DOI: 10.15276/hait 03.2020.1
UDK 004.021

Reducing the search area of genetic algorithm using neural network autoencoder

Oleksandr V. Komarov
Odessa National Polytechnic University, Odessa, Ukraine
ORCID: https://orcid.org/0000-0001-7651-6300

ABSTRACT

The article discusses the problem of developing a genetic representation for solving optimization problems by means of genetic algorithms. Traditionally, a genotype representation is a set of N features that defines an N-dimensional genotype space in which algorithm performs a search for the solution. Due to the non-optimal choice of features, the genotype space becomes redundant, the search area for a solution unnecessary increases, which slows down the convergence to the optimum, and leads to the generation of infeasible candidates for the constraints of the problem. The reason for this is the desire to cover all legal candidates for solution of the problem by the search area, since the optimum is feasible by the conditions of the problem. In constrained optimization problems, to find the optimum, it would be sufficient to cover only the area of feasible candidates that fall within the constraints specified by the problem. Since the set of feasible candidates is smaller than the set of all legal candidates, the search area may be narrower. The search area can be reduced by obtaining a more efficient set of features that is representative of the set of feasible solutions. But in the case of a small amount of domain knowledge, developing of an optimal feature set can be a nontrivial task. In this paper, we propose the use of feature learning methods from a sample of feasible solutions that fall under the constraints of the optimization problem. A neural network autoencoder is used as such a method. It is shown that the use of the preparatory stage of learning a set of features for constructing an optimal genotype representation allows to significantly accelerate the convergence of the genetic process to the optimum, making it possible to find candidates of high fitness for a smaller number of iterations of the algorithm.

Keywords: genetic algorithm; feature engineering; neural network autocoder; search area; optimization problem

For citation: **Komarov O. V. Reducing the search area of genetic algorithm using neural network autoencoder. *Herald of Advanced Information Technology*. 2020; Vol.3 No.3: 113–124. DOI: 10.15276/hait 03.2020.1**

INTRODUCTION

Genetic algorithms have proven to be effective optimization methods that provide directed random search on complex surfaces. By combining elements of directed and random search, genetic algorithms offer a good balance between exploration and exploitation of the search area.

However, it was shown that genetic algorithms demonstrate a low convergence rate in combinatorial optimization problems [1]. The resource intensity of genetic algorithms strongly depends on the size of the solution search area [2–3]. For a number of problems, the search area in genetic algorithms increases critically with an increase in the size of the input data, up to exponential growth [4].

Genetic methods work simultaneously in the genotype and phenotype spaces. The phenotype is the desired type of solution, and the genotype is its encoded representation. The search for solutions and their transformations are performed in the genotype space, whereas the assessment and selection of the obtained solutions are performed in the phenotype space. Genotype to phenotype mapping is performed using the decoding operation.

The phenotype space can be basically divided into areas of legal and illegal solutions. Any object of the phenotype space that can be considered as a solution to the problem is considered a legal solution.

Within the area of legal solutions the area of feasible solutions can be distinguished. In constrained optimization problems this area is defined by the constraints of the problem. The desired optimal solution is located within this area (*Fig. 1*).

Although fitness function can be determined on the genotype space, it usually can be calculated only in the phenotype space. In constrained optimization problems, the domain of definition of a fitness function is most often the domain of feasible solutions, and the constraints of the problem determine either the coefficients of the fitness function or additional terms that define the domain of feasible solutions.

When using genetic approaches, the choice of the search area is one of the important problems, along with other aspects, such as the choice of the selection strategy, the constitution of the initial state and size of the population, the mutation chance etc. A badly selected search area leads to the following problems: the appearance of illegal, infeasible, or equivalent genotypes that map to the same phenotypes.

In genetic algorithms, the main resource costs are the resource costs of calculating the fitness function [5]. Therefore, when developing genetic methods, it is efficient to aim to reduce the number of algorithm iterations, thus reducing the number of estimations of the fitness function. The presence of a large number of infeasible and equivalent genotypes significantly inhibits convergence, "stretching" it over a large number of iterations of the algorithm. Thus, to enhance the performance of the genetic algorithm, it is crucial to pay special attention to the choice of the solution search area.

LITERATURE REVIEW

There is a large number of works devoted to the problem of choosing a search area. The approaches used in them notably differ, but some general outlines can be pointed out: heuristic methods (elimination of infeasible solutions) [6–7], the development of new genetic operators [8–9], the development of selection and crossover strategies based on systems of penalties and rewards [10–11].

The main goal of such methods is the so-called feasibility preservation, an approach according to which, once the algorithm reached the area of feasible solutions; it must continue the search within it, avoiding generation of infeasible solutions. Such methods show good results, but they do not perform any operations on the search area itself. Thus, the risk of generation of the infeasible solutions remains.

There are methods that rely on preliminary investigation of the search area and the exclusion of knowingly infeasible areas [12]. Such techniques can be combined with heuristics and specific genetic strategies to improve performance. To limit such negativistic behaviour, the method has to retain any areas that have a chance of containing the optimum, or risk the elimination of it.

Much more popular is the opposite approach, which could be called positive. It focuses on exploring the area of feasible solutions in order to build a search area in it. This is usually accomplished by creating a new genetic representation.

It is shown that efficiency can be improved not only with the help of genetic operators, but also with the help of representations [13].

Traditionally, the search area is an N-dimensional space, where N is the number of features with which the genotype of the solution is encoded. An arbitrary vector in this space is considered as the genotype of the some solution. Non-optimal coding thus leads to unnecessary expansion of the search area.

The main method of genotype coding is the binary representation of the phenotype [14], but it has been shown that this method has a number of significant disadvantages [1]. The direction of research in this area concentrates on the search for a shorter set of features that are significant only for legal and feasible solutions. By constructing such a set of features, it is possible to reduce the search area to a set that excludes illegal and infeasible solutions, and also to close genetic operators on the desired subset of solutions.

The extraction of features can face many difficulties caused by the existence of hidden internal connections in these solution sets. An effective way to overcome this problem is to build knowledge-rich representations [15–16].

