The Problem of Assigning Evaluators to the Articles Submitted in an Academic Event: A Practical Solution Incorporating Constraint Programming and Heuristics*

B. Jesús Aranda, F. Juan Francisco Díaz, and V. James Ortiz

Universidad Del Valle, Escuela de Ingeniera de Sistemas y Computacin,
Ciudad Universitaria - Melendez
{jesarana, jdiaz, jaortiz}@univalle.edu.co

Abstract. This article shows a practical solution to *The Problem of Assigning Evaluators to the Articles Submitted in an Academic Event*, a problem of combinatorial optimization. Apart from stating the problem formally and proposing a constraint model, the article describes the heuristics designed to find solutions. The application was developed using *Mozart*; different distribution strategies were implemented based on the already mentioned heuristics. The experimental partial results turned out to be competitive for real problems (180 articles, 25 evaluators).

1 Introduction

An academic event or congress consists of a series of conferences in which different research works or articles, previously referenced and selected by a Program Committee, are presented. These articles may cover different research areas, but they should be related to the main topic of the congress. Each representative of the Program Committee relies on a work group for the process of evaluating those articles.

The Program Committee receives the articles, and according to certain criteria including the strengths of the evaluators and the topics of the articles among others, assigns them for evaluation so that each article is evaluated by the maximum number of evaluators determined by the organizers of the event.

The quality of the solution (understood as the adaptation of it to all the distribution criteria) and the time taken to estimate it are critical aspects of the process. The first one minimizes the task of reassigning evaluators when they are not satisfied with the article assigned. The latter makes things easier for the Program Committee since it depends on time to test different solutions and choose the best one in quality. In the case of the organization of the Conferencia Latinoamericana de Informática (Latin American computing conference) of the

* This work is supported in part by grant 298-2002 from Colciencias.

CLEI [1], this process can take approximately three days and the solution found produces dissatisfaction among evaluators.

Therefore, it is interesting to have an application that allows us to find a solution that maximizes its quality (that is, a solution that minimizes the number of inconsistencies between assignation criteria and the real solution) as quickly as possible.

Based on what was previously stated, the design and construction of such an application was proposed. This application was developed using the Constraint Programming paradigm and the **Oz** language. As expected, the application of general mechanisms to find solutions was not very useful in real size entries. For that reason, problem-dependent mechanisms were designed and implemented. More specifically, different distribution strategies based on heuristics designed especially for the problem were implemented.

This article presents a formal description of the problem (Section 2), the most appropriate constraint model that we have found up to now (Section 3) and the heuristics designed to orientate the distribution strategies (Section 4). Finally, the results obtained (Section 5), a global description of the application architecture (Section 6) and the conclusions (Section 7) are presented.

2 Description of the Problem

According to the organizers of the **CLEI**, the process of distributing articles takes from 3 to 4 days. This is true for an entry of approximately 300 articles, 80 evaluators and 3 evaluations per article. The greatest difficulty lies in assigning enough evaluators to each of the articles complying with certain constraints the event involves (those constraints will be described later.) Most of the times, the article distribution results in many article assignations that do not comply with the constraints.

Considering that, the organization of the **CLEI** 2005 event, which will be held in Colombia in 2005, needs an application that supports the article distribution process, aiming to reduce the time the process takes and minimize the number of assignations that do not comply with the constraints set for the event.

The entry for the article distribution process in the academic event of the **CLEI** consists of the following data:

- **A set of articles or works:** The quantity of articles sent to an event as **CLEI** 2005 is approximately 300 from which the articles that will be presented in the conferences are selected.
- **The number of evaluations per article:** An article is reviewed by 3 evaluators; therefore, if there are 300 articles for the event, 900 evaluations would be necessary, and they would be done by the evaluators of the program committee.
- **A group of evaluators:** They are the program committee. Events as **CLEI** 2005 usually have, more or less, 80 members.

[1] Centro Latinoamericano de Estudios en Informática.

- **A set of constraints:** They are the requirements that should be met in any assignation of the articles received.

