Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 169 (2020) 226–229

Procedia Computer Science

www.elsevier.com/locate/procedia

Postproceedings of the 10th Annual International Conference on Brain-Inspired Cognitive Architectures for Artificial Intelligence, BICA*AI 2019 (Tenth Annual Meeting of BICA Society)

Adaptive computer training system GraphLabs

Maria Korotkova[a], George Carpow[a]*, Svyatoslav Zakhryapin[a]

[a]*NRNU MEPhI, Kashirskoye highway, 31, Moscow 117525, Russia*

Abstract

A modular computer system provides laboratory work on graph theory. Creating an option for a student takes into account the student model. The student model is build and modified during laboratory work. An automaton model is used to build an individual work option for a student.

Keywords: GraphLabs, automaton, modular, extendable system, adaptation, training

1. Introduction

The software package GraphLabs [1] is being developed at the Department of Cybernetics in NRNU MEPhI. It is able to create applications for laboratory support for courses of various purposes. The support of the course on graph theory is the original scope of the complex.

GraphLabs uses a client-server architecture model for building applications. The client-side application uses web technology, making it cross-platform and able to be run in any user's browser. Both the client and the server parts use a modular architecture [2] to make the software more flexible, versatile and adaptable to changing conditions of using.

* Corresponding author. Tel.: +7-965-149-37-26.
E-mail address: eakarpov@yandex.ru

10.1016/j.procs.2020.02.139

2. Adaptive approach

When the course starts, the degree of student preparation and their skills are different. Thus, they require an individual approach in training. The efficiency of the education will be diverse for different students with the same learning curve. An automatic adaptation of the course to the characteristics of the student increases the effectiveness of its study.

The adaptability of the system can be increased in various ways, for example, using a group of properties that characterize the student at a certain point in time. The using student characteristics to create a personalized version of the assignment is one way to adapt the system.

Student actions in the software modules are the source of knowledge about him. If you study the state of the system after the student's possible actions in the module and identify the equivalent state of the system, you can get a set of states and possible transitions between them.

Meanwhile the software module has a single initial state S_0 and 1 or more terminal ones $F_1..F_n$. With this system of constrains we can describe a model of system adaptation as a Mealy machine [3] where $A = < S,\ I,\ O, T,\ Fee, S_0 >$, where S and T are states and transition functions correlatively, I – a set of values available for a transition weigh, O – a set of values available for a fee weigh, Fee – a function to juxtapose the weigh and the transition.

A set of Input and Output values are defines as follows:

$$I = Rat^{+} \cup \{0\}$$
$$O = Rat^{+} \cup \{0\}$$

where Rat+ is a set of positive rational numbers. A general view of such a network you can see on the following figure (Fig. 1)

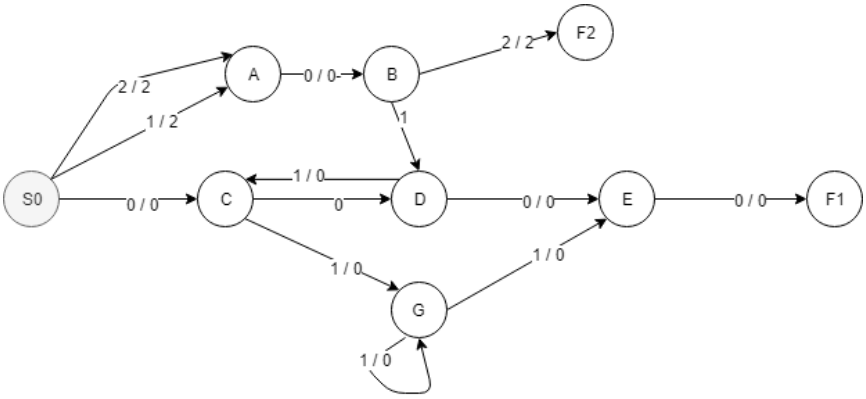

Fig. 1. A general automaton model of the software module

Each of the transitions is weighed with some value. A zero weigh corresponds with the expected student behavior while a positive one corresponds with any deviations in it from the optimal one. Reflexive edges describe user actions while keeping the module state. The weigh is summarized while shifting the system from the initial state to the terminal one. The total weight shows how far the student deviated from the optimal path when completing the assignment. The larger the value, the more unnecessary or erroneous actions the student has taken.

The design of the machine suggests that a notification of a penalty to the server should be made at each transition. Given the implementation, we can assume that only positive commission values should be sent to the server. Zero weight only leads to internal weight correction within the software module.

3. Views on implementation

A final notification is submitted to the server when the transition to the terminal state occurs. It contains a total deviation value that was gathered by the user executing the laboratory work. The eccentricity of the initial vertex is also sent in this notification. It is used to estimate a maximum acceptable user deviation. The following criterion is used to characterize the user:

$$Value^1 = \frac{Value + \frac{D}{E}}{N+1} * (1 - v) \quad (1)$$

E – eccentricity [4] of the current module (considering no reflexive edges), D – deviation [5] of the user, N – a number of executed laboratory tasks by the user, v – velocity of executing the current program module.

