Applied Soft Computing 131 (2022) 109779

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Improving online education through automatic learning style identification using a multi-step architecture with ant colony system and artificial neural networks

Jason Bernard [a,b,*], Elvira Popescu [c], Sabine Graf [d]

[a] McMaster University, Hamilton, Canada
[b] Vector Institute, Toronto, Canada
[c] University of Craiova, Craiova, Romania
[d] Athabasca University, Edmonton, Canada

ARTICLE INFO

Article history:
Received 16 September 2021
Received in revised form 9 September 2022
Accepted 24 October 2022
Available online 7 November 2022

Keywords:
Hybrid intelligent systems
Learning styles
Learner model
Learning management systems
Ant colony systems
Artificial neural networks

ABSTRACT

Learning style is one of the individual differences which play an important role in learning. Being aware of it helps the student to understand their strengths and weaknesses, and the teacher to provide more valuable personalized interventions. Furthermore, learning style-based adaptive educational systems can be designed, which have been shown to increase student satisfaction or learning gain, while reducing the time needed to learn. It is therefore important to have an accurate method for identifying students' learning styles. Since the traditional approach of filling in dedicated psychological questionnaires has several disadvantages, automatic methods have been proposed, based on investigating student observable behavior in a learning environment. Research done so far generally takes a mono-algorithmic approach to identify learning styles, and the precision rates leave room for improvement. Hence, in this paper we propose a novel hybrid multi-step architecture based on ant colony system and artificial neural networks to increase the precision of learning styles identification. Two different variants are proposed and evaluated with data from 75 students; results show high precision values, outperforming existing automatic approaches for learning style identification. The proposed architecture can be integrated into widely used educational systems (e.g., learning management systems) to provide learners and/or teachers with information about students' learning styles. In addition, it can be integrated into adaptive educational systems and plugins of learning management systems to automatically identify learning styles and personalize instruction respectively.

1. Introduction

Learning styles describe students' individual preferences towards learning; according to various definitions, they refer to "different strengths and preferences in the ways they [learners] take in and process information" [1], "a description of the attitudes and behaviors which determine an individual's preferred way of learning" [2] and "the composite of characteristic cognitive, affective, and physiological factors that serve as relatively stable indicators of how a learner perceives, interacts with, and responds to the learning environment" [3].

Identifying and understanding a student's learning style may have a positive impact on learning. On one hand, this could increase students' self-awareness [1,4], allowing them to capitalize on their strengths and minimize the effect of their weaknesses through self-regulation and perceived quality of experience in blended learning environments may also be influenced by learning style [5] On the other hand, teachers may use knowledge of their students' learning styles to provide alternative activities to reach broad sections of the class whose needs are not met by the standard curriculum [6–8]. In addition, adaptive educational systems can offer a personalized learning experience based on students' individual learning style [9–15], as surveyed in several reviews [16–18]. This adaptation can bring various benefits, such as an increase in student satisfaction and engagement [11,13,19], a higher learning gain [10,13,20] or a decrease in the time needed to learn [12,21,22].

Despite the popularity and appeal of learning styles, some criticism has also been raised in the literature [23,24]. Firstly, Coffield et al. [23] argue that in some cases claims made by researchers without the support of empirical data are used to provide advice to practitioners. In order to eliminate this problem,

* Corresponding author at: McMaster University, Hamilton, Canada.
E-mail addresses: bernac12@mcmaster.ca (J. Bernard), elvira.popescu@edu.ucv.ro (E. Popescu), sabineg@athabascau.ca (S. Graf).

https://doi.org/10.1016/j.asoc.2022.109779
1568-4946/© 2022 Elsevier B.V. All rights reserved.

in the current research we take a data-driven approach to learner modeling. In addition, we focus our literature review on studies which use real student data for evaluation.

Secondly, some of the criticism is related to the burden which would be placed on teachers in face-to-face education, in case they would have to routinely change their teaching style for each learner [23]. This issue is alleviated with the use of adaptive learning systems, which are capable of automatically adapting to any number of students [16,17]. Personalization approaches are increasingly being used in learning management systems (LMS) as well, being viewed as a desirable feature by both faculty and students for many years [25].

Further criticism is addressed at the measuring instruments (i.e., questionnaires) of the learning style models, which sometimes suffer from psychometric flaws [24]. Hence, in this paper we propose an implicit learner modeling method based on students' behavioral patterns, which overcomes the weaknesses of the dedicated learning style questionnaires.

Out of the large number of learning style models which have been proposed in the literature, in the current research we selected the popular Felder-Silverman learning style model (FSLSM) [8]. This model was found well-suited for use in adaptive educational systems [26] and correspondingly employed for providing personalized courses in many research studies (as summarized by [16,17]). FSLSM consists of four dimensions: active/reflective (A/R), sensing/intuitive (S/I), visual/verbal (V/V) and sequential/global (S/G). The A/R dimension describes the preference for processing information: through learning by doing, experimentation and collaboration (in case of active students) vs. thinking and absorbing the information alone or in small groups (in case of reflective students). The S/I dimension describes the preference for gathering or perceiving information: by the use of senses and interacting with the real world (in case of sensing students) vs. the use of speculation or imagination (in case of intuitive students). The V/V dimension describes the preference for inputting or receiving information: by means of graphs, charts or videos (in case of visual students) vs. written or spoken words (in case of verbal students). The S/G dimension describes the preference for information organization and understanding: in a linear (serial) fashion, by making small steps through the learning material (in case of sequential students) vs. the requirement to see the "big picture" first, and making larger leaps from non-understanding to understanding (in case of global students) [8].

The model assumes that each learner has a preference on each of the four dimensions, which means their learning style can be described in a detailed and accurate manner. In addition, FSLSM treats each dimension as a tendency instead of an absolute type, by contrast to many other learning style models, thus providing a more nuanced view. This is done by representing the preference strength on a scale from $+11$ to -11, for each dimension. Another advantage of FSLSM is the existence of a valid and reliable measuring instrument [1], called Index of Learning Styles Questionnaire (ILS) [27].

However, using questionnaires for identifying learning styles has several drawbacks: (i) it requires an additional amount of work for the students; (ii) it may be difficult to motivate students to fill them in carefully, without skipping questions or giving random or wrong answers on purpose; (iii) it can only be applied once, so the student model cannot be subsequently updated; (iv) results can be influenced by learners' mood or perceived importance of the questionnaire [28]. To overcome these drawbacks, automatic approaches have been proposed in order to identify students' learning styles based on their behavioral patterns (as summarized in Afini Normadhi et al. [29] and Feldman et al. [30]). While some methods based on computational intelligence (CI) algorithms provide good results [31], there is still room for improvement.

Research done so far generally takes a mono-algorithmic approach to identify learning styles [31–36]. In this paper, we introduce a novel hybrid architecture to further improve the precision of identification of learning styles according to FSLSM from student behavior while interacting in a LMS. The proposed hybrid architecture combines two leading learning style identification approaches introduced in [31], using ant colony systems (LSID-ACS) and artificial neural networks (LSID-ANN). More specifically, the proposed hybrid multi-step architecture uses a complex three-step process to combine ant colony system and artificial neural networks, feeding the results from each step forward to the next algorithm. In this fashion, each step has additional information from which to refine the model of the relationship between behavior patterns and learning styles. This approach is called "Learning Style Identifier – Simplify and Solve" (LSID-SISO) since the initial steps simplify the problem by dividing students into more easily classified subgroups, as the subgroups have different characteristics on which a model can be based, and then solve the given problem in the final step. Two different variants of this approach are investigated and we show that they outperform the existing approaches to identifying learning styles.

The remainder of this paper is structured as follows. Section 2 discusses related work on automatic learning style identification. Section 3 provides an overview of hybrid architectures and the algorithms used in LSID-SISO. Section 4 describes the hybrid architecture proposed for identifying learning styles. Section 5 presents the methodology used to evaluate LSID-SISO, together with a discussion of the results. Section 6 concludes the paper and outlines future research directions.

2. Related work

The traditional method for identifying learning styles is to have students fill in a dedicated psychological questionnaire (explicit method). In order to alleviate the various problems entailed by the use of these questionnaires (e.g., static model, additional time and effort required from the students, undesirable impact of students' mood and motivation on the results), alternative implicit methods have been proposed. These are based on the analysis and interpretation of student observable behavior while interacting with an educational system. Two main approaches have been identified: data-driven (i.e., applying data mining and artificial/computational intelligence algorithms on behavior patterns) and literature-based (i.e., building rules from reviewing literature or from expert knowledge) [9]. Since the method proposed in this paper falls in the first category, in what follows we provide an overview of existing data-driven approaches for learning style identification according to FSLSM.

Several such automatic approaches have been proposed in the literature. Some of them were evaluated with artificial (simulated) datasets only; examples include: Villaverde et al. [35], who used Artificial Neural Networks, Yannibelli et al. [36], who used a Genetic Algorithm and Dorca et al. [32], who used a Reinforcement Learning method. While those approaches show great promise and demonstrate that the identification of learning styles from user behavior is possible, an evaluation with actual users is needed to show their accurateness in identifying learning styles.

When looking at identification approaches for the FSLSM closer, there are approaches that simplify the identification process by considering only a limited number of classes per FSLSM dimension (e.g., only two classes such as active and reflective) instead of identifying learning styles on the whole range of the dimension. For example, Crockett et al. [37] proposed a Conversational Intelligent Tutoring System (CITS), called OSCAR, which identifies the learning style of the students by using natural language dialogue during tutoring. A method based on fuzzy decision trees was used to build a series of fuzzy learning style predictive models, starting from behavior variables collected from CITS. The approach was applied on 75 undergraduate students who followed an SQL tutorial using OSCAR-CITS; 41 behavior variables were included in the learner dataset. A set of four fuzzy decision trees were built, one for each FSLSM dimension, considering only two classes per dimension. Another approach which considers only two classes per FSLSM dimension was proposed by Gomede et al. [34]; a deep artificial neural network method was used to identify the learning styles of students based on their behavior in a MOOC (massive open online course) environment.

A few approaches consider the whole range of the FSLSM dimensions and therefore support the strength of this model to describe learning styles in a comprehensive way. For example, Garcia et al. [33] investigated the use of a Bayesian Network (BN) to identify learning styles. As a first step, they extracted some student behaviors that are expected to be relevant for learning style identification (e.g., participation to forum and chat, preference towards concrete or abstract reading material, access to examples and exercises, answer changes, exam delivery time and results etc.). The initial probabilities were set using expert knowledge, then the BN was trained with data from 50 students and then evaluated with data from 27 students. The precision rates obtained were: 58% for A/R dimension, 77% for S/I dimension and 63% for S/G dimension; the V/V dimension was not taken into account.

Another example was proposed by Özpolat and Akar [38] who used an NBTree classification algorithm in conjunction with a Binary Relevance classifier to identify students' learning styles based on their preferred learning objects (LO). More specifically, students were classified based on the selected LOs and their associated keywords; for example, if the student selects LOs with keywords such as "practically", "real world applications", "experimental data results", then the student is more likely to be classified with a sensing learning style. The NBTree was trained using data from 10 students and then evaluated using data from 30 students. The following precision values were reported: 70.0% for A/R dimension, 73.3% for S/I dimension, 73.3% for S/G dimension and 53.3% for V/V dimension.

