Proceedings of the 5th WSEAS Int. Conf. on Computational Intelligence, Man-Machine Systems and Cybernetics, Venice, Italy, November 20-22, 2006, pp. 117-122

Effect of Using Neural Networks in GA-Based School Timetabling

JANIS ZUTERS
Department of Computer Science
University of Latvia
Raina bulv. 19, Riga, LV-1050
LATVIA
janis.zuters@lu.lv

Abstract: - The school timetabling problem is a specific kind of timetabling problems. It is characterized by similar sets of subjects used among schools in different years, and by great extent of human factor involved. This particularity lets us to hope existing timetables to be useful information for actual timetabling process, and neural networks to be a suitable technique to assist it. This paper describes experiments on using neural networks as part of the fitness function of a GA-based school timetabling system, the model of what has been proposed by the author earlier. The experimental results show ability of neural networks to be applied for timetable evaluation, as well as reveal various side effects of using neural networks within GA-based school timetabling.

Key-Words: - Neural Networks, School Timetabling, Genetic Algorithm, Fitness Function

1 Introduction

This paper is to finalize a series of experiments on neural networks ([5], [6], [7]), and the goal of the paper is to show possible effect of using neural networks within GA-based optimization, specifically school timetabling.

The main idea of using neural networks as part of timetabling system comes from existing school timetables recognized to be useful information to build new ones [7]. Neural networks trained on existing solutions (i.e., timetables) can be helpful with evaluating the actual ones within GA-based optimization. GAs themselves are often applied to solve optimization problems such as the timetabling problem [1], [2].

The possible application of neural networks could target problems for which evaluation of solutions is a complicated task. To build evaluation rules, strict criteria are required, and these could be hard to extract or even recognize, so neural networks could benefit with reducing effort on defining such rules.

Although the proposed approach is very problem-specific one and is not to be regarded a universal method, the experimental results show the described technique of using neural networks to be able to improve upon the optimization process.

2 GA-Based School Timetabling

The timetabling problem is highly constrained combinatorial problem, and GAs are typical means to solve such problems. GAs exploit principles of evolutionary biology, including such biology-originated techniques as inheritance, mutation, natural selection, and recombination.

A population is a main object dealt by GA. It consists of individuals, which are being improved during an evolutionary process. When solving optimization problems by GA, the single solutions are regarded as individuals. The operation of a GA is a cyclic process, which resembles alternation of generations in biological systems.

An important requirement for the problem domain using GAs, is a possibility to evaluate (rate) solutions at any phase of the evolutionary process. The rating is done by the fitness function. [4]

Various approaches of using GAs in timetabling differ mostly in terms of heuristics applied, e.g., using problem-specific genetic operators, as well as selection and reproduction techniques [1], [2], [3].

In this paper GA-based school timetabling serves as a framework to show the possible effect of using neural networks within the fitness function of a genetic algorithm to solve some optimization problem.

3 Neural Network as Part of a GA-Based School Timetabling Model

3.1 The Proposed Model in General

[7] proposes a school timetabling model (Fig. 1), where fitness function has been reinforced by neural networks trained on existing timetables using a specific technique with randomly generated training patterns [6]. Unlike most approaches of GA-based timetabling, the proposed technique is particularly

focused on the fitness function, i.e. on evaluation of timetables. Design of genetic operators and other algorithms altering the population has been left out of the scope of this paper.

The possible benefits of the proposed model where a neural network is used as part of the fitness function can be viewed from two aspects [7]:

1. Replace one or more soft constraints by a neural network, thus facilitating the effort of defining the constraints;
2. Assuming the complete set of constraints, it is hard and practically impossible to define – complement existing constraints with a neural network in order to improve upon timetable evaluation.

The effect of the 2nd aspect is very hard to verify, as it would need a massive estimation of experts. Therefore, in this paper the proposed model will be examined just according to the 1st aspect.

