Computer Science and Information Systems 19(2):763–781 https://doi.org/10.2298/CSIS210820004S

A Neuroevolutionary Method for Knowledge Space Construction

Milan Segedinac[1], Nemanja Milićević[2], Milan Čeliković[1] and Goran Savić[1]

[1] Faculty of Technical Sciences, Trg D. Obradovića 6,
21000 Novi Sad, Serbia
{milansegedinac, milancel, savicg}@uns.ac.rs
[2] SmartCat, Danila Kiša 3V/14,21000 Novi Sad, Serbia
nemanja.milicevic@smartcat.io

Abstract. In this paper we propose a novel method for the construction of knowledge spaces based on neuroevolution. The main advantage of the proposed approach is that it is more suitable for constructing large knowledge spaces than other traditional data-driven methods. The core idea of the method is that if knowledge states are considered as neurons in a neural network, the optimal topology of such a neural network is also the optimal knowledge space. To apply the neuroevolutionary method, a set of analogies between knowledge spaces and neural networks was established and described in this paper. This approach is evaluated in comparison with the minimized and corrected inductive item tree analysis, de facto standard algorithm for the data-driven knowledge space construction, and the comparison confirms the assumptions.

Keywords: Genetic algorithms, Knowledge Space Theory, Neural networks, Educational technology,

1. Introduction

Knowledge Space Theory (KST) gives a theoretical framework for assessing the quality of student's knowledge by representing their knowledge state instead of just quantifying it by giving a numerical grade that would represent the amount of knowledge. Identifying the precise knowledge state is of a key importance since it can direct the forthcoming learning process and suggest which units of knowledge a student should study next.

One of the most important issues in KST [1] is the construction of knowledge spaces, the mathematical models of the structure of students' knowledge [2]. There are two classes of methods that serve this purpose: theory-driven and data-driven. In theory-driven methods, the knowledge space construction is based on the experts' theoretical knowledge about the domain. On the other hand, data-driven methods construct knowledge spaces by analysing students' tests results. Such methods do not require any theoretical assumptions about the domain problems, the relationships among them, nor about the skills that the problems assume. Even though theory-driven techniques are highly useful, they are time consuming and highly labour intensive. To avoid this disadvantage, the method that we propose in this paper belongs to the family of data-driven knowledge space construction algorithms.

Real educational settings often deal with large and highly interconnected domains [2]. Such domains typically call for large knowledge spaces for representing students' knowledge states. Constructing knowledge spaces for large domains with numerous interconnected problems is a great challenge for data-driven algorithms. The number of potential knowledge spaces grows exponentially as the number of problems in the domain increases. The complexity of this problem is the main motivation for examining the possible alternative methods for knowledge space construction conducted in this research.

The rapid development in the field of Deep Learning in the past decade has called for efficient methods for solving complex optimization problems and has led to the development of new and powerful optimization algorithms. These algorithms can be applied to other fields if analogies between the models in the new fields of application and the originally intended Deep Learning models can be established. The hypothesis of this paper is that, since data-driven knowledge space construction can be observed as a combinatorial optimization problem, optimization techniques developed for the Deep Learning purposes can be directly applied to knowledge space construction as well, if analogies between the original field of application and KST are identified. In that way, we propose a novel method that it is convenient in constructing large knowledge spaces. The method uses a set of analogies between knowledge spaces and neural networks that we establish. The result of this paper is a neuroevolutionary, data-driven method for constructing knowledge spaces.

As such, this method will be useful for both educators and researchers: it will allow educators to utilize KST in teaching subjects with large and complex domains; it will also help educational researchers to study the way students learn such subjects; and the set of analogies between knowledge spaces and neural networks will contribute further development of the field of knowledge space theory by applying other Deep Learning techniques.

The paper consists of 5 sections. Section 2 gives an overview of the related work and the theoretical framework utilized in this paper, namely the Knowledge Space Theory, Neuroevolution, and the NEAT algorithm. In Section 3 we give the description of our method. The evaluation of our method for the knowledge space construction against the most commonly used knowledge space construction algorithm is given in Section 4. The paper concludes with suggestions for future improvement of the proposed method.

2. Related Work

KST [3], [4] is a subfield of mathematical psychology that was initially used mostly for the adaptive assessment of students' knowledge. KST starts from the premise that a domain can be defined as a potentially large but essentially discrete set of units of knowledge, i.e. the problems that students should master. A student that can solve all the problems from a domain is considered to have completely mastered the domain. Most often, a student can solve some, but not all of the problems from the domain. That set of the problems that a student is able to solve is termed as the knowledge state.

The set of all possible knowledge states can be very large. For a domain that consists of 30 problems, there are potentially 2^{30} = 1,073,741,824 knowledge states. But, luckily, in practice most of them are not feasible. For example, a student that cannot find the first derivative of a function will not be able to solve the problem of finding the function minima. In this example, all knowledge sates that would include the latter problem, but would not include the former one would be implausible. Following KST terminology we say that the latter problem surmises the former one.

A knowledge structure consists of a domain together with all feasible knowledge states with requirement that the empty set (representing the student who has just started learning and has not mastered any problem yet) and the domain itself (representing the student who has mastered all the problems from the domain) are feasible knowledge states.

A knowledge structure in which a union of every pair of knowledge states is also a knowledge state (that is closed under union) is termed knowledge space. In such a knowledge structure, if two students were engaged in extensive interactions while studying, it is conceivable that one of them would, at some point in time, acquire the joint knowledge of both [4].

Fig. 1 shows one knowledge space. The nodes in this graph represent the knowledge states and the edges represent surmise relation, which can be defined as follows: problem *a* surmises problem *b*, if from knowing that a student is able to solve the problem *a* we can infer that student is capable of solving problem *b*. In the figure we can see that the domain consists of a set of problems *{a, b, c, d}*. A student can master problems *a* and *d* independently, but in order to master problem *b* they need to be able to solve problem *a*.

