Journal of Applied Research on Industrial Engineering
www.journal-aprie.com

# A Deep-Belief Network Approach for Course Scheduling

***Hassan Rashidi*[1*,1] *, Maryam Hassanpour*[2]**
[1]*Department of Computer Science, Faculty of Statistics, Mathematics and Computer Science, Allameh Tabataba'i University, Tehran, Iran.*
[2]*Department of Computer Engineering, Islamic Qazvin Azad University, Qazvin, Iran.*

**PAPER INFO**

**Chronicle:**
*Received: 09 April 2020*
*Reviewed: 22 May 2020*
*Revised: 08 August 2020*
*Accepted: 23 August 2020*

**Keywords:**
*Scheduling.*
*Course Scheduling.*
*Network Deep-Belief.*
*Learning Ability.*

**ABSTRACT**

The scheduling of academic courses is a problem in which a weekly schedule is produced for educational purposes. Many different types of scheduling problems exist at various universities in accordance with their laws, needs, and constraints. These problems fall into the category of NP-hard problems and are incredibly complex. In this paper, an intelligent system for scheduling courses using the Deep-Belief Network (DBN) is proposed. The reason why the proposed system is intelligent is that it can learn the constraints, inputs, and other necessary parameters in one step by receiving the inputs as well as the training needed by the deep-belief network. The deep-belief network used has one output layer, four hidden layers, and four input layers. The experimental results of this research show that the deep-belief network proposed for the scheduling of academic courses provides a better score, less error, and execution time compared with Sequence-Based Selection Hyper-Heuristic (SSHH) approach.

## 1. Introduction

The problem of scheduling is one of the issues that is considered in various areas such as education, transportation, business, and organizational plans. In each of these areas, the proper schedule can improve the performance of each component in a workgroup and ultimately achieve the desired goal. On the other hand, lacking an appropriate timetable leads to problems and disadvantages such as lack of optimal use of resources, dissatisfaction in individuals, etc., which ultimately reduces productivity.

The key to an appropriate timetable is the degree of satisfaction of the constraints defined for the scheduling problem in each training center, in which it is desirable to achieve the most benefit with the least time and facilities [1]. There are many contradictory and heterogeneous constraints in choosing course units in modeling, combined with variables such as class, day, time period, teacher and course, at the same time [2]. The aim of course scheduling, therefore, is to design a timetable in such an approach so as to convey the complexity of such problems in two simple ways: (a) reducing the number of interactions between the courses with college students or joint teachers and (b) eliminating the synchronization of courses that require a standard class to provide the views of the education system,

* Corresponding author
E-mail address: s.khalili1367@yahoo.com

10.22105/jarie.2020.243184.1185

teachers, and students as much as possible [3]. Increasing the number of students and disciplines on the one hand and the limited capacity and educational facilities on the other, as well as increasing the time constraints defined by teachers to attend classes, make it more difficult to create an appropriate and acceptable timetable, in which all restrictions should not be violated [4]. Therefore, the variety of constraints and flexibility of the educational centers make it possible to achieve more efficient and efficient solutions. But at the same time, the variety of constraints will increase the size of the problem so that solving the problems take more time.

This research is motivated to obtain a suitable solution for university course timetabling subject to the constraints defined for the problem. This suitability is concerned with getting a better score, less error, and execution time. We propose the Deep-Belief Network (DBN) to solve the problem and compare the results with the Sequence-Based Selection Hyper-Heuristic (SSHH) approach. The remaining sections of this paper are structured as follows. In the Second 2, we will review the literature. The proposed approach is presented in Section 3. In Section 4, we evaluate the proposed approach. The conclusions and future work are given in Section 5.

## 2. Literature Review

The problem of course scheduling, due to the consideration of variables corresponding to faculty members, classes, days of the week, and other constraints, creates an integer timing problem. To solve the problem, early approaches include techniques such as Graph Coloring and Constraint Programming (See [7], [8], [9]), which could be used. Solving the problem in a large-scale in any educational center generates several issues. Usually, these issues cannot be solved in a reasonable and acceptable time. Hence, different approaches have been devised to solve these issues. Many of these approaches depend on the specific educational constraints of the implementation environment. The methods that have been proposed in recent decades as innovative new and super-structural methods have also largely contributed to solving these issues. In the following, the most important methods used to solve this problem are discussed.

### *2.1. Heuristics Algorithms*

Heuristics algorithms are looking for relatively good solutions at very low rates in huge problem. There is no specific guarantee for finding the optimal solution or close solutions. These algorithms can find a solution near to optimal in a limited time. They usually provide a higher convergence rate and escape from a poor local solution. The heuristics algorithms are divided into three categories [14]:

- Constructive algorithms. In these algorithms, a solution to the problem is gradually, and step-by-step will be generated, according to the problem data. For example, the nearest neighbor's algorithm for solving a traveling salesman problem is a constructive algorithm in which the first city is randomly selected and then, according to the distance matrix in each replication, the nearest city is selected and the tour added. The most important features of the constructive algorithms are that they are very fast and most of them use greedy ideas, but in some cases, the distance of the generated solution to the optimal solution is too far.
- Improvement algorithms. Improvement algorithms, also referred to as local search algorithms, are another type of innovative algorithm in which searches usually start from an initial solution. This initial solution may be derived from a constructive algorithm or generated randomly. Then, by a local search in the neighbors of this solution, they try to improve the solution. They do this in a recursive manner in each repetition of the algorithm. For example, in hill-climbing algorithms (for maximization problems)

and the gradient decent method (for minimization problems), we use the idea of local search to find better solutions in the neighborhood of the current solution. But the main problem with this type of algorithm is that they are often trapped in an optimal local trap that is much worse than the global optimization.

- Meta-Algorithms. The first transcendental term was introduced by Glawer [11] by introducing the Tabu search algorithm. In fact, the superconductivity algorithms are the general search strategies that can be used as a solution method to a wide range of issues (See [1], [16], [19]). According to Glawer, an algorithm of complexity is the system or framework used to implement it using a number of innovative or precise methods. In other words, meta-algorithms are algorithms that search for the space of algorithms (low-level algorithms) and obtain the best combination for solving the problem. On the other hand, the meta-physical methods take steps that can effectively flee the trap of local optimization. A meta-burgh method has more general objectives designed to achieve in the complete problem-solving space. To use any meta-method for solving a particular problem, the existing rules and method parameters must be designed in such a way that the best possible use of the method in the solution of the problem is obtained.

### *2.2. Sequence-Based Selection Hyper-Heuristic*

Traditionally, a single point search selection hyper-heuristic framework employs two methods invoking them successively. The first method selects a suitable low-level heuristic from a suite of heuristics, applying that chosen heuristic to a candidate solution, thereby generating a new one. The second method, then, makes a decision on whether to accept or reject the newly generated solution [13]. The SSHH method is proposed to solve the problem subject to some constraints and create an initial solution, which consists of two parts: The component of choice and the acceptance component of the movement. In the selection section, a low-level initiative is selected and applied to the current solution and will create a new solution. Then, with respect to and, the acceptance component of the motion decides whether to accept or reject. Usually, five meta-phrasal methods are used to accept the movement. These are local hill-outs, gradual cooking, great deluge, record moves to the record, and late acceptance. For example, if we use hiking as a way of accepting the motion, then it will be accepted if its quality is better. Otherwise, it will be rejected. The sequences of innovative techniques are capable of delivering improved performance over a single, innovative method. But as searches progress, it becomes increasingly difficult to find solutions that achieve better performance than the current solution.

