S©MR

СИБИРСКИЕ ЭЛЕКТРОННЫЕ
МАТЕМАТИЧЕСКИЕ ИЗВЕСТИЯ
Siberian Electronic Mathematical Reports
http://semr.math.nsc.ru
ISSN 1813-3304

Vol. 23, No. 1, pp. 554–570 (2026)
https://doi.org/10.33048/semi.2026.23.034

УДК 519.168
MSC 90-08

A RESTART RULE FOR GENETIC ALGORITHMS BASED ON SCHNABEL CENSUS

A.V. EREMEEV, **YU.V. ZAKHAROVA**

Communicated by A.V. Pyatkin

Abstract: A new adaptive restart rule for Genetic Algorithms (GAs) is considered. The rule is based on the Schnabel Census method, originally developed for statistical estimation of a size of animal population. In this paper, the Schnabel Census method is applied as a heuristic 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 a 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 applied in a GA with steady-state replacement scheme, using three different combinatorial optimization problems as examples. Computational experiments on well-known benchmarks show a statistically significant advantage of the GAs with the new restarting rule over the original versions of GAs. The new restart rule also tends to be superior to the well-known rule, which restarts an algorithm when the current iteration number is twice the number of iterations till the current best incumbent was found.

Eremeev A.V., Zakharova Yu. V., A Restart Rule for Genetic Algorithms Based on Schnabel Census.

The work is supported by the Mathematical Center in Akademgorodok under the agreement № 075-15-2025-349 with the Ministry of Science and Higher Education of the Russian Federation.

Received August, 20, 2025, Published June, 5, 2026.

Keywords: genetic algorithm, restart rule, maximum likelihood, combinatorial optimization.

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 [29]. A GA manipulates with a *population* of λ *individuals*, using the random operators that model mutation and crossover in nature. Suppose that a GA is applied to an optimization problem with a finite space of solutions D and an objective function $f : D \to \mathbb{R}$ to be maximized or minimized. In this paper, we assume that the individuals of GA population are tentative solutions in D. The search in GAs is guided by the values of the *fitness* function $\Phi(x) = \phi(f(x))$, which have to be evaluated for all individuals x in the current population Π^t on iteration t. Here $\phi : \mathbb{R} \to \mathbb{R}$ is an increasing function in the case of maximization problem, or a decreasing function in the case of minimization problem. The individuals of the initial population Π^0 are generated according to some a priori defined probability distribution.

A GA may be considered as a Markov chain in a number of ways. For example, the states of the chain may correspond to different vectors of λ solutions that constitute the population Π^t, see e.g. [47]. Some GAs are characterized by reducible Markov chains, and the population of individuals can be "trapped" in an area of the search space, from which the global optimum can not be reached. Other GAs satisfy the ergodicity requirements [48], ensuring the global optimum is found after a finite number of iterations with probability 1. Nevertheless, it was shown in [13] that many elitist evolutionary algorithms, also with the ergodicity property, require exponential time on average to leave some areas associated with local optima before they reach the global optimum. A similar behaviour may be expected from the GAs with crossover.

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 an evolutionary algorithm is stopped and restarted again (a restart rule) was considered in a number of papers on single-objective problems [3, 30, 31, 35, 22] as well as the multi-objective problems [36]. In some works, like [3, 30], the restart times are chosen *adaptively*, i.e. the decision to restart can be made at any iteration, based on the search history of the algorithm on the given problem instance. Other (non-adaptive) restart rules, like [31, 35, 22], are set *a priori*, and do not directly make use of the search history at a given time step. Theoretical analysis of non-adaptive restart rules may be found in [22, 31, 34], but most of the theoretical results are proved asymptotically, assuming that the problem size tends to infinity.

This paper is a journal extended version of the conference publication [17], where the new restart rule was proposed and tested on a GA for the set covering problem. Here in addition to the set covering problem, the performance of the new rule is evaluated on the basis of GAs for the asymmetric traveling salesman problem and total weighted tardiness problem. This rule is based on the Schnabel Census method, transferred into the computer science from the statistics, where it was originally developed for the estimation of a size of animal population [50]. The Schnabel Census, originally developed for estimation of animal population size, is not used to count individuals here (the population size is a known parameter of a GA, kept constant throughout the run), but to estimate the number of different solutions that may be visited with positive probability in this distribution, i.e., to estimate the size of support of this distribution. To make the Schnabel Census method applicable, we accept several simplifying assumptions, in particular, that during the latest iterations, the GA population was generated according to the same distribution.

The new restart 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 behind 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 there are two appropriate ways to continue the search: either to restart the GA with a random new generation, or to modify the distributions of mutation and crossover operators in order to leave the explored area. In the present paper we consider the first approach. The latter approach has received attention in [15, 40, 41].

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 GA [29] is an example of non-elitist GAs. The new restart rule is applicable in both types of GAs, although in the present paper the rule is tested only with elitist steady-state GAs.

