Journal of Engineering Managment and Soft Computing (JEMSC)

Improving the Learner Classifier System with a Basic Memetic Algorithm for Rule-Based Problem Solving

Mohammadreza Dehghani Mahmoudabadi[1]✉ **and Nagmalsadat nabavizadeh**[2]

1. Corresponding Author Ph.D. Student in Computer Engineering Software, Faculty of Computer Engineering, Islamic Azad University of Maybod, Maybod, Iran, Email: m.r.dehghani.m.a@gmail.com
2. MSc student, Medical Biotechnology Research Center, Ashkezar Branch, Islamic Azad University of Ashkezar, Yazd, Iran, Email: phdmrdma@gmail.com

Article Info

Article type:
Research Article

Article history:
Received 21 October 2024
Received in revised form 10 December 2024
Accepted 20 November 2025
Published online 1 January 2026

Keywords:
Genetic Algorithm, Learning Classifier System, Memetic Algorithm and Optimization.

ABSTRACT

Memetic algorithms are used to optimize the expensive target performance. The evaluation of the current population number is conducted by searching in the previous generations and preserving the values by the memetic algorithm. A significant number of generations are required to find the optimal value of the objective function in rule-based systems. The learning classifier system is one of the methods of generating value and classification for law. Each rule includes a set of properties. The function of the learning classifier systems is based on the genetic algorithm that it is not possible to search and save the previous steps in order to find a better solution to the problem. In this article, the memetic algorithm is used to improve and optimize the learner classifier system. In the proposed system, the memetic algorithm is used to create a population to improve the learning classifier system in the state space. The efficiency, convergence speed, and standard deviation of the proposed method are revealed using the implementation. The results indicated that the proposed hybrid method of replacing the memetic algorithm in the learning classifier system can significantly speed up the system and improve the quality to maintain better rules according to the search of previous generations.

Cite this article: Dehghani Mahmoudabadi, M. & et al, (2026)., Improving the Learner Classifier System with a Basic Memetic Algorithm for Rule-Based Problem Solving. *Journal of Engineering Management and Soft Computing*, 12 (1). 48-58.

DOI: https://doi.org/10.22091/jemsc.2023.8700.1166

Publisher: University of Qom
DOI: https://doi.org/10.22091/jemsc.2023.8700.1166

1) Introduction

In 1975, John Holland published a seminal book on genetic algorithms in the context of adaptation in natural and artificial systems, which later became widely recognized. In 1976, Holland conceptualized the genetic algorithm as a cognitive system and provided a detailed description of what is recognized as the first Learning Classifier System (LCS) (Compiani et al., 1990). He subsequently introduced this framework in his work on adaptive cognitive systems. The algorithms of this system, referred to as a cognitive system, were designed as modeling tools to be applied to real and dynamic systems through a population of human-interpretable rules. The objective of the cognitive system was to develop a set of online machine-learning rules capable of adapting to the environment based on feedback, reward, and reinforcement learning, thereby producing behavior consistent with the real system. However, this initial implementation was considered overly complex and yielded inconsistent results (Dorigo, 1995a).

Beginning in the 1980s, Kenneth De Jong and Stephen Smith proposed a different approach to rule-based machine learning, in which learning was viewed as an offline optimization process rather than an online adaptation process. This new perspective more closely resembled a standard genetic algorithm that evolved independent rule sets (Riedl et al., 2022). Since then, LCS methodologies, inspired by the online learning framework and introduced by Holland at the University of Michigan, have become known as the Michigan-style LCS, while those inspired by Smith and De Jong at the University of Pittsburgh are referred to as the Pittsburgh-style LCS. In 1986, Holland developed what can be considered the standard Michigan-Style Learning Classifier System for the following decade (Dorigo, 1995b).

Interest in LCSs was revived in the mid-1990s due to two major developments: the advancement of reinforcement learning algorithms and the introduction of a Michigan-style architecture by Stewart Wilson (Wilson, 1995).

