mathematics

MDPI

Article

Mathematical Models for the Design of GRID Systems to Solve Resource-Intensive Problems

Valeriya V. Tynchenko [1,2], Vadim S. Tynchenko [3,4,5,*], Vladimir A. Nelyub [3], Vladimir V. Bukhtoyarov [3,5], Aleksey S. Borodulin [3], Sergei O. Kurashkin [3,4,6], Andrei P. Gantimurov [3] and Vladislav V. Kukartsev [1,3,7]

1 Department of Computer Science, Institute of Space and Information Technologies, Siberian Federal University, 660041 Krasnoyarsk, Russia; 051301@mail.ru (V.V.T.); vlad_saa_2000@mail.ru (V.V.K.)
2 Department of Computer Science and Computer Engineering, Institute of Computer Science and Telecommunications, Reshetnev Siberian State University of Science and Technology, 660037 Krasnoyarsk, Russia
3 Scientific and Educational Center "Artificial Intelligence Technologies", Bauman Moscow State Technical University, 105005 Moscow, Russia; vladimir.nelub@emtc.ru (V.A.N.); vladber@list.ru (V.V.B.); alexey.borodulin@emtc.ru (A.S.B.); scorpion_ser@mail.ru (S.O.K.); agantimurov@emtc.ru (A.P.G.)
4 Information-Control Systems Department, Institute of Computer Science and Telecommunications, Reshetnev Siberian State University of Science and Technology, 660037 Krasnoyarsk, Russia
5 Department of Technological Machines and Equipment of Oil and Gas Complex, School of Petroleum and Natural Gas Engineering, Siberian Federal University, 660041 Krasnoyarsk, Russia
6 Laboratory of Biofuel Compositions, Siberian Federal University, 660041 Krasnoyarsk, Russia
7 Department of Information Economic Systems, Institute of Engineering and Economics, Reshetnev Siberian State University of Science and Technology, 660037 Krasnoyarsk, Russia
* Correspondence: vadimond@mail.ru; Tel.: +7-95-0973-0264

Abstract: Artificial neural networks are successfully used to solve a wide variety of scientific and technical problems. The purpose of the study is to increase the efficiency of distributed solutions for problems involving structural-parametric synthesis of neural network models of complex systems based on GRID (geographically disperse computing resources) technology through the integrated application of the apparatus of evolutionary optimization and queuing theory. During the course of the research, the following was obtained: (i) New mathematical models for assessing the performance and reliability of GRID systems; (ii) A new multi-criteria optimization model for designing GRID systems to solve high-resource computing problems; and (iii) A new decision support system for the design of GRID systems using a multi-criteria genetic algorithm. Fonseca and Fleming's genetic algorithm with a dynamic penalty function was used as a method for solving the stated multi-constrained optimization problem. The developed program system was used to solve the problem of choosing an effective structure of a centralized GRID system that was configured to solve the problem of structural-parametric synthesis of neural network models. To test the proposed approach, a Pareto-optimal configuration of the GRID system was built with the following characteristics: average performance–103.483 GFLOPS, cost–500 rubles per day, availability rate–99.92%, and minimum performance–51 GFLOPS.

Keywords: performance model; reliability model; GRID system; genetic algorithm; multi-criteria optimization; optimization problem; Pareto-optimization; neural network models

MSC: 68M10

Citation: Tynchenko, V.V.; Tynchenko, V.S.; Nelyub, V.A.; Bukhtoyarov, V.V.; Borodulin, A.S.; Kurashkin, S.O.; Gantimurov, A.P.; Kukartsev, V.V. Mathematical Models for the Design of GRID Systems to Solve Resource-Intensive Problems. *Mathematics* **2024**, *12*, 276. https://doi.org/10.3390/math12020276

Academic Editors: Man-Fai Leung, Wenming Cao and Hangjun Che

Received: 21 December 2023
Revised: 11 January 2024
Accepted: 13 January 2024
Published: 15 January 2024

1. Introduction

To date, machine learning (ML) methods have been widely used in various fields [1–3]. In [4], a group of scientists developed an automated system for controlling and diagnosing the condition of a pumping unit with its identification as an object of vibration loading using ML methods. ML is used for classification of pump wear conditions, which allows optimizing the pumping unit operation [5,6].

In [7], the authors investigated the usefulness of ML methods for predicting and analyzing the diverse physical characteristics of polymers. As a result of the study, it was found that the best results were achieved using a Random Forest with the highest R2 scores of 0.71, 0.73, and 0.88 for glass transition, thermal decomposition, and melting temperatures, respectively. One study [8] considered the forecasting of the completion storage area networks (SAN) and developed a system to model and simulate the further loading of SAN on previously observed load measurements. The authors of [9] used logical analysis of data (LAD) to develop an algorithm to search for pairs of strong patterns (prime and spanned). Traditional learning algorithms have a number of drawbacks, such as getting "stuck" in local minima and low convergence rates. Currently, scientists more often use metaheuristic algorithms for tuning the neural networks' weights. To solve the described problem, the authors of study [10] developed an algorithm based on the spider's life cycle, which involves the application of a new metaheuristic optimization algorithm for weight tuning. Tynchenko et al. [11] analyzed the applicability of algorithms for solving the problem of selecting effective parameters of the electron beam welding (EBW) process. As a result, they developed a mathematical model that applies machine learning to predict effective process parameters.

Currently, one of the effective ways to solve complex resource-intensive optimization problems is the use of various techniques of parallelization of the computational process. In particular, when implementing parallel genetic algorithms (PGA), the initial population is divided into several subpopulations, each of which will be processed independently from other subpopulations with the subsequent exchange (migration) of individuals, which improves the accuracy and efficiency of the algorithm.

Geographically distributed information (GRID) systems are a form of distributed computing in which a "virtual supercomputer" is represented as clusters of networked, loosely coupled, heterogeneous computers working together to perform a huge number of tasks. This technology is used to solve scientific and mathematical problems that require significant computational resources [12].

Thus, the authors of [13] considered the issue of training neural networks when applying genetic algorithms (GA) parallelized in GRID systems. The use of GRID systems for this purpose makes it possible to effectively utilize a large computational potential for training neural systems.

The authors of the study [14] presented a description of approaches to creating single-level and two-level GRID systems. The advantages of using single-level systems, in particular large-scale systems, are highlighted. Although GRIDs and peer-to-peer networks are decentralized, distributed computing environments, GRID systems tend to focus on downstream issues, such as resource allocation, performance, reliability, and security. Thus, ref. [15] provides an overview of various methods for solving security problems in the grid computing environment.

Understanding GRID environments is incomplete without considering scheduling, which is the process of determining the end goal of matching jobs submitted to the GRID with the corresponding available resources. The study [16] examines algorithms related to communication cost, execution time, error factors, task duplication, data processing intensity, and heterogeneous network behavior. Syed Nasir et al. [17] proposed two GRID scheduling algorithms: a dynamic multi-level algorithm using the median and a dynamic multi-level algorithm using the square root. In turn, the authors of [18] presented a multi-level hybrid scheduling algorithm and a multi-level scheduling algorithm with two queues.

The authors of [19] consider the relationships among high-performance computing, GRID computing and cloud computing from the point of view of their motivation, strengths, and weaknesses. The authors aim to combine the best characteristics of each into a model of how these paradigms can work together and how scientific workloads can be managed in such a hybrid computing environment. One study [20] proposed a distributed load balancing algorithm that can handle any type of GRID structure. The proposed algorithm is divided into two steps to reduce the time spent on executing jobs or to reduce the response

time and communication cost between transferring jobs from one computing node to another in a GRID architecture. The authors of [21] classify methods for solving GRID system load balancing and load balancing problems based on their objective functions into two categories: application-oriented and resource-intensive methods.

The authors of [22] propose a distributed grid and dynamic load balancing algorithm using a forest-based GRID model. The problem of resource heterogeneity was also considered (a single node is independent of the physical architecture of the GRID network).

However, the above studies do not describe the problem of designing GRID systems aimed at solving complex resource-intensive problems regarding the structural-parametric synthesis of artificial neural networks that impose certain features on the construction of such computing systems. The implementation of this approach requires the use of data mining techniques, which are described in Section 2. In Section 3, new mathematical models are developed for evaluating the performance and reliability of GRID systems, and a new multi-criteria optimization model for designing a GRID system is proposed to solve highly resource-intensive computing problems. The research results in the development of a decision support system (DSS) for solving multi-criteria problems are described in Section 4.

Thus, the main contributions of the research are as follows:

- New mathematical models of GRID system performance and reliability estimation are built on the basis of a mass service system apparatus;
- A new multi-criteria optimization model of the GRID system is designed to solve highly resource-intensive computational problems, which could be solved with the multi-criteria genetic algorithms to provide the wide range of Pareto-optimal solutions;
- A new decision-support system for GRID systems is designed with the use of a multi-criteria genetic algorithm.

2. Problem Statement

Currently, an effective and much cheaper alternative to multiprocessor and multi-machine computing systems for the realization of parallel computations is the use of distributed computer networks. The intensive development and introduction of perspective GRID network technology allows the efficient use of computing capacities by combining them into a high-performance, highly reliable and inexpensive GRID system for solving complex and resource-intensive tasks.

The automation of GRID system resource allocation and their coordination in the process of solving complex scientific and technical tasks requires the development and application of formal modeling and optimization methods to form an effective configuration of GRID system computing resources implementing the main functions.

As an example of a computationally complex task, this study considers the task of designing artificial neural networks, including the synthesis of their architectures, using distributed evolutionary algorithms.

2.1. Application of Parallel Genetic Algorithms for the Synthesis of Neural Network Model Structure

The formalized formulation of the problem of multi-criteria unconditional optimization of the artificial neural network (ANN) structure is as follows [23,24] (1):

$$\begin{cases} \frac{1}{m}\sum\limits_{j=1}^{m}\sqrt{\frac{\sum_{k=1}^{n}\left(OUT_k^j\left(C,W,\overline{af}\right)-y_k^j\right)^2}{n}} \to \min\limits_{C,\,W,\,\overline{af}}, \\ N_{link}(C)+\sum\limits_{i=1}^{N_n}K_i\left(\overline{af}\right) \to \min\limits_{C,\,\overline{af}}. \end{cases} \quad (1)$$

where $OUT_k^j\left(C,W,\overline{af}\right)$—the real value of the k-th output of the ANN having the structure $(C,W,\overline{af})$, when the j-th image is fed to its inputs; y_k^j—the ideal (desired) output state of

the k-th image; C—the matrix of ANN links of dimension $N_n \times N_n$; W—matrix of weights of links of ANN of dimension $N_n \times N_n$; $\overline{af}$—vector of activation functions on neurons in ANN of dimension N_n; N_n—number of neurons in ANN; $N_{link}(\cdot)$—number of links of ANN; $n = 0, 1, 2, \ldots$—number of processed links; $K_i(\overline{af}) = \frac{T_i^{act}(\overline{af},P)}{T^{link}(P)}$—the coefficient of relative complexity of calculating the activation function on the i-th neuron; and m—the size of the training sample.

Performing calculations using adaptive search algorithms to solve the above problem of optimizing the structure and parameters of artificial neural networks requires significant computational resources (speed, memory capacity) and takes a large amount of time. The application of PGA and calculations using a distributed computing system (DCS) is caused by the reduction of time spent on solving the task at hand.

The choice of how to parallelize a genetic algorithm depends on the following [25,26]:

- How fitness is assessed and mutation is applied;
- Whether a single population is or multiple subpopulations are used;
- If multiple subpopulations are used, how individuals are exchanged between them;
- How selection is applied (globally or locally).

Depending on how each of these points is implemented, we obtain different ways of parallelizing a genetic algorithm. Let us give the most general classification of PGAs [27]:

- PGAs with distributed fitness evaluation (Master-Slave PGA);
- Multi-population PGAs (coarse grained algorithms, island model);
- PGAs with mass parallelism (fine grained algorithms);
- PGAs with dynamic subpopulations;
- Stationary PGAs;
- Non-stationary PGAs;
- Hybrid methods (static subpopulations with migration or with distributed fitness estimation within each subpopulation).

The parallel genetic algorithm (PGA) with distributed fitness evaluation [28] stands out as one of the pioneering and successful implementations of parallel genetic algorithms. This approach is often referred to as global parallelization, employing a client-server model or utilizing distributed fitness evaluation.

In this algorithm, a single population is employed, and the evaluation of individuals as well as the application of genetic operators occurs in parallel. The global execution includes selection and inter-breeding, allowing each individual to compete and interbreed with any other individual. The primary aspect that undergoes parallelization is the computation of the fitness function. Since this typically requires knowledge only of the individuals being evaluated (not the entire population), there is no need for information exchange during this phase.

To implement this method, client–server software systems are commonly utilized. The server stores the population, while the clients handle fitness calculation, mutation application, and, occasionally, bit swapping in the genome as part of crossbreeding.

The algorithm can be categorized as synchronous if the server halts and awaits fitness values for the entire population before progressing to the next generation. A synchronous PGA with a distributed fitness estimation shares the same properties as a simple genetic algorithm, with the primary distinction lying in its speed.

An asynchronous version of PGA with a distributed fitness estimation is also feasible. In this case, the algorithm does not pause to wait for slower processors. Consequently, the asynchronous genetic algorithm deviates from a simple GA, resembling parallel steady-state GAs, with the primary difference occurring in the selection operator. In asynchronous algorithms, selection waits until a portion of the population has been processed, whereas in steady-state GAs, selection is applied without waiting, considering the available population.

Implementing a synchronous GA with a distributed fitness estimation is relatively straightforward, and significant performance improvements can be expected if the cost of information transfer remains lower than the cost of computation. However, a classic

limitation exists in that the entire process must wait until the slowest processor completes the fitness computation before the selection operator can be applied. The asynchronous algorithm addresses this limitation but alters the dynamics of the GA significantly, making it challenging to analyze. One straightforward method to implement an asynchronous PGA with distributed fitness estimation is to employ tournament selection, considering only the fraction of individuals in the population for whom fitness values have already been computed.

Multi-population PGAs [29,30] adopt a distinctive approach, utilizing a small number of large populations (subpopulations) that evolve independently on different computational nodes, effectively achieving geographic isolation. Individuals within a subpopulation compete solely within that specific subpopulation. Introducing a migration operator allows periodic movement of individuals from one subpopulation to another.

The nature of migration depends on the following parameters:

- Topology of connectivity between subpopulations (isolated populations, complete graph, ring, etc.);
- Migration rate, which determines the number of individuals to be moved;
- Migration pattern, which determines which individuals will be moved to another population ("better", "worse", randomly selected) and which individuals will be replaced ("better", "worse", randomly selected);
- Migration interval, which determines the frequency of movement (may be constant-synchronous migration and may vary depending on some event-asynchronous migration).

In cases the involve the application of multi-population PGAs, it is possible to achieve a reduction in the running time of the algorithm by a factor greater than the number of computational nodes, which is explained by the possibility of a more efficient parallel implementation of the algorithm than in the case of sequential GA [31,32]. Other advantages of multi-population PGAs include the coverage of a large search space and a low probability of premature convergence.

2.2. Multi-Criteria Multi-Population Parallel Genetic Algorithm with Topology Restructuring of Linkage between Populations

2.2.1. Insights into Multi-Criteria Multi-Population PGA with Restructuring of the Topology of Links between Populations

In this paper, a multi-criteria multi-population parallel genetic algorithm (MP PGA) was used to solve the multi-criteria optimization problem of selecting an efficient ANN structure that was supplemented by a procedure of restructuring the topology of links between populations in the course of solving the optimization problem [33,34]. This procedure is implemented as follows: temporal links between populations isolated from each other are dynamically added to the initially chosen basic topology of links between populations so that populations that are not sufficiently performing at the current moment can obtain additional migrants from the best individuals of those populations that are sufficiently performing.

The main characteristics by which the quality of algorithms can be evaluated are noted as follows [35]:

- The number of globally Pareto-undominated individuals in the algorithm population;
- The spread of globally Pareto-undominated individuals of the algorithm on the Pareto set (or Pareto front).

Here, globally Pareto-undominated individuals refers to individuals who are undominated not only when compared to individuals of their own population, but also when compared to all individuals of all other parallel functioning populations.

