Software and Data Technologies: 4th International Conference, ICSOFT 2009, Sofia, Bulgaria, July 26-29, 2009. Revised Selected Papers. Communications in Computer and Information Science, Volumen 50. Springer, 2011. 218-230.

DOI: http://dx.doi.org/10.1007/978-3-642-20116-5_17

Educational Resource Scheduling based on Socio-inspired agents

Juan I. Cano[1,2], Eloy Anguiano[2,3], Estrella Pulido[2], and David Camacho[2⋆]

[1] Instituto de Ingeniera del Conocimiento
[2] Escuela Politcnica Superior - Universidad Autnoma de Madrid
[3] Centro de Referencia Linux UAM–IBM
{inaki.cano,david.camacho,estrella.pulido,eloy.anguiano}@uam.es

Abstract. Scheduling a set of constrained resources is a difficult task, specially when there is no clear definition of 'optimal'. When the constraints depend not only on physical or temporal issues but also in human desires or preferences the task gets harder. This is the case of educational resources, for example when a set of students must be distributed into a limited set of finite laboratories to attend to periodical, in this case weekly, practical sessions. The preferences of the students may vary during the process for reasons such as the number of people already in that group. This paper presents a socio–inspired solution implemented as a multiagent system. The agents enroll themselves in the lab sessions based on their preferences and negotiate with other agents, using the resources they already have, to obtain desired groups that were already full.

Keywords: Scheduling, Multiagent System, Multiagent Resource Allocation, Constraint Satisfaction Problem, Socio–inspired, Complexity Science

1 Introduction

Resource allocation has always been a huge concern for administrators. These problems, from the family of constraint satisfaction problems, can be seen in several domains, as for example the allocation of processing time to the users of a mainframe [5] or the assignment of runways to planes in an airport [4]. Finding a way to solve these problems efficiently is important as some of them appear in time critical situations.

In general, in a resource allocation problem we have a set of resources that are limited and a set of agents that need these resources in some specific way. The nature and characteristics of these resources are very important when deciding a solution as they define part of the constraints to take into account. The rest of the constraints are defined by the agents or are imposed externally. A solution of the problem is found when we can make a feasible allocation (every agent has

⋆ This work has been partially supported by the Spanish Ministry of Science and Innovation under grants TIN 2007-65989, TIN 2007-64718, and TIN2010-19872.

a resource assigned) or we find the optimal allocation. In the latter case we must decide on a way to measure the optimality of a solution. Also, there are cases where there is no possible solution.

This kind of problems have been treated in many different ways [3, 7]. One approach recently developed is MultiAgent Resource Allocation (MARA)[2], which uses multiagent systems to solve the allocation problem. This approach comes very natural as instead of programming an abstract algorithm we design a model of the problem, create some behavior for the agents and let system evolve to a solution. The key of this method is to capture the relevant aspects of the problem and define some utility function for the agents.

In the first section of this chapter there is a precise description of the problem, starting with an overview of the problem and then dealing with the relevant details to build the solution. In the second section we explain the design of the built system and why some decisions were made based on the description of the problem. Next, we will try to analyze the system from a complexity science point of view, identifying inputs, outputs, feedbacks and how they affect the system. Finally, we will show some results and conclusions, and how this system could be improved.

2 Description of the Problem

The problem to be solved was chosen for its familiarity and difficulty. Every year, in our university, the students enroll in some courses, usually five per term. The majority of this courses have two parts, one theoretical and one practical. The theoretical part of the courses are usually not a problem when allocating students, there are enough teachers and classrooms for the lectures, but for the practical part of the courses the laboratory space is limited. Even if there were huge computer labs, the relation between student and teacher is crucial for this part.

The practical sessions take place once a week. At the beginning of each term, the coordinator of the practical sessions for each course defines the number of groups, who is going to teach each of them and when the practical session for each group takes place. The group size is limited by space available in the lab. Students can join any of these groups, but once they choose a group they are required, unless there's a good reason, to attend to that group until the end of the term. This creates some conflict and competition among the students as they have time restrictions and preferences over the teachers (see fig. 1), and the group size limit makes that not every student is satisfied with her assigned groups.

