International Conference
FROM DELC TO VELSPACE
Plovdiv, 26–28 March 2014

OPTIMIZING THE GLOBALIZATIONS IN INFOSTATION ARCHITECTURE

Ivan Dimitrov, Vladimir Valkanov, Asya Stoyanova-Doycheva

Abstract. *This paper considers the problem of finding an optimal deployment of information resources on an InfoStation network in order to minimize the over-head and reduce the time needed to satisfy user requests for resources. Two program realizations for solving the problem of optimal deployment of resources are presented.*

Key words: distributed eLearning Centre (DeLC), InfoStation network, request globalization, over-head optimization.

INTRODUCTION

During the last decade in the Faculty of Mathematics and Informatics at the Plovdiv University "Paisii Hilendarski" an educational portal is being developed which is called Distibuted eLearning Center (DeLC) [3]. The communication infrastructure of DeLC is based on a local area network (LAN) reinforced with information stations (InfoStations), working as serving points with intelligent wireless access. In the standardized version the information stations operate as mediators between mobile user devices and a server on which all system applications and information resources are situated. During the build-up of DeLC, this classic architecture is developed by location services on each InfoStation which can be locally activated. By this decentralized placement of services and information resources a balanced overhead of the communication network and better productivity are aimed [5, 6].

However the decentralized placement of information resources has its setbacks as far as system overload in accordance to the internal relations among information resources during the execution of a query for a service. This overload is especially heavy when the related resources are situated on different information stations. Respectively, the task for reducing the overload is of vital importance in the decentralized approach and the degree of overload depends on the way in which the information resources are distributed [2]. The goal of this current paper is to present the realization of a model (model for Optimizing Resource Locations – OReL) [4] for optimizing information resource locations on DeLC's educational nodes. After the conducted research a conclusion was reached that for the achievement of the goal a heuristic approach would be suitable, specifically – the evolutionary strategy.

The program realization of OReL was performed as an iterative process which led to the creation of two software products:

- OReL^{O} – an object-oriented version which was realized in the Java programming language in the Eclipse environment. This software can be integrated in object-oriented applications which have to solve similar types of optimization problems;
- OReL^{A} – an agent-oriented version which was realized in the Java programming language in the JADE environment [1]. This component version will be an integrated part of the virtual learning space [7].

OBJECT-ORIENTED REALIZATION OF THE MODEL – OREL^{O}

The information resources are pieces of data which can be combined in different ways according to the needs for a certain service. We would like the information resources to be spread equally within the InfoStation network in order to avoid overload in the communication environment. In order to achieve optimal distribution of the resources we have to run a simulation of that system which we will do by using two graphs – one dynamic and one static. The dynamic graph is formed from the InfoStation network where:

- each node is an access point from the InfoStation network containing resources;
- the edges between the nodes will be called global connections or globalizations and each one is a link (of a certain weight) between two resources;

The static graph is formed by the resources within the system and their relations where:

- each node is a resource which is on a certain InfoStation;
- each edge illustrates the relation between two resources, i.e. the dependence of one on the other;

For the realization of the algorithm we have to present the static graph in programming language. For that goal we present the resources (nodes) as objects with the information important for the system: name, position, list of connections of that resource with other resources.

- The name serves for differentiating between the different nodes and for a key for access to the object;
- The field "position" will store information for the resource's location on the InfoStation network.
- The list of the resource's locations will contain objects of the type "connection" which in turn will contain an object of type "node" (resource which is linked to

the current) and weight of the connection (a positive integer determining the importance of that connection).

The meaning of that algorithm is in the change of positions of two randomly chosen resources. After the change it is completely natural that the resources' connections change, i.e. some turn from global to local or vice versa. To find these changes out we run a matrix (containing information for the resources' location), the rating algorithm and we find the mark of its globalizations (the sum of the global connections' weights). Our goal is to decrease this mark so if after the change the mark is a smaller number than the initial, we say that there is improvement in the distribution, in the opposite case we do not have a change (the result is worse or the same). In the case of improvement, the initial state is terminated and the new state is saved as the same algorithm is applied again until a local optimum is achieved. If the change is not successful, another random change is made and the same rating process is repeated. To avoid program loops and introduce an end condition (for finding a local optimum), we use a counter for the unsuccessful iterations. If a certain iteration is an "improvement", we reset the counter, if it is not, we add one to it. When that variable reaches its critical value, which is predetermined, the program stops. As a result, the optimized resource distribution which is found is returned. The result may not be the absolute optimum for the entire system, but local one.