2.2.2. Scheme of the Proposed MP PGA

Let us introduce the following notations:

N—number of populations;

m_i—migration rate of the i-th population ($i = \overline{1, N}$);

k_i—migration period of the i-th population ($i = \overline{1, N}$).

Let us consider the operation of the algorithm on the example of the l-th population.

1. Perform k_1 cycles of multi-criteria GA.
2. Calculate the values Q_i, $i = \overline{1, N}$ of the quality indicators of all algorithms.
3. If $Q_l < \frac{1}{N}\sum_{i=1}^{N} Q_i$, then add a temporal link from population j, for which $Q_j > \frac{1}{N}\sum_{i=1}^{N} Q_i$, to population l with probability $P_j = \frac{Q_j}{\sum_k Q_k}$, where $Q_k > \frac{1}{N}\sum_{i=1}^{N} Q_i$.
4. Perform migration according to the obtained topology.

The efficiency of the developed modified multi-population multi-criteria PGA is proven by the results of computational experiments in the structural-parametric synthesis of ANN.

Thus, PGAs are successfully applied to solve complex scientific and technical problems. However, despite all the advantages, these solutions require large computing power, which entails large material costs. The solution to this problem is the use of GRID systems. To apply GRID systems to solve complex scientific and technical problems, it is necessary to analyze the requirements for the task and to select the appropriate architecture of the GRID system. The application of GRID systems for solving complex scientific and technical problems can improve computational efficiency, reduce time costs, and increase productivity. In this study, a DSS is developed to select an efficient structure of a centralized GRID system. The application of this solution involves the use of idle computing power at the enterprise. Using available resources of the enterprise, it is possible to solve a complex scientific and technical problem, for example, the choice of an effective structure of the centralized GRID system.

3. Materials and Methods

Currently, the concept of GRID is being intensively developed around the world, which is based on the idea of integrating a certain set of spatially distributed information and computing resources of various types in order to provide the ability to run a wide range of applications on any set of these resources, regardless of their location. In reality, GRID is a computer infrastructure that consists of resources located in different places with telecommunications connecting them (network resources) and interconnecting middleware (between base and application software) coordinated throughout the infrastructure, supporting remote operations and performing functions of control and management of the operating environment.

GRID technology includes only the most general and universal aspects that are the same for any GRID system architecture, protocols, interfaces, and services. Using this technology and filling it with specific content, it is possible to realize one or another GRID system intended for solving one or another class of applied tasks.

There are two main criteria that distinguish GRID systems from other systems that provide shared access to resources [36]:

- The GRID system coordinates disparate resources that do not have a common control center.
- The GRID system is built on the basis of standard and open protocols, services, and interfaces.

In addition, any GRID system should have the following properties:

- Flexibility, i.e., the ability to provide shared access to potentially any type of resource;
- Scalability, i.e., the ability to maintain the performance of the GRID system with a significant increase or decrease in its composition;
- A flexible and powerful security subsystem;
- The possibility to control resources: application of local and global policies and quotas;
- Quality of service guarantees;
- The possibility of simultaneous, coordinated work with several resources.

At present, many large-scale projects involving the creation of GRID systems have already been realized and are being implemented around the world, the main purpose of which is to test the developed GRID technologies and their development. Now, these projects are mainly created in various scientific applications. For example, in China, the project of creating a distributed supercomputer MilkyWay-2 was realized [37], a new version of supercomputer Stampede 1-Stampede2 was developed and in America [38]. A group of scientists from the USA developed a Jetstream-based system for NSF to serve as a distributed production cloud resource [39]. According to their idea, Jetstream provides interactive virtual machines through the Atmosphere interface (VMs) [40]. Based on the results of analyzing the projects, it can be concluded that there are three directions of GRID technology development:

- Computational GRID;
- GRID for intensive data processing;
- Semantic GRID for operating data from different databases.

The goal of the first direction is to achieve the maximum speed of computations due to global distribution of these computations between computers by connecting supercomputer centers (DEISA project). The goal of the second direction is to process huge amounts of data with relatively simple programs according to the principle of "one task-one processor", which is carried out under the auspices of the European Union. It involves more than 90 scientific and educational institutions from all over the world, including Russia.

The EGEE project is closely connected at this stage of development with the LCG project, which, in fact, is its technological base. Active work is underway to expand the Russian GRID infrastructure (RDIG) [41].

3.1. GRID System Performance Evaluation Model

In general, the infrastructure for GRID technology realization is a parallel multicomputer network (PMC-network), which, according to [42], is a hardware–software complex consisting of a set of computers connected with each other by a data transmission medium as well as software communication tools. In a functioning GRID system, it is necessary to perform automatic monitoring of available computing resources and the distribution of computing tasks on them with the possibility of setting the distribution strategy. The high performance of the selected configuration of the GRID system is reduced in response to the performance evaluation of the studied variants of the system structure as well as to optimize the structure and mode of operation to achieve the specified performance at a minimum cost of the system. The main tool in the study of the performance of computing systems are performance models, allowing assessment of the characteristics of computational processes and resource utilization. These characteristics are necessary to identify factors affecting performance as well as bottlenecks and underutilized resources [43].

Since the choice of an effective configuration of a GRID system for the distributed solution of complex tasks of a certain class requires matching the structure of the GRID with the structure of the parallel algorithm of the task being solved, let us consider the main approaches to the realization of the computational process based on GRID technology.

In a general case, the task that can be solved using GRID technology can be represented as a set of process-servers, which perform task distribution, and a set of computational process-clients. The problem is solved as follows: the process-server sends a package of tasks to each client computational process, which performs calculations and sends the result back to the process-server, and it, in turn, collects all the results and performs the general part of processing [44].

When building a PMC network, there are two possible approaches to organizing the management of the computing process:

- Centralized, when there is a special computer, which is intended for the distribution of PMC network resources;
- Decentralized, when all computers participate in the process of resource management based on equal rights.

Both variants of PMC network implementation considered above allow the organization of infrastructure for GRID task execution with a synchronous start. In this case, the first of the launched processes becomes the coordinator for the centralized variant; it is usually the process launched directly on the server. In turn, all other available calculators' processes, on which calculations will be performed directly, are started at the moment of startup. In a centralized version of the PMC network, it is also possible to implement tasks with an asynchronous start. This is done by sending notifications with information about a new computer to all already running processes.

The considered methods of interaction of GRID system computational nodes in the process of the distributed solution of a complex problem agree well with the computationally efficient technique of large-block parallelization of the algorithm of a complex problem [45]. This technique consists of designing a parallel algorithm consisting of identical sequential branches that are weakly data linked and comparable in the number of operations. One of the design techniques of large-block parallelization is parallelization by cycles, when separate branches of the algorithm parallelize the computation of all iterations of one cycle [46,47].

Based on the above, let us build an analytical model of performance evaluation of the centralized variant of the GRID system when implementing the task with a synchronous start. Let the configuration of the GRID system includes an arbitrary number of client computing nodes of different performances and a multiprocessor server.

Characteristics of the modeled GRID system:

- N—number of client resources of the GRID system (client computing nodes);
- n—number of homogeneous processors of the server node of the system;
- ω_i—performance of the i-th client resource of the GRID system (FLOPS);
- ω_{srv}—performance of processors on the server node of the system (FLOPS);
- ν_i—speed of the communication channel of the i-th client resource of the GRID system with its server node (bit/s).

The generalized structure of the considered GRID system is presented in Figure 1.

Characteristics of the problem to be solved [48]:

NO_{alg}—average computational complexity (number of machine operations) of one iteration of calculations on one branch of the problem solving algorithm (op.);

NO_{ctrl}—average computational complexity of the control algorithm of one branch of the problem-solving algorithm (op.);

V_{alg}—average data volume of client–server exchange (bits);

T_{lim}—maximum allowable time of the problem solution.

Let us denote the following:

t_i^{on}—time of turning on the i-th resource of the GRID system, $i = \overline{1,N}$;

t_i^{off}—time of turning off the i-th resource of the GRID system, $i = \overline{1,N}$;

$\delta_i(t)$—i-th node availability indicator-determines whether the i-th GRID node is present in the network at a given time t.

The availability indicator can be evaluated with the following expression (2):

$$\delta_i(t) = \begin{cases} 1, \; if \begin{pmatrix} \left(t_i^{on} < t_i^{off}\right) \wedge \\ \wedge\left(t \geq t_i^{on}\right) \wedge \\ \wedge\left(t < t_i^{off}\right) \end{pmatrix} \vee \begin{pmatrix} \left(t_i^{on} > t_i^{off}\right) \wedge \\ \wedge\left(t \geq t_i^{on}\right) \wedge \\ \wedge\left(t > t_i^{off}\right) \end{pmatrix} \vee \begin{pmatrix} \left(t_i^{on} > t_i^{off}\right) \wedge \\ \wedge\left(t < t_i^{on}\right) \wedge \\ \wedge\left(t < t_i^{off}\right) \end{pmatrix} \\ 0, \; otherwise. \end{cases}, \quad (2)$$

The process of GRID system functioning can be considered as a sequential change of states at some time interval Δt, and the apparatus of mass service theory can be applied to describe this process [49]. Without the limitation of generality, we can assume that during the specified time interval Δt, there is no scheduled connection of new computational nodes to the GRID system, and there is no scheduled disconnection of functioning computational

nodes. Thus, the availability indices of GRID nodes are constant, predetermined values that do not depend on time, i.e., there is a set of values δ_i, $i = \overline{1, N}$.

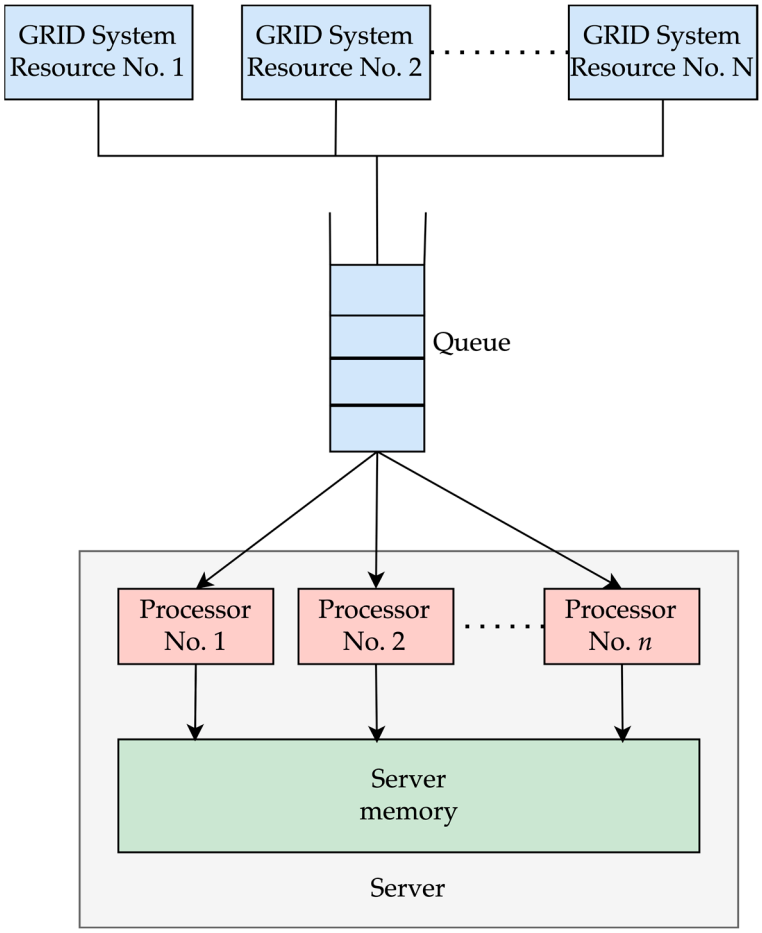

Figure 1. Generalized structure of the centralized GRID system.

Let us represent the process of GRID system functioning as a closed mass service system (MSS) with waiting and random distribution of requests of all types to all server processors without interaction among themselves [50]. Let the GRID system have N client nodes, each of which needs service at some random moments of time.

Service is performed through n homogeneous processors of the server node. The flow of requests for service from clients of each type is Poisson with the parameter λ_i, where $i = \overline{1, N}$. The service intensity of each request is subject to an exponential distribution law with the parameter μ_i. A newly received request for service is sent with equal probability to any of the free processors on the server and accepted for service. If all processors are busy, then the request received for service is queued and awaits service. Service discipline is a random equally probable selection from the queue.

The considered MSS may be in the following states:

- $a^{0,0}_{0,0,\ldots,0}$—there are no requests for service in the system, all server processors are free;
- $a^{1,0}_{1,0,\ldots,0}$—there is one request in the system from the 1st client resource of the GRID system, one request is served on one server processor, and there is no queue;
- $a^{1,0}_{0,0,\ldots1}$—there is one request in the system from the N-th client resource of the GRID system, one request is served on one server processor, and there is no queue;
- $a^{k,l}_{j_1,j_2,\ldots,j_N}$—the system contains j_i requests from the i-th client resource of the GRID system, where $i = 1, 2, \ldots, N$; k server processors are busy with servicing; and l requests are in queues for service;

- $a_{1,1,\ldots,1}^{h,M-h}$—the system contains one request from each client resource of the GRID system, h server processors are busy with servicing, and (N-h) requests are in queues for service,

where $h = \begin{cases} n, if N > n \\ N, if N \leq n \end{cases}; k = \overline{0,h}; l = \overline{0,(N-k)}$.

A fragment of the state graph is shown in Figure 2.

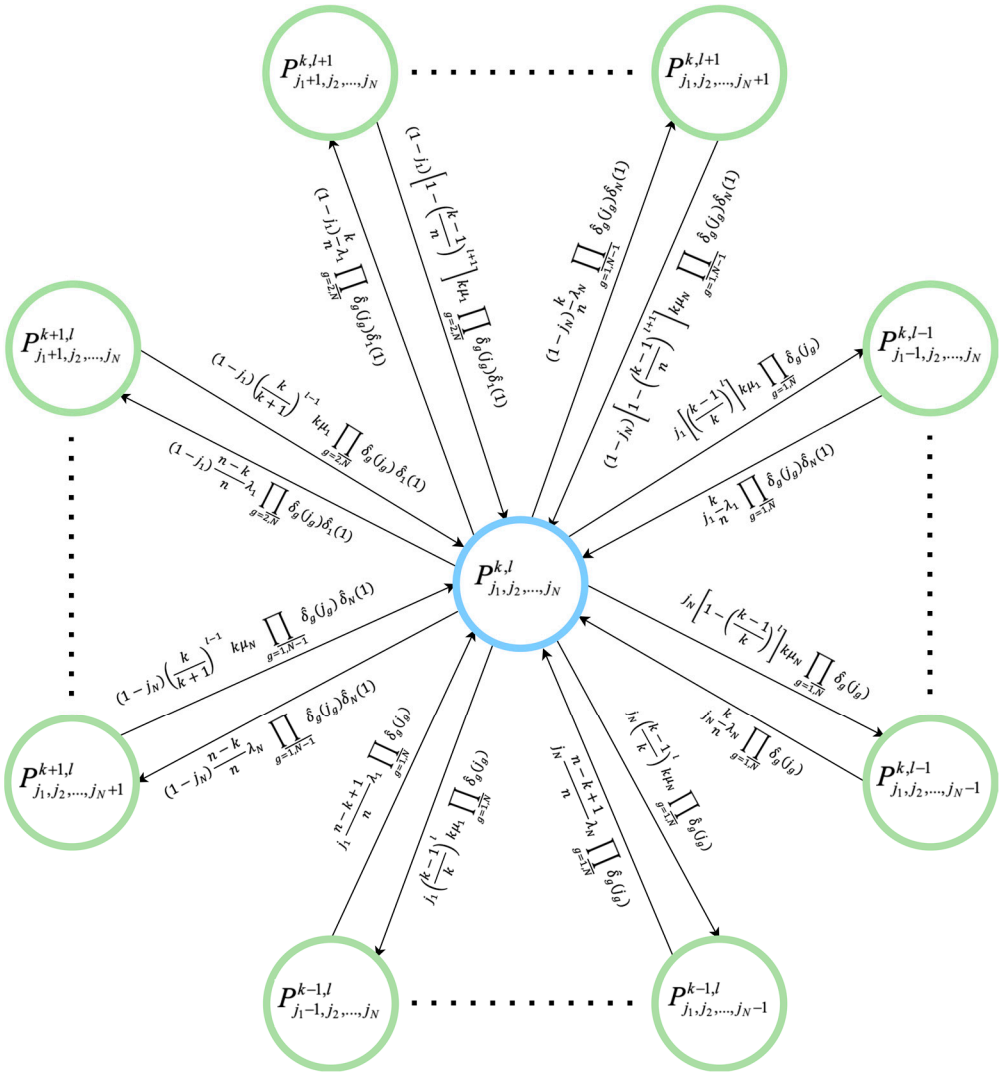

Figure 2. Fragment of the MSS state graph for GRID system performance modeling.

