Multi-Agent System for Reconstruction Constructive Models of Stochastic Fractal Time Series

Viktor Shynkarenko[1,*,†], Artem Zhadan[1,*,†], Oleksandr Halushka[1,†]

[1] *Ukrainian State University of Science and Technologies, 2 Lazaryana str., Dnipro, 49010, Ukraine*

Abstract

In previous works, the theoretical tools for constructive-synthesizing modeling of deterministic and stochastic fractal time series were developed, as well as the constructive model's recovery process by means of genetic algorithm. To evaluate the effectiveness of these tools, a single-threaded software solution has been developed for reconstructing constructive models of given time series. Despite the positive results obtained, problems were identified with the time characteristics of the restoration process, especially in series with a complex generative model. In this study, agents-oriented programming tools are used to increase the time efficiency of the computing process, without which it is almost impossible to restore stochastic series. Experimental studies on the reconstruction of model fractal time series with evaluation by the set of indicators of time efficiency have been conducted. The indicators of timing characteristics of both single-threaded and agent-based software solutions are determined. One of the inventions in the work is the proposed method of comparing model and reconstruction stochastic time series based on several reconstructed and specified time series.

Keywords

L-system, constrictive-synthesizing modeling, fractal, time series, genetic algorithm, multi-agent system, formal grammars, software, information technologies

1. Introduction

A time series is one of the most common ways of representing the set of states of a complex system over a period. Thanks to this presentation of data, it is possible to predict future states of the system [1].

Nowadays, there are many methods for the prediction of time series' future values: moving median filter [2], using neural networks [3, 4, 5], Stochastic differential equations [6], dynamic Bayesian predictive synthesis [7], etc.

Another possible approach is the use of fractal properties of the time series [8, 9]. By determining the rules of self-similarity, it is possible to trace the regularities based on which it is possible to predict the future values.

1.1. Using constructive-synthesizing modeling with time series

The fractal time series' modeling has been already resolved by the means of the constructive-synthesizing approach [10, 11], which is a development of the formal grammars` theory [12]. The process is based on a defined constructive model [13] with the usage of L-system [14].

The constructive model consists of two main parts: the generating L-system (axioms, replacement rules, and alphabet of terminal and non-terminal symbols) and mathematical parameters (initial mathematical expectation and dispersion, values of their transformations). Each terminal symbol of the system had a certain logical purpose - changing the initial values by their growth or throwing out

Information Technology and Implementation (IT&I-2024), November 20-21, 2024, Kyiv, Ukraine

* Corresponding author.

† These authors contributed equally.

shinkarenko_vi@ua.fm (V. Shynkarenko); gadan998@gmail.com (A. Zhadan); galushka.alex.1998@gmail.com (O. Halushka)

0000-0001-8738-7225 (V. Shynkarenko); 0009-0003-1133-1630 (A. Zhadan); 0009-0005-3447-2676 (O. Halushka)

a point according to the law of normal distribution and current values of mathematical expectation and dispersion.

To generate a deterministic series, generative models with zero variance and its growth were used (Figure 1). In the opposite case, the output series has a stochastic structure (Figure 2), and one model can generate an infinite number of different time series. Based on the fractal nature of the L-system, the generated time series also has self-similarity properties, which can be clearly seen in the example of a deterministic series.

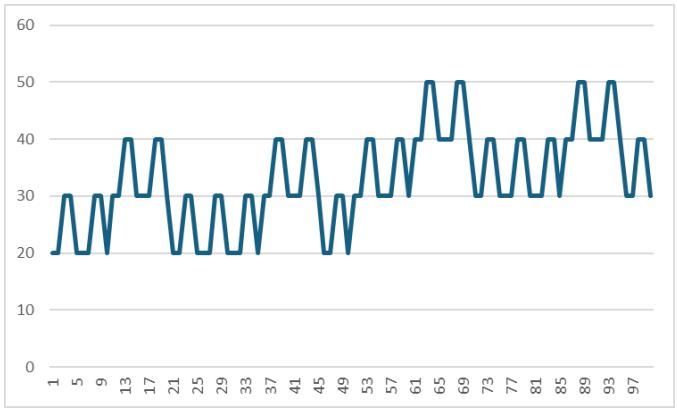

Figure 1. System high-level architecture. Rule:
$f \to ff + ff - f; V_f = 20; dV_f = 10$

