Time series forecasting method based on genetic algorithm for predicting the conditions of technical systems

V M Kureychick¹, T G Kaplunov¹,²

¹ Computer-Aided Design Chair, Computer Technologies and Information Security Institute, Southern Federal University, 44, Nekrasovsky lane, Taganrog, 347922, Russia
² tkaplunov@sfedu.ru

Abstract. The pragmatic aim of this paper is to provide a genetic algorithm for predicting the technical systems state. The research novelty is to represent the genuine approach to forecast the technical systems states. The given approach implies finding future values by extrapolating current observation results. Forecast can be considered as a diagnostic control at zero-time extrapolation, or as a general case of diagnosis. The developed genetic algorithm is based on the classical representation of genetic algorithms with the changes required for forecasting. So, a function that validates alternative solutions outlaying from the geometric representation of the average values of the time series plot is exploited as a fitness function. The method based on the use of Shewhart process-behavior charts is also applied to exclude failures of the sensor collecting measured data and to control the mutation. The algorithm performs a prediction for one time interval ahead for processes that are not affected by external factors or processes, or the influence of external factors on which is not significant within one time interval. Our experiment confirmed the efficiency of the suggested algorithm. It resulted in obtaining a predictive solution.

1. Introduction

The pragmatic purpose of forecasting changes in technical system properties is inextricably linked to the purpose of diagnosing this system. The main difference lies in the fact that the state is assessed for a future point in time. The suggested approach involves finding future states by extrapolating the current results of monitoring the system state at a future point in time. The accuracy of the forecast is determined by the amount of information about the mechanism of the processes occurring in the system being diagnosed, the presence of sound mathematical models of changes in range of parameters, and the ability to take into account a variety of factors. Its reliability depends on the individual nature of changes in device parameters associated with the initial variation of component parameters, assembly errors and other factors.

Therefore, the purpose of forecasting the quality of technical systems operation is reduced to the problem of extrapolating a random vector process on the results of its observation at a certain time interval. At the same time, it is necessary to take into consideration that the forecast predicts the future state not exactly, yet with some probability or predicts the area in which the future value of the process hits.

The given paper presents a forecasting method based on genetic algorithm (GA), which has high accuracy in predicting for short periods of time. The observed problem is relevant today, as modern
technical systems are characterized by hundreds of processes that significantly affect their performance. The various combinations of indicators of these processes can lead to diverse situations, including abnormal ones. Obviously, the operator is not able to track all processes simultaneously. That’s why, corresponding algorithms are used for such cases, as they are able to forecast various ways of a particular process development, and warn the operator about a possible abnormal situation in the future.

Algorithms based on neural networks, regression models, Markov chains, etc. are applied for these purposes. The paper proposes a new approach based on the genetic algorithm methods.

2. Research Problem
Forecasting the technical system state can be viewed as a diagnostic control at zero-time extrapolation, and, hence, as a more general case of diagnosis.

If we are given a certain technical system, characterized by a number of states, then it is necessary to find out its next state at the nearest point in time. If the system state indicators can be formalized, then the most convenient way to present them in time is the time series (TS). Then the accepted formulation of the TS forecasting problem looks like the following. Let the time series values be available at some, discrete points in time \( t = 1, 2, ..., T \). Thus, the time series will be defined as \( Z(t) = Z(1), Z(2), ..., Z(T) \).

It means that at \( T \) point in time it is essential to determine the \( Z(t) \) process value at \( T + 1, T + 2, ..., T + P \) points in time, where \( T \) is the forecast moment, \( P \) is the lead time [1].

To obtain the forecast, it is necessary to establish a dependence reflecting the connection between past and future time series values:

\[
Z(t) = F(Z(t - 1), Z(t - 2), Z(t - 3), ...) + \varepsilon_t.
\]

The given dependence turns is a forecast model. It is necessary to develop such a forecast model for which the average absolute deviation of the ideal value from the predicted one tends to a minimum for any given \( P \):

\[
\bar{E} = \frac{1}{P}\sum_{t=T+1}^{T+P}|\varepsilon_t| \rightarrow \min.
\]

Further, expression (1) could be presented as:

\[
\hat{Z}(t) = F(Z(t - 1), Z(t - 2), Z(t - 3), ...)
\]

where \( \hat{Z}(t) \) are \( Z(t) \) time series forecasted values.

When describing the operating features of the genetic algorithm, it is essential to represent \( Z(t) \) values as binary vectors, where \( t = 1, 2, ..., T \) (points in times at which the time series are available). It is an initial population. The forecast accuracy directly depends on its size. Expression is the applicability function for algorithm operation. Forecast will be done one step ahead, i.e. for \( Z(T + 1) \).