The constraints that should be taken into account when distributing the articles are:

- **Constraint 1** : The number of evaluations per article must be higher than or the same as the minimum required.
- **Constraint 2**: The number of evaluations per article must be less than or the same as the maximum required.
- **Constraint 3**: The number of articles assigned to each evaluator must be less than or the same as his capacity.
- **Constraint 4**: For each article, each of the evaluations should be done by a different evaluator.
- **Constraint 5**: The article's country must be different from the evaluator's country.
- **Constraint 6**: At least one of the main topics of the paper must coincide with a preferred topic stated by each one of the assigned evaluators.
- **Constraint 7**: The language of the paper must coincide with one of the languages each assigned evaluator masters.

The idea is minimizing the number of assignments that do not comply with the previous constraints during the assignment process. When it is not possible to assign an article complying with all the preferences, it is assigned considering the most important constraints, complying just with some of them. The organizers of the event may also consider that some preferences are mandatory and, therefore, a total distribution may not be achieved. In that case, the missing assignments are analyzed by the organizers of the event in order to find a solution.

Our application considers the factors mentioned, providing a solution to the distribution process.

3 Constraint Model of the Problem

The model developed to solve the problem is presented below.

- **Parameters**
  - m : Number of evaluators.
  - n : Number of articles
  - nT : Number of topics of the event
  - $minEP$: Minimum number of evaluators per article.
  - $maxEP$: Maximum number of evaluators per article.
  - cP_i : The country of article i, $\forall i = 1, \ldots, n$.
  - sP_i : The set of topics of article i, $\forall i = 1, \ldots, n$.
  - lP_i : The language in which the article i, is written $\forall i = 1, \ldots, n$.
  - cE_j : The country of evaluator j, $\forall j = 1, \ldots, m$.
  - sE_j : The set of topics the evaluator j masters, $\forall j = 1, \ldots, m$.

  - lE_j : The set of languages in which evaluator j is willing to evaluate, $\forall j = 1, \ldots, m$.
  - $capE_j$: The number of evaluations evaluator j is willing to do, $\forall j = 1, \ldots, m$.

- **Decision variables**
  - $dom_{i,k} = \begin{cases} j & \text{if the } k\text{-th evaluation of the article } i \text{ is assigned to evaluator } j, \\ 0 & \text{if no evaluator could be assigned} \end{cases}$ $\forall i = 1, \ldots, n,\ k = 1, \ldots, maxEP$
  - c : Total number of evaluations assigned, in which $n * minEP \leq c \leq n * maxEP$.

- **Objective function**
  - $Maximizing\ c = |\{(i,k) : dom_{i,k} \neq 0, 1 \leq i \leq n, 1 \leq k \leq maxEP\}|$

- **Constraints**
  - Constraint 1:
  $$minEP \leq |\{(i,k) : dom_{i,k} \neq 0, 1 \leq k \leq maxEP\}|, \quad \forall i = 1, \ldots, n$$
  - Constraint 2:
  $$maxEP \geq |\{(i,k) : dom_{i,k} \neq 0, 1 \leq k \leq maxEP\}|, \quad \forall i = 1, \ldots, n$$
  - Constraint 3:
  $$capE_j \geq |\{(i,k); dom_{i,k} = j, 1 \leq i \leq n, 1 \leq k \leq maxEP\}|,$$
  $$\forall j = 1, \ldots, m$$
  - Constraint 4:
  $$(dom_{i,k_1} = dom_{i,k_2} = 0) \vee (dom_{i,k_1} > dom_{i,k_2}),$$
  $$\forall i = 1, \ldots, n, \quad \forall k_1, k_2 : 1 \leq k_1 < k_2 \leq maxEP$$
  - Constraint 5:
  $$\forall i = 1, \ldots, n \ \ \forall j = 1, \ldots, m$$
  $$[cP_i = cE_j \rightarrow (\ \forall 1 \leq k \leq maxEP : dom_{i,k} \neq j)]$$
  - Constraint 6:
  $$\forall i = 1, \ldots, n \ \ \forall j = 1, \ldots, m$$
  $$[sP_i \cap sE_j = \emptyset \rightarrow (\forall 1 \leq k \leq maxEP : dom_{i,k} \neq j)]$$
  - Constraint 7:
  $$\forall i = 1, \ldots, n \ \ \forall j = 1, \ldots, m$$
  $$[lP_i \notin lE_j \rightarrow (\forall 1 \leq k \leq maxEP : dom_{i,k} \neq j)]$$

4 Heuristics Designed to Orientate the Distribution Strategies

One of the most important factors that influences efficiency on the application is the distribution strategy. Usually, the distribution strategy is defined based on a variable sequence. When distribution is needed, the strategy selects one of the non-determined variables present in the sequence and distributes based on that variable.