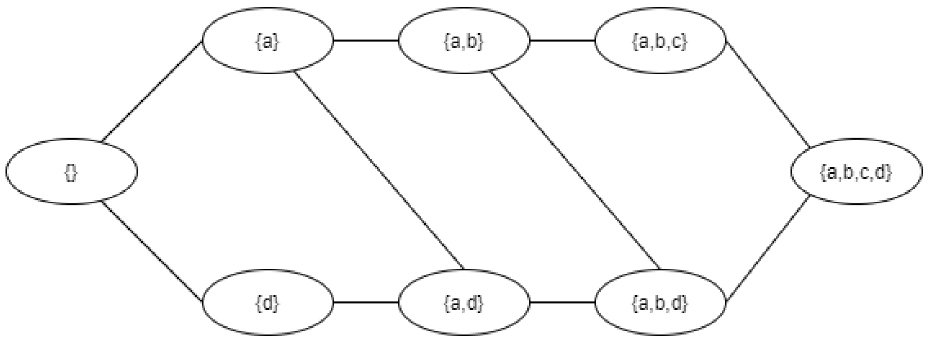

Fig. 1. Knowledge space

Surmise relation can be interpreted so that it gives a set of prerequisites for every knowledge state. Formally, it is defined as the function that associates a family of knowledge states to each knowledge state. The particularly important class of knowledge spaces are those that are closed under intersection. Such knowledge spaces can be represented by a quasi-order, without loss of information [4].

A knowledge space is considered to be a learning space if the following two additional axioms hold:

1. If knowledge state K_2 includes knowledge state K_1, a student can reach K_2 starting from K_1 by mastering one problem at the time.

2. If both K_1 and K_2 are knowledge states for which $K_1 \subset K_2$ holds and if $K_1 \cup \{q\}$ is also a knowledge state where q is an arbitrary problem from domain, then $K_2 \cup \{q\}$ is a knowledge state, as well. This axiom asserts that mastering new problems from a domain will never disable a student from mastering the ones they were able to master before.

2.1. Knowledge Space Construction

There are two categories of the knowledge space construction methods:

1. *Theory driven methods*, that either utilize the experts' knowledge about the domain or rely on the analysis of the problem-solving process, and
2. *Data-driven methods*, based on the analysis of students' response

The methods that utilize experts' knowledge about the domain use specialized algorithms for selecting appropriate combinations of questions and presented them to the experts and construct the knowledge space from the responses. Examples of this approach are the QUERY algorithm [5] and the method proposed by Cosyn and Thiery that combines QUERY algorithm with the analysis of solved tests [6].

Other theory-driven knowledge space construction methods decompose the problems in the domain into sets of motives. An example of such methods is proposed in paper [7]. In this method the information about the motives required for solving the problems is used for constructing the knowledge space.

Another similar theory-driven method observes competencies involved in problem solving process [8]. This method represents competencies by uses revised Bloom's taxonomy for representing the cognitive processes and an ontology for the domain knowledge. Other competence-based methods ([9], [10]) use domain ontology and cognitive process taxonomy to defined the dimensions of the competences and values of these dimensions are interpreted as attributes. The knowledge space constructed by using such methods reflects the assumption that, if there are two problems testing the same domain knowledge, the one that requires a lower cognitive process will precede the one that requires higher cognitive process.

All of these theory-driven methods share the same advantage, that they can be used prior to evaluating students. The most important disadvantage is that the need for continuous involvement of experts in the knowledge space construction process makes them labor intensive. Even though some of them tend to reduce the involvement of experts to some extent (e.g., the one that decomposes the problems in the sets of motives), all of them require a substantial amount of manual work. Another disadvantage of such methods, identified in [11] is that knowledge spaces that experts expect, often do not fit the test data properly so it was concluded that the real knowledge space often differs from the expected one. Theory-driven techniques are also inappropriate for constructing large knowledge spaces.

On the other hand, data-driven methods build a knowledge space starting from a set of student's answers to test questions that can be either right or wrong, i.e. response patterns. Most of these algorithms fall into one of two categories: those that use Boolean analysis to construct the surmise relation and those that construct the knowledge structure directly from the data. The most prominent examples of the first category are Item Tree Analysis (ITA) [12] and Inductive Item Tree Analysis (IITA) [13] while the examples of the second category can be found in the Schrepp's paper [14]. There are also hybrid methods that combine the data-driven algorithms with skill maps, such as D-SMEP method [15], or with consulting domain experts [6]. All the data-driven algorithms follow a three-step procedure: 1) construct the set of candidate knowledge spaces (2) test the knowledge spaces against a given criterion and (3) choose the best one according to the given criterion. Since all knowledge states are derived from the empirical data, the mentioned methods either consider all the possible knowledge states or impose certain structural constraints to data that allow them to derive the knowledge states that do not occur directly in the students' response patterns. For example, ITA and IITA construct quasi-ordinal knowledge spaces (both closed under union and intersection).

IITA derives a set of quasi-ordinal surmise relations for the given domain and chooses the one that fits the data best. That is achieved by estimating the number of counterexamples for each of them, and the one with minimal discrepancy between the observed and expected number of counterexamples is the fittest [15], [12]. This algorithm has been criticized because of its inductive approach to the construction of the knowledge space; namely is possible the addition of two implications causes an intransitivity if they are added together, but not if added separately [16]. Overcoming this problem by introducing the corrected estimator, and minimizing fit criterion so that it favors quasi-orders that favor smallest minimum discrepancies proposed in paper [16] led to the minimized and corrected IITA, de facto standard algorithm for data drive knowledge space construction. Because of that, the method that we propose is to be compared against the minimized and corrected IITA.

The method proposed in this paper is a purely data-driven one, meaning that it does not require any experts' involvement and results with a knowledge space aligned with the test results. In contrast to inductive data-driven techniques (like IITA), it does not impose any additional restriction to the knowledge space that is constructed, but it can be easily adopted to follow restrictions if needed. In addition, it can be applied to wide domains with large numbers of questions. The method establishes a set of analogies between artificial neural networks and knowledge spaces, and uses these analogies to apply well developed Deep Learning techniques for knowledge space construction. To the best of our knowledge, this approach is novel and there are no other researches that have established such analogies or utilized neuroevolution for the construction of knowledge spaces.