A selection hyper-heuristic is a high-level search methodology that operates on top of the traditional heuristics (or neighborhood move operators) [11]. The combination of these simple operators to build a more complex sequence of operators is the logic behind the development of a SSHH framework.

### *2.3. Neural Networks*

Many years ago, the use of neural networks had been considered as one of the essential tools in the application of artificial intelligence. Object identification, audio and speech analysis, and text review are among the uses in which neural networks are used. Corr et al. [10] examined the application of neural networks as a construction heuristic for the examination timetabling problem. The approach uses a Kohonen self-organizing neural network and is shown to have broad applicability. Building on the heuristic ordering technique, where events are ordered by decreasing scheduling difficulty, the neural network allows a novel dynamic, multi-criteria approach to be developed. The problem of each event to be scheduled is assessed on several characteristics, removing the dependence of order based on a single heuristic. Furthermore, this technique allows the sequence to be reviewed and modified as each event as a necessary step since the timetable and constraints are altered as events are placed. The experimental results are presented for a range of examination timetabling problems using standard benchmark datasets.

Zhang et al. [20] studied a learning style classification approach based on the DBN for large-scale online education. The authors proposed a DBN that is called DBNLS, which is based on the Index of Learning Style theory by Felder and Soloman [31] and the Readiness for education at a distance indicator. The aim is to identify students' learning styles and classify them in two steps. In the first step, a learning style model is built to identify indicators of learning style based on the experiences of experts. In the second step, it relates the indicators to the different learning styles. This research improved the DBN model and identify a student's learning style by analyzing each individual's learning style features using the improved DBN. This research verifies the DBNLS by conducting practical experiments and the various learning styles by soliciting questionnaires from students. Then, this research utilized those data to train the DBNLS model. The experimental results show that the proposed DBNLS method has better accuracy than do the traditional approaches.

Sanchis-Font et al. [21] did a study with aim how to automatically evaluate user experience by sentiment analysis techniques. For this aim, a corpus with the opinions given by a total of 583 users (107 English speakers and 476 Spanish speakers) around three learning management systems in different courses was built. All the collected opinions were manually labeled with polarity information (positive, negative, or neutral) by three human annotators, both at the whole view and sentence levels. Then, the authors applied the state-of-the-art sentiment analysis models, trained with a corpus of a different semantic domain (a Twitter corpus), to study the use of cross-domain models. The cross-domain models based on deep neural networks (convolutional neural networks, transformer encoders and attentional BLSTM models) have been tested. In the experiments, three commercial systems for the same task (Meaning Cloud, Microsoft Text Analytics and Google Cloud) were also tested. The results indicated that the models provide a better and accurate understanding on human needs in the interaction with virtual learning environments. This research is a step towards the development of automatic tools that capture the feed-back of user perception for designing virtual learning environments.

Based on theoretical and biological reasons, it is suggested that such deep-seated architectures include a large number of nonlinear processing layers. But these deep models have a lot of hidden layers and a lot of parameters that need to be trained. This computational complexity and the large space of parameters have led to the use of common methods in neural networks less than large numbers of layers. The problem of the high number of layers in these types of networks, in addition to the low speed of training, leads to the presence of local minima, which in most cases, does not lead us to the desired result. In general, the available approaches to solve the problem can be summarized in *Table 1*. In this table, existing approaches, their main features, their strengths, and weaknesses are reflected.

## 3. The Proposed Approach

The proposed approach to solve the problem is based on deep-belief networks. These networks allow us to create networks with a large number of layers [6]. The deep-belief networks are used not only in categorization but also as a feature extraction method. The advantages of the deep-belief networks in learning are that they can extract high levels of information from training unlabeled data [15] so that the power of differentiation between different categories in data is increased [12]. The studies have shown that loopback networks in their layers can achieve the same characteristics as those extracted in different brain layers.

In this section, we describe the deep-belief networks briefly. Then, the steps in the proposed approach are presented.

***Table 1.*** *A summary of the approaches to solve the course scheduling problem.*

| Approach [Ref.] | Main features | Strengths | Weaknesses |
|---|---|---|---|
| *Graph coloring [7]* | *Getting the optimal solution accurately.* | *Avoid getting stuck in the local optimal.* | *Dependent on initial conditions.* |
| *Tabu search ([3], [16])* | *Preventing any repeat and repetitive operation.* | *Not to be in local optimality.* | *Long execution time.* |
| *Simulated annealing [22]* | *Use temperature factor.* | *Ability to escape from the local maximum.* | *There is no greedy performance.* |
| *Random repeat optimization algorithm with hybrid neighbors [7]* | *Possibility of accepting a worse program with probability.* | *Provides a good solution to the optimum.* | *In practice, due to the high number of possible solutions, it is unlikely to be used.* |
| *Particle swarm optimization algorithm [23]* | *Needs memory.* | *Flexibility against local optimization problem; high convergence rate.* | *Early convergence and cohesiveness in local optimization as well as reduction of population diversity.* |
| *Hill climbing algorithm, the first choice [26]* | *Suitable for problems with a large number of great functions.* | *High speed; low memory consumption.* | *Not perfect and not optimal.* |
| *Genetic algorithm [19]* | *Truly random.* | *No need for a high-level mathematics.* | *Fast convergence and lack of utilization of information distribution.* |
| *Bee's algorithm [24]* | *Local search along with general search.* | *High efficiency in finding an optimal solution.* | *Use the coordinate number of variables; dependent parameters can be defined.* |
| *Memetic algorithm [25]* | *Local search.* | *Able to solve problems with a topical solution.* | *High storage and high simulation time.* |
| *Neural network [10]* | *Inspired by the biological nervous system for processing data and information.* | *Effective for solving complex and large problems.* | *As the environment grows and the number of layers increases, they become faulty.* |
| *SSHH method [13]* | *Selection and use of sequences of innovative methods.* | *Extracting the inherent features and relationships between data and generalizing them in other situations.* | *Long runtime.* |

## *3.1. Deep-Belief Networks*

The deep-belief networks are made up of layers called the Boltzmann Machine Limited. Each this limited machine is a productive and directional probability model that uses a hidden layer to model a distribution on its observed variables. In fact, by confining limited Boltzmann machines, we can create deep-belief systems for hierarchical processes. The restricted Boltzmann machine is a Boltzmann machine that cuts connections between hidden units and between observation units. This machine is a directional graphic model, which is called a Markov Random Field (MRF).