The paper has the following structure. In the next section, we briefly describe the general outline of genetic algorithm with steady state population update, which applies to all algorithms considered in this paper. In Section 3, we describe the Schnabel Census and its usage in the new restart rule. Section 4 briefly outlines how this restart rule has been incorporated into a GA for the set cover problem and what computational results were obtained for it. Two applications of the general GA scheme to optimization problems on permutations are described in Section 5. The results of experiments with these GAs, using the new restart rule and other alternatives are described in Section 6. Concluding remarks a given in Section 7.

2 The Genetic Algorithm with Steady State Population Update

The GA starts with a random initial population, and proceeds from the current population to the next one applying selection, crossover (recombination) and mutation. The initial population includes solutions generated randomly or using constructive heuristics. A result of the GA is the best solution found w.r.t. the objective function (fitness function). The number of individuals λ remains constant during the search. In a steady-state GA, only one offspring is generated at each iteration. Algorithm 1 outlines the GA with steady state replacement [12]. Here the choice of parent individuals is made by means of one of the standard selection operators, e.g the tournament selection or proportional selection, or even with the uniform distribution. To keep the population size constant, it is necessary to choose the individuals to be replaced by the offspring.

As in many practical GA applications, in Algorithm 1, before an offspring is added into the population, it undergoes a local improvement procedure. This procedure may implement some local search algorithm (then a GA is usually called a *memetic algorithm* [37]) or a greedy heuristic or may just keep a solution unchanged.

Algorithm 1 Hybrid Evolutionary Algorithm with Steady State Replacement Scheme

1: Construct the initial population Π^0, set $t := 0$.
2: Until a termination condition is satisfied, perform steps
   2.1 Select two parent solutions π^1 and π^2 from Π^t.
   2.2 Apply a mutation operator to each parent solution, obtaining solutions x^1 and x^2.
   2.3 Generate an offspring x' by applying a crossover operator to x^1 and x^2.
   2.4 Perform local improvements on x' if necessary.
   2.5 Replace the individual of lowest fitness in the population Π^t by x', denote the new population Π^{t+1}, set $t := t + 1$.
3: Return the best incumbent solution as a result.

3 Restart Rule Based on Schnabel Census

The new restart rule is based on the Schnabel Census method, which was developed in biometrics for statistical estimation of the size of animal populations [49, 50]. 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 [43, 44] to estimate the number of local optima in combinatorial optimization problems on the basis of repeated local search outcomes with random starting points.

In general, an event of falling into a basin of attraction of a local optimum number i, or the event of catching a particular animal number i in a biological population can be viewed as realization of an integer random variable ξ and one can set up a task to compute a statistical estimate for the number ν of outcomes which have a positive probability measure. Unfortunately, without a-priory information on how the probabilities of different outcomes are related to each other, this problem does not have a satisfactory solution, see e.g. [35]. The reason is that in the general case, if some value i of the random variable ξ has a probability that tends to zero, then with high probability the time till the first observation of this value will tend to infinity. So for any statistical estimate of ν there is a counter-example demonstrating its negative bias. However, assuming a particular relationship on probabilities of different outcomes, one can obtain the maximal likelihood estimate or a confidence interval for ν. In particular, one may assume that all outcomes are equally likely [43, 44], making the so-called *isotropic assumption*, or fit a certain type of parametric distribution (exponential, gamma, lognormal etc.) as in [24]. Non-parametric estimates, such as the bootstrap or the jackknife, may also be employed [21, 39, 50].

In what follows, we will apply the Schnabel Census method to estimate the number ν of different offspring solutions ξ that have a non-zero probability to be created in the current iteration of a GA. Note that given a specific problem instance, a specific population Π^t, and the settings of all tunable parameters of the GA (population size, mutation and crossover rate etc.), it is possible to compute exactly the transition probabilities for the offspring population. However, such analysis is computationally prohibitive for large scale instances, which motivates our choice to use a statistical estimate for ν.

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 (assumption 1).

In what follows, we assume that in the latest r iterations of a GA, the observed sample consists of r independent offspring solutions (assumption 2). Let us define the random variable K as the number of *distinct* solutions in this sample. We will make a simplifying isotropic assumption that all solutions, that may be generated in the current distribution, have equal probabilities (assumption 3). Then, as it was noticed in [11], 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 [32]. The maximum likelihood estimate $\hat{\nu}^{\mathrm{ML}}$ for 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 $\hat{\nu}(r,k) \geq k$ may be found from (1) by the standard one-dimensional optimization methods (see e.g. [43]).

The proposed rule restarts the GA as soon as the estimate $\hat{\nu}(r,k)$ becomes equal to k. The value of r is tuned adaptively during the GA execution. The rationale behind this rule is that once the inequality $\hat{\nu}(r,k) = k$ 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.

