Object Oriented Modeling of Resource Assignment Problems formulated as CSPs

Luis G. Rueda, Ivanna Arias, Francisco S. Ibáñez and Raymundo Q. Forradellas
Computer Science Department, Computer Science Institute
National University of San Juan,
Cereceto y Meglioli (5400), San Juan, Argentina
e-mail: {lrueda, iarias, fibanez, kike}@iinfo.unsj.edu.ar

Abstract

Discrete combinatorial problems can be solved with Constraint Programming (CP) as long as they are formulated as Constraint Satisfaction Problems (CSP). In this paper we propose an object oriented model to solve combinatorial problems of resource assignment including applications in industry, commerce, and general organizations. Problems of these environments are those having entities that have to be assigned to places. A particular case of these problems is proposed and modeled here. This problem, named the *Classroom Problem*, is in a school with teachers; each teacher is to be assigned to any of the rooms of the school in different schedules and days. Besides there is a set of constraints limiting such assignment. The advantages with respect to other approaches which deal with a particular case of the problem considered in this work are identified and discussed.

1 Introduction

Real world applications, mainly in the resolution of discrete combinatorial problems, can be solved by using Constraint Programming (CP) [Van Hentenryck,89] [Leler,90] [Ibáñez,94] [Ibáñez,95] [Forradellas,95] [Rueda,95] [Rueda,96] [Rueda,97]. In this paper we propose an object oriented model in order to deal with combinatorial problems of resource assignment which are solved by using CP and object oriented programming [Puget,94] [Ilog,95].

A Constraint Satisfaction Problem (CSP) is formed by the following components: a set of variables, a set of domains associated to the variables, and a set of constraints relating the variables. A solution for a combinatiorial problem consists of the assignment of values (belonging to the associated domains) to the variables in such a way that all constraints must satisfy.

Examples of problems which fit in this model are:

- the distribution of the cashiers on the cash posts in a supermarket which works in different turns;
- teachers lecturing in the rooms of a school in certain days and schedules;
- the drivers of a transportation enterprise which has to be assigned to any of the buses in different schedules;
- a factory with machines requiring work with some resources at different times and conditions.

In general, we have entities that have to be assigned to places. In [Rueda,97] a particular case (the first mentioned above) with its correspondent implementation has been developed by using ILOG Solver [Ilog,95]. In this paper we propose a more general model to deal with combinatorial problems of resource assignment.

2 The Theoretical Model

2.1 CSP (Constraint Satisfaction Problem)

A CSP [Mackworth,86] is composed by:

1. a set of *variables*, $V_1,..., V_n$
2. a set of *domains* $D_1,..., D_n$ associated to the variables, and
3. a set of *constraints* relating the variables.

A solution is a set of values $a_1,...,a_n$, where $a_i \in D_i$ and the substitution a_i/V_i verifies all the constraints ($1 \le i \le n$)

Starting from a general CSP, we define a new model which deal with combinatorial problems of resource assignment.

2.1 The CSP-Space

Definition 1: A Resource Assignment CSP, called *CSP-Space*, consists of:

1. a set of *variables*, $Vi_1, ..., Vi_k$
   where $1 \leq i_1 \leq l_1$,
   ....
   $1 \leq i_k \leq l_k$.
2. a unique *domain D* associated to the variables, and
3. a set of *linear* *constraints* relating the variables.

A solution is a set of values $a_{i1},...a_{ik}$ where $a_{i1},...a_{ik} \in D$, and the substitution a_{i1}/Vi_1,..., a_{ik}/Vi_k verifies all the constraints ($1 \leq i_1 \leq l_1$, ..., $1 \leq i_k \leq l_k$).
In the ordinary CSP, the *variables* can be seen as points in a segment, while in the *CSP-Space* the *variables* can be seen as points (or *vectors)* of an *n-dimensional* space denoted as *S*.

Definition 2: A *group*, symbolized as *G*, is defined as a set of *variables* $Vi_1, ..., Vi_k$, where:

each i_j ($1 \leq j \leq k$), vary on $a_j \leq i_j \leq b_j$ with $1 \leq a_j$ and $b_j \leq l_j$. The pair $< a_j, b_j >$ represent the range of the j^{th} index of the variable $Vi_1, ..., Vi_k$.

Then, from the Definition 2 a constraint will be associated to one or several *groups*, depending on the constraint sorting as shown later. In the real world the *variables* would represent places to be taken. *D* represents the set of all possible *candidates* to occupy these places. From now on all elements of *D* will be referred as *candidates*.