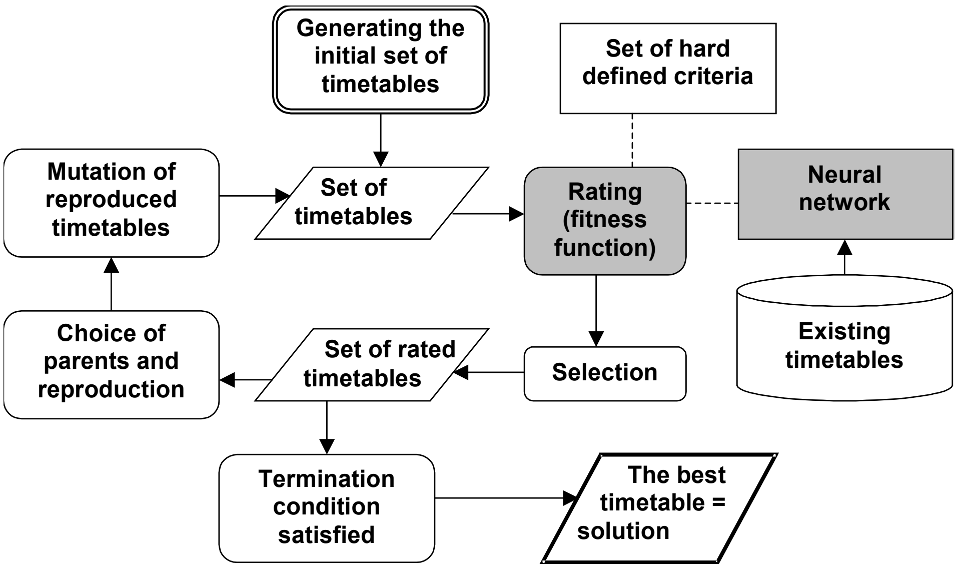
Generating the initial set of timetables
Set of hard defined criteria
Mutation of reproduced timetables
Set of timetables
Rating (fitness function)
Neural network
Existing timetables
Choice of parents and reproduction
Set of rated timetables
Selection
Termination condition satisfied
The best timetable = solution

Fig. 1. GA-based school timetabling model reinforced by neural networks

In the proposed model, outcomes from the functions realizing the hard-defined evaluation functions, and a neural network are combined to obtain the final evaluation of a timetable.

3.2 Neural Networks as Evaluators of Timetables

Neural networks were designed to evaluate timetables or, more precisely, substitute for hard-defined evaluation functions to some extent.

```
PROCEDURE TrainNetwork(NET:NeuralNetwork,
          TRAINING_SET:ReadyClassTimeTables,RANDOM_RATE:0..1)
epoch_size:=Round(SizeOf(TRAINING_SET)/(1-RANDOM_RATE))
REPEAT
    FOR i:=1 TO epoch_size
        rnd:=GetRandomValueBetween(0,1)
        IF rnd<RANDOM_RATE THEN
            pattern:=GenerateRandomPattern()
            desired_value:=1
        ELSE
            pattern:=ChoosePatternFrom(TRAINING_SET)
            desired_value:=0
        END IF
        TrainOnSinglePatternSupervised(NET,pattern,desired_value)
    END FOR
UNTIL FinalConditionMet(NET)
```

Fig. 2. A supervised learning algorithm with additional randomly generated training patterns [6]

Comments on the algorithm depicted at Fig. 2.

- RANDOM_RATE (τ) denotes the proportion of using randomly generated patterns in the learning process.
- An average epoch consists of training on each existing pattern once, and training on randomly generated patterns. The bigger is RANDOM_RATE, the bigger is an epoch.
- For a random pattern the desired value is 1, but for an existing (i.e., good) one – 0. That's because the fitness function operates as penalty function.

The only source of information is ready-made timetables of various schools from recent years. A set of existing and valid timetables is incomplete to be the training set for evaluative neural networks, because it contains just positive patterns. These preconditions require modifications in the training algorithm of neural networks. The main idea of the proposed algorithm (Fig. 2) is to expand the training set on randomly generated patterns to partially compensate for the lack of the complete training set.