This gets even more complicated as they have more than one course, usually five as mentioned earlier (see fig. 2). A student cannot join groups that overlap in time and this adds a dynamic restriction to the problem: once a student is assigned to a group, the time slot when the sessions take place are no longer available for other course's groups. This is a big problem from a CSP point of view, what once was a feasible solution or even an optimal solution can change

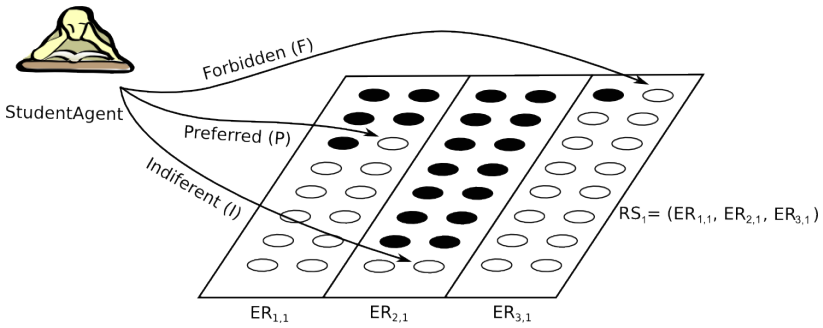

Fig. 1. Student preferences in a course with three groups. The whole square represent the course, the vertical divisions the groups. White dots are available places, black dots are occupied seats. The student in this figure preferes the first group, cannot assist to sessions in the third group and shows herself indifferent about the second group.

after one assignment and become a terrible solution, and the other way around, what once was a terrible solution can become an optimal solution.

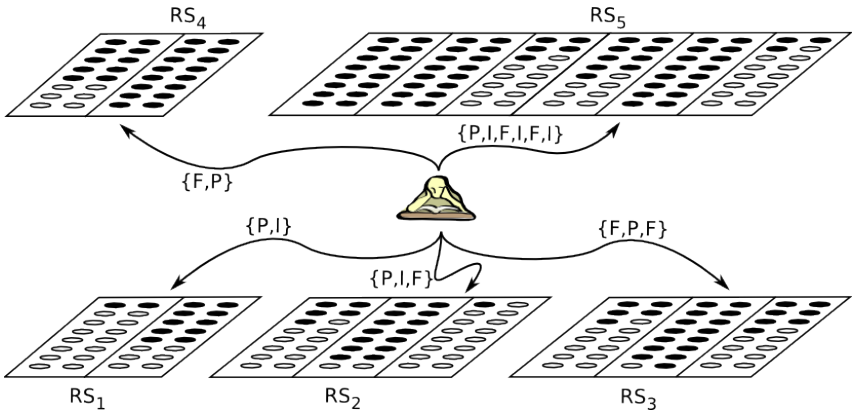

Fig. 2. A student's global situation. Each RS (Resource Set) represents a course, and each course is divided as explained in fig. 1. The values between curly brackets represent the preferences of the student. For example, {P,I,F} means that the student prefers the first group (P), is indifferent about the second group (I) and cannot assist to the third group (F).

Once explained the problem in general terms, we can analyse it precisely to build a solution. In the following description we are going to use Resource Set (RS) for courses and Educational Resource (ER) for the groups. This way ER_1 will be the course number one and $RS_{1,1}$ will be the first group of the first course. To address correctly the problem, it will be described by a set of

properties and characteristics (resource types, preference representation, social welfare, allocation procedures, complexity, and simulation) from [2], which can be used to characterize MARA systems and applications.