The currently leading approach with the most accurate results was proposed by Bernard et al. [31], who applied four computational intelligence algorithms, artificial neural network (ANN), genetic algorithm (GA), ant colony system (ACS) and particle swarm optimization (PSO), to increase the precision of automatic learning style identification. They started from an existing literature-based approach called DeLeS ("Detecting Learning Styles") [39], and aimed to improve it by two methods. The first was through classification, by using ANN with the behavior patterns from DeLeS as inputs. The second was through optimizing the weights of each behavior pattern from DeLeS (by using GA, ACS and PSO). Each algorithm was evaluated with data from 75 students and the following precision values were obtained: (i) 80.2% for A/R, 74.1% for S/I, 72.7% for V/V and 82.5% for S/G when using ANN; (ii) 78.1% for A/R, 78.1% for S/I, 75.5% for V/V and 81.8% for S/G when using GA; (iii) 80.4% for A/R, 76.2% for S/I, 77.1% for V/V and 78.5% for S/G when using ACS; (iv) 80.1% for A/R, 75.5% for S/I, 75.6% for V/V and 81.0% for S/G when using PSO.

As can be seen, the number of data-driven approaches for learning style identification according to FSLSM is quite limited. Furthermore, they all rely on mono-algorithmic methods and the reported precision rates leave room for improvement. In order to address these challenges, we introduce an innovative hybrid approach, as described in the following sections.

3. Background on algorithms

This research proposes to combine multiple artificial intelligence algorithms in a novel hybrid architecture to improve the precision of learning style identification. A literature review was conducted in order to select the candidate algorithms for hybridization, as described in the previous section. Research showed that the best accuracy results were obtained by using ANN and ACS algorithms [31]; hence, these two were selected for inclusion in the hybrid architectures that we propose. In what follows, a description of the two individual algorithms (ANN and ACS) is provided, together with a brief overview on hybrid architectures.

3.1. Artificial neural network

ANNs draw inspiration from neurobiology [40] and are based on interconnected groups of nodes called neurons. Feedforward is the most common topology for an ANN, in which the information flow is unidirectional. Neurons are organized in layers, with multi-layer perceptron being a common configuration; this consists of three layers: input, hidden and output. Each neuron in each layer has a weighted connection from each neuron in the preceding layer, while each input neuron has a single connection to an input variable. A sigmoid function (e.g., *tanh*) is used to calculate the strength (output) of the neuron, based on the sum of each input multiplied by the corresponding weight of the neural link. A feedforward 3-layer perceptron is used in this research.

Both supervised and unsupervised learning strategies can be employed with ANNs. In our case, a supervised learning strategy is used, as the actual learning style of the students in the training set is known. The training technique applied is back propagation, i.e., the ANN is traversed in reverse along each neural link, adjusting the weights and thresholds by calculating a weight modification (ΔW) as a sigmoid function of the error between the output and actual values. A learning rate ($\eta < 0$) is included to keep the ANN from oscillating over the optimum and a momentum ($m \leq 0$) can be added to help escape local optima. Training can be done either in individual or ensemble mode; in the first case, the weight modifications are applied after each sample is run, while in the latter case these modifications are summed and applied at the end of the generation. The back propagation algorithm runs until a particular termination condition is reached.

3.2. Ant colony system

ACS is an optimization algorithm inspired by the behavior of real ants while foraging for food [41]. When an ant discovers a food source, it lays a trail of pheromone to it, which other ants can detect and follow; an indirect communication between ants thus takes place.

To use ACS, the solution space must be described as a graph; each node represents a part of a solution to the problem and the whole path through the graph represents a solution. A colony (population) of P artificial ants is created, with each ant placed at a starting node (which could be a fixed node or a randomly selected one, depending on the problem). The ants then traverse the graph, by iteratively choosing a link to follow through a pseudorandom proportional rule. Thus, a random value from 0 to 1 is selected and if this value is less than the exploitation parameter ($0 \leq q_0 \leq 1$) then the link with the highest quality (Q) is selected; Q is a function of the local quality (l) and amount of pheromones (τ) on the link, weighted by two parameters ($\alpha \geq 0$ and $\beta \geq 0$), as shown in Formula (1). Otherwise, roulette wheel technique is used to select the link, where the selection probability is equal to the quality of the link divided by the total quality of all links from that node, as shown in Formula (2).

A candidate list of nodes may be used (i.e., a pre-generated static list of preferred choices from each node); a tabu list can also be employed, so that an ant does not return to a previously visited node. When an ant has no more nodes to move to, this means the solution is either complete or invalid, depending on the problem.

The algorithm also aims to encourage exploration, by decreasing the amount of pheromone when an ant traverses a link. A portion ($0 \leq \tau_0 \leq 1$) of the pheromone is consumed, as shown in Formula (3); hence, subsequent ants are less likely to take the same path. In addition, once all ants have completed traversing the graph, another portion of pheromone ($0 \leq \rho \leq 1$) from each link is evaporated, according to Formula (4), further encouraging exploration. Conversely, the global best path or the iteration best path has pheromones laid on each link of the path as a function of the fitness of the solution. The algorithm runs with a new generation of ants, until a termination condition is fulfilled.

$$Q = l^{\alpha} \times \tau^{\beta} \tag{1}$$

$$S_n = \frac{Q(l_n, \tau_n)}{\sum_{x=1}^{n} Q(l_x, \tau_x)} \tag{2}$$

$$\tau = (1 - \tau_0) \times \tau' \tag{3}$$

$$\tau = (1 - \rho) \times \tau' \tag{4}$$

3.3. Hybrid architectures

A hybrid artificial intelligence algorithm is a technique for combining multiple individual algorithms together, so as to capitalize on the strengths of each [42,43]. Typically, this can be done in one of three ways. The first technique consists in two algorithms jointly processing data, where one is intended to provide a globally oriented search and the other a locally oriented search [44,45]. A second approach implies that one algorithm provides a configuration for the second algorithm, as in case of evolving artificial neural networks [46]. The third technique consists of two or more algorithms where information from one algorithm is sent to the other(s) in the ensemble to improve their processing [43,47].

Hybrid architectures can be either tightly or loosely coupled. A tightly coupled hybrid architecture allows information to flow both forward and backward (i.e., cyclically) between the individual algorithms; an external controller dictates the order in which the algorithms are trained and executed. By contrast, a loosely coupled hybrid architecture allows information to be passed forward only; no external controller is required, as the algorithms are trained and executed sequentially.

For the current research, a loosely coupled hybrid architecture is used, which is described in more detail in the following section.

4. Simplify and solve

This section introduces two novel LSID-SISO algorithms, each a hybrid architecture that arranges mono-CI algorithms to identify students' learning styles from their behavior patterns in an educational system. The development of these algorithms is based on two existing works from literature [31,39]. As such, the first three subsections provide a brief overview of these two works, starting with identifying learning styles from behavior patterns [39], and then enhancing the resulting model using an ACS and ANN for increased accuracy [31]. Section 4.4 describes how the new hybrid architecture combines these techniques in a novel way to provide even further improvements to accuracy.

4.1. Identifying learning styles from behavior patterns

To identify learning styles from behaviors, the first step was to produce a set of behavior patterns as the features for the model. Graf et al. [39] identified behavior patterns from the educational psychology literature that were known (or at least suspected) to be related to each of the four FSLSM dimensions and used those patterns to create the learning style identification system DeLeS. The selected behavior patterns are generic and trackable by most learning management systems in order to ensure DeLeS is practical. For this research, the same behavior patterns (shown in Table 1) are selected since DeLeS produced results with high accuracy [39]. Also, LSID-SISO inherits the generic and practical nature of DeLeS.

Broadly, the behavior patterns can be classified as: (1) navigational, (2) grade-based, or (3) unique. Navigational behavior patterns track either the average amount of time a student stays on a particular type of learning object or the percentage of learning objects they visit of each type. For example, "example_stay" indicates how long on average a student stays on learning objects classified as examples, and "example_visit" expresses the percentage of examples the student visited out of the total number of examples. As can be seen in Table 1, there are navigation behavior patterns for content, examples, exercises, forums, the outline, self assessments, and quiz results (average time only). Additionally, the percentage of learning objects (of any type) skipped is tracked.

The second classification of behaviors are the student's grades on particular questions of quizzes. Patterns consider questions focused on concepts, details, developing new solutions and interpretation of existing solutions. In addition, questions may be about overviews, facts, knowledge presented in text or knowledge presented as graphics.

There are also three unique behaviors: (1) how often the student posted to forums per week, (2) how often the student revised quiz answers before submitting, and (3) how often the student answers the same question twice wrong.

To calculate learning styles from those behavior patterns, DeLeS uses two thresholds per behavior pattern to classify a student's behavior based on the given pattern as either strong indicative for one learning style on the given dimension, strong indicative for the other learning style on the given dimension or average (and therefore does not provide a particular hint). In addition, no information can be available for a given pattern. To indicate the relationship between a behavior pattern and a learning style the symbols "+" and "-" are used in Table 1. For each behavior pattern a "+" symbol in Table 1 shows that a high occurrence of the given behavior indicates a preference for the active, sensing, visual, and sequential learning styles. Inversely, a "-" symbol shows that a high occurrence of the given behavior indicates a preference for the reflective, intuitive, verbal and global learning styles. Computationally, a preference is encoded as a hint value h of 3 (strong preference for active, sensing, visual or sequential learning style), 2 (neutral preference), 1 (strong preference for reflective, intuitive, verbal or global learning style), and the learning style dimension LS_{dim} is computed as the average of the hint values (as per formula (5)).

$$LS_{dim} = \frac{\sum_{i=1}^{p_{dim}} h_{dim,i}}{p_{dim}}, \tag{5}$$

where dim represents one of the four learning style dimensions and p represents the number of patterns for which information is available.

Table 1
Relevant behavior patterns for each FSLSM dimension [39].

Active/ Reflective	Sensing/ Intuitive	Visual/ Verbal	Sequential/ Global
content_stay (−)	content_stay (−)	content_visit (−)	outline_stay (−)
content_visit (−)	content_visit (−)	forum_post (−)	outline_visit (−)
example_stay (−)	example_stay (+)	forum_stay (−)	question_detail (+)
exercise_stay (+)	example_visit (+)	forum_visit (−)	question_develop (−)
exercise_visit (+)	exercise_visit (+)	question_graphics (+)	question_interpret (−)
forum_post (+)	question_concepts (−)	question_text (−)	question_overview (−)
forum_visit (−)	question_details (+)		navigation_overview_stay (−)
outline_stay (−)	question_develop (−)		navigation_overview_visit (−)
quiz_stay_results (−)	question_facts (+)		navigation_skip (−)
self_assess_stay (−)	quiz_revisions (+)		
self_assess_twice_wrong (+)	quiz_results_stay (+)		
self_assess_visit (+)	self_assess_stay (+)		
	self_assess_visit (+)		

4.2. Identifying learning styles with ACS

The identification approach in DeLeS [39] is based on the assumption that all behavior patterns relevant to a learning style dimension are equally important. While this produces a model with reasonable accuracy (as shown in Section 5.5), it can be expected that some behaviors are more or less important than others. However, it is not clear from the educational psychology literature what the relative importance might be for any of the behavior patterns. As such, finding those weights was seen as an optimization problem. It was reasoned that accuracy could be improved by weighting the hint values from each pattern and calculating the average of the weighted hint values.

To extend the identification approach of DeLeS, a set of N weights must be found for each learning style dimension, where N is the number of relevant behavior patterns for that dimension. To use ACS to find a set of N weights, a graph was constructed for the ants to traverse. This graph was formed such that an ant's path can be decoded into a set of N weights. The design of the graph for LSID-ACS consisted of an N-layered graph with each layer having 100 nodes, with the number of layers varying for each of the FSLSM dimensions, and accordingly a separate LSID-ACS was built for each dimension. The values of the nodes ranged from 0.01 to 1.00 in increments of 0.01, representing the weights. For the purposes of simplifying processing, a "Start" node was added before the first layer and an end node was added after the Nth layer. In each generation, all ants begin in the "Start" node and indicate they are finished when they reach the "End" node. As an example, Fig. 1 shows the graph used for the V/V dimension. Since this is an acyclic graph, an ant's path will traverse a single node per layer, hence selecting N nodes. The N selected nodes, in turn, decode into a weight for each behavior pattern. In essence then, from a mathematical point of view a hyperspace is defined consisting of N dimensions with bounds from 0 to 100.