3. Algorithm Description
In the case of a prediction for \( Z(T + 1) \), the algorithm performs 100 iterations, although it finds the forecasted value approximately within 20-30 iterations. Forecasting is performed for processes developing without the influence of external factors, or processes where external factors do not have a significant impact on them in a short period of time. Statistical data, namely, the value of the time series for the twenty preceding time intervals, which the algorithm receives from the Excel file, are exploited as the initial population (Figure 1):

The presented algorithm [2] forms the initial population of alternative solutions emanated from the statistical data on the previous values of the system state. Then, the population is sorted out according to the degree of compliance to the fitness function. It should be noted that GA is a classical genetic algorithm [1, 5] with a few changes necessary to solve forecasting problems.

The built-in rejection function of alternative solutions provides algorithm with optimal properties, since the search for a solution occurs only along the trend line. The same function makes the algorithm be stable to unforeseen circumstances, since all alternative solutions obtained as a result of a failure will be rejected and not included into the new population.
The selection of chromosomes pairs for crossingover is depicted below. Two random alternative solutions are taken as parents to produce an offspring. Then, the next pair is taken, and so on, until the new population has the same alternative solutions as the previous one. Since the population is sorted out before crossingover, there will be a tendency in a new population to increase the fitness degree from one alternative solution to another after crossing. The fittest individuals will be located at the end of the population. An important feature of this method is that each member of the first generation will be able to extend its genetic material to the latter.

The described function works as follows. Two alternative solutions are selected from the population, and then they are presented in binary code. The next step is to assemble a new solution, where the first half of the chromosomes is inherited from the first parent, and the second from the second one. The crossing point, in which both parents are divided into two halves, is chosen randomly. These steps are carried out until the number of new alternative solutions becomes equal to the number of alternative solutions in the parent population. Moreover, each of the parents may participate in crossing several times, while some may not participate at all.

To obtain forecasted values to satisfy the solution of the problem, at this stage it is necessary to preserve the number of individuals in the population, so two parents must have two offspring.

Let us consider an example to illustrate the abovementioned. Let two parental chromosomes P1 and P2 be selected for the implementation of crossingover based on the described algorithm. It is necessary to perform a crossingover iteration to obtain two chromosomes of P3 offspring.

\[
P_1: 0101|1100 \\
P_2: 0001|1010 \\
P_3: 01011010
\]

The crossingover algorithm written in Python looks like:

```python
def OKFunc(buf):
    newDesig = []
    for i in range(n):
```

![Figure 1. Genetic algorithm flow chart.](image-url)
NumberOne = random.randrange(0, n, 1)
NumberTwo = random.randrange(0, n, 1)
firstGen = toBin(buf[NumberOne])
secondGen = toBin(buf[NumberTwo])
Point = random.randrange(0, len(firstGen), 1)
newGen = ''
for i in range(len(firstGen)):
    if i <= Point:
        newGen += firstGen[i]
    else:
        newGen += secondGen[i]
while len(newGen) < 4:
    newGen = '0' + newGen
newDesig.append(int(newGen, 2))
return (newDesig)

The next step is a mutation. The mutation point is selected randomly, and the chromosome corresponding to this point is inverted:

Let us consider an example. Let the alternative solution P3 from the previous example be chosen for the mutation, the mutation point corresponds to the sixth gene of the alternative solution chosen, then

\[
P_3: 01011010
\]

\[
P_{\text{mut}}: 01011110
\]

The mutation algorithm implemented in Python looks as follows:

forMut = newDesig
for i in range(len(forMut)):
    gen = toBin(forMut[i])
    mutagen = ''
    while len(gen) != 8:
        gen = '0' + gen
    PointMut = random.randrange(0, len(gen), 1)
    if gen[PointMut] == '1':
        for i in range(len(gen)):
            if i != PointMut: mutagen += gen[i]
            else: mutagen += '0'
    else:
        for i in range(len(gen)):
            if i != PointMut: mutagen += gen[i]
            else: mutagen += '1'
    forMut[i] = int(mutagen, 2)
return (forMut)

Finally, a new population is formed using the selection function. This step involves taking two populations – old and new, after mutation. The new population undergoes a screening procedure for individuals that significantly differ from the others, i.e. they demonstrate inadequate values being compared with the rest. Such values can disrupt the forecast process, as they are either a sensor failure, which controls the values of the process, or a consequence of a mutation that has moved away from the trend line (the geometric display of the averaged values of the studied parameter) of the main time series plot. Shewhart control charts [3, 14] are exploited to determine if screening is in a state of control. The formula for calculating the control limits is as follows:
\[ \bar{X} = A_2 \bar{R} \]

where \( \bar{X} \) – the average value of the average values for the subgroup, \( \bar{R} \) – average range, \( A_2 \) – certain engineering coefficient depending on the size of the subgroup. All coefficients and formulas are described in Russian National Standard (GOST) 50779.42-99. The screening algorithm is presented below:

```python
def ShuhartsMap(Young, Old):
    newArr = []
    R = max(Old) - min(Old)
    Xs = sum(Old) // n
    Xmax = Xs + 0.180 * R  # meaning 0.180 is taken from Russian National Standard (GOST) R ISO 7870-
    Xmin = Xs - 0.180 * R  # for selection containing 20 parameters
    for i in range(n):
        if Young[i] < Xmin or Young[i] > Xmax:
            newArr.append(Old[i])
        else:
            newArr.append(Young[i])
    return newArr
```