***Fig. 1*** shows the general Boltzmann machine on the left-hand side and a restricted Boltzmann machine on the right-hand side [17]. The joints between hidden units and also between visible units are disconnected. The term $W$ is the concurrent weights between visible and hidden units, $L$ is the concurrent weights between visible and visible units. Finally, $J$ is the concurrent weights between hidden and hidden units. The diagonal values of $L$ and $J$ are zero. Since Boltzmann machines have a complicated theory and formulations, therefore Restricted Boltzmann Machines (RBM) is used for simplicity. If $J = 0$ and $L = 0$, the famous RBM model is introduced.

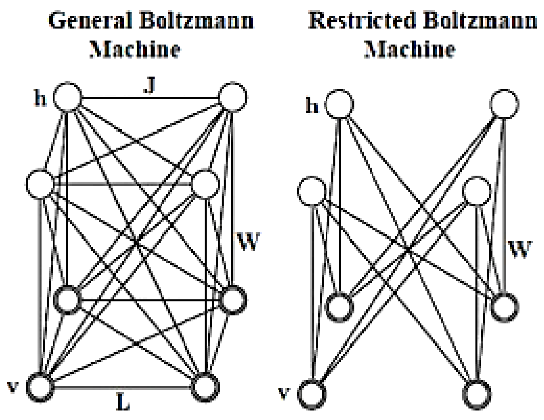


***Fig. 1.*** *A general Boltzmann machine (Left-hand) and a restricted Boltzmann machine (Right-hand) [18]* **.**

## *3.2. The Steps in the Proposed Approach*

The proposed approach is illustrated in *Fig. 2*. As shown in this figure, we must specify the dataset at the first stage. Then, the input parameters are specified as the name of the teacher, the name of the course, the teaching time, the class number, the workshop number, the state of the day (closed or not), and the day number. If there are any changes in the name of the teachers, the name of the courses, and such information, they must be applied to the coding. In other words, any insertion, deletion, or updating of the information is done by changing the input file. For example, if the teacher number 5 is deleted or renamed in the input file that is the Excel file, this change of information is done by the user so that the new data is coded and given to the network. Then the following steps are performed:

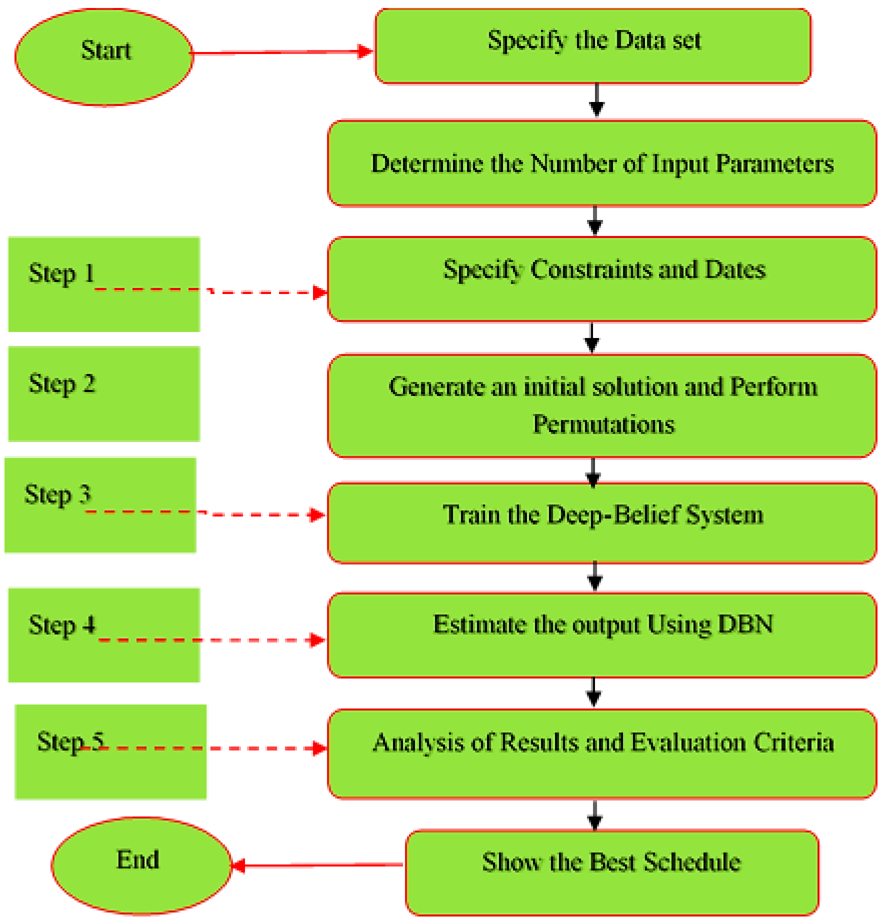

Start
Specify the Data set
Determine the Number of Input Parameters
Step 1
Specify Constraints and Dates
Step 2
Generate an initial solution and Perform Permutations
Step 3
Train the Deep-Belief System
Step 4
Estimate the output Using DBN
Step 5
Analysis of Results and Evaluation Criteria
End
Show the Best Schedule

***Fig. 2.*** *The steps in the proposed approach.*

- Step 1 (Specify Constraints and Dates). There are some constraints on each solution that each teacher has in teaching, the days they cannot attend, and the number of absences, as well as the holidays. Moreover, no two courses in an hour are to be with a teacher in one class. These constraints are classified into hard and soft constraints. The hard constraints must be satisfied to ensure the feasibility of the solution, whereas the soft constraints specify preferences. The violation of the soft constraint does not destroy the feasibility but rather affecting the quality of the solution. In fact, the objective function is set in such a way that the soft constraints are satisfied as much as possible. The quality of a solution is evaluated in terms of the amount of violations of the hard and soft constraints.
- Step 2 (Generate an Initial Solution and perform Permutations). In this step, an initial solution must be generated. This solution is obtained by random generators, at first. Then the initial schedule is refined and improved by permutations. The number of modes in the initial solution equals the number of days of the week multiplied by the number of students multiplied by the number of rows in the input matrix. This states there is a large possible solution space that must be investigated. Therefore, we must consider a limited number of permutation modes.
- Step 3 (Training the Deep-Belief System). The deep-belief network used in this study has one output layer, four input layers, and four hidden layers. The reason for choosing these settings for the network is quite experimental. The input layer receives the data and the output layer displays the output. The hidden layer has two important tasks. The first task is to record the data and the second one is concerned with the learning algorithm. DBNs are composed of multiple layers of RBMs. RBM is a Boltzmann machine where the connections between hidden visible layers are disjointed. Also the Boltzmann

machine is an undirected graphical model (or Markov Random Field). The network training is also modeled on '*pretrainDBN*', which uses the '*pretrainRBM'* method, which itself includes several methods used in this study of Gaussian-Bernoulli Restricted Boltzmann Machines (GBRBM). *Fig. 3* shows a three-layer deep belief network and a three-layer deep Boltzmann machine. The pretraining consists of learning a stack of modified RBM's that are then composed to create a deep Boltzmann machine, a DBN model of three layers. Each RBM model performs a nonlinear transformation on its input vectors and produces an output vector, which is used as input for the next RBM model in the sequence. More details on this step are explained in the *Subsection 4.2*.