PERFORMED TESTS AND RESULTS

The goals of the performed tests are the following:

- To test the algorithm efficiency;
- To test the impact of the size of input data (population) on the quality of the solution given by the algorithm.
- To test the impact of the size of input data (population) on the time for executing the algorithm.
- To make a comparative analysis between the different changes in the algorithm end condition in each of the tests by comparing the quality of the generated solutions and time for executing the algorithm.
- To prove the effectiveness of the algorithm.

The tests are run on a computer configuration which has a 64-bit operating system Windows Windows 7 Ultimate (Service Pack 1) installed with the following parameters:

- Processor: Intel Core 2 Duo CPU T7500 @ 2.20GHz 2.20GHz
- RAM: Hynix DDR2 SODIM 2.00GB @ 667MHz

End condition	100 iterations	1000 iterations	5000 iterations	10000 iterations
Time for execution	1 s	3 s	9 s	18 s
Global connections	17	17	17	17
Local optimum	10	10	10	14
Iterations for achieving the local optimum	52	41	133	26

Table 1. Results with 9 resources and 3 stations OReL^O

End condition	100 iterations	1000 iterations	5000 iterations	10000 iterations
Time for execution	1 s	2 s	9 s	17 s
Global connections	27	27	27	27
Local optimum	16	16	16	16
Iterations for achieving the local optimum	90	36	31	35

Table 2. Results with 9 resources and 5 stations OReL^O

AGENT-ORIENTED MODEL REALIZATION – OREL^A

The idea for this algorithm is already realized with an object-oriented approach. The goal in creating the agent-oriented version is for it to be easily integrated in the already existing DeLC architecture which consists of a large number of software agents. The algorithm allows to separate the code in portions and create a multi-agent system. To realize the goal, we will reengineer the initial object-oriented code. Starting the method (and the method itself) for applying the evolutionary strategy will be integrated in an agent called CrossOver. The agent which will contain in itself the rating algorithm will be called the Rating agent. The agents will work in a container and will be able to communicate among themselves with messages.

AGENTS OPERATION

An agent called ***Transformer*** will display two behaviors. One is constructing a matrix out of the graph which contains the behavior of the InfoStation network resources. After the algorithm is executed, an object from the type EnviromentModel is returned which is ready to transport to the next agent (Rating). It is important to point out that in the object from the type EnviromentModel there is a matrix in which the remaining algorithms are executed (change of resources and rating of the result). When the Rating agent finishes its action, it sends an optimized matrix to the Transformer agent which renews the graph.

Agent (***Rating***): As we mentioned above, the Transformer agent sends a message to the Rating agent. Before the evolutionary strategy is applied, a globalizations' rating is made based on the input matrix (environment). This matrix is sent to the CrossOver agent. Certain processes – mutations, take place there and it returns a new – processed matrix (better or worse). The algorithm for globalizations rating is applied to this matrix. The agent uses the RatingAlgorithm to compare the matrices. The behavior of the Rating agent determines which matrix is better (the initial matrix representing the environment or its evolved copy) and sends it to the CrossOver agent. This operation is repeated several times. Each returned matrix is compared. This cycle continues until a certain number of negative iterations is achieved (iterations in which the evolved copy is not better than the initial). The best matrix is saved and sent to the Transformer agent which in turn has to renew the information in the graph.

Agent ***CrossOver***. This agent receives messages only from the Rating agent. In those messages there is a matrix on which the CrossOver agent executes operations of selection and crossing. The newly-received matrix is placed in an object from the type EnviromentModel and is sent to the Rating agent where it will be rated. This agent is invariably connected to the Rating agent and the performance of the two agents is in a master-slave dependency.

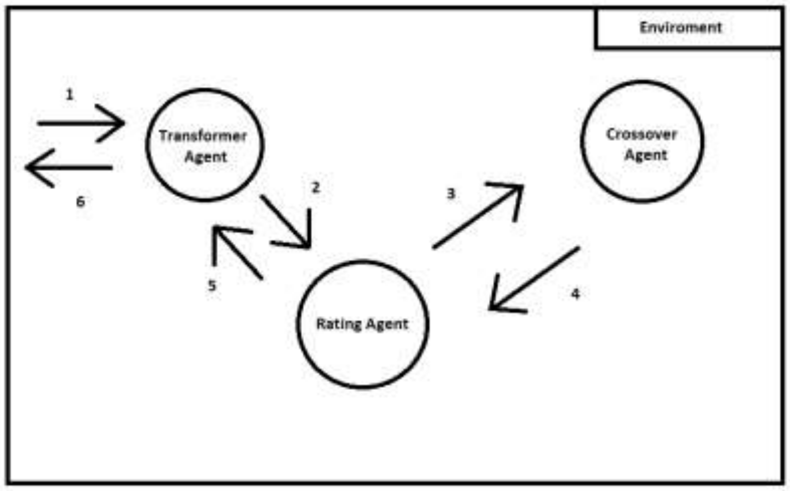

Figure 1.

