applied sciences

MDPI

Article

Synthesis of a Moth and Flame Algorithm for Incorporation into the Architecture of Deceptive Systems with Baits and Traps

Oleg Savenko [1], Bohdan Rusyn [2,3], Sergii Lysenko [1,*], Tomasz Ciszewski [3], Bohdan Savenko [1], Andrii Drozd [1], Andrii Nicheporuk [1] and Anatoliy Sachenko [3,4]

[1] Department of Computer Engineering and Information Systems, Information Technologies Faculty, Khmelnytskyi National University, 29016 Khmelnytskyi, Ukraine; savenko_oleg_st@ukr.net (O.S.); savenko_bohdan@ukr.net (B.S.); andriydrozdit@gmail.com (A.D.); andrey.nicheporuk@gmail.com (A.N.)
[2] Department of Information Technologies of Remote Sensing, Karpenko Physico-Mechanical Institute of NAS of Ukraine, 79601 Lviv, Ukraine; rusyn@ipm.lviv.ua
[3] Department of Informatics and Teleinformatics, Kazimierz Pulaski University of Radom, 26-600 Radom, Poland; t.ciszewski@urad.edu.pl (T.C.); as@wunu.edu.ua (A.S.)
[4] Research Institute for Intelligent Computer Systems, West Ukrainian National University, 46009 Ternopil, Ukraine
* Correspondence: sirogyk@ukr.net

Abstract

This paper proposes a novel method for synthesizing a discrete optimization algorithm based on the moth–flame paradigm for application to the architecture of deceptive systems incorporating decoys and traps. Unlike existing approaches that primarily rely on continuous search spaces or static deception strategies, the proposed method enables the formation of a discrete search space with a coordinate-based representation of deception objects and system states. A spiral search trajectory is synthesized by modeling the dynamic interaction between moths and flames, which allows the algorithm to balance exploration and exploitation effectively and to mitigate premature convergence to local optima. The problem of selecting subsequent operational steps of a deceptive system, which includes the control and reconfiguration of decoys and traps in response to detected events, is formulated as a discrete optimization problem. The objective of this optimization is to increase the effectiveness of cyberattack and malware detection in corporate network environments. The decision variables include the sequence of deception actions, process models, and architectural characteristics of the system, while the constraints are defined by the operational conditions, resource limitations, and structural features of corporate networks. The proposed method supports the identification of an optimal sequence of deception actions under dynamically changing conditions and provides mechanisms for operational adaptation to attacker behavior in real time. This adaptability enables the creation of deceptive systems capable of long-term autonomous operation without continuous administrative intervention, while simultaneously increasing their resistance to adversarial reconnaissance and reverse engineering of their operational principles. The experimental results confirm the feasibility and effectiveness of the proposed approach and demonstrate the potential of integrating population-based optimization algorithms into deceptive system architectures. Comparative analysis shows that the proposed method outperforms its closest competitor, the genetic algorithm, achieving an improvement of 4.82% in terms of the objective function value. Future research directions include deeper integration of population-based optimization methods into decoy-and-trap architectures and the development of a comprehensive framework for organizing their operation in accordance with the proposed conceptual model. Overall, the results contribute to enhancing the cyber-resilience of corporate networks through intelligent, adaptive, and autonomous systems for countering modern cyberattacks and malware.

Check for updates

Academic Editor: Xianpeng Wang

Received: 20 January 2026
Revised: 6 February 2026
Accepted: 27 February 2026
Published: 2 March 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Keywords: corporate networks; computer stations; deceptive systems; population algorithms; moth and flame algorithm; discrete optimization; trap; decoy; malicious software; cyberattacks; systems architecture

1. Introduction

A promising direction for ensuring the security and protection of corporate networks is the use of deceptive systems, decoys, and traps [1,2]. Such solutions are expensive, and decision-making based on them is associated with specific needs for ensuring more reliable security and protection in corporate networks [3,4]. Attackers can adapt to such systems over time, and therefore, for their long-term operation, it is necessary that they have the ability to function for a long time; that is, to fall into a state of convergence and make decisions independently without the involvement of administrators [5,6]. The choice of the next steps [1,3] in deceptive systems, which include the management of decoys and traps, as well as their changes or adjustments during the execution of certain sequences of steps, corresponds to formulation of the optimization problem, the essence of which is to choose the best (optimal) option from a large number of possible ones. The optimal value will be the best possible option that best meets certain conditions, requirements, or tasks, providing the most profitable, most effective, or most suitable result.

The optimization problem of choosing the next steps of deceptive systems with decoys and traps in corporate networks will be formulated according to the achievement of increasing the efficiency of detecting cyberattacks and malware as an optimality criterion, the variability of steps of deceptive systems as variable system parameters, the process models and architecture of deceptive systems in corporate networks, the restrictions on the operating environment of corporate networks, and the features of deceptive systems with decoys and traps. To solve this problem, it is necessary to choose a method that would ensure finding the optimal value, that is, the sequence of system steps in the changing environment of corporate networks during cyberattacks or malware actions and with the possibility of their current change, associated with the need to adjust the response of the system to various influences.

Therefore, an urgent scientific task is to synthesize such optimization methods for deceptive systems that would ensure their long-term functioning without the involvement of administrators and reactions to events caused by the effects of cyberattacks or malicious software actions (MSAs), in such a way that the principles of system functioning are difficult to understand for attackers.

2. Related Work

The selection of next steps in deceptive systems thatinclude the control of decoys and traps, and their changes or adjustments during the execution of certain sequences of steps, corresponds to the formulation of the optimization problem, the essence of which is to select the best (optimal) option from a large number of possible ones. The optimal value will be the best of the possible options that best meets certain conditions, requirements, or tasks, providing the most profitable, most effective, or most suitable result.

Optimization refers to the process of finding the best possible solution or solutions for a particular problem [7]. As the complexity of problems has increased over the past few decades, the need for new optimization methods has become more obvious than before. Before the advent of heuristic optimization methods, mathematical optimization methods were used, which are mainly deterministic and, therefore, suffer from the problem of convergence of the search in local optima. With the advent of methods based on the use

of population algorithms, optimization problems can be solved with the achievement of optimal or suboptimal solutions.

Population algorithms are a class of heuristic optimization methods that are aimed at a set (population) of solutions. They imitate natural processes. Population algorithms are mainly classified into three main categories depending on the sources of inspiration: evolutionary; physical; and swarm. Evolutionary algorithms imitate evolutionary processes in nature (e.g., genetic algorithms, the evolutionary membrane algorithm). The number of swarm-based algorithms (e.g., bee algorithm, gray wolf optimizer) is greater than that of evolutionary algorithms. The third class of algorithms is inspired by physical phenomena in nature (e.g., the gravity search algorithm, optimization of chemical reactions). Population algorithms with different sources of inspiration are also known, including inanimate nature and animate nature. It is difficult to limit oneself to a few population algorithms when solving optimization problems, because there is no one population algorithm or collection of population algorithms that can solve all optimization problems. Therefore, let us consider common population algorithms used to solve optimization problems.

In Ref. [8], an approach to setting and solving the problem of optimal selection of product quality indicators in "producerconsumer" systems is proposed based on fuzzy cognitive maps and a genetic algorithm, taking into account the interests of both the producer and the consumer. The optimization criterion is the maximum approximation between the attractiveness of the product and the desire to purchase it. The controlled variables are the levels of the producer and consumer indicators. The constraints are agreements on the required levels of indicators that are common to the producer and consumer. Fuzzy cognitive maps are used to build dependencies, in particular with reflection in the objective function, and optimal solutions are found using a genetic algorithm.

In Ref. [9], a procedure for clustering datasets based on the improved gray wolf algorithm was implemented. The advantage of the proposed approach is to reduce the time for solving optimization problems under conditions of cluster overlap. A feature of the proposed method is its high speed, due to the fact that the entire array is processed only once, i.e., the need for multi-epoch self-learning implemented in traditional fuzzy clustering algorithms is eliminated. In Ref. [10], a method for clustering datasets based on combined optimization of density distribution functions and the evolutionary method of cat swarms was proposed. The advantage of the proposed approach is that it reduces the time for solving optimization problems under conditions of cluster overlap. In Ref. [11], the problem of fuzzy clustering of large datasets that are sent for processing in both batch and online modes was considered, based on a reliable approach. To find the global extremum of the objective function of reliable fuzzy clustering, a modification of the mad cat swarm algorithm was introduced, which combines the advantages of evolutionary algorithms and global random search. It is shown that different search modes are generated by a single mathematical procedure, some cases of which are well-known algorithms for both local and global optimization. The proposed algorithm is characterized by high speed and reliability in multi-extreme fuzzy clustering problems. Thus, for one typical problem of clustering in datasets, various population algorithms were applied, which made it possible to effectively solve optimization problems. In Ref. [12], a new natural algorithm was proposed, which imitates the hunting mechanism of antlions in nature. Five main steps of hunting for prey were implemented, such as random walking of ants, creating traps, capturing ants in traps, catching prey, and rebuilding traps.

In Ref. [13], a new metaheuristic inspired by gray wolves is proposed. The algorithm simulates the leadership hierarchy and hunting mechanism of gray wolves in nature. Four types of gray wolves, such as alpha, beta, delta, and omega, are used to simulate the leadership hierarchy. In addition, three main stages of hunting, prey search, prey surround,

and prey attack are implemented. Then, the algorithm is validated against 29 known test functions, and the results are verified through comparative research using particle swarm optimization, gravitational search algorithm, differential evolution, evolutionary programming, and evolution strategy. The algorithm is able to provide competitive results compared to known metaheuristic methods.

In Ref. [7], a new optimization paradigm inspired by nature is proposed, called the moth and flame optimization algorithm. The main inspiration for this optimizer is the navigation method of moths in nature, called transverse orientation. Moths fly at night, maintaining a fixed angle relative to the moon, which is a very efficient mechanism for traveling in a straight line over long distances. However, these insects are trapped in a death spiral around artificial light. This behavior is mathematically modeled in the paper to perform optimization. The algorithm is compared with other known natural algorithms applied to 29 test and 7 real engineering problems. The statistical results of the comparative functions show that this algorithm is able to provide competitive results. In addition, the results from real problems demonstrate the advantages of this algorithm in solving complex problems with limited and unknown search spaces. In Ref. [13], the problems of automating the construction of classification trees based on the branched feature selection scheme are considered. The object of the study is classification trees. The subject of the study is methods, algorithms, and schemes for constructing classification trees. The aim of the work is to build an effective method for synthesizing classification tree models based on a group assessment of the importance of discrete features within a branched feature selection. A method for constructing classification trees is proposed, which for a given training sample determines the individual informativeness of feature groups and their combinations in relation to the initial value of the classification function. The developed logical tree method, when constructing the next node of the classification tree, tries to select a group of the most closely related discrete features, which reduces the overall structural complexity of the model, speeds up calculations when recognizing objects based on the model, and also increases the generalizing properties of the model. Thus, the approach [7,14] enables adjustment of the following groups of indicators. In Ref. [15], the object of the study is the methodology for assessing the information security indicator. The problem of unification and simplification of the processes for assessing the degree of information security in order to reduce the involvement of human and material resources in them is considered, using the apparatus of fuzzy set theory to take into account the conclusions of competent experts. A fuzzy production model for assessing the degree of information security is developed, which is based on the use of expert knowledge and fuzzy logic methods. A step-by-step approach to identifying potential risks, classifying them by categories, and calculating impact factors is proposed. An iterative assessment method is created, which allows a numerical indicator of the degree of information security to be obtained. Heuristic rules for determining the effective assessment of the degree of information security are developed, taking into account the criticality factor and impact factors of different risk categories. In Ref. [16], an example of optimizing an engine production line based on real-world modeling is described. The optimization is aimed at maximizing the use of machines and, at the same time, minimizing tied-up capital by manipulating 56 unique decision variables. For this purpose, the metaheuristic algorithm of moth and flame is used, which has obtained successful results in various problem areas.

In Ref. [17], a moth and flame optimization algorithm is presented. It is used to solve complex real-world optimization problems in many areas. The paper provides a broad overview of variants of this algorithm, including the classical version, binary types, modified versions, hybrid versions, multi-objective versions, and application of the algorithm to various sectors.

In Ref. [18], the moth and flame algorithm demonstrates better performance results compared to other considered metaheuristic algorithms for solving global optimization problems with nonlinear constraints. However, it does not obtain a high-quality solution and has a low convergence rate.

In Ref. [19], a technological process is considered multi-faceted if there are a number of defects of different types that are detected and eliminated simultaneously during the process. The quality of the technological process is estimated by the probability of initial zero defects, as well as the probabilities of zero defects for each type of defect. Optimization problems involve choosing a process structure that will provide the required initial level of product quality under certain cost constraints. The optimization problem is solved using genetic algorithms that allow an approximate global optimal solution to be found.

In Ref. [20], the theory and application of ant algorithms are considered, using new methods of discrete optimization based on modeling a self-organized colony of biological ants. The colony is considered as a multi-agent system, where each agent functions independently according to simple rules. In contrast to the almost primitive behavior of the agents, the behavior of the entire system turns out to be intelligent. Ant algorithms have been successfully applied to solve many complex combinatorial optimization problems, such as the traveling salesman problem, the vehicle routing problem, the graph coloring problem, the quadratic assignment problem, the network traffic optimization problem, the store work scheduling problem, etc. Ant algorithms are especially effective for online optimization of processes in distributed non-stationary systems (for example, telecommunication networkrouting).

In Ref. [21], a hybrid active power filter is considered. To obtain more accurate parameters, a new population method based on the moth and flame algorithm is proposed. It is modified by dividing it into an operation group and a research group.

In Ref. [22], obtaining appropriate parameters of photovoltaic models based on measured current and voltage data from the photovoltaic system is important for the evaluation, control, and optimization of photovoltaic systems. To obtain specific parameters of photovoltaic models, a metaheuristic algorithm of moth and flame optimization and symbiotic evolution is proposed. The proposed approach divides the population into two parallel working subgroups.

In Ref. [23], the traditional operation of a reservoir for power generation mainly involves maximizing power generation and ensuring the stability of the power system. However, when considering the objectives of power generation, its impact on the ecological environment and navigation can be ignored. A multi-objective optimization model is created that takes into account the objectives of power generation, ecology, and navigation. A new, improved multi-objective moth and flame optimization algorithm based on dominance was proposed to effectively solve the model. In order to improve the ability of the moth and flame optimization algorithm to overcome the local optimum, it has been improved in three aspects: the update formula; the inspiration of the linear flight trajectory of the moth; and the update strategy of the flame population. In order to distinguish individuals that do not exhibit Pareto dominance toward each other, dominance combined with control points has been proposed.

In Ref. [24], a new learning scheme for the extreme kernel learning machine is proposed, based on the use of the chaotic moth and flame optimization strategy. The proposed scheme performs parameter optimization and feature selection simultaneously.

In Ref. [25], an improved moth and flame optimization algorithm is proposed to alleviate the problems of premature convergence to local minima. From the perspective of diversity, the inertia weight of diversity feedback control is introduced into the moth and flame optimization process to balance the use of the algorithm and the global search

capability. In addition, a small probability mutation is added after the position update step to improve the optimization efficiency.

In Ref. [26], the moth and flame optimization algorithm is considered as a new metaheuristic algorithm based on the ensemble to solve global optimization problems. Flame generation and spiral search are two key components that affect the performance of the algorithm. In order to improve the diversity of the flame and the search ability of the moth, an improved moth and flame optimization algorithm is proposed. Its main features are as follows: the flame is generated by orthogonal oppositional learning, and the moth position update mechanism is modified by a linear search approach and the inclusion of a mutation operator.