3.3 Assessment of the Model

The benefits of using neural networks within GA-based timetabling system were computed with respect to ability to substitute for some hard-defined evaluation functions within the **total fitness function** $\varphi(\cdot)$ [7]:

$$\varphi(\mathrm{T}) = \sum_{i=1}^{c} \alpha_i \varphi_i(\mathrm{T}), \qquad (1)$$

where $\varphi_i(\cdot)$ – the evaluation function according to the constraint i; α_i – the weight of the constraint i; T – the timetable to evaluate.

The total fitness function would be used as a benchmark to examine the model.

```
PROCEDURE RunExperiments(FILE:FileName, TRAINING_SET:ReadyClassTimeTables,
            RAW_SET:UnscheduledTimetables,EVAL_FUNCT_COUNT:Integer)
NET:NeuralNetwork
TT,TTR,TTS:TimeTable
EvalReduced,EvalSoftened:Real
GAConfig:GAConfiguration
NNConfig:NeuralNetworkConfiguration
RANDOM_RATE:0..1
REPEAT
    NNConfig:=GenerateNNConfiguration()
    RANDOM_RATE:=ChooseValueFrom({0,0.2})
    NET:=CreateNeuralNetwork(NNConfig)
    TrainNetwork(NET,TRAINING_SET,RANDOM_RATE)
    GAConfig:=GenerateGAConfiguration()
    TT:=ChoosePatternFrom(RAW_SET)
    FOR func:=1 TO EVAL_FUNCT_COUNT
        TTR:=RunGAWithReducedFitnessFunction(TT,GAConfig,func)
        TTS:=RunGAWithSoftenedFitnessFunction(TT,GAConfig,func,NET)
        EvalReduced:=EvaluateWithTotalFitnessFunction(TTR)
        EvalSoftened:=EvaluateWithTotalFitnessFunction(TTS)
        Register(FILE,NNConfig,GAConfig,EvalReduced,EvalSoftened)
    END FOR
UNTIL BrokenByUser()
```

Fig. 3. The algorithm of the experimentation to gather data to examine the proposed model

Comments on the algorithm depicted at Fig. 3.

- *RunGAWithReducedFitnessFunction(·,·,·)* – runs a GA using the reduced fitness function $\varphi^f(\cdot)$ as the fitness function (3), returns ready timetable.

- *RunGAWithSoftenedFitnessFunction(·,·,·)* – runs a GA using the softened fitness function $\varphi^{f+}(\cdot)$ as the fitness function (3) , returns ready timetable.
- *Register(·,·,·,·,·)* – registers the results for further analysis.
- *TRAINING_SET* – the set of ready-made timetables used as the training set for the neural network.
- *RAW_SET* – the set of unscheduled timetables of various schools, i.e. sets of lessons to be scheduled.

The two following notions, to be used to assess the model, are the reduced fitness function (2) and the softened fitness function (3).

The **reduced fitness function** $\varphi^f(\cdot)$ refers to the total fitness function reduced to have the f^th^ evaluation function eliminated.

$$\varphi^f(\mathrm{T}) = \sum_{i=1}^{c} \begin{Bmatrix} 0; & if\, i = f \\ \alpha_i \varphi_i(\mathrm{T}); & otherwise \end{Bmatrix}, \quad (2)$$

where f – the index of the evaluation function to eliminate.

The **softened fitness function** $\varphi^{f+}(\cdot)$ refers to the according reduced fitness function complemented with a neural network.

$$\varphi^{f+}(\mathrm{T}) = \varphi^f(\mathrm{T}) + \alpha_0 \varphi_0(\mathrm{T}), \quad (3)$$

where $\varphi_0(\cdot)$ – the evaluation function realized by a neural network; α_0 – the weight of the neural evaluating function.

The proposed model was tested using the algorithm described in Fig. 3. The goal of the experimentation is to show results of the softened function to be better than the results of the reduced functions for some evaluation function f.

4 Description of the Experimentation

The goal of the experimentation was to examine the effect of neural networks within the GA-based timetabling system, and to show ability of neural networks to partly substitute for hard-defined evaluation functions. Experiments were run on 8 different schools according the algorithm described in Fig. 3, and then the results were analyzed. The total amount of experiments run and analyzed was 1850. In order to do the research, as well as to run the final experiments; an original software system was designed by the author.