2.2. Neuroevolution

There are topological similarities between feedforward neural networks and knowledge spaces. This fact allows us to observe the problem of knowledge space construction as a special case of the optimization of neural network topology. There are number of ways to optimize the structure of neural networks. One of the most popular is neuroevolution, a family of evolutionary algorithms for the construction and training of neural networks.

Evolutionary Computation. Data driven knowledge space construction requires efficient combinatorial optimization techniques. Evolutionary computation [17], [18] offers a family of such techniques, inspired by the process of biological evolution, that search for suboptimal solutions for a given problem. Such techniques are most often applied in solving optimization problems with large search spaces, that make them particularly interesting candidate for knowledge space construction. In contrast to them, traditional search algorithms choose possible solutions either at random (e.g. random walk algorithm) or by using some heuristics (e.g. gradient descent) and their computational complexity is too high for the mentioned problems.

In order to construct knowledge spaces by applying an evolutionary algorithm it is necessary to choose the appropriate genetic representation, the way in which an individual in the population (i.e. a knowledge space) represents a possible solution of the given problem. That requires representing the set of encoded properties of an individual which forms a genotype. Most often, individuals are represented by the fixed-size sequences of bits (chromosomes), and that will also be the case in this research.

For each individual in the population the fitness function assesses how well it solves the problem. In the case of knowledge spaces, the fitness function tells us how well the knowledge space fits the given test results. The value given by the fitness function is used as a selection criterion when choosing the individuals that are going to be parents for next generation. From the set of parents obtained by selection, reproduction is achieved by applying the crossover operator resulting in the offspring. Just before the new generation is formed, the mutation operator is applied to the individuals that will constitute it. The mutation operator defines small random changes in the chromosome and its goal is to allow the algorithm to check a wider search space, and, in the end, to find a better fitted solution. These steps are repeated until the individual that fulfils the predefined termination condition is met. In this case, that is when the knowledge space that sufficiently fits the assessment results is found.

Neuroevolution. Since evolutionary algorithms are developed for solving complex optimization problems, they can be suitable for construction of knowledge spaces. In that way, the knowledge space construction would start from from a knowledge space and it would proceed by adding new knowledge states and extending the surmise relation throughout the evolution. As knowledge spaces can be observed as feed-forward artificial neural networks there is a possibility of applying a neuroevolutionary algorithm for their construction instead of developing an evolutionary algorithm for that purpose from scratch.

Neuroevolution is a branch of artificial intelligence that uses evolutionary algorithms to construct neural networks – both to generate their topologies and to optimize parameters [19]. The main idea is to iteratively generate, select and cross neural networks until an acceptable suboptimal solution is found, as with the general evolutionary algorithm explained in the previous section. In this process, the neural networks that have better results on the training data and those that generalize the data better are higher ranked by the fitness function.

Neuroevolution gives better results than the traditional methods based on gradient descend in environments with sparse feedbacks, such as training a neural network to win a game that requires a large number of steps to finish. Therefore, neuroevolution can be applied to a broad set of problems where a large set of precisely labelled data is not available. They impose restrictions that performance can be measured during the training process and that the behaviour of the network can be changed as it evolves.

It should be noted that recent research has shown that simple neuroevolutionary algorithms match the performance of modern sophisticated Deep Learning algorithms based on gradient descent optimization [20]. Some of the most prominent neuroevolutionary algorithms currently are GNARL [21], EPNet [22], NEAT [23], HyperNEAT [24], DXNN [25].

In neuroevolution, there are two ways to map genotype into phenotype: direct and indirect encoding. When direct encoding is applied, the genotype is directly mapped onto the phenotype, meaning that each neuron and each synapse in a neural network has its explicit representation in genotype. In our case, it would mean that every knowledge state and every surmise relation would have their explicit representation in genotype. On the other hand, in indirect encoding, the genotype indirectly specifies how the neural network should be generated. Indirect encoding is often useful for compressing large phenotypes into smaller genotypes, narrowing the search space [24], [26], [27]. In this research we will rely on the direct encoding while indirect encoding will be a topic of our future work.

Neuroevolution was first proposed as an alternative to training neural networks by backpropagation algorithm [28]. Those algorithms were used on networks with fix-topology, meaning that for them only the synaptic weights were subjected to evolutionary optimization, while the topology remained unaltered. Thus, the fix-topology neuroevolution differs from the evolution of the biological neural systems in which the structure of the neural systems itself evolves. One drawback of fixed topology neuroevolution is that in such algorithms the neural network cannot grow through the evolution, and this fact stops the network from becoming able to solve problems harder than the ones it was initially intended for. Fixed topology algorithm would be unsuitable for the construction of knowledge spaces because they will not allow for new knowledge states to be discovered. More recent algorithms solve this issue by evolving the topology of the network together with its synaptic weights.

These algorithms fall into a broad category of topology and weight evolving artificial neural network (TWEANN) algorithms. In them, the topology of the neural network can be changed by adding new neurons and synapses among the neurons with respect to certain constraints. Since the problem that we address in this research requires a topology of the knowledge space to be evolved, it will be solved by an algorithm that belongs to the TWEANN family.

3. Neuroevolutionary Knowledge Space Construction

In this section we present our neuroevolutionary approach to the knowledge space construction. This approach is based on the NEAT algorithm, and for that purpose knowledge spaces are observed as a special kind of neural networks.

3.1. NEAT Algorithm

Stanley and Miikkulainen have shown that simulated evolution of topology together with the synaptic weights give better results than fixed-topology neuroevolution [29], and proposed the NEAT algorithm as a member of TWEANN family. NEAT has solved three problems persistent with the algorithms that have preceded it: (1) giving an adequate genetic representation of the neural networks that enabled meaningful crossing-over of substantially different networks (2) preservation of the topological innovations for a few generations until their characteristics show up properly and (3) minimizing the topology during evolution without additional metrics that would measure the topology complexity. NEAT supports three mutation types: synaptic weight modification, addition of a new synapse and addition of a new neuron that shares the existing synapses. Additionally, NEAT has introduced new cross-over mechanisms and a concept of shared fitness function that increases the diversity.

NEAT uses direct genotype to phenotype encoding, meaning that each neuron and each synapse are explicitly represented. Because of that, there is no need to define new rules for neural network generation from the genotype.