Clearly, there are situations where assumptions 1-3 may be irrelevant in practice, but when a GA spends a significant number of iterations without any progress, assumptions 1 and 2 are likely to hold. Assumption 3 is not so likely, but in this work it is accepted for the reason of computational simplicity. We acknowledge that the isotropic assumption might lead to negative bias of the estimate $\hat{\nu}(r,k)$, and more adequate estimates of ν would allow longer runs of the GA.

We compare the new restart rule based on the Schnabel Census with two other alternatives, considering the following three modes of GA execution:

- Mode A. Single run with no restarts.
- Mode B. Restart the GA as soon as the current iteration number becomes twice the iteration number when the best incumbent was found. This adaptive restart rule was used successfully by different authors to restart random hill-climbing method [27] and GAs [3, 19]. 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 adaptive restart rule based on the Schnabel Census. 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.

4 Previous Results for Set Cover Problem

The Set Cover Problem may be formulated 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 is a well-known NP-hard problem [23]. A number of heuristic algorithms have been developed for approximate solving the SCP: Lagrangian relaxation heuristics [9], neural networks [26], local search [57], GAs [5, 16], ant colony algorithms [1] etc. Here we consider the GA for the SCP which was proposed initially without any restart rule in [16]. This GA follows the general scheme of Algorithm 1 and it is denoted as NBGA because of the non-binary representation of solutions [5, 16], 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 problem-specific mutation operator. The offspring are improved by the means of three greedy procedures before they are added into the population.

In [17] the new restart rule was incorporated into the NBGA. Computational experiments on benchmark instances from OR-Library were aimed at comparing three modes of GA execution (A,B and C) given equal computational budget, measured in terms of GA iterations.

A single experiment with a GA, given a certain computational budget, will be called *a trial*. In [17], $N = 30$ independent trials of the NBGA in each of the three modes were carried out. 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 was carried out using the significance test from [6] (see Ch. 8, §2), comparing two algorithms in terms of probability of finding an optimal or a best-known solution.

Comparison of the GA results in modes A and C on these problems showed that among 37 instances, where these two modes yielded different frequencies F_{bst}, mode C had a higher value F_{bst} in 31 cases and in 16 out of these 31 cases the difference was statistically significant. Mode A had a statistically significant advantage to mode C only on a single instance.

Modes B and C show different frequencies F_{bst} on 28 randomly generated instances. On 16 of these 28 instances, mode C had a higher value F_{bst} than mode B and in 5 out of these 16 cases the difference was statistically significant. Mode B had a statistically significant advantage to mode C only on a single instance.

In terms of percentage of deviation from the best-known cost, which was averaged over all instances of series *4-6*, *a,c,d* and *e,f,g,h*, mode C gave the least error. Besides that, it was observed that the restart mode C terminated the GA later than mode B in most of the cases. This is 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 cases where mode C yielded statistically greater frequency of finding the optima, compared to mode B, further analysis included the average number of iterations t_{avg} that were made until the restart rule terminated a GA (over 30 runs). This analysis showed that in mode C the value of t_{avg} was 4-8 times larger than in mode B.

The SCP instances, where the frequency of finding optima was equal to 100%, were considered in order to evaluate the overall CPU cost of the GA execution (including the CPU time for Schnambel Census estimate $\hat{\nu}(r,k)$). 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 tended to be more efficient than the other two modes.

The three modes of running the GA were also tested on two series of combinatorial unicost SCP instances from the OR-library, denoted *clr* and *stein*. In these experiments, restart mode C showed better or equal results compared to the other two modes, except for a single instance clr13 with the largest number of variables. On clr13, the best known solution was found only in mode A and it took more than 4000 iterations. Mode C was inefficient on this instance, probably due to a negative bias of the maximum likelihood estimate $\hat{\nu}(r,k)$ from formula (1). The statistics t_{avg} indicates that the restart rule of mode B triggered too early on the combinatorial problems, precluding the GA from finding good solutions. We suppose that the negative bias of modes B and C on clr13 is more evident than in the case of randomly generated SCP instances because the combinatorial unicost instances tend to have larger plateaus of solutions with equal objective function values.

5 Genetic Algorithms with Restarts for Two Optimization Problems on Permutations

5.1. The Genetic Algorithm for Traveling Salesman Problem. The *Traveling Salesman Problem (TSP)* is one of the classical problems of discrete optimization. Given a complete directed graph G with vertex set $V = \{v_1, \ldots, v_n\}$ and arc set $A = \{(v_i, v_j) : \ v_i, v_j \in V, \ i \neq j\}$. For each arc $(v_i, v_j) \in A$, a weight (arc length) $c(v_i, v_j) \geq 0$ is specified. The objective is to find a Hamiltonian cycle C (traveling salesman's tour) with the minimum total length $\sum_{(v_i,v_j)\in C} c(v_i, v_j)$. If the length of arc (v_i, v_j) differs from the length of the reverse arc (v_j, v_i) for at least one pair of vertices v_i, v_j, then the traveling salesman problem is called *asymmetric* (ATSP). The asymmetric traveling salesman problem arises in various practical applications: scheduling problems considering equipment changeover, vehicle routing problems, shop scheduling problems, goods and services distribution, production planning, etc.

