Available online at http://scik.org

J. Math. Comput. Sci. 6 (2016), No. 3, 360-376

ISSN: 1927-5307

ARIMA FORECASTING AS A GENETIC INHERITANCE OPERATOR IN FLOATING-POINT GENETIC ALGORITHMS

MEHMET HAKAN SATMAN[1,*], EMRE AKADAL[2]

[1]Department of Econometrics, Istanbul University, Istanbul, Turkey

[2]Department of Informatics, Istanbul University, Istanbul, Turkey

Abstract. In this paper, a new operator is developed for the floating-point genetic algorithms (FPGAs). The operator records the family tree of chromosomes, searches a convenient time series model on it and forecasts offspring which will possibly be generated by usual genetic algorithm operators in future generations. A software package is developed as an implementation and it is freely available for downloading. The results of a suite of simulation study show that the proposed operator has a statistically significant effect on reaching the global optima of test functions in many dimensions of search spaces. The results of simulation study also show that the developed operator increases the search capabilities of GAs when the number of function parameters increase by means of obtaining the global optimum more precisely.

Keywords: Genetic Algorithms; Optimization; Forecasting.

2010 AMS Subject Classification: 65K10, 97R40, 65C05, 91G70.

1. Introduction

Genetic algorithms (GAs) are parallel search and optimization algorithms that mimic the principals of natural selection and genetics [18, 14]. Since GAs are not performed directly

*Corresponding author

Received January 2, 2016

on the goal function to be optimized, they are problem independent. This property makes GAs fully applicable in many problems and the function under consideration can be discrete or non-differentiable and even the optimization problem has not a closed mathematical form. However, they may have many drawbacks such as problem of diversity if the population size is not pre-determined correctly or initial population is not well-randomized over the search space. A randomly generated initial population may not have enough information to reach global optimum using the usual genetic operators such as crossover, mutation and elitism, etc. If the goal function has several local optima, algorithm may unluckily get stuck on one of them or the algorithm may return a result near the global optimum as a result of unsuccessful mutation and crossing-over operations. To cope with this drawbacks, several genetic algorithm parameters should be chosen correctly such as population size, crossover and mutation probability, number of elitist solutions among others and/or some enhanced versions of these operators and hybridization tools can be attached to the algorithm [33, 35, 31].

In GAs, a randomly initial population of candidate solutions is created. A fitness value is then calculated for each single solution to measure the qualities of these solutions on optimizing the goal function. The term *cost function* is directly related to fitness if the objective function under consideration is a minimization. Crossover and mutation operators are then applied to generate new solutions. If the number of elitism is set to a number greater than zero, then best k solutions are directly copied into the next generation without any modification. It is expected that the average fitness of a population at current iteration is better than the one at previous iterations as proved as in the *Schemata Theorem* [14].

Since the floating-point genetic algorithms (FPGAs) are directly applied on the real-valued chromosomes instead of binary representations, phenotype-genotype distinction is not necessary. This is why the new types of crossover and mutation operations are developed for the FPGAs [6, 17, 21, 24, 28]. Although chromosomes coded in lower cardinality alphabets enclose much information about the search space [15], FPGAs can still be considered as genetic algorithms even they have *phenotype* operators [38]. However there are several attempts to mimic *genotype* operators using their *phenotype* counterparts [7, 8, 30].

Since the genetic operators that used in GAs are totally blind and the population at iteration t is generated using the population at iteration $t-1$, they can be modelled using Markov chains [5, 23]. In some modified GAs, it is suggested to use ancestors of a chromosome in the fitness calculation process. It is suggested in [26, p.1–23] to use the *Modified Fitness Value* of a chromosome c which is defined as

$$MFV(c) = \alpha \times fit + \sum_{i=1}^{2} \beta_i p_i + \sum_{j=1}^{4} \gamma_j gp_j, \tag{1}$$

where α, β_i and γ_j are weights, fit is fitness of c, p_i is the fitness of ith parent of c, gp_j is the fitness of jth grand parent of c. Selecting of weights may be either manual or automatic. Automatic selection of weights requires longer chromosomes to evolve. It is reported in [26, p.1–23] that the *MFV* based GAs outperform the classical GAs.

