ALGORITMO PARA LA CONSTRUCCIÓN DE ARREGLOS DE CUBRIMIENTO BASADO EN LA MEJOR BÚSQUEDA ARMÓNICA GLOBAL Y TÉCNICAS DE OPTIMIZACIÓN LOCAL

JIMENA ADRIANA TIMANÁ PEÑA

ANEXOS

Director: Ph.D. Carlos Alberto Cobos Lozada
Co-Director: Ph.D. Martha Eliana Mendoza Becerra

Asesor: Ph.D. José Torres Jiménez
(CINVESTAV, Tamaulipas, México)

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Grupo de I+D en Tecnologías de la Información (GTI)
Línea de Investigación: Sistemas Inteligentes e Ingeniería de Software
Popayán, Noviembre de 2017

Anexo A

PARALELIZACIÓN DEL ALGORITMO GHSSA

La Figura 1 muestra la arquitectura empleada para la ejecución del algoritmo GHSSA en paralelo, aprovechando la disponibilidad de múltiples PCs con procesadores i3, i5 o i7 de 2 a 4 cores.

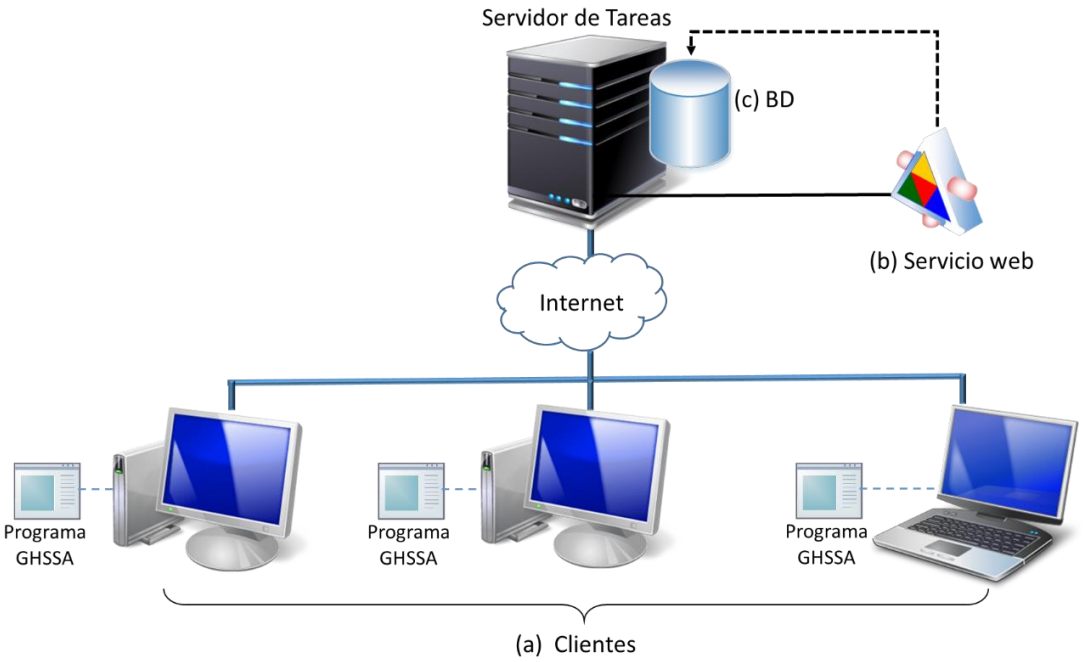


Figura 1 Arquitectura empleada para la ejecución paralela del algoritmo GHSSA

La arquitectura utilizada es Cliente-Servidor, donde en cada uno de los equipos Cliente, se instala el programa GHSSA.exe tal como se muestra en Figura 1(a). En el servidor de tareas se encuentra un servicio web que se muestra en la Figura 1(b) y que es el encargado de recibir las peticiones de asignación de tareas que llegan de los Clientes para posteriormente asignarlas. En el servidor de tareas se encuentra también una base de datos, Figura 1(c) donde se registran las tareas, cada una de ellas con una configuración determinada para construir un CA.

