ALLOCATING EDUCATIONAL RESOURCES THROUGH HAPPINESS MAXIMIZATION AND TRADITIONAL CSP APPROACH

Juan I. Cano[1,2], Luis Sánchez[2], David Camacho[1], Estrella Pulido[1], Eloy Anguiano[1,2]
[1]*Computer Science Department*
[2] *Centro de Referencia Linux UAM–IBM*
Universidad Autónoma de Madrid, C/ Francisco Tomás y Valiente, nº 11, 28049, Madrid, Spain
inaki.cano@uam.es, luis.sanchez@uam.es, david.camacho@uam.es, estrella.pulido@uam.es, eloy.anguiano@uam.es

Keywords: Constraint-Satisfaction Problems, Simulation, Model-based agent optimization, Educational Resource Allocation, Multiagent Resource Allocation

Abstract: An instance of an Educational Resources Allocation (ERA) problem is the distribution of a set of students in different laboratories. This can be a complex and dynamic problem if non-quantitative considerations (i.e. how close the final allocation is to the student preferences or desires) are involved in the decision process. Traditionally, different approaches based on Constraint-Satisfaction techniques and Multi-agent negotiation have been applied to the general problem of Resource Allocation. This paper shows how a Multi-agent approach can be used to model and simulate the assignment of sets of students to several predefined laboratories, by using their preferences to guide the allocation process. This approach aims at finding new solutions that try to satisfy individual student needs with no knowledge about the general allocation problem. The paper shows some experimental results and a comparison, between a CSP-based solution modeled in CHOCO, a CSP Java-based library, and a Multi-agent model implemented using MASON, a multi-agent simulation platform.

1 INTRODUCTION

Efficient resource assignment, or resource allocation (Choueiry, 1994), is a complex problem (usually NP-hard) that has been studied in several research areas, such as Constraint-Satisfaction Problems (Modi et al., 2001; Tsang, 1993), Artificial Intelligence and Multi-Agent Systems (Chevaleyre et al., 2006; Dasgupta and Hoeing, 2007; Winstanley, 1993). Important application areas such as Industrial procurement, Satellites, Manufacturing Systems, Grid computing, Manpower planning/scheduling, or Educational management, have applied different Resource Allocation techniques to solve their particular problems (Dasgupta and Hoeing, 2007).

The main goal in a resource allocation process is to find either a feasible allocation, that is, some kind of assignment over the resources elements considered; or an optimal allocation. In the latter case, it is necessary to define some kind of metric to know when the optimum is reached, for example an allocation of resources that maximizes the average utility of several agents.

The concept of resource can vary over previous research areas. In this work, a resource is defined as a set of *non–sharable indivisible* items that can be distributed amongst several agents. Therefore, a particular distribution of resources amongst agents is called an *allocation*. For instance, in our case study we have a set of laboratories that need to be assigned to a set of student teams (made up of 1, 2 or more students). Students could prefer different timetables to others, and they can express their desires, or *preferences*, as a restriction. An assignment, or simply an allocation, is a distribution of the set of resources (laboratories) amongst the agents (student teams).

The focus of this paper is to design a system based on the real dynamics of student scheduling, that is how students apply for laboratory slots and how they interact (negotiate) with each other. This is done by measuring the student happiness, which is modelled as how far is the assignment from his preference, and maximizing this value. The distribution of students among groups is also taken into account, using this factor as a measure of how loaded is the resource. This model is compared with a traditional

CSP (monolithic) approach where each agent preference is defined as a new constraint. The MAS approach shows how easy it is to model the agent preferences (in form of preferred groups and non-assignable groups) and to look for a solution that tries to maximize their desires.

The paper is structured as follows: Section 2 describes the educational resources assignment problem and how it can be modelled as a constraint-based problem and a MAS negotiation problem; Sections 3 and 4 describes the CSP library (CHOCO) and the MAS platform (MASON) considered, and how the model has been implemented in both frameworks. Section 5 shows the data sets used in the experiments, and the experimental results obtained. Finally, Section 6 describes some conclusions and future lines of work.