Distribution strategies can be classified as follows:

– **Generic distribution strategies:** These are general strategies that do not depend on the problem and are defined in **Mozart** programming system. Some of them are: **first-fail** and **naive**.
– **Problem-specific distribution strategies:** In these strategies the programmer sets criteria to select the variables that will be distributed and their corresponding value. The criteria used in the strategy depend on the characteristics of the problem in order to speed up the distribution process.

In the development of our application, we used generic and problem-specific strategies. The results obtained with both kinds of strategies are presented later.

Problem-specific distribution strategies were implemented based on the following heuristics.

4.1 Heuristics in Variable Selection

In our problem, the variables to be distributed are the *article evaluations* ($dom_{i,k}$) and their values correspond to the assigned evaluator. In a solution, we expect all the variables to have an assigned value.

The selection of variables to be distributed was made using two heuristic functions:

– Heuristic function based on the *comfort* of the evaluators
– Heuristic function based on the *topics* of the article

Using those functions we expect to have an indicator of the difficulty to assign a value to each variable. With this information, the most difficult variable is selected.

Heuristic Function Based on the Comfort of the Evaluators. The comfort of the evaluator regarding a partial assignment is defined as the number of evaluations that he may still be assigned. The comfort of each evaluator is calculated dynamically as the difference between the maximum number of evaluations that he is able to do (cME_j) and the number of evaluations assigned in the partial assignment (cUE_j):

$$hEvaluator_j = cME_j - cUE_j,$$
$$\forall j = 1, \ldots, m.$$

Based on that, the *hComfort* function is defined for each article as the addition of comforts of each of the possible evaluators of the article:

$$hComfort_i = \sum_{j:sP_i \cap sE_j \neq \emptyset} hEvaluator_j,$$
$$lP_i \in lE_j,$$
$$cP_i \neq cE_j,$$
$$j \neq dom_{i,k},$$
$$\forall k = 1, \ldots maxEP,$$
$$\forall i = 1 \ldots n,$$

Based on the values obtained by applying the heuristic function for each article, we choose to distribute one of the $dom_{i,k}$ variables of article i with the lowest $hComfort_i$ value. Intuitively, that means that we choose to distribute a variable representing that paper posing the greatest difficulty for finding a suitable reviewer.

Heuristic Function Based on the Topics of the Article. Again, the idea here is "calculating" the difficulty of evaluating an article. In this case, the heuristics that calculates the difficulty of evaluating article i is directly related to the main topics of the article.

Given any topic, t, its competitiveness ($cSubject_t$) is defined regarding a partial assignation of evaluators, as the difference between the remaining evaluators capacity for topic t ($oSubject_t$) and the number of article evaluations containing topic t that are still to be assigned ($dSubject_t$).

More exactly,

$$cSubject_t = oSubject_t - dSubject_t,$$

in which

$$oSubject_t = \sum_{j:1 \leq j \leq m, \{t\} \cap Subjects(Evaluator_j) \neq \emptyset} cME_j - cUE_j,$$

and

$$dSubject_t = \sum_{i:1 \leq j \leq n, \{t\} \cap Subjects(Article_i) \neq \emptyset} eN_i,$$

being eN_i the number of evaluations of article i still to be assigned (according to the current partial assignation).

The idea here is "estimating" the difficulty of evaluating an article($hSubject_i$). In this case, the heuristics that calculates the difficulty of evaluating article i is directly related to the main topics of the article:

$$hSubject_i = \sum_{t \in Subjects(Article_i)} cSubject_t,$$

Intuitively, $hSubject_i$ measures what our capacity to assign a reviewer to paper i is, given its main topics. We choose the variable with the lowest $hSubject_i$ value.

