Restarting a Genetic Algorithm for Set Cover Problem Using Schnabel Census *

Anton V. Eremeev[1,2]

[1] Dostoevsky Omsk State University, Omsk, Russia
[2] The Institute of Scientific Information for Social Sciences RAS, Moscow, Russia
eremeev@ofim.oscsbras.ru

Abstract. A new restart rule is proposed for genetic algorithms (GAs) with multiple restarts. This rule is based on the Schnabel census method, originally developed for statistical estimation of the animal population size. It is assumed that during a number of latest iterations, the population of a GA was in the stationary distribution and the Schnabel census method is applicable for estimating the quantity of different solutions that can be visited with a positive probability. The rule is to restart a GA as soon as the maximum likelihood estimate reaches the number of different solutions observed at the last iterations.
We demonstrate how the new restart rule can be incorporated into a GA with non-binary representation for 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 restarting rule also turned out to be in advantage to restarting the GA as soon as the current iteration number becomes twice the iteration number when the best incumbent was found.

Keywords: Multistart · Schnabel census · Maximum likelihood · Transfer of methods

1 Introduction

Genetic algorithms (GAs) are randomized search heuristics based on biological analogy of selective breeding in nature, originating from the work of J. Holland and applicable to a wide range of optimization problems. The basic components of a GA are a *population* of *individuals* and random operators that are introduced to model mutation and crossover in nature. An individual is a pair of *genotype* g and *phenotype* $x(g)$ corresponding to a search point in the space of solutions D to a given optimization problem. Here g is a fixed length string of symbols (called *genes*) from some alphabet $\mathbf{A}$. The function $x(g)$ maps g to its phenotype

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

In: S. Belim et al. (eds.): OPTA-SCL 2018, Omsk, Russia, published at http://ceur-ws.org

$x(g) \in D$, thus defining a *representation* of solutions in 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 $f : D \to R$ is the objective function of a problem, and $\phi : R \to R$ may be an identical mapping or a monotone function, chosen appropriately to intensify the search. The genotypes of the initial population are randomly generated according to some a priori defined probability distribution.

In this paper, a new restart rule is proposed for the GAs. The rule is based on the Schnabel Census method, originally developed for statistical estimation of size ν of animal populations [6] where one takes repeated samples of size 1 (at suitable intervals of time) and counts the number of distinct animals seen. This method was also adapted to estimate the number of local optima on the basis of repeated local search [4]. Experiments [4] showed that the estimates based on this approach were good for *isotropic* landscapes (i.e. those with uniform basin sizes), but have a negative bias when basin sizes significantly differ.

We show how the new restart rule can be incorporated into a GA with Non-Binary Representation (NBGA) [3] for the Set Cover Problem (SCP). A detailed description of this algorithm is provided in the appendix. Computational experiments on benchmark instances from OR-Library show a significant advantage of the GA with the new restarting rule in comparison to the original version of the GA from [3]. In particular, given equal CPU time, in 35 out of 74 SCP instances the new version of GA had greater frequency of finding optimal solutions and only in 5 out of 74 instances the new version showed inferior results. The new restarting rule also turned out to be in advantage to the well-known rule of restarting the GA as soon as the current iteration number becomes twice the iteration number when the best incumbent was found.

The Schnabel Census method as a means of estimation of the number of unvisited solutions [4], as well as the genetic algorithm, both emerged as a transfer of ideas from biology into computer science. Interestingly enough, in the present paper both methods are combined together. However Schnabel Census, originally developed for estimation of animals population size, is not used for counting individuals here, but for estimation of the number of solutions which may be visited if the distribution of offspring in the GA remains unchanged.

2 Restart Rule Based on Schnabel Census

One of the methods developed in biometrics for statistical estimation of size of animal populations is the *Schnabel Census method* [6]. According to this method, one takes repeated samples of size n_0 (at suitable intervals of time) from the same population and counts the number of *distinct* animals seen. The usual assumption is that the probability of catching any particular animal is the same. The sampled individuals are marked (unless they were marked previously) and returned back into the population. Then statistical estimates for the total number ν of individuals in population are computed on the basis of the number of already marked individuals observed in the samples. 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.