2.2 Constraint Sorting

Given a *group* of *vectors G*, we will define constraints of different types . We will say that a constraint involving *vectors* and *candidates* has *weight* 0 if the *candidates* "must not" be assigned to the *vectors*, and *weight* 1 if the *candidates* "must" be assigned to the *vectors*. The *weight* is symbolized as *W*.
The following are the different types of constraints sorted from several combinatorial discrete problems of resource assignment:

Type 1: Many *candidates* may occupy a *group* of *vectors* with weight 0.

Type 2: Any *candidate*, which may occupy any *vector* of a *group* with weight 0 or 1, may occupy another *vector* of the same *group* with weight 0 or 1.

Type 3: In a *group*, the *candidate i* may occupy a *vector*, with weight 0 or 1, and the *candidate j* which may occupy another *vector* with weight 0 or 1.

Type 4: A candidate any occupy a *vector* of a *group* with weight 1.

Type 5: A *candidate*, which may occupy a *vector* of a *group* with weight 0 or 1, may occupy any *vector* of another *group* with weight 0 or 1.

The constraints listed previously are general for several problems of resource assignment. However, other constraints may exist which can be sorted and included in any of the types defined previously or a new type can be obtained and included in the model.

3 The Object Oriented Model

Starting from both the theoretical model of the *CSP-Space* and the constraint sorting defined previously, we build an object oriented model for the whole system. The components of the former and the constraint types of the latter are represented as classes, as shown in Figure 1, and the model is built based on the OMT [Rumbaugh,91]. This model describes the structure of the objects in the whole system: their identity, their relationships to other objects, and their attributes.

The operational model is not included in this work. We have to emphasize that the system has to be built around objects rather than functionality, because an object oriented model more closely corresponds to the real world and is consequently more resilient with respect to change.

It can be noted that the space, vectors, candidates, axis, and groups are represented as real world classes according to the concepts of the theoretical model. Each type of constraint is also a sub-class of a generic class *Constraint*. The types 1 and 4 are unified in an only type in order to because they share the same data.

4 Application to Real Problems

4.1. The Classroom Problem

4.1.1. The CSP-Space

Let us consider a School with *t* teachers. Each teacher is to be assigned to one of the *r* rooms in *s* different schedules and *d* different days. Besides a set of constraints limits such assignment.