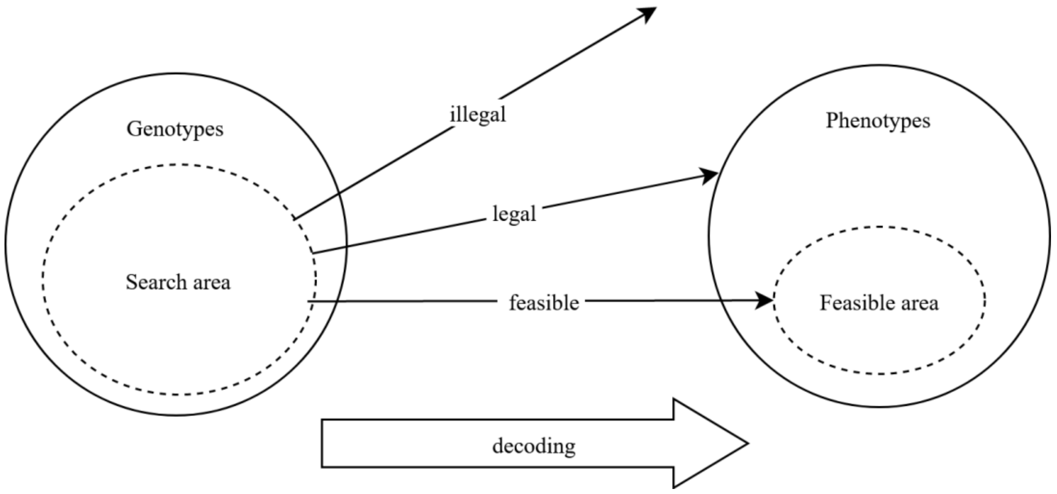

Fig. 1. **Mapping between from genotype to phenotype spaces**

This problem is successfully solved for special cases by developing specialized feasible representations [5–6; 17–22]. Such representations may differ in structure and may require the use of unique genetic operators [9], which does not allow to design a general approach.

An attempt to propose a universal representation in a restricted domain was made in [23]. However, this method still assumes possession of knowledge about the feasible solution area.

The use of domain-dependent knowledge is a significant disadvantage of all these methods, since it implies the need of involving experts, which can be expensive, or even impossible for some cases. There is a need for domain-independent methods for engineering an optimal set of solution features.

FORMULATION OF THE PROBLEM

A solution representation (or genotype representation) is a set of N features. This set defines the N-dimensional space of genotypes, which is mapped into the space of phenotypes using some function. In the classical genetic algorithm, the solution search area for is the entire genotype space.

A good genotypic representation can be characterized by five main properties: nonredundancy, legality, completeness, Lamarckian property and causality [1].

Three of these properties have a direct impact on the size of the search area: nonredundancy, legality and completeness. Nonredundancy requires the absence of equivalent genotypes (genotypes that map to the object in the phenotype space) in the search area. Legality requires that any genotype could mapped to a phenotype that can be solution of the problem. The completeness property requires that for every possible solution of the problem there is a corresponding genotype.

Although a legality and completeness property doesn’t formally contradict each other, representation development usually faces a conflict between them. Usually, especially in the absence of domain knowledge, the developer prefers completeness in order to provide a search across the entire set of possible solutions. Representations build in this way often violate the legality and nonredundancy requirements. It is this problem that developers usually try to overcome by designing new operators, strategies and heuristic improvements of the genetic process.

Note that if the illegality and redundancy of genotypes is a consequence of a redundant set of features, the nature of infeasibility originates in constraints of the optimization problem [1]. Note also that the desired optimum, by definition, is contained in the area of feasible solutions. Therefore, we can replace the completeness property with the weaker requirement of feasible completeness, which states that there is a genotype for every feasible solution. Having designed a new representation based on the weakened requirement, we can narrow the search area around the set of feasible solutions, and also ensure the closure of genetic operators in the area of legal (and ideally, feasible) solutions (*Fig. 2*).

To build such a representation without domain-dependent knowledge, it is necessary to apply the methods of feature learning.

One of the most popular feature learning methods is a neural network autoencoder. Neural networks built on such architecture are capable of non-supervised training, extracting only the most essential properties from the input data [24–28]. It is shown that the neural network autoencoder effectively solves the problem of learning features [29–30], and it is successfully used for a number of practical problems [31]. Applying a neural network autoencoder on a representative set of feasible solutions, we can learn some important features of solutions in this area, and use them to build a genotype representation.

Therefore, the goal of our study is to develop a genetic method for solving an optimization problem on a reduced search space generated by a set of features extracted from a sample of feasible solutions using a neural network autoencoder.

MAIN PART. METHOD DESCRIPTION

The proposed method consists of two stages. At the first stage, the solution features of the desired set are extracted. This stage involves the use of a neural network autoencoder with a narrow central layer, on which a set of genotype features will be learned during training. This is a preparatory step that is performed once for each given search area. At the second stage, the genetic algorithm is launched to solve the optimization problem.

A neural network autoencoder is an artificial neural network of symmetric architecture that learns to reproduce the data from the input layer on the output layer [24]. With a special layer called the central layer, the structure of the autoencoder is divided into two parts, an encoder and a decoder. By changing the size of the central layer, a more redundant or less redundant image of the input data can be obtained on it.

Figure 3 shows an example of an autoencoder with an input and output layers that consist of 4 X and 4 X' nodes and a centre layer of 2 Y nodes. Nodes Z and Z' constitute the internal hidden layers.

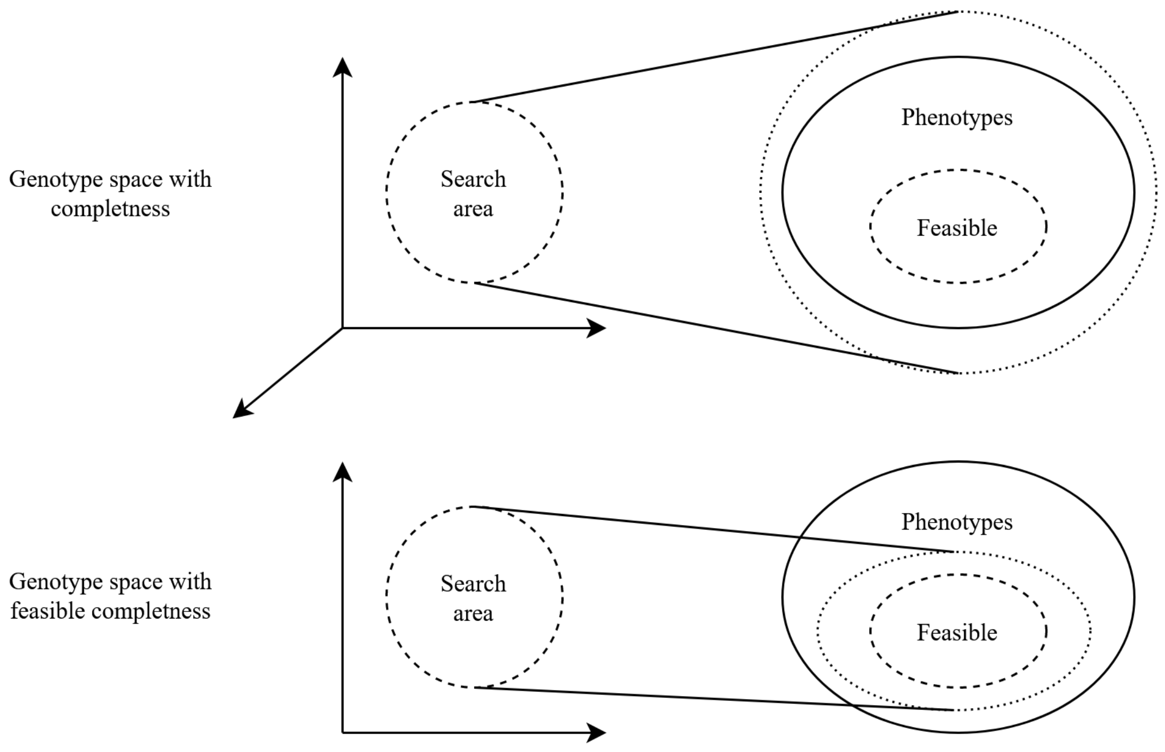
Genotype space with completness
Search area
Phenotypes
Feasible
Genotype space with feasible completness
Search area
Phenotypes
Feasible