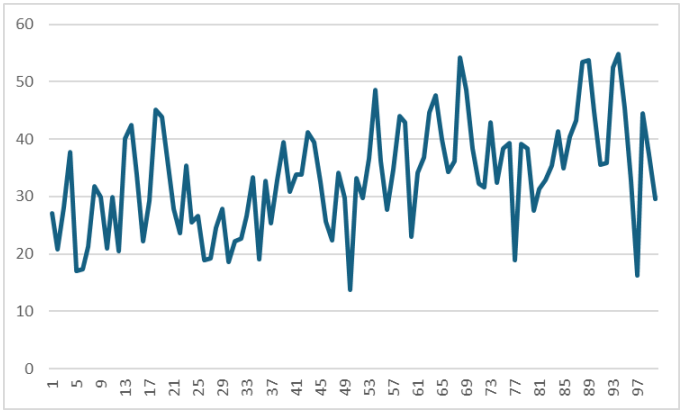

Figure 2. System high-level architecture. Rule:
$f \to ff + ff - f; V_f = 20; dV_f = 10; Df = 20$

1.2 Series model reconstruction process and its automation

During further research, it was decided to consider the possibility of implementing the reverse process [15]. The recovery process and its main stages were specified, based on a genetic algorithm with crossover, mutation, and selection operations. To solve the problem, the necessary modifications were introduced, largely dictated by the structure of the chromosome, which repeated the structure of the constructive model. A fitness function was also described, with the help of which the proximity indicator of the series obtained based on the chromosome with the original series was calculated.

To verify this approach, a software solution was created that automates the process of the constructive model reconstruction of a deterministic series. When verifying the approach, test model series were used, generated by constructive models of different structures and complexity (with 1, 2, or 3 substitution rules of different lengths in the L-system).

For experiments' automation, a single-threaded prototype was developed that implemented the process of reconstruction for the input series. Two stages of experiments were conducted on model deterministic time series generated with a fixed right-hand side length of four symbols and a variable from four to ten. In all experiments, all replacement rules were found for which the model and simulated time series were identical. In the first experiment, for 100 tests, the average number of

generations of the genetic algorithm, at which the result was achieved, was 37.6, and the maximum was 512. In the second, for 100 tests - 2030 and 40518 [15].

The disadvantages of the specified approach were its time efficiency and the possibility of effective work only with deterministic time series. The single-threaded approach allowed us to speed up the process of developing a software prototype and bring the start of experimental research closer. For the restoration of series with a simple constructive structure, it fully justified itself, but for further studies on more complex stochastic time series, a revision of the approach to implementation was required. It was necessary to consider two requirements: time efficiency and ease of scaling.

1.3 Working with Stochastic Series

In addition, with the introduction of the possibility of scaling the computing capabilities, work was started on adapting the process for restoring the constructive models of fractal stochastic series. The main challenge and scientific novelty was the creation of an adequate method for comparing values to obtain the proximity indicator of the model and simulated series. It was decided to use not one series of each type in the calculation, but a certain set, based on the comparison of which the fitness indicator of the model was found.

2. Goals and objectives

According to the raised hypothesis about the presence of the fractal properties of some time series or their components, modeling is performed and based on constructive-production approach with the capabilities of stochastic L-systems.

Developed models designed to solve problems of analysis and forecasting of time series. Main tasks of this study are the improvement the time efficiency and quality of the reconstruction process by distributing calculations to several responsible entities (agents) and developing the approach with implementation of the necessary modifications to provide the possibility of working with stochastic time series. Additionally, the agent-based implementation approach will simplify the process of scaling computing power by adding additional agents if necessary.

Achieving targeted goals will introduce the basis for further work with real time series and unify the implementation of the recovery process, making it more applicable and suitable for modeling the diversity of real stochastic series.

3. Process adaptation to multi-agent approach

There are many ways to organize the topology of the agent system. During the research, the “master-slave” and “island” approaches were considered [16].

The main feature of the “master-slave” model is the delegation of individual stages of calculations to independent atomic agents (slaves), which in turn are operated by one or more main agents (masters). A master agent controls the lifecycle of dependent agents and provides them with data to operate. This model was rejected, since the algorithm itself did not have sufficiently complex tasks, the allocation of which to a separate working agent would give a gain in processing time.

In turn, when building the topology of the agents’ system through the “island” approach, each agent fully implements the reconstruction process, that is, it is completely self-sufficient. This principle is scaling by increasing processing entities and setting connections between them. Also, the ability to integrate an asynchronous approach to sending messages should be included among the advantages.

After weighing all the weaknesses and strengths of both topologies, the “island” approach was chosen. Implementation of this approach will allow each agent to be represented as an autonomous unit of computation, thereby parallelizing the computation process. Also, such architecture will allow the use of an asynchronous communication methods through a message bus or broker as a mechanism for exchanging data according to the principle “fire-and-forget” [17].

Due to the distribution of the process between entities, it became possible to introduce the concept of migration – the exchange of the best representatives of generations between the agents in the process of work. The following approaches were defined for implementation: peer-to-peer and migration through the auxiliary chromosome bank.

