CONSTDROID: Prototipo de interface de un motor de restricciones para dispositivos Android

Ms. Gonzalo José Hernández
Departamento de Sistemas
Universidad de Nariño
Pasto, Colombia
Email: gonzalohernandez@udenar.edu.co

PhD. Ricardo Timarán Pereira
Departamento de Sistemas
Universidad de Nariño
Pasto, Colombia
Email: ritimar@udenar.edu.co

Ms. Alexander Baron
Departamento de Sistemas
Universidad de Nariño
Pasto, Colombia
Email: abaron_98@hotmail.com

***Abstract*—CONSTDROID es una aplicación creada para ser ejecutada en dispositivos Android con el fin de servir de interface entre un usuario diseñador de *CSP*s y un motor de restricciones.**

I. INTRODUCCIÓN

La Programación por Restricciones es un modelo de programación lógico que tiene por objetivo resolver problemas combinatorios, problemas que requieren de un diseño de solución diferente a los problemas convencionales. Para este fin, la programación por restricciones utiliza un modelamiento denominado *CSP* (Constraint Satisfaction Problem) que contiene 3 elementos: Variables, Dominios y Restricciones[1].

Para resolver este tipo de problemas no solo se requiere de diseñar la solución, sino que también se requiere de un motor de búsqueda el cual pueda interpretar esos tres elementos.

Entre los motores de búsqueda más utilizados se encuentran: *Gecode*[2] que utiliza codificación en *C++*, *Choco*[7] que utiliza codificación en *Java* y *Mozart*[5] que utiliza codificación en *Oz*. Todos estos motores requieren de una sintaxis especial para su implementación, pero al final el motor es el encargado de resolver el problema.

CONSTDROID, es un prototipo de software creado con el objetivo de servir de interface gráfica de usuario, de tal forma que un problema ya modelado como *CSP* pueda ser llevado al motor de restricciones de una forma sencilla sin necesidad de aprender una sintaxis especial.

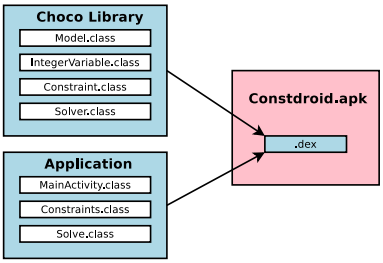
CONSTDROID, es una aplicación *APK*[3] creada para funcionar en plataformas Android, la cual permite que de forma rápida un usuario pueda evaluar un *CSP*.

Este documento pretende mostrar el primer prototipo de CONSTDROID, como una herramienta útil para resolver problemas sencillos que requieren de un modelamiento por restricciones. En los siguientes capítulos se detalla: la Arquitectura de CONSTDROID, algunos aspectos de diseño e implementación, algunas pruebas de funcionalidad, y las conclusiones y trabajos futuros.

II. ARQUITECTURA DE CONSTDROID

Por afinidad en el lenguaje de programación, se seleccionó a *Choco* como el motor de restricciones para resolver las búsquedas en CONSTDROID. La Figura 1 bosqueja los elementos estructurales que utiliza esta aplicación.

Fig. 1: Arquitectura de CONSTDROID



III. ASPECTOS DE DISEÑO E IMPLEMENTACIÓN

Para su integración fue necesario vincular la librería en el código fuente del proyecto, el cual se encuentra alojado en el repositorio de código https://code.google.com/p/constdroid que permite control de versiones. El acceso a los detalles de su implementación es de libre acceso. Y el último APK puede ser descargado en http://sonar.udenar.edu.co/android-constraint-engine así como los detalles de su funcionamiento.

IV. PRUEBAS DE FUNCIONALIDAD

Para comprobar que el prototipo es capaz de resolver un problema combinatorio, se utilizó uno de los problemas que se suele estudiar en esta área. *Grocery Puzzle*.

A kid goes into a grocery store and buys four items. The cashier charges $7.11, the kid pays and is about to leave when the cashier calls the kid back, and says "Hold on, I multiplied the four items instead of adding them; I'll try again; Hah, with adding them the price still comes to $7.11". What were the prices of the four items? [4]

En el ejercicio es necesario encontrar 4 valores que tanto sumados como multiplicados entre sí, se obtenga un total de 7.11, por supuesto es un problema que no es posible resolverlo por un procedimiento matemático directo.

El modelamiento de la solución de este problema es planteado en *Mozart documentation*[6], transformado para que los valores con decimales del problema puedan ser tratados por medio de dominios enteros.