4.2 Heuristics When Choosing the Variable Value

Suppose that $dom_{i,k}$ is the variable chosen for distribution. And let $\{j_1, j_2, \ldots, j_{m_i}\}$ be the set of possible evaluators for article i. The j_l value that is first chosen for the $dom_{i,k}$ variable is the one that corresponds to the evaluator with more comfort in that moment:

$$hEvaluator_{j_l} \geq hEvaluator_{j_s} \forall s = 1, \ldots, m_i.$$

5 Results Obtained

Below we present the results obtained for problems of different sizes and using both models, but always the same set of data (taken from **CLEI** 1996).

In table 1, for each instance of the problem and for each distribution strategy, we show times (in seconds) obtained searching the first partial solution (using **searchOne**), number of non assignments **nns** (that is, the number of evaluations without an evaluator assigned at the end of the running) and its rate with respect to the total number of assignments required **nra** ($nra = n * maxEP$). Columns **ffs**,**hhbs**,**chbs** show results for the standard strategy (first-fail) and the two strategies that use heuristics based on the comfort of the evaluator and the heuristics based on the topics of the article, to select the variables. In both heuristic strategies the same function is used (based on the comfort of the evaluators) to choose the value of the variable to be distributed.

As it can be seen in the table above, the application performance in terms of efficiency in time and quality of the solution is better when using distribution strategies with heuristics.

However, it can not be said that one of the heuristics strategies is better than the other. And in terms of optimal solutions, it can not also be said that the solution found is always the optimum.

Table 1. Solution obtained for CLEI96 problem

Input Size		**ffs**			**hhbs**			**chbs**			**nra**
n	**m**	**time**	**nns**	**%nns**	**time**	**nns**	**%nns**	**time**	**nns**	**%nns**	
50	10	1	32	21.33	1	32	21.33	1.5	32	21.33	150
90	12	3	37	13.7	2	34	12.59	4	34	12.59	270
90	25	3	9	3.33	3	9	3.33	5	9	3.33	270
100	15	5	44	14.66	4	26	8.66	6	26	8.66	300
150	20	9	47	10.44	9	23	5.11	11	25	5.55	450
180	12	14	225	41.66	10	224	41.48	13	224	41.48	540
180	20	26	77	14.26	22	63	11.66	18	54	10	540
180	25	40	37	6.85	30	21	3.88	31	19	3.51	540

6 The Application

The name of the application created is CREAR. It is based on the model and strategies described above. Its architecture can be seen in Figure 1.

Three levels can be observed there:

– Presentation Level: It includes all functionalities that allow interaction between the program and the user.
– Application Level: It includes the program control and the main functionalities of the tool.
– Persistence or Storage Level: It includes the input and output files and the functionalities that allow their communication with the program.

In developing CREAR, besides **Oz** language, **Java** programming language was used, mainly at the presentation level. The description of each level is presented below.

Presentation Level. It includes the input reading and the presentation of results. It has the following modules:

– Data Capture Module: It includes the functionalities that allow to select input data, decide where to store output data, the constraints to be applied and the strategies that will be used to find the solution.