Fig. 2. **Mapping of the search area into the phenotype space from the genotype space of high dimensionality with completeness (top) and the genotype space of low dimensionality with feasible completeness (bottom)**

Using of additional internal layers, along with the central one, has a positive effect on the training of the autoencoder: it provides better data compression and allows decreasing the amount of training data needed [24].

During the training of the autoencoder, all network nodes, except for the input layer X, are neurons, which are nonlinear functions *f* of the sum Σ of the set of input signals x_i weighted by the weights w_i (*Fig. 4*) [24]. However, after training, the network is disunites, and two new networks are formed from the trained layers: an encoder and a decoder.

The subnet from nodes X to nodes Y is an encoder. It can produce a compressed image of the input data. In the architecture of the autoencoder and in the architecture of the encoder derived from it, the nodes Y are neurons, the activation function of which is usually chosen depending on the desired type of output data image. For example, a sigmoidal activation function (followed by rounding) can be used to generate a binary vector image.

The subnet from nodes Y to nodes X' forms the decoder. In the decoder, the layer of Y nodes is not a layer of neurons, but instead replaced by an input layer. Due to this, the decoder is able to restore the original form of the input data image, or to obtain a vector close to it. In this regard, the activation function of layer X' is also selected in such a way as to best cover the range of values of the original dataset, which came to layer X during training of the autoencoder.

Reducing the redundancy of information on the input layer using architecture with a narrow central layer is achieved by identifying non-obvious structural relationships in the input data of the training set. Thus, the training sample should be representative of some class of solutions in which we want to find such connections.

We assume that in the legal (or feasible) area of solutions unites solutions of a certain type, the connection between which is not obvious. In order to correctly extract a set of features, we need a representative sample of legal (or feasible) solutions.

A training sample for a neural network autoencoder can be obtained using the binary representation of a set of feasible phenotypes. Now, having trained the autoencoder with a narrow center layer on this sample, we can use the center layer as a genotype representation, which is a binary feature vector. This feature vector as well as the trained encoder and decoder are the results of the first stage.

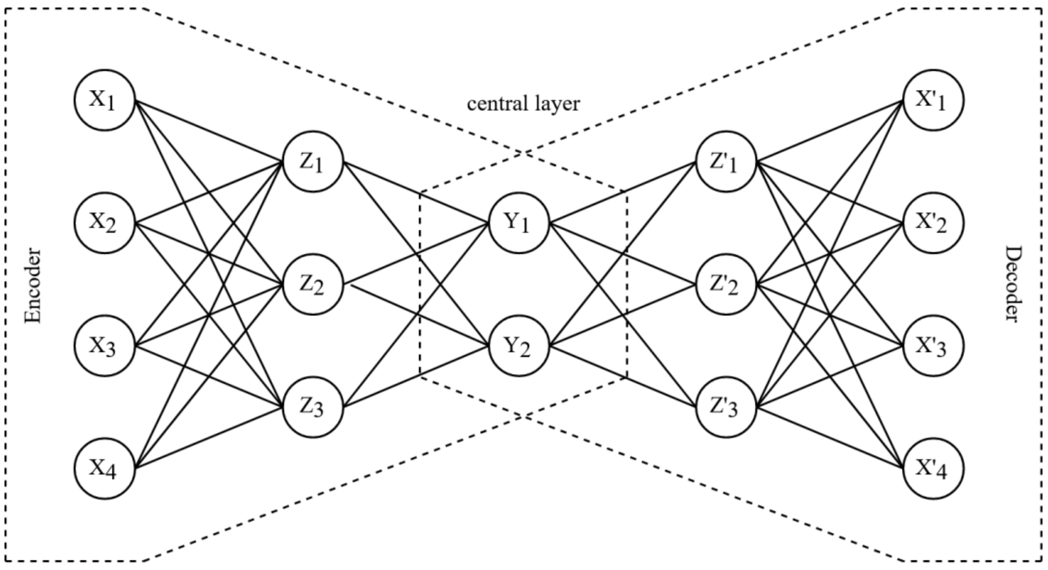

Fig. 3. **An example of the architecture of a neural network autoencoder with a narrow central layer**

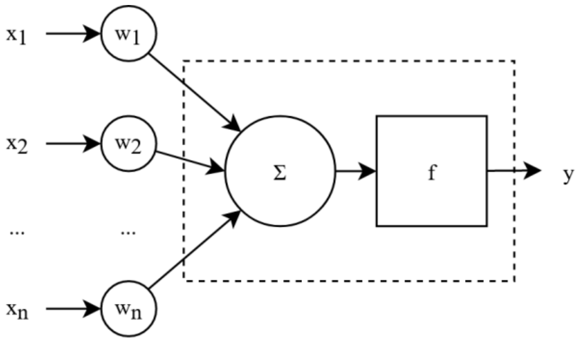

Fig. 4. **Artificial neuron model:**
x_i **– input values;** ***w_i*** **– their weight coefficients, summator** ***Σ*****, activation function** ***f*** **and output value** ***y***

At the second stage, a direct genetic search for a solution is performed. Upon initialization of the algorithm in the search area, defined as the space generated from learned features, a generation of random individuals is created. The population is projected into the phenotype space by a decoder to evaluate individuals using the fitness function. Each individual gets a fitness rating, and all subsequent genetic operations (selection, crossover, mutation) are performed in the genotype space. The process repeats until a solution with the desired fitness value appears at the decoder. The chosen approach of representing solutions allows application of any classical genetic operators.