In this paper, we develop a new genetic operator that shares a similar idea used in [26, p.1–23]. The devised operator simply mimics the genetic *inheritance* by recording ancestors of chromosomes and applying a statistical forecast to predict future values, namely offspring, which will possibly be generated by the classical operators in future generations. In other terms, the devised operator can be seen as *directed* or *controlled* crossover and/or mutation operator. The operator is also a *local search* operator which linearly extrapolates an offspring using its family tree. Using such an operator as a local optimizer makes any GA hybrid. In section 2, we introduce the statistical model of inheritance. In section 3, we present the whole algorithm. In Section 4, we perform some simulations on some well-known set of functions to reveal effects of our algorithm. Finally, in Section 5, we conclude.

2. Inheritance Model

Suppose a stationary time series process in a time domain $t = 1, 2, ..., N$ is generated using the formula

$$Y_t = \alpha_0 + \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + ... + \alpha_p Y_{t-p} + \varepsilon_t \tag{2}$$

where Y_t is a random variable observed in a point at time t, α_i for $i = 0, 1, ..., p$ are unknown parameters, ε_t is the stochastic error term with zero mean and constant variance. (2) follows an $AR(p)$ (Auto-Regressive) process in which the random variable of interest is a function of a constant, its previous values at time $t-1, t-2, ..., t-p$ and an error term [2]. Note that the weak stationary process satisfies that all Y_t have constant mean, variance and auto-covariance parameters over the time. If the data generating process of Y_t is

$$Y_t = \phi_0 + \phi_1 \varepsilon_{t-1} + \phi_2 \varepsilon_{t-2} + ... + \phi_q \varepsilon_{t-q} + \varepsilon_t \tag{3}$$

then (3) is said to be an $MA(q)$ (Moving Average) process in which the random variable of interest is a function of a constant, current and previous values of an error term. Note that an $ARMA(p,q)$ model can both include AR and MA terms as in shown in (4).

$$\begin{aligned} Y_t &= \alpha_0 + \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + ... + \alpha_p Y_{t-p} \\ &+ \phi_1 \varepsilon_{t-1} + \phi_2 \varepsilon_{t-2} + ... + \phi_q \varepsilon_{t-q} + \varepsilon_t \end{aligned} \tag{4}$$

Since a data generating process can be estimated better by using only specific lag terms of variables, an $ARMA(A,B)$ model can be used instead where A is a set of p_i and B is a set of p_j for $0 \leq i \leq p$ and $0 \leq j \leq q$. An $ARMA(A,B)$ model is a nested model of an $ARMA(p,q)$. For instance, an ARMA(2, 4, 5) model can be written as shown in (5) and it is nested by the $ARMA(4,5)$ model which is shown in (6).

$$Y_t = \alpha_0 + \alpha_1 Y_{t-2} + \alpha_2 Y_{t-4} + \phi_1 \varepsilon_{t-5} + \varepsilon_t \tag{5}$$

$$\begin{aligned} Y_t &= \alpha_0 + \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \alpha_3 Y_{t-3} \\ &+ \alpha_1 Y_{t-4} + \phi_1 \varepsilon_{t-1} + \phi_2 \varepsilon_{t-2} \\ &+ \phi_3 \varepsilon_{t-3} + \phi_4 \varepsilon_{t-4} + \phi_5 \varepsilon_{t-5} + \varepsilon_t \end{aligned} \tag{6}$$

Now suppose a chromosome is generated using its ancestors as

$$O_t = f(O_{t-1}^M, O_{t-1}^F) + \delta_t, \tag{7}$$

where O_t is the generated chromosome, O_{t-1}^M and O_{t-1}^F are the mother and the father of chromosome O_t, respectively; $f(x,y) = (1-\omega_i)x_i + \omega_i y$ is the crossover function, δ_t is the m-dimensional mutation vector, ω_i is a random variable that follows a $Uniform(0,1)$ distribution if the ith gene is selected for crossover, m is the chromosome length and $i = 1,2,...,m$. Since the FPGAs are not sexist, labels M and F are assigned randomly when a selection operator is applied and an M labelled chromosome would be labelled as F in previous generations or vice versa. This property takes into account the non-linearity as follows:

$$O_t = \begin{cases} f(O_{t-1}^{(1)}, O_{t-1}^{(2)}) + \delta_t & , \quad if\ O_{t-1}^{(1)}\ is\ mother \\ g(O_{t-1}^{(1)}, O_{t-1}^{(2)}) + \delta_t & , \quad if\ O_{t-1}^{(2)}\ is\ mother \end{cases} \tag{8}$$

where $g(x,y) = \omega_i x + (1-\omega_i)y$. Equation (8) can be expressed in a single line as

$$O_t = I \times f(O_{t-1}^{(1)}, O_{t-1}^{(2)}) + (1-I) \times g(O_{t-1}^{(1)}, O_{t-1}^{(2)}) + \delta_t \tag{9}$$

where I is a function that returns 1 if O_{t-1} is mother, otherwise, returns 0. Note that since the predicted value of O_{t+1} is a function of its whole ancestors, a huge number of parameters should be estimated depending on the current number of generations including the I parameter which is discrete.