The asymmetric traveling salesman problem is strongly NP-hard [23], which justifies the development of heuristic algorithms and metaheuristics, particularly genetic algorithms. Special-purpose metaheuristics were

proposed for the ATSP: SAX/RAI Memetic Algorithm [7], hybrid approach with position-based representation of solutions [55], GA with GAPX crossover [52], GA with EAX crossover [38], Doubly-Rooted Stem-and-Cycle Ejection Chain Algorithm [45], Variable Neighborhood Search [8] and others.

Our genetic algorithm uses the scheme of Algorithm 1. The initial population is constructed using the "arbitrary insertion" heuristic [56] and the heuristic by W. Zhang [58]. During population evolution, problem-specific mutation and optimal recombination [20] are employed. Individuals in both initial and final populations are improved through local search heuristics based on 3-opt neighborhood.

The selection operator employs a tournament selection. Two mutation operators are implemented, performing random descent in either 3-opt or 4-opt neighborhood. The operators are used with equal probability. The crossover operator solves the optimal recombination problem using the algorithm presented in [18].

The balance between the intensity of local optimization and population-based search is regulated through adaptive restart based on the Schnabel census method (mode C) or the classic restarting rule (mode B).

5.2. Genetic Algorithm for the Total Weighted Tardiness Problem. We consider the following problem of scheduling a set of jobs $\mathcal{J} = \{1, \ldots, n\}$ on a single machine. Job $j \in \mathcal{J}$ is characterized by release date r_j, processing time p_j, positive weight w_j, and due date d_j by which it should be completed. The machine can execute at most one job at a time and preemptions are disallowed. Let C_j denote the completion time of job $j \in \mathcal{J}$, then the tardiness T_j of job j is computed as $\max\{0; C_j - d_j\}$. The goal is to construct a schedule such that the total weighted tardiness $\sum_{j\in\mathcal{J}} w_j T_j$ is minimized. Using the classical three-field notation our problem is denoted as $1|r_j, d_j, w_j|\sum_j w_j T_j$ and called *the one-machine total weighted tardiness problem (TWTP)*. This problem is NP-hard and arises in several practical settings, in particular in the chemical industry [53]. In addition, scheduling models with a single machine also have implications for scheduling research involving multiple machines, where the results obtained from single-machine problems can often be applied to more complex scheduling environments, such as parallel machines, flow shops, job shops, and open shops.

The state-of-the-art metaheuristics for the TWTP are genetic algorithm with local improvements (GA-LI) from [2], population-based variable neighborhood search algorithm (PVNS) [54], multiple VNS algorithm (m-VNS) [10] and hybrid evolutionary algorithm (HEA-DS) from [14].

Our GA uses the scheme of Algorithm 1. For the construction of the initial population we use the following main constructive heuristic: a random non-scheduled job is selected at each step; the position is assigned to this job such that it gives the best objective for the current partial solution. Also, the population contains permutations, that correspond non-decreasing order

of due dates d_j, non-increasing order of w_j/p_j and non-increasing order of $(w_j/p_j) \cdot \exp\{-\max\{0,(d_j-p_j)\}/(n \cdot p_{aver})\}$, where $p_{aver} = \sum_{j=1}^{n} p_j/n$.

Individuals of the last population is improved by a local optimization procedure based on the *swap* neighborhood (positions of two jobs are exchanged) and the *insert* neighborhood (a job is inserted in some other position). For reducing the running time of local search heuristics, we use strategies provided in [25] for $1|r_j = 0, d_j, w_j|\sum_j w_j T_j$. In particular, obviously unpromising moves are excluded and only a part of the neighborhood is considered by moving jobs only from positions located at the distance of no more than $20\%n$ from each other.

We use two mutation operators that perform a random jump within swap or insert neighborhood [25]. The operators are used for mutation with equal probability. The mutation is applied with probability p_{mut}. In the crossover operator we solve the optimal recombination problem by method from [18] with probability p_{cross} (this crossover is called *OCX*, Optimized Cycle Crossover), and use the well-known *OX* (Ordered Crossover) [33] with probability $1 - p_{\text{cross}}$.

The preliminary version of our GA and investigation of various crossover operators were presented at the International conference MOTOR-2023 [59], but restarting rules were not analysed. In the current research GA will be tested in three modes A, B and C as in the case of ATSP.

6 Computational Experiments

6.1. Experiments on Traveling Salesman Problem Instances. Computational experiments were performed on 126 test instances with the known optimum from the library [51], for which the number of vertices is varied from 64 to 315. These instances are generated from one series of instances of the mixed general routing problem with turn penalties (see [51]) and characterized by asymmetric but similar-valued edge weights.