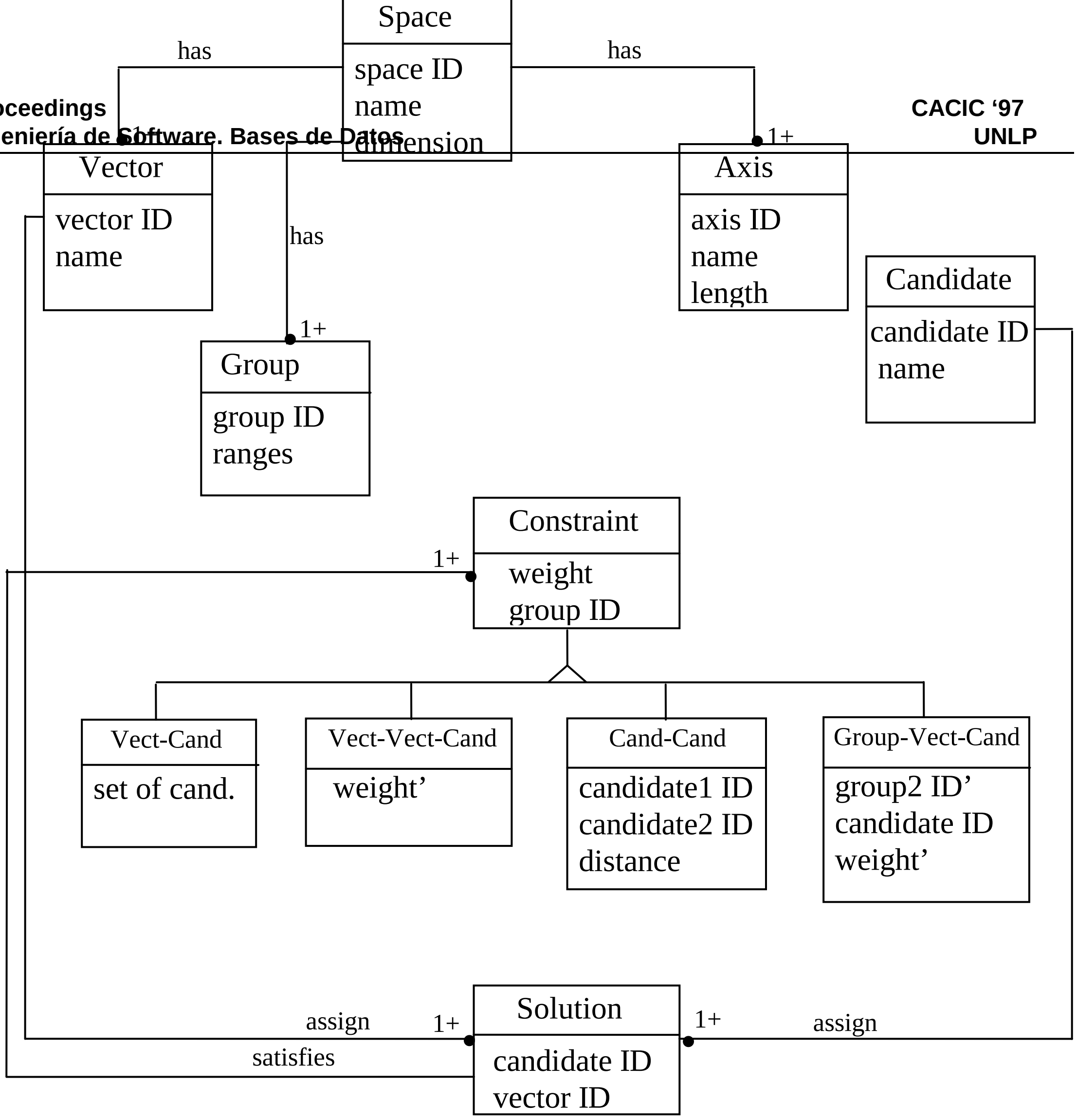

Figure 1: OMT for the CSP-Space

In order to model the problem as a *CSP-Space*, it is important to identify first the axis. The quantity of the axis determines the dimension of the *CSP-Space*.

As a general procedure, we propose here identify the most important entities of the problem. In this example, the most important entities are: *school, teachers, rooms, schedules,* and *days*. *School* corresponds to the *Space*, t*eachers* are the C*andidates*, and the remaining entities: *rooms, schedules,* and *days* will correspond to each of the *axis* of the *CSP-Space*.

Therefore, we can represent the problem by means of a the three-dimensional *CSP-Space* where each *vector* $V_{i,j,k}$ represents the i^{th} room available in the j^{th} schedule and the k^{th} day; and each *candidate* is a teacher.

Let us suppose that the *school* has five rooms working in seven hours and five days, as it is shown in the Figure 2.

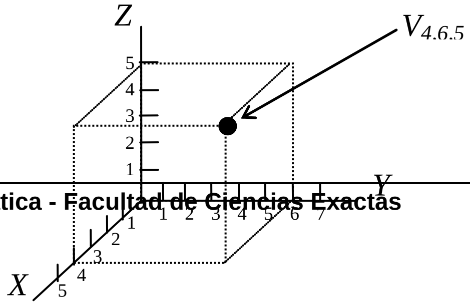

Figure 2: A three-dimensional CSP-Space for the classroom problem

X corresponds to the five rooms, *Y* corresponds to the seven hours, and *Z* corresponds to the five days. The seven hours the School works are, for example: 7:00-7:40, 7:45-8:25, 8:30-9:10, 9:15-9:55, 10:00-10:40, 10:45-11:25, and 11:30-12:10. The five days are: Monday to Friday. Then the vector $V_{4,6,5}$ (Figure 2) represents the second room in the sixth hour (from 10:45 to 11:25) in the third day (Wednesday).

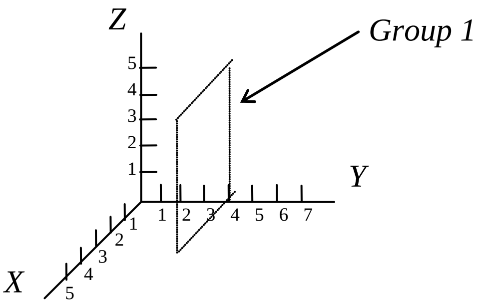

Figure 3: A group of the three-dimensional CSP-Space

The plane $Y = 4$ (*Group 1*) includes all the rooms in the fourth hour in the five days, as shown in Figure 3. The *group 1* is defined by means of the ranges $[1,l_1]$ $[4,4]$ $[1,l_3]$ (all the *vectors* $V_{i,j,k}$ with $j=4$, $i \in [1,l_1]$, and $k \in [1,l_3]$).

4.1.2. Constraints

Once the problem is fit in the *CSP-Space*, where groups are defined, several constraints may be represented and included as a type. The following enumerates an example of each type of constraint:

First type: The teachers number 3 and 5 must not work in any room in the fourth hour, in the five days. That is, c_3 and c_5 (with c_3 and $c_5 \in D$) must be assigned with *weight 0* to the *group 1*.

Second type: Any teacher which occupy a room in an hour and in a day can not occupy another room in the same hour and the same day.

Third type: The teacher number 6 must work next to the teacher number 4.

Fourth type: The teacher number 2 must work in the third room the fifth and sixth hours on Monday.

Fifth type: The teachers working in the third hour must rest in the fourth.