Further, the selection function compares alternative solutions from both populations and their distance from the trend line [4]. The individual whose location is less distant than the trend line gets into a new population. The selection function written in Python looks like that:

```python
NewDesig = []
Old = sorted(Parents)
Young = sorted(NewDecisionMut)
for i in range(n):
    sr = sum(Old) // n
    if sr - 1 < Young[i] and Young[i] < sr + 1:
        NewDesig.append(Young[i])
    else: NewDesig.append(Old[i])
return NewDesig
```

The described functions are repeated within 100 iterations, although the conduction of 30th iteration allows concluding on the uniformity of all decisions in new generations. Besides, these solutions are the desired forecast for the time series. The population obtained at the 100th iteration is served as a basis for calculating the average value, which is the forecast of the time series for one time interval ahead (Figure 1).

4. Experimental Data
The experimental algorithm testing was carried out using temperature records in Taganrog in August 2000. In case of integer values of the time series, the algorithm predicts the exact value in 60% of the conducted tests; otherwise the deviation from the actual values is equal to 1 degree Celsius. In the case of actual values of the time series, the exact value is found in approximately 30%. The margin of error lies in the range between 0.01 and 0.7 degrees Celsius.
5. Conclusion
A genetic algorithm is a heuristic search algorithm employing mechanisms similar to natural selection. The paper analyzes a variant of the algorithm applicable for predicting the technical systems state. The considered algorithm for generating the first generation uses statistical data on observations of previous states of the system to predict its state.

The developed genetic algorithm is based on the classical representation of genetic algorithms with the alterations necessary for forecast. So, a function that validates alternative solutions outlaying from the geometric representation of the average values of the time series plot is applied as a fitness function. The method based on the use of Shewhart control charts is also exploited to exclude failures of the sensor collecting measured data and to control the mutation.

The developed algorithm is characterized with a high degree of stability and optimality, thanks to the protection from data obtained by the system as a result of equipment failures. In addition, the algorithm is distinguished with low production costs and a low degree of complexity when being implemented. Software is written in Python, using third-party libraries to work with data.

The algorithm effectiveness was confirmed by an experiment. It resulted in obtaining a predictive solution. The general direction of the process development was verified in practice, but it had a deviation from the actual values. The forecast error ranged from 0.01 to 0.7 compared to actual values obtained from the sensors.

It allows getting an accurate and precise forecast in a reasonable time.

Acknowledgments
This work is partially supported by Russian Foundation for Basic Research (Grant №18-07-00050 & Grant №18-2922019/18).

References
[1] Gladkov L A, Kureychik V V, Kureychik V M 2010 Genetic algorithms: a tutorial (Moscow: Fizmatlit)
[2] Kureychik V M, Kaplunov T G 2018 Adaptive genetic algorithm based on fuzzy rules Izvestiya of the Southern Federal University Engineering Science 5 26-32
[3] Sinyuk V G, Akopov V N 2008 Controlled statistical genetic algorithms Software products and systems 4 46-49
[4] Box J, Jenkins G M 1974 Time series analysis, forecast and management (Moscow: Mir)
[5] Holland J H 2002 *Emergence: From Chaos to Order* (Canada: Helix book)
[6] Rutkowski D, Pilinsky M, Rutkovsky L 2008 *Neural networks, genetic algorithms and fuzzy systems* (Moscow: Hotline-Telecom)
[7] Baeck T, Fogel D, Michalewicz Z 2000 *Evolutionary computation. Basic algorithms and operators* (Bristol & Philadelphia: IOP publishing LTD)
[8] Vaesk T, Fogel D, Michalewicz Z 2000 *Evolutionary computation. Advanced algorithms and operators* (Bristol & Philadelphia: IOP publishing LTD)
[9] Yarushkina N G, Afanas'eva T V, Perfil'eva I G 2012 *Intelligent time series analysis: textbook* (Moscow: ID. "FORUM": INFRA-M)
[10] Wheeler D, Chambers D 2009 *Statistical process management: business Optimization using Shewhart control maps = Understanding Statistical Process Control* (Moscow: Alpina Publisher)
[11] Chucheva I A 2010 Model of extrapolation of time series for the sample of maximum similarity *Information technology* **12** 43-47
[12] Morariu N, Iancu E, Vlad S 2009 A neural network model for time series forecasting *Romanian Journal of Economic Forecasting* **4** 213-223
[13] Jingfei Yang M Sc 2006 *Power system Short-term Load Forecasting* Thesis for Ph.d degree (Germany, Darmstadt, Elektrotechnik und Informationstechnik der Technischen Universitat)
[14] Interstate Council for standardization 2000 Statistical technology. Shewhart control cards GOST 50779.42-99
[15] Interstate Council for standardization 2016 Statistical methods. Control card. Part 2. Shewhart control cards GOST R ISO 7870-2