–

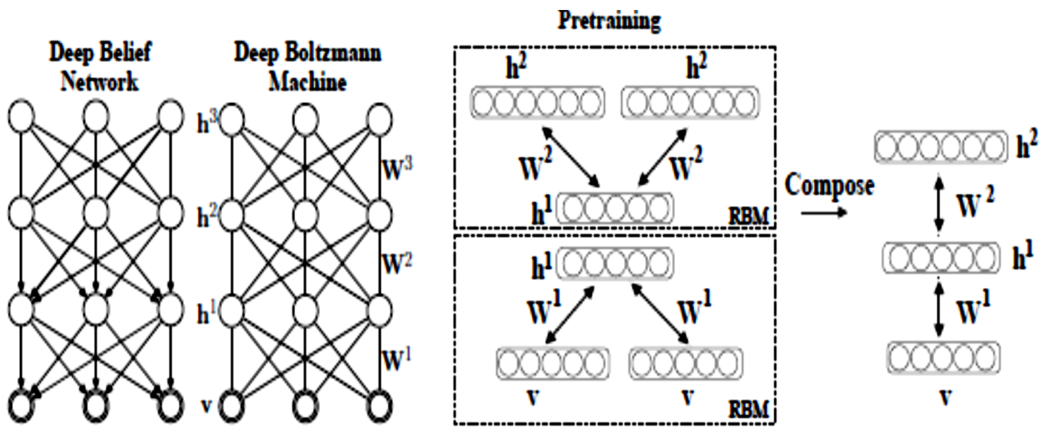


***Fig. 3.*** *A three-layer deep B elief network and a three-layer deep Boltzmann machine (Left side). The pretraining consists of learning a stack of modified. RBM's (Right-Side) [18].*

The process of training the network is such that the data encoded in the input matrix is placed. The input matrix contains the data read at the beginning of the process from the data set. Then the input matrix is given to the network, and the non-interoperability condition is defined using coding in the macro environment for the network.

- Step 4 (Estimate the output Using DBN). In this step, the network is able to process new data according to the previous inputs. We use the trained DBN to produce its output. The network can be based on what was previously learned, make a solution and decision, takes the input, and gives an output. In order to use the network, the string of received codes is given to the network. So, the output is an estimation obtained from the DBN.
- Step 5 (Analysis of Results and Evaluation Criteria). In this step, the estimated output obtained from the DBN and the input is compared. For comparison, the value of the objective function is considered. The criteria for evaluation, here, are the error, execution time, and score for the solution obtained so far. The score is related to the quality of a solution, which is specified with the objective function of the violations of the hard and soft constraints defined for the problem.

We focused our approach of DNN to DBN that is a popular and widely used deep architecture, trained using Hinton et al. algorithm [12]. The main reason for this selection is that DBN architecture significantly reduces the training complexity and makes the deep learning feasible [27]. Moreover, DBN is able to create deep representations at every layer so that the network learns a new and more abstract representation of the input.

The main advantage of the network is its ability to learn that in one step the cases are entered into the network. Additionally, the constraints and conditions are determined by the user. If a teacher is removed from the list, for example, then the inlet of the input matrix is zero and the program runs from the beginning if the course is deleted. If the class number has changed, and if an event occurs, such as atmospheric variations and the time spent on the matrix is deleted, the user is able to modify the input matrix according to the changed cases and retry the network training.

## 4. Evaluate the Proposed Approach

In this section, firstly, we specify the data set and then describe the main program used for the implantation. In the implementation, we must: (a) determine the number of repetitions for not reaching an infinite cycle, (b) read input data from the dataset file and make the input matrix, (c) generate an initial solution by permutation, (d) calculate the cost function in a matrix based on the absence of interference between the input data and violating soft constraints, (e) train the network by the solution found so far, (f) use the network and estimate its output by feeding the inputs, and (f) calculate errors.

### *4.1. The Data Set*

There are several data sets in the field of time scheduling problems. To evaluate the proposed approach, the data set ITC2011 has been used [5]. This data set is suitable for use in scheduling systems and many types of research on the problem. It has several examples from different countries, including Brazil, Finland, Greece, and Netherlands. For each country, *Table 2* is available. A separate table is designed to further explore the educational locations of each country. Each country can have one or more time tables that depend on the number of educational centers that are intended for that country.

As an example, the specification of the data set for 'Brazil-Instance2'[1], is in *Table 2*. The fields in the table are 'Times' (total number of times) and 'Teachers', 'Rooms' and 'Classes' are the total numbers of resources of resource type" Teacher"," Room" and" Class", respectively. The #Events is the total number of events. For the solution, 'Time' represents an indivisible interval of time during which event run. The 'Resource' represents the entity that attend the event.

There are several constraints on every instance of the problem. For the instance 'Brazil-Instance2', the constraints are categorized into three groups: (a) the scheduling constraints that includes *'AssignTimeConstraint'* and *'SplitEventsConstraint'*, (b) the event constraints that comprises *'DistributeSplitEventsConstraint'*, *'PreferTimesConstraint'*, and '*SpreadEventsConstraint*', and (c) the resource constraints that includes *'AvoidClashesConstraint'*, *'LimitIdleTimesConstraint'*, and '*ClusterBusyTimesConstraint'*. The quality of a solution is specified with a values of (hardViolationScore, softViolationScore). For example, a solution with the values of (25, 78) indicates an infeasibility value of 25 (sum of weighted hard constraints violations) and an objective value of 78 (sum of weighted soft constraints violations). The weight value and whether the constraint is hard or soft per each constraint type are presented in the instance.

**Table 2.** *Specification of the dataset 'Brazil-Instance2', used in this research.*

| **Assets** | **Times** | **Teachers** | **Rooms** | **Students** | **Classes** | **#Events** | **Total Duration** |
|---|---|---|---|---|---|---|---|
| **Value** | *25* | *14* | - | - | *6* | *63* | *150* |

[1] https://www.utwente.nl/en/eemcs/dmmp/hstt/

In the proposed method, each item is assigned to an arbitrary integer value. So, the assigned items are then coded as described below, according to the information in *Table 3*.

***Table 3.*** *The encoded sample of the input data.*

| **Variable** | **Name of Courses** | **Teacher's Id.** | **Course-Start Time** | **Class No.** | **Workshop No.** | **Day No.** | **Holiday Status** |
|---|---|---|---|---|---|---|---|
| **How to Code** | *Four digits numbers* | *From 1 to the number of teachers* | *Codes as time interval* | *From 1 to the number of classes* | *From 1 to the number of workshops* | *From 1 to 7* | *Zero/One* |
| **Sample 1** | *2441* | *1* | *1* | *2* | *0* | *3* | *1* |
| **Sample 2** | *1313* | *9* | *2* | *0* | *2* | *3* | *0* |

According to the information in *Table 3*:

- For the name of the course, four digits are used for each course, for example, 2441, 1111, and so on. These assignments are done by the user.
- The Id of the teacher is coded from number one to the number of teachers. For example, if there are fifty teachers in the college, numbers 1 through 50 will be numbered, respectively.
- The class start hour is quantified into intervals. These intervals are 8-10, 10-12, 12-14, 14-16, 16-18, and 18-20. For the first interval, the number one, the second interval of the number two, and so on, are numbered up to six. The courses available for teaching at each faculty can be modified individually by the individual.
- The class No. is numbered from one to the number of classes. For the workshop number, it works in the same way.
- The number of the day is coded from one to seven, which is quoted from Saturday to Friday, and at the end of the campus, the graduation is indicated for ceremonies or special occasions with two numbers of one and zero.