One important feature of the NEAT algorithm is that it gradually evolves small networks, starting from a simple perception to more complex ones that can solve demanding problems. It should be mentioned that the NEAT algorithm has also been modified to be suitable for Deep Learning architectures–CoDeepNEAT [30].

3.2. The Neuroevolutionary Method for Constructing Knowledge Spaces

The NEAT algorithm was initially design to be used for neuroevolution, and in this paper we adapt it so that it can be used for evolutionary construction of knowledge spaces. To achieve this adaptation, we need to define the analogies between knowledge spaces and neural networks. It should be stated that solving problems in other domains by identifying the analogies with artificial neural networks is a well-known approach. In paper [31] neural networks were used for the identification of nonlinear dynamic system by identifying a set of analogies between the artificial neural networks and another nonparametric identification technique. Another example of such an approach is paper [32] where analogies between the inverse problem of choice and neural network learning were utilized. Paper [33] identifies uses the analogies between community structures and neural networks to model the evolution of social networks.

Feedforward neural networks can be considered as directed acyclic graphs where input, hidden and output neurons are vertexes and synapses are edges. Knowledge spaces can also be observed as directed acyclic graphs where knowledge states are vertexes and surmise relations define the edges. In this analogy an empty knowledge state would correspond to an input neuron, the knowledge state consisting of all the items from the domain would be an output neuron, and all the other knowledge states would be hidden neurons, while not every neuron has to be connected to all the neurons in the adjacent layer. The general idea of this analogy is shown in Fig. 2.

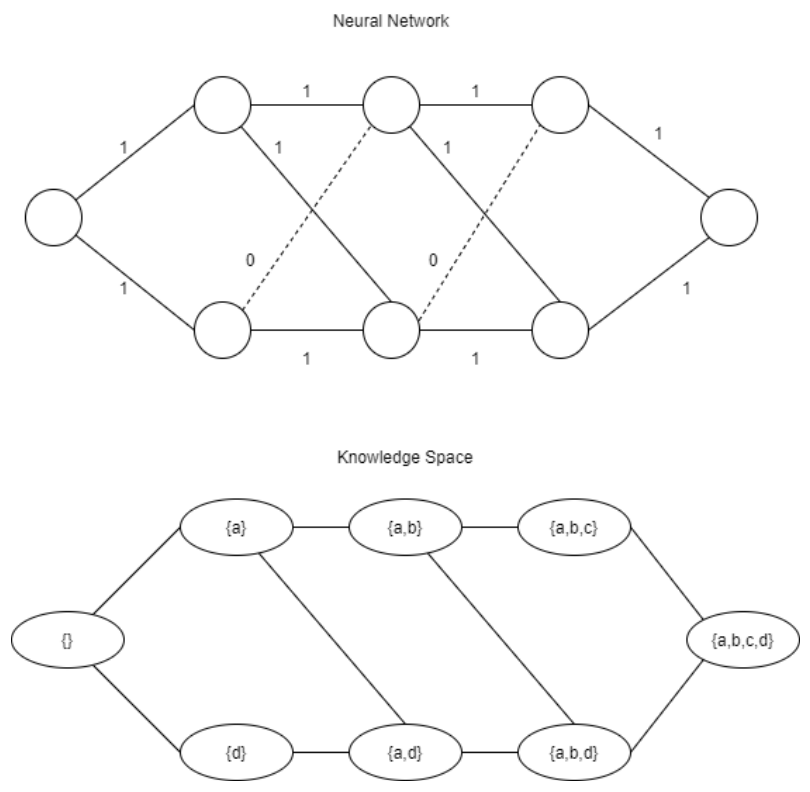

Fig. 2. A feedforward neural network and a knowledge space. The knowledge space can be considered as a neural network with 3 hidden layers and a single neuron in both the input and output layers

The complete set of analogies is given in Table 1.

Table 1. The analogies between neural networks and knowledge spaces

Neural Networks	Knowledge spaces
Neuron	Knowledge state
Neuron in the input layer	Empty knowledge state
Neuron in the output layer	Whole domain
Synapses	Surmise relation
Weights	-
Topology optimization	Knowledge space construction
Training	-

After identification of the analogies, we can observe knowledge spaces as neural networks, we can define the algorithm. It starts with an initial population of knowledge spaces. The initial population consists of knowledge spaces having just one knowledge state – an empty set. The initialization phase is followed by selection, crossing over and mutation. These steps result with a new population of knowledge spaces. This population is then divided into species and the division is made by the similarity between the individuals. The pseudocode of the algorithm is given in the listing below.

```
population = population_size of empty_knowledge_spaces
for generation_number 0 to number_of_generations
  calculate_fitness for all_knowledge_spaces in population
  if best_fitness > fitness_treshold then
    return knowledge_space_with_best_fitness
  end if
  if (best_fitness_does_not_improve in max_generations) then
    return knowledge_space_with_best_fitness
  end if
  new_population = top_k_best_knowledge_spaces
  while size(new_population) < size(population) do
    parent_1, parent_2 = select_knowledge_space_parents
    child_knowledge_space = crossing_over(parent_1, parent_2)
    child_knowledge_space = mutate(child_knowledge_space)
    new_population.append(child_knowledge_space)
  end while
  population = form_species(new_population)
end for
```

Listing. 1. The algorithm

The remaining section describes the methodology for constructing the knowledge spaces using NEAT algorithm.

Genetic representation. In evolutionary computing, the population consists of individuals represented by genes. In neuroevolution, neural networks are individuals. Since neural network consists of neurons and synapses, the genotype that represents the individual also consists of two types of genes: those that represent neurons and those that represent the synapses. Both of these two types of genes have additional attributes. So, for example, a gene that represents a neuron can also contain the information about the activation function or the type of the neuron (input, hidden or output), while a neuron that represents a synapse can contain the data about the weight, the origin and the destination neurons as well as the indicator if the synapse is active.

It has already been mentioned that, if a knowledge state is observed as a neuron and if surmise relation is observed as a set of synapses, knowledge spaces can be interpreted as a special kind of feed forward neural network. The genome of a knowledge space then has two sets of genes: genes that represent the knowledge states and the ones that represent the surmise relation. It should be noted that not all the information encoded in the genome of a general neural network is required for representing a knowledge space. The genes that represent knowledge states should not contain the information about the activation function and the neuron type.