- Resource Type
  - *Discrete vs Continuous:* The ER are discrete, they cannot be represented with real numbers nor it can be divided.
  - *Divisible or not:* The ER are not divisible
  - *Sharable or not:* Each ER can be assigned to more than one agent at the same time, but they have limits (seats/places).
  - *Static or not:* The ERs doesn't change during the negotiation, they're not consumable nor perishable.
  - *Single–unit vs Multi–unit:* Each ER is unique, it can't be confused with another ER.
  - *Resource vs Task:* The ER is assigned as a resource, not as a task.
- Preference Representation
  - *Preference structure:* Although our model distinguishes between three different choices (Preferred (P), Indifferent (I), Forbidden (F)) of a particular educational item, and we could construct an ordinal preference structure (where $P > I > F$), we use an evaluation function that translates the agent preference into an integer which is used later to obtain a quantitative value, so a cardinal preference structure is the structure used.
  - *Quantitative preferences:* A utility function is used to map the bundle of resources assigned to an agent into a quantitative value, which will be later maximized.
  - *Ordinal preferences:* Not applicable.
- Social Welfare
  Our approach is based on Collective Utility Function (CUF) because the aim is to maximize the average value of individual agent welfares. In the egalitarian social welfare, the aim is to improve the agent with the lowest welfare and in the utilitarian social welfare the aim is to improve the sum of all welfares, whereas in our approach the global state of the whole agent society by means of the average welfare is the aim of optimisation. Among the different possibilities (pareto optimality, collective utility function (CUF), leximin ordering, generalisations, normalised utility or envy–freeness) a CUF has been selected that defines the utilitarian social welfare as the total sum of agent cardinal values divided by the total number of agents.
- Allocation Procedures
  - *Centralized vs Distributed:* Our approach is fully distributed, since the solution is reached by means of a local negotiation amongst agents and there is not a global perspective of the ERA problem. An aggregation of individual preferences is used and the agent preferences are used to assess the quality of the global resource allocation.
  - *Auction protocols:* No auction algorithm is considered.

  - *Negotiation protocols:* A simplified version of the Concurrent Contract–Net Protocol (CCNP) has been implemented, where each agent can act as a manager and a bidder in the simulation step[9].
  - *Convergence properties:* Our negotiation algorithm needs a multilateral deal, where any interested agent in a particular educational item can negotiate with the manager (in our approach the agent who is trying to obtain a specific allocation).
- Complexity
  Analysis of the models and assumptions, or the computational vs. communication complexity, used in our approach is not relevant for now.
- Simulation
  A Multi–Agent Simulation Toolkit (MASON) has been used to deploy and test our proposed solution [6, 8].

To summarize the restrictions of this problem, we have that each student must have one, and only one, group assigned for each course with practical sessions. The students have preferences over the groups and are limited by other courses they are assisting, so the solution should maximize the student's happiness and can't assign to a student groups that overlaps in time. To these preferences we add that the groups must be evenly distributed and, by our institution requirement, courses from higher levels must have preference over the rest.

3 The design of the system

The main objective of this work was to design a light system capable of solving this problem efficiently. Another objective was to use ideas and concepts that comes natural with the problem described. The multiagent approach let's the researcher use the problem terminology when constructing solution and the built system can be said to be socially inspired.

While building the solution, we created a model of how students interact between themselves when enrolling into groups. We observed that when signing into a group's list, students negotiate between them exchanging positions they have for the ones they want to have. This process is very local as they only contact friends and usually limit themselves to one subject or one group. In the model created, students negotiate with every other student that have something to offer. The exact negotiation process will be explained bellow.

Before explaining the negotiation, we need to understand how the students perceive their status and evaluate the proposals. We define the student's happiness or utility function as a function that increases with the number of groups assigned that they deem preferred. We have to maintain the groups balanced as this is good for the students as it is for the teachers, so the student happiness also varies with the occupation of the group in relation with the occupation of the other groups of this course. The happiness function would have the following form:

$$H(a_i) = \frac{\sum_{i=1}^{n}(f_1(RS_i(a)) + f_2(q, RS_i(a)))}{n}$$

In this equation we have $RS_i(a)$ that returns the group assigned to student a in the RS_i, q is the group's ocupation, f_1 maps the student preference to an integer value, f_2 represents how the student perceive the balance of the group and n is the number of subjects the student has.