Ref. [27] proposed a data clustering approach based on the moth–flame optimization (MFO) algorithm, aiming to improve clustering quality by optimizing cluster centroids through swarm intelligence. The authors demonstrated that the MFO-based clustering method outperforms several traditional clustering algorithms, such as k-means and other metaheuristic-based approaches, in terms of clustering accuracy and convergence stability across multiple benchmark datasets. A key strength of this work lies in the ability of MFO to effectively explore the solution space and avoid local optima, which is a common limitation of gradient-based or centroid-initialization-dependent clustering techniques.

Ref. [28] investigated evaluation criteria for selecting centralization strategies in the architecture of multicomputer systems incorporating deceptive mechanisms such as traps and baits. The authors proposed a comprehensive framework for assessing architectural trade-offs between centralized and distributed control, considering metrics such as system performance, fault tolerance, scalability, and security effectiveness. A key contribution of this work is the formalization of evaluation criteria that account not only for computational efficiency, but also for the defensive potential of deceptive components in cyber defense architectures. The study provides a structured decision-making basis for designing resilient multicomputer systems with integrated deception technologies. However, the proposed criteria are primarily focused on architectural-level assessment and do not deeply analyze adaptive behavior under evolving attack patterns or real-time reconfiguration capabilities, which are essential for dynamic cybersecurity environments.

In Ref. [29], the gene selection problem is analyzed, which is associated with a large number of genes (relevant, redundant, or noise) that require an effective method to help with disease detection. In this problem, the computational complexity is reduced by selecting a small number of genes, but it is necessary to select appropriate genes to maintain a high level of accuracy. In order to find the optimal subset of genes, a swarm intelligence algorithm for gene selection is presented, called the quantum moth and flame optimization algorithm, which is based on the combination of quantum computing and the moth and flame optimization algorithm.

Ref. [30] proposed an integrated approach that combines an improved population intelligence algorithm with a backpropagation (BP) neural network to predict network security posture. The work focuses on enhancing prediction accuracy by optimizing neural network parameters through population-based intelligence, addressing nonlinear and dynamic characteristics of network security states. The main strength of this study lies in the effective hybridization of intelligent optimization and neural learning, which leads to improved convergence and prediction stability compared to standard BP models. However, the approach is primarily evaluated on simulated or limited datasets, and the analysis concentrates on prediction accuracy without a deeper examination of robustness, adaptability to evolving attack patterns, or resilience in large-scale distributed environments.

In Ref. [31], the use of genetic algorithms (GA) in computer-aided design (CAD) for perimeter security systems was investigated. The study demonstrates how evolutionary

optimization can support system configuration and design decisions in security infrastructures. A key contribution of this work is the early demonstration of GA applicability to security-oriented engineering problems, particularly optimizing design parameters under multiple constraints. Nevertheless, the work reflects the technological context of its time: it does not consider modern large-scale networks, real-time data streams, or adaptive cyber threats, and it lacks integration with learning-based or multi-agent security architectures.

In Ref. [32], a cryptosystem based on genetic algorithms aimed at enhancing cloud data security was introduced. The authors use evolutionary operations to generate cryptographic keys and encryption mechanisms, emphasizing improved resistance to brute-force and statistical attacks. The advantage of this approach lies in its innovative use of evolutionary randomness and adaptability for cryptographic purposes. At the same time, the work focuses mainly on data confidentiality and cryptographic strength, without addressing network-level anomaly detection, system scalability, or the computational overhead that may arise in real-time cloud environments.

In Ref. [33], a new cloud data security algorithm that combines genetic techniques with logical–mathematical functions was proposed. The method aims to increase data protection efficiency while reducing computational complexity. The main contribution of this work is the formulation of a hybrid algorithm that balances security strength and performance. However, similar to other GA-based security schemes, the evaluation is limited to specific scenarios, and the approach does not consider adaptive adversaries, distributed decision-making, or the interactions among multiple security components within a complex system.

In Ref. [34], a hybrid multi-population optimization algorithm for global optimization was developed, and its effectiveness in stock market prediction was demonstrated. Although the application domain is financial forecasting, the methodological contribution is highly relevant to security analytics. The use of multiple interacting populations improves exploration, avoids premature convergence, and enhances solution diversity. Despite these advantages, the study does not directly address cybersecurity problems, nor does it evaluate system stability or robustness under the adversarial or noisy conditions typical of network security environments.

In Ref. [35], a migration-based moth–flame optimization (MFO) algorithm that enhances exploration and exploitation through controlled migration strategies was proposed. The algorithm shows improved performance in benchmark optimization problems. Its relevance to security research lies in its potential application to feature selection, parameter tuning, and anomaly detection. However, the work remains algorithm-centric, with no direct application to real network traffic or security scenarios, and it does not analyze adaptability in dynamic, adversarial contexts.

In Ref. [36], a comprehensive critical review of the moth–flame optimization (MFO) algorithm and its variants was presented. The authors systematically analyzed structural modifications, performance metrics, and statistical validation across benchmark optimization problems. Their work provides a strong methodological foundation for understanding the strengths and limitations of MFO-based approaches, highlighting issues such as slow convergence in high-dimensional spaces and sensitivity to parameter settings. Although the survey offers valuable comparative insights, it remains largely algorithm-centric and does not address application-specific challenges such as real-time processing constraints or robustness when applied to noisy and non-stationary data streams.

In Ref. [37], MFO-SFR, an enhanced MFO variant incorporating a stagnation finding and replacing strategy to avoid premature convergence was introduced. The proposed mechanism detects stagnation phases and dynamically replaces poorly performing solutions, thereby improving convergence speed and solution quality. The experimental

results with regard to benchmark functions confirmed improved optimization performance compared to standard MFO. Despite these improvements, the study focused on static benchmark problems and did not explore the behavior of the algorithm in continuously evolving or dynamic optimization scenarios, which are common in real-world applications such as adaptive system control or online anomaly detection.

Ref. [38] extended this research direction by introducing a control and decision-making framework for deceptive multicomputer systems based on accumulated operational experience. The proposed approach leverages historical attack data and system responses to improve defensive strategies for protecting critical infrastructure. The main strength of this work lies in its emphasis on experience-driven adaptation, enabling the system to refine deception tactics and resource allocation policies over time. This contributes to enhanced situational awareness and more effective threat mitigation.

In Ref. [39], a comprehensive review of modification strategies for nature-inspired algorithms in feature selection problems was presented. This survey systematically analyzes enhancement techniques such as hybridization, adaptive parameter control, and multi-population mechanisms. The strength of this work is its broad analytical perspective, which provides valuable insights for designing more effective optimization-based security models. On the other hand, as a review paper, it does not propose or validate a concrete security system, nor does it assess end-to-end detection performance or system-level resilience.

In Ref. [40], a comprehensive survey of the Runge–Kutta optimization algorithm (RKOA), covering its theoretical foundations, variants, applications, and performance evaluation, was conducted. This work contributes a rigorous mathematical perspective to optimization research and highlights the algorithm's potential in complex, nonlinear problem domains. While the survey is methodologically strong, its application to cybersecurity remains indirect, and issues such as real-time operation, distributed deployment, and adversarial robustness are not addressed.

In Ref. [41], the application of genetic algorithms to network information security management was explored. The study emphasizes optimization-driven decision support for security configuration and management tasks. Its contribution lies in demonstrating GA applicability to practical security management problems. However, the work focuses on static or semi-static optimization and does not incorporate learning-based adaptation, multi-agent coordination, or evaluation metrics beyond basic performance improvements.

In Ref. [42], security from a socio-technical perspective, analyzing how algorithmic security systems are perceived by refugees and data experts, was approached. The contribution is conceptual rather than algorithmic, emphasizing the ethical, social, and governance dimensions of security technologies. While this perspective is valuable for understanding the broader impact of algorithmic security systems, the work does not propose technical methods for detection, optimization, or resilience, and it lacks quantitative evaluation.

In Ref. [43], a hybrid approach combining jellyfish search and moth–flame optimization algorithms to solve the optimal power flow (OPF) problem in power systems was developed. The method demonstrated improved stability, faster convergence, and higher-quality solutions compared to traditional optimization techniques. This work highlights the effectiveness of hybrid metaheuristic frameworks in handling large-scale, nonlinear, and constrained optimization problems. Nevertheless, the application scope remains limited to power system optimization, and the transferability of the approach to other complex domains, such as distributed network security or real-time decision-making systems, was not assessed.

In Ref. [44], a robust fractional-order MPPT-based (maximum power point tracking) MFO algorithm for thermoelectric generation systems was proposed. By integrating fractional calculus with MFO, the authors improved tracking accuracy and system robustness

under fluctuating environmental conditions. The main contribution of the study is the enhancement of control performance in energy harvesting systems, demonstrating the adaptability of MFO to engineering control problems. However, the approach is tailored to a specific physical application and does not directly address scalability, multi-objective trade-offs, or adversarial conditions relevant to information and communication systems.

In Ref. [45], a fractional-order MFO-based approach for Otsu image segmentation was introduced. The proposed method improved segmentation accuracy and robustness against noise by combining fractional calculus with swarm intelligence optimization. The work demonstrates the flexibility of MFO in image processing and computer vision tasks, particularly in optimizing threshold-based segmentation. Despite its strong performance in static image datasets, the method does not consider real-time image streams, computational efficiency constraints, or integration with higher-level decision-making frameworks.

In Ref. [46], a chaos-enhanced moth–flame optimization algorithm for multi-objective dynamic economic dispatch problems involving plug-in electric vehicles was proposed. The incorporation of chaotic maps improved population diversity and enhanced the algorithm's ability to adapt to time-varying system conditions. The study highlights the potential of chaos-based strategies to address dynamic and multi-objective optimization challenges. However, the evaluation focused on energy management scenarios and did not analyze broader system-level properties such as long-term stability, resilience to unexpected disturbances, or coordination among multiple autonomous agents.

The authors of [47,48] proposed an adaptive, improved moth–flame optimization variant that dynamically adjusts the search behavior to better balance exploration and exploitation and showed its effectiveness with regard to numerical benchmark functions and mechanical engineering design problems.

Summary data on the features of the moth and flame algorithm are presented in Table 1.

Table 1. Summary data on the features of the moth and flame algorithm.

Paper	Continuous Optimization	Discrete Optimization	The Essence of the Work
[7]	+	+	General scheme of the moth and flame algorithm
[16]	+		The optimization of the engine production line based on the moth and flame algorithm is presented
[17]	+	+	An overview of algorithm variants is presented, including the classical version, binary types, modified versions, hybrid versions, multi-objective versions, and application of the algorithm to different sectors
[18]	+		Proof of the advantages of the moth and flame optimization algorithm for solving global optimization problems with nonlinear constraints is presented
[21]	+		A hybrid filter is developed based on the moth and flame optimization algorithm
[22]	+		Moth and flame optimization algorithm and symbiotic evolution
[23]	+		Moth and flame optimization algorithm based on dominance
[24]		+	Chaotic moth and flame optimization strategies are used
[25]	+	+	The algorithm is modified: after the position update stage, a small probabilistic mutation is added to improve the optimization efficiency
[26]	+	+	Algorithm modifications: the flame is generated by orthogonal oppositional learning; the moth position update mechanism is modified using a linear search function and a mutation operator
[27]	+	+	A data-clustering approach based on the moth and flame algorithm is developed

Table 1. *Cont.*

Paper	Continuous Optimization	Discrete Optimization	The Essence of the Work
[29]		+	A selection problem based on the moth and flame optimization algorithm
[35]		+	A migration-based moth–flame optimization (MFO) algorithm
[36]		+	Analysis of the algorithm is carried out
[37]		+	An algorithm based on a stagnant detection and replacement strategy to avoid premature convergence
[44]	+		The adaptability of the algorithm to engineering control problems is presented
[45]		+	A fractional order-based algorithm for image segmentation is presented
[46]		+	A chaos-aware moth–flame optimization algorithm for multi-objective dynamic dispatching problems

Thus, the use of population algorithms to solve optimization problems is promising. Among the many population algorithms, there is a class of such algorithms inspired by nature that can be synthesized in the architecture of decoy systems with baits and traps, since their behavior, i.e., the selected next steps to be executed, will be difficult for attackers to understand.

The aim of this work is to improve the functioning of decoy and trap deceptive systems in corporate networks by synthesizing population algorithms in their architecture to confuse attackers and maintain long-term system responses to impacts caused by cyberattacks and the actions of the malware.

3. Presentation of the Main Material

In the architecture of decoy and trap deceptive systems, the synthesis of population algorithms, in particular the moth and flame algorithm, is necessary to optimize the sequence of subsequent steps in their formation. When implementing cyberattacks and malware actions, it is necessary to avoid complete enumeration of options and rapid convergence of the selected steps with ongoing influences, to change the sequence of steps, taking into account current changes in the environment of corporate networks, and to take into account the potential capabilities of attackers to implement dual-purpose cyberattacks. Therefore, we will first develop a method for selecting steps for decoy and trap deceptive systems based on the moth and flame algorithm as part of ensuring their functioning.

Dual-purpose cyberattacks have the following features that are associated with the use of false objects for attacks and their inclusion in the cyberattacks scenario with the allocation of certain stages. This allows the attacker not only to divide real and false objects in corporate networks, but also, according to the developed scenario of the cyberattacks, to involve false objects that are taken under the attacker's control to perform further actions. Such actions of the attacker must be taken into account by the developers of security and protection systems for corporate networks, in which additional false objects are used for attacks. The control and interactions of such false objects for attacks and their internal architecture must take into account the actions of the attacker, which are based on the developed general model of dual-purpose cyberattacks, which requires the development of a new architecture of control and interactions of false objects for attacks in corporate networks, taking into account the following features. The dual-purpose cyberattack model is based on the division of cyberattacks into stages, with allocation of the stages at which the classification of real and false objects in corporate networks occurs. In general, the positioning of dual-purpose cyberattacks through division into stages corresponds to the classic "kill chain" model (Lockheed Martin cyber kill chain), in which eight stages of cyberattacks are distinguished. The eighth stage in the "kill chain" model corresponds to the reward for the attacker and may not be considered. Therefore, the number of stages

or phases of the cyber chain may contain seven stages in this model. But the "kill chain" model is linear, because it is assumed that all stages are performed sequentially, which is a disadvantage, because attackers can perform the stages out of sequence and branch them and divide them into separate substages. Administrators of corporate networks that provide protection and security in networks, when focusing protection precisely on repelling attacks that correspond to the "kill chain" model, lose their advantage over attackers. Also, this model is not very effective against internal threats and attacks that are not related to the CCP. The developed dual-purpose cyberattack model is improved compared to the "kill chain" model in terms of its lack of linearity and takes into account the attackers' focus on classifying corporate network objects.

The multi-stage nature of cyberattacks requires corporate network administrators to take into account the behavior of attackers. Therefore, the MITRE ATT&CK (adversarial tactics, techniques, and common knowledge) model was proposed to develop the kill chain model, in which the number of cyber chain stages was increased to fourteen. The developed dual-purpose cyberattack model isdivided into eighteen stages. In addition, the MITRE ATT&CK model focuses on indicators of compromise, which are classified in the attacker's pain pyramid (David Bianco), and offers a list of tactics, methods, and procedures of attackers to complicate the actions of attackers on the part of users. The proposed dual-purpose cyberattack model can be applied to active protection technologies, in particular deceptive systems with baits and traps, which corresponds to the means and systems for countering attackers, which are assigned to the highest level of the attacker's pain pyramid.