The surmise relation in a knowledge space represents a discrete phenomenon: either a problem is a prerequisite to the other one, or it is not. Therefore, there is no need for genes that represent the surmise relation to contain the information about the weight. It is enough to represent if a connection exists and which knowledge states are connected with it.

To represent knowledge spaces, two types of genes are required: those that represent knowledge states and those that represent the surmise relation. The method that we propose uses direct genotype to phenotype encoding.

For representing the knowledge states, additional information that does not occur in the original NEAT algorithm should be given. Since a knowledge state is determined by the problems from the domain that the student can solve, the knowledge state is represented by a string of bits where each of them corresponds to one of the problems from the domain. If a student can solve the problem qi, then the i-th bit in the array will have a value of 1. In all other cases, it will have a value of 0. For example, for a domain $Q=\{a,b,c\}$ and a set of knowledge states $\{\{a\},\{b\},\{a,b\},\{a,b,c\}\}$ the bit representation would be: $\{a\}\Rightarrow 100$ $\{b\}\Rightarrow 010$ $\{a,b\}\Rightarrow 110$ $\{a,b,c\}\Rightarrow 111$.

Such a representation has the advantage of using bitwise operations that make the calculation of the fitness function efficient. For example, the fitness function described in this paper uses symmetric distance between a knowledge state and a response pattern. That can be achieved by subtraction and conjunction.

Fitness function. Choosing the right fitness function is very important for the convergence of the algorithm. The fitness function should measure how well an individual solves the initial optimization problem. In this approach, the candidate knowledge spaces are being generated through crossover and mutation, and for each of them the fitness function should evaluate how well the obtained knowledge space is aligned with the dataset. In addition to that, the fitness function should be quick to evaluate, because it is being evaluated a large number of times.

Because of these reasons, the discrepancy measure described in the paper [34], and based upon the k-modes algorithm is used as a fitness function. For the sake of the comprehensibility of the paper, the description of this discrepancy measure given in the paper [34] follows.

In this method the dataset is a collection of response patterns, each of which represents the problems from the domain that the student has solved correctly. The observed dataset is represented by the pair ($\mathscr{R}$, F), where $\mathscr{R}\subseteq 2^Q$ is a subset of the partition set of domain Q and $F{:}\mathscr{R}\to\mathbb{R}$ is a function that assigns the number of occurrences to each response pattern. For function F, it holds F(R)≥0 for every $R\in\mathscr{R}$ and $\Sigma F(R)=N$, for $R\in\mathscr{R}$ and for N students' responses.

For every knowledge structure $\mathcal{K}$ over the domain Q, partitioning N response patterns into $|\mathcal{K}|$ classes is represented by partition function $f{:}\mathscr{R}\times\mathcal{K}\to\mathbb{R}$ that fulfils two conditions:

1. f(R,K)≥0 for $R\in\mathscr{R}$ and $K\in\mathcal{K}$,
2. Σf(R,K)=$F(R)$, for $K\in\mathcal{K}$ and $R\in\mathscr{R}$.

The partition function can be interpreted in the following manner: for given $K\in\mathcal{K}$ and $R\in\mathscr{R}$ the function assigns $f(R,\mathrm{K})$ out of $f(R)$ pattern response R occurrences to the class that is being represented by the knowledge state K. The second condition guaranties that each R is going to be assigned to some class represented by the knowledge state K. For all possible partition functions for $\mathscr{R}$ and $\mathcal{K}$, the one that minimizes a certain dissimilarity measure is to be chosen. One such simple measure between $K\in\mathcal{K}$ and $R\in\mathscr{R}$ is the cardinality of their symmetric distance, $d(R, K)=|(K\backslash R)\cup(R\backslash K)|$.

Dissimilarity measure within a class given by the knowledge state $K \in \mathcal{K}$ is the weighted sum of the symmetric distances: $Df(\mathscr{R},)=\Sigma f(R,K)\cdot d(R,K)$, $R \in \mathscr{R}$.

And the total dissimilarity of a knowledge structure $\mathcal{K}$and a dataset $(\mathscr{R},F)$ is the sum of all dissimilarity measures within all the classes: $Df(\mathscr{R},\mathcal{K})=\Sigma\Sigma f(R,K)\cdot d(R,K) R \in \mathscr{R} K \in \mathcal{K}$.

The value of the fitness function for an individual in the population (that is a candidate knowledge space) is obtained from the dissimilarity Df that tells us how much the knowledge space differs from the dataset.

Selection and crossover operators. The proposed method uses the selection and crossover operators as the original the NEAT algorithm. In contrast to traditional genetic algorithms, NEAT divides the population into species. For each species, the total fitness function is calculated as the average fitness of all the individuals inside it. Then to each species a number of offspring is assigned and it is proportional to the value of the species' fitness function. The total number of offspring has to be equal to the predefined number of individuals in the population. That means that the species that have a greater fitness value will give more offspring in the next generation and that good solutions will propagate to next generations. After the numbers of offspring for the species are determined, from each species two parents are chosen to produce the offspring. After the crossover, the obtained individual is subjected to the mutation operator described in the following section.

Mutation. In each step, chosen knowledge spaces mutate and a new knowledge state can be added to them. This new knowledge state that is added to a knowledge space selected from the population differs from a knowledge state that already exists in the knowledge space for a single problem from the domain. That way, it is guaranteed that the knowledge space is well-graded meaning that it is also a learning space.

Every genome goes through the mutation phase in which there is a fixed probability that a mutation will occur. In this phase a knowledge state is chosen at random from the knowledge space that this genome represents and, if the mutation occurs, the resulting knowledge state includes one more problem from the domain. It is possible that such a knowledge state already exists in the knowledge space represented by the selected individual. In that case, the knowledge space is not changed by the mutation. In the other case, if the new knowledge state is added to the knowledge space, the surmise relation is being updated as well.

Since the mutation is random, it can happen that the newly obtained knowledge space has a smaller fitness value than the original one, before the mutation. In that case, the resulting knowledge space will have a smaller chance of survival and reproduction. All the knowledge states have the same probability of being chosen for the mutation.

