Agent-based Approach to Course Timetabling for Nigerian Tertiary Institutions

Osun, O.A. & Sulaiman, M.O.
Department of Computer Science
Lagos State Polytechnic
Ikorodu, Lagos State, Nigeria
E-mail: femiosun@gmail.com, sulaimanmubarak@gmail.com
Phones: +2348036709808; +2348038587759

ABSTRACT

Course time tables in traditional done manually by the Heads of Departments of Nigerian tertiary institutions. Although a number of automated timetabling solutions, called Timetable Management Systems (TMS) exist, they have largely not caught on in Nigeria. These systems while they have recorded varying levels of success produce static timetables that are not suited to the peculiarity of Nigerian academia, where lecturers bear heavy course workloads, administrative responsibilities and often teach in more than one institution. This work proposes an agent-based time tabling approach (AB-TMS) that allows lecturers' agents to negotiate the course timetable slots on behalf of the lecturers, rather than one done by a central authority, and thereby evolve convenient timetables using lecturers' preferences dynamically.

Keyword: course timetabling, educational timetabling, agents systems

iSTEAMS Conference Proceedings Paper Citation Format
Osun, O..A. & Sulaiman, M.O.. (2018): Agent-based Approach to Course Timetabling for Nigerian Tertiary Institutions
Proceedings of the 14th iSTEAMS International Multidisciplinary Conference, AlHikmah University, Ilorin, Nigeria, Vol. 14, No 2 Pp 137-142

1. INTRODUCTION

1.1 Timetable Management Systems

School, examination and course timetabling according to (Ahmad, A.T., and Mohammed I.M, 2016) are the three essential educational timetables. In Nigerian tertiary institutions course timetables are traditionally done manually by the Heads of Departments and are static. The static nature of these timetables does not accommodate the dynamic and peculiar nature of Nigerian academia, where lecturers are often labored with excess teaching workloads, administrative responsibilities and most times teach in more than one institution. For these reasons lecturers frequently re-fix time slots for their lectures irrespective of what is on the timetable. This is usually done without due recourse to other lecturers and the Heads of Departments, and it is indeed, in most cases physically impossible to interface with other lecturers for negotiation. The effects of this are unavoidable clashes on the timetable, the eventual 'operating timetables' that evolve are totally different from the official ones, missed lectures attendance by lecturers.

Manually constructing accurate timetables is quite complex and time consuming (Burke, E.K., Elliman, D.G., and Weare, R.A. 2004), a lot of automated timetabling systems, called Timetable Management Systems (TMS), have been developed over time. TMS have enjoyed very low adoption in Nigeria. Unfortunately these systems while they solve the problem of time and accuracy still produce static timetables that do not make adequate provisions for the soft constraints of lecturers' preferences.

1.2 Agent-based Timetabling Management System

Agents are task-specific autonomous software that act on behalf of their owners. In the proposed system, the TMS is agent based. Each lecturer would have his timetable agent in the TMS agents ecosystem. The agents are aware of the courses their principal will teach for the semester, the available rooms and the preferred times of their principals. These agents then negotiate with themselves for time slots on the timetable to dynamically evolve a timetable that suits each lecturer's schedule. The agent based approach removes centrality, takes care of lecturers' preferences, eliminate clashes and free the Heads of Department from an onerous task.

2. RELATED WORKS

2.1 Scheduling Problem

Course timetabling is a specific instance of general scheduling problem. Given *m* machines M_j(j= 1…m), that have to process *n* jobs J_i(I = 1…n). A schedule is for each job is an allocation of one or more time intervals to one or more machines. The corresponding scheduling problem is to find a schedule satisfying certain constraints.

2.2 Course Timetabling Problem

University course timetabling is the process of assigning a number of events to a fixed number of slots in a week, and rooms which the sessions will take place. Timetabling problem involves the assignment of a set of events (meetings, matches, examinations, courses etc) into a limited number of time slots subject to a number of constraints (Burke, E.K., et al., 2002). (Nurul L.A and Nur A. H., 2016) also defines the course timetabling problem the allocation of events into time slots and rooms considering the list of hard and soft constraints presented in one semester, so that no conflict is created in such allocations.

These constraints must be properly identified and defined in order to have a valid timetable for any particular problem domain (Melicio, F. and Caldera, J.P and Ruso, A., 2004). (Melicio, F. and Caldera, J.P and Ruso, A., 2004) also identified the following data set for describing timetabling problem:

- ❖ A set T = $\{t_1…t_n\}$ of teachers or lecturers
- ❖ A set C = $\{c_1…c_n\}$ of classes. A is a group of students having the same curriculum.
- ❖ A set R = $\{r_1…r_n\}$ of rooms
- ❖ A set H = $\{h_1…h_n\}$ of time slots. Time slots are distributed in d week days and h daily periods, so H = d x h.
- ❖ A set A = $\{a_1…a_n\}$ of lessons. A lesson is a teaching unit. It is an instance of a list of teachers, a list of classes, a list of subject and a list of rooms. Each lesson has a duration expressed in time slots.