So according to the above, the file containing the generated codes will be created according to the high numbering. No two code strings should be in a shape that does not have overlap times. As an example, the code 244111231 indicates that the course of 2441 is carried out by the teacher one, at 8:00 to 10:00, in class 2, on Monday, with a non-closing condition.

### *4.2. The Program Used for Implementation*

*Fig. 4* illustrates the flowchart of the main program used for the implementation in MATLAB (R2017) environment. The main reason to use this environment is that many source code and Toolbox are available [17]. As the figure shows, the input matrix, 'Numeric', are read from an instance Excel file in which there are many rows. Each its row includes the class, the number of students, the week for a course.

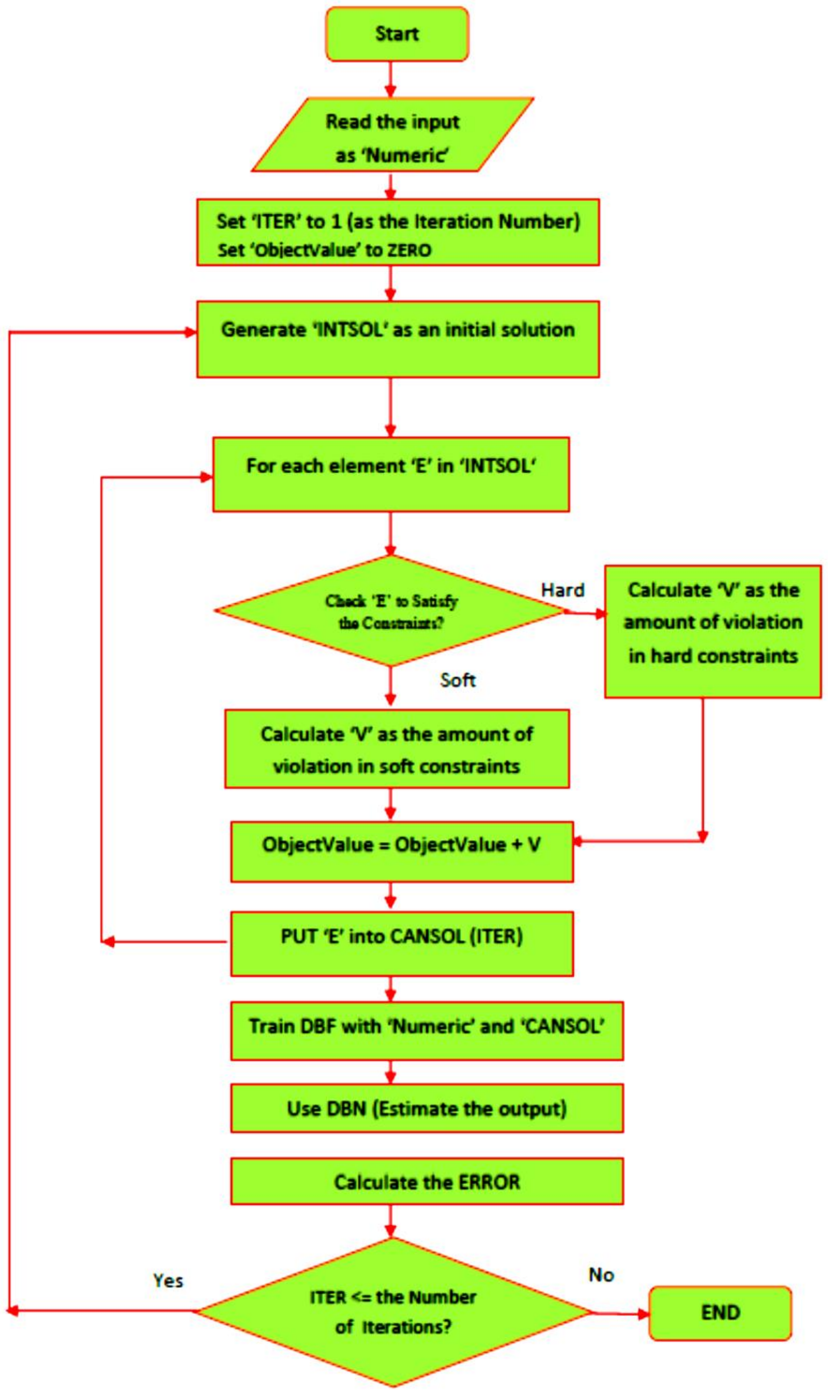

Start
Read the input as 'Numeric'
Set 'ITER' to 1 (as the Iteration Number)
Set 'ObjectValue' to ZERO
Generate 'INTSOL' as an initial solution
For each element 'E' in 'INTSOL'
Check 'E' to Satisfy the Constraints?
Hard
Calculate 'V' as the amount of violation in hard constraints
Soft
Calculate 'V' as the amount of violation in soft constraints
ObjectValue = ObjectValue + V
PUT 'E' into CANSOL (ITER)
Train DBF with 'Numeric' and 'CANSOL'
Use DBN (Estimate the output)
Calculate the ERROR
Yes
ITER <= the Number of Iterations?
No
END

***Fig. 4.*** *Flowchart of the main program used for the implementation.*

The program is run for many iterations. The counter of each iteration is denoted by 'ITER'. In each iteration, an initial solution is generated and put into the matrix 'INTSOL'. The size of this matrix is specified with the number of different modes in the search space. The number of modes of the 'INTSOL equals the number of days of the week multiplied by the number of students multiplied by the number of input matrix states that is 40 modes. So in total, for the number of 30 students, there are 7×40×30 possible states. To improve and refine the initial solution, a number of limited permutations are performed. In our implementation, a maximum of 50 is considered for the initial permutations.

The solutions in *'INTSOL'* must be investigated to be a candidate solution. This investigation is performed subject to violating the soft/hard constraints on the problem. The element *'E'* in *'INTSOL'* must be checked subject to satisfying the hard/soft constraints. Based on the type of the hard/soft constraints, the amounts of violation in the objective function is calculated and put into the variable *'V'*. So the value of the objective function, denoted by *'ObjectValue'*, is increased with the amount of *'V'*. Then, the element *'E'* is put into '*CANSOL(ITER)*'. The same operation is performed for every element of the *'INTSOL'*.