Speciation. Speciation is an idea from the original NEAT algorithm that relies on the fact that new evolutions do not show their strengths right away, but need a couple of generations to emerge. Traditional genetic algorithms put new structures at a disadvantage. NEAT solves this problem by grouping individuals into species, and keeping them protected inside their species until they are fully developed.

The species are formed in accordance to the similarity of the individuals inside it. For that purpose, the function that measures the similarity between two genomes and the threshold that determines if the two individuals should be considered to belong to the same species are defined. The measure of genetic similarity is defined as:

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 S. \tag{1}$$

where *E* is the number of excesses, *D* is the number of disjoint genes and *S* is the sum of the symmetric gene distances. Coefficients c_1, c_2 and c_3 serve as weights that represent the importance of the addends. *N* is the total number of genes and it serves for normalization. Speciation is applied in this research in the same way as in original NEAT algorithm because it can turn out that the evolution of the knowledge space will not show its strengths right away, but it might turn out to show them in future.

4. Evaluation

In this section we present the evaluation of the proposed method. As explained above, today's most widely used data-driven technique for knowledge space construction is minimized and corrected IITA. Hence, the method that we propose is compared against that algorithm

The algorithms were compared with respect to their ability to reconstruct knowledge spaces from the dataset. In order to determine metrices the following was used:

1. The number of knowledge states $|\mathcal{K}e|$
2. The true positive rate (TPR), that is the percentage of knowledge states from the original knowledge space identified in the constructed knowledge space,

$$TPR = \frac{|\mathcal{K}_e \cap \mathcal{K}|}{|\mathcal{K}|}, \tag{2}$$

3. The false positive rate (FPR), that is the percentage of the knowledge states that exist in the constructed knowledge space that do not exist in the original knowledge space

$$FPR = \frac{|\mathcal{K}_e \backslash \mathcal{K}|}{|\mathcal{K}_e|}, \tag{3}$$

4. The discrepancy measure described previously in this paper

$$\boldsymbol{D}(\boldsymbol{\mathcal{R}}, \boldsymbol{\mathcal{K}}_e) \tag{6}$$

The dataset, consisting of response patterns used for the evaluation, is generated with three variable parameters: the number of problems in the domain, the number of knowledge states and the number of responses in the dataset. All the datasets are generated by using basic local independent model (BLIM), a probabilistic model of knowledge structures [35].

4.1. The dataset

BLIM defines the relation between a response pattern R and a knowledge state K with the following equation:

$$P(R) = \sum_{K \in \mathcal{K}} P(R \mid K) \cdot \pi_K, \tag{4}$$

In this equation $P(R)$ is the probability of choosing a student with a response pattern R, $P(R \mid K)$ is the conditional probability of responding with the pattern R for the given knowledge state K, and π_K is the probability of a student having a knowledge state K. Respecting the assumption that domain problems are locally independent in respect to the given knowledge states, for any response pattern R and knowledge state K, the conditional probability $P(R \mid K)$ is given in the following equation:

$$r(R,K) = \left[\prod_{q \in K \setminus R} \beta_q\right]\left[\prod_{q \in K \cap R} (1-\beta_q)\right]\left[\prod_{q \in R \setminus K} \eta_q\right]\left[\prod_{q \in \overline{R \cup K}} (1-\eta_q)\right] \tag{5}$$

Here, $\beta_q, \eta_q \in [0,1]$ are, respectively, the probabilities of a careless mistake and a lucky guess.

In order to use BLIM to simulate the response patterns for N students, we must have the knowledge structure. The first step is to take the knowledge state K with the given probability. Then, for each problem in the domain $q \in Q$, random careless errors and lucky guesses are formed with the probabilities β_q and η_q. For simulating the datasets in this paper, the following parameters have been varied:

- The number of problems in the domain $q \in Q$
- The number of knowledge states $K \in \mathcal{K}$
- The number of response patterns N

The values of the parameters for the datasets simulations used in this paper are given in the table 2.

The appropriate number of knowledge states for the given domain depends on the number of problems in the domain. The larger the domain is, the more knowledge states there are. So, in this simulation, there were 30 or 60 knowledge states for the domains with 10 problems and 100 knowledge states for the domain with 15 problems.

Table 2. the values of the parameters used for simulating the datasets

Combination number	$\|Q\|$	$\|\mathcal{K}\|$	N
1	10	30	250
2	10	60	500
3	10	30	250
4	10	60	500
5	15	100	1000

Larger domains require more response patterns in order to construct a knowledge space. For the knowledge spaces with 30 or 60 knowledge states there were 250 or 500 response patterns. For the one with 100 knowledge states, there were 1000 response patterns.

Three knowledge spaces were constructed at random for the values in the table: $\mathcal{K}1$ for combinations 1 and 2; $\mathcal{K}2$ for combinations 3 and 4; and $\mathcal{K}3$ for combination 5. For each combination, 10 simulated datasets with N response patterns were generated using BLIM. Parameters β_q and η_q are taken with uniform probability distribution from interval (0,0.05]. The probabilities π_K are taken from the interval [0.4, 0.6] also with uniform distribution, and, afterwards normalized so to equal 1. For each combination, 10 datasets were generated making 50 datasets in total.

4.2. The Comparison

This section describes the comparison of the proposed method and minimized and corrected IITA. For each of the generated datasets the learning space was constructed by using these two algorithms. For the proposed method, when there are 10 problems in the domain the population consisted of 1024 individuals, and, for 15 problems in the domain there were 2048 individuals in the population. The number of generations was 100, but early stoppage was allowed if there were no significant improvements in 20 generations. These four metrices were measured for all 10 simulated datasets for each combination. Afterwards the mean was taken to represent the performance of the algorithm for the combination

4.3. The Results

This section gives the results of the evaluation of the proposed method. In order to position it relative to the current state in the field, we have compared these results with the ones obtained by state of the art minimized and corrected IITA which is de facto standard for data-driven knowledge space construction. Table 2 shows the metrices of the neuroevolutionary method in parallel to the minimized and corrected IITA.