As explained in the above section, each student can assign one of three preference categories to each subject: P for Preferred, I for Indifferent and F for Forbidden. The values of f_1 for each preference category is 5 for P, 3 for I and 1 for F. We chose this values so that $P > I + F$ and F is better than not assigning a group.

The f_2 function assigns a value between 1 and 5 to the group occupation. The maximum value is achieved when the group occupation is exactly the same as the mean occupation of the groups, that is the number of students in a course divided by the number of groups that course has. To enroll in an empty group is always better than a full group, so full groups receive the minimum value. The shape of the f_2 function can be seen in fig 3.

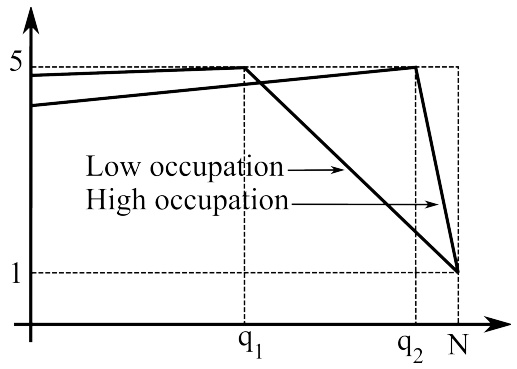

Fig. 3. Shape of the function that assigns a value to the occupation of a group. q_1 and q_2 are mean occupation of the course, N is the total number of students in the course. The slope of the function varies, depending on the number of students a course has.

This is the general version of the happiness function, during the execution of the system there are some variations. The first variation is that, to enforce the assignment of groups for the last year's courses (as required by the institution), when calculating the mean happiness this groups are valued double, we multiply the values of f_1 and f_2 by two. The other variation is that, when negotiating the students can ask once and again for the same group (we will see this later). To avoid this situation, each time a student cannot get a group in a negotiation, the happiness this group gives her will be reduced by a 10%.

The negotiation algorithm employed is a simplification of the Contract-Net Protocol [9]. The negotiation only takes place when a student wants to enter a group that is already full. In that case, the student interested in the group, let's call her the initiator, sends a list of groups she has already been assigned

to to the students in that group, whom we will call the receivers. The list of groups sent only includes the groups that, if changed for the group the initiator is interested in, doesn't decrease her current level of happiness. The first receiver interested in an offered group, that is, his happiness is not reduced, swaps places with the initiator. A pseudocode of the algorithm can be seen below.

```
offer := AssignedGroups(Initiator)
Filter(offer)

for all Receiver in Group do
    if Receiver is interested in $ER_x \in$ offer then
        Swap places
        return true
    end if
end for
return false
```

Before starting a negotiation, the student evaluates where would she be better. With the list of courses and groups she has assigned, she evaluates which group assignment can be improved. If changing one group for another can be done, that is, the new group has enough space, the student swaps there directly. If the new group is already full, the student starts a negotiation with the students enrolled in that group. If there's no one to negotiate with or there's no group we can offer because any change would decrease the happiness, a desist factor is applied to that group. The happiness that group contributes to the student happiness is reduced by a 10% and at some point the student will start asking for a different group, possibly in a different course.

To find a solution for the problem, we let this model evolve until an equilibrium state is achieved. This equilibrium state is the state where no student wants to swap places with another student. The students start with no groups assigned and have complete freedom to enroll any group that is not full. As we stated before, MARA let's the designer use terminology and ideas from the problem to be solved with little abstractions needed.

4 Experimental Setup and Results

Once the system was built it needed to be tested. This section will detail the datasets employed and how the system built responded. To deepen in the study, the system was compared with a traditional CSP approach, as described in [1].

