Crear: Consejero para la Repetición de Artículos y Evaluadores en Eventos Académicos

Ortiz Vega, James Jerson; Diaz, Juan Francisco; Aranda, Jesus

Published in:
Ingeniería Y Competitividad

Publication date:
2004

Document Version
le PDF de l'éditeur

CREAR: Consejero para la Repartición de Artículos y Evaluadores en Eventos Académicos

Jesús A. Aranda*
Juan F. Díaz**
James J. Ortíz***

* Ingeniero de Sistemas - Estudiante del Programa de Doctorado en Ingeniería, Área de Énfasis: Ingeniería de Sistemas y Computación - Universidad del Valle - Santiago de Cali, Colombia.
e-mail:jesarana@eisc.univalle.edu.co

** Ph.D. Profesor Titular - Escuela de Ingeniería de Sistemas y Computación - Universidad del Valle - Santiago de Cali, Colombia.
e-mail:jdiaz@eisc.univalle.edu.co

*** Ingeniero de Sistemas - Estudiante del Programa de Doctorado en Ingeniería, Área de Énfasis: Ingeniería de Sistemas y Computación - Universidad del Valle- Santiago de Cali, Colombia.
e-mail:jaortiz@eisc.univalle.edu.co

Fecha de recepción: Abril 30 de 2004
Fecha de aprobación: Junio 29 de 2004

RESUMEN

Este artículo presenta **CREAR**, una aplicación computacional desarrollada para realizar más eficientemente el proceso de distribución de los artículos candidatos para participar en el evento del **CLEI**[1]. Para su desarrollo se diseñó un modelo computacional del problema, utilizando el Paradigma de Programación por Restricciones y se implementó utilizando **MOzArt**, un lenguaje de programación apropiado para este paradigma. El paradigma y el modelo computacional también se describen en este artículo.

Palabras Claves: Optimización, problemas combinatorios, programación por restricciones, MOzArt, lenguajes de programación.

ABSTRACT

In this paper we present **CREAR**, a software developed for helping in different academic events, specifically in **CLEI 2005**, where it will help in the process of selecting and classifying proposed papers in an orderly fashion. A computational model of the problem was made by using the Concurrent Constraints Programming Paradigm, and it was implemented by using the **MOzArt** programming language. Both the paradigm and computational model are also described here.

Key Words: Optimization, Combinatorial Problems, Programming with Constraints, MOzArt, Languages of Programming

1. INTRODUCCIÓN

Un evento o congreso académico se centra en una serie de conferencias, en las cuales se exponen diferentes trabajos o artículos de investigación, los cuales son previamente referenciados y seleccionados por un Comité de Programa. Estos artículos pueden ser de diferentes áreas de investigación pero relacionadas con el tema central del congreso. Cada representante en el comité de programa, a su vez, tiene un grupo de trabajo que le colabora en el proceso de evaluación de esos artículos.

El tiempo que toma el proceso de distribuir los artículos candidatos, es decir aquellos que son presentados con el fin de participar del evento académico y para lo cual deben ser evaluados, puede ser de varios días y adicionalmente, este proceso culmina con una solución, es decir, una distribución, que ocasiona cierta incomodidad a los evaluadores. Además, este proceso, en la mayoría de ocasiones, se lleva a cabo de forma manual o poco asistida, es decir, utilizando herramientas de escritorio como hojas de cálculo.

Como es de esperar, se busca alcanzar una distribución, lo más rápidamente posible, que satisfaga la mayor cantidad de preferencias por parte de los evaluadores.

Con base en lo anterior, se desarrolló una herramienta para brindar apoyo a la toma de decisiones en lo referente a la distribución de los artículos a ser evaluados.

En este artículo se presentan aspectos relacionados con el problema específico de asignación y la herramienta desarrollada. Inicialmente se realiza una presentación del Paradigma de Programación por Restricciones en la sección 2 y una descripción detallada del problema en la sección 3. Luego, se presenta el modelo de restricciones del problema (ver sección 4), la arquitectura del software desarrollado (ver sección 5) y la interfaz de la aplicación (ver sección 6). Para finalizar se presentan las conclusiones de este trabajo (ver sección 7).