In order to exclude the loss of those states of the MSS when no requests for service are received from switched off client computational nodes, we modify the indicator of computational nodes availability as follows (3):

$$\hat{\delta}_i(j_i) = \begin{cases} 1, \ if \ j_i = 0, \\ \delta_i, \ otherwise. \end{cases} \tag{3}$$

To solve the system of equations in a general form, let us substitute into (4) [51]:

$$P_{j_1,j_2,\ldots,j_N} = \prod_{i=1}^{N} \widetilde{P}_{j_i} \tag{4}$$

and write the system of equations as the following form (5):

$$
\begin{aligned}
&-\sum_{i=1}^{N}\left[(1-j_i)\lambda_i \prod_{\substack{g=\overline{1,N}\\ g\neq i}} \hat{\delta}_g(j_g)\hat{\delta}_i(1) + j_i\mu_i \prod_{g=\overline{1,N}} \hat{\delta}_g(j_g)\right]\widetilde{P}_{j_1}\cdot\widetilde{P}_{j_2}\cdot\ldots\cdot\widetilde{P}_{j_{1N}}+\\
&+\left[j_1\lambda_1\tfrac{n-k+1}{n}\widetilde{P}_{j_1-1}\cdot\widetilde{P}_{j_2}\cdot\ldots\cdot\widetilde{P}_{j_N}+\cdots\right.\\
&\left.\cdots+j_N\lambda_N\tfrac{n-k+1}{n}\widetilde{P}_{j_1}\cdot\widetilde{P}_{j_2}\cdot\ldots\cdot\widetilde{P}_{j_N-1}\right]\prod_{g=\overline{1,N}}\hat{\delta}_g(j_g)+\\
&+\left[j_1\lambda_1\tfrac{k}{n}\widetilde{P}_{j_1-1}\cdot\widetilde{P}_{j_2}\cdot\ldots\cdot\widetilde{P}_{j_N}+\cdots+{}_N\,\lambda_N\tfrac{k}{n}\widetilde{P}_{j_1}\cdot\widetilde{P}_{j_2}\cdot\ldots\cdot\widetilde{P}_{j_N-1}\right]\prod_{g=\overline{1,N}}\hat{\delta}_g(j_g)+\\
&+\left[1-\left(\tfrac{k-1}{k}\right)^{l+1}\right](1-j_1)\mu_1\widetilde{P}_{j_1+1}\cdot\widetilde{P}_{j_2}\cdot\ldots\cdot\widetilde{P}_{j_N}\prod_{g=\overline{2,N}}\hat{\delta}_g(j_g)\hat{\delta}_1(1)+\cdots\\
&\cdots+\left[1-\left(\tfrac{k-1}{k}\right)^{l+1}\right](1-j_N)\mu_N\widetilde{P}_{j_1}\cdot\widetilde{P}_{j_2}\cdot\ldots\cdot\widetilde{P}_{j_N+1}\prod_{g=\overline{1,N-1}}\hat{\delta}_g(j_g)\hat{\delta}_N(1)+\\
&+\left(\tfrac{k}{k+1}\right)^{l}(1-j_1)\mu_1\widetilde{P}_{j_1+1}\cdot\widetilde{P}_{j_2}\cdot\ldots\cdot\widetilde{P}_{j_N}\prod_{g=\overline{2,N}}\hat{\delta}_g(j_g)\hat{\delta}_1(1)+\left(\tfrac{k}{k+1}\right)^{l}(1-j_N)\mu_N\widetilde{P}_{j_1}\cdot\\
&\cdot\widetilde{P}_{j_2}\cdot\ldots\cdot\widetilde{P}_{j_N+1}\prod_{g=\overline{1,N-1}}\hat{\delta}_g(j_g)\hat{\delta}_N(1)=0.
\end{aligned}
\tag{5}
$$

Let us regroup (4) in such a way that $\widetilde{P}_{j_i}$ enters into the corresponding expressions containing λ_i, μ_i, and we make the separation of variables by indices (6).

$$
\begin{aligned}
&\{j_1\lambda_1\tfrac{n-k+1}{n}\prod_{g=\overline{1,N}}\hat{\delta}_g(j_g)\widetilde{P}_{j_1-1}-\left[(1-j_1)\lambda_1\prod_{g=\overline{2,N}}\hat{\delta}_g(j_g)\hat{\delta}_1(1)+j_1\mu_1\prod_{g=\overline{1,N}}\hat{\delta}_g(j_g)\right]\widetilde{P}_{j_1}+\\
&+j_1\lambda_1\tfrac{k}{n}\prod_{g=\overline{1,N}}\hat{\delta}_g(j_g)\widetilde{P}_{j_1-1}+\left[1-\left(\tfrac{k-1}{k}\right)^{l+1}\right](1-j_1)\mu_1\prod_{g=\overline{2,N}}\hat{\delta}_g(j_g)\hat{\delta}_1(1)\widetilde{P}_{j_1+1}+\\
&+\left(\tfrac{k}{k+1}\right)^{l}(1-j_1)\mu_1\prod_{g=\overline{2,N}}\hat{\delta}_g(j_g)\hat{\delta}_1(1)\widetilde{P}_{j_1+1}\}\widetilde{P}_{j_2}\cdot\widetilde{P}_{j_3}\cdots\widetilde{P}_{j_N}+\\
&+\{j_2\lambda_2\tfrac{n-k+1}{n}\prod_{g=\overline{1,N}}\hat{\delta}_g(j_g)\widetilde{P}_{j_2-1}-\\
&-\left[(1-j_2)\lambda_2\prod_{\substack{g=\overline{1,N}\\ g\neq 2}}\hat{\delta}_g(j_g)\hat{\delta}_2(1)+j_2\mu_2\prod_{g=\overline{1,N}}\hat{\delta}_g(j_g)\right]\widetilde{P}_{j_2}+\\
&+j_2\lambda_2\tfrac{k}{n}\Pi_{g=\overline{1,N}}\hat{\delta}_g(j_g)\widetilde{P}_{j_2-1}+\left[1-\left(\tfrac{k-1}{k}\right)^{l+1}\right](1-j_2)\mu_2\Pi_{\substack{g=\overline{1,N}\\ g\neq 2}}\hat{\delta}_g(j_g)\hat{\delta}_2(1)\widetilde{P}_{j_2+1}+\\
&+\left(\tfrac{k}{k+1}\right)^{l}(1-j_2)\mu_2\prod_{\substack{g=\overline{1,N}\\ g\neq 2}}\hat{\delta}_g(j_g)\hat{\delta}_2(1)\widetilde{P}_{j_2+1}\}\widetilde{P}_{j_1}\cdot\widetilde{P}_{j_3}\cdots\widetilde{P}_{j_N}+\cdots\\
&\cdots+\{j_N\lambda_N\tfrac{n-k+1}{n}\prod_{g=\overline{1,N}}\hat{\delta}_g(j_g)\widetilde{P}_{j_N-1}-\\
&-\left[(1-j_N)\lambda_N\prod_{g=\overline{1,N-1}}\hat{\delta}_g(j_g)\hat{\delta}_N(1)+j_N\mu_N\prod_{g=\overline{1,N}}\hat{\delta}_g(j_g)\right]\widetilde{P}_{j_N}+\\
&+j_N\lambda_N\tfrac{k}{n}\prod_{g=\overline{1,N}}\hat{\delta}_g(j_g)\widetilde{P}_{j_N-1}+\left[1-\left(\tfrac{k-1}{k}\right)^{l+1}\right](1-j_N)\mu_N\prod_{g=\overline{1,N-1}}\hat{\delta}_g(j_g)\hat{\delta}_N(1)\widetilde{P}_{j_N+1}+\\
&+\left(\tfrac{k}{k+1}\right)^{l}(1-j_N)\mu_N\prod_{g=\overline{1,N-1}}\hat{\delta}_g(j_g)\hat{\delta}_N(1)\widetilde{P}_{j_N+1}\}\widetilde{P}_{j_1}\cdot\widetilde{P}_{j_2}\cdots\widetilde{P}_{j_{N-1}}=0.
\end{aligned}
\tag{6}
$$

The system of linear Equation (6) will have a solution only when the expressions in curly brackets are zero, since $\widetilde{P}_{j_i}\neq 0$.

Performing a number of transformations, we obtain the following expression for the solution of the system of Equation (7):

$$\widetilde{P}_{j_i} = C_{k+l-1}^{l} C_n^k \frac{(k+\updownarrow)!}{n^{(k+\updownarrow)}} \rho_i^{j_i} \delta_i(j_i) \widetilde{P}_{0i} \tag{7}$$

Next, the inverse substitution of (7) into (5) is performed by taking into account the number of paths $\chi_{(k+l),j_i}$ to reach the state $a^{k,l}_{j_1,j_2,\ldots,j_N}$, which in the general case corresponds to a polynomial coefficient equal to (8) [52].

$$\chi_{(k+l),j_i} = (k+l)!, \tag{8}$$

where $(k+l) = 0, 1, \ldots, \sum_{i=1}^{N} j_i$.

Then, we write the following expression (9):

$$P^{k,l}_{j_1,j_2,\ldots,j_N} = \prod_{i=1}^{N} \widetilde{P}_{j_i} = C_{k+l-1}^{l} C_n^k \frac{(k+\updownarrow)!}{n^{(k+\updownarrow)}} \prod_{i=1}^{N} \rho_i^{j_i} \delta_i(j_i) P^{0,0}_{0,0,\ldots,0} \tag{9}$$

$P^{0,0}_{0,0,\ldots,0}$ is determined from the normalization condition (10):

$$P^{0,0}_{0,0,\ldots,0} = \left[\sum_{k=1}^{h} \sum_{l=0}^{N-k} C_{k+l-1}^{l} C_n^k \frac{(k+l)!}{n^{(k+l)}} \sum_{\substack{j_1 \in \{0,1\} \\ j_2 \in \{0,1\} \\ \ldots\ldots\ldots\ldots \\ j_N \in \{0,1\}}} \prod_{i=1}^{N} \rho_i^{j_i} \delta_i(j_i) \right]^{-1} \tag{10}$$

Substituting (10) into (9), we obtain the solution of the system of equations in general form (11):

$$P^{k,\updownarrow}_{j_1,j_2,\ldots,j_N} = \frac{C_{k+l-1}^{l} C_n^k \frac{(k+\updownarrow)!}{n^{(k+\updownarrow)}} \prod_{i=1}^{N} \rho_i^{j_i} \delta_i(j_i)}{\sum_{k=1}^{h} \sum_{l=0}^{N-k} C_{k+l-1}^{l} C_n^k \frac{(k+l)!}{n^{(k+l)}} \sum_{\substack{j_1 \in \{0,1\} \\ j_2 \in \{0,1\} \\ \ldots\ldots\ldots\ldots \\ j_N \in \{0,1\}}} \prod_{i=1}^{N} \rho_i^{j_i} \delta_i(j_i)} \tag{11}$$

When evaluating the performance of a GRID system, the total number of requests under service and in queues, regardless of their type, is decisive. The total number of requests in queues can be determined using the concepts of a set of stationary probabilities $P_{k,l}$ (12), average queue length l_{avg} (13), and coefficients θ_i (14) of average performance losses for clients of each type [53]:

$$P_{k,\updownarrow} = \sum_{\substack{j_1 \in \{0,1\} \\ j_2 \in \{0,1\} \\ \ldots\ldots\ldots\ldots \\ j_N \in \{0,1\}}} P^{k,\updownarrow}_{j_1,j_2,\ldots,j_{N-1},j_N}, \tag{12}$$

$$\updownarrow_{avg} = \sum_{k=1}^{h} \sum_{\updownarrow=1}^{N-k} P_{k,\updownarrow} \cdot \updownarrow, \tag{13}$$

$$\theta_i = 1 + \frac{\updownarrow_{cp} \tau_i}{T_{oi} + \tau_i} \tag{14}$$

Then, the average performance of a GRID system can be determined using expression (15).

$$\prod_{avg} = \sum_{i=1}^{N} \frac{\omega_i}{\theta_i} \delta_i \tag{15}$$

To evaluate the system performance by expressions (13)–(15) it is necessary to calculate probabilities $P_{k,l}$. When deriving expression (12) for $P_{k,l}$, it was assumed that the flows of requests for service are simple with the parameters λ_i, and the time of service of requests of clients of the i-th type by the server obeys the exponential distribution law with the parameter μ_i. For real computational systems, po-currents differ from Poisson and exponential and depend on the architecture of the computational system and parameters of the algorithms executed on it [54].

The average performance of a given GRID configuration changes during the daily cycle of the system operation when connecting and/or disconnecting computational nodes. Thus, to calculate the average performance of a GRID system at an arbitrary point in time, it is necessary to modify expressions (11)–(15) as follows (16)–(20) [55]:

$$\prod_{avg}(t) = \sum_{i=1}^{N} \frac{\omega_i}{\theta_i(t)} \delta_i(t), \tag{16}$$

$$\theta_i(t) = 1 + \frac{\updownarrow_{avg}(t)\tau_i}{T_{oi} + \tau_i}, \tag{17}$$

$$\updownarrow_{avg}(t) = \sum_{k=1}^{h} \sum_{\updownarrow=1}^{N-k} P_{k,\updownarrow}(t) \cdot \updownarrow, \tag{18}$$

$$P_{k,\updownarrow}(t) = \sum_{\substack{j_1 \in \{0,1\} \\ j_2 \in \{0,1\} \\ \ldots\ldots\ldots\ldots \\ j_N \in \{0,1\}}} P^{k,\updownarrow}_{j_1,j_2,\ldots,j_{N-1},j_N}(t), \tag{19}$$

$$P^{k,\updownarrow}_{j_1,j_2,\ldots,j_N}(t) = \frac{C^l_{k+l-1} C^k_n \frac{(k+\updownarrow)!}{n^{(k+\updownarrow)}} \prod_{i=1}^{N} \rho_i^{j_i} \hat{\delta}_i(t, j_i)}{\sum_{k=1}^{h} \sum_{l=0}^{N-k} C^l_{k+l-1} C^k_n \frac{(k+l)!}{n^{(k+l)}} \sum_{\substack{j_1 \in \{0,1\} \\ j_2 \in \{0,1\} \\ \ldots\ldots\ldots\ldots \\ j_N \in \{0,1\}}} \prod_{i=1}^{N} \rho_i^{j_i} \hat{\delta}_i(t, j_i)}. \tag{20}$$

where $\hat{\delta}_i(t, j_i) = \begin{cases} 1, if & \begin{array}{l} (j_i = 0) \vee \left((j_i = 1) \wedge \left(t_i^{on} < t_i^{off} \right) \wedge \left(t \geq t_i^{on} \right) \wedge \left(t \geq t_i^{off} \right) \right) \vee \\ \vee \left((j_i = 1) \wedge \left(t_i^{on} > t_i^{off} \right) \wedge \left(t \geq t_i^{on} \right) \wedge \left(t > t_i^{off} \right) \right) \vee \\ \vee \left((j_i = 1) \wedge \left(t_i^{on} > t_i^{off} \right) \wedge \left(t < t_i^{on} \right) \wedge \left(t < t_i^{off} \right) \right) \end{array} \\ 0, otherwise \end{cases}$

Thus, the proposed approach allows the selection of a performance-efficient configuration of the GRID system that is customized to solve complex tasks of a certain class.

3.2. GRID System Reliability Assessment Model

The principles underlying the construction of GRID systems require the development and application of new unconventional approaches to solving the problem of the reliability of such systems.