El modelo de la Figura 2, fue ingresado en CONSTDROID, el cual se encontraba corriendo sobre un sistema Android

Fig. 2: CSP

Variables:	{A, B, C, D}
Domains:	{1..708, 1..708, 1..708, 1..708}
Constraints:	{A + B + C + D = 711, A * B * C * D = 711000000 }

Fig. 3: Configuración del dispositivo

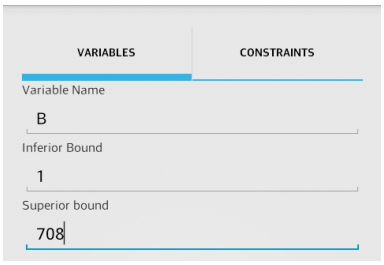
BRAND:	LG-E455g
OS:	Android OS, v4.1.2 (Jelly Bean)
CPU:	1 GHz Cortex-A9
Internal Memory:	4 GB, 512 MB RAM
Display Resolution:	480 x 800 pixels (233 ppi)

instalado en un equipo con las configuraciones presentadas en la Figura 3.

A continuación, se presenta el procedimiento llevado a cabo para la ejecución del ejercicio.

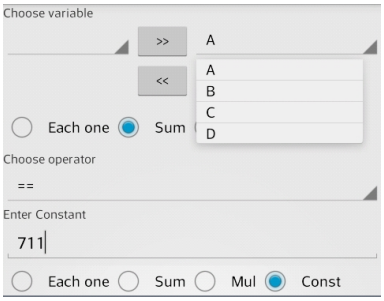
Una vez iniciada la aplicación tal como se muestra en la Figura 4, se procedió a ingresar las variables con sus respectivos dominios. CONSTDROID permite agregar variables independientes y también arreglos de variables.

Fig. 4: Ingreso de Variables



El siguiente paso consistió en agregar las restricciones. Tal como se muestra en la Figura 5, se procedió a relacionar las variables. La interface permite hacer varios tipos de relaciones.

Fig. 5: Ingreso de Restricciones



Con la opción *Status*, se puede visualizar el estado del ingreso del CSP. La Figura 6 presenta el resultado de esta consulta.

Para finalizar, en la sección *Solution*, se ejecuta la búsqueda de una solución. En la Figura 7 se puede ver los valores asignados a cada una de las variables: A=150, B=316, C=125 y D=120.

Fig. 6: Estado de CSP

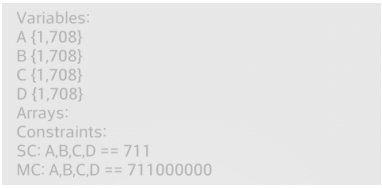
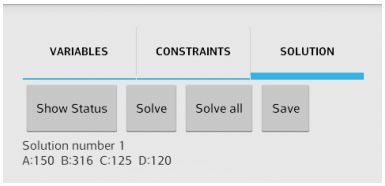


Fig. 7: Solución



Debido a que el CSP fue modelado de tal forma que los valores pudieran ser trabajados como enteros, el resultado del problema correspondería a:

$$1.50 + 3.16 + 1.25 + 1.20 = 7.11$$
$$1.50 * 3.16 * 1.25 * 1.20 = 7.11$$

V. CONCLUSIONES Y TRABAJOS FUTUROS

Este primer prototipo de CONSTDROID tenía como finalidad probar el funcionamiento del motor de restricciones e identificar el mecanismo de diálogo con el usuario final, la interface de usuario es simple y por lo tanto su mejoramiento es uno de los objetivos propuestos para la siguiente versión.

Con pocos pasos, un usuario puede probar si un CSP simple cumple con la necesidad requerida, sin embargo, también es necesario agregarle a CONSTDROID otros tipos de operaciones y funcionalidades. Seguramente, la siguiente versión de este prototipo utilizará *Choco3*, y no *Choco2* como la actual versión.

Este tipo de aplicaciones son bastantes útiles en procesos académicos, donde se requiere que el estudiante adquiera una nueva forma de modelar un problema, evitándole inicialmente recurrir a tener que aprender una sintaxis específica de un lenguaje.

REFERENCES

[1] K. Apt. *Principles of Constraint Programming*. Cambridge University Press, London, 2003.

[2] Gecode. Generic constraint development environment. http://www.gecode.org, 2015. [Online; accessed 12-May-2015].

[3] Google. Android developers. http://developer.android.com, 2015. [Online; accessed 12-May-2015].

[4] S. Loyd. *Cyclopedia of Puzzles*. The Lamb Publishing Company, New York, 1914.

[5] P. V. Roy. *Concepts, Techniques, and Models of Computer Programming*. MIT Press, 2004.

[6] C. Schulte. Example: Grocery. https://mozart.github.io/mozart-v1/doc-1.4.0/fdt/node21.html, 2015. [Online; accessed 12-May-2015].

[7] École des Mines de Nates. choco: an open source java constraint programming library. 2010.