# An Agent Based Technique for Improving Multi-Stakeholder Optimisation Problems.

Neil Urquhart[1] and Simon T. Powers[1]

Edinburgh Napier University, School Of Computing, 10 Colinton Road, Edinburgh, United Kingdom. n.urquhart@napier.ac.uk

**Abstract.** We present an agent based framework for improving multi-stakeholder optimisation problems, which we define as optimisation problems where the solution is utilised by a number of stakeholders who have their own local preferences. We explore our ideas within the domain of the University Timetabling Problem, demonstrating how a solution created by traditional timetabling methods may be further improved from the perspective of individual stakeholders (students) by agent based methods. We also note that this approach lends itself to increasing the level of trust in such systems by potentially allowing the stakeholders to view the actions taken by agents on their behalf.

**Keywords:** Timetabling · Optimisation · Real-World · Explainability

## 1 Introduction

Many real-world optimisation problems exist within a problem domain that has many stakeholders, who each make use of the solution produced and have their own "local" opinions and preferences. Such problems are frequently solved using a centralised solver that produces a feasible solution that addresses the global problem constraints. Examples of such problems include University timetabling [2], Vehicle Routing Problems [1],mobile workforce scheduling [4] or crew scheduling problems [5], see table 1.

In the domains listed in Table 1 there may be many stakeholders each of which will have a view as to how the solution should meet their requirements. For instance in timetabling a member of staff or a student may wish to avoid having classes at specific times, or within crew scheduling an individual may have a preference to work shifts that cover certain times or have a preference for specific types of work.

| Problem Domain | Stakeholders |
|---|---|
| Timetabling | Staff and Students |
| Vehicle Routing | Drivers and Customers |
| Mobile Healthcare Scheduling | Employees and Patients |
| Crew Scheduling | Employees |

**Table 1.** Some optimisation problem domains and associated stakeholders.

In this paper we suggest that the interests of stakeholders can be represented by a multi-agent system (MAS). We propose that existing optimisation techniques are used to produce a solution that satisfies the global hard and soft constraints of the problem. This initial solution is then passed to a multi-agent system for personalisation according to the needs of the stakeholders.

Within the MAS there are two classes of agent: the *Problem Agent*, and a set of *Stakeholder Agents* (see Figure 1 and Table 2). The Problem Agent maintains the solution and ensures that it remains valid (i.e. that it continues to satisfy the hard constraints of the problem). The Stakeholder agents each represent the views of one particular stakeholder and request changes to the solution in order to better meet the needs of the stakeholder.

| Agent | Goal | Possible Actions |
|---|---|---|
| Stakeholder Agent | Achieve as many of the local objectives as possible | Request moves and swaps within the global timetable |
| Problem Agent | Maintain the integrity of the global solution. Action as many of the requests from stakeholders as possible | Allow request<br>Deny Request |

**Table 2.** The goals and actions of the agents within the system.

In this study we will use the University Timetabling domain based on the authors' experiences at Edinburgh Napier University as a test-bed for our ideas.

## 2 Main purpose

The University Timetabling problem domain is well known, [2] and many techniques exist to optimise timetables according specific criterion. In the case of Edinburgh Napier University, timetables for staff and students are created by a commercial software package used by University administrators. With an increasing emphasis on improving the student experience, producing timetables that are tailored towards the needs of students becomes more important.

The authors propose that the University's conventional software is used to produce a feasible timetable, which is then passed to our system for customisation. The role of the Problem Agent (as defined in Section 1) is undertaken by the Timetable Agent and the stakeholders by Student Agents.

To facilitate communication between the Timetable Agent and the Student Agents a timetable ontology has been designed. Within the ontology a timetable is defined as a collection of 45 slots, each occupying 1 hour (from 09:00 to 17:00 giving 45 slots over a 5 day week). Each slot may be occupied by an event, each event being associated with a module and having a type as lecture, practical or tutorial. Each event also takes place within a room, the size of which limits the numbers of attendees.