Thus, the developed model of dual-purpose cyberattacks in corporate networks divides the chain of cyber threats into eighteen stages, allows for their nonlinear execution by attackers, takes into account the peculiarities of attackers' behavior in classifying real and false objects, is aimed at the available indicators of compromise from the attacker's pain pyramid, and is the basis for developing deceptive systems with baits and traps.

The choice of population algorithms in the architecture of deceptive systems corresponds to the tasks that need to be solved in them, which are associated with unknown and limited spaces for searching for options for the next steps. They are used to avoid premature convergence, so that optimization ensures convergence to the global optimum. For this purpose, a set of solutions can be used that form due to the availability of resources in deceptive systems, in particular a certain number of baits and traps, as well as the possibility of forming many options for the next steps of the systems.

The optimization process when choosing solutions is divided into two stages: research into the available options and application of the selected options. At the research stage, we select a step sequence option from the search space, forming a vector of subsequent steps of deceptive systems. At the stage of application of the selected options, we perform the selected steps and improve the selected option through the sequence of subsequent steps in order to approach the global optimum. Division of the optimization process into two stages is possible becausethe duration of the cyberattacks or the actions of the malware and their variability require a flexible response to changes in the environment of corporate networks. The optimization process, with division into two stages, is shown in Figure 1.

Thus, initially, the vector $v_{os,i}$ will be formed in the deceptive system by a given sequence of steps to ensure a response to the impacts in corporate networks from the cyberattacks or the actions of the malware, and this will correspond to the stage of researching the available options. At the stage of applying the selected options, some of the steps will be performed by the deceptive system, and the steps will be included in the execution process to assess the current state in corporate networks, the state of the system, and the

effectiveness of the steps. According to the processed information, the selected step options will be refined, and a new vector of the next system steps will be formed, as follows:

$$v_{os,i,u} = \left(v_{os,i,u,1}, v_{os,i,u,2}, \ldots, v_{os,i,u,N_{os,i,u}}\right), \quad (1)$$

where $v_{os,i,u,j}$ is the step of the decoy system with baits and traps; $j = \overline{1,\ N_{os,i,u}}$; $N_{os,i,u}$ is the number of steps of decoy systems with baits and traps to solve a certain task, which are determined by the system at the current time; i is the i-th sequence of steps to solve a certain task of the system; and u is the refinement of the steps, which is given by the vector $v_{os,i}$.

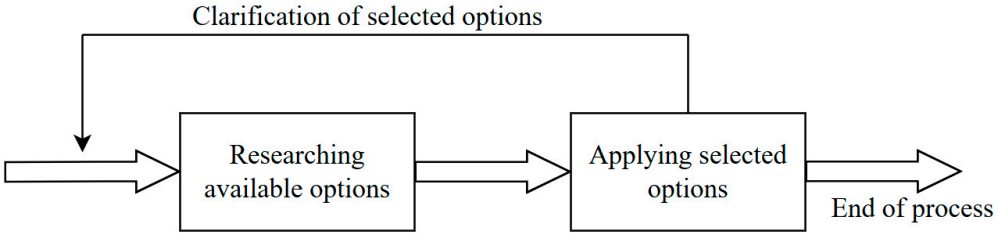


Figure 1. Division of the optimization process into two stages.

Let us introduce the objective function for evaluating the selected options, as follows:

$$F_{os}(v_{os,i}, v_{os,i,u}) \rightarrow [0;1], \quad (2)$$

where $v_{os,i,u,j}$ is the step of the deceptive system with baits and traps; $j = \overline{1,\ N_{os,i,u}}$; $N_{os,i,u}$ is the number of steps of deceptive systems with baits and traps to solve a certain task, which are defined by the system at the current time; i is the i-th sequence of steps to solve a certain task of the system; and u is the refinement of the steps, which are given by the vector $v_{os,i}$.

In Ref. [28], a methodology for constructing evaluations of selected step options using an objective function and corresponding criteria for deceptive systems with baits and traps is presented.

The best result of the evaluation according to the objective function F_{os} (Formula (2)) is the greater of its two values. The objective function F_{os} (Formula (2)) gives an estimate of the refined variant of the sequence of steps, which will form the vector $v_{os,i,u}$ and is responsible in the optimization process for refining the selected options (Figure 1). At the beginning of the decision-making process, when the available options are studied, it is also used, but its arguments will be the same; then it evaluates the selected option, which is fixed in the deceptive system. Part of the steps specified by the vector $v_{os,i}$ can be repeated in the vector $v_{os,i,u}$. The objective function will be applied for as long as the cyberattacks or the actions of the malware continue, which will require formation of the corresponding actions of the deceptive systems with baits and traps. The sequences of the systems' steps will change, and each time will be evaluated by the objective function and directed towards achieving the global optimum. Such a scheme corresponds to the general structure of population algorithms.

Let us define the encoding function of corporate network objects on the coordinate plane, taking into account their different types:

$$F_{os,kod}: \left(M_{kn,v,p,z,vect}, O_{kn}\right) \rightarrow M_{os,kod}\left(X_1, X_2, \ldots, X_{N_{kn,v,p,z}}, K_{os}\right), \quad (3)$$

where $M_{os,kod}$ is a matrix with coordinates of corporate network objects and information about the type of objects; K_{os} is a code of the type of corporate network objects; $K_{os} = \{k_{kod,1}, k_{kod,2}, \ldots, k_{kod,N_{kn,v,p,z}}\}$; O_{kn} is the type of corporate network objects; X_j is a vector j of coordinates; $m_{kn,v,p,z,vect,i,j}$ is a component of a vector from the elements of

the subset $M_{kn,v,p,i}$ ($i = \overline{1,\ N_{kn,v,p}}$, $N_{kn,v,p}$ is the number of subsets of corporate network nodes in the set $M_{kn,p}$); $N_{kn,v,p,z}$ is the maximum number of nodes in network segments, including the router; j is the column number of the matrix; $N_{kn,v,p,z} = \overline{1,\ N_{kn,v,p,z,\ N_{kn,v,p,z,j}}}$; $j = \overline{1,\ N_{kn,v,p,z,j}}$; $N_{kn,v,p,z,j}$ is the number of nodes in the j-th network segment, including the router; $N_{kn,v,p,z,j} \le N_{kn,v,p,z}$; $i = \overline{1,\ N_{kn,v,p}}$; and $N_{kn,v,p}$ is the number of subsets of corporate network nodes in the set $M_{kn,p}$.

Vector j with coordinates X_j is given, taking into account Formula (3), as follows:

$$X_j = \begin{pmatrix} x_{kn,v,p,z,vect,1,j} \\ x_{kn,v,p,z,vect,2,j} \\ \dots \\ x_{kn,v,p,z,vect,N_{kn,v,p},j} \end{pmatrix}, \quad (4)$$

where $x_{kn,v,p,z,vect,i,j}$ is the coordinate on the coordinate plane of the vector component from the elements of the subset $M_{kn,v,p,i}$ ($i = \overline{1,\ N_{kn,v,p}}$, $N_{kn,v,p}$ is the number of subsets of corporate network nodes in the set $M_{kn,p}$); $N_{kn,v,p,z}$ is the maximum number of nodes in network segments including the router; j is the matrix column number; $j = \overline{1,\ N_{kn,v,p,z,\ N_{kn,v,p,z,j}}}$; $j = \overline{1,\ N_{kn,v,p,z,j}}$; $N_{kn,v,p,z,j}$ is the number of nodes in the j-th network segment, including the router; $N_{kn,v,p,z,j} \le N_{kn,v,p,z}$; $i = \overline{1,\ N_{kn,v,p}}$; and $N_{kn,v,p}$ is the number of subsets of corporate network nodes in the set $M_{kn,p}$.

The set K_{os} may contain the following elements:

(1) $k_{kod,1}$—computer stations with components of deceptive systems that do not contain decoys or traps;
(2) $k_{kod,2}$—computer stations with components of deceptive systems that do contain decoys and/or traps;
(3) $k_{kod,3}$—computer stations with decoys and/or traps;
(4) $k_{kod,4}$—routers;
(5) $k_{kod,5}$—servers with components of deceptive systems that do not contain decoys or traps;
(6) $k_{kod,6}$—servers with components of deceptive systems that do contain decoys and/or traps.

Objects of corporate networks $k_{kod,1}$–$k_{kod,6}$ can be divided into those located in the demilitarized zone and those not located in it. And also, objects can be divided according to the territorial principle; that is, if they are located in different buildings or premises, including in private territory. Then, the number of such objects increases. IP cameras can also be included in this category.

Let us construct one of the variants of the function $F_{os,kod}$ (Formula (3)). Let n_R be the number of corporate network objects in R–segment. R will also be the segment number, and on the coordinate plane it will be the value of the circle radius. If there is one circle radius in the corporate network, then it is equal to one $R = 1$, and this will mean the absence of division into segments; that is, the presence of all nodes of the corporate network in one segment. If there are two segments, then there will be two circles on the coordinate plane with radii $R = 1$ and $R = 2$. Similarly, such a representation can be generalized for an arbitrary number of segments in corporate networks. Then, for such a case, the number of circles will be the same as the number of segments, and the radius of the circles will be chosen from the sequence of numbers $R = \overline{1,\ k_R}$, where k_R is the number of segments.

We will place the corporate network objects of one segment on a circle at the vertices of a regular polygon inscribed in this circle. There are options for choosing an arbitrary polygon, but then we need to specify a function that sets the distances on the circle between two points, and for all points this function must correctly set the coordinates. In the case of choosing a regular polygon, all neighboring points on the circle will be equidistant. To

find the coordinates of an n-gon inscribed in a circle with radius R and the center at point O (0;0) of the coordinate plane, we will specify the initial vertex from which the coordinates of all the remaining vertices on the circle will be determined. In order for the initial points to not be on the same semi-axis if there are two or more circles, we will specify the coordinates of the initial points taking into account their offset and depending on the numbers of the circles, which are determined by radii of different lengths. The coordinates of the initial points, taking into account the lengths of the radii on the coordinate plane, are as follows:

$$\begin{gathered} x_{1,R_m} = R_m * \cos\left(m^\circ\right); \\ x_{2,R_m} = R_m * \sin\left(m^\circ\right); \\ R_m = m;\ m \in N;\ m = 1,\ 2,\ 3,\ \ldots, \end{gathered} \tag{5}$$

where m is a natural number that varies from 1 to the value of the number of segments of the corporate network; R_m is the radius of the circle with the origin in the center of the coordinates; and x_{1,R_m} and x_{2,R_m}) are the coordinates of the initial point of the polygon on the coordinate plane.

The angle of displacement of each subsequent initial vertex of the polygon on each subsequent circle increases by 1°.

The correctness of the given coordinates of the initial points of the polygons can be checked by calculating the length of the radius between the center of the coordinate plane and the coordinates of the points obtained by Formula (5) according to the formula of the circle equation:

$$(x_{1,R_m})^2 + (x_{2,R_m})^2 = {R_m}^2. \tag{6}$$

The first starting point on the circle for the polygon will be on the abscissa and is $x_{1,R_1} = 1$, $x_{2,R_1} = 0$. The second starting point is $x_{1,R_2} = 2 * \cos\left(2^\circ\right)$, $x_{2,R_2} = 2 * \sin\left(2^\circ\right)$, the third starting point is $x_{1,R_3} = 3 * \cos\left(3^\circ\right)$, $x_{2,R_3} = 3 * \sin\left(3^\circ\right)$.

Regular polygons inscribed in circles will have a different number of vertices, but some of them may have the same number of vertices. In this case, they will have different sizes, because the radii of their circles are different, and their starting points and the rest of the corresponding points by numbers will not belong to the corresponding lines passing through the center of the circle, due to the offset of the coordinates of the starting points for each polygon.

In fact, according to Formula (5), the designation of points to determine their values will be as follows:

$$x_{kn,v,p,z,vect,R_{m,1}} = x_{1,R_m};\ x_{kn,v,p,z,vect,R_m,2} = x_{2,R_m}. \tag{7}$$

The remaining vertices and their coordinates are obtained by rotating the initial vertex by an angle equal to $\frac{360^\circ}{n_{R_m}}$, where n_{R_m} is the number of objects in the corporate network segment. If there is one object in the segment, then it is placed at the initial point of the polygon of the corresponding circle. The coordinates of the remaining points on the circles that are the vertices of n_{R_m}—polygons are defined as follows:

$$\begin{gathered} x_{1,R_m} = R_m * \cos\left(m^\circ + (n_{R_m} - 1) * \tfrac{360^\circ}{n_{R_m}}\right);\ x_{2,R_m} = R_m * \sin\left(m^\circ + (n_{R_m} - 1) * \tfrac{360^\circ}{n_{R_m}}\right); \\ R_m = m;\ m \in N;\ m = 1,\ 2,\ 3,\ \ldots;\ n_{R_m} \in N;\ n_{R_m} = 1,\ 2,\ 3,\ \ldots \end{gathered} \tag{8}$$

Thus, the preparatory stage for implementation of the moth and flame algorithm consists of representing corporate network objects on a coordinate plane, divided into between segments. Such a representation is necessary to ensure the control of decoy systems by baits and traps in the context of optimizing a portion of the system steps during the SC or the actions of the SPL and correlating them to places that are influenced by the

SC and the actions of the SPL. The geometric interpretation of such a representation, which uses Formulas (5) and (8), and an example of a potential spiral are shown in Figure 2.

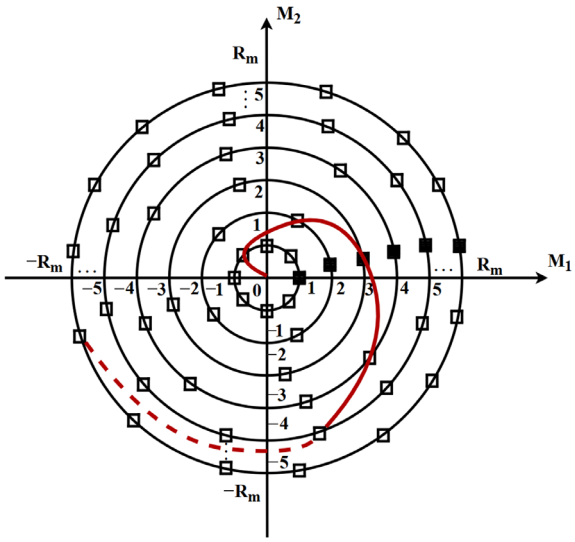


Figure 2. Geometric interpretation of the placement of corporate network objects in segments and the trace of a potential spiral.

Similarly, for three-dimensional and multidimensional spaces, the coordinates of points in space can also be specified by formulas, and for three-dimensional space, corporate network objects will be placed on spheres that will have one center and nest into one another, and the spiral will be a line in space, not in a plane.

Geometric interpretation of the placement of corporate network objects in segments and the potential spiral in Figure 2 can be modified so that each circle is in a separate parallel plane. Then, the spiral will also be located in three-dimensional space, not in a plane.

The function for placing corporate network objects is a discrete value, and the list of subsequent deceptive steps is a discrete value; therefore, we will perform discrete optimization using the moth and flame algorithm.

At the next stage, after the preparatory stage, we will specify the objects on which the effects of the cyberattacks and the actions of the malware are exerted, as well as the components of the deceptive systems and baits and traps, taking into account Figure 2 and Formulas (3)–(5) and (7). The filled squares are the initial labels of the points on the circles, and the unfilled ones are placed taking into account these initial points and formulas. Important: the filled points should not lie on the same line. Otherwise, their corresponding subsequent points will also be on the same line, and this will demonstrate the partial case. And we need to reach the general case in order to diversify the number of options.