LCSs represent a class of rule-based machine learning methods that integrate a genetic algorithm component with supervised, reinforcement, or unsupervised learning mechanisms (Nakata et al., 2014). LCSs aim to identify sets of context-dependent rules that collectively encode predictive knowledge. They support behavior modeling, classification, data mining, regression, function approximation, and game strategy formulation, enabling complex solution spaces to be decomposed into smaller and more manageable subspaces (Pakraei & Mirzaie, 2018). Using rule-based agents, LCSs attempt to model complex adaptive systems to construct artificial cognitive systems (Booker et al., 1989).

Researchers have employed LCSs to improve performance in classification tasks and games. In this paper, a memetic-algorithm-based LCS is proposed, in which a memetic algorithm is used to enhance both the solution quality and the convergence speed of the LCS for rule-based problem solving. In the proposed system, a memetic algorithm exploits the dissemination of the final population results at each stage, combined with local search, to generate and achieve improved optimal values in subsequent generations. This algorithm has been applied to several tasks and has been shown to be effective in reducing the number of generations required for the convergence of LCS, while also achieving significant improvements in solution quality.

2) Research Background

Learning Classifier Systems

LCSs constitute a specific class of genetic-based machine learning approaches. An LCS formalizes the core elements of both classical and modern evolutionary algorithms. For simplicity, this discussion focuses on the Michigan-style architecture. In these systems, the matching operation enables online learning and adaptation to a dynamically changing environment. The model is assumption-light, making only limited hypotheses about the environment or the relationships among data patterns. Consequently, LCSs are capable of discovering and distributing complex and heterogeneous patterns without relying on prior domain knowledge. As a result, a set of related and potentially conflicting rules is produced, which can be interpreted as a form of fuzzy prediction (Hurst & Bull, 2001).

Another important characteristic of LCSs is their stochastic learning mechanism, which avoids the limitations of deterministic learning when addressing large and complex non-deterministic problems. Through both implicit and explicit pressures, the rule population simultaneously optimizes for generality and simplicity in a multi-objective manner. This selective pressure is unique to LCSs. More effective rules tend to appear more frequently in match sets, increasing their likelihood of being selected as parents and propagating their structures to offspring rules (Sigaud et al., 2009).

A major advantage of LCSs lies in their interpretability for data mining and knowledge discovery. By extracting knowledge from the dominant rule population, LCSs provide effective strategies for identifying important features and underlying relational patterns. These strategies are applicable to flexible programming paradigms, single- or multi-step problems, supervised, reinforcement, or unsupervised learning, binary or multi-class classification, discrete and continuous attributes, as well as balanced or imbalanced datasets (Orriols-Puig & Bernadó-Mansilla, 2008).

Despite these strengths, LCSs also suffer from several limitations. First, software availability is limited: only a small number of LCS implementations exist, and they are often not designed to be user-friendly or easily accessible to machine learning practitioners. Second, although LCSs are inherently interpretable, users are often required to analyze a large number of rules to fully understand the learned model. Third, there is relatively little theoretical work providing formal convergence proofs for LCSs, largely due to their inherent complexity and the stochastic nature arising from multiple interacting components (Tavana et al., 2023). Furthermore, despite explicit and implicit pressures toward generalization, LCSs may still suffer from overfitting due to excessive connectivity. LCS execution also involves a large number of parameters that must be considered or optimized. Aside from the maximum rule population size and the maximum number of learning iterations, many parameters are domain-dependent, making parameter optimization particularly challenging (Sigaud et al., 2008).

For the reasons outlined above, LCS algorithms are rarely considered in direct comparison with other machine learning approaches:

1. They constitute a complex algorithmic framework.
2. Rule-based modeling represents a paradigm that differs substantially from most conventional machine learning methods.
3. Software implementations of LCSs are relatively uncommon.

From a computational perspective, LCSs are expensive (Shi et al., 2023). For simple or linearly separable learning problems, the use of LCSs is generally unnecessary. Instead, LCSs are best suited for complex problem spaces or domains in which little prior knowledge is available (Butz, 2002).

The architecture and components of a LCS are inherently variable. An LCS can be viewed as a device composed of multiple interacting components. These components may be added, removed, or modified, and existing algorithmic building blocks can be adapted or replaced to meet the requirements of a specific domain. This flexibility enables LCSs to operate effectively across a wide range of problem domains. Consequently, LCS theory can be adapted to many application areas that require machine learning solutions (Dorigo, 1995c).