2 ALLOCATION OF EDUCATIONAL RESOURCES

The problem of allocating educational resources can be briefly described as the process assigning resources to actors in a solution, or solutions, which optimize (or simply looks for a feasible solution) the available resources amongst those actors. For example, and as a case study, we could consider the problem of distributing students among different laboratories. In this case, the amount of computers, or other material, available in every laboratory sets the number of student teams allowed per laboratory. The difference between these resources (laboratory slots) lies on the number of places available and the slot timetable. Educational resources are grouped into resource sets and each student team must select their preferred laboratory slot among those available in a resource set, or even to indicate what slots are incompatible with their needs. Preference will be represented by three different values: "Preferred" (P), "Indifferent" (I) and "Forbidden" (F). These values are not fixed and can be adapted to other needs. In addition to student restrictions, it could be necessary to use other assignment policies (i.e. first group to make a request has the highest priority) to look for new alternatives. Figure 1 shows a schematic representation of this situation, where a student team has defined its preferences among three different lab slots corresponding to the resource set RS1. ER1.1 is `Preferred` over ER1.2 that is `Indifferent` for this student team. As the student team has indicated that ER1.3 should be avoided in the allocation process, this group is stored as `Forbidden`. This figure also represents the current allocation state for each lab slot in the resource set RS1.

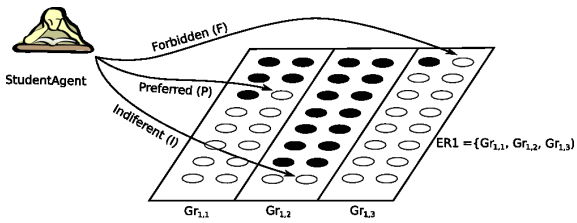

Figure 1: Current assignment for the resource set RS1 that contains three Educational Resources (ER1.1, ER1.2, ER1.3). The agent provides a set of restrictions, or constraints, to describe its preferences (Preferred-P, Indifferent-I, Forbidden-F).

As illustrated in Figure 2, the previous example can be extended to a more realistic and, therefore, complex situation, where two different types of constraints can be established. Horizontal constraints appear between educational resources belonging to the same educational set (solutions with lab slots with overlapping timetables are not allowed). Vertical constraints relate educational resources in different educational sets (this is a quite usual situation in engineering where a student can register for courses from different degree years). Why is it necessary to distinguish between horizontal and vertical constraints? The educational institution is usually responsible of guaranteeing the satisfaction of horizontal constraints. This is not the case with vertical constraints whose satisfaction is usually taken as a problem that must be solved by the student.

As Figure 2 shows the current needs of StudentAgent, could be described as a set of constraint vectors that need to be allocated. The final solution proposed by the algorithm will generate the "happiness", or how close the final allocation is to the StudentAgent preferences.

Constraint vector: $(\langle RS1 : \{P,I\}; RS2 : \{P,I,F\}; RS3 : \{F,P,F\}\rangle, \langle RS4 : \{F,P\}; RS5 : \{P,I,F,I,F,I\}\rangle)$