4.1 Representation of Timetables for a Neural Network

4.1.1 Main principles

As the main goal of the experimentation is to show the effect of using neural networks, then the representation principles of timetables for neural networks are of a great significance. The following two factors were considered to establish the representation of timetables.

- Subjects are the only available information of existing timetables of various schools suitable to support school timetabling process. Classes, teachers and rooms could not be used because of differences among schools and even school years. Thus a timetable can be viewed as a sequence of subjects.
- Subjects can be grouped into categories (e.g., exact or humanitarian). Grouping of subjects can reduce computing resources to process timetable information.

Therefore a timetable of one class k can be defined as a sequence of subject categories (i.e., a set of subject categories arranged over time) [7]:

$$\mathrm{T}_k = \langle \sigma(s_i) \rangle_{i=1}^{p}, \quad (4)$$

where s_i – the subject for class k in period i (can be null); p – the number of available periods in one week; σ(sigma) – function that defines grouping of subjects into categories, described in (5).

$$\sigma : Subjects \rightarrow SubjectCategories \quad (5)$$

(E.g., mathematics belongs to the exact subjects, and history to the humanities.)

The author has defined the function σ, one that groups subjects into 12 different categories.

Former experiments have shown the use of subject categories (instead of subjects) to be representative enough to code timetables.

4.1.2 Available training set

10 different ready-made school timetables were available consisting of 208 different class-timetables. Each class-timetable was represented like in (4). Each period of a timetable (i.e., one lesson) was represented as a tuple of 12 values corresponding to 12 subject categories (value 1 – if the subject category is represented in the period, 0 – if not). Thus a class-timetable for one week was a tuple of 600 values (5 days × 10 periods per day × subject categories).

4.2 Configuration of the Experimental Environment

4.2.1 Configuration of a neural network

For each experiment, a new neural network was created and trained. Neural networks were MLPs (600 inputs, one hidden layer with 5 neurons, and one neuron in the output layer). Neural networks were trained according the algorithm described in Fig. 2. During one iteration, a single class-timetable was being exposed to the network.

4.2.2 Configuration of the GA system

The size of population was 10. 3 individuals (timetables) were replaced in each cycle. Elitism concerned 1 individual. Mutation was the only genetic operator. Mutation was applied on reproduced individuals, thus mutation itself didn't replace the parents. Mutation rate: 20%. 4 functions were defined to evaluate timetables and to be replaced by a neural network (Section 4.2.3). Initially lessons of a timetable were arranged randomly over time. The fitness function was a penalty function operated according equations (2) or (3). In the case of neural networks (3), the weight coefficient (α_0) was set to 0.01, thus the impact of the neural network to the overall performance was established to be rather low. To evaluate a timetable, the evaluation function was applied to the each class-timetable separately, and then the average value taken for the total evaluation.

4.2.3 Hard-defined evaluation functions

To examine the possible effect of neural networks within a GA-based optimization, a fitness function consisting of 4 four different evaluation functions were realized (according the constraints listed in Table 1). All the criteria were related to arrangement of lessons for students, and all the evaluation functions were exploited as ones to be replaced by a neural network ((2) and (3)).

Table 1. Soft constraints realized through appropriate evaluation functions for the experiments.

#	Description of the constraint
1	Arrange lessons uniformly over the week
2	Minimize the gaps between lessons for students
3	As far as possible, align lessons to the beginning of the shift
4	Balance the layout of lessons in terms of themes (e.g., don't schedule to many exact subjects in row)

5 Experimental Results and Conclusion

5.1 Obtained results

The outcome of the experiments was the evaluation values for timetables, ones created by a GA using reduced or softened fitness functions (See Section 3.3). The subject of analysis of obtained data is to find out whether timetables generated, using the softened fitness function as the fitness function, are better then ones generated with assistance of reduced fitness function, i.e., whether some of hard-defined evaluation functions could be replaced by a neural network.

The experimental results showed that only replacement of the function #3 by a neural network brought slightly better results (Fig. 4).