In an LCS, the source environment consists of the data from which the system learns, typically provided as an online training dataset. The system includes independent variables, where each training instance comprises a set of features, and dependent variables representing target endpoints, such as class labels, actions, phenotypes, or predictions. Feature selection is an integral component of LCSs; therefore, not all training features are necessarily informative. The set of feature values for a single instance is referred to as the state. For simplicity, a domain with Boolean or binary features and a binary class is often assumed. In Michigan-style systems, learning is performed incrementally using a single instance from the environment at each learning cycle. In contrast, Pittsburgh-style systems employ batch learning, where the entire training dataset is evaluated by the rule set at each iteration (Shankar & Louis, 2005).

One of the most time-consuming components of an LCS is the matching process. The first step in the LCS learning cycle involves obtaining a single training instance from the environment and presenting it to the population for matching. In the second step, each rule in the population is compared

with the instance to determine whether it matches. In the third step, matching rules are transferred to the action set. In the fourth step, under supervised learning, the action set is divided into correct and incorrect condition sets (Gárate-Escamila et al., 2020). If, at this stage, no rule is able to classify the instance, such as when the population is empty, a covering mechanism is applied. In the sixth step, the parameters of each rule in the action set are updated to reflect the new experience obtained from the current training instance. In the seventh step, a sharing mechanism is typically applied. Subsequently, in the eighth step, the LCS employs an elitist genetic algorithm in which two parent classifiers are selected based on the principle of survival of the fittest. Parent selection is performed from the match sets used during competition. The final step in a general LCS learning cycle is enforcing the maximum population size constraint (Santos et al., 2009).

These steps are repeated for a user-defined number of training iterations or until predefined termination criteria are satisfied. Upon completion of training, the rule population typically consists of a compact set of experienced rules, with weak, redundant, or inexperienced rules removed ("Introduction to Optimization," 2004). The output rule compaction mechanism of an LCS is then applied to the classified population to construct predictions for unseen data. Individual LCS rules collectively form the predictive model, and their ranked parameters can be manually inspected for interpretability and analysis ("The Binary Genetic Algorithm," 2004).

Memetic Algorithm

The memetic algorithm has been demonstrated as an effective approach for solving optimization problems, particularly in exploring large and complex search spaces in an adaptive manner guided by biologically inspired evolutionary mechanisms, such as reproduction, crossover, and mutation. Algorithms derived from this paradigm are grounded in Darwin's theory of survival of the fittest and operate based on stochastic search processes. Memetic algorithms work with a large number of candidate solutions, which are initially generated at random (Tseng et al., 2007).

The underlying objective is to maximize adaptation by selecting the most suitable individuals from the population and exploiting their memetic information during mating operations to generate a new population of solutions. Numerous fundamental variants of the memetic algorithm have been developed; however, the LCS implemented in this study adopts a simple memetic algorithm, as described in the literature (Bereta, 2019).

Memetic algorithms offer several advantages over traditional optimization techniques. In particular, they do not require gradient information or functional derivatives and rely solely on performance evaluations. Memetic algorithms are capable of identifying global optima because they search multiple candidate solutions rather than a single point, thereby enabling effective exploration of design spaces that include combinations of continuous and discrete variables. Moreover, instead of producing a single solution, memetic algorithms provide decision-makers with a set of acceptable optimal solutions from which the most appropriate one can be selected. The probabilistic nature of memetic algorithms helps prevent premature convergence to local optima (Cotta & Moscato, 2003).

One of the main drawbacks of memetic algorithm techniques is that, although they typically exhibit strong initial convergence as global optimization methods, their performance may significantly slow down once an optimal region of the solution space has been identified. To address this issue and improve convergence, elitism is commonly employed. Elitist evolutionary algorithms generally achieve better convergence behavior than non-elitist counterparts (Zhang & Xing, 2018). In general, elitism preserves the best current solutions and transfers them to subsequent generations. In this work, elitism is implemented by directly carrying the best-performing solution of the population into the next generation.

In this paper, to further enhance the convergence of the LCS, a memetic algorithm incorporating both population-based evolution and local search is employed. This approach enables the identification of improved solutions relative to the current population and ensures their propagation to subsequent generations. As a result, the efficiency of the LCS is significantly increased (Zeng et al., 2018).