The reliability of a large-scale distributed computing system with a programmable structure is understood as its ability to automatically (programmatically) adjust and organize the functioning of such structural schemes, which, in case of failures and recovery

of calculators, provide a given level of performance or, to put it differently, the ability to use a fixed number of serviceable calculators (when implementing parallel programs for solving complex problems). This notion characterizes the capabilities of a computer system to process information in the presence of structural redundancy (represented by a part of calculators) and when using parallel programs with a given number of branches.

When studying the reliability of large-scale DCS with a programmable structure, the following peculiarity should be taken into account: the process of restoring the detected failed elementary machines (EM) does not provide for repairing the machines but for reconfiguration of the system. In this case, the DCS operability check and the search for failed machines are performed by means of (self-)control and (self-)diagnostics (controller and diagnostician), respectively. Reconfiguration of the system involves the establishment of a new configuration of the program with a given number n of serviceable Ems; this function is performed by the reconfigurator. To create a new configuration of the main subsystem, in a general case, machines from redundancy and/or reserve can be used. The controller, diagnostician, and reconfigurator are components of a distributed operating system (OS). This composition is essentially a virtual recovery unit (RU) for the DCS.

In GRID-based systems, all the work of managing, allocating, and optimizing the use of system resources is dynamically performed by the system software during the operation of the GRID system. The resource broker, using the monitoring system, constantly monitors the status of resources and automatically takes the necessary measures to effectively reallocate resources in cases of resource failures and to prevent their overload and downtime. Thus, the GRID monitoring system performs the control function, while the diagnostic and reconfiguration functions are performed by the resource broker.

One of the widely used approaches to calculating the reliability of a computing system is to describe and investigate its operation using the apparatus of mass service theory, when DCS is represented as a mass service system with expectation.

To calculate the reliability in this study, an analytical model of the reliability of the centralized version of the GRID system in the implementation of the task with synchronous start will be built.

Let the configuration of the GRID system includes an arbitrary number of client computing nodes of different performance, connected by a hub with a multiprocessor server.

Characteristics of the modeled GRID system:

- N—number of types of client resources of the GRID system (client computing nodes);
- m_i—number of client resources of the GRID system of i-th type;
- n—number of homogeneous processors of the server node of the system.

When calculating reliability indicators, the stochastic model of DCS functioning is taken as a basis. One of the main requirements for the methodology of calculating the quality indicators of computing systems is the requirement of adequacy of stochastic models of DCS functioning to the real process of their work or realization of the principle of quasi-analogy, which in relation to the problem under consideration guarantees not similarity between stochastic models and DCS functioning, but satisfactory for practice accuracy of calculations.

Failures occurring in the GRID system are eliminated by means of the recovery procedure, which is performed by the resource broker together with the monitoring system, which together form a virtual (logical) recovery device. Thus, after a failure, a computational resource either enters the recovery device for maintenance or (if it is busy) is queued for recovery. The service discipline is a random equiprobable selection from the queue. It is considered that, at any given time, RU can be either free or busy restoring at most one computational resource. Let the recovery time for any failed resource of the GRID system obey the exponential distribution law with the parameter μ^{cl}, which is the intensity of recovery of failed resources by virtual RU. For the case of a GRID system, the intensity μ^{cl} is interpreted as the average number of computational resources included per unit of

time by the RU (namely, the resource broker) in the GRID instead of the failed resources. In this case, the average recovery time of one computational resource will be equal to (21).

$$\tau = \left(\mu^{cl}\right)^{-1} = \tau_{ctrl} + \tau_d + \tau_{rec} \tag{21}$$

Here, τ_{ctrl}, τ_d, and τ_{rec} are mathematical expectations of time of control, diagnostics, and reconfiguration of the system, respectively (system reconfiguration).

Each element of the server part of the computing system also fails at some random moments of time and needs to be restored. Failure flows from all elements of the server part of the GRID system are simple and have the following intensities:

λ^{srv}—failure intensity of server processors;

λ^{hub}—failure intensity of the hub.

It is assumed that the recovery of server processors is done at the hardware level. The recovery time for all failed server and hub processors obeys an exponential distribution law with the parameters:

μ^{srv}—intensity of server processors recovery;

μ^{hub}—recovery intensity of the hub.

To account for the scheduling of GRID system computing resources, we introduce the following parameters:

$t_{i,j}^{on}$—time of turning on the j-th resource of the i-th type in the GRID system, $i = \overline{1,N}$, $j = \overline{1,m_i}$;

$t_{i,j}^{off}$—time of turning off the j-th resource of the i-th type in the GRID system, $i = \overline{1,N}$, $j = \overline{1,m_i}$;

$\delta_{i,j}(t)$—availability indicator of the j-th resource of the i-th type determines whether the j-th resource of the i-th type is present in the GRID system at a given time t.

The availability indicator can be evaluated with the following expression (22):

$$\delta_{i,j}(t) = \begin{cases} 1, \ if \begin{pmatrix} \left(t_{i,j}^{on} < t_{i,j}^{off}\right) \wedge \\ \wedge \left(t \geq t_{i,j}^{on}\right) \wedge \\ \wedge \left(t < t_{i,j}^{off}\right) \end{pmatrix} \vee \begin{pmatrix} \left(t_{i,j}^{on} > t_{i,j}^{off}\right) \wedge \\ \wedge \left(t \geq t_{i,j}^{on}\right) \wedge \\ \wedge \left(t > t_{i,j}^{off}\right) \end{pmatrix} \vee \begin{pmatrix} \left(t_{i,j}^{on} > t_{i,j}^{off}\right) \wedge \\ \wedge \left(t < t_{i,j}^{on}\right) \wedge \\ \wedge \left(t < t_{i,j}^{off}\right) \end{pmatrix}, \\ 0, \ otherwise. \end{cases} \tag{22}$$

Without limiting generality, we can assume that during the time interval in which the system is considered, there is no scheduled connection of new computational nodes to the GRID system, and there is no scheduled disconnection of functioning computational nodes. Thus, the availability indices of GRID nodes are constant, predetermined values that do not depend on time, i.e., there is a set of values $\delta_{i,j}$, $i = \overline{1,N}$.

To account for the schedule of GRID system nodes, we introduce $\hat{\delta}_i\left(j_i^{cl}\right)$, which is the probability that at the considered moment of time ${j_i}^{cl}$ resources of the i-th type are enabled (23).

$$\hat{\delta}_i\left(j_i^{cl}\right) = \frac{\sum\limits_{\substack{\omega \subseteq \{1,\ldots,m_i\} \\ |\omega| = j_i^{cl}}} \prod_{g \in \omega} \delta_{i,g}}{C_{m_i}^{j_i^{cl}}}. \tag{23}$$

The process of functioning of such a computing system is represented by a closed MSS with waiting, which can be in the following states:

- $a_{0,0,0,\ldots,0}$—all elements of the GRID system are faulty and are being restored. The computing process has stopped.
- $a_{1,0,0,\ldots,0}$—one server processor is operational, and the $(n-1)$ processor is faulty and is being restored. All other elements of the GRID system are faulty and are being restored. The computing process is stopped.

- $a_{0,1,0,\ldots,0}$—hub is working properly. All other elements of the GRID system are faulty and are being restored. The computing process has stopped.
- $a_{0,0,1,\ldots,0}$—one client resource of the 1st type in the GRID system is operational, and $(m_1 - 1)$ client resources of the 1st type are faulty and are being restored. All other elements of the GRID system are faulty and are being restored. The computing process has stopped.
- $a_{0,0,0,\ldots,1}$—one client resource of the N-th type in GRID system is operational, and $(m_N - 1)$ client resources of the N-th type are faulty and are being restored. All other elements of the GRID system are faulty and are being restored. The computing process has stopped.
- $a_{j^{srv},0,j_1^{cl},\ldots,j_N^{cl}}$—$j^{\text{srv}}$ server processors are healthy, and $(n - j^{\text{srv}})$ are faulty and are being restored. The hub is faulty and is being restored. In addition, j_1^{cl} client resources of the 1st type of GRID system are faulty, and $(m_1 - j_1^{cl})$ are faulty and are being restored, …, j_N^{cl} client resources of GRID systems of the N-th type are operational, and $(m_N - j_N^{cl})$ are faulty and are being restored. The computing process has stopped;
- $a_{j^{srv},1,j_1^{cl},\ldots,j_N^{cl}}$—$j^{\text{srv}}$ server processors are operational and participate in the computing process, and $(n - j^{\text{srv}})$ are faulty and are being restored. The hub is operational and participates in the computing process. In addition, j_1^{cl} client resources of the 1st type of GRID system are operational and participate in the computing process, and $(m_1 - j_1^{cl})$ are faulty and being restored, …, j_N^{cl} of client resources of the N-th type of GRID system are operational and participating in the computing process, and $(m_N - j_N^{cl})$ are faulty and being restored;
- $a_{n,1,m_1,\ldots,m_N}$—all elements of the GRID system are operational and participate in the computing process.

A fragment of the state graph is shown in Figure 3.

The system of equations for the stationary mode has a single solution [56] taking into account the normalization condition (24):

$$\sum_{j^{srv}=0}^{n} \sum_{j^{hub}=0}^{1} \sum_{j_1^{cl}=0}^{m_1} \ldots \sum_{j_N^{cl}=0}^{m_N} P_{j^{srv},j^{hub},j_1^{cl},\ldots,j_N^{cl}} = 1 \tag{24}$$

To solve the system of linear equations in general form (25), we make the following substitution:

$$P_{j^{srv},j^{hub},j_1^{cl},\ldots,j_N^{cl}} = P_{j^{srv}}^{srv} \cdot P_{j^{hub}}^{hub} \cdot \prod_{i=1}^{N} P_{j_i^{cl}}^{cl_i} \tag{25}$$

By analogy with [56], the system of linear equations has the following form (26):

$$\begin{aligned}
&(n - j^{srv} + 1)\mu^{srv} P_{j^{srv}-1}^{srv} - ((n - j^{srv})\mu^{srv} + j^{srv}\lambda^{srv}) P_{j^{srv}}^{srv} + (j^{srv} + 1)\lambda^{srv} P_{j^{srv}+1}^{srv} = \\
&= 0 j^{hub}\mu^{hub} P_0^{hub} - \left(\left(1 - j^{hub}\right)\mu^{hub} + j^{hub}\lambda^{hub}\right) P_{j^{hub}}^{hub} + \left(1 - j^{hub}\right)\lambda^{hub} P_1^{hub} = \\
&= 0 (m_1 - j_1^{cl} + 1)\mu_1^{cl} \cdot \prod_{g=\overline{1,N}} \hat{\delta}_g\left(j_g^{cl}\right) P_{j_1^{cl}-1}^{cl_1} - ((m_1 - j_1^{cl})\mu_1^{cl} \prod_{g=\overline{2,N}} \hat{\delta}_g\left(j_g^{cl}\right) \hat{\delta}_1(j_1^{cl} + 1) + j_1^{cl}\lambda_1^{cl} \cdot \\
&\cdot \prod_{g=\overline{1,N}} \hat{\delta}_g(j_g^{cl})) P_{j_1^{cl}}^{cl_1} + (j_1^{cl} + 1)\lambda_1^{cl} \prod_{g=\overline{2,N}} \hat{\delta}_g(j_g^{cl}) \hat{\delta}_1(j_1^{cl} + 1) P_{j_1^{cl}+1}^{cl_1} = 0 \\
&\ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \\
&(m_N - j_N^{cl} + 1)\mu_N^{cl} \prod_{g=\overline{1,N}} \hat{\delta}_g(j_g^{cl}) P_{j_N^{cl}-1}^{cl_N} - ((m_N - j_N^{cl})\mu_N^{cl} \prod_{g=\overline{1,N-1}} \hat{\delta}_g(j_g^{cl}) \hat{\delta}_N(j_N^{cl} + 1) + \\
&j_N^{cl}\lambda_N^{cl} \prod_{g=\overline{1,N}} \hat{\delta}_g(j_g^{cl})) P_{j_N^{cl}}^{cl_N} + + (j_N^{cl} + 1)\lambda_N^{cl} \prod_{g=\overline{1,N-1}} \hat{\delta}_g(j_g^{cl}) \hat{\delta}_N(j_N^{cl} + 1) P_{j_N^{cl}+1}^{cl_N} = 0.
\end{aligned} \tag{26}$$

The considered GRID system includes several different types of structural elements, including server processors (n elements), hub, and N types of GRID system client resources (m_i—is the number of resources of the i-th type, $i = \overline{1, N}$). Thus, a GRID system consists of N+ different types of building blocks. Each type of GRID system element corresponds to one of the system Equation (26), which in turn is a system of MSS equations consisting

of an appropriate number of demand sources of the same type with one serving unit. Consequently, the state graph of the initial MSS will be a set of $N + 2$ single-line graphs.

Performing a number of transformations [56,57], we obtain the following expression for calculating the probabilities (27):

$$P_{j^{srv},j^{hub},j_1^{cl},\ldots,j_N^{cl}} = \frac{\frac{n!}{(n-j^{srv})!}\rho_{srv}^{j^{srv}}\cdot\rho_{hub}^{j^{hub}}\cdot\prod_{i=1}^{N} j_i^{cl}!\rho_{cl_i}^{j_i^{cl}}\sum\limits_{\substack{\omega\subseteq\{1,\ldots,m_i\}\\ |\omega|=j_i^{cl}}}\prod_{g\in\omega}\delta_{i,g}}{\sum_{j^{srv}=0}^{n}\sum_{j^{hub}=0}^{1}\sum_{j_1^{cl}=0}^{m_1}\cdots\sum_{j_N^{cl}=0}^{m_N}\frac{n!}{(n-j^{srv})!}\rho_{srv}^{j^{srv}}\cdot\rho_{hub}^{j^{hub}}\cdot\prod_{i=1}^{N} j_i^{cl}!\rho_{cl_i}^{j_i^{cl}}\sum\limits_{\substack{\omega\subseteq\{1,\ldots,m_i\}\\ |\omega|=j_i^{cl}}}\prod_{g\in\omega}\delta_{i,g}} \quad (27)$$

Figure 3. Fragment of the MSS state graph for GRID system reliability modeling.

The relationship between the performance of an aircraft and its reliability is established by reliability indices. Consequently, reliability indices allow, firstly, to select such

a composition of the newly compiled DCS, which provides the specified levels of both performance and reliability, and, secondly, to analyze the quality of the operation of the existing DCS and to evaluate its capabilities to solve tasks.

The most common reliability indicator for the stationary mode of DCS operation is the availability factor of the computing system (K_r). The availability factor is understood as the probability that the system will be in a serviceable state at any point in time, except for the planned periods when the system is not intended for use [58]. Thus, the availability factor informs whether a system that has been in service long enough will have the required performance at any given moment of task arrival.