Peer-to-peer migration of chromosomes between gene pools is the exchange of chromosomes according to a clearly defined scheme or freely between agents. The main disadvantage of this approach is the acceleration of the dynamics of degeneration, which negatively affects their diversity and increase the time efficiency of the reconstruction process.

Migration through the auxiliary chromosome bank is a modified method of exchange between gene pools, based on the principle of delegating the migration process to a separate component of the system. When using this method, the process can include a variety of intelligent tools, such as fuzzy logic or neural networks. Using them, it is possible to optimize the process of chromosome migration and the greater efficiency of the distributed genetic algorithm.

Based on the concept of agent-oriented programming, each agent has its own goals, motivation, and set of knowledge to achieve these goals. The primary objective of the worker agent is to construct a fractal model of a time series that accurately reproduces the input data. The knowledge utilized by the agent in its work includes basic settings for the hyperparameters of the genetic algorithm, which influence various aspects of time series construction. Different configurations and their adjustments during computations for each agent lead to the intelligent behavior of the entire system as a whole.

Each worker agent, based on its basic configuration, produces results of varying quality, which can range from highly successful to entirely unusable. It is important to note that the process of the genetic algorithm working to form a more successful fractal model of a time series is not linear by nature. In the initial iterations, the algorithm can operate more freely, crossing and mutating genes with significant changes. However, when the current model is relatively close to the characteristics of the input series, changes in the genes cannot be too drastic; otherwise, the optimal result might ultimately be lost. At the same time, a classic problem of genetic algorithms frequently arises—the convergence of the algorithm at a local minimum. To combat this issue, it is crucial to maintain as much variability in the configurations of the worker agents as possible. This approach allows for the generation of a gene pool with a high degree of variability, which, on one hand, enables escaping local minima and, on the other, helps preserve the achieved results.

Isolated gene pools or "islands" would be useless without a mechanism for gene exchange. Migration is not just the exchange of chromosomes; it also involves the formation of agents' perceptions of each other. The interaction of agents in the context of chromosome exchange allows each agent to assess the quality of the chromosomes it receives, thereby building knowledge about other agents. This opens up the possibility of developing a strategy for the intelligent behavior of the system, based on the characteristics and authority of the agents, which can enhance the efficiency of problem-solving.

4. Practical implementation of the multi-agent approach

At the design stage, the topology and main working agents were determined, which are directly responsible for the implementation of the work of the genetic algorithm and its warehouses. The method of communication between agents was also defined as the asynchronous messaging.

Also, to ensure the full operation of the system, two more entities were added: the cluster agent and the user agent. The responsibilities of the first are the accounting of available agents in the system, initiation of commands of the calculation process, configuration of calculation agents. The second is receiving and validating user commands with further delegation to the cluster agent.

In general, the implemented system consists of four types of agents, each of which has its own specialization and sphere of knowledge (Table 1).

The corresponding types of agents have certain relationships with each other based on their area of responsibility (Table 2). The system can have several worker agents, but only one of any other type.

Table 1
Agent's description

Agent type	Responsibilities	Sphere of knowledge
Worker agent	Execute the reconstruction process to find the constructive model of the time series	Configuration, task specification
Migration agent	Manages the chromosomes' migration process between the gene pool by selecting the necessary sets based on defined rules	The gene pool buffer, the current state of the calculations for each agent
Cluster agent	Keeps records of available agents in the system, initiates commands of the computing process, worker agents	Information about agents, configurations of worker agents, task specification
User agent	Receiving and validating user commands with subsequent delegation to the cluster agent	Information about the user, information about the cluster agent

Table 2
Agents' relationships

Relationship	Description
User agent cluster agent	Receiving the start command to performing calculations and time series data, the model of which needs to be set. The cluster agent provides the user agent with the resulting solution after the computations are finished
Cluster agent worker agent	Local implementation of the "master-slave". A cluster agent initiates the operation of all worker agents by providing them with the required set of inputs. As soon as the worker agent finds a solution with the specified accuracy, it sends a message to the cluster agent, which, after receiving it, stops the work of all worker agents
Cluster agent migration agent	Initiation and completion of the calculation process. The cluster agent provides the coressponding commands to the migration agent

The asynchronous message approach was chosen for the interaction of agents. Rabbit Message Queue (RabbitMQ) [18] technology was chosen as a tool for the approach implementation (Figure 3).

The final structure of the developed software application includes the following components:

- Agents – each of described types with the defined quantity;
- RabbitMQ message bus as the communication tool;
- user application for the interaction with the environment.

Figure 4 presents UML sequence diagram with a description of the interaction between system agents taking into account the indicated relationships described in Tab. 2.