Each Student Agent is supplied with a copy of their timetable by the Timetable Agent. The Student Agent may then evaluate its timetable against criteria supplied by its stakeholder. This criteria takes the form of a list of slots that the stakeholder wishes to keep free. The Student Agent can now evaluate their

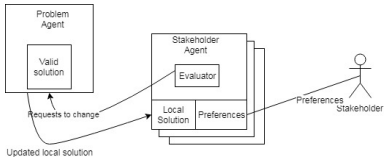


**Fig. 1.** The MAS architecture.

timetable against this local criterion, highlighting events which occupy slots that the student wishes to keep free. A second ontology describes the requests which my be made by the Student Agent. Such requests are possible as many events (especially tutorials and practicals) are duplicated. A move request is suitable where a student wishes to move to an event that has space for an incoming student. Where an event is full, a swap must be arranged with another student making a reciprocal move between events. Where a swap has to be arranged, the student initiating the swap passes the swap request to all other students, who respond if they are able to make the swap.

A fundamental principle is the the Timetable Agent treats all requests as atomic - each request is fulfilled entirely or rejected. Any request that would breach a hard constraint within the timetable is rejected. In this way the Timetable Agent ensures the overall integrity of the timetable.

## 3 Demonstration

The authors have implemented the basic system in Java using the Java Agent Development Environment [3] for the agents, ontologys and message passing. The demonstrator can import data from the University timetabling system and can also be used with simpler test data.

Figure 2 shows a sample sequence of messages passing between 2 stakeholders and a timetable agent. In this simple example the agents can be seen requesting changes (moves) and being notified of the outcomes.

In order to allow the Student Agents to hold realistic criterion, students within the university were surveyed regarding their attitudes to timetabling. One of the questions within the survey required the student to highlight those slots which they wished to keep free. In this initial version that information was used to allow the Student Agents to evaluate their timetable. Each student agent is randomly allocated the survey results of a respondent. In a production system a student would advise their agent directly through a suitable interface.

## 4 Conclusions

We present a technique for using agents to improve University timetables. We believe that this framework could be applied to other multi-stakeholder optimisation problems (see table 1).

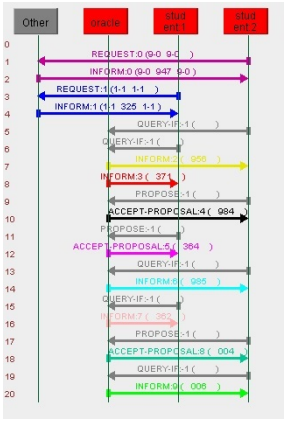


**Fig. 2.** A screen shot from the JADE sniffer demonstrating 2 stakeholders (student1 and student2) optimising their timetables by communicating with the timetable agent (oracle). Requests for changes are Proposals messages and the response (confirming or denying the change) is an Accept_Proposal. We can also see the agents requesting updated timetables (query_if) and being informed of the updated timetable (inform).

A major development to be implemented is to allow a coalition of Student Agents to move an event (subject to timetable constraints). Also to be investigated is measuring the global and local utility of changes and whether that utility value can be used to encourage changes which result in the most effective improvements. We also believe that by allowing stakeholders to specify their own local criteria, and then explaining why the local agent was or was not able to have changes made to accommodate them, we can increase trust in such systems.

## References

1. Adewumi, A.O., Adeleke, O.J.: A survey of recent advances in vehicle routing problems. International Journal of System Assurance Engineering and Management **9**(1), 155172 (Feb 2018)
2. Babaei, H., Karimpour, J., Hadidi, A.: A survey of approaches for university course timetabling problem. Computers  Industrial Engineering **86**, 4359 (2015). https://doi.org/10.1016/j.cie.2014.11.010
3. Bellifemine, F., Bergenti, F., Caire, G., Poggi, A.: Jade — A Java Agent Development Framework, pp. 125–147. Springer US, Boston, MA (2005). https://doi.org/10.1007/0-387-26350-$0_5$, $https://doi.org/10.1007/0-387-26350-0_5$
4. Urquhart, N., Hart, E.: Optimisation and illumination of a real-world workforce scheduling and routing application (wsrp) via map-elites. Parallel Problem Solving from Nature  PPSN XV Lecture Notes in Computer Science p. 488499 (2018). https://doi.org/10.1007/978-3-319-99253-$2_3$9
5. Yen, J.W., Birge, J.R.: A stochastic programming approach to the airline crew scheduling problem. Transportation Science **40**(1), 314 (2006). https://doi.org/10.1287/trsc.1050.0138