Table 3. The results

	Neuroevolutionary method				Minimized and corrected IITA			
	$\lvert\mathcal{K}_e\rvert$	TPR	FPR	$D(\mathcal{R},\mathcal{K}_e)$	$\lvert\mathcal{K}_e\rvert$	TPR	FPR	$D(\mathcal{R},\mathcal{K}_e)$
1	31.20 (1.78)	0.97 (0.04)	0.07 (0.02)	23.20 (31.05)	26.70 (4.12)	0.85 (0.11)	0.04 (0.03)	245.80 (208.02)
2	31.30 (1.79)	0.97 (0.03)	0.08 (0.03)	395.20 (346.45)	29.10 (3.86)	0.90 (0.08)	0.07 (0.05)	673.50 (570.02)
3	57.10 (3.33)	0.90 (0.05)	0.05 (0.04)	48.70 (44.73)	49.60 (10.06)	0.83 (0.17)	0.00 (0.00)	138.00 (185.93)
4	64.3 (2.19)	0.99 (0.01)	0.07 (0.04)	94.90 (23.65)	53.10 (3.33)	0.89 (0.06)	0.00 (0.00)	451.90 (208.75)
5	113.9 (7.19)	0.98 (0.01)	0.13 (0.05)	143.00 (60.52)	63.4 (4.54)	0.63 (0.05)	0.00 (0.00)	3678.60 (351.46)

Neuroevolutionary method results in knowledge spaces of the similar size to the original ones. This is the consequence of the appropriate fitness function which penalizes large knowledge spaces. The results show that our method resulted in slightly larger knowledge spaces than the minimized and corrected IITA algorithm.

The neuroevolutionary method gives good results of TPR since it manages to find almost all the knowledge states from the original knowledge space. It outperformed minimized and corrected IITA and it is particularly noticeable for large learning spaces. For combination 5 with 100 knowledge states the neuroevolutionary method has identified 98% of the knowledge states, while the other algorithm managed to find 63% of them.

On the other hand, minimized and corrected IITA proved to be slightly better in the case of FPR. In the mentioned case of 100 knowledge states the neuroevolutionary method had a FPR of 0.13 while this value for the other algorithm was 0. One of the reasons for this is the fact that the search space in the case of this knowledge space is much larger. Having larger populations might result in a better FPR rate, and it will be a subject of future research. We can also see that the neuroevolutionary method had a smaller discrepancy measure than IITA.

5. Conclusion

We propose in the paper a novel method for data-driven knowledge space construction. The proposed method is based upon the neuroevolutionary computing, which is one of the contributions of this paper. To the best of our knowledge, there are no similar attempts in related works. The main motivation for this approach results from the fact that neuroevolution allows solving complex optimization problems, and, therefore, the proposed method is appropriate for the construction of knowledge spaces for large and highly interconnected domains.

The method was based on the NEAT algorithm. For that purpose, a set of analogies between neural networks and knowledge spaces was proposed. The identification of these analogies itself is also a contribution of this paper, because it allows a wide range of Deep Learning techniques to be applied in the field of Knowledge Space Theory, not just to the construction of knowledge spaces.

In order to apply neuroevolution to the problem of knowledge space construction, we have proposed a genetic representation of the knowledge space, introduced the fitness function for knowledge spaces in accordance with the response patterns, and defined the speciation operator for knowledge spaces. To the best of our knowledge, these problems were not priorly solved.

The neuroevolutionary method has been compared with minimized and corrected IITA which is de facto standard data-driven knowledge space construction algorithm. From this evaluation we can conclude that the neuroevolutionary method is capable of constructing knowledge spaces from the students' response patterns. As expected, the evaluation suggests that it is more appropriate method for constructing large knowledge spaces than minimized and corrected IITA. There is still lot of space for research concerning this algorithm. First of all, optimizing fitness function and population size might result in a better FPR without compromising TPR, and this is one of the topics for future research. Secondly, we can see that both the neuroevolutionary method and minimized and corrected IITA have their strengths. Therefore, future research should combine these two algorithms to harvest the benefits of both. In addition, the proposed method can be combined with theory-driven techniques that will yield the initial knowledge spaces before assessing knowledge, and applying the neuroevolutionary method to refine them afterwards in our future work.

The method is useful for educators and education researchers. To the educators, it will allow KST to be utilized in teaching subjects with large and complex domains. To the educational researchers, it will help studying the way students learn such subjects. In addition to that, the identification of the set of analogies between knowledge spaces and neural networks will contribute further development of the field of Knowledge Space Theory by allowing the application of other Deep Learning techniques.

References

1. Doignon, J.-P., Falmagne, J.-C.: Spaces for the assessment of knowledge. International journal of man-machine studies, Vol. 23, No. 2, 175–196. (1985)
2. Ünlü, A., Sargin, A.: DAKS: an R package for data analysis methods in knowledge space theory. Journal of Statistical Software, Vol. 37, No. 1, 1-31. (2010) 3. Doignon, J.-P., Falmagne, J.-C.: Knowledge spaces, Springer Science & Business Media, (2012)
4. Falmagne, J.-C., Doignon, J.-P.: Learning spaces: Interdisciplinary applied mathematics, Springer Science & Business Media, (2010)
5. Koppen, M.: Extracting human expertise for constructing knowledge space: an algorithm. Journal of mathematical psychology, Vol. 37, No. 1, 1–20. (1993)
6. Cosyn, E., Thiéry, N.: A practical procedure to build a knowledge structure. Journal of mathematical psychology, Vol. 44, No 3, 383–407. (2000)
7. Schrepp, M., Held, T., Albert, D.: Component-based Construction of Surmise Relations for Chess Problems. In D. Albert & J. Lukas (Eds.), Knowledge Spaces: Theories, Empirical Research, and Applications (pp. 41–66). Mahwah: NJ. (1999)