4.1 Data Sets

Several data sets, with incremental constraint–based complexity, have been considered. They have been generated by using real statistical information from the Escuela Politcnica Superior at Universidad Autnoma de Madrid (UAM).

For each course, the number of laboratories available, students registered for each course, capacity of labs, and time tables, has been considered. This data has been used to generate a probability distribution of students/course and the number/capacity of labs that students need to attend. A four year degree has been considered (it corresponds to the current degree in Computer Engineering at UAM). Table 1 shows the student enrollment distributions by course (only those courses with laboratories are considered), the number of courses for which students have enrolled and the distribution of students with this number of courses.

Table 1. Distribution of labs by course.

Year	No. Courses	Percentage of Students
1^{st}	2	100%
2^{nd}	2 - 3 - 4	20% - 50% - 30%
3^{rd}	4 - 5 - 6	15% - 70% - 15%
4^{th}	4 - 5 - 6	15% - 70% - 15%

Table 1 assumes that students from any year has at least one course from that year and no courses from higher years. This means that a third year student has at least one third year course and no fourth year courses. This can also mean that a third year students can be enrolled in first and second year courses. To simplify the tests, we limited this so that a student can only have courses from one year and the immediatly below. The first and second year has a total number of 2 courses with practical sessions, we don't take into account theoretical–only courses, while third and fourth year have 5 courses with practical sessions.

As an example of how to read the table, the second row represents the second year students. This students can have 2, 3 or 4 courses, and some of them can be from the first year, up to 2 as there are only 2 courses in the first year. From the total number of second year students, a 20% have 2 courses, 50% have 3 courses and 30% have 4 courses.

As mentioned above, students can have courses from one year below the current year. In the second year, students with only 2 courses have both from second year, students with 3 have 1 course from first year and students with 4 have 2 from first year. In Table 2 the fraction of courses from another year are shown for the third and fourth years. The table shows the distribution of courses of different years for each student. By examining the real data, it can be noted that the number of students with at least one course from another year is higher than the number of students that have courses only of their own year. In addition, the number students with 6 courses of the same year are very low.

Based on the previous information, six data sets of 1000 students were generated. The synthetic restrictions for each student were also randomly generated by using some historic data related to previous years. Table 3 summarizes the basic features for each data set, where Ds0 considers only one Preferred group per ER and StudentAgent (e.g. $\langle\langle P, I, I\rangle, \langle P, I, I\rangle\rangle$), whereas Ds5 considers that

Table 2. Distribution of labs for 3^{rd} and 4^{th} year courses.

No. enrolled courses	All the same year (%)	One course from one year back (%)	Two courses from one year back (%)	Three courses from one (%)
4	20	60	20	0
5	10	40	50	0
6	5	40	40	15

30% of ERs are marked as Preferred and the rest (70%) are Forbidden. For example, $\langle\langle P, F, F\rangle, \langle P, F, F\rangle, \langle P, F, F\rangle\rangle$ makes a 30–70 distribution. The number of Forbidden and Preferred constraints has been adjusted along different datasets to cover different complexity situations.

Table 3. Student datasets.

Data Set	Preferred groups (P)	Forbidden groups (F)
Test	100%	0%
Ds0	1P/(RS,agent)	0%
Ds1	30%	0%
Ds2	1P/(RS,agent)	20%
Ds3	30%	20%
Ds4	1P/(RS,agent)	50%
Ds5	30%	50%
Ds6	1P/(RS,agent)	70%
Ds7	30%	70%

The difference between the percentage of P and F is the precentage of I. This way, the most restrictive datasets are DS6 and DS7 because the number of I is reduced and in some cases, many cases, there will be no I in the student preferences.

4.2 Results

Two systems were tested using these datasets, a CSP [1] and the multiagent system described in the previous section. Tables 4 and 5 show the results obtained for both systems. Since the happiness functions were obtained by using different preference values, they are not comparable across systems but they give a good estimation of their performance with different restrictions. The key values for the comparison are the percentage of P, I, and F in the final allocation.