For simplicity, we suppose the inheritance model is based only on a chromosome's father or mother and the omitted ancestor is handled by the error term and its lagged values as shown in the formula

$$O_t = \alpha + AR(S_1) + MA(S_2) + \varepsilon_t \tag{10}$$

where S_1 and S_2 are sets of lagged terms of the chosen ancestor and stochastic error term, respectively. Once the correct variable set and parameters of model (10) is estimated, the future value O_{t+1} can be predicted. In GAs, it is expected that the average fitness of a population is

better than the average of a population in previous generations. Since the fitness is a function of parameters in chromosomes, time series of the parameters are not stationary, that is, at least the first moment of a time series changes by time. The *I* term in ARIMA modelling handles this issue. Addition to this, chromosomes that created by crossover operator extends its ancestors in previous generations, that is, using a family tree or at least only a branch of tree can be used to estimate offspring. The *AR* term in ARIMA modelling stands for this issue. As it is mentioned before the (10), the other variables that affects the offspring can be summed up by *MA* terms which are based on the lagged values of the error term.

Although the simplified model given in (10) presents an intelligent crossover like operator, it is not that simple to estimate because of unknown lagged term sets of S_1 and S_2. *Auto-correlation function - ACF* and *Partial Auto-correlation function - PACF* can be manually analysed for determining the correct *MA* and *AR* terms, respectively [3, 22]. Determining the lags of model (6) is also considered as an optimization problem in the literature. [25] and [11] suggested to use a genetic algorithm with binary chromosomes of candidate models to identify correct lag terms by minimizing Schwarz's (Bayesian) Criterion. [13] devised a 2-stage algorithm in which a set of integers indicating the lag terms is determined using simulated annealing in the first stage and estimating the parameters using a genetic algorithm in the second stage. [20] used a step-wise model selection algorithm and packaged in an R [27] library [37].

The average fitness of a population P_t in generation t is expected to be higher than the average fitness of P_{t-1}. If the initial population is well-scattered in the search space, differences between the average fitness values in early iterations are higher than the ones in later iterations. This causes the *sudden level shifts* [12] to be appeared in ancestors of chromosomes in early stages of GAs. In the design of the operator, we apply robust filtering to extract signals from the time series of ancestors. *Repeated Median*, *Least Median of Squares Regression*, *Least Trimmed Squares Regression* and *Deepest Regression* can be used in robust filtering of signals in time series data [12, 4].

3. Proposed Algorithm

The proposed operator consists on recording ancestors of chromosomes, searching a convenient ARIMA model on a family tree of chromosomes and applying forecasts to generate offspring more rapidly before the classical crossover and mutation operators generated them. We also develop a software package [32] to reveal the effect of the proposed operator. Since the algorithm requires many linked lists and complex data structures, many parts of the algorithm are implemented in *C++* and the classes and the methods are wrapped in *R* language using the *Rcpp* package [9].

A diagram and the pseudo-code of the proposed algorithm is shown in Figure 1 and Algorithm 1, respectively. The description of the whole algorithm in great detail is as follows: An initial population of chromosomes is generated using the user-defined ranges for all genes. After fitness calculations, the best chromosome is selected for ARIMA forecasting. If the length of the family tree of the selected chromosome is bigger or equal than the *minimum forecast length* then the forecast process starts. Each gene of the selected chromosome is forecasted respect to the *probability of forecasting*. If a gene is subject to be forecasted, a robust filtering is applied on the family tree. This process is necessary because of the *sudden level shifts* in very early stages of the GAs. After robust filtering, the best ARIMA model is searched and the new offspring is generated by forecasting using the model. Forecasting the offspring is simply by-passing the usual crossover and mutation operators, that is, the linear combination of genes are used to generate future values which will then possibly be generated by the GA operators. After generating a single offspring, usual crossover and mutation operators are applied using a tournament selection. Algorithm continues while the stopping criterion is not met.