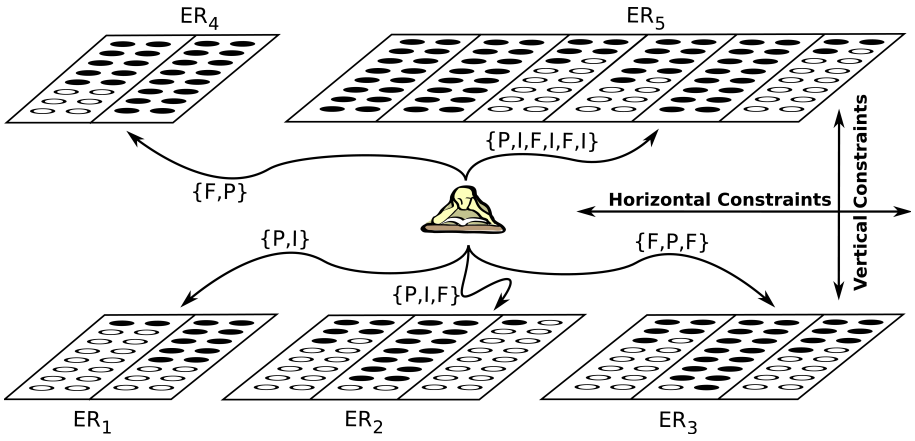

Figure 2: Horizontal and vertical constraints.

3 CONSTRAINT SATISFACTION-BASED MODEL

Constraint Satisfaction and Constraint Programming (Tsang, 1993) can be described as the process (including techniques, methodologies and algorithms) of finding a solution to a set of constraints that impose conditions that a set of variables must satisfy. A solution is, therefore, a set of value assignments to variables that satisfies all constraints. A Constraint Satisfaction Problem (CSP) uses constraints to enumerate the possible values a set of variables may take. This kind or problems can be formally described by using a triple $\langle X, D, C \rangle$, where X represents a set of variables, D is a set of domains that are related to variables (the domain of a variable contains all the values the variable can take), and C is a set of constraints that represents some kind of restriction imposed to the possible values assignments. A constraint is composed of a sequence of variables (or scope), and a set of values satisfying the constraint. It is possible to define a wide range of constraints, unaries ("$x_1 < 2.5$"), binaries that involve exactly two variables (i.e. "$x_1 - 3 \times 2 = 0$"), or n-aries ("$x_1 - 3x_2 + 4x_3 > 0$"). Variables can be assigned values from different domains such as Boolean, Integer, Real. A constraint is used to define some kind of property that the solution must satisfy. Constraints are usually embedded within a programming language (i.e. g-Prolog, Sictus-Prolog) or provided via separate software libraries (i.e. Choco, ILOG CP, JOPt, Koalog, etc...). The Choco Solver is a Java library for CSP designed to easily model CSP-based problems. It allows considering different kinds of variables, domains and constraint types, and provides several implementations of well known algorithms for constraint propagation and heuristic-based search (i.e. Arc Consistency algorithms). The library provides a complete API which helps users to model, implement, and test a CSP-based algorithm (Team, 2008).

Using the formalism and basics of CSP design, the problem described in Section 2, has been translated into the following CSP-based model:

- Resource Set and Educational Resources. The resources that finally are assigned (labs slots) define the possible values of the variables. As an example, in Figure 1, the RS1 generates three different domains (one for each ER), the number of slots available are used to set the domain cardinality.
- Student teams. Student team preferences are used to create both, the variable set that needs to be assigned, and a set of constraints that must be satisfied. In Figure 1, the StudentAgent is represented by variable "x_1" and the restriction $x1 \neq ER1.3$ represents that $ER1.3$ is a forbidden value for this team. Each student team has as many variables as ER to be allocated, and each RS generates a new set of constraints. Every variable can take a value, which represents a lab slot in a particular RS-ER that must satisfy the team constraints. Associated to each team a new variable is defined, the Optimization Variable (OV_i), which takes a different value depending on the relation ¡team preference, lab slot- ER assigned¿. Therefore, $OV_i = 5$ if lab slot assigned was a "P" (Preferred) value for the student team, $OV_i = 3$ if lab slot is "I" and $OV_i = 0$ if lab slot is "F". When two different RSs related to the same course are considered, the previously described values are increased in 2 units (so that Horizontal constraints will have a higher value, than Vertical constraints). These variables (OV_i) are used to find a solution, maximizing their global values (see equation 2).
- Two *incompatibles* ER (i.e. with the same timetable belonging to different RS) imposes a new restriction that makes that only one of them

can be assigned to a student team.

$$H = \sum_{i=1}^{k} H(a_i) \quad (1)$$

$$H(a_i) = \sum_{j=1}^{n} OV_i(RS_j(i)) \quad (2)$$

The CSP algorithm maximizes the global value of $OV_i(RS_j(i))$ for all the student teams considered, since it looks for the maximum value of H as shown in Equation 1, where k is the number of Students teams to be allocated. Equation 2 shows how to calculate the H value, where n represents the number or Resource Sets to be allocated.