2. EL PARADIGMA DE PROGRAMACIÓN CONCURRENTE POR RESTRICCIONES

La expresión "Programación Concurrente por Restricciones", CCP (Constraints Concurrent Programming) [594], integra tres nociones fundamentales. Primero, "programación" se refiere a programación de un computador. Un programa de computador es una expresión de un método computacional en un lenguaje del computador. Segundo, "concurrente" se refiere a que el método computacional ofrece la posibilidad de establecer ordenamientos mínimos de las acciones de modo que varias de ellas se puedan eventualmente realizar

[1] Centro Latinoamericano de Estudios en Informática, anualmente lleva a cabo un evento donde se presentan a la comunidad diferentes investigaciones y trabajos realizados en Latinoamérica que tratan de los adelantos realizados en el área de la informática. .

simultáneamente. Finalmente, "restricciones" hace referencia a que las acciones se expresan mediante condiciones que debe cumplir la solución de un problema para que se acepte como factible. La programación por restricciones es entonces una metodología particular para programar computadores.

Hay dos clases de problemas para los que se estima conveniente utilizar programación por restricciones:

- ☑ Problemas de satisfacción de restricciones.
- ☑ Problemas de optimización combinatoria.

En los problemas de satisfacción de restricciones (CSP) [M94], se busca obtener soluciones factibles. En los problemas de optimización combinatoria, se busca además que la solución factible conduzca a la minimización (o maximización) de una función objetivo. La estructura particular de estos dos tipos de problemas, se adapta a la metodología empleada por CCP para resolverlos:

- ☑ Representar un modelo del problema en un lenguaje de computador.
- ☑ Describir una estrategia de búsqueda para resolverlo.

Para la representación es conveniente que el lenguaje de computador en cuestión permita describir el modelo en términos cercanos al área de aplicación. La estrategia de búsqueda, por otra parte, debe conducir a encontrar soluciones de manera eficaz.

En un CSP se trata de encontrar una solución factible, sujeta a una serie de condiciones o restricciones sobre el conjunto de valores de las variables. Usualmente se requiere que las restricciones, expresadas en el lenguaje de programación, sean fáciles de evaluar. Los valores de las variables toman usualmente valores sobre un conjunto discreto. En los problemas de optimización se requiere así mismo que la función objetivo sea fácil de evaluar.

Una representación de un CSP se caracteriza por lo siguiente:

- ☑ Variables:
  $X_1, X_2, \ldots, X_n$ n corresponde al número de variables y a su vez número de dominios. .
- ☑ Valores posibles (dominios):
  $D_1, D_2, \ldots, D_n$

Donde D_j es el dominio de X_j. El dominio puede ser finito o infinito.

Las restricciones son funciones (predicados, realmente):

- ☑ Restricciones:
  $F(X_1, \ldots, X_n)$ [0,1]

Cuando el valor de la función es uno, se dice que los valores en cuestión satisfacen la restricción. Cuando es 0, no se satisface.

Concretamente, en un CSP, se trata de encontrar valores de $X_1, \ldots, X_n$ tales que:

X_j D_j, para $j = 1,2,\ldots,n$

$f_k(X_1,\ldots,X_n) = 1$, para $k = 1,2,\ldots,m$ donde m es el número de restricciones.

Un dominio puede ser cualquier conjunto. Por ejemplo los enteros 1...100 o los nombres {Pedro, Juan, María}o los reales en el intervalo [0,50]. Similarmente, las restricciones pueden ser de muchas clases. Por ejemplo:

- ☑ Restricciones lógicas: Si cierta variable cumple con alguna propiedad, por ejemplo, Caudal > 67.5 entonces otra variable también, por ejemplo, Apertura = 0.
- ☑ Restricciones globales: Todos los valores de $X_1, \ldots, X_n$ deben ser diferentes.

- ☑ Meta restricciones: El número de veces que 5 aparece en el arreglo A es exactamente igual a 3.

- ☑ Restricciones de elemento: El costo de asignar a la persona i la tarea j es costo[tarea[i]], cuando tarea[i]=j.

En este contexto, la programación concurrente por restricciones dispone de:

- ☑ Una metodología de modelamiento para identificar convenientemente las variables, las restricciones y la función objetivo (si la hay) de un problema.
- ☑ Un lenguaje de programación **MOzArt** [HM95], [SS04], [VH03], para expresar algoritmos de búsqueda que encuentren valores de las variables que satisfagan todas las restricciones y optimicen el objetivo.
- ☑ Un sistema de programación que incluya: Restricciones predefinidas con algoritmos poderosos de filtrado o propagación de valores, que reduzcan de manera efectiva el tamaño del espacio de búsqueda. Funcionalidad para la definición de nuevas restricciones y algoritmos de filtrado.