The proposed hybrid approach is developed by integrating the LCS framework with a memetic algorithm. By combining these two methodologies, the advantages of both are leveraged to produce a

fast and robust hybrid optimization strategy. The form of the memetic algorithm used in this study can be summarized as a simple binary-encoded memetic algorithm, employing roulette-wheel selection, uniform crossover, an elitist replacement strategy, and local search.

3) Research Methodology

Memetic Learning Classifier System

In this paper, a hybrid LCS is considered, which is based on integrating a conventional memetic algorithm into the LCS framework. In the proposed system, the efficiency of the LCS is enhanced by exploiting the capabilities of the memetic algorithm. By embedding a memetic algorithm within the LCS, the strengths of both approaches are combined to form a robust and fast hybrid optimization method. In the proposed system, the memetic algorithm is employed to guide the search toward optimal solutions. Figure 1 illustrates the flowchart of the proposed memetic-based LCS.

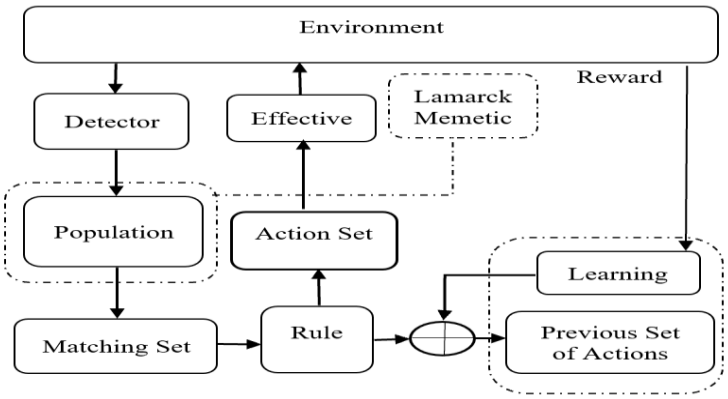
The procedure executed in the proposed algorithm involves the following components and processes: environment, rules, classification, population, matching, covering, parameter updating, fitness and accuracy evaluation, learning, hypothesis formation, rule discovery, memetic algorithm execution, and deletion.

The core idea of LCS is that the system interacts with the environment through both a detector and an effector (Parts 1 and 9). The system input, which enters through the detector (Part 2), is encoded in the form of messages and transferred to the population module as a message list. Within this list, if–then rules, serving as classifiers, are applied. The outcomes of classifier actions are re-encoded and written back into the message list. These newly generated messages (Parts 3, 5, 6, and 7) may trigger the activation of new rules or generate signals that prompt the effector to perform actions. The reward or payoff (Part 8) resulting from the executed actions is generalized and assigned to the rule population through the credit assignment mechanism. Subsequently, the rule discovery system (Parts 10 and 4) is responsible for identifying new rules and incorporating them into the classifier population.

Various implementations of LCSs exist; however, they generally share four fundamental functional components that define the system architecture:

1. A population of classifiers that represents the current knowledge of the system.
2. A performance component that manages the interaction between the environment and the classified population.
3. A reinforcement component, referred to as fitness or credit assignment, which distributes the reward received from the environment among different classifiers.
4. A discovery component that employs various operators to discover improved rules and enhance existing ones.

Figure 1) Diagram of the Proposed Hybrid Method



These modifications to the core components provide a fundamental framework known as a LCS. In this context, discovery via a memetic algorithm refers to the identification or introduction of rules that do not simultaneously exist in the current population and are obtained through local search combined with the preservation of the previous population. Ideally, newly generated rules are able to receive higher rewards than existing ones. Traditionally, this process has been conducted using a genetic algorithm. In the proposed approach, however, the memetic algorithm is employed as a computational search technique that evolves a population of individuals each representing a potential solution to a given problem through the incorporation of local search. In this system, the memetic algorithm is conceptually utilized as a core component of the LCS.

Memetic algorithms are inspired by ideas borrowed from nature. Drawing on natural selection theory and local search for evolutionary improvement, four principal analogies are employed:

1. A code used to represent the genome or condition.
2. The use of genome combination to represent an action.
3. A selection process for survival, in which rules with superior performance have a higher probability of reproduction and of transmitting genetic information to the next generation based on local search.
4. The application of genetic operators to discover new rule-organisms by selecting simple conditions from a pre-existing population.