Let r be a parameter that defines the length of a historical period that is considered for statistical analysis. Given some value of a parameter r, the new restart rule assumes that during the r latest iterations, the GA population was at a stationary distribution and all tentative solutions produced on these iterations may be treated analogously to sampled animals in the Schnabel Census method. The Schnabel Census method is applied here to estimate the number ν of different solutions that may be visited with positive probability if the current distribution of offspring remains unchanged.

In the sequel, we assume that in the latest r iterations of a GA we have a sample of r independent offspring solutions and the random variable K is the number of *distinct* solutions among them. In addition we make a simplifying assumption that all solutions that may be generated in the stationary distribution have equal probabilities. The rule consists in restarting the GA as soon as the estimate $\hat{\nu}^{\mathrm{ML}}$ becomes equal to k. The value of parameter r is chosen 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 case it is more appropriate to restart the GA rather than to wait till the population distribution will significantly change.

3 The Genetic Algorithm for Set Covering Problem

In This Section, We Describe the Ga Application Which is Used for testing the new restart rule. The Set Cover Problem (SCP) can be stated as follows. *Consider a set* $\mathcal{M} = \{1, \ldots, m\}$ *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 may be formulated as an integer linear programming problem:

$$\min\{\mathbf{cx} : \mathbf{Ax} \geq \mathbf{e}, \mathbf{x} \in \{0,1\}^n\}, \tag{1}$$

where $\mathbf{c}$ is an n-vector of costs, $\mathbf{e}$ is the m-vector of 1s, $\mathbf{A} = (a_{ij})$ is an $m \times n$ matrix of 0s and 1s, where $a_{ij} = 1$, iff $i \in \mathcal{M}_j$. Using this formulation one can regard SCP as a problem of optimal covering all rows of $\mathbf{A}$ by a subset of columns.

In this paper, we use the same GA for the Set Covering Problem as proposed in our earlier work [3]. The GA is based on the elitist steady-state population management strategy. It is denoted as NBGA because it uses non-binary representation of solutions. The NBGA uses an optimized problem-specific crossover operator, a proportional selection and a mutation operator that makes random changes in every gene with a given probability p_{m}. Each new genotype undergoes greedy improvement procedures before it is added into the population. A detailed description of this algorithm is provided in the appendix.

4 Results of Computational Experiments

The NBGA was tested on OR-Library 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 problems with randomly generated costs c_j from 1,...,100, while *CLR* and *Stein* consist of unicost problems, i.e. here $c_j = 1$ for all j. We compared three modes of GA execution.

- Mode (i): single run without restarts.
- Mode (ii): restarting 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 for restarting random hill-climbing and genetic algorithms.
- Mode (iii): restarting the GA using the new rule proposed in Section 2. The historic period used for statistical analysis was chosen as r latest iterations of the GA, where value 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, we double the value of r.* Here we reset r to the population size, assuming that whenever the best incumbent is improved, the population reaches a new unexplored area and we should reduce the length of the historic period for analysis. If the best incumbent is not improved long enough, our rule extends the historic period. In order to reduce the CPU cost, the termination condition is checked only at those iterations when the value of r is updated. Fig 1 illustrates a typical behavior of parameter r, together with the number of different offspring solutions k and the maximum likelihood estimate $\hat{\nu}^{\mathrm{ML}}$ over a single GA run, until the termination condition was satisfied.

In our experiments, $N = 30$ trials of GA in each of the three modes were carried out. The population size was 100 and the total budget of GA iterations over all runs was equal to 10 000. A single experiment with the given budget we call *a trial*. Let $\sigma := \sum_{k=1}^{30} \frac{f_k - f^*}{30 f^*} \cdot 100\%$, where f_k is the cost of solution found in 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. The statistically significant difference at level $p \leq 0.05$ between the frequencies of finding optimal solutions is reported below.

For all non-unicost problems, we set the mutation probability $p_{\mathrm{m}} = 0.1$. Comparing the GA results in modes (i) and (iii), among 37 instances, where these two modes yield different frequencies F_{bst}, mode (iii) has a higher value F_{bst} in 31 cases and in 16 out of these 31 cases the difference is statistically significant. Mode (i) has a statistically significant advantage over mode (iii) only on a single instance. Numbers of instances where mode (i) or mode (iii) had a higher frequency of finding optima are shown in Fig. 2 (denoted ">freq"). This figure also shows the numbers of instances where these modes had statistically significant advantage (denoted "$p < 0.05$").