The goal and as such the aim of the fitness function is to maximize the average accuracy (ACC) of the learning style identification process. Accuracy is computed using the predicted learning style values calculated by the algorithm (LS_{pred}) and the actual learning styles values (LS_{actual}) for a student (i) as follows: $ACC_S = 1-|LS_{pred} - LS_{actual}|$. Since no student is preferred, the overall ACC is the average across all students. Hence, the fitness function is computed by Formula (6) for a set of N students (i).

$$F = \frac{\sum_{i=1}^{N} ACC_i}{N} \rightarrow max \tag{6}$$

One of the challenging aspects of using artificial intelligence algorithms is to determine a termination condition. While an obvious condition for stopping the algorithm is when all students are identified perfectly ($ACC = 1.0$ or $error = 0$), some other condition must be used to keep the algorithm from continuing forever if such solution does not exist or cannot be found. As such, for a stopping condition, the generation in which the best solution (gen_{best}) is found and recorded. If an additional gen_{best} generations have passed without finding a better solution, then the algorithm stops. In other words, all best solutions are found in the first half of the number of generations processed. For example, if $gen_{best} = 1000$, then the algorithm will terminate after the 2000th generation; however, if a new best solution is found at the 1500th generation ($gen_{best} = 1500$), then it will terminate after the 3000th generation. In addition, to prevent early termination due to chance, a minimum number of 100 generations need to be processed. As such, the termination condition for ACS is described in Formula (7).

$$T = (error_{best} = 0) \vee ((gen_{nochange} > gen_{best}) \wedge (gen_{current} > min)), \tag{7}$$

where $error_{best}$ is the lowest error found by the algorithm so far, $gen_{nochange}$ is the number of generations since the best solution (i.e., lowest error) was found, gen_{best} is the generation in which the best solution (i.e., lowest error) was found, $gen_{current}$ is the current generation of the algorithm, min is set to 100 to represent the minimum number of generations the algorithm should run.

Algorithm 4.1 shows the pseudocode for identifying learning styles using only the ACS. To keep the algorithm easy to read, the inner workings of the ACS are not shown in the pseudocode since for this algorithm they were unmodified. Instead, the process of progressing forward on generation is represented by an `Iterate()` function.

One of the features of ACS is to consider pheromone values as well as local information of a link between two nodes to decide which node to select next. Local information represents the measurable quality of a link (e.g., distance between two nodes, costs of selecting a link, etc.). For this problem, no local information is available since a single weight on its own does not have a specific measurable quality that would impact the overall solution in a certain way. Therefore, no local information is used in the ACS and only the pheromone information is used for link selection. Dorigo and Stützle [48] consider this a valid use of the ACS. For the sake of simplicity, the local values on all links are set to 1, and the two control parameters α and β are both set to 1. In addition, no candidate lists are used. Candidate lists are optional in ACS and intended only as a mechanism for incorporating local information, which does not exist for this problem.

4.3. Identifying learning styles with ANN

ANNs have been described as a "universal approximator" [49], meaning that they can be used to find complex functions to describe the relationship between a set of inputs and outputs.

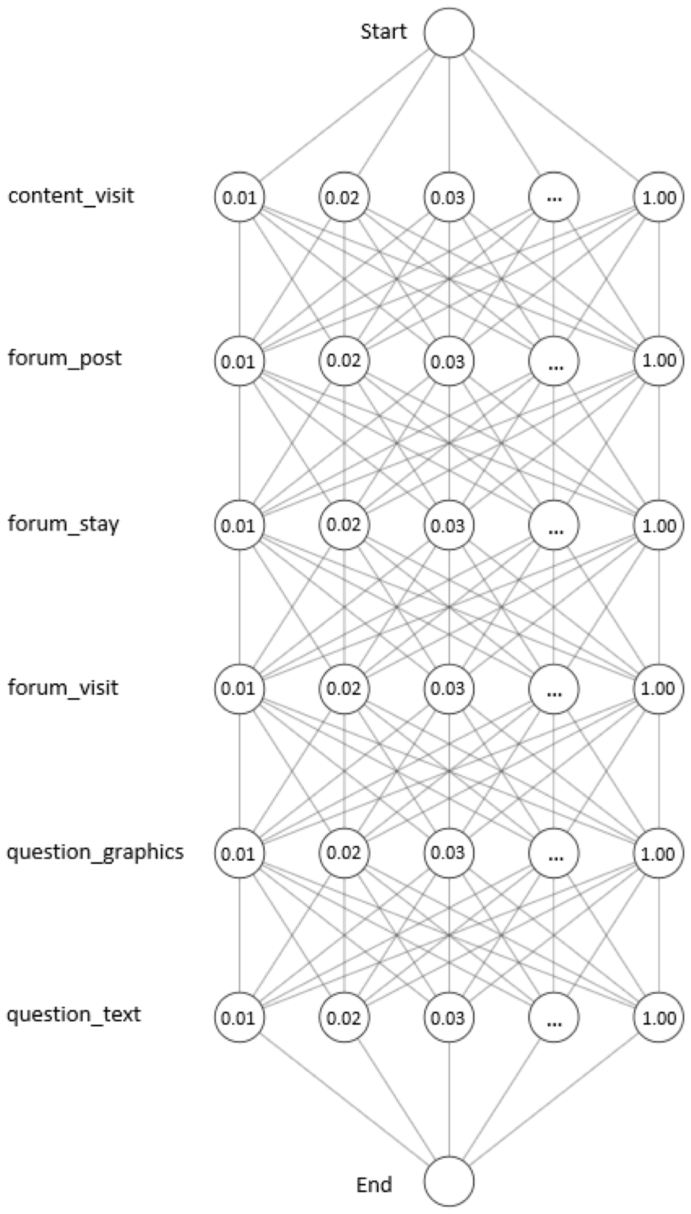

Fig. 1. Graph for finding set of weights for V/V dimension [31].

LSID-ANN [31] uses a multilayer perceptron (MLP) topology to find the function that best describes the relationship between a learning style dimension and relevant behavior patterns for that dimension, based on the identification approach used in DeLeS [39]. While the identification approach in DeLeS assumes that each hint from each pattern is equally important, LSID-ANN extends the approach in DeLeS by finding a function that considers the individual importance of each hint from each pattern.

A separate LSID-ANN was built for each FSLSM dimension using the relevant behavior patterns as the inputs and identified learning style value as output. Fig. 2 shows the MLP topology used to identify the V/V dimension as an example.

The fitness function for the LSID-ANN algorithm is the same as for LSID-ACS (see Formula (6)), where the aim is to maximize the accuracy of the identification process (and thereby minimizing the error).

For the ANN, the termination condition is controlled by computing the predicted future error. To do this, a set of samples (i.e., students) is selected to act as a hypothetical future set to be classified by the ANN called the validation set. The error in the validation set can then be thought of as the error that the algorithm would have in the future. If the future error begins to rise, and the error in the training set continues to fall, then the algorithm has likely reached a local optimum and is now overfitting to the training set, and training can be ceased. The algorithm would also terminate if both errors were 0. Let $error_g$ be the error in the current generation of the training set $error_{g-1}$ be the error in the previous generation of the training set, $error_f$ be the predicted future error in the current generation calculated from the validation set, and $error_{f-1}$ be the predicted future error in the previous generation calculated from the validation set. Then the termination condition (T) can be expressed as shown

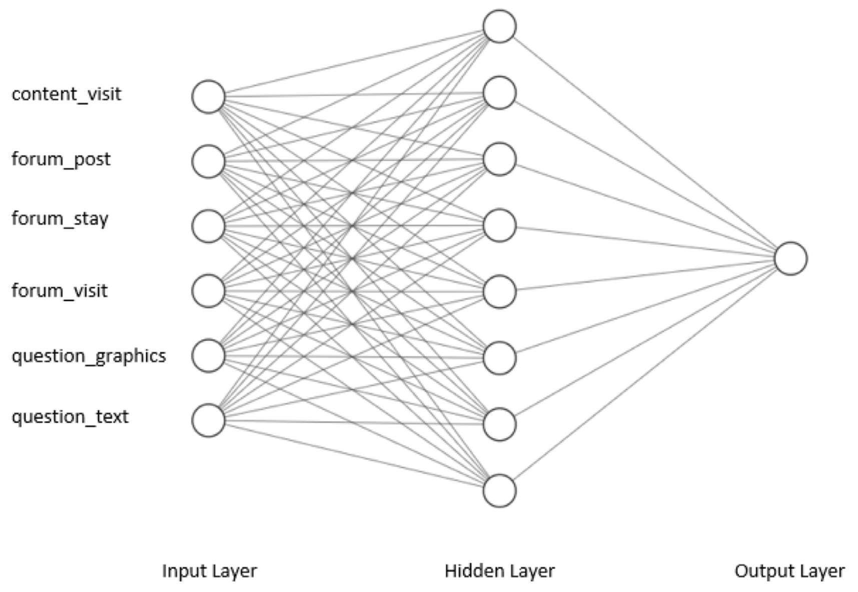

Fig. 2. LSID-ANN topography for the V/V dimension [31].

```
Data: A set of $N$ students each with a list of behavior pattern
      data $D$ and actual learning style value $LS$ (see
      Section 4.1). A minimum number of generations to
      complete $min$.
Result: A list of weights $W_{best}$.
$error_{best} \longleftarrow \infty$ (the best error to date);
$W \longleftarrow \{\}$ (a set of weights);
$W_{best} \longleftarrow \{\}$ (the best list of weights so far);
$gen_{current} \longleftarrow 0$ (the current generation);
$gen_{best} \longleftarrow 0$ (the generation in which the best result was
  found);
$gen_{nochange} \longleftarrow 0$ (the number of generations since the best
  result was found);
$LS_{pred} \longleftarrow 0$ (the predicted learning style value);

// Let $ACS$ be a standard ant colony system algorithm of $A$
  ants;
// Iterate() processes a generation of the ant colony system
  and returns the best weights found in that generation;
repeat
    $error \longleftarrow 0$;
    $W \longleftarrow ACS.Iterate()$;
    $i \longleftarrow 0$ (an index);
    repeat
        $j \longleftarrow 0$ (an index);
        $LS_{pred} \longleftarrow 0$;
        repeat
            $LS_{pred} \longleftarrow LS_{pred} + (D[i][j] \cdot W[j])$ (sum the behavior
              data multiplied by the weight);
        until $j > |W|$;
        $LS_{pred} \longleftarrow LS_{pred}/|W|$ (average the weighted behavior
          data);
        $error \longleftarrow |LS_{pred} - LS[i]|$;
    until $i > N$;
    if $error < error_{best}$ then
        $W_{Best} \longleftarrow W$
    end
until $error_{best} = 0 \vee (gen_{nochange} > gen_{best} \wedge gen_{current} > min)$;
```

Algorithm 4.1: Algorithm to train an ant colony system to find a set of weights for behavior patterns to identify learning styles.

in Formula (8):

$$T = (error_g = 0) \wedge (error_f = 0) \vee ((error_g < error_{g-1}) \wedge (gen_f < error_{f-1})). \tag{8}$$

The pseudocode for training an ANN is similar to that of an ACS. The code consists of a loop through the training samples where each one is sent through the ANN (shown as a `Process()` function) returning a predicted learning style (LS_{pred}). The error is computed by comparing the predicted learning style to the actual learning style (LS_{actual}), and recorded in a list. The list of errors is then used for the backpropagation process (shown as a `Backpropagation()` function). Of course, mathematically, the space defined by the neural network is vastly more complex than that produced by the ACS. In the ACS, the relationship between behaviors and learning styles is a simple weighted sum of the behavior patterns. With an ANN, the behaviors can interact through the edges connecting the input layer and hidden layer, and hidden layer to output layer, although prior research (Bernard et al. 2015 [31]) found the extra complexity did not decisively produce better results. After the backpropagation step, the fitness is computed using the fitness function previously described. However, fitness values, or more accurately, the error values calculated from the fitness values, are only used for the termination condition for ANNs, as backpropagation provides the guidance towards an optimal configuration (ideally global, but not guaranteed). The algorithm for training the ANN is shown in Algorithm 4.2.