*Fig. 5* shows the train stages used in this study in the form of pseudo code. The deep-belief network used in this study has one output layer, four input layers, and four hidden layers. For training the DBF, we use the matrix *'Numeric'* and *'CANSOL'*. In line 7 of the pseudo code, the training is initially modeled by the '*pretrainRBM*' function, which includes several methods. We used GBRBM in this study. In line 8 of the pseudo code, the function '*SetLinearMapping*' sets the RBM associated to the linear mapping to the last layer. It has three input parameters. The first parameter is the '*dbn*' (the DBN model). The second parameter is the matrix '*inputdata*', in which the number of rows is the number of data, and the number of columns is the number rows of visible (input) nodes. The third parameter is the matrix '*outputdata*' is the teaching data. In this matrix, the number of rows is the number of data, and the number of columns is the number of hidden (output) nodes.

```
1    Set outputnum to 1
2    Set hiddennum to 4
3    Set inputnum  to 4

4    Set inputdata to NUMERIC
5    Set outputdata to CANSOL

6    dbn ← randDBN([inputnum, hiddennum, outputnum]);
7    dbn ← pretrainDBN( dbn, inputdata );
8    dbn ← SetLinearMapping( dbn, inputdata, outputdata );
9    dbn ← trainDBN( dbn, inputdata, outputdata );

10   estimate ← v2h( dbn, inputdata ).
```

***Fig. 5.*** *The pseudo code used for training DBF.*

In the training process, a momentum term was added to update the parameters to promote both the convergence speed and the performance the DBF. The initial momentum is set to 0.5 and after 5 epochs the momentum is set to 0.9. The options of the training process that applied to the program are according to the setting in *Table 4*.

***Table 4.*** *The setting used in running the deep-belief network [29].*

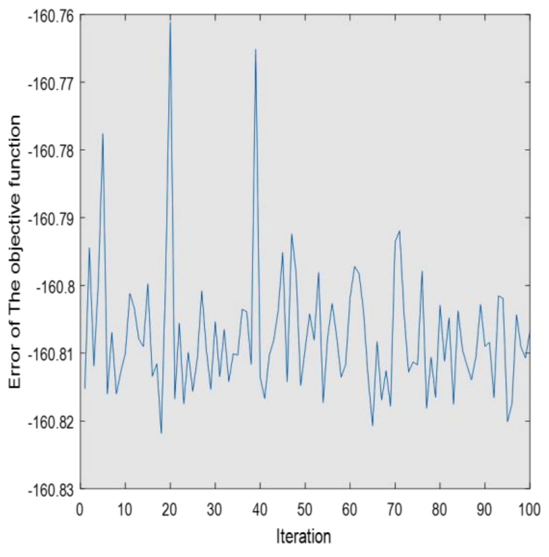
| Options | Setting |
|---|---|
| *Initial momentum.* | *0.5* |
| *Maximum iteration for initial momentum.* | *5* |
| *Final momentum after maximum iteration of initial momentum.* | *0.9* |
| *Maximum iteration number.* | *100* |
| *Number of training RBMs counted from the output layer.* | *all layer* |
| *Weight cost.* | *0.0002* |
| *Dropout rates for each layer.* | *0* |
| *Step size in learning.* | *0.01* |
| *Number of mini-batch data.* | *Number of all data* |

After the training process, we used the network and calculated the objective values based on the new input to the network. In line 10 of the pseudo code in *Fig. 5,* the trained DBF is executed to provide an estimation by the network when the matrix '*inputdata*' is feeding into. There is a function, as h2v [28] to make the transformation from hidden (output) variables to visible (input) variables. The matrix '*inputdata*' is used as its input parameters and the hidden (output) variables, where its number of row is the number of data and its number of column is the number of hidden (output) nodes. The output of the function h2v is the matrix '*estimate*' as the network output, on which the error rate is calculated. In fact, the program error equals the estimates obtained minus the outputs divided by the total outputs available. We run the program for 100 iterations and collect the results. *Table 5* shows an output sample of the proposed approach.

***Table 5.*** *An output sample from the execution of the proposed approach.*

| Teacher's Id | Teaching Hours | Workshop Number | Holiday Status | Week Day | Class Number | Name of Course |
|---|---|---|---|---|---|---|
| *5* | *1* | *0* | *1* | *4* | *30* | *1011* |
| *5* | *1* | *0* | *1* | *3* | *30* | *3021* |
| *4* | *3* | *121* | *1* | *3* | *0* | *4444* |
| *6* | *4* | *0* | *1* | *1* | *10* | *2121* |
| *2* | *1* | *0* | *0* | *4* | *8* | *1313* |
| *9* | *4* | *100* | *0* | *5* | *0* | *8787* |
| *16* | *3* | *0* | *1* | *1* | *15* | *1324* |

As an example, the program is run for one hundred iterations to process a subset of an instance in the dataset. In this dataset, there were 161 instances of the problem. *Fig. 6* shows the sum of errors in the objective function (left-side) and the percentage of the sum of errors (right-side). As the figure (right-side) shows the percentage of error is around 5% in the iteration one and around 1% in the iteration 100. There are fluctuations during running the program. The maximum percentage of the sum of errors is about 4% in the iterations 20 and 40.



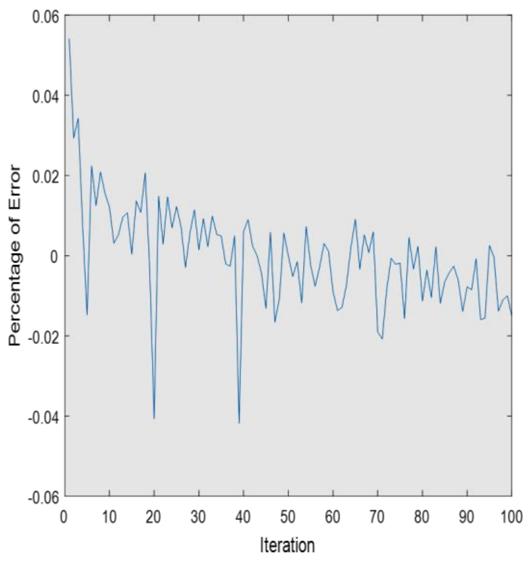


***Fig. 6.*** *The sum of errors of the objective function (left-side), percentage of the sum of errors (right-side).*

The experimentation is conducted on a G4400 CPU at 3.50 GHz with a memory of 16.00 GB. To obtain the execution time, a '*tic*' command is placed at the beginning of the program and a '*toc*' command at the end of each run. This value will vary slightly in different systems with different specifications. The difference in the objective value between the newly generated solution and the best recorded solution after the application of the selected sequence of heuristics is used as a score value. *Table 5* shows the score, execution time, and error rate of SSHH [5] and DBN for each selected instance over ten runs. The score is computed using the scoring scheme utilized in the second round of the ITC 2011 competition for ranking the two approaches. The best values in *Table 6* are highlighted in bold. *Fig. 7* shows the Score, Error Rate, Execution Time, and Average indicators of both approaches, SSH, and DBN, for each selected instance over ten runs.