Modes (ii) and (iii) show different frequencies F_{bst} on 28 instances. On 16 of these instances mode (iii) has a higher value F_{bst} than mode (ii) and in 5

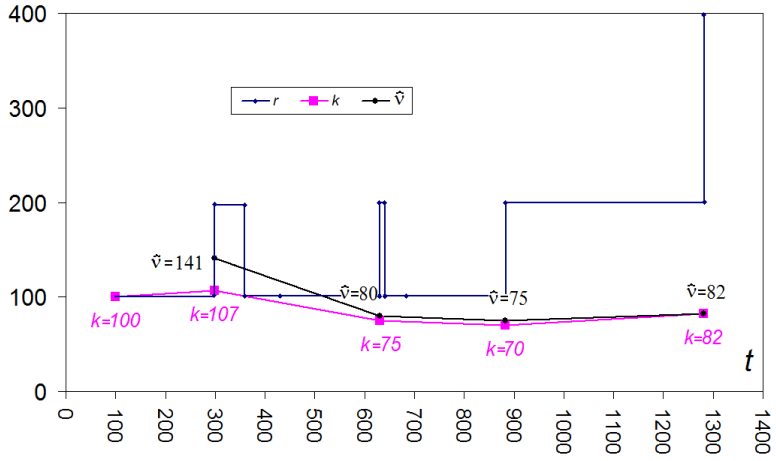

Fig. 1. Dynamics of parameter r, the number of different offspring solutions k and the maximum likelihood estimate $\hat{\nu}^{\mathrm{ML}}$ during 1300 iterations on instance CLR.13, until the termination condition was satisfied ($k = \hat{\nu}^{\mathrm{ML}} = 82$). Here r is reset to the population size 100 when the best incumbent is improved in iterations 359, 430, 639 and 683.

out of these 16 cases the difference is statistically significant. Mode (ii) has a statistically significant advantage over mode (iii) only on a single instance. In terms of percentage of deviation σ, averaged over all instances of series *4-6*, *A,C,D* and *E,F,G,H*, mode (iii) gives the least error.

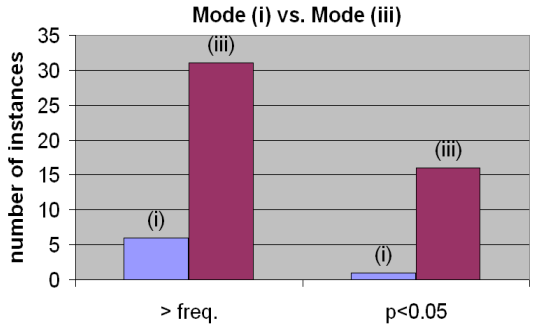

Fig. 2. Comparison of the GA results in modes (i) and (iii)

The three modes of running the GA were also tested on two series of unicost SCP instances *CLR* and *Stein*. Again we use the population size of 100 individuals and the total budget of GA iterations, equal to 10 000 in each trial.

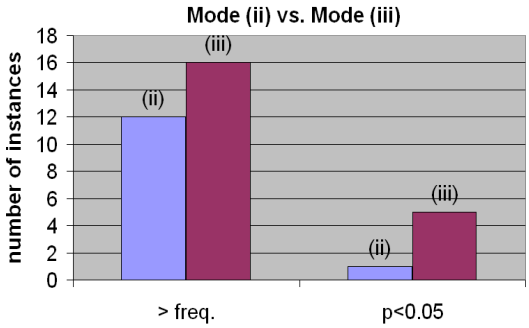

Fig. 3. Comparison of the GA results in modes (ii) and (iii)

The mutation probability p_{m} is set to 0.01 for all instances. On the unicost instances, restart mode (iii) shows better or equal results compared to the other two modes, except for a single instance CLR.13. On CLR.13, the best known solution is found only in mode (i) and it took more than 4000 iterations. Mode (iii) was irrelevant on this instance, which is probably due to a negative bias of the maximum likelihood estimate $\hat{\nu}^{\mathrm{ML}}$ (see [4]).

5 Conclusions

For genetic algorithms, a new restart rule is proposed using the Schnabel Census Method, originally developed to estimate the number of animals in a population. Performance of the new restart rule is shown on a GA with a non-binary representation of solutions for the set cover problem. Computational experiments show a significant advantage of the GA with the new restart rule over the GA without restarting and the GA restarting as soon as the current iteration number becomes twice the number of the current iteration. The new restart rule also demonstrated the most stable behavior compared to the other two GA modes.