4.4. Identifying learnings styles with hybrid architecture

Within the context of the previous two subsections, a model of the relationship between behavior patterns and learning styles is created by the process of optimizing weights or training an ANN. In the former case, the model is created using a priori information gleaned from the literature about behavior patterns and learning styles and then optimized by weighting the different relationships. In the latter, the trained ANN itself is the model, i.e. it takes behavior patterns as inputs and outputs a learning style. It can be said then that some weighted formula or a trained ANN is a model M. Furthermore, since both of these algorithms have been executed then it can be said that there exist multiple models of the relationship between behavior patterns and

```
Data: A set (S) of N students each with a list of behavior
      pattern data D and actual learning style value $LS_{actual}$
      (see Section 4.1). A set (V)) of M students used as a
      validation set. A minimum number of generations to
      complete min.
Result: A trained ANN.
$error_g \longleftarrow \infty$ (error in the current generation);
$error_{g-1} \longleftarrow \infty$ (error in the previous generation);
$error_f \longleftarrow 0$ (predicted future error);
$error_{f-1} \longleftarrow 0$ (predicted future error for the previous
 generation);
$LS_{pred} \longleftarrow 0$ (predicted learning style);

// Let ANN be a standard artificial neural network with |D|
 inputs;
// Process(s) performs a standard forward propagation, given
 a student s as input;
// Backpropagation(e) performs a standard backpropagation
 process on ANN taking a list of errors e as input;
repeat
   $error_{g-1} \longleftarrow error_g$;
   $error_{f-1} \longleftarrow error_f$;
   $errorg \longleftarrow 0$;
   $errorf \longleftarrow 0$;
   $n \longleftarrow 0$ (an index);
   // compute the error and perform the backpropagation;
   foreach s in S do
      $LS_{pred} \longleftarrow ANN.Process(s)$;
      $e[n] \longleftarrow |LS_{pred} - LS_{actual}[n]|$;
      $error_g \longleftarrow error_g + e[n]$;
      $n \longleftarrow n + 1$;
   end
   $ANN.Backpropagation(e)$;
   // compute the future error;
   $n \longleftarrow 0$;
   foreach s in V do
      $LS_{pred} \longleftarrow ANN.Process(s)$;
      $e[n] \longleftarrow |LS_{pred} - LS_{actual}[n]|$;
      $error_f \longleftarrow error_f + e[n]$;
      $n \longleftarrow n + 1$;
   end
until $(error_g = 0$ and $error_f = 0)$ or
 $((error_g < error_{g-1})$and$(error_f > error_{f-1}))$;
```

Algorithm 4.2: Algorithm to train an artificial neural network to find an optimal configuration identify learning styles from a set of behavior pattern inputs.

learning styles. For example, let the model produced by the ACS be, for Active/Reflection as an example, $M_{ACS,AR}$ and the model produced by the ANN be $M_{ANN,AR}$. While, $M_{ACS,AR}$ and $M_{ANN,AR}$ have a different accuracy, they are both valid models for the relationship.

The core concept in designing the hybrid algorithm is the observation that, in some optimization problems, given two unique non-optimal solutions (sol_1 and sol_2) with the same fitness value, these two solutions have some components that are optimal or near optimal while other components are of poorer quality. By combining the optimal or near-optimal components, a better solution can be found. With respect to learning styles identification, two unique solutions with equivalent fitness (assuming fitness is the average accuracy from all learning styles of all students) can occur when different students have differing accuracy values in the resulting models (M_1 and M_2) produced by the solutions. As a simple example, suppose there is a set of ten students (S_1,..., S_{10}) and all but two students (called S_1 and S_2) have 100% accuracy. Suppose in M_1, S_1 has 80% accuracy and S_2 has 50% accuracy, and in M_2, S_1 has 50% accuracy and S_2 has 80% accuracy. The average accuracy in both solutions is identical, and absent any reason to prefer S_1 or S_2 with higher accuracy, then both solutions must be considered equivalent. However, if it were possible to combine both models and identify S_1 using M_1, and S_2 using M_2, then overall accuracy would be improved as both students would have 80% accuracy (and the remaining students in either M have 100% accuracy). This phenomenon of equivalent solutions was observed when LSID-ACS [31] and LSID-ANN [31] were being developed.

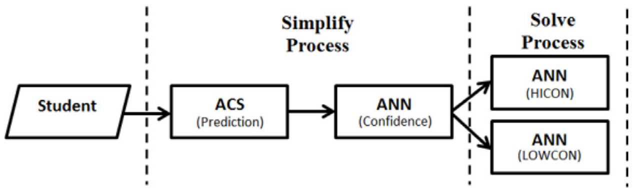

Fig. 3. LSID-SISO (ACS) Architecture.

The proposed hybrid multi-step architecture combines the mono-AI algorithms in a complex way to first simplify the problem and then solve it. This is done by feeding forward the results from one step to the next. In this fashion, the mono-AI algorithms described in Sections 4.2 and 4.3 have more information with which to produce an output. In particular, the goal of the hybrid architecture is to direct the student behavior data to a model (a trained ANN) that is most suited to identify their learning style, since as described above it has been observed that two ANNs trained separately may identify the same student with different accuracy, but with the same overall accuracy for all students. As described above, both such resulting models from the two executions are valid, albeit conflicting. This indicates that for subgroups of students, different behaviors patterns (the evidence) of their learning styles have different evidentiary value. Hence, the first consideration in designing the algorithm was to determine a process for splitting the student data into subgroups that can be identified most accurately using different behavior patterns. The first process considered, perhaps as it was the most intuitive, was to use the output from an LSID approach (e.g., LSID-ACS) to do an initial classification of the students, with students identified with a high learning style preference (≥ 0.5) sent to one algorithm and the rest to another. This resulted in no improvement to overall accuracy and on reflection this makes sense. In effect, splitting the dataset this way is roughly equivalent to the original problem: using a mono-AI algorithm to find a single solution to positively identify learning styles for an entire data set.

Accordingly, a more complex approach is needed. As such, the splitting process was improved, and two architectures are presented in this paper, each consisting of three steps (see Figs. 3 and 4). The two architectures only differ in the first step, where either an ACS or ANN is used. The first step ("prediction step") produces an initial prediction of a student's preferred learning style in a dimension. In this step, either LSID-ANN or LSID-ACS is used to identify students' learning styles. Both ACS and ANN were considered because each was best at identifying two of the four FSLSM dimensions [31].

The second step is the "splitting step" and is based on the student's behavior patterns and initial predicted learning style preference, both provided as inputs. In this step, an algorithm is trained to produce a confidence value (a real value C from 0 to 1) in the correctness of the initial prediction from step 1. In essence, this algorithm is trained to recognize the occurrence of certain

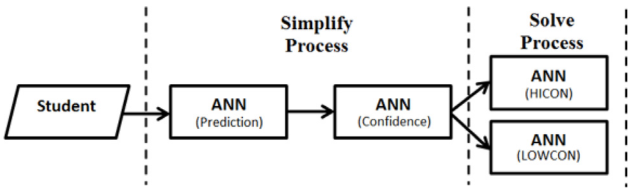

Fig. 4. LSID-SISO (ANN) architecture.

features in the behavior patterns that result in lower or higher accuracy. High confidence for LSID-SISO is defined as $C \geq 0.75$. This setting was determined experimentally by trying values from 0.50 to 0.95 in increments of 0.05.

The final step, called the "solve step", is to make a final prediction of the student's learning style using two separate algorithms called HICON and LOWCON that receive students with high confidence and low confidence values respectively. As loosely coupled hybrid architectures are intended to function by passing information forward, both the initial prediction and confidence value are included as input along with the behavior patterns. Since LSID-ANN was overall the best algorithm for identifying learning styles across all four dimensions and it is unclear how the additional information could be included in the formula used by DeLeS (and accordingly how it could be included in the weighting schema to be optimized by algorithms like ACS) both HICON and LOWCON use ANNs.

LSID-SISO is a multi-step algorithm with each step executed in sequence, and as such each step requires its own fitness function and termination condition. For the first step ("prediction step"), the same fitness function is used as for LSID-ACS and LSID-ANN (see Sections 4.2 and 4.3). For the termination condition, depending on whether an ACS or ANN is used, the termination condition is identical to either LSID-ACS or LSID-ANN.

In the second step ("confidence step") an ANN is always used; therefore, backpropagation provides the means to guide the algorithm towards increasingly more fit configurations, and ideally towards an optimal configuration. While the termination condition is the same as for LSID-ANN, the fitness function is different to the fitness function used so far since the goal of this algorithm is different, focusing on the proper classification of a student's predicted learning style from the first step with the correct high confidence and low confidence label (rather than the accuracy of learning style identification itself). Since the training data contains the actual learning style values, the correct confidence classification is computed by comparing the actual learning style to the predicted learning style. If the difference is less than 0.25, then the expected output is a high confidence classification; otherwise, a low confidence is expected. The fitness value is simply the number of correct classifications (C) divided by the number of samples (N) as shown in Formula (9).

$$F = C/N \tag{9}$$

In the third step ("solve step"), an ANN is again used. As such, the fitness function and termination condition is identical to the ones used in LSID-ANN.

The algorithms for training and using the hybrid architecture are shown in Algorithms 4.3, 4.4, and 4.5. We only show the training algorithm for LSID-SISO (ACS) since the only difference to the training algorithm of LSID-SISO (ANN) is that an ANN is used (instead of an ACS) in the prediction step.

Mathematically, the hybrid architecture adds the variables initial predicted learning style (LS_{pred}) and confidence (C). It is the addition of these variables that help refine the predictions made by the mono-AIs. The hybrid architecture feeds this information forward to help identify the proper subgroup for a student and identify them more accurately.

```
Data: A list of behavior pattern data $D$ (see Section 4.1) for a
      set of $N$ students. A confidence threshold $T$.
Result: A trained $ACS$ algorithm ($ACS_{pred}$), and three trained
        $ANN$s ($ANN_{conf}$, $LOWCON$, $HICON$).
// Assume all algorithms have a function Train(D) that
   trains the algorithm in their corresponding standard fashion

// Prediction Step - Initial prediction of learning style value;
// Assume Algorithm 4.1 has a function Predict(s) that
   returns a learning style for a student $s$;
$ACS_{pred}.Train(D)$;
// Let $s$ be the student data for a student in $D$ ;
foreach $s$ in $D$ do
    // Add the predicted learning style to the student data for
       each student $s.Add(ACS_{pred}.Predict(s))$;
end

// Confidence Step - Trains $ANN_{conf}$ to return a confidence
   value using the student data including the predicted
   learning style. Let Forward(I) be a standard forward
   propagation algorithm for an ANN using inputs $I$ that
   returns the value of the output node;
// Based on the previous step, the input array now consists of
   the behavior pattern data and the predicted learning style;
$ANN_{conf}.Train(D)$;
// Let $s$ be the student data for a student in $D$;
// Let $D_{low}$ and $D_{hi}$ be arrays that contain student data
foreach $s$ in $D$ do
    // Add the confidence value to the student data for each
       student;
    $C \longleftarrow ANN_{conf}.Forward(s)$;
    $s.Add(C)$;
    if $C \geq T$ then
        $D_{hi}.Add(s)$;
    end
    else
        $D_{low}.Add(s)$;
    end
end

// Solve Step - Train the ANNs $LOWCON$ and $HICON$ to output
   a learning style based on student input data.;
// The input arrays $D_{low}$ and $D_{hi}$ now consist of the behavior
   pattern data, the predicted learning style, and the
   confidence value;
$LOWCON.Train(D_{low})$;
$HICON.Train(D_{hi})$;
```

Algorithm 4.3: LSID-SISO (ACS): The algorithm for training LSID-SISO (ACS). Each step is trained in its entirety using the student training data plus the results from the previous step. Hence the "prediction step" uses the student training data. From the "prediction step" to the "confidence step", the initial predicted learning style is added as a feature to the data, while from the "confidence step" to the "solve step", the confidence values are added. Once trained, the algorithm can be used as shown in Algorithm 4.4.