8. Marte, B., Steiner, C. M., Heller, J., Albert, D.: Activity and Taxonomy-Based Knowledge Representation Framework. International Journal of Knowledge and Learning, Vol. 4, No. 1, 189–202. (2008)
9. Albert, D., Held T.: Establishing knowledge spaces by systematical problem construction. In D. Albert (Ed.), Knowledge Structures. New York: Springer Verlag, 78–112. (1994)
10 Albert, D., Held, T.: Component based knowledge spaces in problem solving and inductive reasoning, In D. Albert & J. Lukas (Eds.), Knowledge Spaces: Theories, Empirical Research, and Applications. Mahwah, NJ: Lawrence Erlbaum Associates., 15–40. (1999)
11. Segedinac, M., Horvat, S., Rodić, D., Rončević, T., Savić, G.: Using knowledge space theory to compare expected and real knowledge spaces in learning stoichiometry, Chemistry Education Research and Practice (CERP), Vol. 19, No 3, 670-680. (2018)
12. Ünlü, A., Albert, D.: The correlational agreement coefficient ca (≤, d)—a mathematical analysis of a descriptive goodness-of-fit measure. Mathematical Social Sciences, Vol. 48, No. 3, 281–314. (2004)
13. Schrepp, M.: A method for the analysis of hierarchical dependencies between items of a questionnaire. Methods of Psychological Research Online, Vol. 19, No.1, 43–79. (2003)
14. Schrepp, M.: Extracting knowledge structures from observed data. British Journal of Mathematical and Statistical Psychology, Vol. 52, No. 2, 213–224. (1999)
15. Spoto,A., Stefanutti, L., Vidotto, G.: An iterative procedure for extracting skill maps from data. Behavior research methods, Vol. 48, No. 1, 729–741, (2016)
16. Sargin, A., Ünlü, A.: Inductive item tree analysis: Corrections, improvements, and comparisons. Mathematical Social Sciences, Vol. 58, No 3, 376-392. (2009)
17. Rechenberg, I.: Evolution strategy: Optimization of technical systems by means of biological evolution. Fromman-Holzboog: Stuttgart, Vol. 104, No 1, 15–16. (1973)
18. Holland, J. H., Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press. (1992)
19. Stanley, K. O.: Neuroevolution: A different kind of deep learning. (2017) [Online]. Available: https://www. oreilly. com/ideas/neuroevolution-a-different-kind-of-deep-learning. (current December 2020)
20. Such, F. P., Madhavan,,V., Conti, E., Lehman, J., Stanley, K. O., Clune, J.: Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. arXiv (2017) [Online]. Available: https://arxiv.org/abs/1712.06567 (current December 2020)
21. Angeline, P. J., Saunders. G. M., Pollack, J. B.: An evolutionary algorithm that constructs recurrent neural networks. IEEE transactions on Neural Networks, Vol. 5, No. 1, 54–65. (1994)
22. Yao, X., Liu, Y.: A new evolutionary system for evolving artificial neural networks. IEEE transactions on Neural Networks, Vol. 8, No. 3. 694–713. (1997)
23. Stanley, K. O., Miikkulainen, R.: Efficient evolution of neural network topologies. In Proceedings to CEC'02, Honolulu, HI, USA, USA. (2002)
24. Gauci, J., Stanley, K.: Generating large-scale neural networks through discovering geometric regularities. In Proceedings to GECCO '07, London, England. (2007)
25. Sher, G. I.: Handbook of neuroevolution through Erlang. Springer Science & Business Media. (2012)
26. Gruau, F.: Neural network synthesis using cellular encoding and the genetic algorithm. LIP-IMAG. (1994)
27. Clune, J., Stanley, K. O., Pennock, R. T., Ofria, C.: On the performance of indirect encoding across the continuum of regularity. IEEE Transactions on Evolutionary Computation, Vol. 15, No. 3., 346–367. (2011)
28. Rumelhart, D. E., Hinton, G. E., Williams, R. J.: Learning internal representations by error propagation., ICS, San Diego, CA, USA. (1985)

29. Stanley, K. O., & Miikkulainen, R.: Efficient evolution of neural network topologies. In Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (2002).
30. Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N.: Evolving deep neural networks. In Kozma, R., Alippi, C., Choe, Y., Morabito, F. C. (Eds.) Artificial Intelligence in the Age of Neural Networks and Brain Computing. Elsevier, 293–312. (2019)
31. Masri, S. F., Chassiakos, A. G., Caughey, T. K.: Identification of nonlinear dynamic systems using neural networks. Journal of Applied Mechanics, Vol 60, No 1, 123-133. (1993)
32. Mikoni, S. V.: Neural network approach to the formation models of multiattribute utility. International Journal Information Models & Analyses, Vol 3, No 1, 3-9. (2014)
33. Rituraj, K, Biswal, B.: A model for evolution of overlapping community networks. Physica A: Statistical Mechanics and its Applications, Vol 474, No 1, 380-390. (2017)
34. de Chiusole, D., Stefanutti, L., Spoto, A.: A class of k-modes algorithms for extracting knowledge structures from data. Behavior research methods, Vol 49, No 4, 1212-1226. (2017)
35. de Chiusole, D., Stefanutti, L., Anselmi, P., Robusto, E.: Assessing parameter invariance in the BLIM: Bipartition models. Psychometrika, Vol. 78, No.4, 710–724. (2013)

Milan Segedinac received his M.Sc. degree in 2008 and Ph.D. in 2014 in Computer Science from the University of Novi Sad, Faculty of Technical Sciences. He holds an associate professor position at the same faculty. He has authored papers in international and national journals and conferences in the field of computer enhanced education.

Nemanja Milićević has received his M.Sc. (2019) degree from the Faculty of Technical Sciences at the University of Novi Sad. He is currently a data scientist at SmartCat. His research interests are in the field of deep learning and AI supported education.

Milan Čeliković received his M.Sc. degree from the Faculty of Technical Sciences, at University of Novi Sad in 2009. He received his Ph.D. degree in 2018, at the University of Novi Sad, Faculty of Technical Sciences. Currently, he works as an assistant professor at the Faculty of Technical Sciences at the University of Novi Sad, where he lectures several Computer Science and Informatics courses. His main research interests are focused on: Databases, Database management systems, Information Systems and Software Engineering.

Goran Savić is an associate professor within the Department of Computing and Control, Faculty of Technical Sciences, University of Novi Sad. He received his M.Sc. degree in 2006 and Ph.D. degree in 2011, all in Computer Science from the University of Novi Sad, Faculty of Technical Sciences. His research interests are e-learning and enterprise information systems.

Received: August 20, 2021; Accepted: January 25, 2022.