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 systematically afterwards. As the present research shows, developing this transdisciplinary integration can be productive.

Appendix

This appendix contains a description of the NBGA [3] for SCP. We will assume that the columns are ordered according to nondecreasing values of c_j-s and if two columns have equal costs then the one which covers more rows stands first.

Denote the set of columns that cover the row i by $\mathcal{N}_i = \{j \in \mathcal{N} : a_{ij} = 1\}$. The NBGA is based on a non-binary representation where the genotype

consists of m genes $g^{(1)}, g^{(2)}, \ldots, g^{(m)}$, such that $g^{(i)} \in \mathcal{N}_i$, $i = 1, 2, \ldots, m$. In this representation $x(\cdot)$ maps a genotype g to the phenotype $\mathbf{x}(g)$, where ones correspond to the columns present in genes of g. Obviously, it is a feasible solution. For local improvements we use greedy heuristics that find approximate solutions to a corresponding reduced version P_g of the given SCP. Problem P_g has a matrix that consists of the columns of $\mathbf{A}$ represented in genes of g.

An improved genotype is added to the population only if there are no individuals with the same phenotype in it yet. Genotypes of Π^0 are generated independently and each gene $g^{(i)}$ is uniformly distributed over $\mathcal{N}_i$.

Consider the population on iteration t as a vector of genotypes of its individuals $\Pi^t = (g_1, g_2, \ldots, g_s)$. Then the fitness function on iteration t is $\Phi_t(g) = cx(g_{l(t)}) - cx(g)$, where $l(t)$ is the index of the individual of the largest cover cost in Π^t.

The selection operator in our GA implements the proportional selection scheme, where the probability to choose the k-th individual is

$$p(g_k) = \Phi_t(g_k) \left(\sum_{l=1}^{s} \Phi_t(g_l) \right)^{-1}.$$

General outline of NBGA

1. While the initial population is not complete do
   - 1.1. Generate a random genotype g.
   - 1.2. Apply the column elimination procedure *Prime* to g and add g to the population.
2. For t:=1 to $t_{\max}$ do
   - 2.1. Choose the parent genotypes g_u, g_v with the help of the proportional selection.
   - 2.2. Produce an offspring g from g_u and g_v using a crossover operator.
   - 2.3. Mutate each gene of g with probability p_m.
   - 2.4. Obtain genotype g', applying column elimination procedures *Greedy* and *Dual Greedy* to g.
   - 2.5. If there are no individuals in Π^t with the phenotype $x(g')$, then
     - 2.5.1. substitute the least fit genotype in Π^t by g',
     - 2.5.2. otherwise substitute the least fit genotype in Π^t by g.

The operators of crossover, mutation and the local improvement procedures *Prime*, *Greedy* and *Dual Greedy* will be described in what follows.

We use an optimized crossover operator, the *LP-crossover,* designed for SCP. The goal of this operator is to find the best possible combination of the genes of parent genotypes g_u and g_v, if it is possible without extensive computations. Consider a problem of optimal crossover P_{oc}, which is a reduced version of the initial SCP but with the covering subsets restricted by the set of indices $\mathcal{N}' = \{j : x(g_u)_j = 1\} \cup \{j : x(g_v)_j = 1\}$.

First, in *LP-crossover* a trivial reduction is applied to P_{oc}. Denote

$$\mathcal{S} = \{i \in \mathcal{M} : |\mathcal{N}_i \cap \mathcal{N}'| = 1\}.$$

Then each row $i \in S$ may be covered by a single column $j(i)$ in P_{oc}. We call $Q = \{j : j = j(i), i \in \mathcal{S}\}$ *a set of fixed columns*. For each $i \in \bigcup_{j \in \mathcal{Q}} \mathcal{M}_j$, one of the columns that cover the gene $g^{(i)}$ in $\mathcal{Q}$ is assigned. We refer to these genes as fixed too. As a result of the reduction we obtain a new subproblem P_{r}, that consists of the rows and columns that were not fixed during this procedure.