The value of the criterion is stored in the user model along with the number of modules the user has executed. In fact, the criterion is a normalized average value of deviation. The normal range of the criterion is inside [-1;1], although if a large number of reflexive transitions are used, a value greater than 1 is obtained.

The velocity of module execution is a correction factor indicating user confidence while executing the task. Basically, it can have a zero value. In advance, it can be of the range [0; 1] as the following function value $v = F(L_{min}, l, T)$, where L_{min}- the shortest path in the module, l – a number of transitions made by the user, T – an array of tuples of transitions and time lapses between the transitions in the previous state and into the next one.

The initial value of the user's characteristic is 1. Whenever the laboratory task is completed, the characteristic is recalculated based on the speed of the current execution and the previous value of the characteristic.

When choosing a task variant, the server receives the current value of the user's characteristics and the task complexity correlates with it. A correlation function may take the form:

$$F = Max(N, Round\left(\sqrt[N]{\frac{1}{Value^2}}\right) + 1) \quad (2)$$

where N – a number of difficult levels in the database.

Scenarios of the module can also be automatically configured based on the model of user actions. The set of states can be divided into subsets characterizing the logical subprocesses of the module (Fig. 2). A subset can be a subgraph with no cycles (A) or can contain cycles (B). Subsets are not required to cover the entire graph. Moreover, they can intersect with each other.

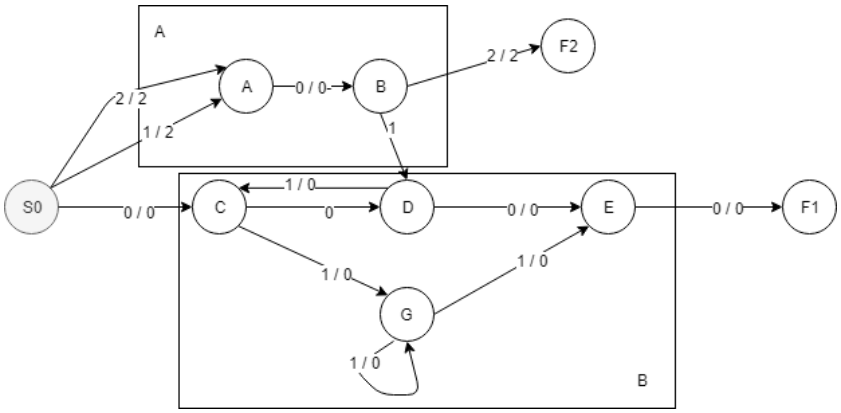

Fig. 2. A logical state grouping.

Nevertheless, every subset contains an internal counter that increases with every non-zero transition inside the subset. Reaching a threshold value can lead to an additional scenario initiation for user-module communication. For example, it may be showing tooltips or instructions to repeat the execution of some actions to enhance the user's skills.

Counters' values are stored inside the module and are not submitted to the server. The use of such mechanisms in software modules can increase the effectiveness of training due to the autonomous use of algorithms for better assimilation and formation of skills.

4. Considering details

It should be noted that the system should track the value of the characteristics of the student during his work in the software package. An efficiency criterion could be introduced as a function with the following parameters: $EF(Student) = \sum_t (Value_t - Value_{t-1}) < 0$. The function value depends on the characteristic value while student is working.

Thus, the system is considered to be efficient if only it keeps the positive EF criterion value during the interaction with the user. That means that the criterion monotonously decreases every time the new recalculation takes place. Otherwise, either any measures should be taken towards the student or the criterion should be reinitialized.

An adaptive approach operates efficiently when training laboratory tasks are used to gain the following results:

- to allow the system to build a user model and adjust the characteristic value
- to allow the student to get acquainted with the software complex user interface before the control task

Thereby to achieve maximum effect, it is advisable to enable users of distance training before the control event. This is able to achieve using GraphLabs laboratory complex.

5. Conclusion

Adaptation of the system behavior during interaction with the user is extremely important for the effective functioning of the training software package. A model based on individual student behavior is proposed in this article. This allows you to dynamically change the complexity of the functioning of the software module when interacting with the student.

The use of the adaptive approach makes it possible to increase the efficiency of educational impact on the student during the training course depending on his initial learning skills. This is achieved by adjusting the student's actions when working with the complex and changing the behavior of software modules when the user performs certain actions.

References

[1] *Карпов Е. А.* (2019) "Модернизация архитектуры и разработка инструментальных средств для создания и использования лабораторных работ комплекса GraphLabs": дис … магистра: 09.04.04 – НИЯУ МИФИ, Москва.

[2] *Len Bass, Paul Clements and Rick Kazman*. (2015) "Software Architecture in Practice." Pearson, Massachusetts.

[3] *Mealy, George H.* (1955) "A Method for Synthesizing Sequential Circuits". Bell System Technical Journal, vol. 5, pp. 1045–1079. AT&T, Dallas

[4] *Berge C.* (2001) "The Theory of Graphs". Courier Corporation, USA.

[5] *Harary F.* (1969) "Graph Theory". Addison-Wesley Publishing Company, USA.