***Table 6.*** *The score, execution time, and error rate of SSHH along with DBN for each selected instance over ten runs.*

| | **Score** | | **Error Rate (%)** | | **Execution Time (Seconds)** | |
|---|---|---|---|---|---|---|
| **Instance** | **SSHH** | **DBN** | **SSHH** | **DBN** | **SSHH** | **DBN** |
| **Brazil-Instance 2** | *1.45* | *1.30* | *6.10* | *5.01* | *180* | *178* |
| **Finland-Elementary School** | *1.50* | *1.50* | *4.30* | *3.02* | *177* | *153* |
| **Greece-Aigio1st High School 2010** | *1.20* | *1.45* | *5.50* | *4.25* | *166* | *156* |
| **Italy-Instance 4** | *1.40* | *1.30* | *4.35* | *3.02* | *171* | *159* |
| **Kosova-Instance1** | *1.20* | *1.15* | *7.41* | *6.25* | *169* | *177* |
| **Netherlands-Kottenpark 2009** | *1.40* | *1.50* | *6.23* | *7.05* | *189* | *152* |
| **Africa-Woodlands 2009** | *1.60* | *1.40* | *5.51* | *5.06* | *175* | *149* |
| **Spain-School** | *1.50* | *1.35* | *4.32* | *3.88* | *198* | *176* |
| **Average** | *1.41* | *1.37* | *5.47* | *4.69* | *178.13* | *162.50* |

From *Table 6* and *Fig. 7*, we can drive the following corollaries:

– Corollary 1. The score for the proposed approach in 5 out of 8 instances of the problem are better than that of SSHHT. So, the proposed approach did better than SSHHT, on average.

- Corollary 2. The error rate for the proposed approach in 6 out of 8 instances of the problem are less than that of SSHHT. So, the proposed approach performed is better than SSHHT, on average.
- Corollary 3. The execution time of the proposed approach in 7 out of 8 instances of the problem are less than that of SSHHT. So, the proposed approach performed is faster than SSHHT, on average.
- Corollary 4. From *Table 6* (last row) and calculation on the averages, we conclude the percentages of improvements in the proposed approach are about 2%, 8%, and 14% in three indicators (the score, the error rate, and the execution time), respectively.
- Corollary 5. We considered three indicators to compare both approaches, namely the score, the error rate, and the execution time. As *Fig. 7* and *Table 6* (last row) show, the proposed approach performed is better than SSHHT, on average.

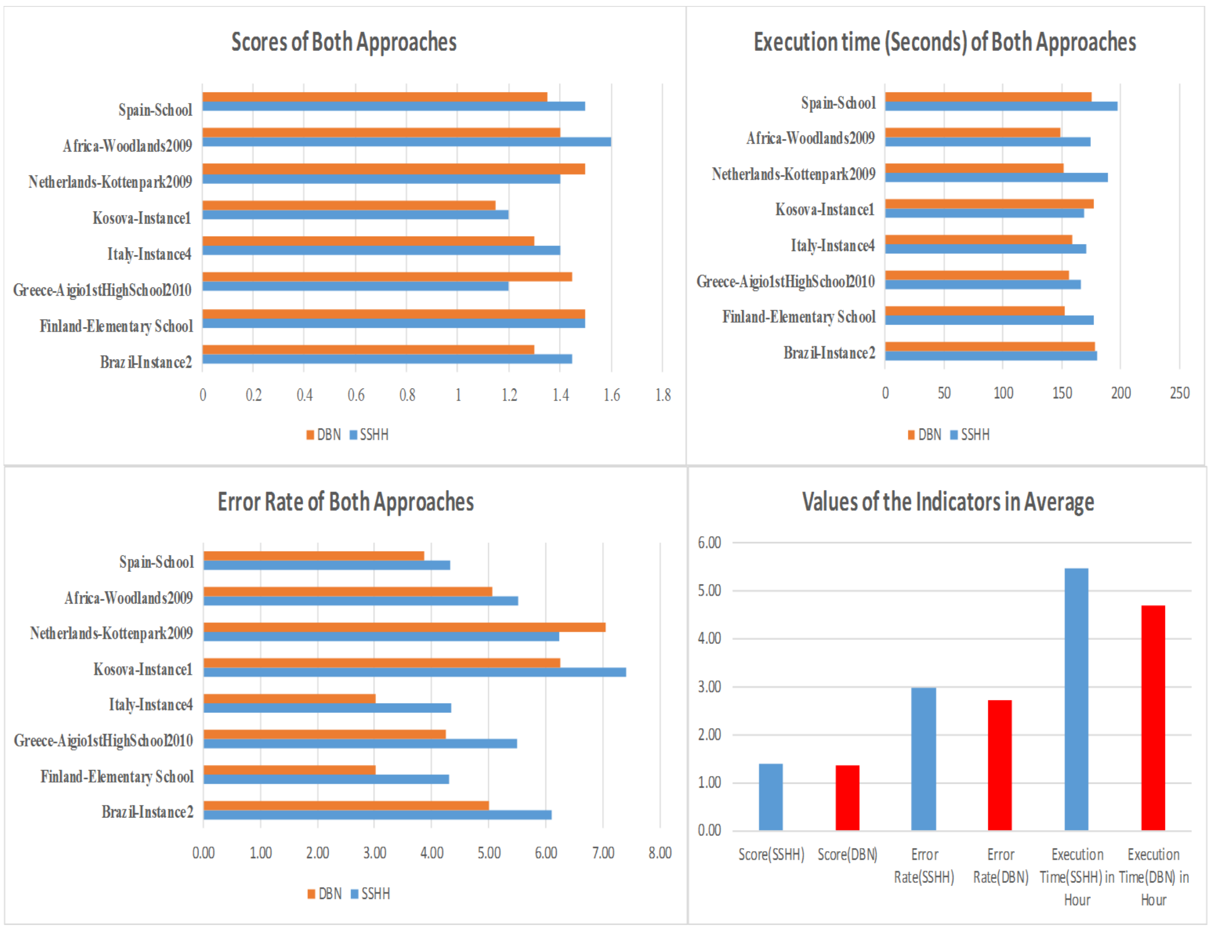


***Fig. 7.*** *The score (left-side up), error rate (left-side down), execution time (right-side up), average (right-side down) of both approaches.*

From our experiments, we can conclude the program provided by the proposed method is highly flexible ahead of the changes and the entry of new data because the network draws the data in a single step in the form of a pattern. Moreover, the amount of human error and computation in the program is reduced because the deep-seated network is simple and obtains a better solution in total in the shortest time.

## 5. Conclusion and Future Work

In this research, the problem of scheduling academic courses has been investigated. The aim is to obtain a suitable timetable according to the constraints defined for the problem. To solve the problem, the data of a specific data set is investigated using the DBN. The reason for using the DBN to solve the scheduling problem is to compete with the other methods mentioned in this field during the runtime and the degree of accuracy of the program to avoid interference. The overall results of this research show that the deep-belief network provides a better score, less error, and execution time for the scheduling of academic courses compared with SSHH approach. In our experience, DBN can be much more efficient in terms of computation and number of parameters for the same level of accuracy.