The overall flowchart of the algorithm is shown in *Fig. 5*.

The first and main limitation of this method is the need for a representative sample of the set of legal (feasible) solutions. This condition cannot be met for some tasks. Nevertheless, there are a number of tasks for which it seems possible to form such a sample.

Another drawback of this method is the duration of the preliminary first stage of training, which in some cases may be comparable to the duration of the second stage. Here we note that this problem is most relevant for situations where the search for a solution must be carried out once for a given fitness function and a given legal (feasible) area. We assume that such tasks, firstly, are quite rare, and secondly, they are rarely demanding in terms of execution time. On the other hand, solving problems that require frequent launches of the algorithm naturally increases attention to the duration of actual genetic search. The more often search is needed, the less significant the cost of the first stage becomes.

A particular difficulty is the determination of the number of required features, i.e. the size of the central layer of the autoencoder. It is obvious that the larger the size of the feature vector, the larger the search area will be. At the same time, by choosing a too small feature vector we noticeably reduce the variety of solutions available in such a search area, up to the point when representation becomes less diverse than the training sample. In such case training of the autoencoder cannot be performed. We proceed from the assumption that the variety of solutions in the search area exceeds the variety of the training sample, since otherwise the optimal solution would be contained in the training sample, and it could be found by a simple brute force.

Composition of optimal autoencoder architecture could be a difficult task itself. The specific parameters of this network should be determined by the nature of the task. To configure the autoencoder, we should seek general recommendations on the architecture and training of neural network autoencoders in dimension reduction and feature learning tasks [24; 26–27].

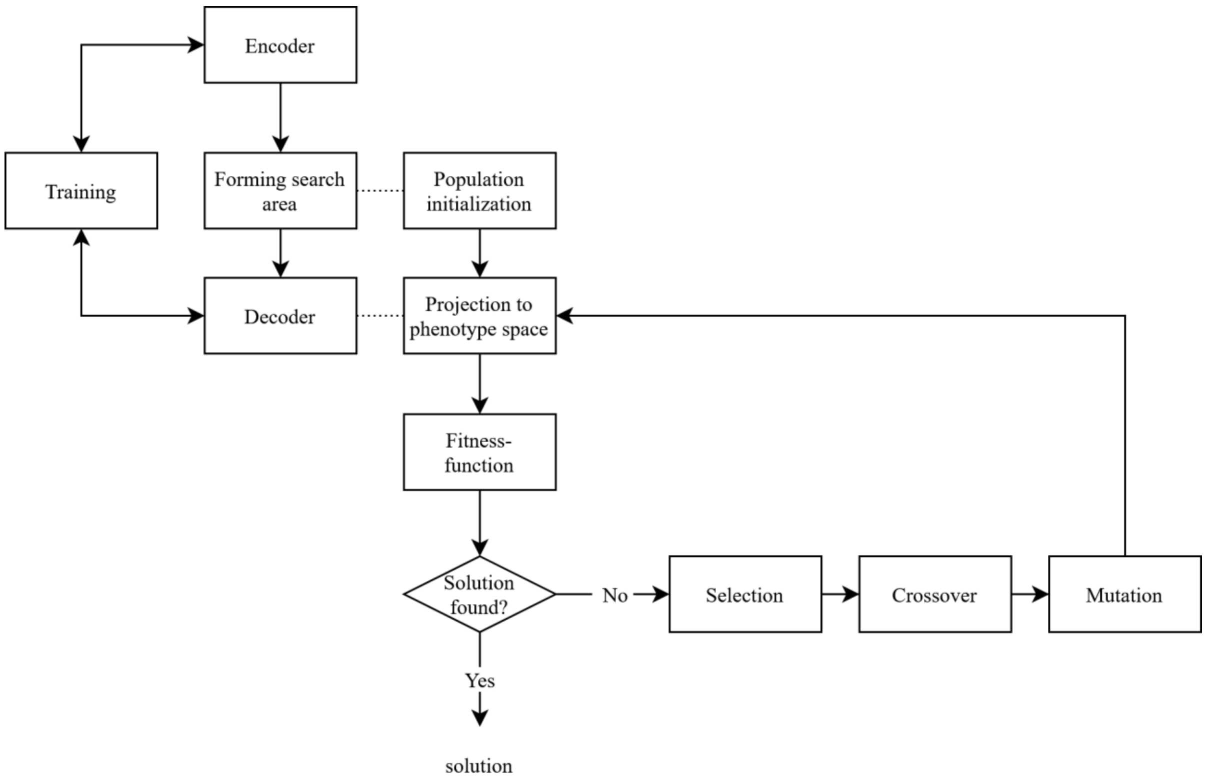

Fig. 5. **Flowchart of a genetic algorithm using an autoencoder**

Note that the proposed method does not conflict with other methods of improving the performance of genetic algorithms, such as the development of genetic strategies, operators, and some heuristic approaches, so they can be applied along with each other.

EXPERIMENT

For the experiment, the task was set to draw black and white images of handwritten decimal digits. Such a task may be faced, for example, in the system of automatic generation of captcha images. The desired solution to the problem is a black and white image of 28 × 28 pixels, which can be visually recognized as the indicated digit.

With use of a fitness function that performs visual recognition of a digit; such a problem can be solved by means of a classical genetic algorithm. Having generated random binary vectors with a length of 784 bits (which can be transformed into 28 × 28 binary matrices), by selecting the most “similar" to the desired digit ones, and subsequently recombinating their content, it is possible to recreate an image acceptable for recognition as such digit. However, it's easy to assume that with such a trivial solution coding, when the genetic representation is just a complete set of pixel values, the number of all possible combinations is extremely large. Search for the feasible solution in such a search area can be long.

At the same time, such images can be easily compressed, which leads us to the assumption that the desired solutions can be encoded in a more efficient way. The same genetic process performed on smaller vectors and then converted to a full-size 28x28 pixel image may be faster due to the fewer possible combinations. If we get a representative sample of feasible solutions, we can train the autoencoder and learn a less redundant set of features that will form the structure of our new, more efficient genetic representation.

As a sample of feasible solutions the MNIST set of handwritten numbers was used [32]. The images were pre-binarized to meet the conditions of the problem.