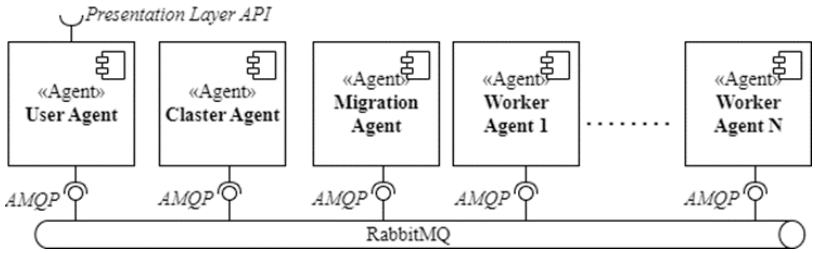

Figure 3. System high-level architecture

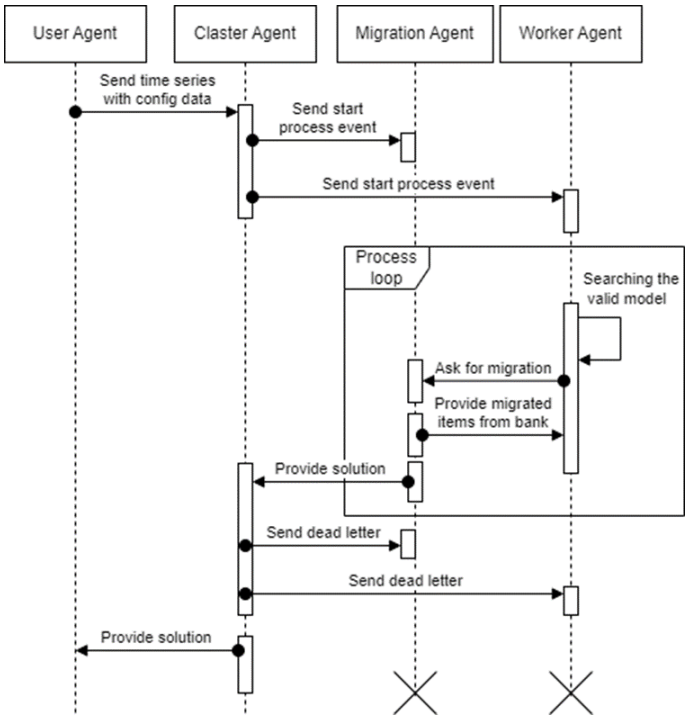

Figure 4. Components' communication sequence

5. Deterministic time series reconstruction in multi-agent environment

The process of constructing a time series consists of two parts, described by constructor [14]:
C_{MS} – constructs a fractal multi-character chain according to a given rule;
C_{TS} – converts a multi-character chain to a fractal time series.

On the basis of the axiom f, the set of replacement rules P and the alphabet of terminals $T = \{f, v, +, -\}$ and non-terminal $N = \{\sigma\}$ the iteration of the L-system is forming in the form of a finite chain of symbols. Iterating stops as soon as the number of corresponding terminal symbols f in the chain becomes greater than the specified parameter n.

The mathematical values of the model are determined, namely the current value of the point of the series V_f, its increment value $\mathrm{d}V_f$, time series period t and its step $\mathrm{d}t$.

For each of the terminal symbols T the corresponding operations are defined [14]:

$$\begin{aligned}
&\{\langle\langle \sigma \to f\sigma,\ \sigma \to v\sigma\rangle, \langle =: (v(t), \mathrm{t}V_z), +(t, t, \mathrm{d}t)\rangle\rangle,\\
&\langle\langle \sigma \to +\sigma\rangle, \langle +(\mathrm{t}V_z, \mathrm{t}V_z, \mathrm{d}V_z)\rangle\rangle,\\
&\langle\langle \sigma \to -\sigma\rangle, \langle -(\mathrm{t}V_z, \mathrm{t}V_z, \mathrm{d}V_z)\rangle\rangle,\\
&\langle\langle \sigma \to \varepsilon\rangle, \langle \varepsilon\rangle\rangle\},
\end{aligned} \tag{1}$$

where $v(t)$ – forming time series, $\mathrm{t}V_f$ – current time series' point value. The rule $\langle \sigma \to \varepsilon, \langle \varepsilon \rangle\rangle$ is use if the other ones are not applicable. The usage of the empty rule is the condition of the construction process ends.

Also, the set of operations over attributes have been developed:
$+(c, a, b)$, $-(c, a, b)$ – addition, subtraction a and b with c as the result;
$=: (c, a)$ – set value a to c.

According to the process described above, with the help of constrictive-synthesizing modeling, it is possible to determine and construct a deterministic time series (Figure 3). This family of series can

describe the behavior of technical systems (mechanical, electrical, etc.) during steady-state process mode.