Given the definitions of the probability $P_{j^{srv},j^{hub},j_1^{cl},\ldots,j_N^{cl}}$ of the system being in state $a_{j^{srv},j^{hub},\ldots\ldots,j_N^{cl}}$, the availability factor of the considered GRID system is defined as follows (28):

$$K_r = \sum_{j^{srv}=1}^{n} \sum_{j_1^{cl}=1}^{m_1} \sum_{j_2^{cl}=0}^{m_2} \ldots \sum_{j_N^{cl}=0}^{m_N} P_{j^{srv},1,j_1^{cl},\ldots,j_N^{cl}} + \ldots + \sum_{j^{srv}=1}^{n} \sum_{j_1^{cl}=0}^{m_1} \ldots \sum_{j_{N-1}^{cl}=0}^{m_{N-1}} \sum_{j_N^{cl}=1}^{m_N} P_{j^{srv},1,j_1^{cl},\ldots,j_N^{cl}} \quad (28)$$

However, the availability factor of a given GRID configuration changes during the daily cycle of the system operation when connecting and/or disconnecting computational nodes. Thus, to calculate the average availability factor of the GRID system at an arbitrary moment of time, it is necessary to modify expressions (27) and (28) as follows to (29) and (30) [59]:

$$K_r(t) = \sum_{j^{srv}=1}^{n} \sum_{j_1^{cl}=1}^{m_1} \sum_{j_2^{cl}=0}^{m_2} \ldots \sum_{j_N^{cl}=0}^{m_N} P_{j^{srv},1,j_1^{cl},\ldots,j_N^{cl}}(t) + \ldots + \sum_{j^{srv}=1}^{n} \sum_{j_1^{cl}=0}^{m_1} \ldots \sum_{j_{N-1}^{cl}=0}^{m_{N-1}} \sum_{j_N^{cl}=1}^{m_N} P_{j^{srv},1,j_1^{cl},\ldots,j_N^{cl}}(t) \quad (29)$$

$$P_{j^{srv},j^{hub},j_1^{cl},\ldots,j_N^{cl}}(t) = \frac{\frac{n!}{(n-j^{srv})!}\rho_{srv}^{j^{srv}} \cdot \rho_{hub}^{j^{hub}} \cdot \prod_{i=1}^{N} j_i^{cl}! \rho_{cl_i}^{j_i^{cl}} \sum\limits_{\substack{\omega \subseteq \{1,\ldots,m_i\} \\ |\omega| = j_i^{cl}}} \prod_{g\in\omega} \delta_{i,g}(t)}{\sum_{j^{srv}=0}^{n} \sum_{j^{hub}=0}^{1} \sum_{j_1^{cl}=0}^{m_1} \ldots \sum_{j_N^{cl}=0}^{m_N} \frac{n!}{(n-j^{srv})!}\rho_{srv}^{j^{srv}} \cdot \rho_{hub}^{j^{hub}} \cdot \prod_{i=1}^{N} j_i^{cl}! \rho_{cl_i}^{j_i^{cl}} \sum\limits_{\substack{\omega \subseteq \{1,\ldots,m_i\} \\ |\omega| = j_i^{cl}}} \prod_{g\in\omega} \delta_{i,g}(t)} \quad (30)$$

where $\delta_{i,j}(t) = \begin{cases} 1, if \left(\begin{array}{l} \left(t_{i,j}^{on} < t_{i,j}^{off}\right) \wedge \\ \wedge \left(t \geq t_{i,j}^{on}\right) \wedge \\ \wedge \left(t < t_{i,j}^{off}\right) \end{array} \right) \vee \left(\begin{array}{l} \left(t_{i,j}^{on} > t_{i,j}^{off}\right) \wedge \\ \wedge \left(t \geq t_{i,j}^{on}\right) \wedge \\ \wedge \left(t > t_{i,j}^{off}\right) \end{array} \right) \vee \left(\begin{array}{l} \left(t_{i,j}^{on} > t_{i,j}^{off}\right) \wedge \\ \wedge \left(t < t_{i,j}^{on}\right) \wedge \\ \wedge \left(t < t_{i,j}^{off}\right) \end{array} \right) \\ 0, otherwise \end{cases}$,

$\rho_{srv} = \frac{\mu^{srv}}{\lambda^{srv}}$, $\rho_{hub} = \frac{\mu^{hub}}{\lambda^{hub}}$, $\rho_{cl_i} = \frac{1}{\lambda_i^{cl}(\tau_{ctrl}+\tau_d+\tau_{rec})}$.

In order to ensure the possibility of the effective application of the developed models and algorithms by specialists in various applied areas in solving practical problems, it is necessary to programmatically implement the mathematical apparatus proposed in this section in the form of an automated DSS, which allows the possibility to make a reasonable choice of efficiency in terms of the performance, reliability, and cost configuration of GRID system of a centralized type that is focused on solving complex problems of a certain class as well as on its basis to make the distribution of the GRID system of centralized type.

3.3. Statement of the Problem of Selecting an Effective Configuration of the GRID System and the Methods of Its Solution

The criteria of reliability and performance can also be considered quite consistent because when the number of computational nodes in a GRID system increases, both performance and reliability increase. Therefore, it is possible to translate the reliability criterion into constraints and optimize it using two criteria, namely, performance and cost, which are subject to certain conditions on reliability and performance.

Based on the above, the problem of selecting an effective variant of GRID system configuration is formalized as a multi-criteria optimization problem with two criteria for performance and cost, as well as a set of significant constraints that are placed onto the other criteria.

For detailed consideration, let us focus on two representatives of different classes of algorithms:

- A parallel genetic algorithm with distributed fitness evaluation;
- A multi-population parallel genetic algorithm.

When implementing a parallel genetic algorithm with a distributed fitness evaluation, the speed of solving the optimization problem is determined only by the total performance of the GRID system. On the contrary, in the case of a multi-population parallel genetic algorithm, the total system performance is and the number of parallel client computing nodes on which the populations evolve are important.

Accordingly, in the task of searching for an effective structure of the GRID system for the implementation of parallel genetic algorithm with a distributed fitness evaluation, it is necessary to maximize the total performance of the system (31).

$$\frac{\sum_{i=1}^{N_T-1} P\left(t_i, n_k, \omega_{srv}^k, \overline{z}\right)\cdot(t_{i+1}-t_i)}{t_{N_T}-t_1} \to \max \tag{31}$$

Here, N—number of client resources of the GRID system; k—number of server node included in the considered configuration of the GRID system ($k = \overline{1, M}$); M—number of server nodes available for inclusion in the considered configuration of the GRID system; n_k—number of processors of the k-th server node of the GRID system; ω_{srv}^k—performance of each processor of the k-th server node of the GRID system; t_i—points of the ascending ordered set of moments of time of switching on and off GRID system resources, including the beginning and the end of the time interval, $i = \overline{1,\ N_T}$; and N_T—number of points in the ascending ordered set of moments of switching on and off GRID system resources, including the beginning and the end of the time interval, $N_T = 2\sum_{j=1}^{N} z_j + 2$, $\overline{z}$ vector of indicators of inclusion of client nodes of the GRID system in the considered configuration (32).

$$\overline{z} : z_i = \begin{cases} 1,\ if\ i\text{–th client node enabled} \\ \text{in the GRID configuration}, i = \overline{1, N} \\ 0,\ otherwise. \end{cases} \tag{32}$$

In the case of realization of a multi-population parallel genetic algorithm, the performance criterion constructed for an algorithm with distributed fitness estimation remains relevant. However, in order to achieve high efficiency of evolutionary search, it is necessary that a sufficiently large number of populations are functioning at each moment of time. This requirement can be formulated in the form of an essential constraint (33).

$$\min_{i=\{1,\ldots,\ N_T-1\}} P(t_i, n_k, \omega_{srv}^k, \overline{z}) \geq P^{lim} \tag{33}$$

Here, P^{lim}—minimum required level of GRID system performance.

Let us summarize the above in the form of the following optimization model (34) and (35):

$$\frac{\sum_{i=1}^{N_T-1} P\left(t_i, n_k, \omega_{srv}^k, \overline{z}\right)\cdot(t_{i+1}-t_i)}{t_{N_T}-t_1} \to \max \tag{34}$$

$$C_k^{srv} + \sum_{i=1}^{N} C_i^{cl}\cdot z_i\cdot\left|t_i^{off} - t_i^{on}\right| \to \min \tag{35}$$

If the following conditions are met (36)–(38).

$$\frac{\sum_{i=1}^{N_T-1} K_r\left(n_k, \lambda_k^{srv}, \mu_k^{srv}, \lambda_k^{hub}, \mu_k^{hub}, \overline{z}\right)\cdot(t_{i+1}-t_i)}{t_{N_T}-t_1} \geq K_r^{lim} \tag{36}$$

$$\min_{i=\{1,\ldots,\ N_T-1\}} P(t_i, n_k, \omega_{srv}^k, \overline{z}) \geq P^{lim} \tag{37}$$

$$N^- \leq \sum_{i=1}^{N} z_i \leq N^+ \tag{38}$$

Here, λ_k^{srv}—failure rate of processors of the k-th server node of the GRID system, λ_k^{hub}—failure rate of the concentrator at the k-th server node of the GRID system, μ_k^{srv}—recovery rate of processors of the k-th server node of the GRID system, μ_k^{hub}—recovery rate of the concentrator at the k-th server node of the GRID system, $P(\cdot)$—performance criterion of the GRID system, P^{lim}—minimum required level of GRID system performance, C_k^{srv}—rental cost of k-th server node of the GRID system per day, C_i^{cl}—current rental cost of i-th client node of the GRID system per hour, $i = \overline{1, N}$, $K_r(\cdot)$—GRID system availability factor, K_r^{lim}—maximum allowable level of the GRID system availability factor, and N^+ and N^-–maximum and minimum possible number of client nodes of the GRID system, respectively.

All criteria and constraints in the optimization problem formulation are algorithmically specified target functions, which are calculated by the proposed models from the parameters of which are generated from tables (specifications of computational nodes) based on the binary vector generated by the optimization algorithm. For example, the number of clients in a GRID system, denoted by N in the performance evaluation model, is $\sum_i z_i$.

The complexity of the properties of the target functions and reliability constraints predetermined the choice of the method for solving the problem, namely, a multi-criteria genetic algorithm, the efficiency of which weakly depends on the properties of the optimized functions, which allows us to cope with the solution of complex global optimization problems.

3.4. Methods of Solving Multi-Criteria Optimization Problems

There are a number of classical methods of multi-criteria optimization: additive convolution method [60], constraint method [61], goal programming [62], minimax approach [63], etc.

Traditional approaches to constructing a Pareto set involve amalgamating all criteria into a single convolution function. Multiple optimization runs, each with varied tuning parameters for this function, are executed to identify the optimal Pareto set. However, classical multi-criteria optimization methods exhibit limitations when applied to intricate problems.

First, acquiring the Pareto-optimal set necessitates several independent optimization runs, introducing potential computational overhead due to coordination challenges. Second, the effectiveness of certain methods may be contingent upon the specific shape of the Pareto-optimal front. Additionally, obtaining supplementary information may prove difficult or even impossible. These factors contribute to the inadequacy of classical methods for addressing complex problems in multi-criteria optimization.

Conventionally, methods can be divided into 5 generalized groups (Figure 4):

- Weighting coefficients methods. In this method, each criterion is assigned a specific weight to reflect its relative importance. The problem comes down to single-criteria optimization, multiplying the values of the criteria by their weights.
- Methods of compromise solutions (Pareto Methods). These methods use the concept of the Pareto set, which is a set of optimal solutions in which it is impossible to improve one criterion without worsening the other. A balance is found between the criteria, preventing one from dominating the other.
- Compromise programming. This method allows one to find a compromise solution, considering the preferences of the decision maker regarding the importance of

each criterion. The problem is formulated as a search for an optimal compromise between criteria.

- Effective point methods. These methods are based on the concept of efficiency, where a point is considered efficient if there is no other point that surpasses it in all criteria. These methods use the concept of the Pareto frontier to find optimal solutions.
- Evolutionary algorithms for multi-criteria optimization. These methods are based on the application of evolutionary algorithms in which a population evolves towards optimal solutions considering several criteria. The principles of selection, recombination, and mutation are combined.

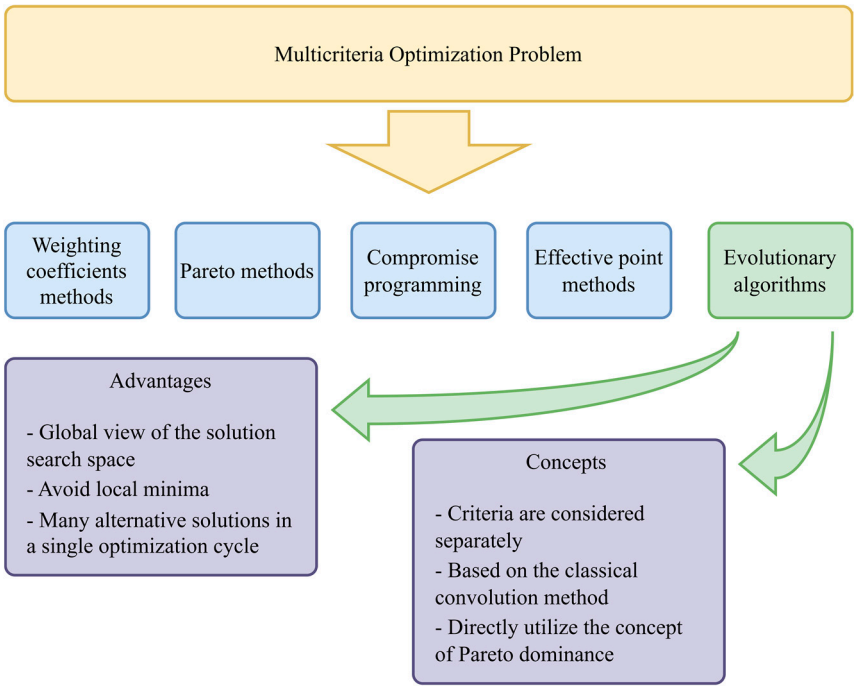

Figure 4. Methods for solving multi-criteria optimization problems.

The choice of method depends on the specific task and requirements of the decision maker. Combining different methods can also be an effective strategy for solving complex multi-objective optimization problems.

The application of the evolutionary approach overcomes the disadvantages inherent in classical multi-criteria optimization methods. In addition, evolutionary algorithms provide a global view of the solution search space, avoid local minima, and obtain many alternative solutions in a single optimization cycle.

The application of an evolutionary algorithm of multi-criteria optimization requires the solution of two main problems. First, it is necessary to select the suitability assignment and selection methods that would achieve the Pareto-optimal set. Second, the question of how to diversify the population to prevent premature convergence to achieve a well-distributed and uniform non-dominated set must be addressed [64].

There are different concepts of suitability assignment and selection in the process of solving a multi-criteria optimization problem. A distinction is made between approaches where the criteria are considered separately, approaches based on the classical convolution method, and methods that directly utilize the concept of Pareto dominance.

In the initial scenario, the algorithm dynamically alternates between criteria during the selection of individuals, following a specific pattern. For instance, different criteria can be employed to fill equal portions of a temporary population [65]. Alternatively, individuals may be compared based on a predetermined or random order of criteria [66]. In a specific investigation [67], each criterion is assigned a probability that dictates its utilization in the subsequent selection step, with the probability being either user defined or randomly

selected. However, these approaches tend to converge towards the so-called "extreme" solution and exhibit sensitivity to the non-convex nature of the Pareto-optimal front [68].

To address the aforementioned challenges, ref. [69] introduced the concept of assigning fitness to an individual based on Pareto dominance. The authors proposed an iterative ranking procedure. Initially, all primarily non-dominant individuals receive a rank of "one" and are temporarily removed from the population. Subsequently, the next non-dominant individuals are assigned a rank of "two", and so forth. Ultimately, an individual's fitness value determines its rank. This approach offers the advantage of assigning fitness based on the overall quality of the population, in contrast to other methods that independently assign fitness to each individual. Following this, additional researchers have suggested novel fitness assignment schemes grounded in the Pareto dominance principle [70,71]. While this evolutionary algorithm class theoretically has the capability to detect any Pareto-optimal solution, its performance may be influenced by the dimensionality of the search space.

There are many other approaches to solving multi-criteria optimization problems using evolutionary methods. Here, we only the main ones: methods based on mass selection with parameter variation, pre-determination, population reinitialization, convolution, elitism, and a whole group of approaches based on population diversity, or "niching", such as fitness alignment, limited crossing, and isolation.

The presented study focuses on the configuration of computing systems, and one of the classical genetic algorithms for multi-criteria optimization, namely, Fonseca and Fleming's Genetic Algorithm (FFGA), which directly uses Pareto-optimality, will be described.

4. Automated Decision Support System "Formation of an Effective GRID System Configuration"

The efficiency of the application of the algorithmic model apparatus to the solution of practical problems depends on the quality of program implementation. Initially, the object-oriented approach was chosen as a method for the design and development of software tools. The next step involved the choice of programming language. The software product required its portability in other operating environments. Thus, C++ language was used for the development of the interface [72,73], and C language was used for the realization of the core of the software system [74,75].

Embarcadero RAD Studio visual programming environment designed for rapid application development (RAD) in C++ language was chosen as the toolkit for software system development [76,77].

The program products were subjected to testing both during the development process and after its completion [78].

Based on a set of mathematical models for assessing the performance and reliability of GRID systems, an automated decision support system "Formation of an effective GRID system configuration" was developed to select an effective GRID system structure in terms of performance, reliability and cost. The structure diagram of the DSS (Figure 5) shows that program system has a calculational core, which implements the proposed mathematical models for GRID performance and reliability estimation and the formulation of a multi-criteria optimization problem.

The block diagram of the DSS (Figure 6) shows how the multi-criteria FFGA [79] is implemented in the system and how the fitness function of each individual in the GA is assessed according to the proposed mathematical models. To solve the stated constrained optimization problem, the dynamic penalty function [80] is used.

