MAS Project – Emergent Timetabling

1. Assignment Objective

The purpose of the project is to study the dynamic solving of the *academic timetabling problem* by means of **cooperative, self-organizing agents**.

High-level problem description
Teachers and *student groups* have to find partners, time slots and rooms to give or to take some courses. Each actor (*teacher* or *student group*) has some constraints concerning its availability or required equipment (e.g. a projector or a white board must be available in the room).

Actors are modeled as agents, while the timetable (together with the rooms and their available equipment) is modeled as the environment in which the agents operate.

The organization of the agents into a correct timetable has to ***emerge*** from the interactions that they undertake in the environment (e.g. partnering of a teacher agent with a student group agent at a particular time slot, in a given room).

Furthermore, there is an interest to see how the organization changes when actors modify *add* or *remove* constraints during the solving process.
The agents may converge on an exact solution or one that violates the fewest constraints (if an exact one is not possible).
An analysis is requested to determine aspects such as: time of convergence, best time grid exploration policy, time to re-convergence if a constraint is added/removed.

This problem is given the name of Emergent Timetabling Organization (ETTO).

2. Assignment Specification Summary

Two different classes of agents are identified to tackle the ETTO problem: **Representative Agents** (RA) and **Booking Agents** (BA).
The exploration of the solution space, an n-dimensional grid of cells, is delegated to Booking Agents. Each cell in the grid represents **a time slot, in a room, for a given day.**

Each cell c_i of the grid is constrained (e.g. availability time slots for a room, number of places in the room, availability of equipment in room). All the constraints of a cell are regrouped in a set C_{ci}. Cooperation between agents must lead to a correct organization by efficiently exploring the grid.

Representative agents (RA) are the interface between human actors (teachers or student groups) and the timetabling system. They own a set of *intrinsic constraints* about availability, equipment requirements (projectors or whiteboards) or any other personal constraints.

An RA creates a set of **Booking Agents (BA),** also known as *delegates*, which perform the concurrent exploration of the timetable grid in search of possible slots in accordance with the constraints set up by the RA.
An RA creates as many BAs as it has classes to teach (in case of a teacher), or to take (in the case of a student group).
The task of the RA is to warn its delegate BAs when its user adds or removes constraints and to inform all its delegate BAs when one of its delegate BAs produces new constraints (called induced constraints), to ensure coherency.

Booking Agents are the real self organizing agents. They reserve time slots and rooms and find partners (student groups for teacher and vice versa) in accordance with constraints owned by their proxy.
While roaming the timetabling grid, BAs can only perceive their current cell and know how to move to:

- adjacent cells
- cells they keep in memory

BAs have a classical perceive-decide-act life cycle:

- perception phase: BA checks its messages (coming from other BAs or its proxy - RA) and updates data about the cell (BAs in the cell, post-its, properties of the cell) in which it is positioned
- decision phase: BA must choose the next action to perform to be as cooperative as possible, in accordance with the **cooperation rules**
- action phase: BA performs chosen action

The actions a BA can perform are:

- partner / un-partner with another BA
- book / un-book a cell – the BA can only book the cell it currently occupies, or cells it remembers from previous visits
- move to another cell (only adjacent cells, or cells it remembers from previous visits)
- send messages to agents it *knows* (a BA knows its proxy (the RA) and any BAs that it encounters at runtime)

In order to perform their actions Booking Agents:

- have a set of **local properties, capabilities and knowledge**
- obey a set of **cooperative self-organization rules**

The complete specification of the agent knowledge and organization rules is given in the article attached to this project description [1].
Please read carefully Sections 2.1, 2.2 and 2.3 related to agent description and their capabilities, as well as Sections 3.1 – 3.5, related to cooperative self-organization rules applied in case of conflicts.

3. Assignment grading and implementation specifications

The assignment implementation must have the following features:

- Represent the default test cases and two additional test cases designed by yourself in format of your choice, such that each test case is represented in its own file (can be a source file or a text file). The name of the test case is given in a command-line argument when running the program.
- Environment implementation – 3.5p
    - implement the grid-like structure representing time slots per room, per day
    - maintain for each grid cell: existing reservations, current occupancy, time constraints (e.g. unavailability of a room at that time because of external constraints, other than timetable reservations) and room capabilities (e.g. contains a projector or not)
    - the initial data for each cell is read from the test case
    - act as the message router between agents, implementing a message queue for each BA
    - implement a turn-based agent action execution in three global phases:
        - set all perceptions for each BA agent:
            - updated message queue

            - current cell reservations
        - collect all agent actions
        - execute agent actions in (possibly prioritized) order
        - at each step, output to the system console all messages exchanged between agents (1 line to describe each message).