In our implementation, we use the *R* packages *robfilter* [10] for robust filtering stage and *forecast* [19] for automatic ARIMA model selection and forecasting, respectively. *C++* classes of *Chromosome* and *Population* are wrapped in *R* language [32].

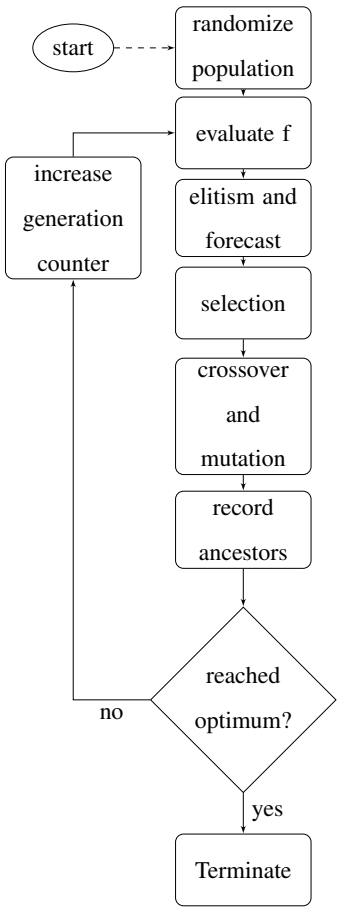

FIGURE 1. FPGA with forecast operator

```
initialize random population;
while currentIter < maxNumberOfIterations do
    calculateFitness(all);
    best := applyElitism()[0];
    family := getAncestors(best);
    offspring := SearchModelAndForecast(family);
    calculateFitness(offspring);
    selection();
    crossover();
    mutation();
    recordAncestors();
end
```

Algorithm 1: Pseudo-code of devised algorithm

4. Simulations

We perform a simulation study on a suite of well-known test functions which are previously used for comparing performances of evolutionary optimization algorithms [16, 34, 36, 29, 1]. These functions are reported in Table 1. As in seen in Table 1, the suite includes both discrete, continuous and non-differentiable functions in several domains. The common property of these functions is that they are 0 at their global minimum.

We compare the performances of FPGAs with or without the inheritance operator. Functions are optimized by the algorithms for $p = 10, 25, 50$ parameters. Each single configuration is repeated 500 times. We set the crossover and the mutation probability to 1 and 0.05, respectively. The best chromosome is directly copied into the next population, that is, number of elitism is 1 for all cases. The forecasting probability parameter fcp is set to 0 for classical FPGA. In order to measure the effect of this operator we set fcp to 0.05. Simulation results are reported in Table 2, 3 and 4.

Table 2 summarizes the simulation results that are performed for the 10-parameters versions of the functions. For each single function, the table includes two rows for the cases in which the devised operator is whether applied or not. Descriptive statistics of final results returned by the test functions are also reported. Average values of standard FPGA for Maxmod, Rastrigin, Rosenbrock and Schaffer are significantly better than the devised GA[1]. Addition to this, FGPA with the devised operator has the smallest minimum value except Maxmod. Contrarily, standard FPGA has a better performance when the maximum values are considered for functions Ackley, Hyperellipsoid, Levy, Maxmod, Rosenbrock, Schaffer, Sphere and Sumsquares. Differences between the mean and the median values are considerably small for all rows, that is, algorithms do not generate extreme results in iterations. It can be said that the GA with the devised operator is successful in 10 of 14 scenarios when the statistical evidence is considered which is based on the averages.

Table 3 summarizes the simulation results that are performed for the 25-parameters versions of the functions. It is shown in Table 3 that the minimum, mean, median and the maximum values are better when the devised operator is applied for all of the functions. The hypothesis of equality of location parameters is also rejected[2] for all cases after performing the Wilcoxon test for all test functions. The results reported in Table 4 are similar with the results reported in Table 3. Addition to this, when the number of parameters p is 50, differences of descriptive statistics are revealed more clearly. In epitome, applying the devised operator has a significantly better impact on optimizing the test functions in higher number of dimensions.