TABLE 1. Comparison of results for algorithms: our GA (modes A, B, C) and GA_{EAX} on ATSP instances

metric	GA (mode C)	GA (mode B)	GA (mode A)	GA_{EAX}
Δ_{aver}	0.00048	0.00091	0.0017	0.00029
Δ_{best}	0.00012	0.00041	0.0010	0.000024
C_{aver}	1229542.62	1229547.39	1229555.99	1229539.10
C_{best}	1229537.12	1229540.44	1229547.33	1229535.35

All modes of our GA with two population sizes (100 and 300) were executed 100 times under the same CPU time limit. The best results were achieved by GA versions with the population size equal to 300. Let C_{best} and C_{aver} denote the best and average objective values over 100 runs on an instance. Let Δ_{aver} and Δ_{best} represent the percentage deviation of C_{aver} and C_{best} from the optimal objective value on an instance. Experimental

results in the context of Δ_{aver}, Δ_{best}, C_{aver} and C_{best} values averaged over all instances are presented in Table 1.

Wilcoxon rank-sum test with significance level $\alpha = 0.05$ was used to verify statistical significance of differences between methods in terms of C_{best} and C_{aver} values across all test instances. Mode C demonstrates better results than mode B, the difference between values of C_{best} and C_{aver} found by algorithms is statistically significant. The versions of the algorithm with restarts outperform the version without restarts, the differences in objective values are also statistically significant. So, the restarts performed in the proposed genetic algorithm allow to avoid localization of search and restore the population diversity, leading to better results when the steady-state population management is used.

Our GA is also compared with one of the most competitive genetic algorithms for ATSP solution proposed in [38]. The algorithm from [38] (GA_{EAX}) uses an EAX crossover operator, it employs a population-based reproduction scheme (the size of population is 300) and uses 3-opt local optimization when building the initial population. This algorithm was implemented in C++ and tested on an Intel Xeon 2.93 GHz computer. Our algorithm was implemented in Java and tested on an Intel Xeon E5420 2.5 GHz computer with 16 GB RAM.

GA_{EAX} was also executed 100 times for each test instance with comparable computation time: our GA was allocated 1.465 times more time than the recorded computation time of GA_{EAX}. This multiplier was chosen based on CPU clock ratios and the fact that Java compiler is estimated to be 25% slower than C++ compiler. The results are presented in Table 1.

The average deviation from optimum Δ_{aver} for GA (mode C) is 0.00048, which is 1.5 times higher than for GA_{EAX}. However, GA (mode C) solved 82 instances with 100% optimality frequency, compared to 68 instances for GA_{EAX}. No significant differences were found between average objective values C_{aver} for GA (mode C) and GA_{EAX}. The same holds for best found objective values C_{best}.

Previously in [20] we tested a similar GA (without postprocessing local optimization) on ATSP instances from TSPLIB library [46]. The collection includes 27 instances from different applications and consists of three series ftv, ft, and rbg, and also individual problems br17, p43, ry48p, kro124p. The dimensions of the instances vary from 17 up to 443. A comparison of the modes B and C was done under the same CPU-time limit. The experiment was executed 1000 times on each instance. The three versions of the algorithm were compared in terms of the frequencies of finding an optimum.

Mode C demonstrated better results than mode B on 12 instances, and the difference between the frequencies of finding an optimum was statistically significant in 5 cases (the significance test [6] is used). Mode B slightly outperformed mode C on four test problems, but the difference was statistically significant only on instance kro124p. Modes B and C demonstrated

Algorithm	Time, sec.	n_{hit}	n_{opt}	Δ_{aver}
GA-LI	29.11	–	100%	0
PVNS	6.19	100%	100%	0
m-VNS	0.908	100%	100%	–
GA (mode C)	0.05	100%	100%	0
GA (mode B)	0.05	100%	100%	0
GA (mode A)	0.92	100%	100%	0
HEA-DS	0.003	100%	100%	0

TABLE 2. Results for TWTP series with $n = 40$.

statistically significantly better results in comparison with mode A on all instances from TSPLIB library.

Moreover, the experimental evaluation from [20] has shown that the proposed algorithm (modes B and C) yields results competitive or superior to those of other state-of-the-art algorithms [7, 55, 52, 38, 45, 8]. Note that most of the TSPLIB instances (except rbg series) exhibit more substantial edge weight differences than instances from [51]. Thus, we conclude that both our GA and GA_{EAX} demonstrate competitive performance across problems with different structures.