The developed DSS can be used in the course of designing or modifying the architecture of GRID-type systems configured to solve complex problems of a certain class.

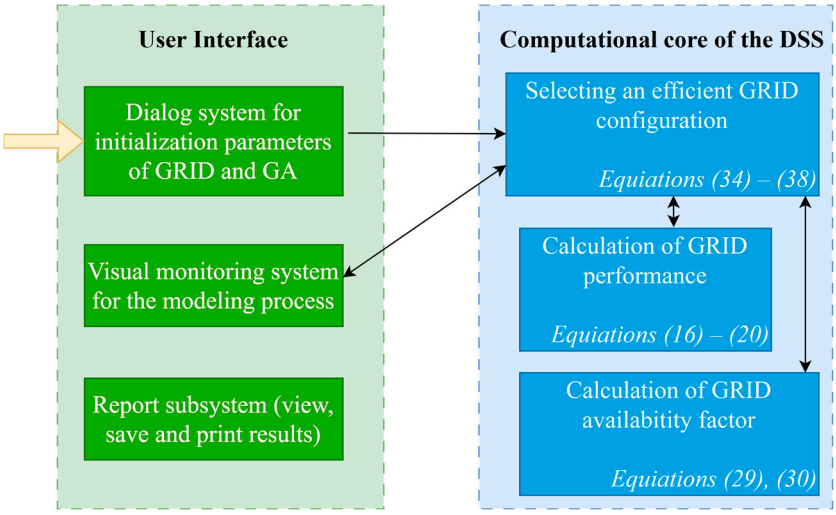

Figure 5. Decision support system structure diagram.

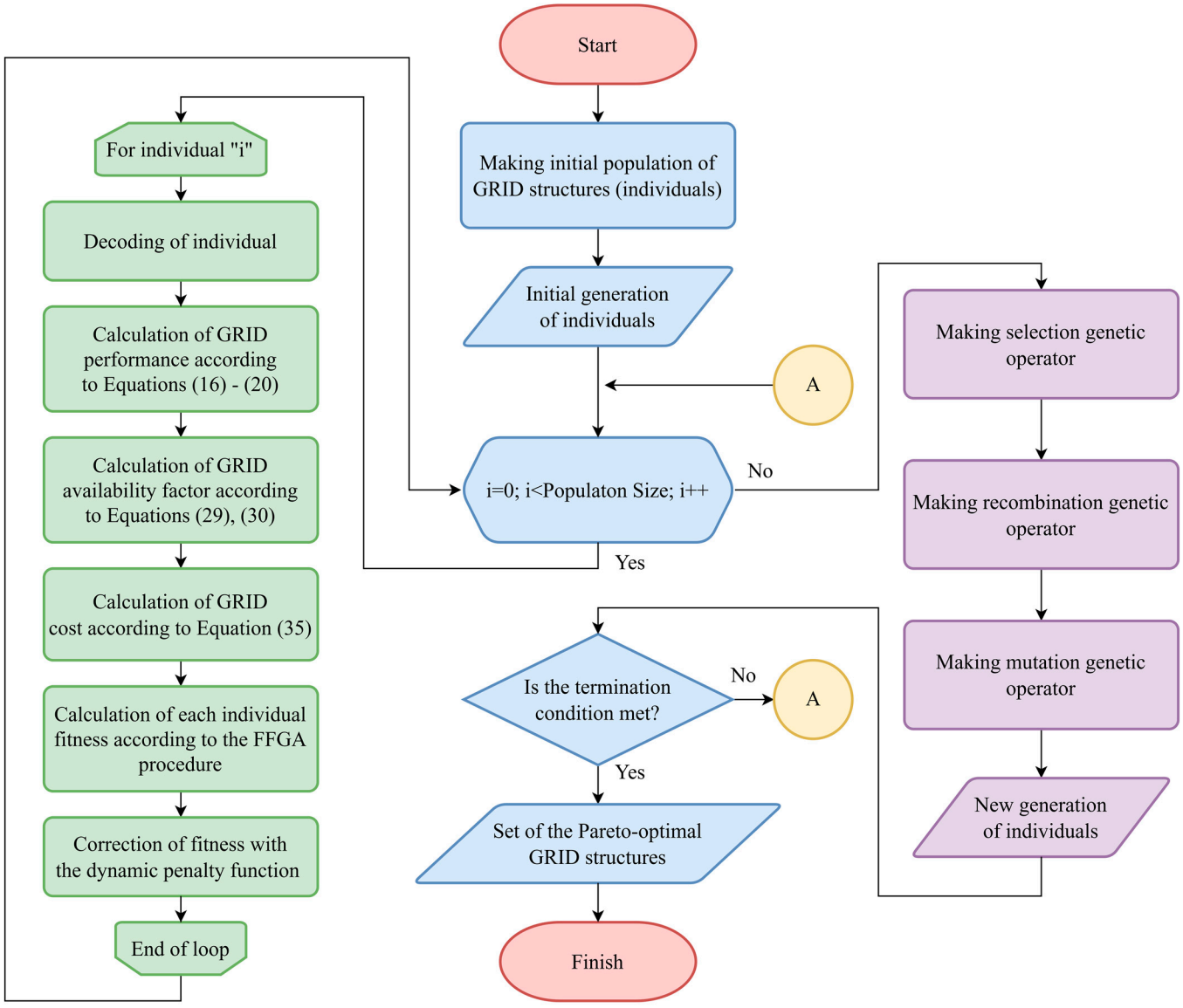

Figure 6. Block diagram of the developed computer program.

This program product is a full-fledged Windows application and is developed in the environment of the fast application development Embarcadero RAD Studio in C++ language.

The main window of the software system has five tabs:

- "Node performance";
- "Reliability of nodes";
- "Parameters of the problem to be solved";
- "Settings and launch of the GA";
- "Results".

The view of the window when selecting the "Node Performance" tab is shown in Figure 7.

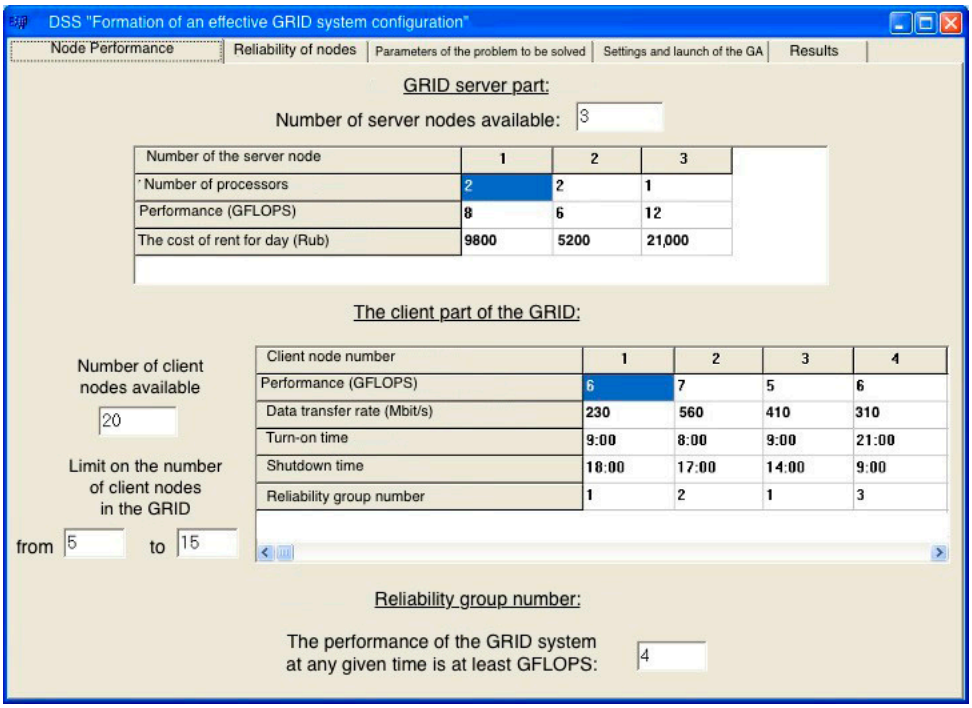

Figure 7. The main window of the DSS. "Node Performance" tab.

In this tab, the user is given the opportunity to configure the number as well as the hardware and cost characteristics of computing nodes available for designing a GRID system.

The characteristics of server nodes include the following:

- Number of processors;
- Single processor performance in GFLOPS;
- The cost of rent for 1 day.

The characteristics of client nodes include the following:

- Performance of client nodes in GFLOPS;
- Data link speed in Mbit/s;
- Node turn-on time;
- Node shutdown time;
- Reliability group number.

Also, on this tab, the user is given the opportunity to set the minimum performance value of the projected GRID system at any time.

The view of the window when selecting the Reliability of nodes tab is shown in Figure 8.

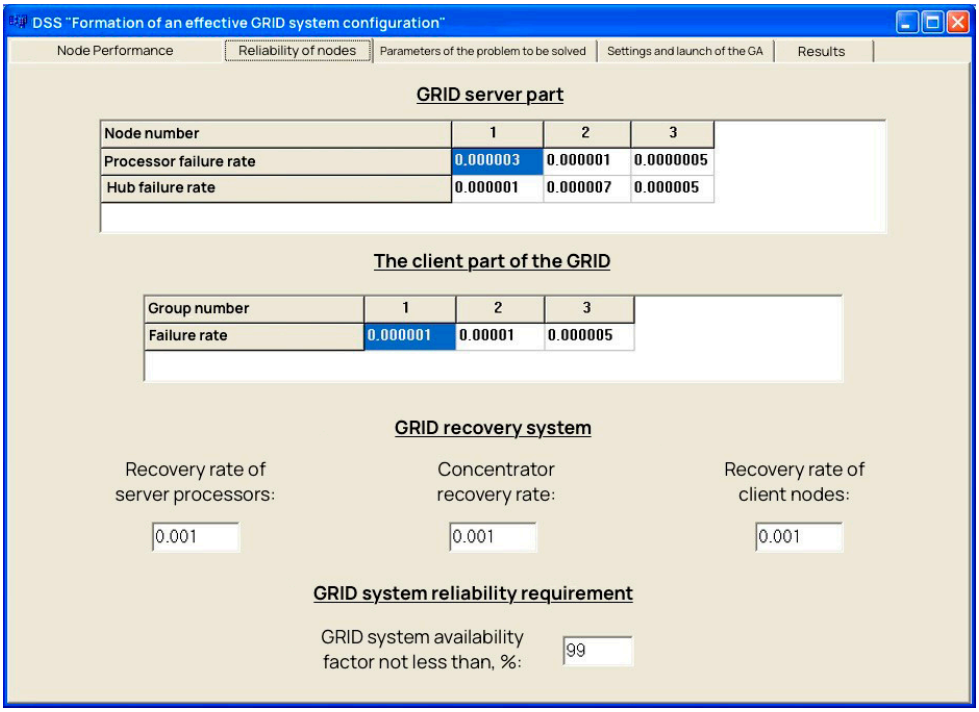

Figure 8. The main window of the DSS. The "Reliability of nodes" tab.

In this tab, the user is given the opportunity to configure the reliability characteristics of computing nodes available for designing a GRID system.

The characteristics of the server node include the following:

- Processor failure rate;
- The failure rate of the hub.

Characteristics of client nodes:

- The failure rate of the client nodes of each group.

Characteristics of the GRID recovery system:

- The recovery rate of the server processors;
- The intensity of the hub recovery;
- The recovery rate of client nodes of each type.

Also on this tab, the user is given the opportunity to set the minimum value of the readiness coefficient of the projected GRID system.

The view of the window when selecting the "Parameters of the problem to be solved" tab is shown in Figure 9.

In this window, the user is given the opportunity to configure the following parameters of the algorithm of the solved class of problems:

- Average computational complexity of one iteration of the algorithm;
- Average computational complexity of the control algorithm;
- The average amount of client-server exchange data.

The view of the window when selecting the "Settings and launch of the GA" tab is shown in Figure 10.

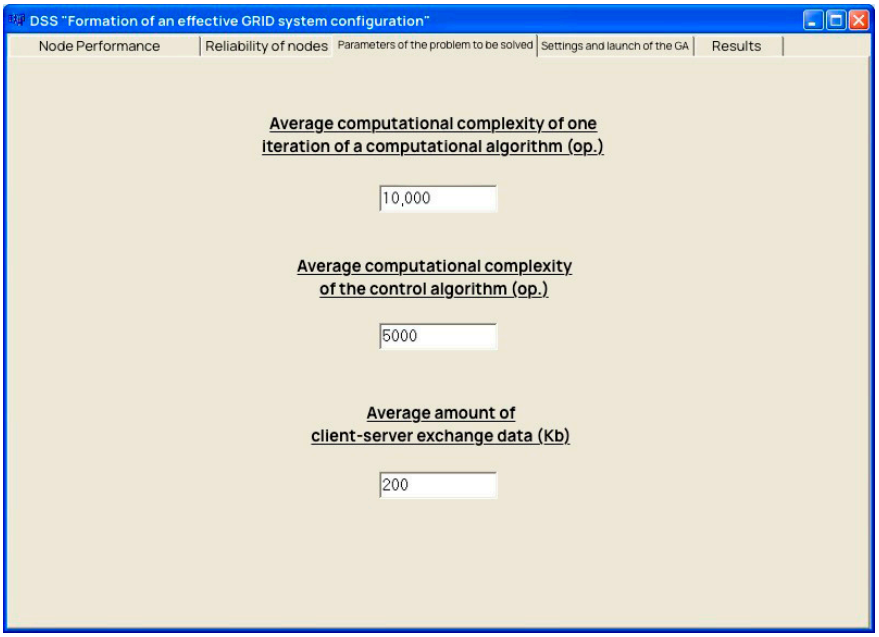

Figure 9. The main window of the DSS. The "Parameters of the problem to be solved" tab.

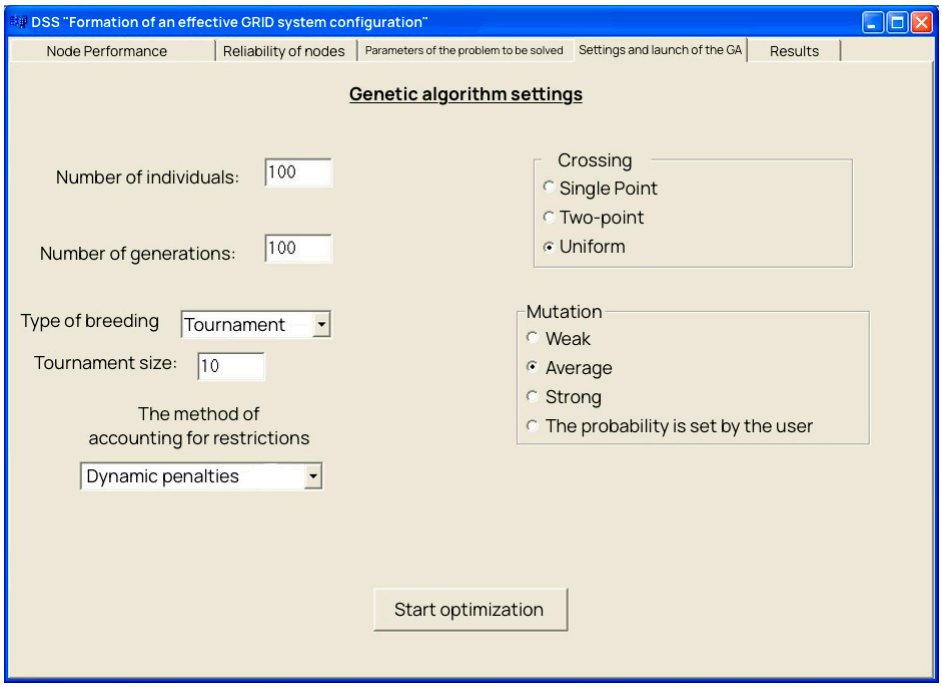

Figure 10. The main window of the DSS. The "Settings and launch of the GA" tab.

In this window, the user is given the opportunity to configure the following parameters of the genetic algorithm: the number of individuals in the population, the number of generations, type of selection, type of crossing, type of mutation, and the method of accounting for restrictions.