To conduct experiments, we will train several autoencoders on a common architecture with different size of the central layer: 128, 64 and 32 bits, respectively. The autoencoder consists of 5 inner layers. The input and output layers are composed of 784 neurons which correspond to 784 pixels of the image. The size of the remaining layers, starting from the central one, doubles, two times in each direction (*Fig. 6*). To ensure compatibility with binary coding, after learning the decoder, the center and output layers use the sigmoidal function as the activation function. Autoencoders have been trained over 50 epochs with 60,000 entries and a test sample of 10,000 entries.

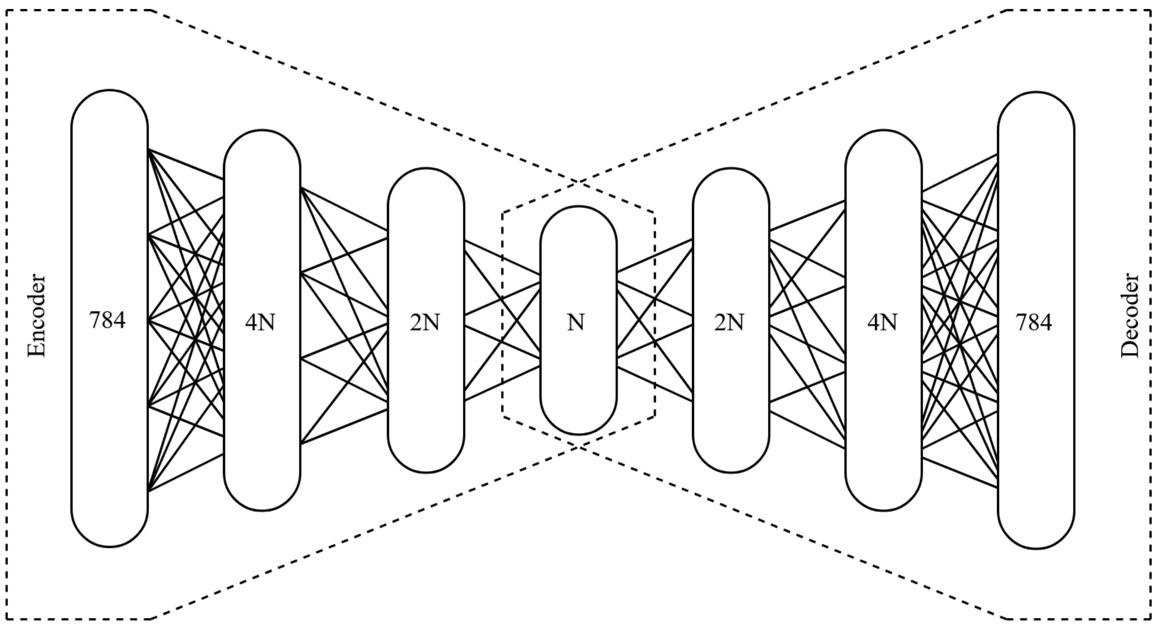

Fig. 6. **Structure of an autoencoder with a central layer of size N**

The fitness function is based on the fast normalized cross-correlation method [33], which has been successfully used to find patterns in images [34]. The fitness of an image σ is determined by the degree of similarity to a given handwritten digit. To assess this similarity, a set E of pattern images ε is selected from the MNIST dataset, which depict this digit. The fitness value of the image σ is the arithmetic mean of the values of the normalized cross-correlation coefficient calculated between σ and each pattern image ε belonging to the set E and is calculated by the formula (1), where x and y are the coordinates of the image

$$f(\sigma, E) = \frac{1}{|E|}\sum\nolimits_{\varepsilon} \frac{\sum\nolimits_{x,y}(\sigma(x,y)-\bar{\sigma})(\varepsilon(x,y)-\bar{\varepsilon})}{\sqrt{\sum\nolimits_{x,y}(\sigma(x,y)-\bar{\sigma})^2\sum\nolimits_{x,y}(\varepsilon(x,y)-\bar{\varepsilon})^2}} \quad .(1)$$

The experiment was carried out in four configurations.

In the first configuration, a trivial genetic representation that encodes an image with a vector of 784 values was used. It is structured as follows. By the condition of the problem, the image σ is represented by a binary matrix consisting of 28 rows and 28 columns, each element of which represents the brightness value of one pixel of the image, where 0 represents a black point and 1 represents a white point. The genetic representation of the matrix σ will be a vector s, each element of which is $s\ (l) = \sigma(l \oplus 28, l \odot 28)$, where $\oplus$ is an integer division operation and $\odot$ is a modulo division operation.

The other three configurations were carried out using genetic representations obtained via feature learning by the autoencoder. These representations are vectors of 128, 64, and 32 bit lengths respectively.

At the beginning of the genetic process, the population of random individuals of the given genetic representation is formed. These representations are determined by the configuration.

To bring these genetic representations to a form that is appropriate for calculating the fitness function, i.e. to a phenotypic representation, additional transformations are required. This decoding is carried out for the each generation of the population (*Fig. 7*).

In a configuration with a trivial genetic representation, for decoding, it is enough to map a vector of 784 bit length to a 28×28 matrix by sequential extraction of its rows. That is, for the vector s, construct a matrix σ, in which each of its elements $\sigma(x, y) = s\ (28 \cdot (x - 1) + y)$. Further, each element equal to 0 is considered a black point, and each element equal to 1 is considered a white point.

In the case of configurations with representations based on sets of learned features, phenotype decoding firstly requires restoring of the representation dimensionality.

To do this, each individual represented by a vector of 128, 64 or 32 bit length (the length depends on the chosen configuration) is fed to the input layer of the corresponding decoder, on the output layer of which a full-size vector of 784 bit length is formed, corresponding to a trivial genetic representation. This vector is further reduced to a two-dimensional 28×28 matrix according to the previously indicated procedure.