- RA and BA implementation – 3.5p
    - implement RA and the constraint initialization (e.g. availability for teachers)
    - the initial data for constraints is read from the test case
    - implement BA creation function
    - implement the reasoning cycle of Booking Agents
    - at each step, output to the system console 1 or 2 rows of text for each BA describing the reasoning of the BA in that step and the chosen action
    - implement a random grid exploration strategy for BAs who don't have a partner or a reservation, or who have suboptimal ones
    - implement the cooperative self-organization rules

- Evaluator implementation – 2p
    - implement environment and agent initialization (e.g. availability constraints, room equipment, assign weights for each)
    - implement the evaluation function which computes the total number of constraints violated by the current configuration of the agents on the timetable grid
    - implement the simulator which runs the environment steps and agent actions
        - one must be able to stop / restart the simulator at will (via a visual interface – a button)
        - when stopped the simulator should display a graphical output (format at choice, in the same visual interface) showing the current timetable configuration (reservations, positions of Bas, etc) and the set of violated constraints (if any)
    - implement the ability to add/remove BAs (e.g. remove some of the teachers holding a class) and constraints (e.g. add, remove time slot availabilities)

- Analysis – 1p
    - analyze impact on convergence/reconvergence speed (interpreted as stabilization on a solution) of the following:
        - number of BAs (i.e. number of student groups + teachers)
        - number of constraints
    - There must be at least one analysis of a test run where the following happens:
        - the evaluator starts the solving of an instance
        - the evaluator is halted after a given number of cycles
        - n1 BA(s) for teachers are removed and n2 time availability constraints are added, where n1 >= 1 and n2 >= 1
        - the evaluator resumes agent solving
        - after a given number of cycles using the new configuration, the evaluator is halted again
        - n1 BAs are added back and n2 constraints are removed. However, the change is not a perfect mirror image of the first one (i.e. new BAs with new constraints are added and constraints are relaxed elsewhere than in the previous case)
        - the evaluator is resumed
        - the analysis looks at the time to re-stabilize on a solution after each change

- Bonus – 2p
  - implement a heuristic strategy (different from random) for the timetable exploration actions of BAs
  - analyze the same performance metrics as above for the case of using such a strategy

4. Default test cases

The assignment contains two default test cases. They cover a scheduling problem running over 2 **days,** for **3 teachers** and **3 student groups** in varying number of lecture rooms.

- **Case 1**
  - time slots of 2h: 8h-10h, 10h-12h, 14h-16h, 16h – 18h
  - two days: d1, d2
  - 3 teachers: T1, T2, T3
  - 3 student groups: SG1, SG2, SG3
  - 3 lecture rooms (one for each student group): R1, R2, R3 – no restriction on lecture room availabilities

  - T1: not available D1: 16h – 18h, D2: 14h – 16h
  - T2: not available D1: 16h – 18h, D2: 10h – 12h
  - T3: not available D1: 14h – 16h, D2: 8h – 10h

  - Over the course of D1 and D2, each SG must take exactly two courses of 2h each, taught by each of the 3 teachers (i.e. 2 * 2 * 3 = 12 h of class over the course of 2 days)
    - for convenience you may further consider that one 2h slot is the course work and the other is lab work

  - For this test case the program must reach a constraint free solution (e.g.)

Time Slot	**SG1**	**SG2**	**SG3**
8h-10h	T1	T3	T2
10h-12h	T3	T2	T1
12h-14h	▨	▨	▨
14h-16h	T2	T1	
16h-18h			T3
	Day 1		

SG1	**SG2**	**SG3**
T1	T2	
	T1	T3
▨	▨	▨
T2	T3	T1
T3		T2
Day 2		

- **Case 2**
  - Constraints are added on lecture rooms. Rooms R1, R2 and R3 are available. **R1** and **R2** are lecture rooms and each **have a projector**, while R3 is a laboratory and has no projector.
  - R1 not available D1: 10h-12h
  - R2 not available D2: 8h-10h, 16h-18h
  - R3 not available D1: 14h-16h, D2: 16h-18h

  - All teachers want to use a projector (for lectures) at least once for each of the 3 student groups, over the course of the two days.

- **We allow one or more of the actors (student groups, teachers, rooms) to relax their constraints in order to arrive at a solution.**

- A possible solution is thus the following:

Time Slot	SG1	SG2	SG3
8h-10h	T1/R1	T3/R2	T2/R3
10h-12h	T3/R2	T2/R3	T1/R1
12h-14h			
14h-16h	T2/R1	T1/R2	
16h-18h			T3/R1
	Day 1		

SG1	SG2	SG3
T1/R3	T2/R1	
	T1/R1	T3/R2
T2/R3	T3/R1	T1/R2
T3/R2		T2/R1
Day 2		

In case of relaxing the constraints: R1 not available in D1: 10h-12h, to place T1/R1 at that time slot, and the constraint R2 not available D2: 16h-18h, to place T3/R2 at that time slot.
One could also try to relax the constraint T3 not available D2: 8h-10h (in this case for SG1 on day 2, one could put T3/R2 in slot 8h-10h and push T1/R3 at slot 10h-12h, without breaking any constraints).

References

[1] Picard, Gauthier, Carole Bernon, and Marie-Pierre Gleizes. "ETTO: emergent timetabling by cooperative self-organization." International Workshop on Engineering Self-Organising Applications. Springer Berlin Heidelberg, 2005.