In previous studies, experiments were already conducted for series with one and several rules with a fixed and floating number of symbols on the right side. The method itself was successfully used and its shortcomings were specified [14].

In current scope the main task was to determine the dependence of calculation time on the number of working agents performing calculations. The initial dependency function f_{dep} could be described in the next form:

$$\begin{cases} \bar{N}_{iter} = f_1(N_P, \bar{N}_{T_P}, N_{agent}) \\ \bar{t}_{iter} = f_2(N_P, \bar{N}_{T_P}, N_{agent}) \end{cases} \quad (2)$$

where N_P – substitution rules' count in L-system, $\bar{N}_{T_P}$ – average terminals` count in right part of the rule, N_{agent} – the amount of worker agent during the execution, $\bar{N}_{iter}$ – average iterations' count, $\bar{t}_{iter}$ – average time of the execution, f_1 and f_2 – dependency functions.

Increasing the efficiency of genetic algorithm for solving the problem of deterministic time series' reconstruction has been reached by the horizontal scaling of computing resources (parallel execution of calculations by the required number of agents) and increasing complexity and variability of the genetic algorithm for different agents.

Increasing the number of rules, as well as their length, complicates the model and allows the formation of time series with more complex structure. Some variability is applied when carrying out the mutation and crossover phases of the genetic algorithm.

Several strategies for carrying out mutation and crossover for chromosomes have been developed. One strategy is to fix the number of substitution rules and the average number of characters on the right side of the mutating chromosome rules, applying mutations only to the rules themselves. This approach allows you to inherit the fractal complexity of the structure by modifying the shape of the time series, which reduces the search area for the optimal solution. The second strategy, on the contrary, is to mutate the number of substitution rules and the average number of symbols on the right side of the rules of the mutating chromosome with a corresponding change (cutting or adding symbols on the right side of the rule). This approach expands the search area, which is useful for overcoming local extremes.

Both strategies will include the ability to mutate, and parameters.

The crossover process involves the random combinatorial recombination of genome components of two randomly selected chromosomes. The result of the crossover is a single chromosome. It was experimentally found that the best crossover result is obtained by crossing chromosomes that have a genome that is dissimilar in structure. This increases the diversity of the gene pool, which allows to find the best solutions quicker for restoring input time series and prevents premature degeneration of the population.

The problem of local convergence of genetic algorithm negatively affects time efficiency. During the process, a moment inevitably comes when the algorithm converges at a local optimum and, over a large number of epochs, cannot find the optimal solution. An iterative search for the best solution within the local optimum inevitably leads to degradation and subsequent degeneration of the entire population of chromosomes. The genome of chromosomes in the population becomes similar, the degree of variability and diversity decreases significantly. This leads to the waste of computing resources over a long period of time with an extremely low probability of further finding a chromosome with a genome structure better than the one that dominates in the population.

To solve this problem, it is necessary to change the direction of the search for the optimal solution to the many possible options represented by the best chromosomes in the population. Changing the search direction is achieved by sharply increasing the level of randomness during chromosome generation. To do this, it was proposed to introduce the concept of chromosome age and the mechanism of regeneration. Chromosome age automatically allows chromosomes to be removed from the population when they reach a certain age, which frees up space for new chromosomes with unique genes, facilitating a more efficient search for the optimal solution. The regeneration

mechanism is designed to clear the population of genetically similar chromosomes. Upon reaching a certain age and level of population contamination, this mechanism replaces most chromosomes with new ones. The results of numerous experiments confirm a significant improvement in the efficiency of the genetic algorithm.

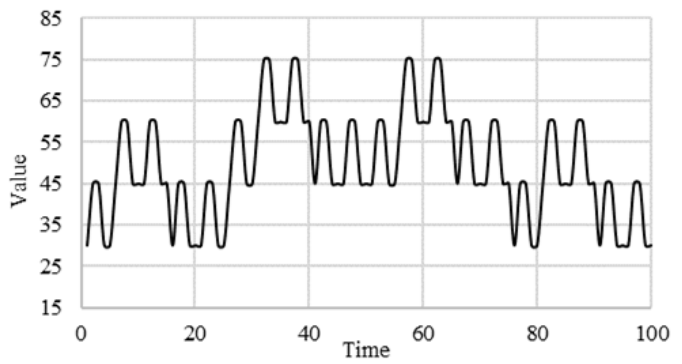

Figure 5. Reconstruction model of the deterministic time series. Rule:
$f \to f + ff - ff; V_f - 30; dV_f = 1$

The use of a multi-agent approach provides new opportunities to improve the efficiency of genetic algorithm. To expand the search area for solutions, it is necessary to use different configurations of computing agents. With this approach, each agent has its own search vector and features during the generation, mutation and crossing of chromosomes. This enriches the gene pool of all agents with various genes, which is important for the efficient operation of the algorithm. The more diverse the gene pool it has, the more efficiently the algorithm works.