The work carried out in this study included a set of constraints. The most important future work that can be done is to add other constraints to the problem. One of these constraints is taking a course at another faculty by a student at the college of origin. Another direction to future research is to modify the objective function so that the distribution of students in the classes is maximized. This is necessary to keep health protocols to avoid the Crona-19 virus in the education environments.

## References

[1] Kazarlis, S., Petridis, V., & Fragkou, P. (2005). Solving university timetabling problems using advanced genetic algorithms. *GAs*, *2*(7), 8-12.

[2] McCollum, B. (2006, August). A perspective on bridging the gap between theory and practice in university timetabling. *International conference on the practice and theory of automated timetabling* (pp. 3-23). Springer, Berlin, Heidelberg.

[3] Aladag, C. H., & Hocaoglu, G. (2007). A tabu search algorithm to solve a course timetabling problem. *Hacettepe journal of mathematics and statistics*, *36*(1), 53-64.

[4] Carter, M. W. (2000, August). A comprehensive course timetabling and student scheduling system at the University of Waterloo. *International conference on the practice and theory of automated timetabling* (pp. 64-82). Springer, Berlin, Heidelberg.

[5] Kheiri, A., & Keedwell, E. (2017). A hidden markov model approach to the problem of heuristic selection in hyper-heuristics with a case study in high school timetabling problems. *Evolutionary computation*, *25*(3), 473-501.

[6] Liu, Y., Zhou, S., Chen, Q. (2011). Discriminatory deep faith networks for visual data classification. *Pattern recognition*, *44*(10-11), 2287-2296.

[7] Burke, E. K., & Petrovic, S. (2002). Recent research directions in automated timetabling. *European journal of operational research*, *140*(2), 266-280.

[8] Burke, E., Jackson, K., Kingston, J. H., & Weare, R. (1997). Automated university timetabling: The state of the art. *The computer journal*, *40*(9), 565-571.

[9] Carter, M. W., & Laporte, G. (1995, August). Recent developments in practical examination timetabling. *International conference on the practice and theory of automated timetabling* (pp. 1-21). Springer, Berlin, Heidelberg.

[10] Corr, P. H., McCollum, B., McGreevy, M. A. J., & McMullan, P. (2006). A new neural network based construction heuristic for the examination timetabling problem. In *Parallel problem solving from nature-PPSN IX* (pp. 392-401). Springer, Berlin, Heidelberg.

[11] Glover, F. (1987). Tabu search methods in artificial intelligence and operations research. *ORSA artificial intelligence*, *1*(2). ci.nii.ac.jp/naid/10026173744

[12] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504-507.

[13] Kheiri, A., Özcan, E., Lewis, R., & Thompson, J. (2016). A sequence-based selection hyper-heuristic: A case study in nurse rostering. *Proceedings of the 11th international conference on practice and theory of automated timetabling* (pp. 503-505). Udine, Italy.

[14] Lee, M., Pham, H. & Zhang, X. (1999). Methodology for priority with application to software development process. *European journal of operational research*, *118*(2), 375-389.

[15] Lee, H., Ekanadham, C., & Ng, A. (2007). Sparse deep belief net model for visual area V2. *Advances in neural information processing systems*, *20*, 873-880.

[16] Lai, X., Hao, J. K., Glover, F., & Lü, Z. (2018). A two-phase tabu-evolutionary algorithm for the 0–1 multidimensional knapsack problem. *Information sciences*, *436*, 282-301.

[17] Keyvanrad, M. A., & Homayounpour, M. M. (2014). A brief survey on deep belief networks and introducing a new object oriented toolbox (DeeBNet). https://arxiv.org/abs/1408.3264

[18] Salakhutdinov, R., & Hinton, G. (2009, April). Deep boltzmann machines. *Artificial intelligence and statistics* (pp. 448-455). http://proceedings.mlr.press/v5/salakhutdinov09a/salakhutdinov09a.pdf

[19] Dener, M., & Calp, M. H. (2018). Solving the exam scheduling problems in central exams with genetic algorithms. https://arxiv.org/abs/1902.01360

[20] Zhang, H., Huang, T., Liu, S., Yin, H., Li, J., Yang, H., & Xia, Y. (2020). A learning style classification approach based on deep belief network for large-scale online education. *Journal of cloud computing*, *9*(1), 9-26.

[21] Sanchis-Font, R., Castro-Bleda, M. J., Gonzalez, J. A., Pla, F., & Hurtado, L. F. (2020). Cross-Domain Polarity Models to Evaluate User eXperience in E-learning. *Neural processing letters*. https://doi.org/10.1007/s11063-020-10260-5

[22] AlHadid, I., Kaabneh, K., Tarawneh, H., & Alhroob, A. (2020). Investigation of simulated annealing components to solve the university course timetabling problem. *Italian journal of pure and applied mathematics*, *44*, 282-290. https://www.researchgate.net/profile/Huan_Nan_Shi/publication/343609279_Schur_convexity_of_the_dual_form_of_complete_symmetric_function_involving_exponent_parameter/links/5f33e6d192851cd302ef63fa/Schur-convexity-of-the-dual-form-of-complete-symmetric-function-involving-exponent-parameter.pdf#page=306

[23] Hossain, S. I., Akhand, M. A. H., Shuvo, M. I. R., Siddique, N., & Adeli, H. (2019). Optimization of university course scheduling problem using particle swarm optimization with selective search. *Expert systems with applications*, *127*, 9-24.

[24] Chen, T., & Xu, C. (2015). Solving a timetabling problem with an artificial bee colony algorithm. *World transactions on engineering and technology education*, *13*(3), 438-442. http://www.wiete.com.au/journals/WTE&TE/Pages/Vol.13,%20No.3%20(2015)/41-Chen-T.pdf

[25] Nugroho, M. A., & Hermawan, G. (2018). Solving University course timetabling problem using memetic algorithms and rule-based approaches. *MS&E*, *407*(1), 012012.

[26] Bolaji, A.L., Khader, A.T., Al-Betar, M.A., & Awadallah, M.A. (2014). University course timetabling using hybridized artificial bee colony with hill climbing optimizer. *Journal of computational science*, *5*(5), 809-818.

[27] Rizk Y., Hajj, N., Mitri, N., & Awad, M. (2019). Deep belief networks and cortical algorithms: A comparative study for supervised classification. *Applied computing and informatics*, *15*(2), 81-93.

[28] *MathWorks: Deep Neural Network, Program Code in MATLAB*. Retrieved April 12, 2020, from https://ww2.mathworks.cn/matlabcentral/fileexchange/42853-deep-neural-network

[29] *MathWorks*. Retrieved April 12, 2020, from https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/42853/versions/21/previews/DeepNeuralNetwork/trainDBN.m/index.html

[30] Fukushima, K. (1988). A hierarchical neural network capable of visual pattern recognition. *Neural newt.* *1*(2) 119–130.

[31] Felder, R. M., Soloman, B. A. (1996). *Index of learning styles questionnaire*. Retrieved 18 September, 2020 from http://www.engr.ncsu.edu/learningstyles/ilsweb.html