5. Evaluation

The following section describes the materials and methods used to evaluate LSID-SISO. To begin, the data and preprocessing is discussed, including describing the validity of the data set for

```
Data: A list of behavior pattern data $D$ (see Section 4.1) for a
      student $s$. An $ACS_{pred}$ that returns a learning style for a
      student $s$ (see Algorithm 4.1). A trained ANN ($ANN_{conf}$)
      that returns a confidence value $C$ using $D$ and a
      learning style value as input. Two trained ANNs
      ($HICON$ and $LOWCON$) that return a final learning style
      value using $D$, a predicted learning style value, and a
      confidence value as input. A confidence threshold $T$.
Result: The learning style value ($LS_{final}$) in one of the four
        Felder-Silverman dimensions for student $s$.
// Prediction Step - Initial prediction of learning style value;
// Assume $ACS_{pred}$ has a function $Predict(x)$ that returns a
  learning style value given behavior data $D$ for a student $s$ as
  input;
$input_{pred} \longleftarrow D$;
$LS_{pred} \longleftarrow ACS_{pred}.Predict(input_{pred})$ (the initial predicted
  learning style value);

// Confidence Step - Compute confidence value for the
  predicted learning style;
// The input array consists of the behavior pattern data and
  the predicted learning style;
$C \longleftarrow 0$ (the confidence in the initial prediction);
$input_{conf} \longleftarrow D$;
$input_{conf}.Add(LS_{pred})$;
// Let Forward(I) be a standard forward propagation
  algorithm for an ANN using inputs $I$;
$C \longleftarrow ANN_{conf}.Forward(input_{conf})$;

// Solve Step - Compute the final learning style value;
// The input array consists of the behavior pattern data, the
  predicted learning style, and the confidence value;
$LS_{final} \longleftarrow 0$ (the final predicted learning style value);
$input_{solve} \longleftarrow D$;
$input_{solve}.Add(LS_{pred})$;
$input_{solve}.Add(C)$;
if $C \geq T$ then
    $LS_{final} = HICON.Forward(input_{solve})$;
end
else
    $LS_{final} = LOWCON.Forward(input_{solve})$;
end
```

Algorithm 4.4: LSID-SISO (ACS): An algorithm for identifying the learning style of a student from behavior pattern data using an *ACS* and *ANN*s. This represents LSID-SISO if it were part of a learning management system. It would receive student behavior data as input, and using the multistep process, return a learning style.

```
Data: A list of behavior pattern data $D$ (see Section 4.1) for a
      student $s$. A trained ANN, $ANN_{pred}$, that returns an
      initial predicated learning style value using $D$ as input.
      A trained ANN, $ANN_{conf}$, that returns a confidence value
      $C$ using $D$ and a learning style value as input. Two
      trained ANNs ($HICON$ and $LOWCON$) that return a final
      learning style value using $D$, a predicted learning style
      value, and a confidence value as input. A confidence
      threshold $T$.
Result: The learning style value in one of the four
        Felder-Silverman dimensions for student $s$.
// Prediction Step - Initial prediction of learning style value;
$input_{pred} \longleftarrow D$;
$LS_{pred} = ANN_{pred}.Forward(input_{pred})$ (the initial predicted
  learning style value);

// Confidence Step - Compute confidence value for the
  predicted learning style;

// Solve Step - Compute the final learning style value;

// Confidence and Solve Steps are identical to Algorithm 4.4;
```

Algorithm 4.5: LSID-SISO (ANN): An algorithm for identifying student learning styles from behavior pattern data using *ANN*s in a loosely couple hybrid architecture. This represents the pseudocode for LSID-SISO (ANN) as if it were part of a learning management system, and hence takes one student's behavior data as input and returns a learning style.

evaluating the algorithm. This is followed by a description of the metrics used to assess LSID-SISO's ability to identify learning styles. As the function of the algorithms selected are parameter dependent, Section 5.3 describes the bounded search techniques used to optimize the parameters, and then Section 5.4 briefly describes the techniques used to reduce overfitting. Finally, this section concludes with a presentation of the results of the evaluation, and a discussion on the results and observations made while developing LSID-SISO.

5.1. Data

The data set used to evaluate LSID-SISO is the same data set used for evaluating LSID-ACS [31], LSID-ANN [31] and DeLeS [39]. The data set consists of both behavior data and the student's actual learning styles (as identified by the ILS questionnaire [27]) for 127 students from an undergraduate computer science/information technology course. Preprocessing of the data was done identically as for DeLeS [39]. To ensure the identified learning styles are reliable, any student who spent less than 5 minutes filling in the questionnaire was eliminated from the data set. In addition, to ensure there is sufficient data about each student, only students who submitted more than half of the assignments and took the final exam were used. After these removals, the final dataset consists of 75 students.

The validity of this data set is based on two criteria: size and distribution of learning styles. With respect to size, our data set is of similar size to those used in related works. For example, García et al. [33] used data from 77 students in their data set (50 for training and 27 for testing). Another example is the work by Özpolat and Akar [38] who used data from 40 students in their data set (10 for training and 30 for testing).

To ensure that the LSID data set fairly represents learning styles for students, the distribution of learning styles is examined. Table 2 shows the percentage of students in the data set with an active, sensing, visual or sequential learning style (shown in the "LSID" row) and the range of values found in literature [1]. The distribution of learning styles for the data used by this research is well within the range of expected values.

A 10-fold cross validation process is used for control parameter optimization, evaluating overfitting reduction and producing a final result to ensure that LSID-SISO is generalized by exposing the approaches to different data sets. With the 10-fold cross validation process, the algorithm is executed 10 times with the results averaged over the 10 executions each with a different training and assessment data sets, i.e., a fold, extracted from the overall data set. For each fold, 1/10th of the students is selected for the assessment set and chosen such that each student is selected for only a single fold's assessment set; thereby, guaranteeing that each assessment set is unique. Additionally 1/10th of the students are selected as a validation set, such that each

Table 2
Comparison of the distribution of learning styles.

	Active	Sensing	Visual	Sequential
LSID	52.7%	63.0%	81.9%	53.4%
Felder and Spurlin [1]	47 − 70%	46 − 81%	66 − 91%	45 − 85%

student is selected for a validation set only once. A student may not be both in the assessment and validation sets. The remaining unselected students are used as the training data for the fold.

5.2. Performance metrics

Four performance metrics are used to measure the performance of LSID-SISO. Two of the metrics, similarity (*SIM*) and accuracy (*ACC*), measure the overall performance, while the remaining two, lowest *ACC* (*LACC*) and percentage matched with reasonable accuracy (*%Match*), measure the performance for individual students.

The first metric, *SIM*, is a classification metric used commonly by other approaches [33,39]. *SIM* works by assuming that learning styles are divided into three regions, for example, with the A/R dimension: active, reflective and balanced. Then, the student's actual learning style (LS_{actual}) and identified learning style (LS_{id}) are compared. As shown in Formula (10), if the values are in the same region, the *SIM* value is 1, if they are in adjacent regions, i.e., one is balanced and the other indicating a preference, the *SIM* value is 0.5 and finally if they are in opposite regions the *SIM* value is 0.

$$SIM = \begin{cases} 1.0 & \text{if } LS_{id} = LS_{actual} \\ 0.5 & \text{if } LS_{id} \neq LS_{actual} \\ & \text{and } (LS_{id} = balanced \\ & \text{or } LS_{actual} = balanced) \\ 0.0 & otherwise \end{cases} \tag{10}$$

The *SIM* metric is suitable for classification algorithms that identify learning styles with a label, e.g., strong, neutral or low. However, this may cause a misleading result when the LS_{id} or LS_{actual} are near the thresholds for a label. Consider a student with $LS_{actual} = 0.76$ and $LS_{id} = 0.74$. This is considered only a moderate match by *SIM*, while it is actually a near perfect match. Since LSID-SISO produces a numeric value to represent a student's learning styles preference, the second metric used is accuracy (*ACC*). *ACC* is computed as 1 minus the absolute difference between LS_{id} and LS_{actual}, i.e., the error (shown in Formula (11)). Both the *SIM* and *ACC* metrics are reported as averages over all students across all 10 folds.

$$ACC = 1 - |LS_{id} - LS_{actual}| \,. \tag{11}$$

Although the preceding metrics are useful for measuring the overall performance, when using a metric based on an average it might be that some individual values across the set of results are low. A poor result for an individual student has the possibility of a corresponding negative impact on the student as they may be provided with mismatched material from an adaptive learning system or misleading advice from a teacher. To address this and identify any possible deficiency, the results are examined on a student-by-student basis and the *LACC* and *%Match* metrics are computed. *LACC* is the lowest *ACC* value for any student and so measures the worst-case scenario for any single student (shown in Formula (12)). *%Match* measures how many students are identified reasonably well, thus also implying a percentage identified poorly. The criterion for considering a match to be reasonably good was to have an *ACC* result greater than half the range of actual values in the dataset. Therefore, a reasonable match is defined as having an *ACC* result greater than or equal to 0.5 and encoded as a value of 1. Inversely, an encoding of 0 is used for an unreasonable match ($ACC < 0.5$). Formula (13) shows how the *%Match* is computed as the average of the sum of the encoded values for each student across all 10 folds.

$$LACC = \min_{0<x<n} ACC(LS_{actual,x}, LS_{id,x}) \tag{12}$$

$$\%Match = \frac{\sum_{x=1}^{n} \begin{cases} 0.0 & \text{if} ACC\left(LS_{actual,x}, LS_{id,x}\right) < 0.5 \\ 1.0 & otherwise \end{cases}}{n} \tag{13}$$

5.3. Parameter tuning

The functioning of CI algorithms is dependent on control parameter settings, and optimal settings are problem dependent. Therefore, an iterative bounded search was used where different parameter settings are evaluated. The settings that produce the best results are selected, and then reexecuted to produce a final result, which is reported in Section 5.5. The following process was used for both ACS and ANN with the only difference being the specific parameters optimized.

Although the literature does not define optimal settings, as they are problem specific, the literature does provide some guiding principles for setting the control parameters [41,48,50–55]. After deciding on a suitable range of parameter values, a default value (typically mid-range) is selected for each control parameter (shown in bold below). The algorithm is then executed iteratively over the set of values for the 1st control parameter, with the remainder using the default values, and a result is obtained. The value for the parameter that produces the best result is considered the optimal parameter setting. The iterative process is then repeated for each parameter, using the already identified optimal parameters.