The chromosome migration mechanism, organized as a chromosome bank and controlled by a migration agent, allows for various strategies for exchange between chromosomes, including both deterministic and non-deterministic approaches. The exchange of chromosomes between parallel computing agents is carried out based on an analysis of the state of the gene pool of each agent, including the degree of its "degeneration" and the dynamics of finding the best solutions.

For each possible number of rules in the L-system, experiments were conducted with 1, 2, and 3 computational agents (Table 3).

During the experiments, each computing agent used different strategies and settings when performing operations of generation, mutation and crossing of chromosomes. The results show that using different strategies for different agents leads to a significant increase in the time efficiency of the system. With an exponential increase in the complexity of the problem for a different number of substitution rules, the increase in the number of iterations of the algorithm is close to polynomial.

To evaluate the improvement of time efficiency, a comparison of the work with the previous implementation of the software application [14] based on the same time series was carried out. The comparison was made based on the process time indicator calculated for different number of substitution rules (1, 2 and 3 rules, respectively). In the table 4 shows the obtained results, which show that the multi-agent approach significantly improves the speed performance at each stage of the experiment for fractal models of different complexity.

In accordance with the comparison results (Table 4), the use of a variable approach when carrying out the main phases of the work of a genetic algorithm, multiple parallel computational agents, and organization of chromosome migration between agents allows one to increase the time efficiency of the method for restoring the fractal model of the input time series. The primary metric chosen was the average convergence rate of the genetic algorithm, measured by the number of iterations required to achieve a specified quality of time series reconstruction. This metric allows for an objective comparison of the algorithm's efficiency under different system parameter values, such as the number of agents, the average length of substitution rules, and the average number of rules in the L-system.

Additionally, temporal performance, measured in seconds required to reach the desired result, was considered. The hardware configuration was taken as a constant.

Table 3
Comparison of the systems' operation with different numbers of computing agents

Number of worker agents	Number of substitution rules	Average length of rules' right parts	Average Number of iterations	Average experiments' time, c
1	1	8	762	83.7
1	2	7.5	3954	449.3
1	3	7	31831	3789.4
2	1	8	538	57.8
2	2	7.5	2373	275
2	3	7	12632	1540.4
3	1	8	312	33.9
3	2	7.5	1287	146.2
3	3	7	6842	823.3

The time series used in the experiments were complex deterministic and stochastic sequences, constructed from manually designed fractal constructive models. The complexity of the time series, as well as the primary system parameters (number of agents, average number of rules, average rule length), was varied to assess the temporal efficiency of the system.

Table 4
Comparison of different implementations' performance indicators

Number of substitution rules	Average experiments' time for single-thread implementation	Average experiments' time for multi-agent implementation
1	1217	574
2	7324	1212
3	19380	4794

6. Defining reconstruction process for model stochastic time series

The main disadvantage of deterministic time series is the simplicity of their structure. Based on their values, the reconstruction process is trivial. The lack of variability of values allows to build a clear and unambiguous evaluation algorithm. In turn, stochastic time series have a more complex structure and variability of values. Due to this, they can be used to describe more complex natural processes (social, financial, biological, etc.).

In order to achieve the possibility of processing a model stochastic time series, the process of obtaining separate series' value was complicated by the implementation of the normal distribution operation, where the $\mathrm{t}V_f$ value is considered as mathematical expectation, and a separate parameter D_f of the constructive model is introduced for the dispersion value.

Accordingly, obtaining individual values of the time series $\nu(t)$ (1) will have the following form:

$$\begin{aligned}
&\{\left\langle\begin{array}{l}\{\langle\sigma\to f\sigma,\ \sigma\to\nu\sigma\rangle,\\ \langle=:(\nu(t),\mathrm{Norm}(\mathrm{t}V_z,\sqrt{D_f})),+(t,t,\mathrm{d}t)\rangle\end{array}\right\rangle,\\
&\langle\langle\sigma\to+\sigma\rangle,\langle+(\mathrm{t}V_z,\mathrm{t}V_z,\mathrm{d}V_z)\rangle\rangle,\\
&\langle\langle\sigma\to-\sigma\rangle,\langle-(\mathrm{t}V_z,\mathrm{t}V_z,\mathrm{d}V_z)\rangle\rangle,\\
&\langle\langle\sigma\to\varepsilon\rangle,\langle\varepsilon\rangle\rangle\},
\end{aligned} \tag{3}$$

where Norm – normal distribution function with mathematical expectation $\mathrm{t}V_f$ and dispersion D_f

Changes to the series modeling process complicate its structure (Figure 3) and, accordingly, the reconstruction process of the constructive model. Increasing the complexity of the process directly affects its time efficiency, but this side effect is leveled by the already implemented multi-agent approach to calculations.