Timetabling constraints are hard and soft constraints. Hard constraints cannot be physically violated; for example, a class cannot receive two lessons at the same time, a lecturer cannot be with two classes at the same time and so on. Soft constraints signify preferences; a class may prefer a certain lecturer to take some lessons, a lecturer may prefer certain rooms or time slots.

The timetabling aim, data set and constraints effectively define the timetabling problem.

2.3 Approaches to timetabling problems

(Ahmad A., and Mohammed I.M., 2016) developed an acclaimed conceptual model for users' requirement categories for lecturers' in a timetabling system. These categories are perceived values, quality of service, efficient transaction, and functionality. These requirements still falls within the hard and soft constraints that specific timetabling systems must respect to be valid.

University course timetabling is divided into two, curriculum based tabling and post-enrollment based timetabling. When combination of courses must be taken to fulfill the requirement of a degree, the assignment of courses to time slots based on the curriculum is curriculum-based university timetabling. When timetabling uses the enrollment data of students to determine where courses are placed on the timetable, such that all students can attend the events for which they are enrolled, this is called post-enrollment based timetabling (Kristiansen, S. and Stidsen, T.R., 2013). (Lach, et al, 2012), Burke et al, 2013 and Cacchiani A., Caparan, R. and Toth, P., 2013) give very succinct description of the curriculum-based timetabling problems. Cambazard H., et al, 2008) and Cachia, S., Gaspero L., and Schaef A., 2012) describe the post-enrollment timetabling problem. These two approaches are not mutually exclusive and may be combined by a university.

Course timetabling problem is well researched and a lot of approaches exist for the resolution of the problem. Some of the approaches that have been used are Local Search Algorithms(Gaspero, L. and Schaerf, A,, 2003), Graph Colouring Algorithms (Azlan A., and Husin N., 2013), Swarm Intelligence algorithms (Chen, R. and Shih, H., 2013) and Exact Methods (Cachianni et al, 2013). (Pillay N., 2018) gives an overview of some other approaches.

All these approaches produce static timetables and do not give room for real-time flexible negotiations of time slots between contending parties.

2.4 Agents based systems

The goal of Artificial Intelligence had always been to replicate autonomous, intelligent and useful entities. As far back as 1977 (Hewitt, 1977) espoused the concept of "actors"; self contained concurrently–executing objects that encapsulate internal state and could respond to messages from similar objects. In recent times we have seen the emergence of software systems that behave if not exactly but quite close to this vision. Jasper (Davies, Weeks, & Revett, 1997), CMU Visitors' Hosting System (Sycara, 1995) and ADEPT (O'Brien and Wiegand, 1996) are points in reference.

(Nwana and Ndumu, 1997) identified the important features of agents as collaboration, learning and autonomy and put agents into types based on the emphasis they put on these characteristics. Collaborative agents, that emphasis autonomy and collaboration ahead of learning (O'Brien and Wiegand, 1996) and collaborate with other agents in its ecosystem to get useful work done for its owner. Interface agents emphasis autonomy and learning, they act as assistance to their owners and go between their owners and a third system the owner is trying to learn or use, providing useful guides and information (Maes, 1994), an example of interface agent is the ill-fated Microsoft Office Assistant. Mobile agents are agents that can leave their host systems and migrate to foreign hosts to perform tasks for their users and "return home". Issues of privacy, security and rogue behavior are of importance consideration with this type of agents. Information gathering agents gather, manage and process information for their owner from distributed source without leaving their native hosts. Jasper is such an example.

Reactive agents provide no internal representation of their environment or maintain a percepts history of interaction with the environment, but are simple stimulus-reaction devices (Russel and Norvig, 2003), they do not have a fixed architecture, the most popular architecture is one proposed by (Brooks, 1999), which is based on Augmented Finite State Machine (AFSM). Very few applications outside of simple games exist for reactive agents. Hybrid agents combine two or more of these architectures and leverage on their strengths to achieve its tasks. With the prolificacy of agent technologies, ontologies and increasing demands for agents to collaborate, (Genesereth, and Ketchpel, 1994) argues the need heterogeneous agent that combine the architecture of one or more agent types. Agent-based software engineering is a field that has emerged for developing heterogeneous agents.