The agent-oriented version of the optimization model was tested with the same input data as the previous realization in order to compare the effectiveness of the two architectures.

End condition	100 iterations	1000 iterations	5000 iterations	10000 iterations
Time for execution	406 ms	1 s	2 s	5 s
Global connections	17	17	17	17
Local optimum	10	10	10	14
Iterations for achieving the local optimum	92	13	10	5

Table 3. Results with 9 resources and 3 stations OReLA

End condition	100 iterations	1000 iterations	5000 iterations	10000 iterations
Time for execution	374 ms	1 s	3 s	5 s
Global connections	27	27	27	27
Local optimum	16	16	16	16
Iterations for achieving the local optimum	27	22	22	42

Table 4. Results with 9 resources and 5 stations OReL^{A}

CONCLUSION

Both algorithm realizations produce positive results. We can make the following generalization of the results:

- When the size of the input date is increased, the quality of the algorithm solution is deteriorated.
- When the size of the input data is increased, the time for the algorithm execution is increased.
- In both realizations, when the number of negative iterations is increased, the time for execution of the program is increased. When the size of the input data is decreased, the number of negative iterations can be decreased which will lead to identical results. When the size of the input data is increased, the negative iterations need to be increased because of the increase in the variations during crossing and changing.

Having in mind the approximately similar results from the two program realizations, the OReL^{A} would be more suitable from an architectural standpoint, for solving the optimization problems in DeLC for the following reasons:

- Agents mobility;
- Easy integration of the agents in the existing DeLC agent environment.

Nevertheless, OReL^{O} would be applicable in other systems which require the solution of similar optimization problems.

ACKNOWLEDGMENT

The authors wish to acknowledge the support of the Science Fund of the University of Plovdiv “Paisii Hilendarski” (Research Project Ref. No. NI13-FMI-02).

REFERENCES

[1] Bellifemine, F., G. Caire and D. Greenwood, *Developing Multi-Agent Systems with JADE*, Wiley, 2007.

[2] Stoyanov, S., I. Ganchev, I. Popchev and M. O’Droma, An Approach for the Development of InfoStation-Based eLearning Architecture, *Compt. Rend. Acad. Bulg. Sci.*, Vol. 62, No. 9, 2008, 1189–1198.

[3] Stoyanov, S., I. Ganchev, I. Popchev, M. O’Droma and R. Venkov, DeLC – Distributed eLearning Center, *1st Balkan Conference in Informatics*, Thessaloniki, Greece, 2003, 327–336, ISBN: 960-287-045-1.

[4] Stoyanov, S., I. Ganchev, I. Popchev and I. Dimitrov, Request Globalization in an InfoStation Network, *Compt. Rend. Bulg. Acad. Sci.*, Vol. 63, No. 6, 2010, 901–908.

[5] Stoyanov, S. and I. Popchev, Evolutionary Development of an Infrastructure Supporting the Transaction from CBT to e-Learning, *Cybernetics and Information Technologies (CIT)*, Vol. 2, 2006, 101–114.

[6] Stoyanov, S., I. Popchev, I. Ganchev and M. O’Droma, From CBT to e-Learning, *Information Technologies and Control*, Vol. 4, 2005, 2–10, ISSN: 1312-2622.

[7] Вълканов, В., *Контекстно-ориентирано управление на електронни услуги*. София, България: Академично издателство „Проф. Марин Дринов”, 2013, ISBN: 978-954-322-701-3.

Иван Марков Димитров,
Владимир Николаев Вълканов,
Ася Стоянова-Дойчева Пловдивски университет „Паисий Хилендарски“
Факултет по математика и информатика
4003 Пловдив
бул. „България“ 236
ivandimitrov@uni-plovdiv.bg,
vvalkanov@uni-plovdiv.net,
astoyanova@uni-plovdiv.bg

ОПТИМИЗИРАНЕ НА ГЛОБАЛИЗАЦИИТЕ В ИНФОСТЕЙШЪН АРХИТЕКТУРА

Иван Димитров, Владимир Вълканов, Ася Стоянова-Дойчева

Резюме. *Публикацията разглежда проблема за намиране на оптимално разпределение на ресурси в една Инфостейшън мрежа, с цел да се намали натоварването на комуникационната среда при изпълнение на потребителските заявки. Представени са две програмни реализации на модел за оптимално разпределение на ресурси в разпределена среда.*