En los lenguajes CCP, las restricciones constituyen componentes básicos, que expresan información parcial sobre las variables. Cada restricción está a cargo de un agente, cuyo papel es velar por el cumplimiento de esa restricción, propagando a los otros agentes sus efectos (esto es, la nueva información que se deduzca de ella). Todos estos agentes operan de manera concurrente.

La memoria contiene predicados que representan información parcial sobre las variables y los agentes interactúan con la memoria agregando información (agentes tell) o haciendo preguntas (agentes ask). Los agentes tell agregan una restricción a la memoria y los agentes ask preguntan si, de las restricciones ya contenidas en la memoria, se puede deducir una restricción dada.

El grupo de investigación **AVISPA**[2] cuenta entre sus trabajos con aplicaciones desarrolladas bajo este paradigma, tales como **PATHOS** [QR02], un sistema de apoyo al planeamiento de horarios de clase aplicado a universidades y **VISiR,** un software de soporte para la toma de decisiones de vertimiento de agua en la represa del alto anchicayá.

3. DESCRIPCIÓN DEL PROBLEMA

De acuerdo a los organizadores de la conferencia latinoamericana de informática, **CLEI**, el tiempo que toma el proceso de distribución de artículos es de 3 a 4 días. La mayor dificultad radica en poder asignar evaluadores suficientes a cada uno de los artículos cumpliendo con ciertas restricciones que se tienen en el evento (restricciones que serán descritas posteriormente). Adicionalmente, la distribución resultante presenta no pocas asignaciones de artículos que no cumplen con las restricciones.

Considerando lo anterior, la organización del evento **CLEI** que se va a realizar en Colombia en el 2005, ha desarrollado una herramienta que brinde soporte al proceso de distribución de artículos, buscando con ello, reducir el tiempo que demora el proceso y minimizar el número de asignaciones que no cumplen con todas las restricciones planteadas para el evento.

Los elementos principales en el proceso de distribución de artículos en el evento académico del **CLEI** son los siguientes:

- ☑ **Un conjunto de artículos o trabajos:** La cantidad de artículos presentes en un evento como **CLEI 2005** se estima de 300 a 400 aproximadamente y a partir de este conjunto

[2] Ambientes Visuales de Programación Aplicativa, este grupo de investigación está formado por profesores y estudiantes de las Universidades del Valle y Javeriana, se encuentra actualmente llevando a cabo investigaciones que giran alrededor del paradigma CPP. Los integrantes de este grupo están actualmente desarrollando CREAR.

se seleccionan aquellos que se presentarán en las conferencias.

- ☑ **Un grupo de evaluadores:** Un artículo es revisado por 3 evaluadores; por lo tanto, si llegan 300 artículos al evento, se necesitarían 900 evaluaciones las cuales serían realizadas por los evaluadores que conforman el comité de programa.
- ☑ **Un conjunto de restricciones:** Son las condiciones que deberían cumplirse en cualquier distribución de los artículos o trabajos recibidos.

Las restricciones que se deben tener en cuenta al momento del proceso de distribución de los artículos son las siguientes:

- ☑ **De los evaluadores:** Cada evaluador define los temas, el idioma en que puede evaluar y el número máximo de artículos que está en capacidad de evaluar.
- ☑ **De los artículos:** Cada artículo debe ser evaluado por conocedores de los temas principales del mismo, los cuales están definidos para cada artículo.
- ☑ **De artículos y evaluadores:** Los evaluadores asignados deberían ser de nacionalidad diferente a la del país de donde proviene el artículo y deberían leer en el idioma en que éste está escrito.

Durante el proceso de distribución, se busca minimizar el número de asignaciones que no cumplan las anteriores preferencias, cuando no es posible que un articulo sea asignado de tal manera que cumpla con todas las preferencias, se asigna considerando cuáles preferencias son más importantes, de tal manera que algunas se cumplan y otras no, la organización del evento también puede considerar que ciertas preferencias son de obligatorio cumplimiento y por ende puede ocurrir que no se logre una distribución total en principio, en tal caso, las asignaciones faltantes son analizadas por la organización del evento con miras a encontrar una solución.

La herramienta desarrollada, considera los diversos factores expuestos brindando una posible solución al proceso de distribución.