5. Conclusion

The idea of including the fitness values of ancestors of chromosomes in selection procedure is not new. Standard GA operators in floating-point optimization are totally blind. However, a generated offspring is a linear combination of its parents and a random error term which

[1]The null hypothesis of equality of location parameters is rejected for all cases for $\alpha = 0.01$ significance level using the Wilcoxon test. The alternative hypothesis is inequality of location parameters of final values returned by the functions

[2]p-value < 0.01

Function	Definition	Domain
Ackley	$20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	$-30 \leq x_i \leq 30$
Bohachevksy	$\sum_{i=1}^{n}(x_i^2 + 2x_{i+1}^2 - 0.3\cos(3\pi x_i) - 0.4\cos(4\pi x_{i+1}) + 0.7)$	$-50 \leq x_i \leq 50$
Griewank	$1 + \frac{1}{4000}\sum_{i=1}^{n}x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})$	$-600 \leq x_i \leq 600$
Holzman	$\sum_{i=1}^{n} i x_i^4$	$-10 \leq x_i \leq 10$
Hyperellipsoid	$\sum_{i=1}^{n} i^i + x_i^2$	$-5.12 \leq x_i \leq 5.12$
Levy	$\sin^2(\pi y_0) + \sum_{i=0}^{n-2}(y_i - 1)^2(1 + 10\sin^2(\pi y_i + 1)) + (y_{n-1} - 1)^2(1 + \sin^2(2\pi x_{n-1}))$	$y_i = 1 + \frac{x_i - 1}{4}$ $-10 \leq x_i \leq 10$
Maxmod	$\max(\lvert x_i \rvert)$	$-10 \leq x_i \leq 10$
Multimod	$\sum_{i=1}^{n}\lvert x_i \rvert \prod_{i=1}^{n}\lvert x_i \rvert$	$-10 \leq x_i \leq 10$
Rastrigin	$\sum_{i=1}^{n} x_i^2 - 10\cos(2\pi x_i) + 10$	$-5.12 \leq x_i \leq 5.12$
Rosenbrock	$\sum_{i=2}^{n} 100(x_i - x_{i-1}^2)^2 + (1 + x_{i-1})^2$	$-10 \leq x_i \leq 10$
Schaffer	$\sum_{1}^{n-1}((x_i^2 + x_{i+1}^2)^{1/4}\sin(50(x_i^2 + x_{i+1}^2)^{1/10}))^2 + 1$	$-100 \leq x_i \leq 100$
Schwefel	$\sum_{i=1}^{n}\{\sum_{j=1}^{j<i} x_i\}^2$	$-10 \leq x \leq 10$
Sphere	$\sum_{i=1}^{n} x_i^2$	$-10 \leq x_i \leq 10$
Sumsquares	$\sum_{i=0}^{n-1} i x_i^2$	$-10 \leq x \leq 10$

TABLE 1. Test Functions Used in Simulations

	Prob	Min	Mean	Median	Max	Std. Dev.	MAD
Ackley	0	0.0579	0.2287	0.2208	0.5719	0.0756	0.0716
	0.05	0.0038	0.0734	0.0574	0.7333	0.0689	0.0409
Bohachevsky	0	0.1754	1.7035	1.6385	4.3864	0.6745	0.6985
	0.05	0.0015	0.3747	0.1923	3.2220	0.4621	0.2227
Griewank	0	0.0003	0.0025	0.0023	0.0103	0.0013	0.0011
	0.05	0.0000	0.0004	0.0002	0.0088	0.0009	0.0002
Holzman	0	0.0000	0.0000	0.0000	0.0001	0.0000	0.0000
	0.05	0.0000	0.0000	0.0000	0.0001	0.0000	0.0000
Hyperellipsoid	0	0.0001	0.0019	0.0018	0.0058	0.0009	0.0008
	0.05	0.0000	0.0005	0.0002	0.0149	0.0011	0.0002
Levy	0	0.3372	3.1317	2.8697	9.6854	1.5662	1.4541
	0.05	0.0005	2.3189	2.0138	11.1370	2.1362	2.6094
Maxmod	0	0.0148	0.0318	0.0312	0.0686	0.0081	0.0075
	0.05	0.0356	0.1489	0.1407	0.4299	0.0552	0.0530
Multimod	0	0.0000	0.0836	0.0872	0.1987	0.0355	0.0261
	0.05	0.0000	0.0182	0.0149	0.0938	0.0142	0.0129
Rastrigin	0	0.0825	1.4130	1.2820	6.5973	1.2180	1.2539
	0.05	0.0002	1.4272	1.1186	6.4626	1.3441	1.5114
Rosenbrock	0	5.4774	8.1814	8.3045	9.6554	0.5774	0.4317
	0.05	0.9982	12.9854	8.1374	153.7519	16.4957	0.8131
Schaffer	0	3.1016	5.0875	5.0796	7.2582	0.6915	0.6448
	0.05	1.0455	6.3968	6.0946	14.5236	2.3211	2.1870
Schwefel	0	17.4613	34.0956	32.5590	66.8198	7.9051	6.4944
	0.05	0.2968	1.2817	1.1317	4.3681	0.7613	0.6316
Sphere	0	0.0010	0.0074	0.0069	0.0225	0.0035	0.0034
	0.05	0.0000	0.0019	0.0008	0.0534	0.0036	0.0009
Sumsquares	0	0.0011	0.0075	0.0069	0.0240	0.0036	0.0033
	0.05	0.0000	0.0019	0.0008	0.0367	0.0034	0.0010