5 Conclusion

The advantages with respect to other approaches which deal with a particular case of the problem considered in this work [Fernández,96] [Rueda,96] are the following:

- A general model for several kinds of resource assignment problems in industry, commerce, etc. is stated where the constraints are common for different problems and the space is the same.
- The constraints are sorted as classes of objects and a lot of redundancy is eliminated, moreover new rules and specific searching algorithms can be defined in the CP environment.
- The different components of the *CSP-Space* are represented in an object oriented model taking all the advantages provided by this paradigm, and therefore, increasing the declarativeness.
- Starting from the object model proposed for general resource assignment problems, new classes may be defined, or the existent ones may be modified for a particular problem whenever it fits in the *CSP-Space*.

6 Future Extensions

A future extension of the present work, consist of considering *W* as a real number between 0 and 1, that is, a probabilistic value. Therefore, a candidate can occupy a resource with probability *W*. Then the problem could be thought as a discrete optimization problem [Smith,96] which has an additional variable representing the objective; each time a solution to the CSP is found, a new constraint is added to ensure that any future solution must have an improved value of the objective. This procedure continues until the problems becomes infeasible, and the last solution found is the optimal.

Another extension consists of considering the *candidates* and *vectors* as abstract objects instead of variables and domains respectively, so that many attributes may be added to that classes and the constraints relate not only the identification of both but also different attributes, and the model takes a true approximation to the real world.

7 Acknowledgments

We thank CICITCA for the support of research program framework. Besides we thank

ILOG SA Spain who provided us the development tools related to the CP environment.

8 References

[**Forradellas,95**] R. Forradellas, "Un modelo para el tratamiento de Sistemas Dinámicos con Restricciones en el marco CLP", Tesis Doctoral, Univ.Politecnica de Madrid, 1995.

[**Ibáñez,94**] F. Ibáñez, R. Berlanga, F. Barber y R. Forradellas, "Dos Enfoques de la Programación Lógica con Restricciones", Revista de Informática Teórica y Aplicada, Brasil 4/1994.

[**Ibáñez,95**] F. Ibáñez, "CLP (Temp): Integración de Restricciones Temporales Métricas y Simbólicas en el Marco CLP", Tesis Doctoral, Univ.Politécnica de Valencia, 1995.

[**Ilog,95**] "Ilog Solver - Reference Manual Versión 3.0", Ilog, France, 1995.

[**Jaffar,94**] J. Jaffar and M.J. Maher, " Constraint Logic Programming: A Survey", J. Logic Programming, to appear, 1994.

[**Leler,90**] V. Leler, "Constraints Programming Languages - Their Specification and Generation", Addison-Wesley Publishing Co. Inc., 1990

[**Macworth,86**] A.K.Mackworth, "Constraint Satisfaction", Encyclopedia of Artificial Intelligence, 1986.

[**Martin,94**] Martin J. y Odell J., "Análisis y Diseño Orientado a Objetos", Prentice Hall Hispanoamérica S.A., 1994.

[**Puget,94**] Puget J. "A C^{++} Implementation of CLP", Ilog Solver Collected Papers, Ilog tech. report, 1994.

[**Rueda,95**] L.Rueda, R.Klenzi, L.Gutierrez, F.Ibañez y R.Forradellas, "Tratamiento de Problemas de Combinatoria Discreta mediante el Paradigma CLP", 2das.Jornadas Informáticas, San Juan, 1995.

[**Rueda,96**] L. Rueda, I.Arias, F.Ibañez and R.Forradellas, "Representación y Tratamiento mediante POO de Problemas de Combinatoria Discreta en la Programación con Restricciones", 25as. Jornadas Argentinas de Informática e Investigación Operativa (25as. JAIIO), 1996.

[**Rueda,97**] L. Rueda, I. Arias, F. Ibáñez and R. Forradellas, "Tecnologías de Inteligencia Artificial aplicadas a Problemas Industriales Complejos", VII Reunión de Trabajo en Procesamiento de la Información y Control (VII RPIC), 1997.

[**Rumba,91**] Rumbaugh J, Blaha M., Premerlani W., Frederick E. and Lorensen W., "Object Oriented Modeling and Design", Prentice Hall, 1991.

[**Smith,96**] Smith B, Brailsford P., Hubbard P. and Williams P. "The Progressive Party Problem: Integer Linear Programming and Constraint Programming Compared", Ilog Solver Collected Papers, Ilog tech. report, 1996.

[**Van Hentenryck,89**] P. Van Hentenryck, "Constraints Satisfaction in Logic Programming", MIT Press, 1989.

[**Van Hentenryck,92**] P. Van Hentenryck, H. Simonis and M. Dincbas "Constraint Satisfaction using CLP", Artif. Intell., Vol. 58, 1992.