4. MODELO DE RESTRICCIONES

Para resolver el problema, se han analizado los datos con que se cuenta al momento de la distribución y las preferencias que se plantean. Con base en esa información se plantean las siguientes restricciones:

- ☑ El número de evaluadores por artículo debe ser 3, donde un evaluador no puede evaluar dos veces el mismo artículo.
- ☑ El número de evaluaciones por evaluador no debe ser superior a la capacidad máxima definida para él.
- ☑ El país del artículo debe ser diferente al país del evaluador que le sea asignado.
- ☑ El idioma en el cual ha sido escrito el artículo debe ser uno de los idiomas que esté en capacidad de leer el evaluador que le sea asignado.
- ☑ Alguno de los temas principales del artículo debe coincidir con alguno de los temas que domina el evaluador asignado.

Las anteriores restricciones fueron consideradas en el sistema **CREAR**, de tal manera, que la mayoría de ellas, puedan imponerse o no, al momento de encontrar la solución.

5. LA HERRAMIENTA CREAR

El sistema está organizado en módulos, los cuales se encuentran clasificados según el nivel al que pertenezcan.

Además de **MOzArt**, se utilizó el lenguaje de programación Java fundamentalmente en el nivel de presentación.

La herramienta requiere que el equipo cliente disponga de la máquinas virtuales de MOzArt y Java. Estas máquinas virtuales trabajan sobre

sistemas operativos Windows y Linux. Las características de hardware deseables para la aplicación son: memoria superior o igual a 64MB, procesador Pentium II, equivalentes y superiores, disponibilidad en disco duro superior o igual a 20 MB.

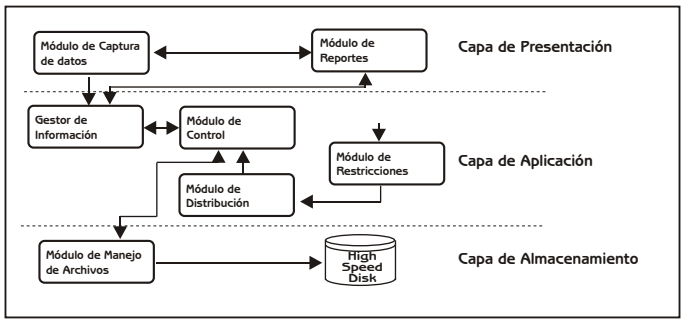

Figura 1. Diagrama de la Arquitectura de CREAR

Además de **MOzArt**, se utilizó el lenguaje de programación Java fundamentalmente en el nivel de presentación.

La herramienta requiere que el equipo cliente disponga de la máquinas virtuales de MOzArt y Java. Estas máquinas virtuales trabajan sobre sistemas operativos Windows y Linux. Las características de hardware deseables para la aplicación son: memoria superior o igual a 64MB, procesador Pentium II, equivalentes y superiores, disponibilidad en disco duro superior o igual a 20 MB.

El sistema presenta 3 niveles:

- ☑ **Nivel de presentación:** en este nivel se encuentran todas las funcionalidades que permiten la interacción entre el programa y el usuario.
- ☑ **Nivel de aplicación:** en este nivel se lleva a cabo el control de programa y se encuentran las funcionalidades principales de la herramienta.
- ☑ **Nivel de persistencia o almacenamiento:** en este nivel se tienen los archivos de entrada y salida, así como las funcionalidades que permiten su comunicación con el programa.

A su vez, los niveles presentan los siguientes módulos:

Nivel de Presentación

- ☑ **Módulo de Captura de Datos:** Presenta las funcionalidades que permiten seleccionar los datos de entrada, decidir dónde almacenar los datos de salida, las restricciones que desea aplicar y las estrategias para encontrar la solución.
- ☑ **Módulo de Reportes:** Presenta las funcionalidades que permiten al programa exhibir reportes con estadísticas y datos de interés de la solución encontrada. Estos reportes son fundamentales para la organización del evento dado que con base en ellos se realiza un análisis final que conduzca a una mejor solución.

Nivel de Aplicación

- ☑ **Gestor de Información:** Presenta las funcionalidades que permiten la comunicación entre la interfaz y el motor de la aplicación, de manera que puedan trabajar conjuntamente.
- ☑ **Módulo de Control:** Presenta las funcionalidades que permiten velar por la integridad del sistema, básicamente en el proceso de búsqueda de la solución.
- ☑ **Módulo de Distribución:** Presenta las diferentes estrategias que se pueden usar para encontrar la solución.
- ☑ **Módulo de Restricciones:** Presenta todas las restricciones a considerar.

Nivel de Persistencia

Módulo de manejo de archivos: Presenta las funcionalidades que permiten leer los datos de entrada del problema y generar el archivo con la solución.