Let us first consider a direct scheme in which the center of the coordinate system is the bait and/or trap, and the effects of the cyberattacks or the actions of the malware are reflected in the corporate network objects that are placed on circles, keeping in mind that these effects may not be present in all corporate network objects. The malware may be active in certain nodes of corporate networks, in which case such nodes will be considered as objects in which there are moths. Similarly, objects to which the cyberattacks is directed and to which certain effects occur will also be considered as those that may have moths. In the case of gaining control over corporate network objects during a cyberattack, when the attacker implements malware in them to exercise control over them or destroys everything in them, we will also consider this option as the presence of a moth, which will correspond to the first two with the already existing malware and the effects of the cyberattacks. Thus,

the cyberattacks or the actions of the malware, as well as the malware itself, directed or present in corporate network objects, will be positioned as a moth. That is, we assume that the candidates in the context of solutions are a moth, that there can be more than one of them, and that the variables of the problem are the position of the moth in space. Then, more than one mothcan fly in space in any dimension by changing its their position vector. We will assign a certain color to such objects on the coordinate plane and specify them with a matrix of coordinates in a certain dimensional space, as follows:

$$A_M = \begin{pmatrix} a_{1,1} & \dots & a_{1,N_{A_M,s}} \\ \vdots & \ddots & \vdots \\ a_{N_{A_{M,r}},1} & \dots & a_{N_{A_{M,r}},N_{A_M,s}} \end{pmatrix}, \quad (9)$$

where $a_{i,j}$ is the coordinate of the moth in a certain dimensional space; $N_{A_M,s}$ is the dimension of the space; $i = \overline{1,\ N_{A_{M,r}}}$; $N_{A_{M,r}}$ is the number of objects in which the moth is present; and $j = \overline{1,\ N_{A_M,s}}$.

In the representation by matrix A_M, the population of moths is formed at the current moment in time. After a certain period of time, it can change quantitatively. For Formulas (5) and (8) and the image in Figure 2, $N_{A_M,s} = 2$.

The impact, movement, and presence of the cyberattacks or the actions of the malware in corporate networks objects is the basis for determining the further steps of deceptive systems. Among these objects there may be false objects for attacks, but they are controlled by deceptive systems. In this case, it is possible to assess the impact, movement, or presence of the spacecraft or the actions of the malware in corporate network objects, and for such an assessment, we will introduce a fit function that will reflect the proximity of the constructive solution to the set goal. We will determine such a value of the fit function for each moth by its coordinates, which are given by Formula (9), and then we will obtain the following matrix:

$$F_1^{A_M} = \begin{pmatrix} F_{1,1}^{A_M} \\ F_{1,2}^{A_M} \\ \dots \\ F_{1,N_{A_{M,r}}}^{A_M} \end{pmatrix}, \quad (10)$$

where $F_{1,j}^{A_M}$ is the value of the fitness function $F_1^{A_M}$ for the j-th moth; $N_{A_{M,r}}$ is the number of objects in which the moth is present; and $j = \overline{1,\ N_{A_M,s}}$.

Baits and traps can be placed in space and defined by coordinates. The center of coordinates may be a bait or trap that the moth seeks to approach, but baits and traps may be inactive and be in the path of the moth. Let us define the flame corresponding to baits and traps, as well as components of deceptive systems, by the matrix B_M, similarly to the matrix A_M, as follows:

$$B_M = \begin{pmatrix} b_{1,1} & \dots & b_{1,N_{A_M,s}} \\ \vdots & \ddots & \vdots \\ b_{N_{A_{M,r}},1} & \dots & b_{N_{A_{M,r}},N_{A_M,s}} \end{pmatrix}, \quad (11)$$

where $b_{i,j}$ is the coordinate of the flame in a certain dimensional space; $N_{A_M,s}$ is the dimension of the space; $i = \overline{1,\ N_{A_{M,r}}}$; $N_{A_{M,r}}$ is the number of objects in which the flame is present; and $j = \overline{1,\ N_{A_M,s}}$.

Let us assume that the dimensions of the matrices A_M and B_M are the same, assuming that the moths are search agents that move in the search space, and the flame is their best position at the current time. In the context of improving the functioning of deceptive systems

and discrete optimization, the same dimensions of matrices are suitable for cyberattacks or actions of the malware. To counteract them, appropriate means are selected, and their number depends on the detected malicious influences, which are steps of deceptive systems, as well as the general selection of steps from a set of system steps. That is, the flame can be considered a reference point for the moth in the search space. Then, the moth searches around the flame and updates it when a better solution is found. Therefore, as a result of such targeted actions, the best solution is never lost. For cyberattacks or malware actions, targeting occurs to objects that are the goal of the attackers. In this process, approaching the goal improves the following options. On the part of deceptive systems, there is also a focus on processing malicious influences, which is achieved by adding new baits and traps, which is reflected in certain steps of the systems. At the same time, the choice of steps is carried out with clarification of the state in corporate network environments; that is, at each stage, the best option of steps is selected. Let us introduce, similarly to the fit function $F_1^{A_M}$, the fit function $F_2^{B_M}$ for evaluating options from the matrix B_M, as follows:

$$F_2^{B_M} = \begin{pmatrix} F_{2,1}^{B_M} \\ F_{2,2}^{B_M} \\ \dots \\ F_{2,N_{A_{M,r}}}^{B_M} \end{pmatrix}, \tag{12}$$

where $F_{2,j}^{B_M}$ is the value of the fitness function $F_1^{B_M}$ for the j-th flame; $N_{A_{M,r}}$ is the number of objects in which there is a flame; and $j = \overline{1,\ N_{A_M,s}}$.

Let us consider the steps of the discrete optimization algorithm for moths and flames, by which the global optimum is approximated.

We define three functions by Formulas (13)–(15) according to the standard approach from [7].

The function F_1^A generates a random population of moths, places it in space, i.e., forms the matrix A_M (Formula (9)), calculates the value of the fitness function $F_1^{A_M}$ (Formula (10)) for the generated population of moths, and defines it as follows:

$$F_1^A :\ \varnothing \to \left\{ A_M, F_1^{A_M} \right\}. \tag{13}$$

The function F_2^A is the main function. It moves the moth in the search space. After receiving the matrix A_M, this function returns it with the updated values of the elements. That is, after its execution, we again get the matrix A_M. Let us define the function F_2^A as follows:

$$F_2^A :\ A_M \to A_M. \tag{14}$$

The function F_3^A returns the value "true" if the termination criterion of the algorithm is satisfied and the value "false" if the termination criterion is not satisfied.

Let us define the function F_3^A as follows:

$$F_3^A :\ A_M \to \{true,\ false\}. \tag{15}$$

According to the inputs, we define the algorithm for discrete optimization of moths and flames as follows:

$$\begin{array}{l} A_M = F_1^A(\ ); \\ \text{while } F_3^A(A_M) = \{false\} \\ \qquad A_M = F_2^A(A_M); \\ \text{end} \end{array} \tag{16}$$

The function F_1^A must generate initial solutions and calculate the value of the objective function $F_1^{A_M}$, which is given by Formula (10). Any random distribution can be used in function F_1^A. Let us specify the following distribution:

$$\begin{array}{l} for i = 1\ ,\ n\ ; step\ 1 \\ \qquad for i = 1\ ,\ n\ ; step\ 1 \\ \qquad\qquad a_{i,j} = (c_i - d_i)\ *\ rand(\ ) + d_i; \\ \qquad End \\ End \\ F_1^{A_M} = F_1^{A_M}(A_M); \end{array} \tag{17}$$

where $a_{i,j}\ \epsilon\ A_M$; $a_{i,j}$ is the coordinate of the moth in a certain dimensional space; $N_{A_M,s}$ is the dimension of the space $= \overline{1,\ N_{A_{M,r}}}$; $N_{A_{M,r}}$ is the number of objects in which the moth is present; and $j = \overline{1,\ N_{A_M,s}}$.

For the previous algorithm, the matrices C and D, whose elements determine the upper and lower limits of the change in values, are given by their elements, as follows:

$$C = \left(c_1, c_2, \ldots, c_{N_{A_{M,r}}}\right);\ D = \left(d_1, d_2, \ldots, d_{N_{A_{M,r}}}\right), \tag{18}$$

where the elements of matrix C are the values of the upper bound of the i-th variable; the elements of matrix D are the values of the lower bound of the i-th variable; c_i is an element of matrix C; d_i is an element of matrix D; $i = \overline{1,\ N_{A_{M,r}}}$; and $N_{A_{M,r}}$ is the number of objects in which the moth is present.

The function F_2^A is executed iteratively and moves the moth in the search space. After the function F_3^A returns the value {"true"}, its execution will be completed. The search space is discrete and is defined by the vector matrix $M_{kn,v,p,z,vect}$ and the matrix A_M (Formula (9)). It is displayed on the coordinate plane in Figure 2. The algorithm for discrete optimization of the moth and flame is characterized by a feature that consists of maintaining a transverse orientation when choosing the next position of the moth relative to the flame. We specify such a movement as follows:

$$A_{M,i} = Q(A_{M,i},\ B_{M,i}), \tag{19}$$

where $\bigcup_{i=1}^{N_{A_{M,r}}} A_{M,i} = A_M$; $A_{M,i}$-i is the moth; $\bigcup_{i=1}^{N_{A_{M,r}}} B_{M,i} = B_M$; $A_{M,i}$-i is the bait or trap or component of the deceptive system; $i = \overline{1,\ N_{A_{M,r}}}$; $N_{A_{M,r}}$ is the number of objects in which the moth is present; and Q is a function that will specify the transverse orientation when choosing the next position of the moth relative to the flame.

The function Q will determine the movement. Since the task of choosing steps belongs to discrete optimization, and the constructed search field is also discrete, the function Q will be discrete. The spiral is the basis of the scheme according to which the moth should be updated. There can be many variants of spirals, and they can be specified by different variants of the Q function. But the following conditions must be met when constructing variants of the Q function:

(1) the movement must be in a spiral;
(2) the movement must be performed exclusively in compliance with the transverse orientation;
(3) the initial boundary point of the spiral must be in one of the corporate network objects in which potentially malicious manifestations are established, or in a general approach—from the point or step at which the process is started;
(4) the final boundary point of the spiral must be a point with a flame;

(5) the deviation of the spiral should not go beyond the search space; that is, the spiral should be within the extreme points that are maximally distant from the center (from the point with the flame);
(6) the spiral should not degenerate into a segment connecting the extreme points in a straight line;
(7) the spiral should not contain reverse arcs and should be directed either clockwise or counterclockwise;
(8) the construction of the spiral should be consistent with time as a parameter;
(9) the flame can change its coordinates over time;
(10) each subsequent arc between the steps selected for the spiral should be on the same circle as the previous arc or closer to the flame;
(11) when updating the flame location, the coordinates of the moth are updated;
(12) if there is more than one moth and one flame, the first moth must approach the flame before the rest;
(13) if there is more than one moth and one flame, the last moth must receive a solution from the rest;
(14) if there is more than one flame and more than one moth, with the same number of flames and moths, the movement of each moth is directed towards a separate flame, and the number of flames decreases with time;
(15) if there is more than one flame and more than one moth, but with different numbers of flames and moths, the movement of each moth tends towards a separate flame, but in the absence of a sufficient number of flames, it is distributed between them, and the number of flames decreases with time.

To construct the function Q according to the specified conditions, we introduce the following rules:

(1) $\forall\, x_{l,i} : x_{l,i+1} \leq x_{l,i}$; l is the number of coordinates; $i = \overline{1,\ N_{A_{M,r}}}$; $N_{A_{M,r}}$ is the number of objects in which there is a moth;

(2) $\forall\, x_{l,i} : x_{l,i+1} = x_{l,i}$ or $x_{l,i+1} < x_{l,i}$; l is the number of coordinates; $i = \overline{1,\ N_{A_{M,r}}}$; $N_{A_{M,r}}$ is the number of objects in which there is a moth;

(3) a_1 (coordinate in matrix A_M, which is given by Formula (9)) is contained in one of the elements of the matrix $M_{kn,v,p,z,vect}$; $a_{i,j}$ is the coordinate of moth a_1 in a certain dimensional space; $N_{A_M,s}$ is the dimension of space; $i = \overline{1,\ N_{A_{M,r}}}$; $N_{A_{M,r}}$ is the number of objects in which a moth is present; $j = \overline{1,\ N_{A_M,s}}$;

(4) $a_{N_{A_{M,r}}}$ (coordinate in matrix A_M, which is given by Formula (9)) is contained in one of the elements of matrix $M_{kn,v,p,z,vect}$; $a_{i,j}$ is the coordinate of moth a_1 in a certain dimensional space; $N_{A_M,s}$ is the dimension of space; $i = \overline{1,\ N_{A_{M,r}}}$; $N_{A_{M,r}}$ is the number of objects in which there is a moth; $j = \overline{1,\ N_{A_M,s}}$;

(5) $\forall\, x_{l,i} : \left(x_{l,1,R_l}\right)^2 + \left(x_{l,2,R_l}\right)^2 \leq {R_{max}}^2$; l is the number of coordinates; $i = \overline{1,\ N_{A_{M,r}}}$; $N_{A_{M,r}}$ is the number of objects in which there is a moth; R_{max} is the radius of the largest circle;

(6) $\exists\ a_s\ :\ \ a_s\ \epsilon$ spirals $\left[a_1; a_{N_{A_{M,r}}}\right]$, i.e., there is at least one point between the extreme points;

(7) if the spiral is formed counterclockwise, then the angle of rotation of the spiral $\varphi > 0$ and cannot be negative; if, on the contrary, the spiral is formed clockwise, then the angle of rotation of the spiral $\varphi < 0$ and cannot be positive;

(8) if t_1 is the time during which the deceptive system supports actions and determines steps within one coordinate system (Figure 3), then $t_0 = 0$ i $\lim\limits_{t=t_1,t_2,t_3,\ldots,t_p,t_0} t = 0$ (p is

the number of the penultimate iteration of time determination), $t_1 > t_2 > t_3 > \cdots > t_p > t_0$;

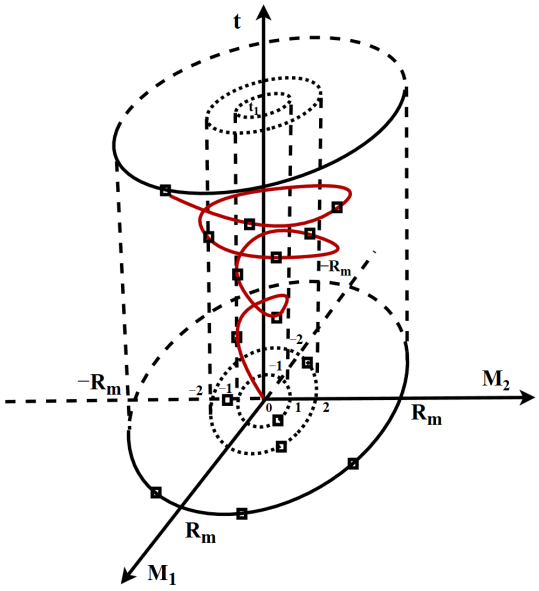


Figure 3. Potential spiral with time.