6.2. Experiments on Total Weighted Tardiness Problem Instances. We use the TWTP instances with the known optimal solutions from OR-Library.[1] The number of jobs in the series of instances are 40, 50 and 100. Each series contains 125 tests. Our experiment war repeated on each instance for 50 times. The algorithm is set to stop when it obtains an optimal solution or reaches a maximum number of generations, equal to 6000 (such condition is selected in order to compare our results to the ones form [2, 10, 14, 54]). We set $p_{\text{mut}} = 0.15$, $p_{\text{cross}} = 0.8$, and use the tournament selection with the tournament size $s = 5$. Our experiment was carried out on a computer with Intel Core i3-10100F CPU 3.60 GHz, 16 Gb, GA-LI [2] was tested on a computer with Pentium II 400MHz, 96Mb; PVNS [54] was tested on a computer with Pentium IV 3.0GHz, 512Mb; m-VNS [10] was tested on a computer with Intel Core 2.3GHz, 2Gb; HEA-DS [14] was tested on a computer with Xeon E5440 2.83GHz, 16Gb.

The results of the experiment are provided in Tables 2, 3 and 4. Here n_{hit} is the average percentage the cases when an optimal solution was found for an instance out of the given trial runs, n_{opt} is the percentage of instances, where the optimum is found, Δ_{aver} is the average relative deviation from the optimum, Time is the average CPU time for an instance.

The CPU time of mode C is lower than the CPU time of mode B on series with $n = 50,\ 100$ and the difference is statistically significant (the Wilcoxon rank-sum test with significance level $\alpha = 0.05$ was used). GAs with restarts

[1] http://people.brunel.ac.uk/ mastjjb/jeb/orlib/wtinfo.html

Algorithm	Time, sec.	n_{hit}	n_{opt}	Δ_{aver}
GA-LI	41.02	–	99.2%	0.000
PVNS	12.15	100%	100%	0
m-VNS	1.62	100%	100%	–
GA (mode C)	0.12	100%	100%	0
GA (mode B)	0.14	100%	100%	0
GA (mode A)	1.11	94%	100%	0
HEA-DS	0.006	100%	100%	0

TABLE 3. Results for TWTP series with $n = 50$.

Algorithm	Time, sec.	n_{hit}	n_{opt}	Δ_{aver}
GA-LI	118.97	–	66.4%	0.02
PVNS	183.47	100%	100%	0
m-VNS	5.845	99.84%	100%	–
GA (mode C)	1.4	100%	100%	0
GA (mode B)	1.5	100%	100%	0
GA (mode A)	3.18	67%	74%	0.017
HEA-DS	0.032	100%	100%	0

TABLE 4. Results for TWTP series with $n = 100$.

significantly outperformed the GA without restarts on all series in terms of CPU times and frequency of success in finding an optimum.

We see that our algorithm with restarts in mode C demonstrates competitive results, finding the best-known solutions in all cases within comparable time. Our algorithm shows slightly worse results than HEA-DS in terms of the time for reaching of the best-known solutions but the main focus of the present paper was not a competition to other algorithms but experimental evaluation of the impact from usage of the new restart rule.

7 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, although the specifics of this GA is not crucial for application of the proposed restart rule. Computational experiments show a significant advantage of the GA with the new restarting rule over the GA without restarts and the GA which is restarted as soon as the current iteration number becomes twice the iteration number of the currently best incumbent.

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. [21, 28]. The experimental research in future might also 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 variation operators in GAs, if instead of restarting the GA, it will change the probability distribution of mutation and/or crossover.

Funding

The work is supported by the Mathematical Center in Akademgorodok under the agreement № 075-15-2025-349 with the Ministry of Science and Higher Education of the Russian Federation.

Code availability

Java code of the restart rule and GAs from Section 5 are available at https://gitlab.com/YuliaKovalenko−gl/schnabel_census/

References

[1] D. Alexandrov, Yu. Kochetov, *Behavior of the ant colony algorithm for the set covering problem*, In Proc. of Symp. on Oper. Res. (SOR'99), 255–260. Springer, Berlin, 2000. Zbl 0991.90105

[2] S. Avci, M.S. Akturk, R.H. Storer, *A problem space algorithm for single machine weighted tardiness problems*, IIE Transactions, **35**:5 (2003), 479–486.

[3] E. Balas, W. Niehaus, *Optimized crossover-based genetic algorithms for the maximum cardinality and maximum weight clique problems*, J. Heuristics, **4**:2 (1998), 107–122. Zbl 0917.68162

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

[5] J.E. Beasley, P.C. Chu, *A genetic algorithm for the set covering problem*, European Journal of Operation Research, **94**:2, (1996), 394–404. Zbl 0953.90565

[6] B.W. Brown, M. Hollander, *Statistics: A Biomedical Introduction*, John Wiley & Sons, Inc., NJ, 1977. Zbl 1136.62069

[7] L.S. Buriol, P.M. Franca, P. Moscato, *A new memetic algorithm for the asymmetric traveling salesman problem*, Journal of Heuristics, **10**:5 (2004), 483–506. Zbl 1062.90052

[8] E.K. Burke, P.I. Cowling, R. Keuthen, *Effective local and guided variable neighbourhood search methods for the asymmetric travelling salesman problem*, In: EvoWorkshop 2001, Applications of Evolutionary Computing, LNCS, 203–212, Berlin: Springer, 2001. Zbl 0978.68574

[9] A. Caprara, M. Fischetti, P. Toth, *Heuristic method for the set covering problem*, Operations Research, **47**:5 (1999), 730–743. Zbl 0976.90086

[10] T.-P. Chung, Q. Fu, C.-J. Liao, Y.-T. Liu, *Multiple-variable neighbourhood search for the single-machine total weighted tardiness problem*, Engineering Optimization, **49**:7 (2016), 1133–1147.