In the same window, the computational process is started, after which the user automatically enters the "Results" tab, which is shown in Figure 11.

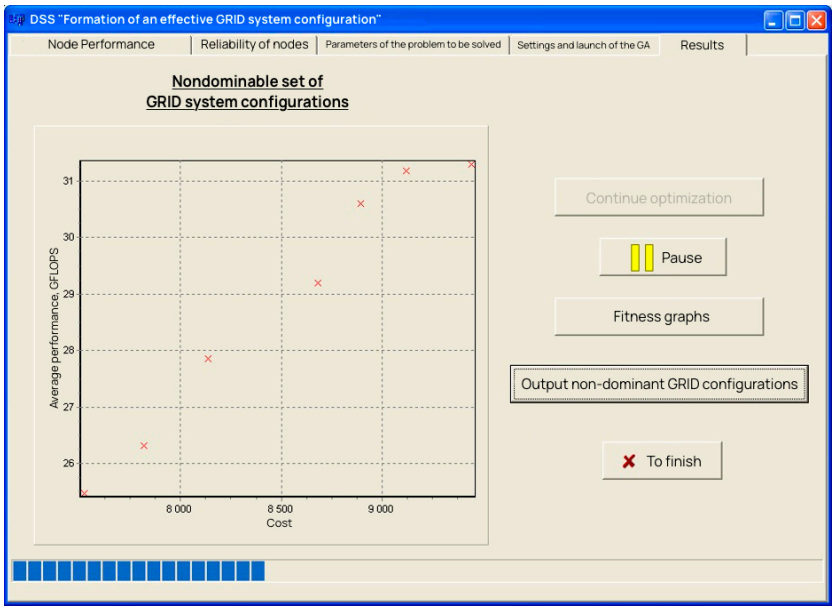

Figure 11. The main window of the DSS. The "Results" tab.

The process of solving an optimization problem is visualized by graphically depicting the current non-dominant set of solutions found. Each point on the graph corresponds to a specific configuration of the GRID system with certain performance and cost values that meet the availability factor limit.

During the execution of the program, when pressing the "Pause" button, the user is given the opportunity to go to the "Settings and launch of the GA" tab to change all GA parameters, except for the number of individuals in the population.

The decision support system "Formation of an effective GRID system configuration" has passed industry and state expertise and is registered in the industry fund of algorithms and programs [81].

The performance of the system for automating the design of artificial neural networks based on a multi-criteria multi-population parallel genetic algorithm was tested when solving the practical problem of neural networks predicting the structural-parametric synthesis of models to forecast the sales of goods in the pharmacy № 310 of LLC "Gubernskie Apteki" (Krasnoyarsk, Russia) [82,83].

5. Results and Discussion

During the course of the conducted research, the problem of choosing an effective configuration of a GRID system designed to solve complex scientific and technical problems of a certain class was considered.

The complex of analytical models developed with the application of the apparatus of mass service theory for evaluating the efficiency of GRID system functioning when solving complex scientific and technical problems of a certain class is proposed. The mathematical model of performance evaluation allows one to choose a performance-efficient configuration of the GRID system configured to solve complex problems of a certain class with a given limitation on the allowable solution time. The mathematical model of GRID system reliability indicators estimation allows the selection of the choice of its configuration to satisfy the main requirements for the reliability of the designed system. The developed models represent a mathematical apparatus, which allows one to formulate and solve optimization problems of choosing an effective configuration of the GRID system that is focused on solving problems of a certain class.

The software decision support system "Formation of an effective GRID system configuration" was developed on the basis of the proposed models of GRID system functioning to assess their performance and reliability and an evolutionary algorithm of multi-criteria

conditional optimization. This system was used at the Krasnoyarsk enterprise to solve the problem of selecting an effective structure of a centralized GRID system.

The distributed solved problem has the following average values for various parameters:

- Average computational complexity of one iteration of calculations on one branch of the algorithm for solving the problem–10,000 op.;
- Average computational complexity of the control algorithm of one branch of the problem solving algorithm–5000 op.;
- Average data volume of client–server exchange–5 GB.

There are 153 client nodes available for building a GRID system, the parameters of which vary in the following ranges:

- Performance from 3 to 12 GFLOPS;
- Data link speeds from 100 Mbit/s to 10 Gbit/s;
- Rental cost per 1 h of use from 100 rubles to 1000 rubles;
- According to reliability, all client nodes are divided into 2 groups with failure rates of 0.00001 and 0.000001.

The following restrictions are imposed on the designed GRID system:

- The number of client nodes of the GRID system should be no less than 15 and no more than 20;
- The GRID system availability factor is not less than 99.9%;
- The minimum performance of the GRID system is not less than 9 GFLOPS.

The constructed GRID system configuration has the following characteristics:

- Average performance–103.483 GFLOPS;
- Cost–500 rubles per day;
- Availability factor–99.92%;
- Minimum performance–51 GFLOPS.

The performance graph of the built GRID system configuration is shown in Figure 12.

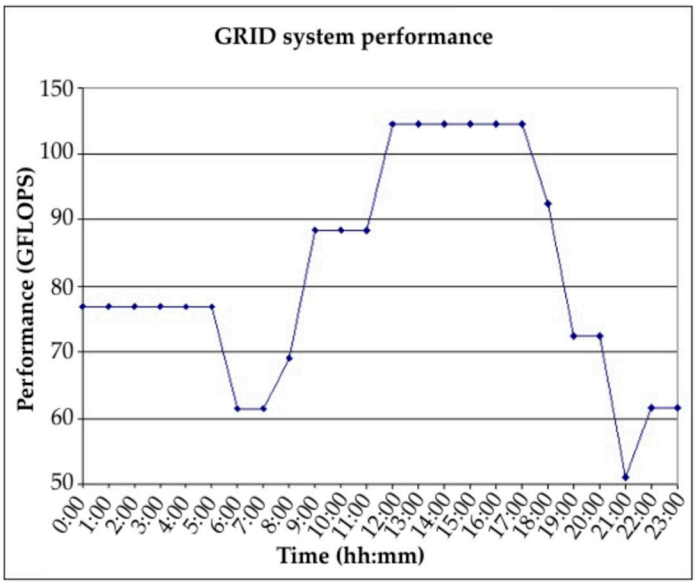

Figure 12. Performance graph of the built GRID system configuration.

When solving the problem, we obtained an approximation of the Pareto-multiplicity consisting of 6 points as well as its front. The approximation of the Pareto front is presented in Figure 13.

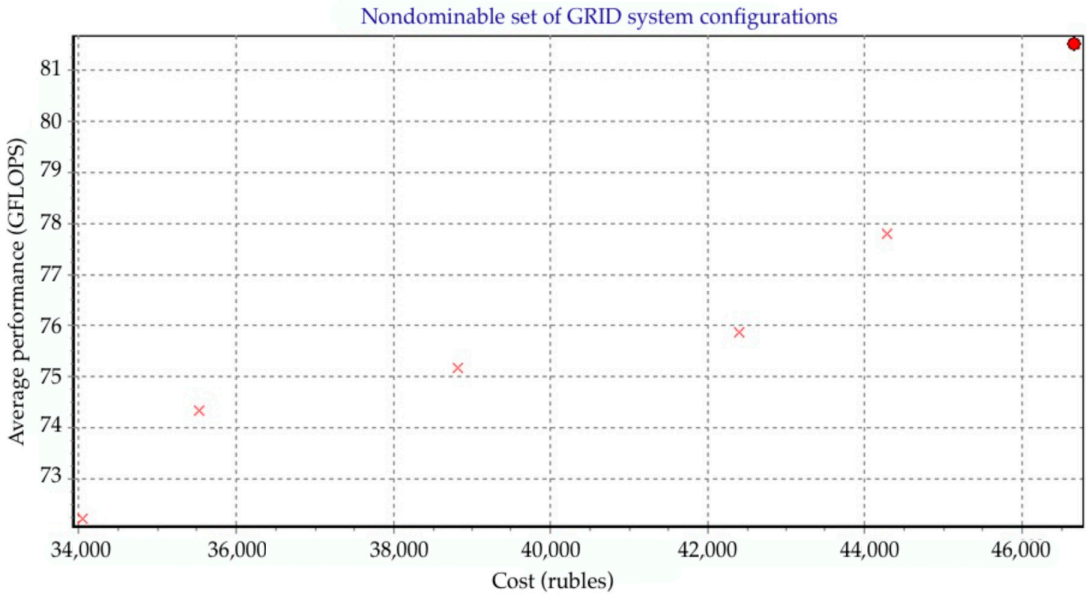

Figure 13. Nondominated set of GRID system configurations.

Thus, using the proposed approach, the configuration of a GRID system of the centralized type was selected, which allows complex distributed computing.

6. Conclusions

Here, the analysis of existing neural network modeling technologies was performed, revealing that the most suitable and proven method for solving the task at hand is a multicriteria multi-population parallel genetic algorithm that uses procedures for restructuring the topology of links between populations during the course of the solution.

The used models and algorithms allow effective solutions to the problems of structural-parametric synthesis of artificial neural networks for modeling complex objects and processes in distributed computing networks using GRID technology, and these models increase the validity of the choice of effective configuration of computing resources of GRID systems, which is essential for the theory and practice of system analysis and information processing.

The automated DSS "Formation of an effective GRID system configuration" was developed on the basis of the previously developed set of mathematical models for assessing the efficiency of GRID systems and was used to solve the problem of selecting an effective structure of a centralized GRID system configured to solve the problem of structural-parametric synthesis of neural network models. The built configuration of the GRID system has the following characteristics: average performance–103.483 GFLOPS, cost–500 rubles per day, availability factor–99.92%, and minimum performance–51 GFLOPS. Approbation of this software tool has shown that the developed software can be used to successfully solve the problems of designing systems based on GRID technology, which is designed to solve complex scientific and technical problems.

However, along with the advantages of the approach proposed in this study, it also has limitations. In particular, when the characteristics of a resource-intensive problem solved on the GRID infrastructure change, it may be necessary to rebuild the structure of the GRID system, which necessitates additional optimization. Moreover, two options for solving this problem can be considered, including the design of a GRID system "from scratch" or carrying out optimization of the GRID system based on the already formed effective configuration. In addition, another limitation of the proposed approach is the possible change in both the computing potential of available geographically distributed resources and their failure without the possibility of recovery. In this case, reconfiguration of the GRID system will also require additional calculations during optimization.

In the course of further research, the developed automated decision support system will be integrated into various state and commercial enterprises, which will improve the efficiency of solving the distributed problem of structural and parametric synthesis of neural network models of complex systems based on GRID technology.

Author Contributions: Conceptualization, V.V.T., V.S.T. and V.A.N.; Data curation, V.V.B., S.O.K., A.P.G. and V.V.K.; Formal analysis, V.V.T. and A.S.B.; Investigation, V.V.T., V.A.N., A.S.B. and S.O.K.; Methodology, V.V.T., V.S.T., V.V.B., A.S.B. and V.V.K.; Project administration, V.S.T., V.A.N. and A.S.B.; Resources, V.A.N. and V.V.K.; Software, V.V.T., V.S.T., V.A.N., V.V.B., S.O.K. and A.P.G.; Supervision, V.S.T.; Validation, V.V.T., V.A.N., V.V.B., A.S.B., S.O.K. and A.P.G.; Visualization, V.S.T., V.V.B., S.O.K. and A.P.G.; Writing—original draft, V.V.T., V.S.T., V.A.N., V.V.B., A.S.B., S.O.K., A.P.G. and V.V.K.; Writing—review & editing, V.V.T., V.S.T., V.A.N., V.V.B., A.S.B., S.O.K., A.P.G. and V.V.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Restrictions apply to the availability of these data. Data was obtained from LLC "Gubernskie Apteki" (Krasnoyarsk, Russia) and are available from the authors with the permission of LLC "Gubernskie Apteki" (Krasnoyarsk, Russia).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Halbouni, A.; Gunawan, T.S.; Habaebi, M.H.; Halbouni, M.; Kartiwi, M.; Ahmad, R. Machine Learning and Deep Learning Approaches for CyberSecurity: A Review. *IEEE Access* **2022**, *10*, 19572–19585. [CrossRef]
2. Dhall, D.; Kaur, R.; Juneja, M. Machine Learning: A Review of the Algorithms and Its Applications. *Lect. Notes Electr. Eng.* **2020**, *597*, 47–63.
3. D'souza, R. Optimizing Utilization Forecasting with Artificial Intelligence and Machine Learning. Available online: https://www.datanami.com/2020/ (accessed on 19 December 2023).
4. Bukhtoyarov, V.; Tynchenko, V.; Nelyub, V.; Borodulin, A.; Gantimurov, A. Classification of Technical Condition of Pumping Units Using Intelligent Fault Classification. *Mathematics* 2024; *in press*.
5. Lu, S.; Chai, H.; Sahoo, A.; Phung, B.T. Condition Monitoring Based on Partial Discharge Diagnostics Using Machine Learning Methods: A Comprehensive State-of-the-Art Review. *IEEE Trans. Dielectr. Electr. Insul.* **2020**, *27*, 1861–1888. [CrossRef]
6. Myklestad, N.O. *Fundamentals of Vibration Analysis*; Dover Publications: New York, NY, USA, 2018; p. 259.
7. Malashin, I.; Tynchenko, V.; Nelyub, V.; Borodulin, A.; Gantimurov, A. Estimation and Prediction of the Polymers' Physical Characteristics Using the Machine Learning Models. *Polymers* **2024**, *16*, 115. [CrossRef]
8. Masich, I.S.; Tynchenko, V.S.; Nelyub, V.A.; Bukhtoyarov, V.V.; Kurashkin, S.O.; Gantimurov, A.P.; Borodulin, A.S. Prediction of Critical Filling of a Storage Area Network by Machine Learning Methods. *Electronics* **2022**, *11*, 4150. [CrossRef]
9. Masich, I.S.; Tyncheko, V.S.; Nelyub, V.A.; Bukhtoyarov, V.V.; Kurashkin, S.O.; Borodulin, A.S. Paired Patterns in Logical Analysis of Data for Decision Support in Recognition. *Computation* **2022**, *10*, 185. [CrossRef]
10. Mikhalev, A.S.; Tynchenko, V.S.; Nelyub, V.A.; Lugovaya, N.M.; Baranov, V.A.; Kukartsev, V.V.; Sergienko, R.B.; Kurashkin, S.O. The Orb-Weaving Spider Algorithm for Training of Recurrent Neural Networks. *Symmetry* **2022**, *14*, 2036. [CrossRef]
11. Tynchenko, V.S.; Kurashkin, S.O.; Tynchenko, V.V.; Bukhtoyarov, V.V.; Kukartsev, V.V.; Sergienko, R.B.; Tynchenko, S.V.; Bashmur, K.A. Software to Predict the Process Parameters of Electron Beam Welding. *IEEE Access* **2021**, *9*, 92483–92499. [CrossRef]
12. Doerr, B.; Le, H.P.; Makhmara, R.; Nguyen, T.D. Fast Genetic Algorithms. In Proceedings of the Genetic and Evolutionary Computation Conference, New York, NY, USA, 15 July 2017.
13. Liu, Y.Y.; Wang, S. A Scalable Parallel Genetic Algorithm for the Generalized Assignment Problem. *Parallel Comput.* **2015**, *46*, 98–119. [CrossRef]
14. Matveev, Y.N.; Stukalova, N.A.; Stukalov, D.O. Some approaches to the creation of grid systems. In Proceedings of the IX International Scientific and Practical Conference, Penza, Russia, 29–30 April 2019.
15. Cody, E.; Sharman, R.; Rao, R.H.; Upadhyaya, S. Security in grid computing: A review and synthesis. *Decis. Support Syst.* **2008**, *44*, 749–764. [CrossRef]
16. Amalarethinam, D.I.G.; Muthulakshmi, P. An Overview of the scheduling policies and algorithms in Grid Computing. *Int. J. Res. Rev. Comput. Sci.* **2011**, *2*, 280.
17. Shah, S.N.M.; Mahmood, A.K.B.; Oxley, A. Dynamic multilevel hybrid scheduling algorithms for grid computing. *Procedia Comput. Sci.* **2011**, *4*, 402–411. [CrossRef]
18. Mehmood Shah, S.N.; Mahmood, A.K.B.; Oxley, A. Analysis and evaluation of grid scheduling algorithms using real workload traces. In Proceedings of the International Conference on Management of Emergent Digital EcoSystems (MEDES'10), Bangkok, Thailand, 26 October 2010. [CrossRef]