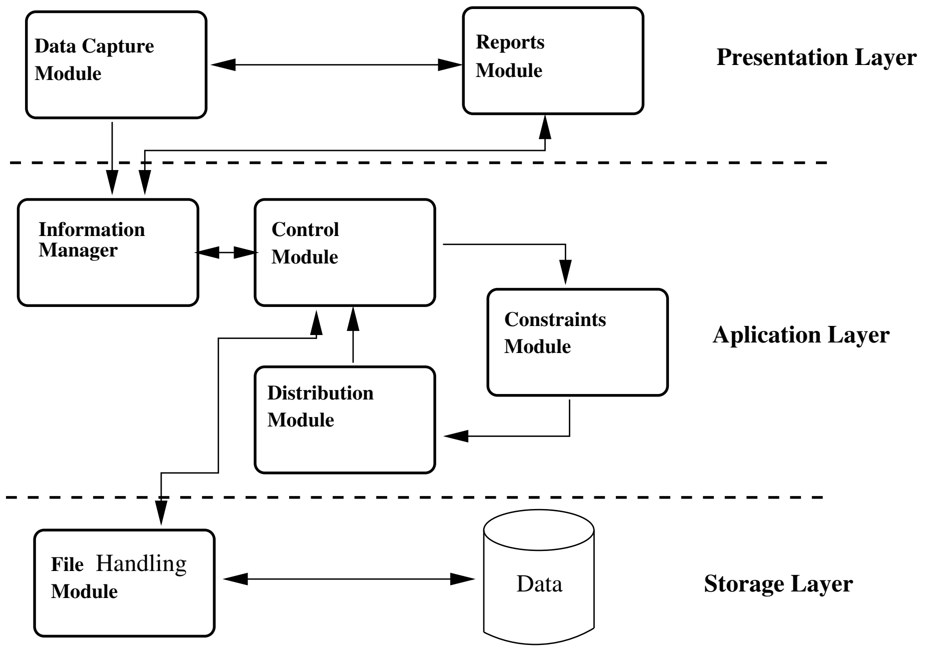


Fig. 1. Architecture of CREAR

– Report Module: It includes the functionalities that allow the program to show reports with statistics and interesting data of the solution found. These reports are fundamental for the organization of the event since, based on them, a final analysis which aims to lead to a better solution is done.

Application Level. It is here where solutions are to be found. It includes the following modules:

– Information Agent: It includes the functionalities that allow the communication between the interface and the driving force of the application so that they can work jointly.
– Control Module: It includes the functionalities that allow to ensure the system's integrity basically in the process of finding a solution.
– Distribution Module: It includes the different strategies that can be used to find the solution.
– Constraints Module: It includes all constraints to be considered.

Persistence Level. At this level, input and output data are stored.

– File Handling Module: It includes the functionalities that allow to read the input data of the problem and create the file with the solution.

6.1 Flexibility of Constraints

One of the most important characteristics of the application is the flexibility for imposing constraints.

Since in the problem entries are naturally over-constrained, the system lets the user choose the constraints he wishes to apply. The constraints that the user can choose are the following:

– Capacity Constraint: The number of articles to be evaluated must not exceed the capacity of the evaluator.
– Language Constraint: The language of the article must be one of the languages mastered by the evaluator.
– Country Constraint: The country of the main author of the article and of the evaluator must not be the same.

It should be observed that topic constraint is not optional since it is not convenient that an evaluator reviews a topic that he does not master.

6.2 Step by Step Solution

Another distinctive characteristic of the application lies in the possibility to find an incremental solution which disqualifies constraints as the solution is coming near.

To do that, the system allows to handle up to four steps as follows:

1. The system searches for a solution that meets the constraints chosen by the user. At the end, there may be still evaluations without an evaluator assigned.
2. The system considers those assignations that could not be done in the previous step and tries to do them taking into account the constraints chosen at that moment; generally these constraints are less than the ones in the previous step. At the end, there may be still evaluations without an evaluator assigned. This step can be repeated up to three times.

At the end of all steps it is possible that there may be not enough evaluations assigned to some of the articles; however, the system makes this number to be fairly reduced.

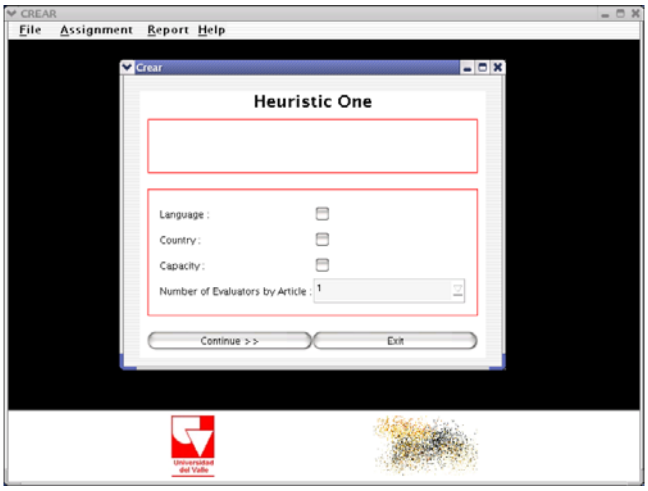
It is important to underline that the user may choose the number of steps and the constraints to be taken into account in each of them.