(9) update matrix B_M (Formula (11)), taking into account the requirement of clause 5;

(10) $\forall\, x_{l,i}, x_{l+1,i}\ : (x_{l,1,R_s})^2 + (x_{l,2,R_s})^2 \le {R_s}^2$; $\left(x_{l+1,1,R_{s-1}}\right)^2 + \left(x_{l+1,2,R_{s-1}}\right)^2 \le {R_{s-1}}^2$; l is the number of coordinates; $i = \overline{1,\ N_{A_{M,r}}}$; $N_{A_{M,r}}$ is the number of objects in which there is a moth; R_s is the radius of the larger circle; R_{s-1} is the radius of the smaller circle; and $[x_{l,i}; x_{l+1,i}]$ is the spiral arc;

(11) update matrix A_M (Formula (9)), taking into account the requirement of clause 9;

(12) if matrix A_M (Formula (9)) has more than one row, i.e., more than one moth, then taking the upper rows of the matrix as those corresponding to the first in the sequence of moths and the lower rows as the last, i.e., the sequence of moths is formed by numbers $i = \overline{1,\ N_{A_{M,r}}}$ ($N_{A_{M,r}}$ is the number of objects in which there is a moth) from the first to the last, then the difference of distances, taking into account the coordinates from matrix A_M of each moth and the coordinates of the center, will form an increasing sequence;

(13) if matrix A_M (Formula (9)) has more than one row, then the number of solutions for the sequence of moths decreases from the first to the last, which are given by the numbers $i = \overline{1,\ N_{A_{M,r}}}$ ($N_{A_{M,r}}$ is the number of objects in which there is a moth) from the first to the last;

(14) updatematrix B_M (Formula (11)) with its division into two or more matrices;

(15) update matrix B_M (Formula (11)) with its division into two or more matrices and update matrix A_M (Formula (9)).

The introduced rules 1–15 correspond to the algorithm of discrete optimization of moths and flames and allow the formation of variants of steps or nodes, in particular with baits and traps, into which the moth can move during movement along the spiral. There can be several such steps.

Taking into account rules 1–15, we define the function Q as follows:

$$Q(a_k) = \begin{cases} \varphi = \varphi + \theta, \ \ if\ M_k = \varnothing; else \\ m_s, \ \ if\ M_k \neq \varnothing \ \ and \min\limits_{M_k} F_{os}(v_{os,i}, v_{os,i,u}), \end{cases} \quad (20)$$

where $m_s\ \epsilon\ M_k$; F_{os} is the objective function for evaluating selected options; $v_{os,i,u,j}$ is the step of the deceptive system with baits and traps; $j = \overline{1,\ N_{os,i,u}}$; $N_{os,i,u}$ is the number of steps of deceptive systems with baits and traps to solve a certain task, which are defined by the system at the current time; i is the i-th sequence of steps to solve a certain task of the system; φ is the angle of rotation of the spiral from the extreme value of the angle; θ is the value of the additional angle for the case where there are no options to choose from in the given segment; and u is the refinement of steps, which are given by the vector $v_{os,i}$.

Let us introduce into the scheme of Figure 2 a third axis, which will represent time. The geometric interpretation of the placement of corporate network objects in segments and the potential spiral, taking into account time, is shown in Figure 3. When the time that reflects completion of the actions of deceptive systems to select the next steps, including the selection of baits and traps, is reached, the flame is turned off. Otherwise, the flame changes coordinates, and the process continues in the new coordinates, i.e., the system is rebuilt. Thus, convergence to the local optimum is avoided.

Restructuring the flame location and the number of flames will affect restructuring of the moth location and, as a result, will contribute to achieving a better solution. The direction of each moth to a separate flame can be reflected by different coordinate systems with centers at the points of each separate flame.

In Figure 3, compared to Figure 2, an Ot axis is introduced that reflects time. This is necessary because the time of implementation of the cyberattacks or the actions of the malware must be such that the deceptive systems take this parameter into account when choosing their steps and maintaining the stability of the systems, reconfiguring them, and changing the next steps during this time, so as not to complete their steps ahead of schedule, before completion of the cyberattacks or the actions of the malware. The spiral in Figure 3 is directed towards the center of the coordinates, and it reflects a decrease in time, i.e., the time of the cyberattacks or the actions of the malware, taking into account the use of baits and traps, converging to zero, since any cyberattacks or actions of the malware have time constraints that may be different. For example, if the cyberattacks or the actions of the malware begin at time t_1, and assuming that the effects from them are noticed by the deceptive system, then it begins to choose its steps, in particular activation of decoys in the nodes of corporate networks in which potentially malicious effects are noticed. The activation of decoys is primarily necessary to direct attackers to false objects for attacks on the network. To interest attackers in the computer stations where the decoys and traps are installed, the deceptive system opens certain ports and begins to simulate communication to attract the attention of attackers. As a result, the cyberattacks spread to new nodes of corporate networks. Some of the computer stations are blocked by deceptive systems or closed after manifestation of activity during cyberattacks or actions of malware. The deceptive system thus directs the attacker to a trap located in the center of coordinates, changing its steps in the process of performing the cyberattacks or the malware actions by the attacker. If complications arise for the deceptive systems in terms of blocking the cyberattacks or malware actions or their detection, the deceptive systems change the location of the trap, thereby prompting the attacker to rebuild the vector and plane of the cyberattacks. In the new coordinate system, the deceptive systems repeat some of their actions and update some. For example, they introduce another center of coordinates, bifurcating and dissociating the attacker into carrying out two cyberattacks.

Since the attacker, according to the constraints of the task, performs a dual-purpose cyberattack, i.e., a cyberattack with the aim of detecting both real and false objects in corporate networks, the second attempt after rebuilding the decoys and traps in corporate networks cannot take into account the previous step, because identification of false objects did not fully occur. This is due to the fact that the deceptive system estimated the duration of the cyberattacks or the actions of the malware and performed the reconstruction, as a result of which the center of the coordinate system shifted to one of the circles in which the decoy and trap will be activated, and the remaining nodes will be blocked by the deceptive system. In this case, the previous center of the coordinate system will be displayed on a circle, for example, of the same radius, or the deceptive system will distribute the objects located in one coordinate system into two coordinate systems, thus forming two flames.

The introduced rules 1–15 correspond to the algorithm of discrete optimization of moths and flames and allow the formation of variants of steps, or nodes, in particular with baits and traps, into which the moth can move during movement along the spiral. There can be several such steps. Let us define them formally (the elements are relabeled) by the set $M_k = \left\{ m_1, m_2, \ldots, m_{N_{M_k}} \right\}$, in which $m_{N_{M_k}}$ is the number of steps and $q = \overline{1, m_{N_{M_k}}}$. This set is contained in the local sector, if we reflect the geometric interpretation of the execution of rules 1–15 with respect to the variants of steps that can be the next among all the steps that are available. Figure 4 shows the elements of set M_k.

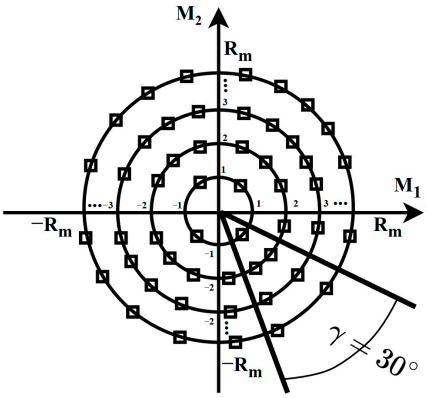


Figure 4. Location of the elements of set M_k.

All elements of set M_k must be in the sector given by angle φ. Angle φ is determined from the ray that starts at the origin and passes through the current point where the moth is planning to move. Angle φ can be different, but the smaller it is, the faster the spiral will move towards the center of the coordinates and a change in origin will be necessary. If the angle is greater than 180°, for example, then the spiral will be gentler, and the approach to the center of the coordinates will be slower. It may be that, in the next segment, there is no element from set M_k, in which case we increase the angle by θ and repeat the actions.

If there are elements of set M_k in the sector, then we choose the one for which the objective function F_{os} of evaluating the selected options will have the smallest, i.e., the worst, value, which ensures continuation of the selection of options in the search space with a smooth approach to the center of the coordinates in the next sector.

The main steps (Figure 5) of the method of synthesis of population algorithms in the architecture of deceptive systems with baits and traps are as follows:

(1) data preparation and coding for the formation of a discrete search space (Formulas (1)–(8), Figure 2);
(2) formation of moth and flame matrices and their corresponding fitness functions (Formulas (9)–(12));

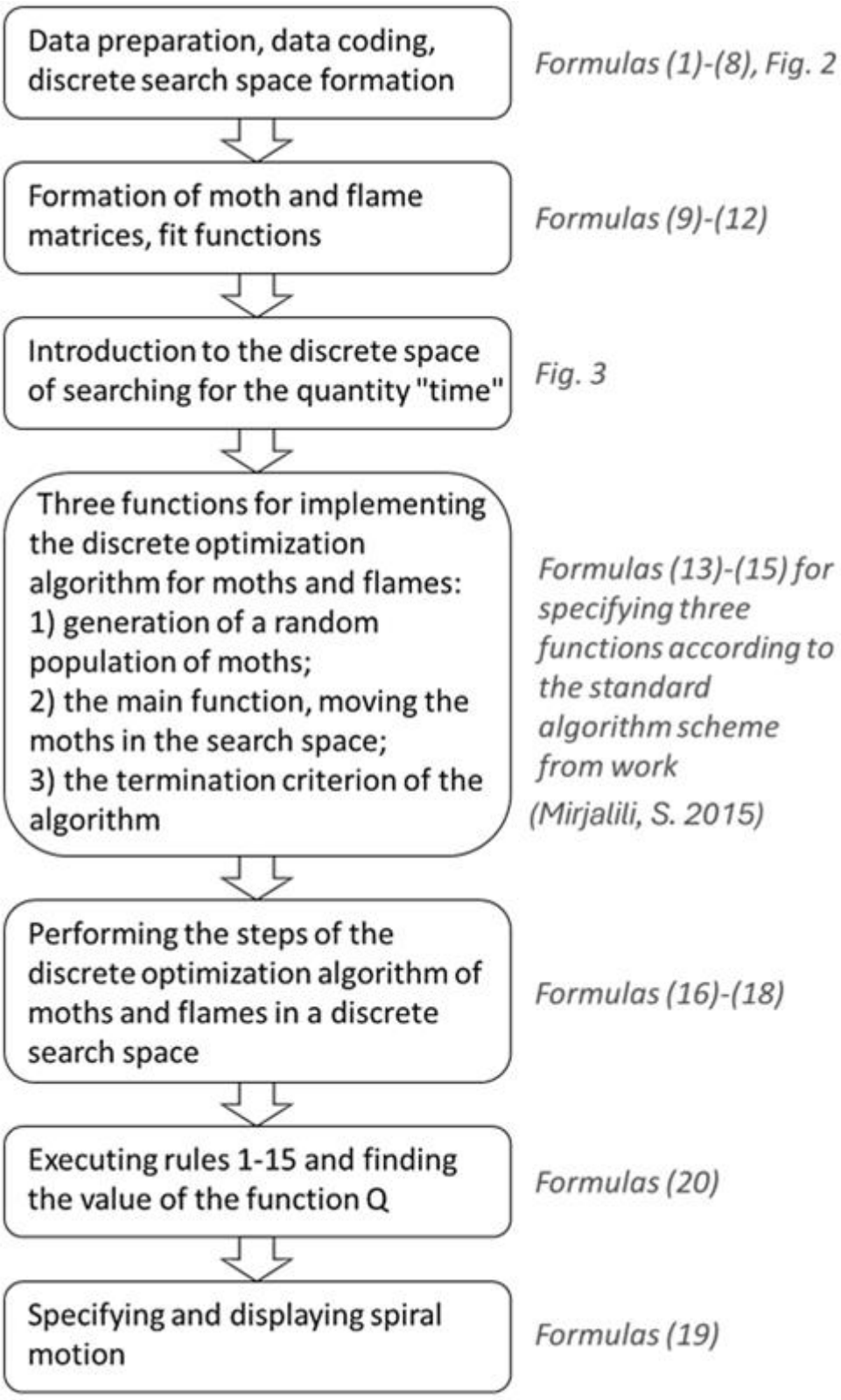


Figure 5. Main steps of the method [7].

(3) introduction of the time value to the discrete search space (Figure 3);
(4) we set three functions (Formulas (13)–(15)) for implementation of the discrete moth and flame optimization algorithm;
(5) according to the introduced functions from clause 4, according to Formulas (16)–(18), we perform the steps of the discrete moth and flame optimization algorithm;

(6) we perform rules 1–15 and find the value of the function Q (Formula (20)) for the formation of step options according to the given angle for the sector and values;
(7) we specify and display the movement along the spiral according to Formula (19).

The seven main steps of the method are not clearly determined in the context of sequential execution. Some of the steps, except for step 1, can be executed in parallel or repeatedly; that is, they will be called from certain other steps.

Thus, synthesis of the algorithm for combinatorial optimization of moths and flames in the architecture of deceptive systems with decoys and traps in corporate networks was carried out, including formation of a discrete search space with representation of objects by coordinates in dimensional spaces and synthesis of a spiral in a discrete search space based on the evaluation of potential options for the next steps and the set of elements in a certain sector, taking into account time in the context of the choice of steps, ensuring a change in the location of the flame and moth, which made it possible to ensure the selection and change of steps for the functioning of deceptive systems with decoys and traps for a long time during SC or the actions of the malware, to confuse attackers.

The spiral trace connects objects in a discrete search space. It reflects the movement of the moth towards the flame, which corresponds to the moth and flame algorithm, and it specifies the necessary steps for forming the spiral trace and changing the positions of the moth and flame. The algorithm also takes into account time as a parameter, which allows the system to record the construction of the spiral trace and, accordingly, the change of steps by the system. The formation of the spiral trace based on the angle for the sector in which the next steps of the system are located is studied, and rules for forming spiral segments are defined. To avoid rapid convergence in the local optimum, the discrete moth and flame optimization algorithm specifies steps for updating the placement of the flame and the moth, which allows the moth to stay around the flame for a long time and choose step options from the global optimum. Transverse orientation, a feature of this algorithm inspired by nature, is provided by the movement of the moth along a tangent to one circle or its transition to a new position in a circle with a smaller radius than the previous circle. If there is more than one moth, the first moth chooses the best options, and the following moths choose from the options that remain. Thus, the last moth chooses according to the residual principle. This enables further sorting by the degree of fit and assessment of the feasibility of moving the flame in a discrete search space.

The developed method of synthesizing population algorithms in the architecture of deceptive systems with baits and traps enables the creation of deceptive systems to confuse attackers in corporate networks.

4. Experimental Results

To confirm the ability to implement the proposed methods and architecture of deceptive systems, a deceptive system with decoys and traps was developed. The developed deceptive system can operate in corporate networks with a number of nodes equal to up to 2000 units of individual components, support communication and manage decoys and traps, activate decoys in its components, collect data in corporate network nodes, process network traffic indicators, make decisions about its next steps in the process of implementing cyberattacks by attackers, and detect cyberattacks of the denial of service type [36,37].

To ensure decision-making about its next steps, the system's decision-making center is organized according to the discrete optimization algorithm of moths and flames. The method for detecting cyberattacks of the denial of service type is implemented in one of the components of the deceptive system. The deceptive functions of the developed system relate to changes in the functioning of the system during the cyberattack or the established

actions of the malware, changing the nodes with the decision-making center of the system, reducing the activity of real objects in corporate networks, increasing the activity between the components of the system, including decoys, and activating decoys and traps.

Let us first set up an experiment with the developed deceptive system with decoys and traps in terms of its functions, which implement selection of the next steps of the system and their adjustment in the process of implementing the cyberattacks, controlling decoys and traps, and increasing and decreasing activity in the nodes of corporate networks.