5.1 Flexibilidad en las Restricciones:

Una de las características más importantes del

proyecto es la flexibilidad en la imposición de las restricciones.

Debido a que en el problema se manejan múltiples restricciones, y en muchas ocasiones, imponerlas todas, lleva a que no exista una solución que las satisfaga, el sistema permite que el usuario escoja cuáles restricciones desea aplicar. Las restricciones que el usuario puede escoger o no, son las siguientes:

- ☑ **Restricción de Capacidad**, la cual obliga a que el número de artículos a evaluar por cada evaluador no supere una capacidad máxima.
- ☑ **Restricción de Lenguas**, la cual obliga a que el idioma del artículo deba ser uno de los idiomas que dice manejar el evaluador que le corresponde.
- ☑ **Restricción de País**, la cual obliga a que el país de origen del artículo, se diferente al país de procedencia de los evaluadores.

Nótese que la restricción de temas no es opcional. Esto porque mientras el idioma o la capacidad se pueden cambiar sin mayores inconvenientes con los evaluadores, en caso de ser necesario, lograr que un evaluador evalúe un tema que no domina no es conveniente.

5.2 Solución paso a paso:

Otra de las características distintivas del sistema, radica en la posibilidad de encontrar una solución de manera incremental permitiendo inhabilitar restricciones a medida que se acerca a la solución.

Para ello, el sistema permite manejar hasta cuatro pasos, de la siguiente manera:

Paso 1: el sistema busca una solución que satisfaga las restricciones escogidas por el usuario, al final puede haber aún evaluaciones sin responsable asignado.

Paso 2: el sistema considera aquellas asignaciones que no se pudieron realizar en el paso anterior y las intenta realizar teniendo en cuenta las restricciones que se escojan en ese momento, generalmente menos que la original, al final puede haber aún evaluaciones sin responsable asignado.

Lo anterior ocurre también en los pasos 3 y 4.
Al finalizar todos los pasos es posible que no haya evaluaciones suficientes asignadas para algunos de los artículos; sin embargo, el sistema logra que este número sea bastante reducido.

Es importante destacar que el usuario puede escoger el número de pasos y las restricciones a tener en cuenta en cada uno de ellos.

6. LA INTERFAZ CREAR

La interfaz ofrece una gran flexibilidad al usuario, permitiéndole escoger las restricciones, los pasos y las estrategias para encontrar la solución.

El sistema, en primer término, permite al usuario determinar el archivo con los datos de entrada del problema y dónde debe quedar almacenada la solución. Con base en lo anterior, el usuario puede proceder a decidir qué estrategia usar para encontrar la solución.

El sistema ofrece diversas estrategias para encontrar la solución; cada una de ellas, considera diferentes criterios para decidir qué artículo se asigna primero y a qué evaluador le debería corresponder.

Posteriormente, el sistema solicita qué restricciones quiere aplicar; una vez el usuario las haya escogido, tiene que definir cuántas etapas posteriores se intentarán (máximo 3) y qué restricciones se considerarán en cada una de ellas.

El sistema ofrece diferentes reportes, los cuales sirven de apoyo al usuario final, para tomar la decisión de cuál es la distribución más apropiada y qué medidas hay que tomar con aquellas asignaciones faltantes.