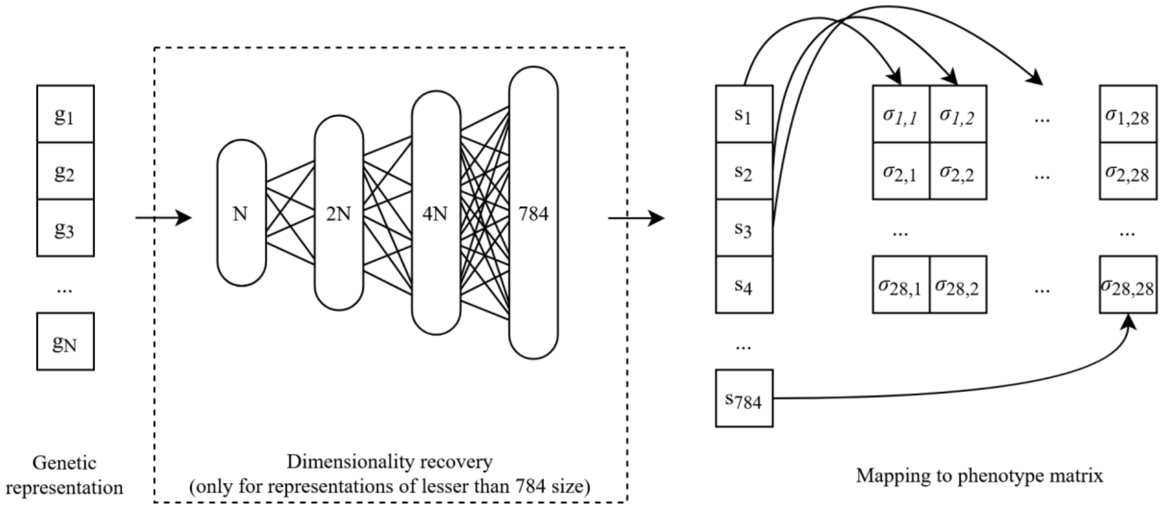
g1
g2
g3
...
gN
Genetic representation
N
2N
4N
784
Dimensionality recovery (only for representations of lesser than 784 size)
s1
s2
s3
s4
...
s784
σ1,1
σ1,2
σ1,28
σ2,1
σ2,2
σ2,28
σ28,1
σ28,2
σ28,28
Mapping to phenotype matrix

Fig. 7. **Genotype representation decoding**

Figure 8 shows the graphs of the ftiness growth of the generated images depending on the number of iterations performed. The genetic process on chromosomes with a learned feature set tends to high fitness values much faster. Note also that convergence increases with decreasing representation dimensionality.

Figure 9 shows the successive steps of generating an image of the digit 3.

As you can see, genetic method with a trivial representation requires at least 20-30 iterations, before a visually distinguishable image is achieved. At the same time, a sufficiently large number of noises are retained in the image. It will take at least 20 more iterations of algorithm to get rid of it.

At the same time, an algorithm with genetic representations based on a learned feature set generates images close to a distinguishable image of a digit from the very first iterations. Through the course of iterations only the correct shape of the pattern is selected, and the images are free of noise.

CONCLUSIONS

A new genetic method for solving optimization problems on a reduced search area was proposed. The reducing of the search area is achieved by constructing a feasible genotype representation by learning the features of feasible solutions using a neural network autoencoder. It is shown that due to the generation of a less redundant presentation, it is possible to achieve a reduction in the number of algorithm iterations required to find a solution of the desired fitness.

The use of feature learning methods is possible due to the development of a new set of requirements for genotype representations. Replacing the strict requirement of legal completeness with a weaker requirement of feasible completeness allows to reduce the set of possible candidates and focus primarily on the most promising solutions.

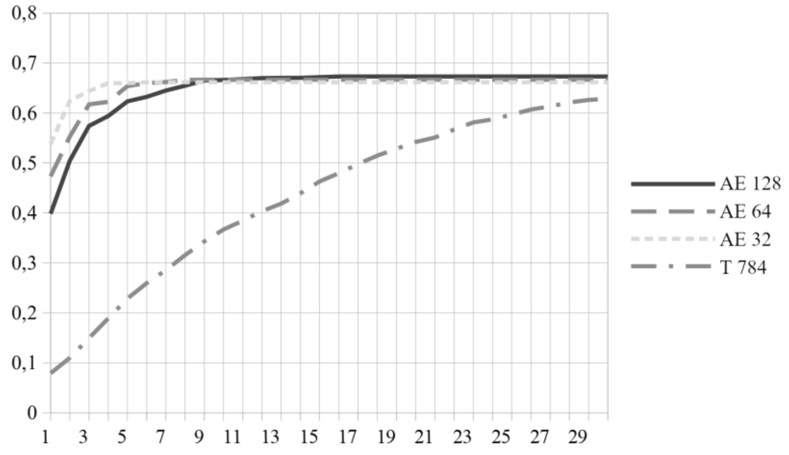

Fig. 8. **Fitness growth of the best individuals of the population during the first 30 iterations of performing a search by the genetic method with learned (AE) and trivial (T) genetic representations**

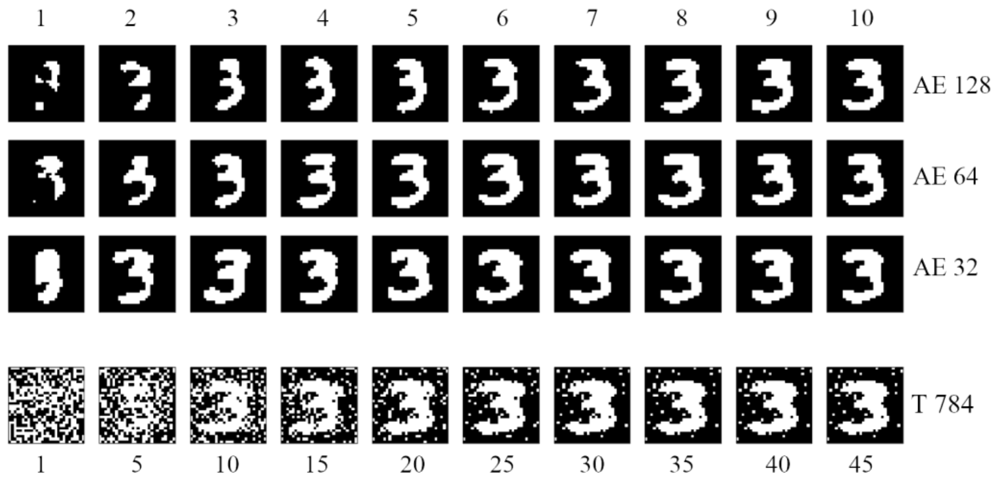

Fig. 9. **Stages of generating an image of the digit 3 using learned representations (AE) for 10 iterations (top), and trivial representation (T) for 50 iterations (bottom)**

The disadvantage of the proposed method is the necessity of a preparatory stage for feature learning, as well as the necessity of a representative sample of feasible solutions. These disadvantages limit the field of application of the method. However, technologies and systems that use genetic search for a solution actively and regularly were chosen as the target ones. With a large number of launches of the genetic method, the duration of a single preparatory stage becomes insignificant.

The positive result of the proposed method opens up an opportunity for further research. The first direction of research should be the application of other feature learning methods. In this sense, the use of generative models is of particular interest.

The second direction of research is the development of modifications of the method that ensure fulfilment of the feasible completeness requirement. The method presented in this article can still exclude some feasible solutions, including the global optimum, even if its neighbourhood is kept within the search area.