The developed deceptive system complements the means of protecting corporate networks. It is aimed at performing the following actions:

(1) providing false objects for detecting cyberattacks or malware actions;
(2) activating false objects for cyberattacks (decoys: 37, false servers: 3);
(3) collecting data on cyberattack vectors and processes used by the attacker;
(4) collecting indicators of compromise to warn of potential cyberattacks;
(5) creating and removing false objects for attacks to confuse attackers.

Among the false objects for attacks, there are traps that save all the actions of the attacker during the attack on them. This helps to analyze the attacker's path. They do not simulate network interactions. Their presence delays attackers and distracts them, as well as encouraging them to waste resources and time. There are ten such objects in the installed system. The trap notifies the system center of any attempt to interact with it, that is, any indicator of an attack. From this, we can obtain details of the event, in particular the address, source and target port, interaction protocol, and response time.

In the deployed system, there are four decoys that simulate full-fledged operating systems, to emulate individual workstations and servers, and they are separate services.

Each component of the deceptive system is a program that is installed in computer stations and servers to communicate with the system center and among the rest of the components, execute commands from the system center, and transmit data to the system center. Thus, each component of the deceptive system performs four universal actions:

(1) communication with the system center;
(2) execution of system commands received from the system center;
(3) data transmission to the system center;
(4) communication with system components.

Also, the system component performs the following special actions:

(1) collection of data on the status of corporate network nodes in which it is installed;
(2) distribution of decoys according to commands from the system center;
(3) emulation of activity in a node in the network;
(4) response to events.

The system components are configured to run in a privileged mode to avoid detection by corporate network users. The decoys contain data to gain access to the trap, i.e., login, password, or key. Also, the decoys form a parallel environment of the corporate network to confuse the attacker. To do this, messages are exchanged precisely between them at a certain random time with a certain density. These messages contain prepared files in various formats that contain open and encrypted information [39,40].

Data collection by the system components is carried out during their installation and with a certain period of time. Such data include data on the installed software, its versions, and its installation dates. Such information can be compared with vulnerability databases.

To obtain information about the attack generated by the attacker from the middle of the corporate network perimeter, the deceptive system analyzes the contents of the trap with respect to time and the source port of the connection to these components, in order to establish the process that launched it.

To control real and false objects in corporate networks, the developed deceptive system performs the following actions:

(1) closes the possibility of communication with a computer station or server for a certain time until receiving a corresponding message with a command from the system center to the component;
(2) closes some of the ports in the computer station or server during the cyberattack or malware actions for a certain time until receiving a corresponding message with a command from the system center to the component;
(3) returns the possibility of communication with a computer station or server for a certain time until receiving a corresponding message with a command from the system center to the component;
(4) activates the decoy, in particular by creating a parallel false corporate network from several decoys to simulate activity;
(5) closes some of the ports in the computer station or server with the decoy during the cyberattack or malware actions for a certain time until receiving a corresponding message with a command from the system center to the component;
(6) opens all ports in the computer station or server with a decoy during the cyberattack or the actions of the malware for a certain time until receiving the corresponding message with a command from the system center to the component;
(7) activates the network node with the trap;
(8) directs the traffic of the cyberattacks or the spread of the malware to the trap;
(9) leaves the current state of the component and the computer station or server unchanged;
(10) activates the system center in the component;
(11) deactivates the system center in the component.

These actions are specified by a vector, as follows:

$$v_{os,r,i} = (v_{os,r,i,1}, v_{os,r,i,2}, \ldots, v_{os,r,i,11}), \quad (21)$$

where $v_{os,r,i,j}$ is the j-th coordinate of i-th vector and $j = \overline{1,11}$.

Actions according to Formula (21) concern exclusively one component. If we apply these actions to each individual component of the system, we obtain, in particular, for the experiment under consideration, 275 vectors. These vectors will determine the search space. Each vector will characterize a separate state of a separate component, and in general they will reflect the state of the entire system. Thus, the deceptive system, in the search space of its next steps, will choose sequences of steps that will concern the transfer of corporate network nodes to new states or leave them in current states according to the actions defined by Formula (21). All 40 decoys can be activated in the process of KA in the system components; that is, in the corporate network nodes in which they are installed.

Based on the results of the functioning of the developed deceptive system with decoys and traps, we will analyze the choice of the next steps in the decision-making process by the system center. Information about the nodes of corporate networks is given, in part, in Table 2. The coordinates of the location of the nodes of corporate networks are given, in part, in Table 3.

The formation about the spiral traces during the time of the system study is given by their coordinates in Table 4. The time during which the spiral traces were obtained is the third coordinate of the point in the coordinate three-dimensional space. The experiment lasted 2 h. Seventeen updates of steps took place for three separate impacts on the corporate network.

After each iteration of the moth and flame algorithm was performed in the center of the system, the system steps were updated. The number of spirals corresponds to the number of step updates.

Table 2. Partial information about corporate network nodes.

System Component Number	Node Type: Computer Station (CS); Server	Segment Number: 1–25	Host IP Address	Presence and Type of Active Bait and/or Trap	Node Localization to the Demilitarized Zone	Type of Public Service	Operating System	Presence of a System Center	Building Number
1	CS	1	10.0.1.10	–	–	–	Windows 10	–	1
2	CS	1	10.0.1.11	–	–	–	Windows 10	–	1
3	server	1	10.0.1.12	–	–	–	Windows 10	–	1
4	CS	1	10.0.1.13	–	–	–	Ubuntu 22.04	–	1
5	CS	1	10.0.1.14	–	–	–	Ubuntu 22.04	–	1
6	CS	1	10.0.1.15	–	–	–	Ubuntu 22.04	–	1
7	server	1	10.0.1.16	–	–	–	Windows 10	+	1
8	CS	1	10.0.1.17	–	–	–	Windows 10	–	1
9	CS	2	10.0.2.10	–	–	–	Windows 10	–	1
10	CS	2	10.0.2.11	–	–	–	Windows 10	–	1
11	CS	2	10.0.2.12	–	–	–	Windows 10	–	1
…	…	…	…	…	…	…	…	…	…
273	CS	25	10.0.25.17	–	–	–	Windows 10	–	3
274	CS	25	10.0.25.18	–	–	–	Windows 10	–	3
275	CS	25	10.0.25.19	–	–	–	Windows 10	–	3

Table 3. Coordinates of some corporate network nodes.

System Component Number	Segment Number: 1–25	Host IP Address	x_{1,R_m}	x_{2,R_m}	R_m	Action Number: 1–11
1	1	10.0.1.10	0.5403023	0.841471	1	3
2	1	10.0.1.11	−0.432178	0.9017883	1	6
3	1	10.0.1.12	−0.994367	0.1059875	1	5
4	1	10.0.1.13	−0.612548	−0.790433	1	7
5	1	10.0.1.14	0.3507973	−0.936451	1	3
…	…	…	…	…	…	…
273	25	10.0.25.17	21.44469	12.850108	25	4
274	25	10.0.25.18	10.000324	−22.91274	25	11
275	25	10.0.25.19	−24.00405	−6.986111	25	4

Thus, for 2 h of operation of the deceptive system, 17 new steps were obtained. Three different effects were applied to the objects in time, and the number of step updates made to process these effects is given in Table 4. The step vector contains 275 coordinates, each of which can take one of the eleven values from Formula (21).

Table 5 contains information about the number of steps that were updated in the general vector of steps for each update, which are given in Table 4.

Table 4. Coordinates of some spiral traces.

System Component Number	Segment Number: 1–25	Host IP Address	x_{1,R_m}	x_{2,R_m}	t, ms	R_m	Action Number: 1–11
			Track 1				
249	23	10.0.23.13	21.28941656	−8.704064725	18.432	23	3
237	21	10.0.21.17	17.41586021	−11.73404504	39.785	21	11
219	19	10.0.19.14	1.715469356	−18.9223985	61.204	19	7
198	17	10.0.17.22	6.878939022	−15.54606696	75.519	17	8
174	16	10.0.16.12	−1.846605705	−15.89308175	104.883	16	9
153	14	10.0.14.20	7.961602657	11.51576672	129.447	14	8
131	13	10.0.13.12	−0.336948651	12.99563256	171.902	13	5
			Track 2–16				
…	…	…	…	…	…	…	…
			Track 17				
221	18	10.0.18.23	−17.84158	−2.382869	6,869,327	18	4
171	16	10.0.16.11	−10.27336	−12.26613	6,931,140	16	7
170	16	10.0.16.10	−15.32255	−4.606453	6,984,023	16	6
115	12	10.0.12.10	10.126248	−6.438875	7,029,918	12	11
83	9	10.0.9.11	−5.688675	−6.974165	7,083,376	9	10
32	9	10.0.3.19	−1.112535	2.7860843	7,198,642	3	9

Table 5. Results of the experiment regarding the number of influences in the corporate network.

Number	Impact Number: 1–3	Cheating System Step Update Time, ms	Number of Steps Updated in the Total Step Vector
1	1111111	18,432	7
2	11111	236,926	5
3	111	562,043	3
4	111111	757,114	6
5	11111111	1,147,255	8
6	22222	1,667,443	5
7	2222222	1,992,560	7
8	222222222	2,447,725	9
9	22222222222	3,032,936	7
10	222222222	3,488,101	5
11	22222	3,813,218	7
12	3333333	4,268,383	8
13	33333333	4,788,571	10
14	3333333333	5,438,806	8
15	33333333	5,958,994	7
16	333333	6,414,158	6
17	3333333	6,869,327	7

Each impact corresponds to several spirals. The result of completing each impact was that information about the attack (impact) fell into the trap. In addition, during the attack, the corporate network nodes were scanned and, at the same time, the components of the system with decoys allowed scanning of the ports of the computer stations in which they were located, and, for a significant number of the computer stations, the ports were gradually closed. During the cyberattack, the components of the deceptive system communicated with each other to create a false corporate network and gradually removed their components from the external segments (outer circles), orienting the impacts to the internal segments (inner circles). The limitations of this experiment are the lack of change in the center of the corporate network node system, as well as the trap coordinates, since the main goal of the experiment was to test the system's ability to implement the selection of system steps in the corresponding components based on the moth and flame algorithm.

Figure 6 shows seventeen traces of spirals on the coordinate plane and 275 objects on 25 circles. Since the change in the state of objects occurred 17 times according to the data in Table 5, Figure 6 shows 17 different coordinate systems.