19. Mateescu, G.; Gentzsch, W.; Ribbens, C.J. Hybrid computing—Where HPC meets grid and cloud computing. *Future Gener. Comput. Syst.* **2011**, *27*, 440–453. [CrossRef]
20. Patni, J.C.; Aswal, M.S. Distributed load balancing model for grid computing environment. In Proceedings of the 2015 1st International Conference on Next Generation Computing Technologies (NGCT), Dehradun, India, 4–5 September 2015. [CrossRef]
21. Raj, J.S.; Fiona, R. Load balancing techniques in grid environment: A survey. In Proceedings of the 2013 International Conference on Computer Communication and Informatics, Coimbatore, India, 4–6 January 2013. [CrossRef]
22. Patni, J.C.; Aswal, M.S. Distributed approach of load balancing in dynamic grid computing environment. *Int. J. Commun. Netw. Distrib. Syst.* **2017**, *19*, 1–18. [CrossRef]
23. Babaeian, M.; Sereshki, F.; Ataei, M.; Nehring, M.; Mohammadi, S. Application of Soft Computing, Statistical and Multi-Criteria Decision-Making Methods to Develop a Predictive Equation for Prediction of Flyrock Distance in Open-Pit Mining. *Mining* **2023**, *3*, 304–333. [CrossRef]
24. Ali, H.M.A.; Abdi, M. Multi-Objective Parametric Shape Optimisation of Body-Centred Cubic Lattice Structures for Additive Manufacturing. *J. Manuf. Mater. Process.* **2023**, *7*, 156. [CrossRef]
25. Pedroso, D.M.; Bonyadi, M.R.; Gallagher, M. Parallel Evolutionary Algorithm for Single and Multi-Objective Optimisation: Differential Evolution and Constraints Handling. *Appl. Soft Comput.* **2017**, *61*, 995–1012. [CrossRef]
26. Hassanat, A.; Almohammadi, K.; Alkafaween, E.; Abunawas, E.; Hammouri, A.; Prasath, V.B.S. Choosing Mutation and Crossover Ratios for Genetic Algorithms—A Review with a New Dynamic Approach. *Information* **2019**, *10*, 390. [CrossRef]
27. Kramer, O. *Genetic Algorithm Essentials*; Springer: Cham, Switzerland, 2017; p. 92.
28. Skorpil, V.; Oujezsky, V. Parallel Genetic Algorithms' Implementation Using a Scalable Concurrent Operation in Python. *Sensors* **2022**, *22*, 2389. [CrossRef]
29. Canali, C.; Lancellotti, R. GASP: Genetic Algorithms for Service Placement in Fog Computing Systems. *Algorithms* **2019**, *12*, 201. [CrossRef]
30. Angelova, M.; Roeva, O.; Vassilev, P.; Pencheva, T. Multi-Population Genetic Algorithm and Cuckoo Search Hybrid Technique for Parameter Identification of Fermentation Process Models. *Processes* **2023**, *11*, 427. [CrossRef]
31. Petrosov, D.A.; Lomazov, V.A.; Petrosova, N.V. Model of an Artificial Neural Network for Solving the Problem of Controlling a Genetic Algorithm Using the Mathematical Apparatus of the Theory of Petri Nets. *Appl. Sci.* **2021**, *11*, 3899. [CrossRef]
32. Kotyrba, M.; Volna, E.; Habiballa, H.; Czyz, J. The Influence of Genetic Algorithms on Learning Possibilities of Artificial Neural Networks. *Computers* **2022**, *11*, 70. [CrossRef]
33. Albadr, M.A.; Tiun, S.; Ayob, M.; AL-Dhief, F. Genetic Algorithm Based on Natural Selection Theory for Optimization Problems. *Symmetry* **2020**, *12*, 1758. [CrossRef]
34. Zhang, J.; Wang, M. Special Issue: Neural Networks, Fuzzy Systems and Other Computational Intelligence Techniques for Advanced Process Control. *Processes* **2023**, *11*, 2278. [CrossRef]
35. Abdolrasol, M.G.M.; Hussain, S.M.S.; Ustun, T.S.; Sarker, M.R.; Hannan, M.A.; Mohamed, R.; Ali, J.A.; Mekhilef, S.; Milad, A. Artificial Neural Networks Based Optimization Techniques: A Review. *Electronics* **2021**, *10*, 2689. [CrossRef]
36. Behabtu, H.A.; Messagie, M.; Coosemans, T.; Berecibar, M.; Anlay Fante, K.; Kebede, A.A.; Mierlo, J.V. A Review of Energy Storage Technologies' Application Potentials in Renewable Energy Sources Grid Integration. *Sustainability* **2020**, *12*, 10511. [CrossRef]
37. Liao, X.; Xiao, L.; Yang, C.; Lu, Y. MilkyWay-2 Supercomputer: System and Application. *Front. Comput. Sci.* **2014**, *8*, 345–356. [CrossRef]
38. Stanzione, D.; Barth, B.; Gaffney, N.; Gaither, K.; Hempel, C.; Minyard, T.; Mehringer, S.; Wernert, E.; Tufo, H.; Panda, D.; et al. Stampede 2. In Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact, New York, NY, USA, 9 July 2017.
39. Stewart, C.A.; Hancock, D.Y.; Vaughn, M.; Fischer, J.; Cockerill, T.; Liming, L.; Merchant, N.; Miller, T.; Love, J.M.; Stanzione, D.C.; et al. Jetstream: Performance, early experiences, and early results. In Proceedings of the XSEDE16 Conference on Diversity, Big Data, and Science at Scale, Miami, FL, USA, 17–21 July 2016.
40. Hancock, D.Y.; Stewart, C.A.; Vaughn, M.; Fischer, J.; Lowe, J.M.; Turner, G.; Swetnam, T.L.; Chafin, T.K.; Afgan, E.; Pierce, M.E.; et al. Jetstream—Early operations performance, adoption, and impacts. *Concurr. Comput. Pract. Exp.* **2019**, *31*, e4683. [CrossRef]
41. Ryu, S.; Noh, J.; Kim, H. Deep Neural Network Based Demand Side Short Term Load Forecasting. *Energies* **2017**, *10*, 3. [CrossRef]
42. Passian, A.; Imam, N. Nanosystems, Edge Computing, and the Next Generation Computing Systems. *Sensors* **2019**, *19*, 4048. [CrossRef] [PubMed]
43. Merelli, E.; Rucco, M.; Sloot, P.; Tesei, L. Topological Characterization of Complex Systems: Using Persistent Entropy. *Entropy* **2015**, *17*, 6872–6892. [CrossRef]
44. Goudarzi, A.; Ghayoor, F.; Waseem, M.; Fahad, S.; Traore, I. A Survey on IoT-Enabled Smart Grids: Emerging, Applications, Challenges, and Outlook. *Energies* **2022**, *15*, 6984. [CrossRef]
45. Henriques, J.; Caldeira, F.; Cruz, T.; Simões, P. Combining K-Means and XGBoost Models for Anomaly Detection Using Log Datasets. *Electronics* **2020**, *9*, 1164. [CrossRef]
46. Rodriguez, D.; Gomez, D.; Alvarez, D.; Rivera, S. A Review of Parallel Heterogeneous Computing Algorithms in Power Systems. *Algorithms* **2021**, *14*, 275. [CrossRef]
47. Mittal, S. A Survey of ReRAM-Based Architectures for Processing-In-Memory and Neural Networks. *Mach. Learn. Knowl. Extr.* **2019**, *1*, 75–114. [CrossRef]

48. Orgerie, A.-C.; de Assuncao, M.D.; Lefevre, L. A Survey on Techniques for Improving the Energy Efficiency of Large-Scale Distributed Systems. *ACM Comput. Surv.* **2014**, *46*, 1–31. [CrossRef]
49. Ferretti, I.; Camparada, M.; Zavanella, L.E. Queuing Theory-Based Design Methods for the Definition of Power Requirements in Manufacturing Systems. *Energies* **2022**, *15*, 7621. [CrossRef]
50. Liu, F.; Tang, G.; Li, Y.; Cai, Z.; Zhang, X.; Zhou, T. A Survey on Edge Computing Systems and Tools. *Proc. IEEE* **2019**, *107*, 1537–1562. [CrossRef]
51. Bradley, J.M.; Atkins, E.M. Optimization and Control of Cyber-Physical Vehicle Systems. *Sensors* **2015**, *15*, 23020–23049. [CrossRef]
52. Matsoukas, T. Combinatorics and Statistical Mechanics of Integer Partitions. *Entropy* **2023**, *25*, 385. [CrossRef] [PubMed]
53. Feitelson, D.G. *Workload Modeling for Computer Systems Performance Evaluation*; Cambridge University Press: New York, NY, USA, 2015; p. 541.
54. Aparicio, M.; Bacao, F.; Oliveira, T. An E-Learning Theoretical Framework. *J. Educ. Technol. Soc.* **2016**, *19*, 292–307.
55. Elmouatamid, A.; Ouladsine, R.; Bakhouya, M.; El Kamoun, N.; Khaidar, M.; Zine-Dine, K. Review of Control and Energy Management Approaches in Micro-Grid Systems. *Energies* **2021**, *14*, 168. [CrossRef]
56. Lebedev, V.A.; Terskov, V.A. *Modeling and Optimization of Multiprocessor Operational Control Systems*, 3rd ed.; MAKS Press: Moscow, Russia, 2018; 330p.
57. Alkhanak, E.N.; Lee, S.P.; Khan, S.U.R. Cost-Aware Challenges for Workflow Scheduling Approaches in Cloud Computing Environments: Taxonomy and Opportunities. *Future Gener. Comput. Syst.* **2015**, *50*, 3–21. [CrossRef]
58. Armenta, M.; Jodoin, P.-M. The Representation Theory of Neural Networks. *Mathematics* **2021**, *9*, 3216. [CrossRef]
59. Chiroma, H.; Noor, A.S.M.; Abdulkareem, S.; Abubakar, A.I.; Hermawan, A.; Qin, H.; Hamza, M.F.; Herawan, T. Neural Networks Optimization through Genetic Algorithm Searches: A Review. *Appl. Math. Inf. Sci.* **2017**, *11*, 1543–1564. [CrossRef]
60. Bao, Z.; Erdős, L.; Schnelli, K. On the Support of the Free Additive Convolution. *J. d'Anal. Math.* **2020**, *142*, 323–348. [CrossRef]
61. Liu, L.; Mu, H.; Yang, J. Generic Constraints Handling Techniques in Constrained Multi-Criteria Optimization and Its Application. *Eur. J. Oper. Res.* **2015**, *244*, 576–591. [CrossRef]
62. Lukic, D.; Cep, R.; Vukman, J.; Antic, A.; Djurdjev, M.; Milosevic, M. Multi-Criteria Selection of the Optimal Parameters for High-Speed Machining of Aluminum Alloy Al7075 Thin-Walled Parts. *Metals* **2020**, *10*, 1570. [CrossRef]
63. Ceschia, A.; Azib, T.; Bethoux, O.; Alves, F. Multi-Criteria Optimal Design for FUEL Cell Hybrid Power Sources. *Energies* **2022**, *15*, 3364. [CrossRef]
64. Yang, M.-D.; Chen, Y.-P.; Lin, Y.-H.; Ho, Y.-F.; Lin, J.-Y. Multiobjective Optimization Using Nondominated Sorting Genetic Algorithm-II for Allocation of Energy Conservation and Renewable Energy Facilities in a Campus. *Energy Build.* **2016**, *122*, 120–130. [CrossRef]
65. Schaffer, J.D. Multiple objective optimization with vector evaluated genetic algorithms. In Proceedings of the First International Conference on Genetic Algorithms and Their Applications, New York, NY, USA, 2 December 2013.
66. Fourman, M.P. Compaction of symbolic layout using genetic algorithms. In Proceedings of the First International Conference on Genetic Algorithms and Their Applications, New York, NY, USA, 2 December 2013.
67. Wątróbski, J.; Jankowski, J.; Ziemba, P.; Karczmarczyk, A.; Zioło, M. Generalised Framework for Multi-Criteria Method Selection. *Omega* **2019**, *86*, 107–124. [CrossRef]
68. Kim, K.; Walewski, J.; Cho, Y.K. Multiobjective Construction Schedule Optimization Using Modified Niched Pareto Genetic Algorithm. *J. Manag. Eng.* **2016**, *32*, 04015038. [CrossRef]
69. Palakonda, V.; Mallipeddi, R. Pareto Dominance-Based Algorithms with Ranking Methods for Many-Objective Optimization. *IEEE Access* **2017**, *5*, 11043–11053. [CrossRef]
70. Fan, Z.; Fang, Y.; Li, W.; Lu, J.; Cai, X.; Wei, C. Comparative Study of Constrained Multi-Objective Evolutionary Algorithms on Constrained Multi-Objective Optimization Problems. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, Spain, 5 June 2017.
71. Nastasi, G.; Colla, V.; Cateni, S.; Campigli, S. Implementation and Comparison of Algorithms for Multi-Objective Optimization Based on Genetic Algorithms Applied to the Management of an Automated Warehouse. *J. Intell. Manuf.* **2018**, *29*, 1545–1557. [CrossRef]
72. Janmaijaya, M.; Shukla, A.K.; Abraham, A.; Muhuri, P.K. A Scientometric Study of Neurocomputing Publications (1992–2018): An Aerial Overview of Intrinsic Structure. *Publications* **2018**, *6*, 32. [CrossRef]
73. Leijnen, S.; Veen, F.v. The Neural Network Zoo. *Proceedings* **2020**, *47*, 9. [CrossRef]
74. Chilimbi, T.; Suzue, Y.; Apacible, J.; Kalyanaraman, K. Project Adam: Building an Efficient and Scalable Deep Learning Training System. In Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), Broomfield, FL, USA, 6 October 2014.
75. Rece, L.; Vlase, S.; Ciuiu, D.; Neculoiu, G.; Mocanu, S.; Modrea, A. Queueing Theory-Based Mathematical Models Applied to Enterprise Organization and Industrial Production Optimization. *Mathematics* **2022**, *10*, 2520. [CrossRef]
76. Plauska, I.; Liutkevičius, A.; Janavičiūtė, A. Performance Evaluation of C/C++, MicroPython, Rust and TinyGo Programming Languages on ESP32 Microcontroller. *Electronics* **2023**, *12*, 143. [CrossRef]
77. Lee, S.; Kim, J.; Kang, H.; Kang, D.-Y.; Park, J. Genetic Algorithm Based Deep Learning Neural Network Structure and Hyperparameter Optimization. *Appl. Sci.* **2021**, *11*, 744. [CrossRef]

78. Ren, P.; Xiao, Y.; Chang, X.; Huang, P.; Li, Z.; Chen, X.; Wang, X. A Comprehensive Survey of Neural Architecture Search. *ACM Comput. Surv.* **2022**, *54*, 1–34. [CrossRef]
79. Fonseca, C.M.; Fleming, P.J. Genetic algorithms for multiobjective optimization: Formulation discussion and generalization. *ICGA* **1993**, *93*, 416–423.
80. Si, C.; Shen, J.; Zou, X.; Duo, Y.; Wang, L.; Wu, Q. A Dynamic Penalty Function for Constrained Optimization. *Lect. Notes Comput. Sci.* **2015**, *9141*, 261–272. [CrossRef]
81. Ahmadizar, F.; Soltanian, K.; AkhlaghianTab, F.; Tsoulos, I. Artificial Neural Network Development by Means of a Novel Combination of Grammatical Evolution and Genetic Algorithm. *Eng. Appl. Artif. Intell.* **2015**, *39*, 1–13. [CrossRef]
82. Rani, S.; Suri, B.; Goyal, R. On the Effectiveness of Using Elitist Genetic Algorithm in Mutation Testing. *Symmetry* **2019**, *11*, 1145. [CrossRef]
83. Torres-Salinas, H.; Rodríguez-Reséndiz, J.; Cruz-Miguel, E.E.; Ángeles-Hurtado, L.A. Fuzzy Logic and Genetic-Based Algorithm for a Servo Control System. *Micromachines* **2022**, *13*, 586. [CrossRef]