The following steps are obtained through a single iteration of a simple genetic process:

1. Evaluation of all rules in the current population.
2. Search for and selection of parent rules from both the current and preserved populations.
3. Transformation of rules by applying crossover operators to the selected parents.
4. Addition of newly generated rules to the next generation.
5. Deletion of a sufficient number of rules from the next generation to maintain a fixed population size N.

The main idea of the proposed system is to replace the existing set of variables with a new set that yields a quantitatively improved objective function value. By incorporating local search and preserving the population from the previous stage, the proposed approach can lead to the improved convergence of the LCS. In this method, the memetic algorithm guides the search process and the evolution of the population toward optimal regions of the solution space, enabling the system to identify better optima across successive generations. It is worth noting that while the proposed system transfers the improved solution to the next generation, other individuals in the population maintain diversity, thereby preventing premature convergence.

In the next section, the rule-generation mechanism of the proposed system will be presented to demonstrate that the approach can significantly improve both the convergence speed of the LCS and the quality of the resulting solutions.

4) Findings

Performance of the Proposed System

To demonstrate the computational approach of the memetic-algorithm-based LCS, standard benchmark experiments were employed, and the obtained results were compared with those of a conventional memetic algorithm. All experiments were conducted using identical memetic algorithm parameters: a crossover probability of 80%, a mutation probability of 0.3%, and the same population size. The memetic-based LCS was evaluated under two experimental settings, consisting of 5,000 and 10,000 execution steps. In each case, different populations were generated randomly. To ensure result consistency and reproducibility, identical random number sequences were used within the algorithm.

Several factors must be considered when constructing a LCS. The final classified population is evaluated based on the following criteria:

Performance, that is, the evolved solution derived from the rule set.

Scalability, measured in terms of learning time or system size as the problem complexity increases.

Adaptability, referring to the ability of online learning systems to adjust to rapidly changing conditions.

Temporal efficiency, defined as the time required by the learning system to reach a satisfactory solution.

A significant portion of the experimental focus was devoted to optimizing system performance. The results indicate that the proposed memetic-algorithm-based LCS achieves substantially better solutions across all test cases within a limited number of generations. The training coverage and training accuracy obtained from experimental runs in the maze game environment for the proposed system are presented in Table 1.

Table 1) Performance Statistics

Iterations	Training Accuracy	Test Accuracy	Training Coverage	Test Coverage
5000	0.96875	–	1	–
10000	1.00000	–	1	–

Table 2 reports the population parameters defined for 5,000 and 10,000 iterations of the memetic-based LCS. The minimum number of steps represents the shortest path from the starting point to the goal, while the maximum number of steps reflects the longest agent trajectory from the start to the target. The values in Table 2 indicate the number of steps obtained during the experiments.

Table 2) Population Characteristics

Iterations	Macro Population Size	Micro Population Size	Overall Measure
5000	228	1000	0.5368333333333338
10000	209	1000	0.527166666666667

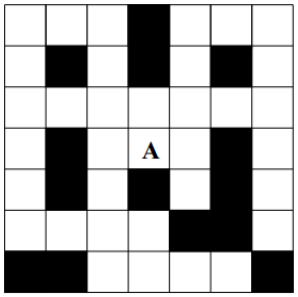
Table 3 presents the characteristics of the action and movement performance set of the memetic-algorithm-based LCS. The numerical values in the table represent the number of agent rules required to reach the target state.

Table 3) Action Set Characteristics

Iterations	A_0	A_1	C_0	C_1	C_2	C_3
5000	561	556	470	402	348	442
10000	564	516	443	421	418	475