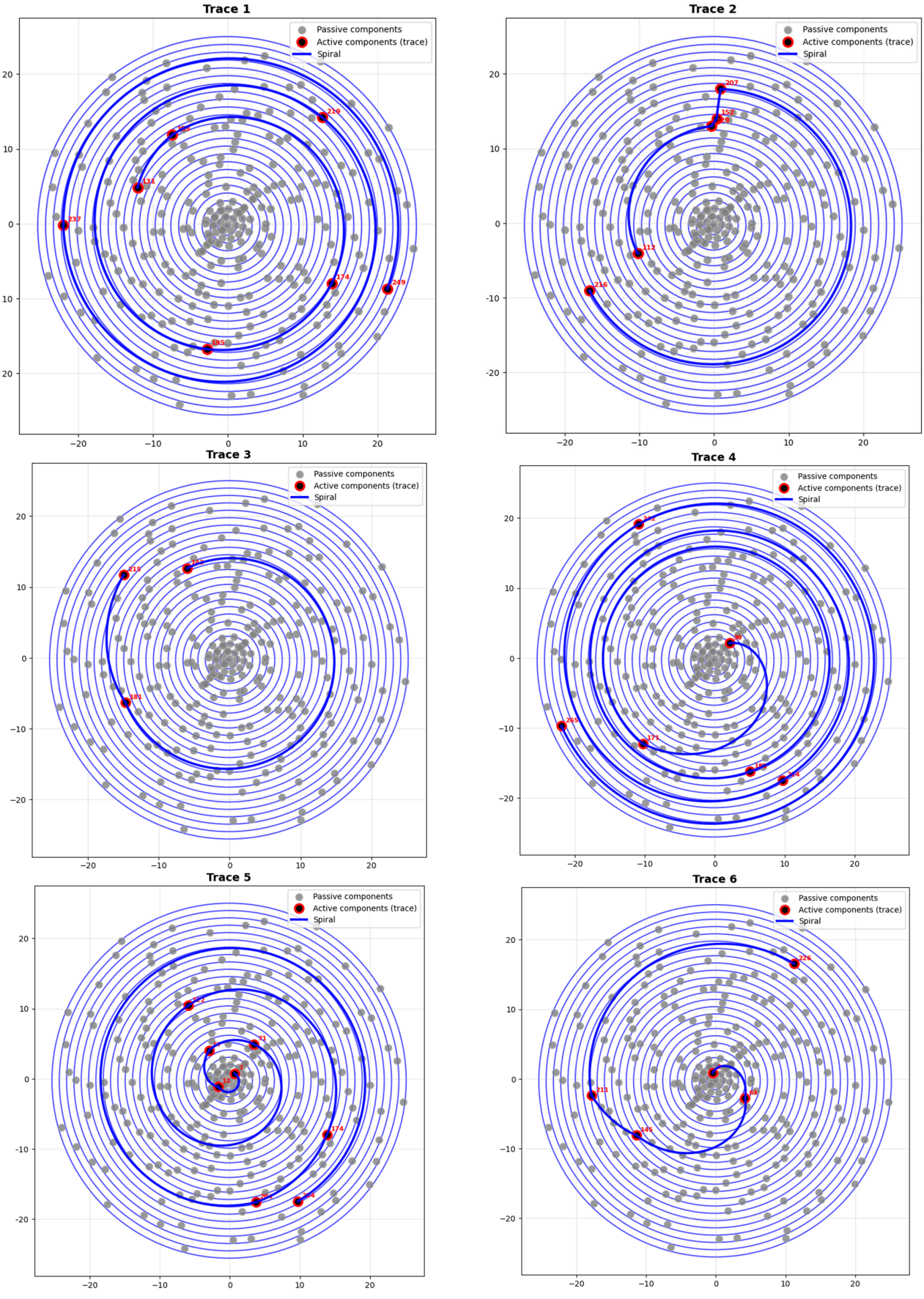


Figure 6. *Cont.*

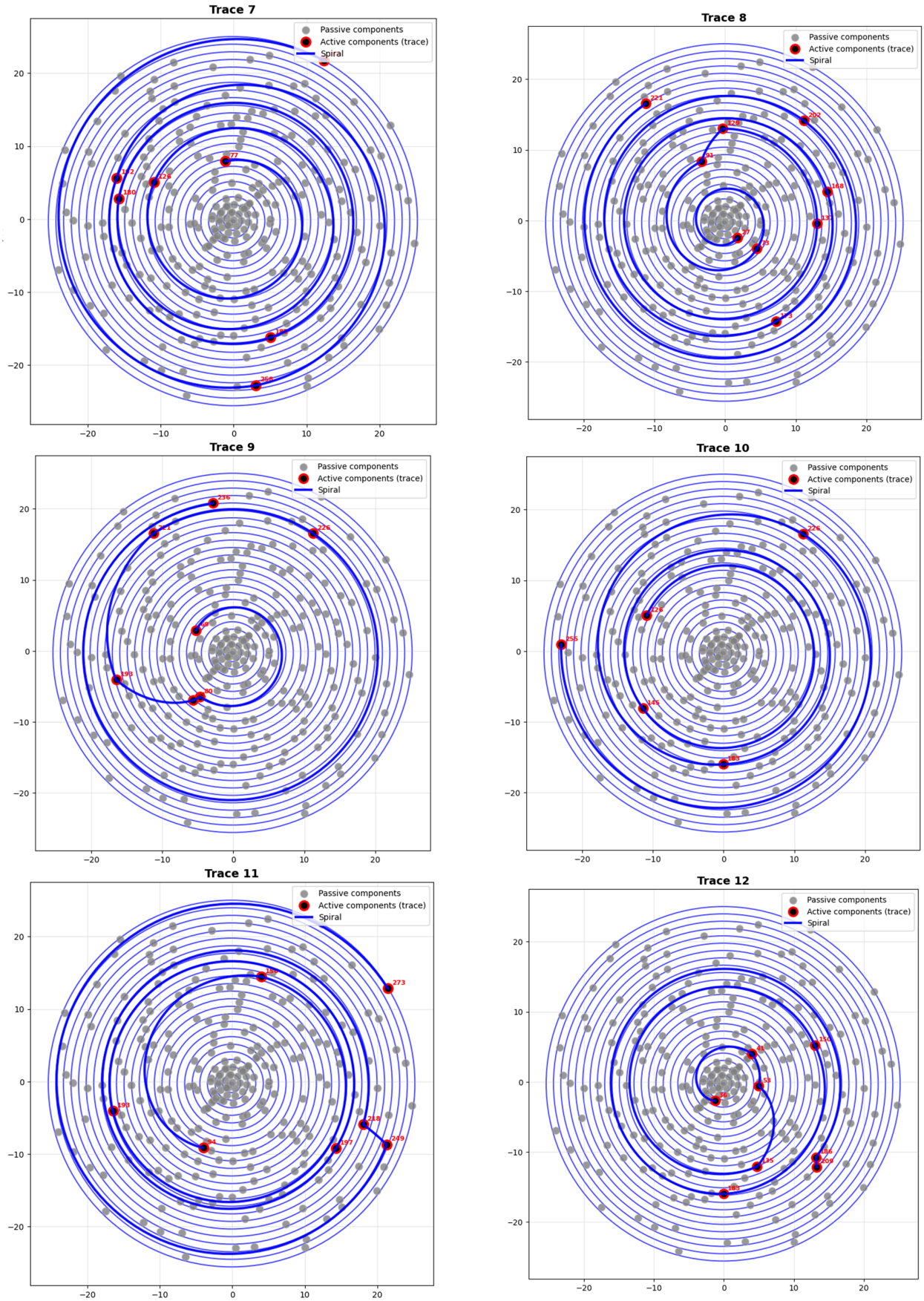


Figure 6. *Cont.*

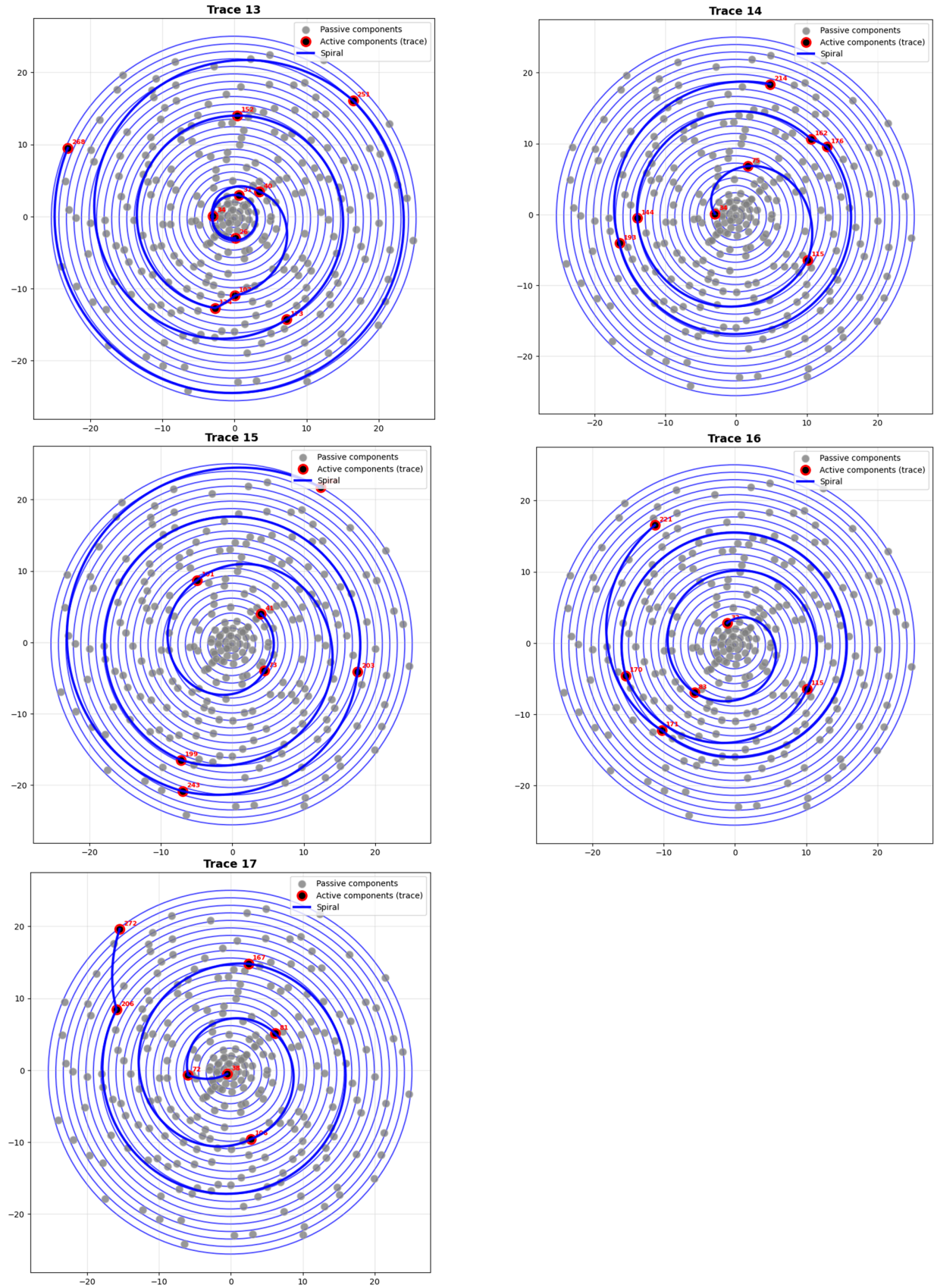


Figure 6. Seventeen traces of spirals on the coordinate plane.

The aim of the experiment is to quantify the effectiveness of the moth and flame (MFO) algorithm in controlling a decoy system with baits and traps and to compare it with alternative metaheuristic algorithms.

The moth and flame algorithm is designed with a focus on a discrete search space. In this case, it is necessary not only to evaluate its correct execution and make a comparison with other similar solutions, but also to evaluate its effectiveness for determining the next steps of deceptive systems, that is, the strategy of actions during a computer attack. Thus, research on the moth and flame algorithm should concern not only classical optimization of the function, but also how it selects and changes the next steps of deceptive systems during a computer attack, the impact on the behavior of attackers, avoiding convergence in local optima, and the functioning of deceptive systems in the dynamically changing environment of corporate networks during a computer attack. Therefore, to evaluate the synthesis of the moth and flame algorithm in the architecture of deceptive systems, it is not enough to use standard optimization metrics; it is necessary to develop a multi-criteria evaluation function and apply it to this algorithm and other algorithms that can be synthesized in the architecture of deceptive systems with baits and traps and perform the task of selecting the next steps of the systems.

Let us define the decision space as the set $M_{PR}^{E} = \left\{ m_{PR,1}^{E}, m_{PR,2}^{E}, \ldots, m_{PR,N_{M_{PR}^{E}}}^{E} \right\}$, where $N_{M_{PR}^{E}}$ is the number of elements in the set M_{PR}^{E} that form the decision space. Let us define the elements of the set M_{PR}^{E} as follows: $m_{PR,1}^{E}$ is the configurations of decoys and traps in corporate networks; $m_{PR,2}^{E}$ is the activation of decoys and traps; $m_{PR,3}^{E}$ is the sequence of steps of deceptive systems and reaction time; and $m_{PR,4}^{E}$ is the migration of attackers between nodes of corporate networks.

Since, for deceptive systems with baits and traps, the level of disorientation of the attacker, the time the attacker is kept in the deceptive environment, resource consumption, and problems in the event of disclosure of deceptive objects by the attacker are important, we will introduce an objective function using an additive criterion to evaluate the effectiveness of various algorithms, including the moth and flame algorithm, which are synthesized in the architecture of deceptive systems with baits and traps for choosing the next steps during a computer attack, as follows:

$$F_1^E(A) = \sum_{i=1}^{N_{F_1^E}} \alpha_i * q_i, \tag{22}$$

where A is the number of the algorithm for which the evaluation is performed; $N_{F_1^E}$ is the number of indicators taken into account during the evaluation; α_i is the weight coefficient for evaluating the weight of each i that is an indicator in the function $F_1^E(A)$; q_i is the value of i (that indicator); $i = \overline{1, N_{F_1^E}}$; $\sum_{i=1}^{N_{F_1^E}} \alpha_i = 1$; and $q_i \in [0, 1]$.

According to Formula (22), we will take the values of the indicators as follows: q_1 is the level of disorientation of the attacker; q_2 is the percentage of the time the attacker spent in the deceptive environment out of the total time during which the computer attack lasted; q_3 is the cost of resources used; and q_4 is the percentage of deceptive objects detected by the attacker in corporate networks. To establish the quantitative values of the indicators from Formula (1), we will introduce the metrics by which they will be determined.

We will determine the behavior of attackers, taking into account the following quantitative indicators:

(1) $t_{p,1}^{E}$—the average time for the attacker to establish the fact of an attack on the bait or trap;
(2) $n_{p,2}^{E}$—the number of steps the attacker took in the fraudulent environment of corporate networks;

(3) $q^E_{p,3}$—the proportion of repetitive actions, i.e., a sign of disorientation, out of all actions;
(4) $t^E_{p,4}$—the time during which the attacker acts, considering the corporate network environment and interaction objects to be real.

We define the indicators for algorithm optimization as follows:

(1) $w^E_{o,1}$—the average speed of finding effective strategies, i.e., choosing the next steps of deceptive systems;
(2) $n^E_{o,2}$—the average number of different options for choosing steps, i.e., options to avoid convergence in the local optimum;
(3) $q^E_{o,3}$—the proportion of selected steps out of all considered steps of deceptive systems;
(4) $n^E_{o,4}$—the proportion of selected repeated next steps from all steps considered, i.e., the frequency of being in a separate local optimum.

The average number of different step choices, i.e., options to avoid convergence in a local optimum, and the proportion of selected repeated next steps out of all steps considered, i.e., the frequency of staying in a particular local optimum, are critical in the context of evaluating population algorithms, in particular the moth and flame algorithm, and comparing them with each other.

To assess the security of corporate networks with existing deceptive systems that also manage baits and traps, we will introduce the following indicators:

(1) $n^E_{s,1}$—the number of real objects in corporate networks that the attacker did not notice during the computer attack;
(2) $t^E_{s,2}$—the time to detection of a computer attack;
(3) $q^E_{s,3}$—the percentage of computer attacks that were directed at traps and decoys.

To compare the moth and flame algorithm with other population algorithms, we will choose the following algorithms.

The peculiarity of particle swarm method PSO (particle swarm optimization) is that, to find the best solutions, each particle moves in the solution space, adjusting its speed and direction based on its own experience and the best experience of other particles in the swarm, which allows the system to gradually converge to the optimal value. It is suitable for the task under considerationthat is, for choosing the next steps of deceptive systems—-because it has fast convergence, and the particles have two parameters that characterize the direction and speed, which is suitable for controlling baits and traps.

The genetic algorithm (GA) provides proper global optimization and can be used in decoy systems with baits and traps, since the selection of the next steps can be based on modifying their sequence, which is permissible.

Ant colony algorithms (ACOs) can be used in the presence of paths, which is suitable for adaptation to the architecture of deceptive systems with baits and traps.

The multidimensional mathematical optimization method DE (differential evolution) belongs to the class of stochastic optimization algorithms; that is, it works using random numbers and uses some ideas from genetic algorithms.

Random/rule-based machine learning can be used, because the initial characteristic is that what is learned is based on rules. For consideration, we choose to implement such an approach from [29], in which the selection of the next steps of the deceptive systems is made based on rules.

For the correctness of the comparison, all algorithms were implemented separately in the architecture of the deceptive systems. Also, during the experiment, the same attack scenarios, the same resources, the same time constraints, and a large number of repetition series were provided.

Obtaining a static estimate will be based on the average value of the objective function $F^E_1(A)$ (Formula (22)) for each algorithm A, the variance for the values of the objective

function $F_1^E(A)$, and the duration of the countermeasure against computer attacks during its implementation.

The corporate network environment for the experiment contains 100 nodes, of which 20 are real objects and 80 are false objects. Computer attacks were simulated with the following stages: reconnaissance; privilege escalation; and horizontal movement. During reconnaissance, information about IP addresses and the software used was collected.

To increase privileges in the system, it was based on obtaining the role of a regular user. The implementation of horizontal movement consisted of attempts to move from one node to another.

The attacker model was based on the implementation of random processes with partial learning. For comparison, the analyzed algorithms were selected, which are presented in Table 6.

Table 6. Comparable algorithms.

Algorithm	Appointment
MFO	base
PSO	rapid convergence
GA	global optimization
ACO	route attacks
Random	basic level

The results are presented in Table 7.

Table 7. Results.

Algorithm	Objective Function Value	Standard Deviation	Computer Attack Countermeasure Support Time	Detection of Real Objects by an Attacker
MFO	0.87	0.03	78	1.2
PSO	0.81	0.05	45	2.7
GA	0.83	0.04	92	1.9
ACO	0.79	0.06	110	3.1
Random	0.62	0.08	—	5.6

Thus, the moth and flame algorithm has a spiral search trajectory, which provides a smooth transition, as well as avoiding premature convergence in the local optimum. Due to its use when choosing the next steps of deceptive systems with baits and traps, the choice is not limited to one solution. There is a change in the placement of objects in corporate networks, which confuses the attacker with its behavior, does not minimize the path, and converges in the local optimum less often. Also, this algorithm supports multivariate analysis, is more effective for long-term deception, and, unlike others, does not provide fast convergence and is more adaptive to the non-stationary behavior of computer attacks.

The PSO algorithm is rapidly convergent, which can lead to exhaustion of options with the choice of steps in a long-term computer attack that can continue. The genetic algorithm GA is resource-intensive compared to the moth and flame algorithm. Ant algorithms ACOs are focused on finding a path, not a strategy. The algorithm based on random rules does not allow the steps of deceptive systems to be changed in the process of long-term computer attacks.

For an adaptive attacker with limited memory and partial visibility, choosing the moth and flame algorithm has a lower probability of revealing deception than strategies optimized by algorithms with fast convergence, because the adaptive agent requires stationarity, and, according to the algorithm, such stationarity is violated.

5. Discussion

The obtained results indicate the feasibility of using population metaheuristic algorithms, in particular the moth and flame algorithm, to solve discrete optimization problems in the architecture of deceptive systems with baits and traps. Unlike traditional approaches to controlling such systems, which are often based on static scenarios or heuristic rules, the proposed method provides an adaptive choice of the sequence of steps, taking into account dynamic changes in the behavior of the attacker and the characteristics of the corporate environment.