The control parameters for the ANN are optimized in the following order: number of hidden nodes (H), learning rate (η), momentum (α) and training mode. An H value between $logT$ [55] (where T is the size of the training set) and 2× the number of inputs [54] was selected. In this case, the lower bound is $log67$ or 1.82 and the upper bound varies by learning styles dimension since the number of behavior patterns (inputs) varies by learning styles dimension. As evaluating more values will increase the chance of finding the optimal parameters, the lower bound is reduced to 1 instead of rounded up to 2. For learning rate, a low value is suggested [54] so the values evaluated were between 0.01 and 0.1 in steps of 0.01. Momentum is also recommended to be low so that it does not cause the ANN to skip past good areas during training [54]. So, the values evaluated for momentum are 0 to 0.1 in steps of 0.01. Both individual and ensemble training modes are evaluated for LSID-ANN.

The control parameters for ACS were optimized in the following order: population size (P), evaporation ratio (ρ), consumption ratio (τ_0), and exploitation factor (q_0). As mentioned above, α and β are both set to 1 since no local information is used. The recommended value for population size ranges from 10 to 100 [50,52, 53]; however, as the graph for learning style identification is quite a bit bigger than those in the literature, it was expanded to give a set of values (10, 25, 50, 100, 200). The evaporation ratio plays a significant role in determining the preference towards an exploration vs. exploitation strategy by affecting how long pheromone persists. A higher evaporation ratio lowers the pheromone persistance and so creates a preference for exploration. The evaporation

Table 3
Prediction step: Optimal ANN parameter settings for LSID-SISO (ANN).

	H	η	α	Training mode
A/R	1	0.08	0.10	Individual
S/I	5	0.06	0.09	Individual
V/V	8	0.08	0.06	Individual
S/G	2	0.07	0.01	Individual

Table 4
Confidence step: Optimal ANN parameter settings for LSID-SISO (ANN).

	H	η	α	Training mode
A/R	3	0.07	0.01	Individual
S/I	8	0.05	0.04	Individual
V/V	6	0.06	0.02	Individual
S/G	7	0.03	0.00	Individual

ratio is generally preferred to be somewhat high [41] and so the set of values selected is $(0.5, 0.6, \mathbf{0.7}, 0.8, 0.9)$. As the consumption ratio also affects the pheromone on links it also influences the preference towards exploration and exploitation. However, unlike the evaporation ratio, it is generally preferred to be small otherwise all of the pheromone is quickly consumed by the ants traversing links [41]. The set of values for the consumption ratio is $(0.01, 0.05, \mathbf{0.10}, 0.20, 0.30)$. The exploitation factor is generally preferred to be high so that the ants will use previously found good solutions [41]. While the exploitation parameter is not explicitly associated solely with local information, as all local information is set to the same value (i.e., 1), it is possible that the exploitation parameter could cause some confusion for the ants. Therefore, $q_0 = 0.0$ was also evaluated giving the set of values $(0.0, 0.5, 0.6, \mathbf{0.7}, 0.8, 0.9)$. As can be seen in Table 6, as hypothesized, the exploitation parameter being off was the optimal setting.

Tables 3, 4, 5 show the optimal parameter settings for the prediction, confidence and solve steps for LSID-SISO (ANN). Tables 6–8 show the optimal parameter settings for the prediction, confidence and solve steps for LSID-SISO (ACS).

5.4. Overfitting reduction strategies

Overfitting is a common problem with machine learning and computational intelligence algorithms, where patterns in the data are memorized into the model such that it fits well during experimentation, but would not actually be a good model in general [40,56,57]. For this research, stratification [56] was used to help prevent overfitting. Weight decay [57] was also used when training the neural networks as it is reported to reduce overfitting by preferring lower weights on the edges of the network. The optimal weight decay for the different ANNs are shown in Tables 9 and 10.

Stratification [56] can be employed when N-folds are used for creating training data sets. It works by selecting samples such that the training and testing sets from producing a fold have a sample distribution similar to the entire data set. In principle, if the entire data set is a valid representative of reality, which was tested for this research (shown in Table 2), then the resulting model is more likely to be generalized to reality. For this research, the distribution of learning style preferences for each fold was made to resemble the distribution found by Felder and Spurlin [1] as close as possible since picking randomly could result in distributions that would not reflect the general population and therefore, could affect accuracy when tested with different data sets.

Weight decay functions simply by removing a percentage $(0 < \lambda < 1)$ of the weight from each neural link in an ANN. This was found to provide better generalization of ANNs [57]. If the optimal configuration is to have a high weight on a link, then excessive weight decay will make it difficult to achieve, hence, it is recommended that weight decay be kept low. As such, the values assessed for weight decay were $(0.00, 0.001, 0.01, 0.02, 0.03, 0.04, 0.05)$.

5.5. Results and discussion

The following subsection provides the results of the evaluation for both variants of LSID-SISO. Table 11 shows the *SIM* results for both LSIS-SISO variants and several related works which also provided *SIM* values. Even though the LSID-SISO variants do not optimize based on *SIM* values but based on accuracy values (*ACC*), they achieved good results. In examining the *SIM* values, a different algorithm is best for each FSLSM dimension. In the V/V dimension, LSID-SISO (ACS) provides a notable difference having a *SIM* value of 0.827 compared to the second-best algorithm (LSID-ACS) achieving a value of 0.771. In the A/R dimension, LSID-ACS achieved the best performance with a *SIM* value of 0.804, closely followed by the two variants of LSID-SISO and LSID-ANN, all three achieving a *SIM* value of 0.802. In the S/G dimension, LSID-ANN achieved the best result (0.825), with LSID-SISO (ACS) ranking second with a *SIM* value of 0.804. In the S/I dimension, DeLeS achieved the highest result with a *SIM* value of 0.773 while the LSID-SISO variants only rank on fourth and fifth position with *SIM* values of 0.761 and 0.755.

Table 12 shows the *ACC*, *LACC* and *%Match* results for DeLeS, LSID-ANN, LSID-ACS and LSID-SISO for each FSLSM dimension. For those approaches, the respective performance metrics were available and according to the *SIM* results presented in Table 11 those approaches are also the most relevant works to compare the new hybrid architecture with. Comparing the results for LSID-SISO (ACS) against the other approaches shows that LSID-SISO (ACS) is the top (or tied for top) approach in each FSLSM dimension for every metric except *LACC* for the A/R and S/G dimension where LSID-SISO (ANN) and LSID-ANN respectively achieve best results.

In particular, LSID-SISO (ACS) improves the *%Match* metric to 1.000 for every FSLSM dimension. In a practical sense, in having *%Match* at 100% means that no student is identified poorly. Although the absolute increase for *%Match* is generally fairly small, it is very significant for those individual students since

Table 5
Solve step: Optimal ANN parameter settings for LSID-SISO (ANN).

		H	η	α	Training mode
A/R	HICON	4	0.06	0.01	Individual
	LOWCON	3	0.05	0.01	Individual
S/I	HICON	5	0.03	0.04	Individual
	LOWCON	6	0.03	0.01	Individual
V/V	HICON	3	0.03	0.03	Individual
	LOWCON	3	0.06	0.02	Individual
S/G	HICON	7	0.02	0.01	Individual
	LOWCON	9	0.04	0.01	Individual

Table 6
Prediction step: Optimal ACS parameter settings for LSID-SISO (ACS).

	P	ρ	τ_0	q_0
A/R	100	0.80	0.20	0.0
S/I	200	0.80	0.20	0.0
V/V	50	0.90	0.05	0.0
S/G	200	0.50	0.20	0.0

Table 7
Confidence step: Optimal ANN parameter settings for LSID-SISO (ACS).

	H	η	α	Training mode
A/R	2	0.08	0.01	Individual
S/I	8	0.03	0.01	Individual
V/V	3	0.02	0.05	Individual
S/G	9	0.01	0.02	Individual

having poor advice can have a serious negative impact on their education.

Improvements in the *LACC* metric mean even for the student with the worst identification, any advice will be closer to their true preference. For this metric LSID-SISO (ACS) achieved the highest result for two dimensions (S/I and V/V), while LSID-SISO (ANN) performed best for the A/R dimension and LSID-ANN performed best for the S/G dimension. The improvements in the *ACC* metric mean that, in general, guidance will be more precise to the students' true learning style preferences. Although the improvements are small, every bit helps individual students learn the best they can.

In examining the results for LSID-SISO (ANN), they are quite mixed and overall, not as good as LSID-SISO (ACS). Only for *LACC* in the A/R and S/G dimensions is LSID-SISO (ANN) better than LSID-SISO (ACS) (with the best result in the A/R dimension and the second-best result in the S/G dimension). The likely explanation for this is that the LSID-SISO (ANN) architecture is similar to having a 9-layer ANN and adding additional layers as a hybrid architecture does not increase the ability of the algorithm to describe the relationship between the behavior patterns and FSLSM dimensions as much as adding an entirely different algorithm such as ACS. It is possible that adding additional layers in a different fashion, for example, a deep learning neural network [58], could provide different results and this would be an area for future investigation.

Having compared the algorithms overall, the next few paragraphs describe how the improvements were obtained for the S/I, V/V and S/G dimensions by looking at the results from the perspective of individual students. No discussion is made on the A/R dimensions as the *ACC* and *%Match* results were identical to LSID-ACS and no telling observations were made in the analysis of the *LACC* results. The remainder of this section focuses on LSID-SISO (ACS) as it has better overall results than LSID-SISO (ANN).

The analysis of the individual results in the S/I dimension showed a general improvement in *ACC* for all students. No student was identified worse by LSID-SISO (ACS) than LSID-ACS (which is the best mono-CI approach for the S/I dimension). By improving the *ACC*, the *LACC* and *%Match* metrics were also improved.

For the V/V dimension, an analysis of identified values from LSID-ANN (which is the best mono-CI approach for the V/V dimension) shows that students with a learning style value > 0.5 (those with a visual or balanced towards visual preference) are identified very well and students with a value < 0.5 (those with a verbal or balanced towards verbal preference) less so. The reason for this is that about 85% of the students in the data set have a visual or balanced towards visual preference, hence it is optimal to identify those students well. Promisingly, the improvement for LSID-SISO (ACS) is obtained almost entirely by improving the accuracy in identifying students with a verbal or balanced towards verbal preference (shown in Table 13) with an increase in *ACC* per student ranging from 0.078 to 0.352. However, despite the improvements, two students are still identified with a visual preference with learning style values > 0.666 (student #175 and #593) although having a balanced towards verbal preference (values between 0.333 and 0.5). In addition, the remaining four students who have a balanced towards verbal preference were identified with a balanced towards visual preference (values between 0.5 and 0.666). Out of the five learners with a verbal learning style preference (values < 0.333), one was identified with a balanced towards verbal preference (student #255) while the other four were identified with a balanced towards visual preference. To improve the identification of students with a verbal or balanced towards verbal preference further, a modified fitness function may provide a fairer result by including *LACC* or *%Match* in computing the fitness values. Alternatively, the students with a visual preference could be weighted to provide a smaller contribution to the fitness value.

For the S/G dimension, the increase in accuracy compared to the best mono-CI algorithm is small and split among many students. In six of the folds a single student had a significant drop in accuracy, hence the drop in *LACC*. No correlation to learning style preference is found as four of the six students have a sequential preference. However, it was observed that for those six students the confidence values were not accurate, although no common feature of the students' data could be found to explain the cause for the inaccuracy.

Overall, using confidence to separate the students worked well at providing an improvement in the performance metrics. But it is important to see if the algorithm is dividing them in some meaningful, justifiable way. As a neural network was used, the resulting model does not have a high degree of interpretability; therefore, this was examined statistically. Table 14 shows the percentage of students who were sent to the proper solver (HICON or LOWCON), i.e., had their confidence properly identified. A student is considered to have been sent to the correct solver if: (1) a student's *ACC* value based on the predictor's identification is $<$ 0.75 and this student's confidence value was identified to be low (< 0.75) and therefore the student was sent to LOWCON or (2) a student's *ACC* value based on the predictor's identification

Table 8
Solve step: Optimal ANN parameter settings for LSID-SISO (ACS).