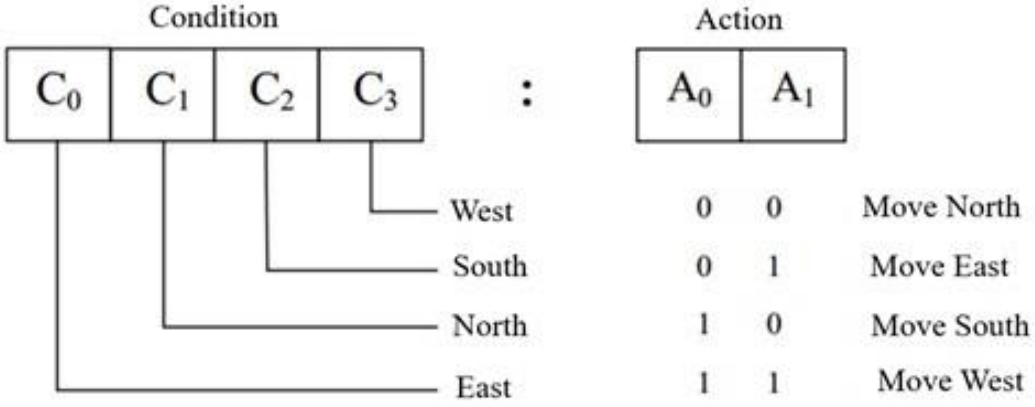
In this problem formulation, a starting point, a target point, and a set of obstacles are defined within the environment. The agent begins its movement from the starting point and navigates using four possible actions—up, left, down, and right—based on the governing problem-solving method, which in this case is the LCS. As the agent operates within its environment, it encounters obstacles and must navigate in such a way that it reaches the target without colliding with any obstacles.

Figure 2) Sample Maze Environment in Which "A" Denotes the Agent, and Black Squares Represent the Obstacles



The size (dimensions) of the maze environment is defined conventionally according to the problem under consideration. The encoding of classifiers for the agent navigation problem in the maze environment is illustrated in Figure 3.

Figure 3) Classifier Encoding for the Agent Navigation Problem in the Maze



In this encoding scheme, the value "1" in the condition part indicates the presence of an obstacle at the corresponding position. For example, the condition 0010 indicates that there is an obstacle to the south of the agent, while no obstacles exist to the north, east, or west. In this case, the agent moves toward the north (00).

Comparison of Results

As the number of rules increases, the agent reaches the target more quickly and achieves better performance. Table 4 presents the accuracy results obtained from 5,000 and 10,000 iterations of the memetic-algorithm-based LCS in the given environment, from the starting point to the goal. The minimum value represents the smallest number of agent rules required in an iteration, while the maximum value indicates the largest number of agent rules needed to reach the target in the environment. In this study, the maximum number of agent rules in the environment is set to 10,000. This is because, in the memetic-based LCS, the agent does not become trapped in loops or dead ends.

Table 4) Set Accuracy

Iterations	A_0	A_1	R_0	R_1	R_2	R_3
5000	475.2324	474.8925	382.3326	331.2291	284.4018	360.2260
10000	488.4703	454.4712	367.0733	354.5223	349.6856	396.8692

Table 5 reports the execution time of the system for 5,000 and 10,000 iterations in the given environment.

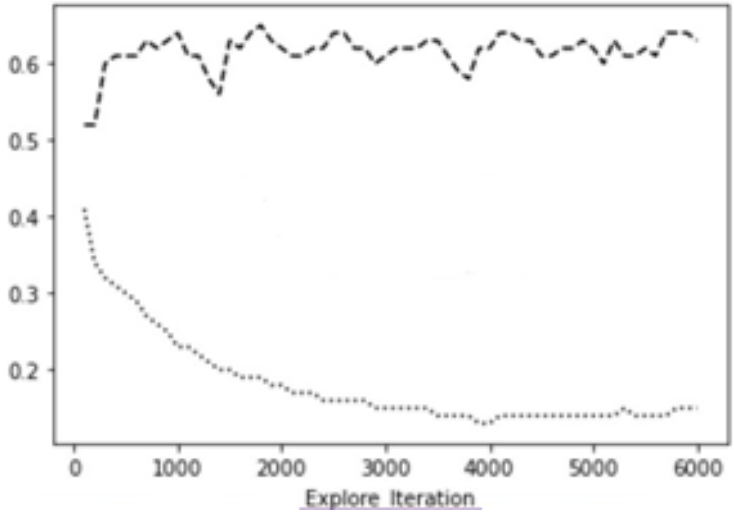
Table 5) Execution Time

Iterations	Evaluation	Selection	Sharing	Deletion	Matching	Total

5000	0.0032	0.0011	0.0038	0.0021	0.0219	0.0449
10000	0.0069	0.0023	0.0088	0.0093	0.0485	0.1034