Agents by their nature should be autonomous, proactive and social (Bellifemine, Caire, G. & Greenwood, 2007). Autonomy is the ability to carry out independent tasks of varying complexities, proactivity makes it possible for agents to take initiative and carry out needed task without explicit directives from their users, sociability implies it must be able to communicate with other agents to achieve its goal. Along with this core requirements are needs for agent registrations, service advertisement and life cycle management for agents, communication and coordination. Agent systems exist to provide these low level “plumbing” activities, so developer can concentrate on the business logic of the agent. Agent systems that support many agents are called Multi Agent Systems (MAS).

The most important services provided by MAS are communication and coordination. Agents need to communicate with the user, other agents and system resources. Inter-agent communication is by special Agent Communication Languages (ACL), that are based on Speech Act theory of (Searles, 1969). An ACL provides means for the exchange of information and knowledge between agents, ACLs must be both agent and semantics independent. (Genesereth and Ketchpel, 1994) reduced agency simple to the ability to communicate with an ACL. Knowledge Query and Manipulation Language (KQML) was the first ACL developed (Mayfield, Labrou & Finn, 1996). The new standard for ACL is a Foundation for Independent Physical Agents (FIPA) specification which is built on KQML called FIPA-ACL; this is the most commonly used ACL today (Labrou, Finn, & Peng, 1999). FIPA-ACL is fully implemented in Java Agent Development Framework, a MAS, also by FIPA.

FIPA has defined a specification for MAS for easy interoperability. FIPA MAS specification is implemented by Java Agent Development (JADE) Framework.

3. METHODOLOGY

3.1Agent Platform

Java Agent Development Environment (JADE) framework was chosen for this work. JADE is a implementation of Foundation for Independent Physical Agents (FIPA) specification. JADE is a distributed Multi-agent System (MAS) platform that is fully implemented in Java language; Java programming language is widely used and it is also cross platform. JADE provides the environment for the creation, instantiation and coordination of agents’ activities using Agent Communication Language (ACL) that is based on FIPA specification.

3.2 Time slots negotiation and the Dynamic Evolution of the Timetable

AB-TMS is an ecosystem of Lecturer Agents, HOD Agents, Class Agents and Rooms Agents constructed in the MAS. At the beginning of the semester, the HOD gives the course allocation to his agents and instantiate the Timetable Agent ans Room Agents with the appropriate constraint.

The HOD Agent communicate each lecturers courses to the Lecturer Agents and sets a deadline within which all time slots negotiation should be concluded. Each lecturer either before or after receipt of his allocation informs his agent about his time slots preferences and rooms preferences within the constraint laid out for those resources. Lecturers may specify a number of preferences for a lesson, in this implemented system, lecturers have a choice of three preferences for each weekly lesson for a course. The Lecturer Agents are then left to negotiate with each other to populate the timetable. Lecturers may change their preferences with their agents within the deadline, and after the timetable has been generated, if accommodation can be found with the specific Class Agent, Lecturer Agents and Room Agent.

In situations where resolution cannot be reached between the agents about a particular time slot and the deadline expires, the Timetable Agent Notifies the HOD Agent, who makes the final decision and completes the timetable.

4. DISCUSSION AND CONCLUSION

AB-TMS was implemented and tested with the Ordinary National Diploma programme at the Computer Science Department of the School of Part time Studies (Day) of Lagos State Polytechnic for course timetabling for first semester 2018/2019 session. The department employed fifteen lecturers and eleven technologist for that semester. There were six classes, eight rooms, that consisted of six classrooms and two laboratories. The National Diploma programme had six time slots per day, making a total of thirty time slots for the 5-day week. Each class had a dedicated room for lessons and only the two laboratories were shared for practical lessons.

The system generated the semester timetable without difficulty. Two cases were reported to the HOD Agent for arbitration. Those cases had to do with the use of the laboratories. The preliminary testing shows promising results. The timetable was done within the stipulated time, lesson clashes were eliminated and lecturers put up better attendances for their classes. The test was done in just one department of the entire polytechnic, extending the system to a more complex environment with more stringent hard constraints will reveal its weakness and where work needs to be done.

