater frequency of finding optima for randomly generated SCPs: The top segments show the instances with statistically significant difference in frequency

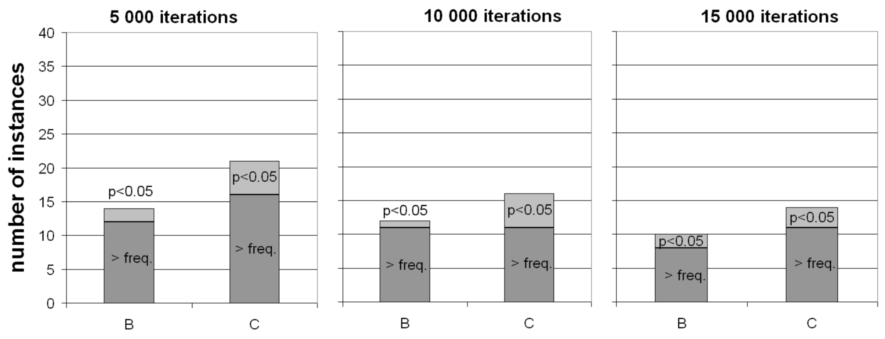

Figure 2: Number of instances where modes B or C have a greater frequency of finding optima for randomly generated SCPs: The top segments show the instances with statistically significant difference in frequency

6 Conclusions

A new restart rule is proposed for Genetic Algorithms, using the Schnabel Census method, originally developed for statistical estimation of size of animal populations. The performance of the new restart rule is demonstrated on a steady-state GA with non-binary representation for the Set Cover Problem. Computational experiments show a significant advantage of the GA with the new restarting rule over the GA without restarts and the GA restarted as soon as the current iteration number becomes twice the iteration number of the currently best incumbent.

Applying Schnabel Census in this research is an attempt to further benefit from the convergence between computer science and biology. That interface had been crucial for evolutionary computation to emerge, but was scarcely maintained afterwards [26]. As the present research shows, developing this transdisciplinary integration can be productive.

Further improvements of the restart strategy are expected via usage of less biased methods, developed for estimation of the number of local optima and for estimation of the abundance of populations, see e.g. [11, 17]. The further research might address the usefulness of the proposed restart rule for other types of evolutionary algorithms and other optimization problems. The Schnabel Census method may be also applicable for a dynamical control of the mutation, if instead of restarting, the GA would increase the mutation probability by a certain factor.

Acknowledgements

This research is supported by the Russian Science Foundation grant 17-18-01536.

References

[1] Alexandrov, D., Kochetov, Y.: Behavior of the ant colony algorithm for the set covering problem. In Proc. of Symp. on Oper. Res. (SOR'99), pp. 255–260. Springer, Berlin (2000)

[2] Balas E., Niehaus W.: Optimized crossover-based genetic algorithms for the maximum cardinality and maximum weight clique problems. J. Heuristics 4(2), 107–122 (1998)

[3] Beasley, J. E.: OR-Library: distributing test problems by electronic mail. J. Oper. Res. Soc. 41(11), 1069–1072 (1990)

[4] Beasley, J.E., Chu, P.C.: A genetic algorithm for the set covering problem. European Journal of Operation Research 94(2), 394–404 (1996)

[5] Brown, B. W., Hollander, M.: Statistics: A Biomedical Introduction. John Wiley & Sons, Inc. (1977)

[6] Caprara, A., Fischetti, M., Toth, P.: Heuristic method for the set covering problem. Operations Research 47(5), 730–743 (1999)

[7] Craig, C.C.: Use of marked specimens in estimating populations. Biometrika 40(1-2), 170–176 (1953)

[8] Erdös, P.: On a combinatorial problem I. Nordisk Mat. Tidskrift 11, 5–10 (1963)

[9] Eremeev, A.V.: A genetic algorithm with a non-binary representation for the set covering problem. In Proc. of OR'98, pp. 175–181. Springer-Verlag (1999)

[10] Eremeev, A.V., Kovalenko, Yu.V.: Genetic algorithm with optimal recombination for the Asymmetric Travelling Salesman Problem. In Proc. of The 11th International Conference on Large-Scale Scientific Computations (LSSC-17). LNCS, vol. 10665. pp. 332–339. Springer, Cham (2018).