As seen in Table 4, the CSP gives a good result for the first datasets, but when the restrictions are increased the overall happiness decreases and the percentage of F assigned is very high. Altough the execution time was not registered, it was comparatively high when comparing with the MARA approach. While MARA took only a few seconds, the CSP approach need hours to do the assignment.

Table 4. CSP experimental results for datasets considered.

Data Set	Mean Happiness	Happiness Deviation	Mean Distribution	Distribution Deviation	P's (%)	I's (%)	F's (%)
Test	9.85	0.07	81.5	0,66	100	0	0
Ds0	9.69	0.15	81.3	21.8	86.9	13.1	0
Ds1	9.78	0.1	81.2	15,2	94.2	5.8	0
Ds2	9.32	0.25	81	17.7	89.8	1.07	9.19
Ds5	7.82	1.09	80.6	17.9	61.3	15.6	23.1
Ds6	7.03	1.12	80.8	15.8	66	8.2	25.8
Ds7	6.02	1.59	80.9	19.5	71.9	0	28.1

For datasets 3 and 4, it took so long that it was stopped before completion, after 6 hours of execution.

Table 5. MAS experimental results for datasets considered.

Data Set	Mean Happiness	Happiness Deviation	Mean Distribution	Distribution Deviation	P's (%)	I's (%)	F's (%)
Test	9.85	0.07	81.5	0,66	100	0	0
Ds0	9.3	0.29	81.3	4.21	85.3	14.7	0
Ds1	9.7	0.14	81.4	0.64	94	6	0
Ds2	9.4	0.24	81.1	4.36	86.9	12.75	0.35
Ds3	9.8	0.12	80.4	2.04	95.8	3.67	0.49
Ds4	9.2	0.33	80.8	6.94	85.7	12.5	1.8
Ds5	9.7	0.19	81.0	0.89	95.5	2.42	2.05
Ds6	9.1	0.43	81.0	5.92	86.0	11.1	2.9
Ds7	9.7	0.29	80.9	0.91	95.8	0	4.2

The results obtained for the MAS model are shown in Table 5. This method is able, by using the negotiation-based approach, to maintain the global happiness of the solutions found. Although "happiness" values cannot be directly compared between CSP and MAS solutions (because their equations are different), the percentage of allocated F can be compared. In the worst situation (DS7) only the 4% of the students needs to be assigned to a forbidden ER (F), and the 96% of student teams are satisfactory allocated. Finally, the low variation of the happiness among the different datasets can be remarked compared to the variation of this value for the CSP solutions. This is due to the facility (given by the Multi-agent approach) to change preference values, or to exchange the current ER allocation with other agent in the system.

5 A Complexity Science View

Although there is no agreement on a definition of complex, we will use its etymological definition. The word comes from latin complexus: com- ("together")

and plectere ("to weave, braid"). If we refer to its etimology, something complex should have at least two elements and have some intricate relationship between its elements. Thus, a complex system should be a system with different elements and a mesh of relationships among them.

In this system we can identify two sources of complexity. On one hand we have the problem itself. The courses and groups are related to each other, and also are the students. The relationships between the different parts of the problem makes it a complex environment and if we introduce some change in it, e.g. change a group's time slot or the student preference, the reaction is unpredictable as we can't determine how other students will react or how the constraints will change during a simulation, among other things.

On the other hand we have the built system. As cyberneticists say, the best way to deal with a complex system is with another complex system. The built system is completely based on the problem, each part of it is a model of some aspect or aspects of the problem. This makes the built system to be adaptive and flexible: after a solution is found, we can translate any change in the original environment into the system and let it find a new solution. This adativity and flexibility is not available in other methods as, for example, backtracking. If we want to introduce a change in the problem, we would need to run the backtracking algorithm from the begining because some branches that where not useful before can lead to solutions now.