6.3 The Interface

The interface offers great flexibility to the user. It allows him to chose the constraints, steps, and the strategies to find the solution.

First of all, the interface lets the user determine the file with the input data of the problem and where the solution is to be stored. Based on what was previously said, the user can choose the strategy he wants to use to find the solution among the strategies described in Section 4.

After that, the system asks which constraints are to be applied (see Figure 2). Once the user has chosen them, he has to determine how many later stages will be tried (maximum 3), and which constraints are to be considered in each of them with a similar interface.



Heuristic One

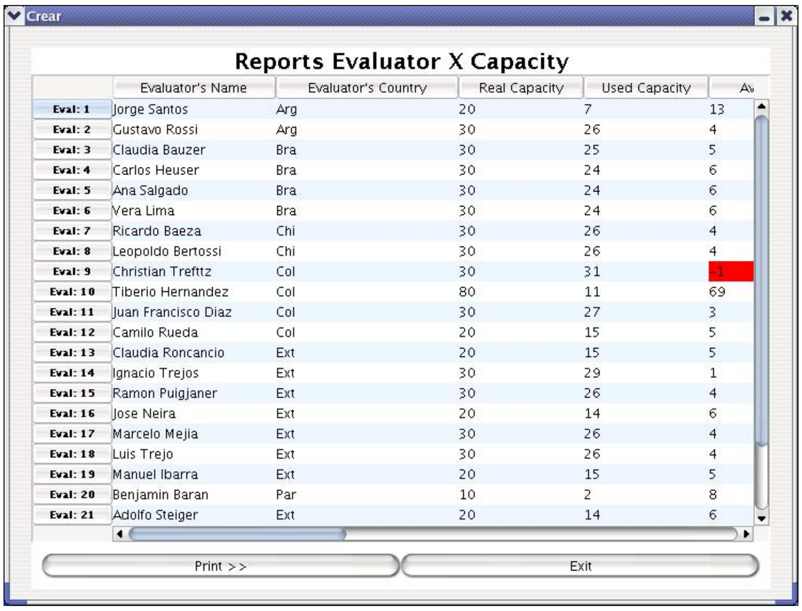
Language :

Country :

Capacity :

Number of Evaluators by Article : 1

Fig. 2. Interface to choose constraints



Reports Evaluator X Capacity

	Evaluator's Name	Evaluator's Country	Real Capacity	Used Capacity	Av
Eval: 1	Jorge Santos	Arg	20	7	13
Eval: 2	Gustavo Rossi	Arg	30	26	4
Eval: 3	Claudia Bauzer	Bra	30	25	5
Eval: 4	Carlos Heuser	Bra	30	24	6
Eval: 5	Ana Salgado	Bra	30	24	6
Eval: 6	Vera Lima	Bra	30	24	6
Eval: 7	Ricardo Baeza	Chi	30	26	4
Eval: 8	Leopoldo Bertossi	Chi	30	26	4
Eval: 9	Christian Trefttz	Col	30	31	-1
Eval: 10	Tiberio Hernandez	Col	80	11	69
Eval: 11	Juan Francisco Diaz	Col	30	27	3
Eval: 12	Camilo Rueda	Col	20	15	5
Eval: 13	Claudia Roncancio	Ext	20	15	5
Eval: 14	Ignacio Trejos	Ext	30	29	1
Eval: 15	Ramon Puigjaner	Ext	30	26	4
Eval: 16	Jose Neira	Ext	20	14	6
Eval: 17	Marcelo Mejia	Ext	30	26	4
Eval: 18	Luis Trejo	Ext	30	26	4
Eval: 19	Manuel Ibarra	Ext	20	15	5
Eval: 20	Benjamin Baran	Par	10	2	8
Eval: 21	Adolfo Steiger	Ext	20	14	6