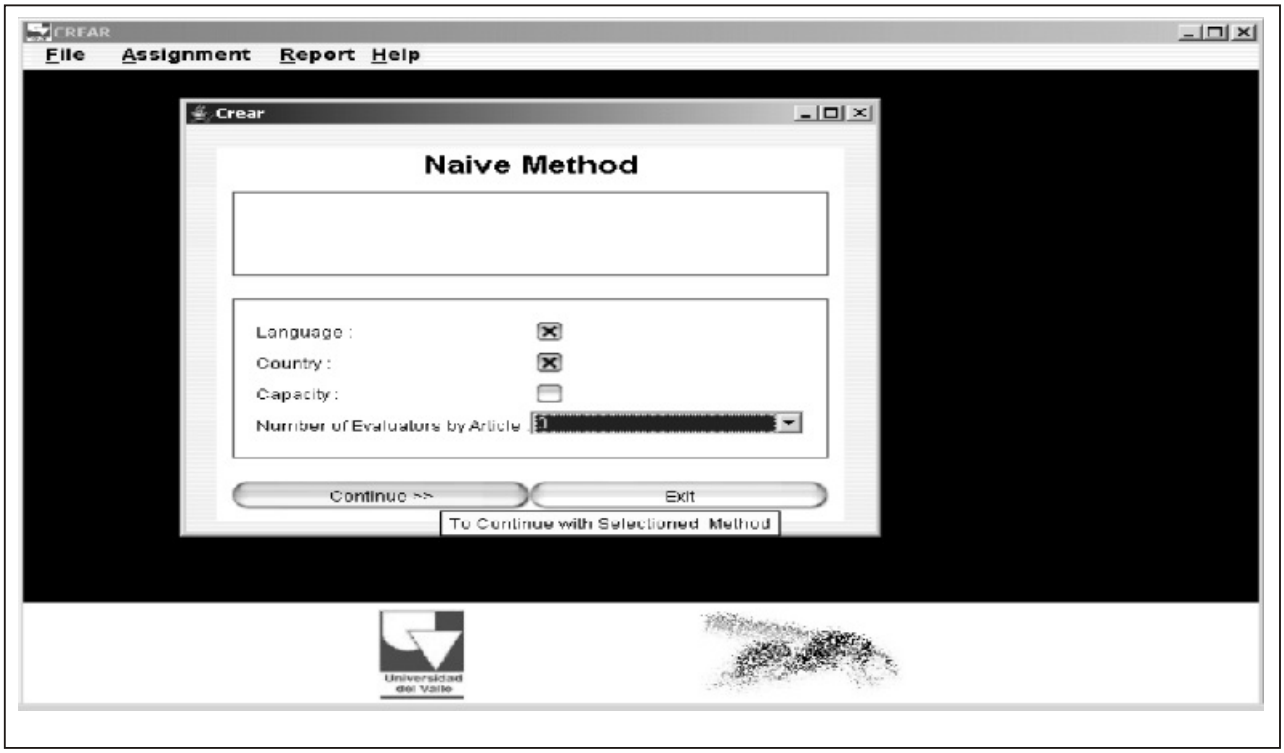

Figura 2. Selección de las Restricciones en CREAR

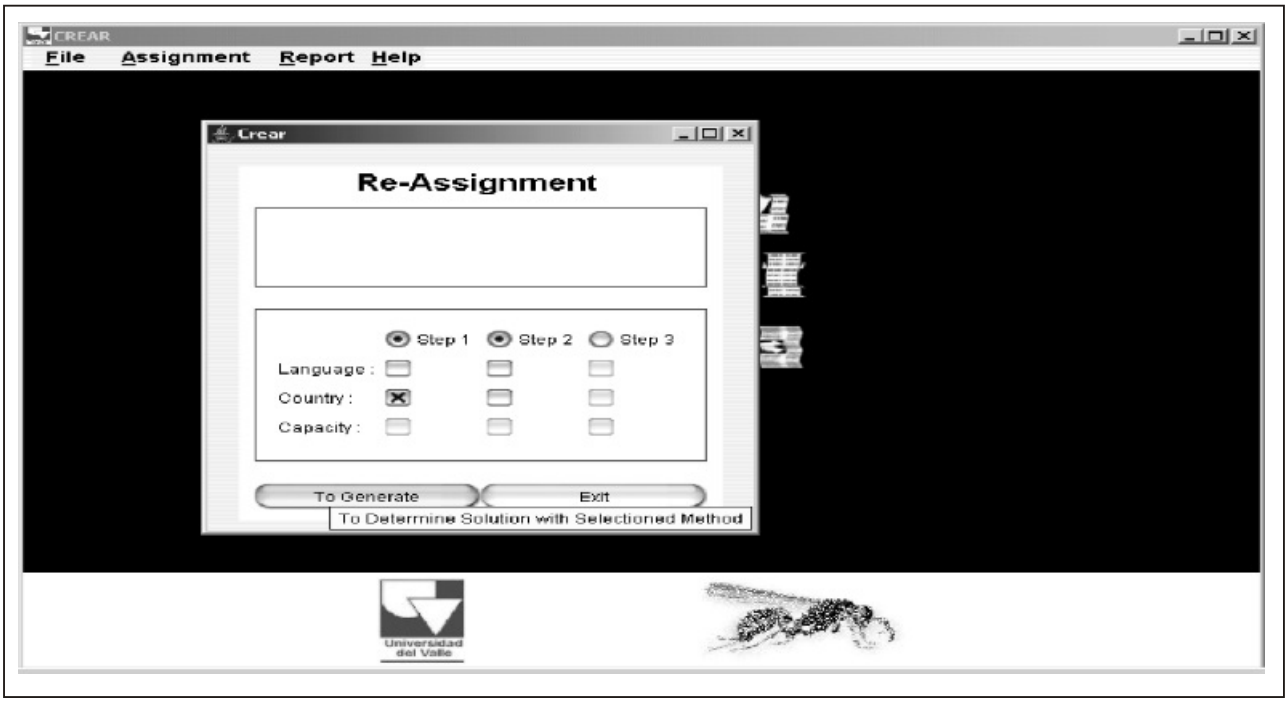

Figura 3. Múltiples pasadas en CREAR

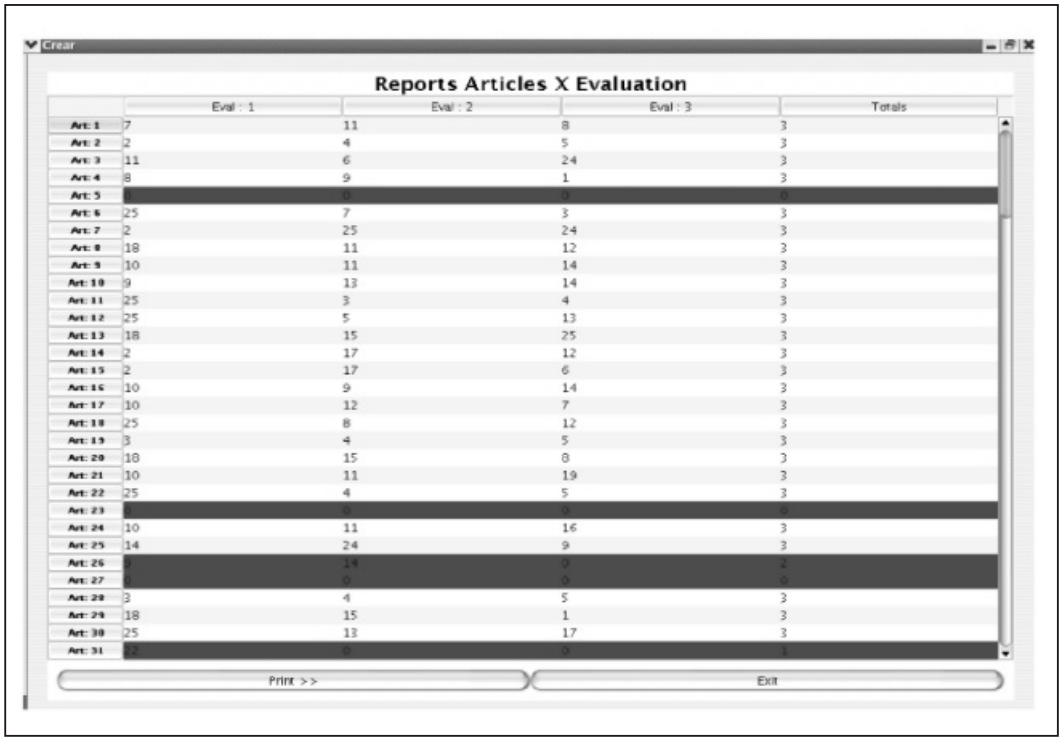

Figura 4. Reporte con evaluaciones que le corresponden a cada artículo

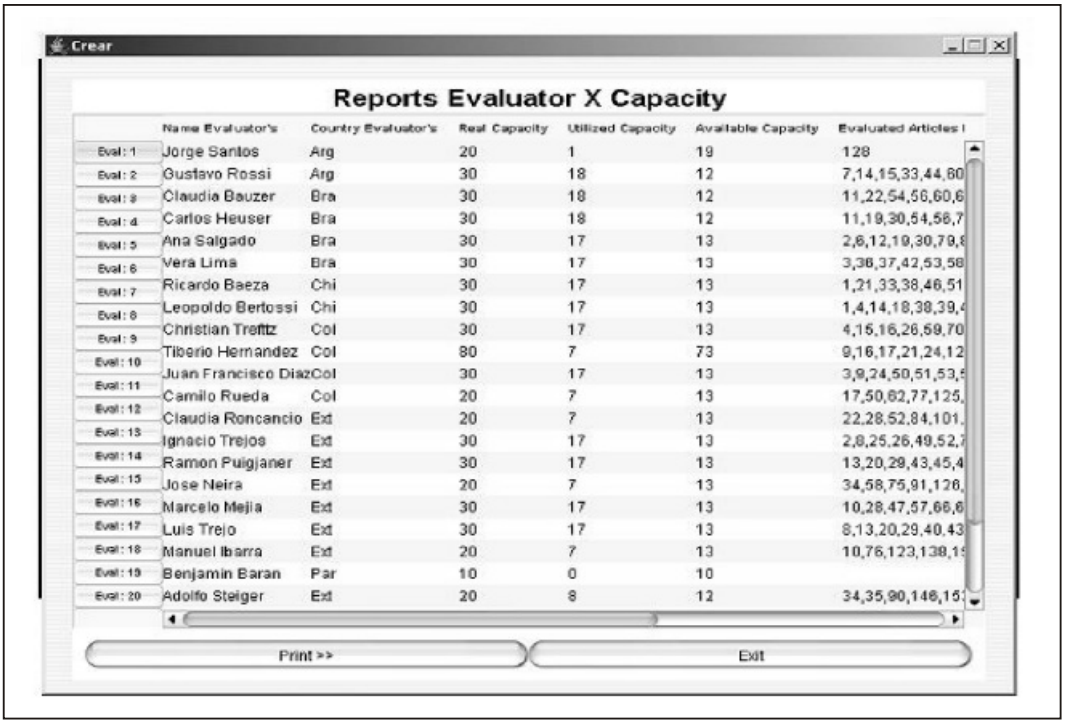

Figura 5. Reporte con la información de cada evaluador: su capacidad, número de artículos a evaluar y los artículos respectivos.

7. CONCLUSIONES

- El Paradigma de Programación Concurrente por Restricciones usado para resolver el problema, se mostró bastante apropiado en el diseño y la implementación posterior.
- El uso de **MOzArt** con otros lenguajes de programación como Java, es una alternativa válida en la elaboración de proyectos, para los cuales, ciertos lenguajes de programación diferentes a **MOzArt** ofrecen ciertas facilidades, además de una difusión más amplia.
- Se logró aplicar el conocimiento adquirido por el Grupo de Investigación **AVISPA** a la solución de un problema real combinatorio que ocurre en el **CLEI**.

8. TRABAJOS FUTUROS