Negotiation is not possible if the previous restrictions are used because the CSP approach is monolithic. A heuristic has been designed for those cases where there are a high number of very restrictive student teams to be allocated. If a solution is not feasible, this new heuristic incrementally selects those student teams with higher restrictions, and their Forbidden lab slots become Indifferent so their restrictions are not considered (those restrictions that do not affect horizontal constraints). Once a solution is found, these modifications appear as *"suggestions"* for a feasible allocation.

Finally, the value of OV_i variables can be used to see how close the solution found is to the student preferences. The higher the value, the 'happier' the students will be with the given assignment. The global value of gives us an idea about the 'global happiness' of OV the solution found.

4 THE AGENT-BASED MODEL

The main characteristics of ERA problem described in Section 2 make Multi-agent technologies a natural framework to test and prove these kinds of algorithms. For this reason there exist several multi-agent frameworks, such as NetLogo, Swarm, Repast or Mason (Railsback et al., 2006), designed to facilitate the specification of collective behaviors (represented as agents) that can be simulated into a simulation toolkit. Among existing MAS-based simulation toolkits, the Mason library was selected.

Mason is a discrete-event multi-agent simulation library core in Java, designed as a foundation for large custom-purpose Java simulations. This toolkit provides some graphical facilities, and other utilities to easily modify the simulation domain (Luke et al., 2004). A basic model in Mason requires the definition of two main elements; the environment and a set of agents with a defined behavior. For this reason, we have modeled an environment containing the Resource Set (RS) from which the agents can consume the available lab slots (ER). On the other hand, the student teams are modeled as Mason-based agents that evolve in each simulation step looking for a solution. One of these agents executes three different actions in each step. First, it updates its utility function, called *"happiness"*, then it looks for a new *"location"* that improves its happiness, and finally it tries to *"go"* there (the agent tries to obtain the selected ER). The maximization process can be described as follows:

1. The *happiness function* is a tradeoff between the particular happiness of the student team, and the current distribution of the available ER. This function (see equation 3) rewards an egalitarian distribution of student teams among RS and ER, and tries to satisfy the agent allocation preferences.

$$H(a_i) = \frac{\sum_{j=1}^{n}(f_1(RS_j(i)) + f_2(q\prime, RS_j(i)))}{n} \quad (3)$$

   $H(a_i)$ represents the average happiness of agent-i and is calculated by using $f_1(RS_j(i))$ the total number (n) of Resource Sets to be allocated, and the sum (for each ER) of how close the current allocation is to the agent preferences. This is calculated by using to transform the agent preference into an integer value (currently the values $\langle P = 5; I = 3; F = 0\rangle$ are used), and the level of occupancy ($q\prime$) of the ER (this is evaluated through function $f_2(q\prime, RS_j(i))$). This latter function has been introduced to quantify the occupation of the ER. The social reason is related to the problem that appears when a student team prefers attending a not overloaded lab group (they will receive more attention from educators). On the other hand, educators would prefer an egalitarian distribution of students (and therefore an egalitarian distribution of the teaching work). The f_2 function measures how close the ER occupation is to an egalitarian distribution.

2. Therefore, each agent in the system tries to maximize its own happiness function, so it will try to look for an ER (i.e. lab slot) that maximizes its preferences although a global egalitarian distribution is indirectly used by the agent. In the first simulation step any agent (randomly selected) can select only one ER from its Resource Set, to avoid initial and premature resource occupancy.