REFERENCES

1. Ahmad A. and Muhammed I. (2016). *User Requirement for model for university timetabling and constraints manipulation.* International Journal of Software Engineering and Application, May 2016. DOI: 10.5121/ijsea2016.7301
2. Azlan, A. and Hussin, N.(2013). *Implementing graph colouring heuristics in the construction phase of curriculum-based course timetabling problem.* Computer Informatics (ISCS), 2013 IEEE Symposium, pp. 25-29, 2013. DOI: 10.1109/isci.2013.6612369.
3. Burke, E.K, MacCarthy, B.L, Petrovic S., and Qu R.(2002). *Knowledege discovery in hyper-heuristics for course timetabling using case-based reasoning.* Proceedings of PATAT02, pp. 90-103. Aug., 2002.
4. Burke, E.K., Elliman, D.G. Weare, R.A.(1994). *University timetabling based on graph colouring constraints manipulation.* Journal of Educational Computing Research, vol 27, no 1, pp. 1-18
5. Burke, K.E., Marc, J., Andrew, J. and Rudova, H.(2012). *A branch and cut procedure for udine timetabling problem.* Annals of Operations Research. 194(1): pp. 71 -87. ISSN:0254-5350.
6. Cacchiani A., Capara, R. and Toth, P. (2013). *A new lower order bound for curriculum-based course timetabling.* Computer & Operations Research, 40(10), pp. 2466-2477, ISSN:0350-4548.
7. Cambazard, H., Hebrand, H. and O'Sullivan, B.(2008). *Local search and enrollment programming for post-enrollment timetabling problems. programming for post-enrollment timetabling problems.* Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, 2008.
8. Cechia, S., Gaspero, L., Shaerf, A.(2012). *Engineering and experimental analysis of a simulated annealing approach to the post-enrollment course timetabling problem.* Computer & Operations Research, 39(7): pp. 1615-1625, 2012.
9. Chen, R. and Shih, H.(2013). *Solving university course timetabling problem using particle swarm optimization with local search algorithm.* 6(2): pp. 227-244, 2013, ISSN 1999-4893.
10. Davies, N.J., Weeks, R., Revett, M.C. (1997). *Information agents for the world wide. web.*BT Laboratories, Martlesham Heath, Ipswich, Suffolk, IP57re, UK.
11. Gaspero, L. and Schaerf, A.(2003). *Multi-neighbourhood local search with application to course timetabling.* In Burke, K.E. and Causmaecker, P. (eds), Practice and Theory of Automated Timetabling IV, vol. 2740 of Lecture Notes on Computer Science, pp. 262-275. Berling Heidelberg, 2003. ISBN 978-3540-10-46699-0.
12. Hewitt, C. (1997). *Viewing control structures as patterns of passing messages.* Artificial
13. Intelligence, 8. No.3 323-364 (1997).

14. Kristiansen, S and Stidsen, T.R.(2013). *A comprehensive study of educational timetabling : a survey.* DTU Management. DTU Management Engineering Reprint, no. 8, 2013.
15. Lach, G and Lubeck M.U.(2012). *Curriculum-based timetabling: new solution to udine behavior instances.* Annals of Operations Research, pp. 255 – 272. ISSN: 0254-5350.
16. Woodridge, M., Muller, J. and Tambe, M. (1996) (eds.) Intelligent Agents II (LNA 1037, Springer-Verlag, Heidelberg, 347-360.
17. Lassila, O and Webick, R. (1999). *"Resource Description Framework (RDF)model and syntax*
18. Maes, P.(1994). *Agents that reduce work and information overload.* Communications of ACM 37, no. 7, 31-40.
19. Mayfield J., Labrou, Y and Finn, T. (1996). *Evaluating KQML as an agent communication language*
20. Melicio, F., Caldera, J. and Ruso A. *Two neighbhourhood approach to timetabling problem.*Proceedings of the International Conference on Practice and Theory of Automated Timetable, 2004.
21. Nurul, L. A. and Nur A.H.(2018). *A brief review of on the features of university course timetabling problem.* AIP Conference Proceedings, 2016. 0020001(2018). http://doi.org/10.1063/15055430.
22. Nwana, H.S., Ndumu, D.T. (1997). *An introduction to software agent technology.* Chapter in Book.
23. O'Brien, P. and Wiegand, M.(1996). *Agent of change in business process management.* BT Technology Journal, 14, No.4 . (1996).
24. Pilley N.(2014). *A survey of school timetabling research.* Annals of Operations Research, vol 218, no.1, pp. 268-293, July 2014.
25. Searles, J. (1969). *Speech act.* Cambridge University Press, Cambridge MA.
26. Sunyato, S.(2010). *An informed generic algorithm for university course and student timetabling problem.* Proceedings of the 10th International Conference on Artificial Intelligence and Soft Computing Part II. ICAISC'10, pp. 229-236. Berling Heidelberg 2010. Springer-Velag ISBN: 3-642-13231-6, 978-3—642-13231-5.
27. *W3C rcomendation.* Retrieve from http://www.w3.org/TR/PR-rdf-syntax in January 2015.