REFERENCES

1. Gen, M. & Cheng, R. “Genetic algorithms and engineering optimization”. New York. *Wiley-Interscience.* 2008. 512 p. DOI: 10.1002/ 9780470172261.

2. Baraka, H. A., Eid, S., Kamal, H. & Abdel Wahab, A. H. “Unified chromosome representation for large scale problems”. *In Multiple Approaches to Intelligent Systems*, Berlin, Heidelberg: *Springer Berlin Heidelberg.* 1999. p. 753–760. DOI: 10.1007/978-3-540-48765-4_80.

3. Jacob, B. L. “Composing with Genetic Algorithms”. *Proceedings of the International Computer Music Conference.* Banff, Canada: 1995. p. 452–455.

4. Bellgard, M. I. & Tsang, C. P. “Harmonizing music the Boltzmann way”. *Connection Science.* 1994; 6(2–3): 281–297. DOI: 10.1080/09540099408915727.

5. Komarov. O., Galchonkov. O., Nevrev. A. & Babilunga, O. “Consonant chord model of musical compositions for harmonizing melodies by a genetic algorithm”. *Odes’kyi Politechnichnyi Universytet Pratsi.* 2018; 3(56): 63–79. DOI: 10.15276/opu.3.56.2018.07.

6. Ebrahimi Moghaddam, M. & Bonyadi, M. R. “An immune-based genetic algorithm with reduced search space coding for multiprocessor task scheduling problem”. *International Journal of Parallel Programming.* 2012; 40(2): 225–257. DOI: 10.1007/s10766-011-0179-0.

7. Srinivas, M. & Patnaik, L. M. “Learning neural network weights using genetic algorithms-improving performance by search-space reduction”. Proceedings of *1991 IEEE International Joint Conference on Neural Networks.* 1991. p. 2331–2336. DOI: 10.1109/IJCNN.1991.170736.

8. Chen, S. & Smith, S. “Improving Genetic Algorithms by Search Space Reduction (with Applications to Flow Shop Scheduling)”. *In GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference.* San Francisco, United States: *Morgan Kaufmann.* 1999. p. 13–17.

9. Freitas, A. R. R. & Guimaraes, F. G. "Melody harmonization in evolutionary music using multiobjective genetic algorithms". *Proceedings of the 8th Sound and Music Computing Conference*. Sound and Music Computing network. 2011. p. 346–353. Available from: http://www.alandefreitas.com/en/papers/melody-harmonization-in-evolutionary-music-using-multiobjective-genetic-algorithms-2011-padua. [Accessed 17th September 2020].

10. Gen, M. & Cheng R. "A survey of penalty techniques in genetic algorithms". *In Proceedings of IEEE International Conference on Evolutionary Computation*. Nagoya, Japan: 1996. p. 804–809.

11. Yeniay, O. "Penalty function methods for constrained optimization with genetic algorithms". *Mathematical & Computational Applications*. 2005. 10(1): 45–56.

12. de Melo, V. V., Delbem, A. C. B., Pinto, D. L. & Junior Federson F. M. "Improving global numerical optimization using a search-space reduction algorithm". *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation – GECCO '07. Publ. ACM Press*. New York, USA: 2007. p. 77–82.

13. Chen, C.-F., Wu, M.-C., Li, Y.-H., Tai, P.-H. & Chiou C.-W. "A comparison of two chromosome representation schemes used in solving a family-based scheduling problem". *Robotics and Computer-Integrated Manufacturing*. 2013; 29(3): 21–30. DOI: 10.1016/j.rcim.2012.04.009.

14. Holland, J. H. "Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence". Cambridge, USA*: Bradford Books*. 1992. 228 p.

15. Janikow, C. Z. "A knowledge-intensive genetic algorithm for supervised learning". *Machine Learning*, 1993; No.13: 189–228. DOI: 10.1007/BF00993043.

16. Togelius, J., Yannakakis, G. N., Stanley, K. O. & Browne C. "Search-based procedural content generation: A taxonomy and survey". *IEEE Transactions on Computational Intelligence and AI in Games*. 2011; 3(3): 172–186. DOI: 10.1109/TCIAIG.2011.2148116.

17. Korkmaz, E. E., Du, J., Alhajj, R. & Barker K. "Combining advantages of new chromosome representation scheme and multi-objective genetic algorithms for better clustering". *Intelligent Data Analysis*. 2006; 10(2): 163–182. DOI: 10.3233/IDA-2006-10205.

18. Mesquita, A., Salaza, F. A. & Canazio P. P. "Chromosome representation through adjacency matrix in evolutionary circuits synthesis". *Proceedings 2002 NASA/DoD Conference on Evolvable Hardware. IEEE Comput. Soc.*, Alexandria. VA, USA: 2003. p. 102–109. DOI: 10.1109/EH.2002.1029872.

19. Yusof, U. K., Budiarto, R. & Deris S. "Constraint-chromosome genetic algorithm for flexible manufacturing system machine-loading problem". *International Journal of Innovative Computing, Information and Control*. 2012; Vol. 8 No. 3A: 1591–1609.

20. Tai, K. & Wang, N. "An enhanced chromosome encoding and morphological representation of geometry for structural topology optimization using GA". *2007 IEEE Congress on Evolutionary Computation*. Singapore. 2007. p. 4178–4185. DOI: 10.1109/CEC.2007.4425016.

21. Linden, D. S. "Using a real chromosome in a genetic algorithm for wire antenna optimization". *IEEE Antennas and Propagation Society International Symposium 1997*. Digest. IEEE. 2002. p. 1704–1707. DOI: 10.1109/APS.1997.631505.

22. Lee, J.-Y., Seok, J.-H. & Lee, J.-J. "A chromosome representation encoding intersection points for evolutionary design of fuzzy classifiers". *Intelligent Automation & Soft Computing*. 2012; 18(3): 237–246. DOI: 10.1080/10798587.2008.10643240.

23. Koziel, S. & Michalewicz, Z. "Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization". *Evolutionary Computation*. 1999; 7(1): 19–44. DOI: 10.1162/evco.1999.7.1.19.

24. Goodfellow, I., Bengio, Y. & Courville, A. "Deep Learning". *Mass. MIT Press*. Cambridge, USA: 2016. 800 p.

25. Buduma, N. & Locascio, N. "Fundamentals of deep learning". *CA: O'Reilly Media*. Sebastopol, California, USA: 2017. 298 p.

26. Bengio, Y. "Learning Deep Architectures for AI". *Foundations and Trends in Machine Learning*. Hanover, MD, Berkeley, USA: 2009; Vol. 2 No. 1: 1–127. DOI: 10.1561/2200000006.