3. Once no slots are available in a particular ER, a negotiation procedure is executed by the agent which is interested in:

(a) The bidder agent (a_i) sends a message to all the current agents (a_j) assigned to the desired ER.

(b) .The a_i agent proposes to exchange one of its own ER (currently allocated) for this one. The list of available ERs is evaluated by each agent (a_j). Each a_j answers to a_i with their preferred ER from the exchange list (or sends a void value if it is not interested in any). If the bidder agent (a_i) is not interested in the proposal from a_j, it removes the ER from the exchange list and sends it back. This process continues until an exchange is made, or no agent is available for negotiation (no one is interested to give up this ER).

4. Finally, to avoid a cyclic negotiation process from an agent that tries to access a particular ER, a discouraging factor is used. Equation 4 shows how when an agent repeatedly tries to get an ER that it is already full and there is no negotiation possible (we are considering s negotiation cycles; $t \in [0 \cdots s]$), the initial ER preference value ($f_1(RS_j(i)) \equiv f_1^{(0)}(RS_j(i))$) is decreased by a constant factor (currently 0.75).

$$f_1^{(t)}(RS_j(i)) = 0.75^{t-1} \times f_1^{(0)}(RS_j(i)) \quad (4)$$

With this structure, the system is completely autonomous and there is no need for a coordinator or central structure (no auctioneer is used). Agents select ERs by themselves and try to maximize their preferences while keeping a balanced distribution of resources. If a group is completely full, and an agent is interested in it, a negotiation process with other agents is started in order to find an agreement. Any agent that tries to access into an ER insistently decreases the happiness that could gain by obtaining that ER (so that it is better for an agent to shift its preferences from preferred groups that are overloaded, to others with an initial lower interest). By using this approach, the system converges to a solution that is measurable not only by the preference satisfaction, but also by the social welfare and distribution of the resources.

5 EXPERIMENTAL RESULTS

This section describes how the experimental datasets have been generated to compare both algorithms, and the experimental results obtained.

5.1 Data Sets

Several data sets, with incremental constraint-based complexity, have been considered. They have been generated by using real statistical information from the Technical School 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. These data have 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 Bachelor in Computer Science 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: Student datasets

Year	No. courses	Distribution (%)
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 2: Distribution of labs by course.

A row in Table 1, for example the second one, shows that 2^{nd} year students can register for up to 2 first-year courses, up to 3 second-year courses, and up to 4 third year courses. The distribution shows that it is more common to register for courses of the year for which you have enrolled.

Based on the previous information, six data sets of 1000 student teams were generated. The synthetic restrictions for each team were also randomly generated by using some historic data related to previous years. Table 2 summarizes the basic features for each data set, where Ds0 considers only one Preferred group per ER and StudentAgent (i.e. $SA_i = \langle\langle P,I,I\rangle,\langle P,I,I\rangle\rangle$), whereas Ds5 considers that 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.

5.2 Results

The previously described datasets were generated as data files of student preferences[1], and translated, by following the CSP and MAS models described in sections 3 and 4, into an adequate representation. Tables

[1]Data sets available at: arantxa.ii.uam.es/~dcamacho/mason/mara.htm

Table 3: Student datasets

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

3 and 4 show the results obtained for both algorithms. 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.

As it could be expected, the CSP model provides good results when there is a feasible solution, this usually happens when the student teams are not very restrictive (Test, DS0 and DS1). However, when the number of restrictions is increased (forbidden ERs), we use a heuristic that penalize those teams with a higher number of restrictions, allowing to allocate teams to initially forbidden groups (these restrictions are relaxed; some ERs are translated from F to I). This allows finding a partial solution but not the optimum. As Table 3 shows, initial datasets can be allocated using only preferred and indifferent groups, however once the dataset includes a 20% of Forbidden ERs, the algorithm needs to allocate about the 9% in forbidden groups. This problem increases linearly with the increment of F groups (e.g. using a 70% of Forbidden ERs the solution allocates up to 28% of the teams to Forbidden ERs), therefore the global happiness of the solution decreases.