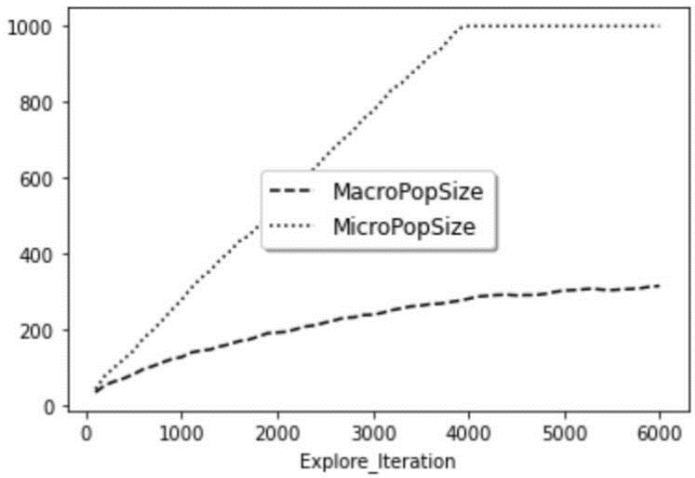
The results obtained in the experimental environment indicate a clear superiority of the memetic-algorithm-based LCS, particularly with fewer iterations. The proposed system demonstrates better performance in more complex environments, leading to improved solutions and superior overall results. The findings show that incorporating the memetic component into the LCS improves the average solution quality and significantly reduces the number of generations required to obtain an acceptable solution.

Figure 4) Convergence Speed of the Learning Classifier System Compared with the Memetic Learning Classifier System



It should be noted that the standard deviation of the test results obtained using the memetic-algorithm-based LCS is small.

Figure 5) Standard Deviation of the Learning Classifier System and the Memetic Learning Classifier System



This characteristic is particularly important for applying the algorithm to real-world problems, where cost and time constraints prohibit repeated execution of computationally intensive optimization procedures. The execution time and set accuracy results for the memetic-algorithm-based LCS presented here indicate that obtaining an appropriate optimal solution does not require repeated runs of a powerful optimization process.

5) Conclusions and Future Work

The proposed hybrid LCS is based on the integration of a LCS with a memetic algorithm. In the proposed framework, the efficiency of the LCS is enhanced by exploiting the local search capability and population memory preservation mechanisms of the memetic algorithm. In this hybrid approach, a memetic algorithm is embedded within the LCS to improve its convergence behavior during the search process.

The performance of the proposed method has been evaluated in a limited manner using several criteria, including performance statistics, population characteristics, dataset features, classification accuracy, and execution time. For each experimental scenario, 5,000 and 10,000 runs were conducted, where each run was initialized with a randomly generated population. The experimental results indicate that integrating a memetic algorithm into the LCS can play a significant role in improving both the convergence speed and the quality of the obtained solutions.

The application of the proposed system to selected real-world problems is currently under investigation by the authors. In addition, ongoing research focuses on developing extended versions of the proposed approach to address high-dimensional problem settings. It is anticipated that when the dimensionality of the problems increases substantially, certain modifications to the algorithmic structure will be required. In the present study, the memetic algorithm has demonstrated that local search and population memory preservation can be highly effective in improving the convergence of LCSs; however, the applicability of other types of LCSs may also be considered in future work.

References

Bereta, M. (2019). Baldwin effect and Lamarckian evolution in a memetic algorithm for euclidean steiner tree problem. *Memetic Computing*, *11*(1), 35–52.

Booker, L. B., Goldberg, D. E., & Holland, J. H. (1989). Classifier systems and genetic algorithms. *Artificial Intelligence, 40*(1–3), 235–282.

Butz, M. (2002). Biasing exploration in an anticipatory learning classifier system.

Compiani, M., Montanari, D., & Serra, R. (1990). Learning and bucket brigade dynamics in classifier systems. *Physica d Nonlinear Phenomena*, *42*, 202–212.

Cotta, C., & Moscato, P. (2003). A memetic-aided approach to hierarchical clustering from distance matrices: Application to gene expression clustering and phylogeny. *Biosystems*, *72*(1), 75–97.