27. Bengio, Y. "Deep Learning of Representations for Unsupervised and Transfer Learning". *Proceedings of ICML Workshop on Unsupervised and Transfer Learning, in PMLR* (27). 2012. p. 17–36. *JMLR Workshop and Conference Proceedings*.

28. Bengio, Y., Courville, A.C. & Vincent, P. "Representation Learning: A Review and New Perspectives". *IEEE Transactions on Pattern Analysis and Machine Intelligence*. New York, USA: 2013; Vol.35 No.8: 1798–1828.

29. Hinton, G. E. & Salakhutdinov, R. R. “Reducing the dimensionality of data with neural networks”. Science. New York, N.Y.: 2006; 313(5786): 504–507. DOI: 10.1126/science.1127647.

30. Le, Q., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G., Dean, J., & Ng, A. “Building High-Level Features Using Large Scale Unsupervised Learning”. *Proceedings of the 29th International Coference on International Conference on Machine Learning*. *Publ. Omnipress*. Madison, USA: 2012. p. 507–514.

31. Salakhutdinov, R. & Hinton, G. “Semantic hashing. International”. *Journal of Approximate Reasoning: Official Publication of the North American Fuzzy Information Processing Society*. 2009; 50(7): 969–978. DOI: 10.1016/j.ijar.2008.11.006.

32.“The MNIST Database of handwritten digits”. Available from: http://yann.lecun.com/exdb/mnist/. – [Accessed 17th September 2020].

33. Lewis, J. P. “Fast Template Matching”, Vision Interface 95, Canadian Image Processingand Pattern Recognition Society. Quebec City, Canada: 1995. p. 120–123.

34. Sridevi, M., Sankaranarayanan, N., Jyothish, A., Vats, A. & Lalwani, M. “Automatic traffic sign recognition system using fast normalized cross correlation and parallel processing”. *In 2017 International Conference on Intelligent Communication and Computational Techniques (ICCT).* 2017. p. 200–204. DOI: 10.1109/INTELCCT.2017.8324045.

DOI: 10.15276/hait 01.2020.1
УДК 004.021

Зменшення області пошуку генетичного алгоритму з використанням нейромережевого автокодувальника

Олександр В. Комаров
Одеський національний політехнічний університет, Одеса, Україна
ORCID: https://orcid.org/0000-0001-7651-6300

АНОТАЦІЯ

У статті розглядається проблема формування генетичного представлення для вирішення оптимізаційних задач за допомогою генетичних алгоритмів. Традиційно генетичне представлення являє собою набір з N ознак, що задають N-мірний простір гепотіпов, в якому виконується пошук рішення. Внаслідок неоптимального вибору набору ознак генотипний простір стає надмірним, область пошуку рішення збільшується, а це в свою чергу сповільнює пошук оптимуму, а також призводить до генерування кандидатів, не придатних до вимог задачі. Причиною цього є бажання охопити областю пошуку всі допустимі кандидати в рішення задачі. В оптимізаційних задачах з обмеженнями для пошуку оптимуму досить було б охопити тільки область придатних кандидатів, які потрапляють в задані задачею обмеження. Оскільки множина придатних кандидатів має меншу потужність, ніж множина всіх допустимих кандидатів, область пошуку рішення може бути вужчою. Зменшити область пошуку можна побудовою більш вигідного набору ознак, репрезентативного для множини придатних рішень. Але в разі малої кількості знань про предметну область конструювання оптимального набору ознак може виявитися нетривіальним завданням. У даній роботі пропонується використання методів навчання ознакам на основі вибірки придатних за умовами обмежень оптимізаційної задачі рішень. В якості такого методу використовується нейромережевий автокодувальник. Показано, що застосування підготовчого етапу конструювання набору ознак для побудови оптимального генетичного представлення дозволяє значно прискорити збіжність генетичного процесу до оптимуму, дозволяючи знаходити кандидатів високої пристосованості за меншу кількість ітерацій алгоритму.

Ключові слова: генетичний алгоритм; конструювання ознак; нейромережевий автокодувальник; область пошуку; оптимізаційна задача

DOI: 10.15276/hait 01.2020.1
УДК 004.021

Сужение области поиска генетического алгоритма с использованием нейросетевого автокодировщика

Александр В. Комаров
Одесский национальный политехнический университет, Одесса, Украина
ORCID: https://orcid.org/0000-0001-7651-6300

АННОТАЦИЯ

В статье рассматривается проблема формирования генетического представления для решения оптимизационных задач посредством генетических алгоритмов. Традиционно генетическое представление представляет собой набор из N признаков, который задаёт N-мерное пространство гепотипов, в котором производится поиск решения. Вследствие неоптимального выбора набора признаков, генотипное пространство становится избыточным, область поиска решения увеличивается, что в свою очередь замедляет поиск оптимума, а также способствует генерированию кандидатов, не пригодных к требованиям задачи. Причиной этого является желание охватить областью поиска все допустимые кандидаты в решения задачи. В оптимизационных задачах с ограничениями для поиска оптимума достаточно было бы охватить только область пригодных кандидатов, которые попадают в заданные задачей ограничения. Поскольку множество пригодных кандидатов обладает меньшей мощностью, чем множество всех допустимых кандидатов, область поиска решения может быть более узкой. Уменьшить область поиска можно построением более выгодного набора признаков, представительного для множества пригодных решений. Но в случае малого количества знаний о предметной области конструирование оптимального набора признаков может оказаться нетривиальной задачей. В данной работе предлагается использование методов обучения признакам на основе выборки пригодных по условиям ограничений оптимизационной задачи решений. В качестве такого метода используется нейросетевой автокодировщик. Показано, что применение подготовительного этапа конструирования набора признаков для построения оптимального генетического построения позволяет значительно ускорить сходимость генетического процесса к оптимуму, позволяя находить кандидатов высокой приспособленности за меньшее количество итераций алгоритма.

Ключевые слова: генетический алгоритм; конструирование признаков; нейросетевой автокодировщик; область поиска; оптимизационная задача

ABOUT THE AUTHOR

Oleksandr V. Komarov – PhD Student of Information Systems Department, Odessa National Polytechnic University, Odessa, Ukraine
o.w.komarow@gmail.com

Олександр В. Комаров – аспірант каф. інформаційних систем, Одеський національний політехнічний університет, Одеса, Україна

Александр В. Комаров – аспирант каф. информационных систем, Одесский национальный политехнический университет, Одесса, Украина

Received	21.08.2020
Received after revision	18.09.2020
Accepted	22.09.2020