[11] Eremeev, A.V., Reeves, C.R.: Non-parametric estimation of properties of combinatorial landscapes. In Applications of Evolutionary Computing: Proceedings of EvoWorkshops 2002. LNCS, vol. 2279. pp. 31-40. Springer-Verlag, Berlin (2002)

[12] Fulkerson, D.R., Nemhauser, G.L. Trotter, L.E.: Two computationally difficult set covering problems that arise in computing the 1-width of incidence matrices of Steiner triple systems. Math. Programming Study 2, 72–81 (1974)

[13] Garey, M.R., Johnson, D.S.: Computers and Intractability. A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, San Francisco (1979)

[14] Garnier, J., Kallel, L.: How to detect all maxima of a function? In: Proceedings of the Second EVONET Summer School on Theoretical Aspects of Evolutionary Computing (Anvers, 1999), pp. 343–370 Springer, Berlin (2001)

[15] Grossman, T., Wool, A.: Computational experience with approximation algorithms for the set covering problem. Eur. J. Oper. Res. 101(1), 81–92 (1997)

[16] Hampson, S. and Kibler, D.: Plateaus and plateau search in Boolean satisfiability problems: When to give up searching and start again. In: Proceedings of the second DIMACS Implementation Challenge "Cliques, Coloring and Satisfiability", pp. 437–456. American Mathematical Society (1996)

[17] Hernando, L., Mendiburu, A., Lozano J. A.: An evaluation of methods for estimating the number of local optima in combinatorial optimization problems. Evolutionary Computation 21 (4), 625–658 (2013)

[18] Holland, J.: Adaptation in natural and artificial systems. University of Michigan Press (1975)

[19] Hulin, M.: An optimal stop criterion for genetic algorithms: A Bayesian approach. In Proc. of the Seventh International Conf. on Genetic Algorithms (ICGA '97), pp. 135–143. Morgan Kaufmann (1997)

[20] Jansen, T.: On the analysis of dynamic restart strategies for evolutionary algorithms. In Proc. of Parallel Problem Solving from Nature (PPSN VII), LNCS Vol. 2439, pp. 33–43, Springer (2002)

[21] Johnson, N.L., Kotz, S.: Discrete distributions. Wiley (1969)

[22] Luke, S.: When short runs beat long runs. In Proc. of he Genetic and Evolutionary Computation Conf. (GECCO 2001), pp. 74–80. Morgan Kaufmann (2001)

[23] Marti, L., Garcia, J., Berlanga, A. and Molina, J. M.: An approach to stopping criteria for multi-objective optimization evolutionary algorithms: The MGBM criterion. In Proc. of 2009 IEEE Congress on Evolutionary Computation, Trondheim, pp. 1263–1270. IEEE (2009)

[24] Van Nimwegen, E., Crutchfield, J. P., Mitchell, M.: Finite populations induce metastability in evolutionary search. Physics Letters A 229(2), 144–150 (1997)

[25] Ostrowski, J., Linderoth, J.T., Rossi, F., Smriglio, S.: Solving Steiner triple covering problems. Oper. Res. Lett. 39, 127–131 (2011)

[26] Paixao, T. Badkobeh, G., Barton, N. et al.: Toward a unifying framework for evolutionary processes. Journal of Theoretical Biology 383, 28–43 (2015)

[27] Pledger, S.: The performance of mixture models in heterogeneous closed population capture-recapture. Biometrics 61(3), 868–876 (2005)

[28] Reeves, C.R.: The "crossover landscape" and the Hamming landscape for binary search spaces. In Foundations of Genetic Algorithms 7, pp. 81–98, Morgan Kaufmann, San Francisco (2003)

[29] Reeves C.R.: Fitness landscapes and evolutionary algorithms. In Artificial Evolution: 4th Europ. Conf. LNCS Vol. 1829. pp. 3-20, Springer-Verlag, Berlin (2000)

[30] Reeves, C.R.: Landscapes, operators and heuristic search. Annals of Operations Research 86, 473–490 (1999)

[31] Reeves, C.R.: Direct statistical estimation of GA landscape properties. In Foundations of Genetic Algorithms 6, pp. 91–108, Morgan Kaufmann, San Francisco (2001)

[32] Reeves, C.R.: Estimating the number of optima in a landscape, part II: Experimental investigations. Coventry University Technical Report SOR#01-04 (2001)

[33] Schnabel, Z.E.: The estimation of the total fish population of a lake. American Math. Monthly 45, 348–352 (1938)

[34] Seber, G.A.F.: The Estimation of Animal Abundance. Charles Griffin, London (1982)

[35] Vose, M.D.: The Simple Genetic Algorithm: Foundations and Theory, MIT Press (1999)

[36] Vose, M.D., Wright, A.H. Stability of vertex fixed points and applications. In Foundations of Genetic Algorithms 3, pp. 103–114, Morgan Kaufmann, San Mateo, CA (1995)

[37] Wright, A.H., Rowe, J.E.: Continuous dynamical systems models of steady-state genetic algorithms. In Foundations of Genetic Algorithms 6, pp. 209–226, Morgan Kaufmann, San Francisco, CA (2001)

[38] Yagiura M., Kishida M., Ibaraki. T.: A 3-flip neighborhood local search for the set covering problem. Eur. J. Oper. Res. 172, 472–499 (2006)