[11] C.C. Craig, *On the Utilization of Marked Specimens in Estimating Populations of Flying Insects*, Biometrika, **40**:1-2 (1953), 170–176.

[12] L. Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[13] D.-C. Dang, A. Eremeev, P.K. Lehre, *Non-elitist evolutionary algorithms excel in fitness landscapes with sparse deceptive regions and dense valleys*, In: Proceedings of the GECCO, 1133–1141. NY, USA, 2001.

[14] J.L.Z. Ding, T.C.E. Cheng, L. Xu, *A hybrid evolutionary approach for the single-machine total weighted tardiness problem*, Computers and Industrial Engineering, **108** (2017), 70–80.

[15] B. Doerr, H.P. Le, R. Makhmara, T.D. Nguyen, *Fast genetic algorithms*, In: Proceedings of GECCO'17, 777–784, 2017.

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

[17] A. V. Eremeev. *A restarting rule based on the Schnabel Census for genetic algorithms*, Proceedings of International Learning and Intelligenece Optimization Conference (Kalamata, Greece, June 10-15, 2018), Lecture Notes in Computer Science, **11353**, Springer, Cham, 2019, 337–351.

[18] A.V. Eremeev, J.V. Kovalenko, *Experimental evaluation of two approaches to optimal recombination for permutation problems*, In European Conference on Evolutionary Computation in Combinatorial Optimization, 138–153, 2016, Cham: Springer International Publishing.

[19] A.V. Eremeev, Yu. V. Kovalenko, *Genetic algorithm with optimal recombination for the Asymmetric Travelling Salesman Problem.* In: Large-scale scientific computing. 11th international conference, LSSC 2017, Sozopol, Bulgaria, June 5–9, 2017. Revised selected papers, 341–349, Springer, Cham, 2018. Zbl 1437.90136

[20] A.V. Eremeev, Y.V. Kovalenko *A memetic algorithm with optimal recombination for the asymmetric travelling salesman problem* Memetic Computing, **12**:1 (2020), 23–36.

[21] A.V. Eremeev, C.R. Reeves, *Non-parametric estimation of properties of combinatorial landscapes* In Applications of Evolutionary Computing: Proceedings of EvoWorkshops 2002. LNCS, **2279** (2002), 31–40. Zbl 1044.68752

[22] T. Friedrich, T. Kötzing, T. Quinzan, and A.M. Sutton, *Improving the run time of the (1 + 1) evolutionary algorithm with Luby sequences*, In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '18), ACM, New York, NY, USA, 301–308, 2018.

[23] M.R. Garey, D.S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, San Francisco, 1979. Zbl 0411.68039

[24] J. Garnier, L. Kallel, *How to detect all maxima of a function?* In: Proceedings of the Second EVONET Summer School on Theoretical Aspects of Evolutionary Computing (Anvers, 1999), 343–370, Springer, Berlin, 2001. Zbl 1001.68194

[25] M. J. Geiger, *On heuristic search for the single machine total weighted tardiness problem – Some theoretical insights and their empirical verification*, Eur. J. Oper. Res., **207**:3 (2010), 1235-1243. Zbl 1206.90042

[26] T. Grossman, A. Wool, *Computational experience with approximation algorithms for the set covering problem*, Eur. J. Oper. Res., **101**:1 (1997), 81–92. Zbl 0929.90072

[27] S. Hampson, and D. Kibler, *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", 437–456, Providence, RI: American Mathematical Society, 1996. Zbl 0859.68065