The results obtained for the MAS model are shown in Table 4. 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 Forbidden can be compared. In the worst situation (DS5) only the 4% of the students needs to be assigned to a Forbidden ER, and the 95% 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.

6 CONCLUSIONS AND FUTURE WORK

The experimental results obtained with our algorithm within the simulating platform MASON, and its comparison with a traditional CSP approach, show that the negotiation-based simulation algorithm allows us to dynamically find new solutions that maximize non-quantitative parameters, such as student happiness, i.e. how close the proposed solution is to the needs of specific students. These results show how a multi-agent (lightweight and collective) model can be used to optimize the allocation of a set of educational resources based on a set of distributed (negotiable) restrictions that represent the student team preferences.

In the near future some properties of our approach will be modified and tested, modifying the happiness function considered and incorporating other actors, such as the interests of educators in the allocation of the educational resources. On the other hand, the model proposed for ERA problems could easily be extended to consider a broker, or auctioneer agent, that could be designed to compare with other MAS classical algorithms (i.e. trading systems).

ACKNOWLEDGEMENTS

This work has been supported by research projects TIN2007-65989 and TIN2007-64718. We also thank IBM for its support to the Linux Reference Center.

REFERENCES

Chevaleyre, Y., Dunne, P., Endriss, U., Lang, J., Lemaitre, M., Maudet, N., Padget, J., Phelps, S., Rodrguez-Aguilar, J., and Sousa, P. (2006). Issues in multiagent resource allocation. *Informatica*, 30.

Choueiry, B. Y. (1994). *Abstraction methods for resource allocation*. PhD thesis, Dpartement d'informatique, Institut d'informatique fondamentale IIF (Laboratoire d'intelligence artificielle LIA).

Dasgupta, P. and Hoeing, M. (2007). Task selection in multi-agent swarms using adaptive bid auctions. In *Proceedings of the First International Conference on Self-Adaptive and Self-Organizing Systems*.

Luke, S., Cioffi-Revilla, C., Panait, L., and Sullivan, K. (2004). Mason: A new multi-agent simulation toolkit. In *Proceedings of the 2004 SwarmFest Workshop*.

Modi, P., Jung, H., Shen, W., Tambe, M., and Kulkarni, S. (2001). A dynamic distributed constraint satisfaction approach to resource allocation. In *Proceedings of the*

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
Ds3	7.82	1.09	80.6	17.9	61.3	15.6	23.1
Ds4	7.03	1.12	80.8	15.8	66	8.2	25.8
Ds5	6.02	1.59	80.9	19.5	71.9	0	28.1

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	9.29	0.29	81.3	4.21	85.3	14.7	0
Ds1	9.74	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	9.72	0.19	81	0.89	95.53	2.42	2,05
Ds4	9.09	0.43	81	5.92	86	11.1	2.9
Ds5	9.67	0.29	80.9	0.91	95.8	0	4.2

7th International Conference on Principles and Practice of Constraint Programming.

Railsback, S. F., Lytinen, S. L., and Jackson, S. K. (2006). Agent-based simulation platforms: review and development recommendations. *Simulation*, 82(9):609–623.

Team, T. C. (2008). Choco: an open source java constraint constraint programming library. In van Dongen, M., Lecoutre, C., and Roussel, O., editors, *Proceedings of the Third International CSP Solver Competition (CSP08)*.

Tsang, E. (1993). *Foundations of Constraint Satisfaction*. Academic Press.

Winstanley, G. (1993). Resource allocation in model-based planning systems. In *Proceedings of the IEE Colloquium entitled "Resource Scheduling for Large-Scale Planning Systems"*.