Fig. 3. Report 1

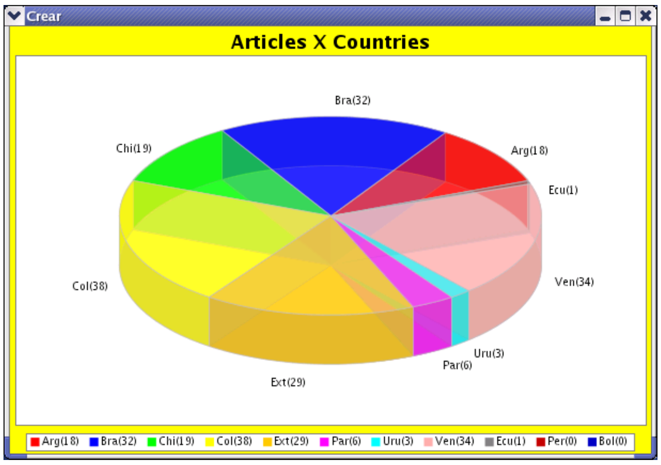


Fig. 4. Report 2

Once the solution is calculated, the system offers different options of reports, which are shown in Figure 3 and 4.

7 Conclusions and Further Works

This work shows the expressiveness of the CCP paradigm and the versatility of a programming language like **Oz** in developing applications that solve combinatorial optimization problems. Particularly, the ability to handle flexible constraints and to handle partial values was very important for building a flexible application with the ability to handle over-constrained problems and to perform iterative refinements of potential solutions.

Defining specific heuristics for the problem and the implementation of distribution strategies based on them are an important contribution of this work. The performance of the application in real problems was clearly superior using these strategies rather than generic distribution strategies.

One of the fundamental aspects in using CCP applications is the ability of the application to give an answer even when the entry is over-constrained. In this case, other important characteristics of our application were (1) modeling the problem as an optimization problem of just a CSP, (2) the flexibility in imposing constraints, and (3) the possibility of using constraints to increase the partially found solution.

Anyway, the possibility of using first class constraints is a characteristic that would give **Oz** more flexibility when facing over-restricted entries.

In a further work, we expect to integrate the application in current support systems like **WIMPE** [6] and **OpenConf** [10].

References

1. C. Castro and S. Manzano. (2001) Variable and value ordering when solving balance academics curriculum problem. In: Proc. of the ERCIM WG on constraints.
2. Juan F. Daz and Camilo Rueda. (2001) VISIR: Software de soporte para la toma de decisiones de vertimiento de agua en la represa del alto anchicay usando programacin concurrente por restricciones. Ingeniera y Competitividad. Vol 3, No. 2. Universidad del Valle, Cali (Colombia)
3. M. Henz and M. Muller. (1995) Programming in Oz. In G. Smolka and R. Treinen, Editors, DFKI Oz, Documentation Series. Mozart Documentation
4. Brahim Hnich, Zeynep Kiziltan and Toby Walsh.(2002) Modelling a Balanced Academic Curriculum Problem. In: Proceedings of CP-AI-OR-2002.
5. K. Marriott and P. J. Stuckey. (1998) Programming with Constraints: An Introduction. MIT Press, Cambridge, Mass
6. David M. Nicol. (2001) WIMPE: Web Interface for Managing Programs Electronically. http://www.crhc.uiuc.edu/ nicol//wimpe/wimpe6.1.html
7. C.Rueda and J.F. Daz and L.O. Quesada and C. Garca and S. Cetina. (2002) PATHOS: Object Oriented concurrent constraint timetabling for real world cases. In Proceedings XXVIII Conferencia Latinoamericana de Informtica, Montevideo, Uruguay
8. Christian Schulte and Gert Smolka. (2004) Finite Domain Constraint Programming in OZ. A Tutorial. Mozart Documentation
9. Peter Van Roy and Seif Haridi. (2004) Concepts, Techniques, and Models of Computer Programming. MIT Press
10. Zacon Group. (2004) OpenConf-Conference Manual Management System. http://www.OpenConf.org