- Acoplar la herramienta para que haga parte de sistemas de apoyo a la distribución de artículos como WIMPE [N01] u OpenConf [Z04].
- Incorporar restricciones acerca de la disposición de cada evaluador para leer artículos de ciertos temas; esta disposición puede estar a varios niveles: alto, medio, bajo o no disposición, por ejemplo.
- Permitir que la herramienta utilice una gran variedad de restricciones, de manera que el usuario de la misma, indique a priori cuáles desea tener en cuenta y así, desde la misma interfaz, la herramienta se acomoda a estas preferencias. Con esto se busca lograr ampliar el campo de acción de la herramienta a diversos eventos académicos.
- El modelo y la estrategia de distribución serán descritos y analizados en próximas publicaciones.

9. REFERENCIAS

- [DR01] Juan F. Díaz and Camilo Rueda. VISIR: Software de soporte para la toma de decisiones de vertimiento de agua en la represa del alto anchicayá usando programación concurrente por restricciones. Ingeniería y Competitividad. Volumen 3 No. 2. 2001.
- [Hm95] M. Henz and M. Muller. Programming in Oz. In G. Smolka and R. Treinen, Editors, DFKI Oz, Documentation Series. 1995.
- [M94] Alan K. Mackworth , Eugene C. Freuder, The complexity of constraint satisfaction revised, Artificial intelligence in perspective, MIT Press, Cambridge, MA, 1994
- [MS98] K. Marriot and P. J. Stuckey. Programming with Constraints: An Introduction. MIT Press, Cambridge, Mass, 1998.
- [N01] David M. Nicol. WIMPE: Web Interface for Managing Programs Electronically.
- http://www.crhc.uiuc.edu/~nicol//wimpe/wimpe6.1.html. 2001.
- [QR02] Luis Quesada and Camilo Rueda. Planeamiento Horario Universitario Orientado-Objetos en Programación Concurrente por Restricciones. Epicliclos. Vol.1, No. 1. 2001.
- [S93] V. Saraswat, Concurrent Constraint Programming. MIT Press, 1993.
- [SS04] Christian Schulte and Gert Smolka. Finite Domain Constraint Programming in OZ. A Tutorial. **MOzArt** Documentation, 2004.
- [Vh03] P. Van Roy and Seif Hairidi. Concepts, Techniques, and Models of Computer Programming. MIT Press, 2004.
- [Z04] Zacon Group. OpenConf-Conference Manual Management System.
- http://www.OpenConf.org. 2004.