The results for functions #1 and #2 were even worse, but for the function #4 – neutral, i.e., neither improvement, nor worsening.

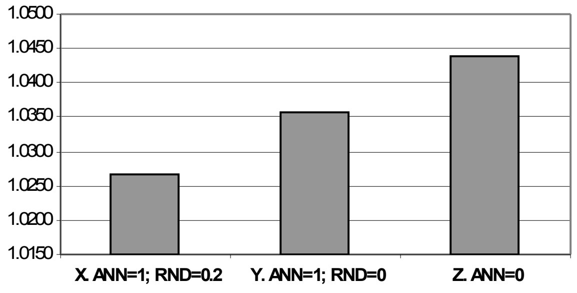

Fig. 4. The average results for the replacement of the evaluation function #3 by neural networks

Fig. 4 shows a part of experimental results in a form of average evaluation values. These results represent the effect of neural networks in replacing the evaluation function #3 ("Align lessons to the beginning of the shift") in the above-described school timetabling system. As the fitness function operates as penalty function, smaller values mean better evaluation.

The results, shown in Fig. 4, reveal 3 groups:

- Group X (1.0265). Neural networks were used within the fitness function (3), and the random rate (τ) was set to 0.2, i.e., randomly generated patterns had been used.
- Group Y (1.0356). Neural networks were used within the fitness function (3), but the random rate (τ) was set to 0, i.e., neural networks were trained only on positive patterns.
- Group Z (1.0439). No neural networks were used within the fitness function (2).

5.2 Conclusions for the Current Experimentation

The obtained results infer the following conclusions:

- As the groups X and Y (Fig. 4) show better results than the group Z, it can be stated that neural networks are able to substitute, in part, for the evaluation function #3.
- As the group X shows better results then the group Y, it can be stated that using randomly generated training patterns to train neural networks (described in Section 3.2) is suitable in this case.

The little impact of the neural networks to the results can be explained by the small weight value used for a neural network (α_0=0.01). Bigger weight values for the neural network got the fitness function dramatically degraded.

5.3 Final Conclusions

The experimental results demonstrate neural networks to be able to substitute for hard-defined evaluation functions in certain cases, as well as approve the use of the supervised training method for neural networks, proposed by the author earlier.

Using neural networks to successfully solve similar problems could require a massive problem-specific investigation. The experimentations, conducted by the author, have shown the progress of adaptation of neural networks for such specific problems to demand big efforts.

Although this paper doesn't provide with know-how of school timetabling, nevertheless it shows neural networks to be applied in solving such an uncommon problem like timetable evaluation.

References:

[1] N. Arous et. al., Evolutionary Potential Timetables Optimization by Means of Genetic and Greedy Algorithms, *Proceedings of the 1999 International Conference on Information Intelligence and Systems (ICIIS)*, 1999

[2] C. Fernandes et. al., High School Weekly Timetabling by Evolutionary Algorithms, *Proceedings of the 1999 ACM symposium on Applied computing*, 1999, pp. 344-350

[3] V. Tam, D. Ting, Combining the Min-Conflicts and Look-Forward Heuristics to Effectively Solve A Set of Hard University Timetabling Problems, *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'03)*, 2003

[4] J. Zuters, An Adaptable Computational Model for Scheduling Training Sessions, *Annual Proceedings of Vidzeme University College "ICTE in Regional Development"*, 2005, pp. 110-113.

[5] J. Zuters, An Extension of Multi-Layer Perceptron Based on Layer-Topology, *Proceedings of the 5th International Enformatika Conference'05*, 2005, pp. 178-181.

[6] J. Zuters, Towards Multi-Layer Perceptron as an Evaluator Through Randomly Generated Training Patterns, *Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases (AIKED 2006)*, 2006, pp. 254-258.

[7] J. Zuters, An Ensemble of Neural Networks as Part of a GA-based Model to Solve the School Timetabling Problem, *Local proceedings of the 7th International Baltic Conference on Databases and Information Systems (Baltic DB&IS 2006)*, 2006, pp. 175-182