The introduction of the additional parameter in the process led to the expansion of the chromosome structure and the addition of the dispersion parameter with its integration into the crossover and mutation processes.

When modifying the fitness function, the stochastic nature of the time series was considered. To reflect this factor, when calculating the chromosome fitness value based on the chromosome parameters, *N* time series are generated.

Due to its stochastic nature, the input series can have an infinite number of different forms that are subject to the parameters of its design model. Based on this statement, it was decided to include in the input parameters not one model series, but M, with the same length and constructive model's parameters.

During the fitness calculation, each of the N generated series in scope of the same chromosome would be matched against each of the M inputs. The result of the calculation is the sum of the smallest differences between one generated series and M inputs:

$$q(X_n) = \sum_{i=0}^{N} \min_{j=1..M} \left(\sum_{k=1}^{K} (TS_{In_{j,k}} - TS_{i,k}(C_{MS}, C_{TS}))^2 \right) \quad (4)$$

where $q(X_n)$ – the fitness indicator of the chromosome X_n, K – defined length of the time series, $TS_{In_{j,k}}$, $TS_{i,k}(C_{MS}, C_{TS})$ – the values of the *k*-th point of the *j*-th input TS_{In} and *i*-th generated $TS_i(C_{MS}, C_{TS})$ time series accordingly.

This approach covers the stochastic nature of time series and increases the accuracy and controllability of the reconstruction process. The only drawback of this approach is the large number of series comparison operations, which is equal to $N * M$ and leads to deterioration of the time efficiency of work. To mitigate the impact a decision was made to reduce the number of chromosomes in population to limit the calculations number.

7. Experiments with stochastic time series reconstruction

The testing process of the developed approach were carried out based on model stochastics time series (Figure 6), the constructive models of which consist of a mathematical part with (V_f, dV_f, D_f) parameters and an L-system with one replacement rule, the right part of which is 4 to 10 characters long. For each of the experiments, the components of the model series were generated by a random image in compliance with the specified rules, such as the length of the right-hand side of the rules, as well as the limits of possible V_f, dV_f, D_f parameter values.

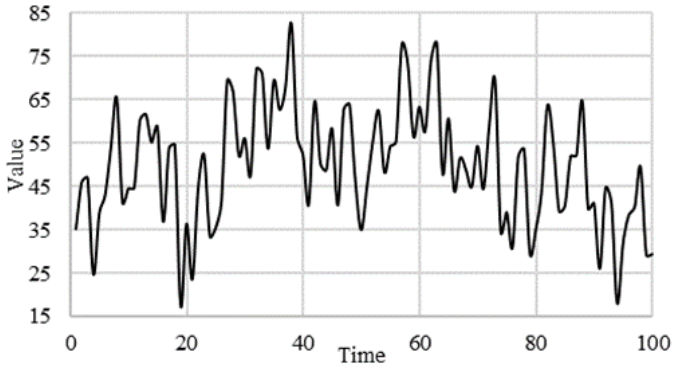

Figure 6. Reconstruction model of the stochastic time series. Rule:
$f \to f + ff - ff; V_f = 30; dV_f = 15; Df = 45$

The process of each experiment took place before reconstruction of the constructive model and its parameters. At the same time, it should be noted that an error of 1.5% from the $V_f^{\max}(V_f^{\max}=100)$ is allowed for the values of V_f and $\mathrm{d}V_f$. The components of the L-system must be identical.

In total, 50 experiments were conducted for M = 50 and N = 50. For each of them, constructive models were reconstructed within the limits of correspondence described above.

8. Conclusions

In this study, the constructive-synthesizing approach for time series' model reconstruction has been adapted to multi-agent environment. The main working entities – agents and their responsibilities defined.

To ensure a parallel computing process, a cluster structure and logical connections between agents were developed. The found solution makes it possible to ensure scalability by increasing the number of worker agents in the system.

The multi-agent environment implements a distributed genetic algorithm with an additional migration process and the chromosome age attribute. Migration produces an exchange between the gene pools of working agents.

Research was conducted with two types of series: deterministic and stochastic. In the case of deterministic series, a comparison was made with the previous single-threaded implementation on the same data sets, the results of which showed a significant improvement in performance.

In our models, deterministic series represent a special case of stochastic ones, and all methods applied to stochastic series can also be applied to deterministic ones.

For the stochastic series, the structure of the chromosome and the calculating model the fitness function were modified. The fitness function is calculated considering the proximity of several time series generated based on the constructive model of the chromosome with several original series.