Dorigo, M. (1995a). ALECSYS and the AutonoMouse: Learning to control a real robot by distributed classifier systems. *Machine Learning*, *19*(3), 209–240.

Dorigo, M. (1995b). Alecsys and the AutonoMouse: Learning to control a real robot by distributed classifier systems. Machine Learning, 19(3), 209–240.

Dorigo, M. (1995c). Alecsys and the AutonoMouse: Learning to control a real robot by distributed classifier systems. Machine Learning, 19(3), 209–240.

Gárate-Escamila, A. K., Hajjam El Hassani, A., & Andrès, E. (2020). Classification models for heart disease prediction using feature selection and PCA. *Informatics in Medicine Unlocked*, *19*, 100330. https://doi.org/https://doi.org/10.1016/j.imu.2020.100330

Hurst, J., & Bull, L. (2001). Self-Adaptation in Learning Classifier Systems.

Introduction to Optimization. (2004). In Practical genetic algorithms (pp. 1–25). John Wiley & Sons, Ltd.

Nakata, M., Kovacs, T., & Takadama, K. (2014). A modified XCS classifier system for sequence labeling. In *Genetic and evolutionary computation conference* (pp. 565–572).

Orriols-Puig, A., & Bernadó-Mansilla, E. (2008). Learning classifier systems in data mining. *Studies in Computational Intelligence*, *125*(July), 123–145.

Pakraei, A. R., & Mirzaie, K. (2018). The introduction of a heuristic mutation operator to strengthen the discovery component of XCS. *Journal of Advances in Computer Research*, *9*(1), 51–70.

Riedl, M. A., Johnston, D. T., Anderson, J., Meadows, J. A., Soteres, D., LeBlanc, S. B., Wedner, H. J., & Lang, D. M. (2022). Optimization of care for patients with hereditary angioedema living in rural areas. *Annals of Allergy, Asthma & Immunology*, *128*(5), 526–533. https://doi.org/https://doi.org/10.1016/j.anai.2021.09.026

Santos, M. F., Mathew, W., Kovacs, T., & Santos, H. (2009). A grid data mining architecture for learning classifier systems. *WSEAS Transactions on Computers*, *8*, 820–830.

Shankar, A., & Louis, S. (2005). Learning classifier systems for user context learning. *2005 IEEE Congress on Evolutionary Computation*, *3*, 2069-2075.

Shi, Y., Li, L., Yang, J., Wang, Y., & Hao, S. (2023). Center-based transfer feature learning with classifier adaptation for surface defect recognition. *Mechanical Systems and Signal Processing*, *188*, 110001. https://doi.org/10.1016/J.YMSSP.2022.110001

Sigaud, O., Butz, M., Kozlova, O., & Meyer, C. (2008). Anticipatory learning classifier systems and factored reinforcement learning. 321–333.

Sigaud, O., Butz, M. V., Kozlova, O., & Meyer, C. (2009). Anticipatory learning classifier systems and factored reinforcement learning [Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics]. *LNAI*, 321–333.

Tavana, P., Akraminia, M., Koochari, A., & Bagherifard, A. (2023). An efficient ensemble method for detecting spinal curvature type using deep transfer learning and soft voting classifier. *Expert Systems with Applications*, *213*, 119290. https://doi.org/10.1016/J.ESWA.2022.119290

The Binary Genetic Algorithm. (2004). In Practical genetic algorithms (pp. 27–50). John Wiley & Sons, Ltd.

Tseng, H.-E., Wang, W.-P., & Shih, H.-Y. (2007). Using memetic algorithms with guided local search to solve assembly sequence planning. *Expert Systems with Applications*, *33*(2), 451–467.

Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computation*, *3*(2), 149–175.

Zeng, Z. Z., Yu, X. G., Chen, M., & Liu, Y. Y. (2018). A memetic algorithm to pack unequal circles into a square. *Computers and Operations Research*, *92*, 47–55.

Zhang, G., & Xing, K. (2018). Memetic social spider optimization algorithm for scheduling two-stage assembly flowshop in a distributed environment. *Computers and Industrial Engineering*, *125*, 423–433. https://doi.org/10.1016/j.cie.2018.09.007