TABLE 2. Simulations for p=10

	Prob	Min	Mean	Median	Max	Std.Dev.	MAD
Ackley	0	2.9002	3.7817	3.7926	4.5383	0.2622	0.2639
	0.05	0.0228	0.1600	0.1241	1.5651	0.1455	0.0673
Bohachevsky	0	40.4351	79.1801	78.1808	123.5290	13.9832	13.3394
	0.05	0.1502	2.2641	1.9431	8.0950	1.4313	1.3431
Griewank	0	0.3901	0.9723	0.9455	1.9924	0.2452	0.2299
	0.05	0.0001	0.0018	0.0014	0.0122	0.0015	0.0010
Holzman	0	0.2529	1.2502	1.1423	3.7206	0.5489	0.5072
	0.05	0.0000	0.0000	0.0000	0.0006	0.0001	0.0000
Hyperellipsoid	0	0.9571	2.3115	2.2869	4.5050	0.5560	0.5261
	0.05	0.0003	0.0050	0.0038	0.0355	0.0042	0.0024
Levy	0	66.2106	119.7350	119.2752	175.3249	16.9267	16.6394
	0.05	0.1509	8.6419	8.0528	25.3093	5.2961	5.9740
Maxmod	0	0.3165	0.4764	0.4749	0.6781	0.0583	0.0572
	0.05	0.1364	0.3041	0.2984	0.5150	0.0612	0.0623
Multimod	0	0.0000	3.2797	3.3788	5.7044	0.7397	0.4303
	0.05	0.0000	0.0726	0.0706	0.1887	0.0339	0.0288
Rastrigin	0	33.4126	64.9973	64.7518	96.3186	11.0987	11.2791
	0.05	0.0287	2.0839	1.6287	9.6301	1.9067	1.7919
Rosenbrock	0	59.9610	131.7415	129.5522	226.7899	27.1603	27.2142
	0.05	17.1686	44.8274	25.8228	196.6983	29.3024	7.3546
Schaffer	0	40.8998	52.0038	52.0877	61.9053	3.6027	3.4867
	0.05	13.3959	23.8951	23.3254	38.2602	4.9126	5.0899
Schwefel	0	6044.304	8075.2456	8086.3155	10351.01	973.1779	944.1812
	0.05	152.0629	262.5874	255.6143	567.9562	66.9413	54.6873
Sphere	0	3.5367	8.9277	8.8302	16.5280	2.0252	2.0143
	0.05	0.0009	0.0187	0.0147	0.1891	0.0152	0.0099
Sumsquares	0	3.4760	8.9354	8.8027	16.1854	2.0651	1.9490
	0.05	0.0014	0.0189	0.0150	0.1199	0.0141	0.0097