The temporal and functional efficiency of the proposed modeling method significantly depends on the quality and size of the modeled time series. In this work, the effectiveness of the method in the studied domain was experimentally confirmed.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] A. S. Weigend, "Time series prediction: forecasting the future and understanding the past", Routledge, 2018.

[2] D. Kosiorowski, D. Mielczarek and J. P. Rydlewski, "Double functional median in robust prediction of hierarchical functional time series", arXiv preprint arXiv:1710.02669, 2017.

[3] R. Graf, S. Zhu, B. Sivakumar, "Forecasting river water temperature time series using a wavelet–neural network hybrid modelling approach". Journal of Hydrology, vol. 578, 2019, pp. 124115. doi: 10.1016/j.jhydrol.2019.124115

[4] Y. Liang, S. Ke, J. Zhang, X. Yi and Y. Zheng, "Geoman: Multi-level attention networks for geo-sensory time series prediction" International Joint Conference on Artificial Intelligence, 2018, pp. 3428-3434.

[5] J. Zhang, Y. Zheng and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction", Thirty-First AAAI Conference on Artificial Intelligence, 2017, pp. 1655-1661.

[6] K. McAlinn and M. West, "Dynamic Bayesian predictive synthesis in time series forecasting", Journal of econometrics, vol. 210(1), 2019, pp. 155-169. doi: doi.org/10.1016/j.jeconom.2018.11.010

[7] M. C. Mariani, M. A. M. Bhuiyan, O. K. Tweneboah and H. G. Huizar, "Forecasting the volatility of geophysical time series with stochastic volatility models" International Journal of Mathematical and Computational Sciences, 11(10), 2017, pp. 444-450

[8] A. Kuchansky, A. Biloshchytskyi, Y. Andrashko, S. Biloshchytska, T. Honcharenko and V. Nikolenko, “Fractal Time Series Analysis in Non-Stationary Environment”, International Scientific-Practical Conference Problems of Infocommunications, Science and Technology, IEEE, 2019, pp. 236-240. doi: 10.1109/PICST47496.2019.9061554
[9] V. Bulakh, L. Kirichenko and T. Radivilova, “Time series classification based on fractal properties”, IEEE Second International Conference on Data Stream Mining & Processing (DSMP), IEEE, 2018, pp. 198-201.
[10] V. Skalozub, V. Ilman and V. Shynkarenko, “Ontological support formation for constructive-synthesizing modeling of information systems development processes”, Eastern-European Journal of Enterprise Technologies, vol. 5, issue 4 (95), 2018, pp. 55–63. doi: 10.15587/1729-4061.2018.143968
[11] V. Skalozub, V. Ilman and V. Shynkarenko, “Development of ontological support of constructive-synthesizing modeling of information systems”, Eastern-European Journal of Enterprise Technologies, vol. 6, issue 4 (90), 2017, pp. 58–69. doi: 10.15587/1729-4061.2017.119497
[12] S. Müller, “Grammatical theory: From transformational grammar to constraint-based approaches”, Language Science Press, 2023
[13] V. Shynkarenko, K. Lytvynenko, R. Chyhir and I. Nikitina, “Modeling of Lightning Flashes in Thunderstorm Front by Constructive Production of Fractal Time Series”. In: Advances in Intelligent Systems and Computing IV, vol. 1080, Springer, 2020, pp. 173-185/ doi: 10.1007/978-3-030-33695-0_13
[14] B. R. Bielefeldt, G. W. Reich, P. S. Beran and D. J. Hartl, “Development and validation of a genetic L-System programming framework for topology optimization of multifunctional structures”, Computers & Structures, vol. 218, 2019, pp. 152–169. doi: 10.1016/j.compstruc.2019.02.005.
[15] V. Shynkarenko and A. Zhadan, "Modeling of the Deterministic Fractal Time Series by One Rule Constructors" 2020 IEEE 15th International Conference on Computer Sciences and Information Technologies (CSIT), Zbarazh, Ukraine, 2020, pp. 336-339, doi: 10.1109/CSIT49958.2020.9321923.
[16] K. Zhang, Z. Yang, T. Başar, “Multi-agent reinforcement learning: A selective overview of theories and algorithms” Handbook of reinforcement learning and control, 2021, pp. 321-384.
[17] “Fire-and-Forget conversation pattern” Enterprise Integration Patterns book website. https://www.enterpriseintegrationpatterns.com/patterns/conversation/FireAndForget.html (accessed Apr. 07, 2024).
[18] “RabbitMQ is the most widely deployed open source message broker” RabbitMQ official website. https://rabbitmq.com/ (accessed Aug. 21, 2023).