An important feature of the developed approach is the formation of a discrete search space with coordinate representation of objects, which made it possible to correctly apply the moth and flame algorithm to problems that are not continuous by nature. The synthesis of the spiral trace of the movement of moths and flames in a discrete space allowed us to reduce the probability of premature convergence to local optima, which is a typical problem for many population algorithms in complex multidimensional problems.

Compared to existing methods for optimizing the control of deceptive systems, the proposed approach demonstrates better flexibility in adapting to the changing operating conditions of corporate networks. This is especially important in the context of countering modern computer attacks and malicious software, which are characterized by the rapid evolution of tactics, techniques, and procedures. The ability to quickly adjust the steps of decoys and traps increases the effectiveness of attack detection and makes it more difficult for an attacker to identify the deceptive nature of the system.

However, the results of the study indicate a number of aspects that require further analysis. In particular, the effectiveness of the algorithm significantly depends on the choice of population parameters and adaptation mechanisms, which can affect computational costs under real-time conditions. In addition, the integration of the proposed method into existing corporate networks requires taking into account performance limitations and compatibility with existing cyber security tools.

The main limitation of the approach is that it is inefficient when the number of baits is less than 5% of the total number of nodes or the number of nodes is less than ten.

In general, the results obtained confirm the prospects of using the moth and flame algorithm for the synthesis of intelligent adaptive deceptive systems. The proposed approach creates the basis for further development of autonomous systems to counter computer attacks, capable of long-term operation without constant administrator intervention and with increased resistance to the actions of attackers.

6. Conclusions

A method for synthesizing a discrete optimization algorithm for moths and flames has been developed and adapted to the architecture of deceptive systems with baits and traps. The proposed approach provides for the formation of a coordinate discrete search space and the creation of a dynamic spiral trace, which allows it to avoid premature convergence to local optima. Solving the problem of choosing the next steps of a deceptive system under conditions of variable parameters in the corporate environment is considered as an optimization problem aimed at increasing the efficiency of cyber threat detection.

The method allows an adaptive optimal sequence of steps for responding to the actions of an attacker to be found, which contributes to the creation of deceptive systems with increased autonomy, resistance to the disclosure of the principles of operation and minimal need for administrator participation. The experimental results confirm the effectiveness and prospects of using population algorithms in this context. The proposed approach performs 4.82% better than its closest competitor GA.

Further research should be aimed at deeper integration of population methods into the architecture of deceptive systems and development of mechanisms for their functioning, in accordance with the proposed conceptual model. The results obtained contribute to increasing the level of cyber protection of corporate networks by creating intelligent, adaptive means of countering modern attacks and malicious software.

Author Contributions: Conceptualization, O.S. and A.D.; methodology, O.S., A.D., B.S. and A.S.; software, B.S. and A.N.; validation, S.L. and A.N.; formal analysis, O.S., A.S., T.C. and B.R.; investigation, A.D., O.S. and B.R.; resources, B.S.; data curation, O.S. and T.C.; writing—original draft preparation, S.L.; writing—review and editing, S.L.; visualization, S.L.; supervision, O.S. and S.L.; project administration, O.S. and S.L.; funding acquisition, B.R. and T.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original data presented in the study are openly available at [3–5].

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

References

1. Kashtalian, A.; Lysenko, S.; Kysil, T.; Sachenko, A.; Savenko, O.; Savenko, B. Method and Rules for Determining the Next Centralization Option in Multicomputer System Architecture. *Int. J. Comput.* **2025**, *24*, 35–51. [CrossRef]
2. Savenko, B.; Kashtalian, A.; Lysenko, S.; Savenko, O. Malware Detection By Distributed Systems with Partial Centralization. In Proceedings of the IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Dortmund, Germany, 7–9 September 2023; pp. 265–270. [CrossRef]
3. Jayapandian, N. Securing cloud data against cyber-attacks using hybrid AES with MHT algorithm. *Int. J. Comput.* **2020**, *19*, 561–568. [CrossRef]
4. Balyk, A.; Karpinski, M.; Naglik, A.; Shangytbayeva, G.; Romanets, I. Using graphic network Simulator 3 for DDOS attacks simulation. *Int. J. Comput.* **2017**, *16*, 219–225. [CrossRef]
5. Kashtalian, A.; Lysenko, S.; Savenko, O.; Nicheporuk, A.; Sochor, T.; Avsiyevych, V. Multi-computer malware detection systems with metamorphic functionality. *Radioelectron. Comput. Syst.* **2024**, *1*, 152–175. [CrossRef]
6. Connolly, G.; Sachenko, A.; Markowsky, G. Distributed traceroute approach to geographically locating IP devices. In Proceedings of the Second IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Lviv, Ukraine, 8–10 September 2003; pp. 128–131. [CrossRef]
7. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl.-Based Syst.* **2015**, *89*, 228–249. [CrossRef]
8. Rotshtein, A.P.; Zelinska, O.V.; Kaminskyi, V.P. Optimization of Product Quality Indicators in the "Producer–Consumer" System Based on Fuzzy Cognitive Maps and Genetic Algorithm. *Cybern. Syst. Anal.* **2024**, *60*, 600–612. [CrossRef]
9. Shafronenko, A.Y.; Bodyanskiy, Y.V.; Holovin, O.O. Clusterization of data arrays based on the modified gray wolf algorithm. *Radio Electron. Comput. Sci. Control* **2023**, *1*, 73. [CrossRef]
10. Bodyanskiy, Y.V.; Pliss, I.P.; Shafronenko, A.Y. Clusterization of data arrays based on combined optimization of distribution density functions and the evolutionary method of cat swarm. *Radio Electron. Comput. Sci. Control* **2021**, *4*, 61. [CrossRef]
11. Bodyanskiy, Y.V.; Shafronenko, A.Y.; Pliss, I.P. Credibilistic fuzzy clustering based on evolutionary method of crazy cats. *Syst. Res. Inf. Technol.* **2021**, *3*, 110–119. [CrossRef]
12. Mirjalili, S. The ant lion optimizer. *Adv. Eng. Softw.* **2015**, *83*, 80–98. [CrossRef]
13. Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
14. Povkhan, I.; Leheza, A.; Mulesa, O.; Melnyk, O.; Kintonova, A. Method of hybrid logical classification trees based on group selection of discrete features. *Inform. Autom. Pomiary Gospod. Ochr. Sr.* **2025**, *15*, 15–21. [CrossRef]
15. Mulesa, O.; Bohdan, Y. Development of a fuzzy production model for assessing the degree of information security in international cooperation. *Technol. Audit. Prod. Reserves* **2024**, *6*, 6–10. [CrossRef]
16. Syberfeldt, A.; Lidberg, S. Real-world simulation-based manufacturing optimizations using cuckoo search. In Proceedings of the 2012 Winter Simulation Conference (WSC), Berlin, Germany, 9–12 December 2012; pp. 1–12. [CrossRef]

17. Sahoo, S.K.; Saha, A.K.; Ezugwu, A.E.; Agushaka, J.O.; Abuhaija, B.; Alsoud, A.R. Moth Flame Optimization: Theory, Modifications, Hybridizations, and Applications. *Arch. Comput. Methods Eng.* **2023**, *30*, 391–426. [CrossRef]
18. Sahoo, S.K.; Saha, A.K. A hybrid moth flame optimization algorithm for global optimization. *J. Bionic Eng.* **2022**, *19*, 1522–1543. [CrossRef]
19. Rotshtein, A.; Shtovba, S. Genetic Optimization of Multidimensional Technological Process Reliability. In *Computational Intelligence in Reliability Engineering. Studies in Computational Intelligence*; Levitin, G., Ed.; Springer: Berlin/Heidelberg, Germany, 2007; Volume 39. [CrossRef]
20. Shtovba, S.D. Ant Algorithms: Theory and Applications. *Program. Comput. Soft.* **2005**, *31*, 167–178. [CrossRef]
21. Wu, Y.; Chen, R.; Li, C.; Zhang, L.; Dai, W. An adaptive sine-cosine moth-flame optimization algorithm for parameter identification of hybrid active power filters in power systems. *IEEE Access* **2020**, *8*, 156378–156393. [CrossRef]
22. Wu, Y.; Chen, R.; Li, C.; Zhang, L.; Cui, Z. Hybrid symbiotic differential evolution moth-flame optimization algorithm for estimating parameters of photovoltaic models. *IEEE Access* **2020**, *8*, 156328–156346. [CrossRef]
23. Zhang, Z.; Qin, H.; Yao, L.; Liu, Y.; Jiang, Z.; Feng, Z.; Ouyang, S. Improved multi-objective moth-flame optimization algorithm based on R-domination for cascade reservoirs operation. *J. Hydrol.* **2020**, *581*, 124431. [CrossRef]
24. Wang, M.; Chen, H.; Yang, B.; Zhao, X.; Hu, L.; Cai, Z.; Huang, H.; Tong, C. Toward an optimal kernel extreme learning machine using a chaotic moth-flame optimization strategy with applications in medical diagnoses. *Neurocomputing* **2017**, *267*, 69–84. [CrossRef]
25. Ma, L.; Wang, C.; Xie, N.; Shi, M.; Ye, Y.; Wang, L. Moth-flame optimization algorithm based on diversity and mutation strategy. *Appl. Intell.* **2021**, *51*, 5836–5872. [CrossRef]
26. Zhao, X.; Fang, Y.; Liu, L.; Li, J.; Xu, M. An improved moth-flame optimization algorithm with orthogonal opposition-based learning and modified position updating mechanism of moths for global optimization problems. *Appl. Intell.* **2020**, *50*, 4434–4458. [CrossRef]
27. Singh, T.; Saxena, N.; Khurana, M.; Singh, D.; Abdalla, M.; Alshazly, H. Data Clustering Using Moth-Flame Optimization Algorithm. *Sensors* **2021**, *21*, 4086. [CrossRef] [PubMed]
28. Kashtalian, A.; Lysenko, S.; Sachenko, A.; Savenko, B.; Savenko, O.; Nicheporuk, A. Evaluation criteria of centralization options in the architecture of multicomputer systems with traps and baits. *Radioelectron. Comput. Syst.* **2025**, *1*, 264–297. [CrossRef]
29. Dabba, A.; Tari, A.; Meftali, S. Hybridization of Moth flame optimization algorithm and quantum computing for gene selection in microarray data. *J. Ambient. Intell. Hum. Comput.* **2021**, *12*, 2731–2750. [CrossRef]
30. Li, Y.; Wu, F. Improved population intelligence algorithm and BP neural network for network security posture prediction. *Int. J. Distrib. Sens. Netw.* **2023**, *1*, 9970205. [CrossRef]
31. Bykovyy, P.; Kochan, V.; Sachenko, A.; Markowsky, G. Genetic Algorithm Implementation for Perimeter Security Systems CAD. In Proceedings of the 2007 4th IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Dortmund, Germany, 6–8 September 2007; pp. 634–638. [CrossRef]
32. Tahir, M.; Sardaraz, M.; Mehmood, Z.; Muhammad, S. CryptoGA: A cryptosystem based on genetic algorithm for cloud data security. *Clust. Comput.* **2021**, *24*, 739–752. [CrossRef]
33. Thabit, F.; Alhomdy, S.; Jagtap, S. A new data security algorithm for the cloud computing based on genetics techniques and logical-mathematical functions. *Int. J. Intell. Netw.* **2021**, *2*, 18–33. [CrossRef]
34. Alizadeh, A.; Gharehchopogh, F.S.; Masdari, M.; Jafarian, A. A hybrid multi-population optimization algorithm for global optimization and its application on stock market prediction. *Comput. Econ.* **2025**, *65*, 2133–2178. [CrossRef]
35. Nadimi-Shahraki, M.H.; Fatahi, A.; Zamani, H.; Mirjalili, S.; Abualigah, L.; Abd Elaziz, M. Migration-based moth-flame optimization algorithm. *Processes* **2021**, *9*, 2276. [CrossRef]
36. Zamani, H.; Nadimi-Shahraki, M.H.; Mirjalili, S. A Critical Review of Moth-Flame Optimization Algorithm and Its Variants: Structural Reviewing, Performance Evaluation, and Statistical Analysis. *Arch. Comput. Methods Eng.* **2024**, *31*, 2177–2225. [CrossRef]
37. Nadimi-Shahraki, M.H.; Zamani, H.; Fatahi, A.; Mirjalili, S. MFO-SFR: An Enhanced Moth-Flame Optimization Algorithm Using an Effective Stagnation Finding and Replacing Strategy. *Mathematics* **2023**, *11*, 862. [CrossRef]
38. Kashtalian, A.; Ścisło, Ł.; Rucki, R.; Lysenko, S.; Sachenko, A.; Savenko, B.; Savenko, O.; Nicheporuk, A. Control and Decision-Making in Deceptive Multi-Computer Systems Based on Previous Experience for Cybersecurity of Critical Infrastructure. *Appl. Sci.* **2025**, *15*, 12286. [CrossRef]
39. Abu Khurma, R.; Aljarah, I.; Sharieh, A.; Abd Elaziz, M.; Damaševičius, R.; Krilavičius, T. A review of the modification strategies of the nature inspired algorithms for feature selection problem. *Mathematics* **2022**, *10*, 464. [CrossRef]
40. Khurma, R.A. The Runge–Kutta Optimization Algorithm: A Comprehensive Survey of Methodology, Variants, Applications, and Performance Evaluation. *Arch. Comput. Methods Eng.* **2025**, *32*, 5075–5122. [CrossRef]
41. Qiang, X.J. Computer application under the management of network information security technology using genetic algorithm. *Soft Comput.* **2022**, *26*, 7871–7876. [CrossRef]

42. Kasapoglu, T.; Masso, A. Attaining security through algorithms: Perspectives of refugees and data experts. Theorizing criminality and policing in the digital media age. *Stud. Media Commun.* **2021**, *20*, 47–65. [CrossRef]
43. Mayouf, C.; Salhi, A.; Haidara, F.; Aroua, F.Z.; El-Sehiemy, R.A.; Naimi, D.; Aya, C.; Kane, C.S.E. Solving Optimal Power Flow Using New Efficient Hybrid Jellyfish Search and Moth Flame Optimization Algorithms. *Algorithms* **2024**, *17*, 438. [CrossRef]
44. Rezk, H.; Zaky, M.M.; Alhaider, M.; Tolba, M.A. Robust Fractional MPPT-Based Moth-Flame Optimization Algorithm for Thermoelectric Generation Applications. *Energies* **2022**, *15*, 8836. [CrossRef]
45. Fan, Q.; Ma, Y.; Wang, P.; Bai, F. Otsu Image Segmentation Based on a Fractional Order Moth–Flame Optimization Algorithm. *Fractal Fract.* **2024**, *8*, 87. [CrossRef]
46. Yang, W.; Zhu, X.; Nie, F.; Jiao, H.; Xiao, Q.; Yang, Z. Chaos Moth Flame Algorithm for Multi-Objective Dynamic Economic Dispatch Integrating with Plug-In Electric Vehicles. *Electronics* **2023**, *12*, 2742. [CrossRef]
47. Nadimi-Shahraki, M.H.; Fatahi, A.; Zamani, H.; Mirjalili, S.; Abualigah, L. An Improved Moth-Flame Optimization Algorithm with Adaptation Mechanism to Solve Numerical and Mechanical Engineering Problems. *Entropy* **2021**, *23*, 1637. [CrossRef] [PubMed]
48. Nadimi-Shahraki, M.H.; Moeini, E.; Taghian, S.; Mirjalili, S. DMFO-CD: A Discrete Moth-Flame Optimization Algorithm for Community Detection. *Algorithms* **2021**, *14*, 314. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.