Second, we solve the linear relaxation of P_{r}, i.e. an LP problem where the Boolean constraints are replaced with the conditions $x_j \geq 0$ for all j. If the obtained solution $\mathbf{x}'$ turns out to be integer, then it is used to complete the genotype g by an assignment of the non-fixed genes that corresponds to $\mathbf{x}'$. To avoid time-consuming computations we do not solve P_{r} if the number of rows in P_r exceeds a threshold μ (we use $\mu = 150$). If this is the case or if the number of simplex iterations exceeds its limit (equal to 300 in our experiments), or if the solution $\mathbf{x}'$ is fractional, then *LP-crossover* returns an unchanged genotype g_u.

The mutation operator works as follows: Suppose, i-th gene is to be mutated, then the probability to assign a column $j \in \mathcal{N}_i$ to $g^{(i)}$ is

$$p_i(j) = \frac{1}{c_j} \left(\sum_{k \in \mathcal{N}_i} \frac{1}{c_k} \right)^{-1}.$$

We use three greedy-type heuristics to exclude redundant columns from a solution. The most simple heuristic *Prime* starts with a given cover and discards the columns in decreasing order of indices. A column is discarded only if the remaining solution is still a cover. The second heuristic is the well-known *Greedy* algorithm [2]. This algorithm may find a solution which is not minimal, therefore *Prime* is run after it to eliminate redundant columns. The third heuristic we call the *Dual Greedy*. It combines the successive columns discarding of *Prime* and the adaptive columns pricing similar to *Greedy*. Denote the set of columns in the subproblem P_g by $\mathcal{N}' := \{j \in \mathcal{N} : x(g)_j = 1\}$. A cover $\mathcal{J}$ is obtained as follows.

The Dual Greedy Algorithm

1. Set $\mathcal{N}'_i := \mathcal{N}_i \cap \mathcal{N}'$ for all $i = 1, ..., m$, $\mathcal{M}' := \mathcal{M}, \mathcal{J}' := \mathcal{N}', \mathcal{J} := \emptyset$.
2. While $\mathcal{J}' \neq \emptyset$ do
   - 2.1. If there is $i \in \mathcal{M}'$ such that $|\mathcal{N}'_i| = 1$, then
     using the $j \in \mathcal{N}'_i$, set $\mathcal{J} := \mathcal{J} \cup \{j\}, \mathcal{M}' := \mathcal{M}' \backslash (\mathcal{M}_j \cap \mathcal{M}')$.
     Otherwise
     choose $j \in \mathcal{J}'$ such that $\frac{c_j}{|\mathcal{M}_j \cap \mathcal{M}'|} \geq \frac{c_k}{|\mathcal{M}_k \cap \mathcal{M}'|}$ for all $k \in \mathcal{J}'$.
   - 2.2. Set $\mathcal{J}' := \mathcal{J}' \backslash \{j\}$, $\mathcal{N}'_i := \mathcal{N}'_i \backslash \{j\}$ for all $i \in \mathcal{M}_j$.

In Step 1.2 of NBGA we use the *Prime* algorithm, while *Greedy* and *Dual Greedy* are used in Step 2.4 of NBGA (both heuristics are called and the best of the two obtained solutions is accepted to construct a new genotype g').

Before solving the non-unicost problems (where c_j have different values) we apply a core-refining reduction which is similar to the approach used in [1]. The

reduction keeps only the columns belonging to the set $\mathcal{N}_{\text{core}} = \bigcup_{i=1}^{m} \alpha_i^{(10)}$, were $\alpha_i^{(10)}$ is the set of the least 10 indices in $\mathcal{N}_i, i = 1, \ldots, m$.

References

1. Beasley, J.E., Chu, P.C.: A genetic algorithm for the set covering problem. European Journal of Operation Research 94(2), 394–404 (1996)
2. Chvátal, V.: A greedy heuristic for the set covering problem. Mathematics of Operations Research 4(3), 233–235 (1979)
3. 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)
4. Eremeev, A.V., Reeves, C.R.: Non-parametric estimation of properties of combinatorial landscapes. In: Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M. and Raidl, G. (eds.): Applications of Evolutionary Computing: Proceedings of EvoWorkshops 2002. LNCS. vol. 2279, pp. 31–40. Springer-Verlag, Berlin (2002)
5. Paixao, T. Badkobeh, G., Barton, N. et al.: Toward a unifying framework for evolutionary processes. Journal of Theoretical Biology 383, 28–43 (2015)
6. Seber, G.A.F.: The Estimation of Animal Abundance. Charles Griffin, London (1982)