TABLE 3. Simulations for p=25

	Prob	Min	Mean	Median	Max	Std.Dev.	MAD
Ackley	0	5.5889	6.8039	6.8284	7.6777	0.3368	0.3520
	0.05	0.8090	1.6895	1.7099	2.5268	0.3158	0.3152
Bohachevsky	0	540.0593	915.1087	913.2032	1280.9830	139.5234	147.5285
	0.05	26.4656	41.9644	41.9394	67.8257	5.3180	5.1912
Griewank	0	5.8961	11.4845	11.4045	16.1720	1.7567	1.7140
	0.05	0.0605	0.1808	0.1718	0.4146	0.0589	0.0508
Holzman	0	73.7300	224.1600	218.9521	474.2187	67.5767	69.4216
	0.05	0.0130	0.0778	0.0641	0.5580	0.0533	0.0383
Hyperellipsoid	0	28.1804	65.5965	65.8457	97.4032	9.8273	9.5392
	0.05	0.3328	0.8536	0.8137	1.6776	0.2595	0.2346
Levy	0	283.5280	387.2117	388.8155	458.0209	29.3325	31.2710
	0.05	73.6633	128.5590	128.5692	193.9864	22.1059	21.9551
Maxmod	0	0.9268	1.3112	1.3093	1.6250	0.1085	0.1129
	0.05	0.2247	0.3593	0.3583	0.5580	0.0525	0.0465
Multimod	0	0.0000	17.8652	18.4618	22.7881	3.4284	1.5021
	0.05	0.0000	1.6498	1.7046	2.6037	0.4293	0.2851
Rastrigin	0	188.8021	252.2721	252.6259	301.4020	20.9102	21.4464
	0.05	16.5305	36.4057	35.9026	71.6754	9.1899	9.0363
Rosenbrock	0	1088.8180	2245.9238	2212.1760	4064.0820	505.4592	494.2128
	0.05	59.4509	97.8905	90.4685	199.1299	26.6319	26.6519
Schaffer	0	151.8794	173.0054	173.1660	196.5205	7.7092	7.6203
	0.05	46.5284	66.3682	65.7527	104.6872	7.3255	6.6193
Schwefel	0	124804.6	177800.0845	179921	202647.9	14372.2681	12031.4473
	0.05	14228.99	21601.6323	21356.92	31799.56	2811.7234	2864.6501
Sphere	0	138.6793	246.6506	246.2349	359.1078	38.1239	39.6904
	0.05	1.0224	3.1682	3.0029	7.8608	0.9843	0.8992
Sumsquares	0	159.9675	247.3759	246.1421	341.7786	35.5817	37.8696
	0.05	1.0709	3.2555	3.1176	7.2591	0.9795	0.9014

TABLE 4. Simulations for p=50

corresponds a random mutation. In this paper, we develop a new genetic operator in floating-point genetic algorithms. The operator searches for a convenient time series model that fits the data of genes which are collected from the family tree of chromosomes. ARIMA modelling and forecasting are widely used in the liteature. ARIMA models are also convenient to forecast future values of chromosomes as they include the lagged terms of the variable of interest and the lagged terms of error term. Since it is expected to get a better population in each generation in GAs by means of average fitness, genes in a family tree of chromosomes may not be stationary in time domain. The I part of ARIMA models are also capable to cope with this issue. This operator generates new offspring which will possibly be generated by the usual GA operators such as crossover and mutation. As a result of this, the operator is a *short-cut hill-climber* which mimics a local search inner optimizer and inheritance in nature. A software package is developed as an implementation of the algorithm and it is freely available for downloading and use. A simulation study is performed on a suite of well-known test functions in dimensions of $p = 10, 25$, and 50. The results of the simulation study show that the proposed algorithm significantly increases the search capabilities of GAs. The results of the simulation study also show that the GA obtains more precise results when the number of function parameters increase.

Conflict of Interests

The authors declare that there is no conflict of interests.

REFERENCES

[1] Ernesto P Adorio and U Diliman. Mvf-multivariate test functions library in c for unconstrained global optimization. `http://www.geocities.ws/eadorio/mvf.pdf`, 2005.

[2] George EP Box, Gwilym M Jenkins, and Gregory C Reinsel. *Time series analysis: forecasting and control*, volume 734. John Wiley & Sons, 2011.

[3] G.E.P. Box, G.M. Jenkins, and G.C. Reinsel. *Time Series Analysis: Forecasting and Control*. Wiley Series in Probability and Statistics. Wiley, 2008.

[4] P Laurie Davies, Roland Fried, and Ursula Gather. Robust signal extraction for on-line monitoring data. *Journal of Statistical Planning and Inference*, 122(1):65–78, 2004.

[5] Thomas E Davis and Jose C Principe. A markov chain framework for the simple genetic algorithm. *Evolutionary computation*, 1(3):269–288, 1993.

[6] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.

[7] Kalyanmoy Deb and Ram Bhushan Agrawal. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995.

[8] Kalyanmoy Deb and A Kumar. Real-coded genetic algorithms with simulated-binary crossover: Studies on multi-modal and multi-objective problems. *Complex systems*, 9(6):431–454, 1995.

[9] Dirk Eddelbuettel, Romain François, J Allaire, John Chambers, Douglas Bates, and Kevin Ushey. Rcpp: Seamless r and c++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011.