		H	η	α	Training mode
A/R	HICON	2	0.08	0.01	Individual
	LOWCON	5	0.06	0.02	Individual
S/I	HICON	2	0.04	0.03	Individual
	LOWCON	7	0.03	0.02	Individual
V/V	HICON	3	0.02	0.05	Individual
	LOWCON	3	0.03	0.03	Individual
S/G	HICON	8	0.01	0.00	Individual
	LOWCON	7	0.04	0.03	Individual

Table 9
Optimal ANN weight decay settings for LSID-SISO (ANN).

	Prediction	Confidence	HICON	LOWCON
A/R	0.05	0.02	0.03	0.03
S/I	0.05	0.02	0.01	0.00
V/V	0.01	0.01	0.01	0.01
S/G	0.10	0.03	0.02	0.02

Table 10
Optimal ANN weight decay settings for LSID-SISO (ACS).

	Prediction	Confidence	HICON	LOWCON
A/R	n/a	0.02	0.05	0.03
S/I	n/a	0.01	0.01	0.00
V/V	n/a	0.01	0.01	0.01
S/G	n/a	0.02	0.01	0.02

is ≥ 0.75 and this student's confidence value was identified to be high (≥ 0.75) and therefore, the student was sent to HICON.

As a first attempt for expressing confidence in the accuracy of a learning style prediction the correctness is subjectively promising, varying between 72% and 85%. As random guessing should result in a 50/50 split, there appears to be some (unknown) reasoning to the ANN used for the confidence step. As the architecture overall shows some potential to improve accuracy, it would be worth investigating this step more closely and look for alternatives. It might be helpful to use an algorithm that provides an explainable model (e.g., a decision tree). First, such a model might help verify the logic used by the algorithm to separate the data, but it also may reveal some pedagogical understanding about the differences in the behavior patterns and how they relate to learning styles.

6. Conclusions

This paper has introduced a new hybrid architecture called "Learning Style Identifier – Simplify and Solve" (LSID-SISO) for identifying the learning style preferences of students from their behavior patterns when using a learning management system (LMS). The underlying principle of LSID-SISO is that unique non-optimal solutions to an optimization problem may have identical fitness values but may consist of different parts (e.g., a single student's identified learning style) where some parts are of higher quality and some are of lower quality. By combining the parts that are of higher quality, a better solution can be constructed. In the context of learning style identification, this means that if the models from different executions of an algorithm can be preserved (which is trivial, it is just a configuration), and each student could be identified by the model which best identified them, then overall accuracy can be improved. This defines the problem into two broad steps: splitting the data as optimally as possible and identifying the learning styles. It is hypothesized that splitting the data makes training the model a simpler problem as the data should be more homogeneous, hence the architecture's name "Simplify and Solve".

To accomplish this, a loosely coupled multi-step hybrid architecture was designed where the main feature is the computation of additional information at each step to feed forward into the next step of the architecture. For identifying learning styles, the architecture was broken down into three steps. The first step was to produce an initial prediction of the learning style preference. This initial prediction was combined with the behavior data in the second step to compute a confidence value (C) in the initial prediction. The data is then split based on the confidence value, with $C \geq 0.75$ considered high confidence, and the remainder as low confidence. The third and final step has two identification algorithms, one each for high confidence and low confidence data.

Overall, it was found that there was an improvement in accuracy for three of the four FSLSM dimensions compared to the leading current approaches. In the fourth FSLSM dimension, the architecture produced the same accuracy as the leading mono-CI approach and in addition, there was an improvement in the worst-case scenario (identified by finding the lowest individual accuracy for any student in the data set). Therefore, the concept of separating data based on a computed confidence value seems

Table 11
Comparison of *SIM* results (ranks in parenthesis and top result bolded).

	Approach	*SIM*
A/R	Bayesian [33]	0.580 (5)
	Naïve Bayes Tree [38]	0.700 (4)
	DeLeS [39]	0.793 (3)
	LSID-ANN [31]	0.802 (2)
	LSID-ACS [31]	**0.804** (1)
	LSID-SISO (ACS)	0.802 (2)
	LSID-SISO (ANN)	0.802 (2)
S/I	Bayesian [33]	0.770 (2)
	Naïve Bayes Tree [38]	0.733 (7)
	DeLeS [39]	**0.773** (1)
	LSID-ANN [31]	0.741 (6)
	LSID-ACS [31]	0.762 (3)
	LSID-SISO (ACS)	0.761 (4)
	LSID-SISO (ANN)	0.755 (5)
V/V	Bayesian [33]	–
	Naïve Bayes Tree [38]	0.533 (6)
	DeLeS [39]	0.767 (3)
	LSID-ANN [31]	0.727 (5)
	LSID-ACS [31]	0.771 (2)
	LSID-SISO (ACS)	**0.827** (1)
	LSID-SISO (ANN)	0.739 (4)
S/G	Bayesian [33]	0.630 (6)
	Naïve Bayes Tree [38]	0.733 (5)
	DeLeS [39]	0.733 (5)
	LSID-ANN [31]	**0.825** (1)
	LSID-ACS [31]	0.785 (3)
	LSID-SISO (ACS)	0.804 (2)
	LSID-SISO (ANN)	0.780 (4)

Table 12
Comparison of *ACC*, *LACC*, *%Match* results (ranks in parenthesis and top result bolded).

	Approach	*ACC*	*LACC*	*%Match*
A/R	DeLeS	0.799 (4)	0.435 (5)	0.987 (2)
	LSID-ANN	0.802 (3)	0.610 (3)	0.986 (3)
	LSID-ACS	**0.819** (1)	0.599 (4)	**1.000** (1)
	LSID-SISO (ACS)	**0.819** (1)	0.615 (2)	**1.000** (1)
	LSID-SISO (ANN)	0.813 (2)	**0.627** (1)	**1.000** (1)
S/I	DeLeS	0.790 (4)	0.389 (5)	0.960 (4)
	LSID-ANN	0.790 (4)	0.575 (3)	0.961 (3)
	LSID-ACS	0.797 (3)	0.583 (2)	0.971 (2)
	LSID-SISO (ACS)	**0.814** (1)	**0.608** (1)	**1.000** (1)
	LSID-SISO (ANN)	0.800 (2)	0.573 (4)	0.960 (4)
V/V	DeLeS	0.788 (5)	0.226 (5)	0.987 (2)
	LSID-ANN	0.840 (3)	0.656 (2)	0.986 (3)
	LSID-ACS	0.799 (4)	0.534 (4)	0.909 (4)
	LSID-SISO (ACS)	**0.861** (1)	**0.673** (1)	**1.000** (1)
	LSID-SISO (ANN)	0.844 (2)	0.638 (3)	0.986 (3)
S/G	DeLeS	0.702 (5)	0.134 (5)	0.880 (3)
	LSID-ANN	0.797 (2)	**0.613** (1)	0.986 (2)
	LSID-ACS	0.737 (4)	0.426 (4)	0.879 (4)
	LSID-SISO (ACS)	**0.802** (1)	0.583 (3)	**1.000** (1)
	LSID-SISO (ANN)	0.796 (3)	0.608 (2)	**1.000** (1)

Table 13
ILS, LSID-ANN and LSID-SISO (ACS) identified learning style and *ACC* for verbal students.

Student ID	ILS	LSID-ANN	*ACC*	LSID-SISO (ACS)	*ACC*	*ΔACC*
75	0.438	0.726	0.712	0.648	0.790	0.078
175	0.438	0.805	0.633	0.683	0.755	0.122
200	0.438	0.691	0.747	0.583	0.855	0.108
242	0.438	0.837	0.601	0.621	0.817	0.216
295	0.438	0.831	0.607	0.630	0.808	0.201
593	0.438	0.759	0.679	0.675	0.763	0.084
129	0.313	0.693	0.620	0.589	0.724	0.104
177	0.313	0.690	0.623	0.590	0.723	0.100
225	0.313	0.704	0.609	0.515	0.798	0.189
72	0.214	0.756	0.458	0.590	0.624	0.166
255	0.214	0.687	0.527	0.335	0.879	0.352

Table 14
Percentage of students sent to proper solver.

	HICON % Correct	LOWCON % Correct
A/R	72%	84%
S/I	83%	74%
V/V	85%	73%
S/G	73%	80%

to have potential, and the algorithm did not appear to be guessing as 72%...85% of all students were split into the correct subset.

By improving the precision of learning styles identification, students benefit in three ways. By knowing their learning styles, a student can better self-regulate their learning and capitalize on their strengths. In addition, when they struggle it may help them understand why. Similarly, teachers are provided some insight into their struggling students which can aid them in providing more appropriate interventions. Lastly, by providing adaptive learning systems with this information, they can match content, recommendations and/or interfaces to a student's preferred learning styles, which may lead to improved performance [10,13,20], increased satisfaction with the content/course [11,13,19], and a reduction in the time to learn [12,21,22].

The architecture proposed in this paper can be integrated into LMSs like Moodle which are used by millions of learners. There exist many LMS plugins that consider learning styles. For example, some plugins provide information about students' learning styles to learners or teachers [7]. Other examples include plugins that create adaptive courses that are personalized to a student's learning style [59,60] or that provide recommendations to learners, for example, on learning objects within the course [61,62], from a repository [63] or from the web [64]. Most of these plugins use a questionnaire to identify a student's learning style, which – as discussed before – has several disadvantages from the additional time that students need to spend to fill out the questionnaire to problems with the accuracy of their answers. Our proposed approach could extend those plugins by replacing the questionnaire with an automatic approach for identifying learning styles, therefore, automatically identifying the learning styles of students by observing their behaviors. Even plugins that currently use an automatic approach can benefit from our algorithm as our algorithm reaches very high accuracy and therefore can be used to replace less accurate approaches. More accurate identification of learning styles leads to more accurate personalization and allows students to optimally benefit from the personalized support that respective plugins provide.

A limitation of this work is the size of the data set as well as that the data are collected from students in only one course. A data set in the size of 75 students, as used in this study, is common for studies on identifying learning styles and is at the higher end compared to other related studies. Using students from only one course is very common as well. In fact, there is not a single study on identifying learning styles that uses data from students in different study programs or disciplines. While limited, such data sets have shown promising results in related works and also in this work. However, a larger data set, preferably with students from courses in different study programs and disciplines would provide results that are more generalizable. Using such a large data set with LSID-SISO will be one of the directions for future work, to show, on one hand, the generalizability of LSID-SISO across disciplines and, on the other hand, to get further insights into the relation between learning styles and student behaviors across disciplines.

Several other directions for the future work on learning style identification are planned rests in several directions. For example, LSID-SISO was designed to maximize the *ACC* results, and although the improvements obtained are important for individual students there is much more room for improvement in the *LACC* metric than the *ACC* metric. Future algorithms can focus on maximizing both *ACC* and *LACC*. A further investigation will be done into understanding the relationship found by the ANN used in the confidence step to split the student data, in particular, an algorithm that builds an explainable model should be employed to understand the underlying logic being used. This may include looking at alternate methods for training the ANNs in lieu of backpropagation.

CRediT authorship contribution statement

Jason Bernard: Conceptualization, Methodology, Software, Validation Formal analysis, Investigation, Writing – original draft. **Elvira Popescu:** Writing – review & editing, Supervision. **Sabine Graf:** Conceptualization, Data curation, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgments

The authors wish to thank Athabasca University, Alberta Innovates – Technology Futures, Alberta Innovation and Advanced Education, and the Natural Sciences and Engineering Research Council of Canada (NSERC) [funding reference number: RGPIN-2020-05837] for their support of this research.