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

[29] J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

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

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

[32] N.L. Johnson, S. Kotz, *Distributions in statistics: Discrete distributions*, Boston: Houghton Mifflin Company, 1969.
[33] T. Kellegöz, B. Toklu, J. Wilson, *Comparing efficiencies of genetic crossover operators for one machine total weighted tardiness problem*, Appl. Math. Comput., **199**:2 (2008), 590-598. Zbl 1140.90402
[34] M. Luby, A. Sinclair, and D. Zuckerman, *Optimal speedup of Las Vegas algorithms*, Inf. Process. Lett. **47**:4 (1993), 173–180.
[35] S. Luke, *When short runs beat long runs*, In Proc. of the Genetic and Evolutionary Computation Conf. (GECCO 2001), 74–80, 2001.
[36] L. Marti, J. Garcia, A. Berlanga, and J.M. Molina, *An approach to stopping criteria for multi-objective optimization evolutionary algorithms: The MGBM criterion*, In Proc. of 2009 IEEE Congress on Evolutionary Computation, Trondheim, 1263–1270, 2009.
[37] F. Neri, C. Cotta, P. Moscato, *Handbook of Memetic Algorithms*, Springer, Berlin, Heidelberg, 2012.
[38] Y. Nagata, D. Soler, *A new genetic algorithm for the asymmetric traveling salesman problem*, Expert Syst. with Applications, **10** (2012), 8947–8953.
[39] S. Pledger, *The performance of mixture models in heterogeneous closed population capture-recapture*, Biometrics, **61**:3 (2005), 868–873.
[40] A. Rajabi, C. Witt, *Stagnation detection with randomized local search*, Evol. Comput. **31**:1 (2023), 1–29.
[41] A. Rajabi, C. Witt, *Stagnation detection in highly multimodal fitness landscapes*, Algorithmica, **86** (2024), 2929–2958.
[42] C.R. Reeves, *The "crossover landscape" and the Hamming landscape for binary search spaces*, In Foundations of Genetic Algorithms 7, 81–98, Morgan Kaufmann, San Francisco, 2003.
[43] C.R. Reeves, *Direct statistical estimation of GA landscape properties*, In Foundations of Genetic Algorithms 6, 91–108, Morgan Kaufmann, San Francisco, 2001.
[44] C.R. Reeves, A.V. Eremeev, *Statistical analysis of local search landscapes*, J. Oper. Res. Soc., **55**:7 (2004), 687–693.
[45] C. Rego, D. Gamboa, F. Glover, *Doubly-rooted stem-and-cycle ejection chain algorithm for the asymmetric traveling salesman problem.* Networks. **68**(1), 23–33 (2016)
[46] G. Reinelt, *TSPLIB – a traveling salesman problem library*, ORSA Journal on Computing **3**:4 (1991), 376–384.
[47] G. Rudolph, *Convergence analysis of canonical genetic algorithms*, IEEE Transactions on Neural Networks, **5**:1 (1994), 96–101.
[48] G. Rudolph, *Finite Markov chain results in evolutionary computation: A tour d'horizon*, Ann. Soc. Math. Pol., Ser. IV, Fundam. Inf., **35**:1-4 (1998), 67–89.
[49] Z.E. Schnabel, *The estimation of the total fish population of a lake*, Amer. math. Monthly, **45** (1938), 348–352.
[50] G.A.F. Seber, *The Estimation of Animal Abundance*, Charles Griffin, London, 1982.
[51] D. Soler, E. Martínez, J.C. Micó, *A transformation for the mixed general routing problem with turn penalties*, J. Oper. Res. Soc., **59**:4 (2008), 540–547. Zbl 1153.90363
[52] R. Tinós, D. Whitley, G. Ochoa, *Generalized asymmetric partition crossover (GAPX) for the asymmetric TSP*, In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO '14), 501–508, ACM New York, NY, 2014.
[53] B.J. Wagner, D.J. Davis, H.V. Kher, *The production of several items in a single facility with linearly changing demand rates*, Decision Sciences, **33**:3 (2002), 317–346.
[54] X. Wang, L. Tang, *A population-based variable neighborhood search for the single machine total weighted tardiness problem*, Comput. Oper. Res., **36**:6 (2009), 2105–2110.

[55] L.N. Xing, Y.W. Chen, K.W. Yang, F. Hou, X.S. Shen, H.P. Cai, *A hybrid approach combining an improved genetic algorithm and optimization strategies for the asymmetric TSP*, Eng. Applications of Art. Int., **21**:8 (2008), 1370–1380.

[56] M. Yagiura, T. Ibaraki, *The use of dynamic programming in genetic algorithms for permutation problems*, Eur. Jour. Oper. Res., **92** (1996), 387–401.

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

[58] W. Zhang, *Depth-first branch-and-bound versus local search: A case study*, 17th National Conf. on Artificial Intelligence, 930–935, AAAI Press, Austin, TX, 2000.

[59] Y. Zakharova, *Hybrid evolutionary algorithm with optimized operators for total weighted tardiness problem*, In: Mathematical Optimization Theory and Operations Research. MOTOR 2023, Lecture Notes in Computer Science, **13930**, 224–238, Springer, Cham, 2023.

Antion Valentinovich Eremeev
Novosibirsk State University,
1, Pirogova str.,
Novosibirsk, 630090, Russia
Sobolev Institute of Mathematics,
pr. Koptyuga, 4,
630090, Novosibirsk, Russia,
Email address: eremeev@ofim.oscsbras.ru

Yulia Viktorovna Zakharova
Sobolev Institute of Mathematics,
pr. Koptyuga, 4,
630090, Novosibirsk, Russia,
Novosibirsk State University,
1, Pirogova str.,
Novosibirsk, 630090, Russia
Email address: yzakharova@ofim.oscsbras.ru