The built system is composed of different agents, each one of them can be differentiated from the others by the courses they have and the preferences over the groups in each course. The interaction between the agents is local, they can only communicate with a limited part of the system. At most, an agent can establish communication with other agents in the same courses it has, but normally they will communicate with only a subset of these agents. Although this communication is local, we have a global situation that the agents are not aware of.

The outcome of the system, the stable state, cannot be traced back to its agents. We can say that agent X enrolled in a group she can't attend, i.e. has an F for that group in her preference vector, because the group she wanted was full and no one there is interested in other groups she has. But then we need to go back and see why nobody is interested or why she couldn't enter the group in the first place because everyone had an equal chance to get into it. This causal chain is long, complicated and full of suppositions and, although we could find some probabilities over the links, we cannot fully explain why the resulting state is the one it is.

Complex self–organizing systems have feedback mechanisms. Feedback mechanisms defines how outputs are related to inputs, positive feedback increases the effect of an input while negative feedback reduces this effect. Before defining inputs and outputs we must define where are the borders of the system, in other words, what is part of the system and what is part of the environment. As we don't expect changes in the courses or groups, we define the system as the group

of agents and the environment as the courses and groups. This way we have that the problem defines an environment with the following variables:

- List of courses
- Number of groups for each course
- Size of the groups
- Groups' hours
- Number of agents
- Enrolled courses for each agent
- Preference vectors for each agent

This variables can be understood as how the agent models its environment and its goals, they belong to both, the environment and the system, but they are not fed into the system nor they are extracted from it. Previously we discussed the adaptivity of this kind of systems and how we could change the environment while the system is running. This stays true after this separation, what changes is how we introduce these modifications.

The rest of variables depends directly on the agents and they can modify them as they wish. From this variables we will define as an output, i.e. variables that the system shows to its environment, the assigned groups for each agent, and the only input would be the distribution of students in groups. To define the state of an agent we use the list assigned groups, the number of time it has attempted to enter a group and its happiness.

To recapitulate, we have that the problem is defined by variables that determine the courses and groups and their relationship with the agents. We can consider this variables as immutable and part of the agents' internal representation of the world. The list of groups assigned, the number of times an agent has attempted to enter a group and its happiness are the only variables necessary to determine the agents state. The system constantly receives information about the state of the groups, meaning that it knows what students are already in a group or if a group is full. In exchange, the system informs about changes it makes in the groups.

We can see clearly that there is a feedback loop, the state of the groups is fed to the agents and the agents inform about any changes in them. This loop doesn't determine by itself if it is a positive or negative feedback, it depends on how the system treats this information. In our case, this loop is managed in two ways: with the insistence factor and the f_2 function, both in the happiness equation.

The insistence factor is applied when the agents try to enter any group more than once without any success. The relation with the feedback is a little difficult to see, first the agent receives the status of the groups for the courses it has enrolled. The agent evaluates if there is a group that would increase its happiness and, if the group is full, it tries to negotiate with other student to enter. It is possible that the list of groups an agent receives doesn't change from one attempt to the next and the agent finds that the group that improves it happiness is the same full group each turn. To dissuade the agents, the insistence factor is applied.

This makes the feedback loop negative as it turns the output to be, after some time, the same as the input, whatever the input is, and the agents to stop trying to change groups.

For the other way to manage feedback, the f_2 function, it is clearer that it affects the feedback, but wether it does in a positive or negative way is hard to decide. When a group is below what was defined as the optimal occupancy this function encourages the agents to enroll this group. Once this quota is filled, the function dissuades agents from entering and sending them to other groups that are less full by reducing very fast the happiness this group can provide.

The value for the insistince factor and the shape of the f_2 function must be chosen carefully and with a goal in mind. This two mechanisms reduce the agent activity and drives the system to a stable state. The insistence factor is negative in nature as it forces the students to stop their activity. The f_2 can have a positive and negative effect, depending if the occupancy of the group is below the optimal or over it. This function also increases the variety of options, as agents does not only search for preferred group but also for groups that are not overcrowded.