[10] Roland Fried, Karen Schettlinger, and Matthias Borowski. *robfilter: Robust Time Series Filters*, 2014. R package version 4.1.

[11] Carlo Gaetan. Subset arma model identification using genetic algorithms. *Journal of Time Series Analysis*, 21(5):559–570, 2000.

[12] Ursula Gather, Karen Schettlinger, and Roland Fried. Online signal extraction by robust linear regression. *Computational Statistics*, 21(1):33–51, 2006.

[13] Adelina Gnanlet and Chandrasekharan Rajendran. Meta-heuristics in arma forecasting. *California Journal*, 7(1):38–48, 2009.

[14] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[15] David E Goldberg. Real-coded genetic algorithms, virtual alphabets, and blocking. *Urbana*, 51:61801, 1990.

[16] Nikolaus Hansen and Stefan Kern. Evaluating the cma evolution strategy on multimodal test functions. In *Parallel problem solving from nature-PPSN VIII*, pages 282–291. Springer, 2004.

[17] Francisco Herrera, Manuel Lozano, and Jose L. Verdegay. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial intelligence review*, 12(4):265–319, 1998.

[18] John H Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[19] Rob J Hyndman and Yeasmin Khandakar. Automatic time series for forecasting: the forecast package for r. Technical report, Monash University, Department of Econometrics and Business Statistics, 2007.

[20] Rob J. Hyndman and Yeasmin Khandakar. Automatic time series forecasting: The forecast package for r. *Journal of Statistical Software*, 27(3):1–22, 7 2008.

[21] Zbigniew Michalewicz. Genetic algorithms+ data structures= evolution program. *Artificial Intelligence, Berlin: Springer, 1992*, 1, 1992.

[22] Brian K Nelson. Time series analysis using autoregressive integrated moving average (arima) models. *Academic emergency medicine*, 5(7):739–744, 1998.

[23] Allen E Nix and Michael D Vose. Modeling genetic algorithms with markov chains. *Annals of mathematics and artificial intelligence*, 5(1):79–88, 1992.

[24] Tatsuya Nomura and Tsutomu Miyoshi. Numerical coding and unfair average crossover in ga for fuzzy rule extraction in dynamic environments. In *Fuzzy Logic, Neural Networks, and Evolutionary Computation*, pages 55–72. Springer, 1996.

[25] Chorng-Shyong Ong, Jih-Jeng Huang, and Gwo-Hshiung Tzeng. Model identification of {ARIMA} family using genetic algorithms. *Applied Mathematics and Computation*, 164(3):885 – 912, 2005.

[26] Sankar K Pal and Paul P Wang. *Genetic algorithms for pattern recognition*. CRC press, 1996.

[27] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014.

[28] Nicholas J Radcliffe. Equivalence class analysis of genetic algorithms. *Complex systems*, 5(2):183–205, 1991.

[29] Shahryar Rahnamayan, Hamid R Tizhoosh, and Magdy MA Salama. Opposition-based differential evolution. *Evolutionary Computation, IEEE Transactions on*, 12(1):64–79, 2008.

[30] Mehmet Hakan Satman. Machine coded genetic algorithms for real parameter optimization problems. *Gazi University Journal of Science*, 26(1):85–95, 2013.

[31] Mehmet Hakan Satman. Hybridization of floating-point genetic algorithms using hooke-jeeves algorithm as an intelligent mutation operator. *Journal of Mathematical and Computational Science*, 5(3):320–332, 2015.

[32] Mehmet Hakan Satman. forega: Floating-point genetic algorithms with statistical forecast based inheritance operator. `http://CRAN.R-project.org/package=forega`, 2016. R package version 1.0.

[33] M Srinivas and Lalit M Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 24(4):656–667, 1994.

[34] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[35] Rasmus K Ursem. Diversity-guided evolutionary algorithms. In *Parallel Problem Solving from NaturePPSN VII*, pages 462–471. Springer, 2002.

[36] Jakob Vesterstrom and Rene Thomsen. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 1980–1987. IEEE, 2004.

[37] Rob J Hyndman with contributions from George Athanasopoulos, Slava Razbash, Drew Schmidt, Zhenyu Zhou, Yousaf Khan, Christoph Bergmeir, and Earo Wang. *forecast: Forecasting functions for time series and linear models*, 2014. R package version 5.6.

[38] Alden H Wright et al. Genetic algorithms for real parameter qptimization. In *FOGA*, pages 205–218. Citeseer, 1990.