References

[1] R.M. Felder, J. Spurlin, Applications, reliability and validity of the index of learning styles, Int. J. Appl. Eng. Educ. 21 (1) (2005) 103–112.
[2] P. Honey, A. Mumford, The Manual of Learning Styles, Peter Honey Maidenhead, 1992.
[3] J.W. Keefe, Learning style: An overview, Stud. Learn. Styles: Diagn. Prescr. Programs 1 (1) (1979) 1–17.
[4] C. Mejia, B. Florian, R. Vatrapu, S. Bull, S. Gomez, R. Fabregat, A novel web-based approach for visualization and inspection of reading difficulties on university students, IEEE Trans. Learn. Technol. 10 (1) (2016) 53–67.
[5] T. Vasileva-Stojanovska, M. Vasileva, T. Malinovski, V. Trajkovik, An ANFIS model of quality of experience prediction in education, Appl. Soft Comput. 34 (2015) 129–138.
[6] G. Cheng, J. Chau, Exploring the relationships between learning styles, online participation, learning achievement and course satisfaction: An empirical study of a blended learning course, Br. J. Educ. Technol. 47 (2) (2016) 257–278.
[7] M.M. El-Bishouty, A. Aldraiweesh, U. Alturki, R. Tortorella, J. Yang, T.-W. Chang, S. Graf, et al., Use of Felder and Silverman learning style model for online course design, Educ. Technol. Res. Dev. 67 (1) (2019) 161–177.
[8] R.M. Felder, L.K. Silverman, Learning and teaching styles in engineering education, Eng. Educ. 78 (7) (1988) 674–681.
[9] S. Graf, Adaptivity in learning management systems focussing on learning styles (Ph.D. thesis), Vienna University of Technology, 2007.
[10] F. Mampadi, S.Y. Chen, G. Ghinea, M.-P. Chen, Design of adaptive hypermedia learning systems: A cognitive style approach, Comput. Educ. 56 (4) (2011) 1003–1011.
[11] E. Popescu, Adaptation provisioning with respect to learning styles in a Web-based educational system: an experimental study, J. Comput. Assist. Learn. 26 (4) (2010) 243–257.
[12] M. Soflano, T.M. Connolly, T. Hainey, An application of adaptive games-based learning based on learning style to teach SQL, Comput. Educ. 86 (2015) 192–211.
[13] C. Troussas, F. Giannakas, C. Sgouropoulou, I. Voyiatzis, Collaborative activities recommendation based on students' collaborative learning styles using ANN and WSM, Interact. Learn. Environ. (2020) 1–14.
[14] A. Klašnja-Milićević, M. Ivanović, B. Vesin, Z. Budimac, Enhancing e-learning systems with personalized recommendation based on collaborative tagging techniques, Appl. Intell. 48 (6) (2018) 1519–1535.
[15] A.F. Martins, M. Machado, H.S. Bernardino, J.F. de Souza, A comparative analysis of metaheuristics applied to adaptive curriculum sequencing, Soft Comput. 25 (16) (2021) 11019–11034.
[16] Y. Akbulut, C.S. Cardak, Adaptive educational hypermedia accommodating learning styles: A content analysis of publications from 2000 to 2011, Comput. Educ. 58 (2) (2012) 835–842.
[17] H.M. Truong, Integrating learning styles and adaptive e-learning system: Current developments, problems and opportunities, Comput. Hum. Behav. 55 (2016) 1185–1193.
[18] A.H. Nabizadeh, J.P. Leal, H.N. Rafsanjani, R.R. Shah, Learning path personalization and recommendation methods: A survey of the state-of-the-art, Expert Syst. Appl. 159 (2020) 113596.
[19] Ö. Özyurt, H. Özyurt, A. Baki, Design and development of an innovative individualized adaptive and intelligent e-learning system for teaching–learning of probability unit: Details of UZWEBMAT, Expert Syst. Appl. 40 (8) (2013) 2914–2940.
[20] A. Latham, K. Crockett, D. McLean, An adaptation algorithm for an intelligent natural language tutoring system, Comput. Educ. 71 (2014) 97–110.
[21] A. Klašnja-Milićević, B. Vesin, M. Ivanović, Z. Budimac, E-learning personalization based on hybrid recommendation strategy and learning style identification, Comput. Educ. 56 (3) (2011) 885–899.
[22] J.C. Tseng, H.-C. Chu, G.-J. Hwang, C.-C. Tsai, Development of an adaptive learning system with two sources of personalization information, Comput. Educ. 51 (2) (2008) 776–786.
[23] F. Coffield, D. Moseley, E. Hall, K. Ecclestone, F. Coffield, D. Moseley, E. Hall, K. Ecclestone, et al., Learning Styles and Pedagogy in Post-16 Learning: A Systematic and Critical Review, Learning and Skills Research Council, 2004.
[24] P.A. Kirschner, J.J. van Merriënboer, Do learners really know best? Urban legends in education, Educ. Psychol. 48 (3) (2013) 169–183.
[25] E. Dahlstrom, D.C. Brooks, J. Bichsel, The current ecosystem of learning management systems in higher education: Student, faculty, and IT perspectives, 2014.
[26] J. Kuljis, F. Liu, A comparison of learning style theories on the suitability for eLearning, Web Technol. Appl. Serv. 2005 (2005) 191–197.
[27] R. Felder, B. Soloman, Index of learning styles, 1998, Available at: https://educationdesignsinc.com/index-of-learning-styles.
[28] E. Popescu, Diagnosing students' learning style in an educational hypermedia system, in: Cognitive and Emotional Processes in Web-Based Education: Integrating Human Factors and Personalization, IGI Global, 2009, pp. 187–208.
[29] N.B.A. Normadhi, L. Shuib, H.N.M. Nasir, A. Bimba, N. Idris, V. Balakrishnan, Identification of personal traits in adaptive learning environment: Systematic literature review, Comput. Educ. 130 (2019) 168–190.
[30] J. Feldman, A. Monteserin, A. Amandi, Automatic detection of learning styles: state of the art, Artif. Intell. Rev. 44 (2) (2015) 157–186.
[31] J. Bernard, T.-W. Chang, E. Popescu, S. Graf, Learning style identifier: Improving the precision of learning style identification through computational intelligence algorithms, Expert Syst. Appl. 75 (2017) 94–108.
[32] F.A. Dorça, L.V. Lima, M.A. Fernandes, C.R. Lopes, Comparing strategies for modeling students learning styles through reinforcement learning in adaptive and intelligent educational systems: An experimental analysis, Expert Syst. Appl. 40 (6) (2013) 2092–2101.
[33] P. García, A. Amandi, S. Schiaffino, M. Campo, Evaluating Bayesian networks' precision for detecting students' learning styles, Comput. Educ. 49 (3) (2007) 794–808.
[34] E. Gomede, R. Miranda de Barros, L. de Souza Mendes, Use of deep multi-target prediction to identify learning styles, Appl. Sci. 10 (5) (2020) 1756.
[35] J.E. Villaverde, D. Godoy, A. Amandi, Learning styles' recognition in e-learning environments with feed-forward neural networks, J. Comput. Assist. Learn. 22 (3) (2006) 197–206.
[36] V. Yannibelli, D. Godoy, A. Amandi, A genetic algorithm approach to recognise students' learning styles, Interact. Learn. Environ. 14 (1) (2006) 55–78.
[37] K. Crockett, A. Latham, N. Whitton, On predicting learning styles in conversational intelligent tutoring systems using fuzzy decision trees, Int. J. Hum.-Comput. Stud. 97 (2017) 98–115.

[38] E. Özpolat, G.B. Akar, Automatic detection of learning styles for an e-learning system, Comput. Educ. 53 (2) (2009) 355–367.

[39] S. Graf, Kinshuk, T.-C. Liu, Supporting teachers in identifying students' learning styles in learning management systems: An automatic student modelling approach, J. Educ. Technol. Soc. 12 (4) (2009) 3–14.

[40] T.M. Mitchell, Artificial neural networks, Mach. Learn. 45 (1997) 81–127.

[41] M. Dorigo, L.M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput. 1 (1) (1997) 53–66.

[42] A. Misevicius, An improved hybrid genetic algorithm: New results for the quadratic assignment problem, in: International Conference on Innovative Techniques and Applications of Artificial Intelligence, Springer, 2003, pp. 3–16.

[43] S. Wermter, R. Sun, An overview of hybrid neural systems, in: International Workshop on Hybrid Neural Systems, Springer, 1998, pp. 1–13.

[44] J.F. Gonçalves, J.J. de Magalhães Mendes, M.G. Resende, A hybrid genetic algorithm for the job shop scheduling problem, European J. Oper. Res. 167 (1) (2005) 77–95.

[45] Y.-T. Kao, E. Zahara, A hybrid genetic algorithm and particle swarm optimization for multimodal functions, Appl. Soft Comput. 8 (2) (2008) 849–857.

[46] R.C. Eberhart, Y. Shi, Evolving artificial neural networks, in: Proceedings of the International Conference on Neural Networks and Brain, 1998, pp. 84–89.

[47] M. Birjali, A. Beni-Hssane, M. Erritali, A novel adaptive e-learning model based on Big Data by using competence-based knowledge and social learner activities, Appl. Soft Comput. 69 (2018) 14–32.

[48] M. Dorigo, T. Stützle, Ant colony optimization: Overview and recent advances, Handb. Metaheuristics (2019) 311–351.

[49] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.

[50] M.H. Aghdam, N. Ghasem-Aghaee, M.E. Basiri, Text feature selection using ant colony optimization, Expert Syst. Appl. 36 (3) (2009) 6843–6853.

[51] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. B 26 (1) (1996) 29–41.

[52] M. Dorigo, L.M. Gambardella, Ant colonies for the travelling salesman problem, Biosystems 43 (2) (1997) 73–81.

[53] S.-J. Huang, Enhancement of hydroelectric generation scheduling using ant colony system based optimization approaches, IEEE Trans. Energy Convers. 16 (3) (2001) 296–301.

[54] K. Swingler, Applying Neural Networks: Aa Practical Guide, Morgan Kaufmann, 1996.

[55] N. Wanas, G. Auda, M.S. Kamel, F. Karray, On the optimal number of hidden nodes in a neural network, in: Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering, 2, IEEE, 1998, pp. 918–921.

[56] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in: Proceedings of the International Joint Conference on Artificial Intelligence, 14, 1995, pp. 1137–1145.

[57] A. Krogh, J.A. Hertz, A simple weight decay can improve generalization, Adv. Neural Inf. Process. Syst. 4 (1992) 950–957.

[58] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Netw. 61 (2015) 85–117.

[59] M. Saleh, R. Salama, S. Bokhary, Augmenting moodle with adaptability environment based on learning styles, Internat. J. Engrg. Sci. 7 (6) (2018) 17–23.

[60] J. Perišić, M. Milovanović, Z. Kazi, A semantic approach to enhance moodle with personalization, Comput. Appl. Eng. Educ. 26 (4) (2018) 884–901.

[61] H. Imran, M. Belghis-Zadeh, T.-W. Chang, Kinshuck, S. Graf, PLORS: A personalized learning object recommender system, Vietnam J. Comput. Sci. 3 (1) (2016) 3–13.

[62] S.M. Nafea, F. Siewe, Y. He, On recommendation of learning objects using Felder-Silverman learning style model, IEEE Access 7 (2019) 163034–163048.

[63] N.S. Raj, V. Renumol, A rule-based approach for adaptive content recommendation in a personalized learning environment: An experimental analysis, in: Proceedings of the IEEE Tenth International Conference on Technology for Education, IEEE, 2019, pp. 138–141.

[64] M. Belghis-Zadeh, H. Imran, M. Chang, S. Graf, WEBLORS–a personalized web-based recommender system, in: International Conference on Web-Based Learning, Springer, 2019, pp. 258–266.