As we can see, the built system is complex and self–organizes. This gives great flexibility when finding a solution, but makes the system hard to control. By introducing mechanisms to control the feedback we can ensure that the system will come to an equilibrium where the agents would stop searching, but a fine tuning of the system or proving that it will find the optimal solution are very difficult tasks if not impossible.

6 Conclusion

Scheduling problems are always difficult to solve. The elements of the problems usually have intrincated relations between them and usually, as in our case study, this relations makes the restrictions to vary during the process of solution. Traditional approaches as backtracking and related prove to be very accurate finding the solutions, but the cost in time and computional power is very high. Also, they are not easily comprehensible and cannot adapt to changes in the initial conditions.

The use of multiagent systems to solve this kind of problems overcomes these obstacles, as the agents can be simple computationally speaking and their behavior can be easily explained using the problem terminology. The concurrent and distributed nature of multiagent systems makes them very robust and adaptive.

The problem with the multiagent approach is the design process, as small perturbations in the agents behavior can result in very different behaviors of the system as a whole. One way to deal with this is to design the system as a model of some natural or social system that solves the same problem or a similar one. In this case, the agents are modelled as the students of the university that negotiate with each other for the groups they want.

The models considers three characteristics of the students: their preferences, the search for a group that is not empty nor full and the fatigue from trying

to enter a group without success. With these three characteristics included in the model, the results are very good in comparison with an implementation of a backtracking algorithm.

For a successful implementation of a system like the one presented, it is needed to understand the roles of positive and negative feedback. Positive feedback makes a system more controllable but unpredictable, because small inputs can render big changes. The system can be moved to another state, but the new state is not predictable. On the other hand, negative feedback makes the system uncontrollable but predictable as even big inputs cannot change the system state. A balance between these two feedback mechanisms can drive a system to self–organize.

To sum up, when trying to solve a complex problem it is useful to adopt ideas from natural and social systems that solve similar problems. The use of multiagent systems is highly recommendable as they can be, if well constructed, resilient and adaptive. Taking into account the interplay of positive and negative feedback in the design process is important for self–organization, giving independence to the agents and not considering this mechanisms can lead to disastrous outcomes as the system can never get to a stable state or get there too soon.

References

1. Cano, J.I., Sánchez, L., Camacho, D., Pulido, E., Anguiano, E.: Allocation of educational resources through happiness maximization. In: Proceedings of the 4th International Conference on Software and Data Technologies (2009)
2. Chevaleyre, Y., Dunne, P.E., Endriss, U., Lang, J., Lemaître, M., Maudet, N., Padget, J., Phelps, S., Rodríguez-Aguilar, J.A., Sousa, P.: Issues in multiagent resource allocation. Special Issue: Hot Topics in European Agent Research II Guest Editors: Andrea Omicini 30 (2006)
3. Choueiry, B., Faltings, B., Noubir, G.: Abstraction methods for resource allocation. Tech. Rep. TR-94/47, Dèpartement d'informatique, Institut d'informatique fondamentale IIF (Laboratoire d'intelligence artificielle LIA) (1994)
4. Gilbo, E.: Optimizing airport capacity utilization in air traffic flow management subject to constraints at arrival and departure fixes. IEEE Transactions on Control Systems Technology 5(5) (1997)
5. Jain, R., Chiu, D., Hawe, W.: A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. DEC Research Report TR-301 (1984)
6. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K.: Mason: A new multi-agent simulation toolkit. In: Proceedings of the 2004 SwarmFest Workshop (2004)
7. Modi, P., Jung, H., Shen, W., Tambe, M., Kulkarni, S.: A dynamic distributed constraint satisfaction approach to resource allocation. Lecture Notes In Computer Science 2239 (2001)
8. Railsback, S.F., Lytinen, S.L., Jackson, S.K.: Agent-based simulation platforms: review and development recommendations. Simulation 82(9) (2006)
9. Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Transactions on Computers (1980)