Cada tarea tiene configurado los siguientes elementos a saber:

- Tar_Id, para identificar cada una de las tareas.
- Tar_CA_Requerido, tiene la configuración del Covering Array que se busca construir.
- Tar_Algoritmo, indica el algoritmo con el que se desea construir el CA. Que puede ser el algoritmo GHSSA, GHS o SA.
- Tar_Parametros, parámetros de los algoritmos, indican en orden el número de improvisaciones definidas para el algoritmo de Búsqueda Armónica y el número máximo de iteraciones definidas para el algoritmo de Recocido Simulado. Los otros parámetros de HHS y SA están previamente registrados en el programa cliente GHSSA.exe.
- Tar_Semilla, define la semilla con la que el algoritmo meta-heuristico iniciará la generación de números aleatorios.
- Tar_Estado, indica el estado que puede tener una tarea, definida como (N) para tarea no asignada. (P) para tarea en proceso o ejecución (ya un proceso la está ejecutando). (I) para tarea interrumpida (cuando el CA buscado es encontrado por otra tarea con semilla distinta, el proceso de búsqueda de las otras tareas para el mismo CA se interrumpe y se da paso a la búsqueda de otros CA). (F) para tarea fallida, cuando no se construye la configuración del CA y (S) para tarea exitosa cuando si se pudo construir el CA.
- Tar_CA_Encontrado: almacena el CA encontrado.
- Tar_Fecha: registra la fecha en la que se construyó el CA.
- Tar_Faltantes: cuando una tarea es Fallida (no se logró construir el CA) se registra en este campo las t-adas faltantes para que sea un CA.
- Tar_Tiempo: registra en milisegundos el tiempo que demoró en construir el CA.

En la Figura 2 se muestra un conjunto de tareas registradas en la base de datos, cada tarea con su correspondiente configuración de elementos.

Tar_Id	Tar_CA_Requerido	Tar_Algoritmo	Tar_Parametros	Tar_Semilla	Tar_Estado	Tar_CA_Encontrado	Tar_Fecha	Tar_Faltantes	Tar_tiempo
1	N5K4V2^4t2.ca	GHSSA	5000 5000	1	N	*NULL*	*NULL*	*NULL*	*NULL*
2	N5K4V2^4t2.ca	GHSSA	5000 5000	2	N	*NULL*	*NULL*	*NULL*	*NULL*
3	N5K4V2^4t2.ca	GHSSA	5000 5000	3	N	*NULL*	*NULL*	*NULL*	*NULL*
4	N5K4V2^4t2.ca	GHSSA	5000 5000	4	N	*NULL*	*NULL*	*NULL*	*NULL*
5	N5K4V2^4t2.ca	GHSSA	5000 5000	5	N	*NULL*	*NULL*	*NULL*	*NULL*
6	N5K4V2^4t2.ca	GHSSA	5000 5000	6	N	*NULL*	*NULL*	*NULL*	*NULL*
7	N5K4V2^4t2.ca	GHSSA	5000 5000	7	N	*NULL*	*NULL*	*NULL*	*NULL*
8	N5K4V2^4t2.ca	GHSSA	5000 5000	8	N	*NULL*	*NULL*	*NULL*	*NULL*
9	N8K4V2^4t3.ca	GHSSA	5000 5000	1	N	*NULL*	*NULL*	*NULL*	*NULL*
10	N8K4V2^4t3.ca	GHSSA	5000 5000	2	N	*NULL*	*NULL*	*NULL*	*NULL*
11	N8K4V2^4t3.ca	GHSSA	5000 5000	3	N	*NULL*	*NULL*	*NULL*	*NULL*
12	N8K4V2^4t3.ca	GHSSA	5000 5000	4	N	*NULL*	*NULL*	*NULL*	*NULL*
13	N8K4V2^4t3.ca	GHSSA	5000 5000	5	N	*NULL*	*NULL*	*NULL*	*NULL*
14	N8K4V2^4t3.ca	GHSSA	5000 5000	6	N	*NULL*	*NULL*	*NULL*	*NULL*
15	N8K4V2^4t3.ca	GHSSA	5000 5000	7	N	*NULL*	*NULL*	*NULL*	*NULL*
16	N8K4V2^4t3.ca	GHSSA	5000 5000	8	N	*NULL*	*NULL*	*NULL*	*NULL*

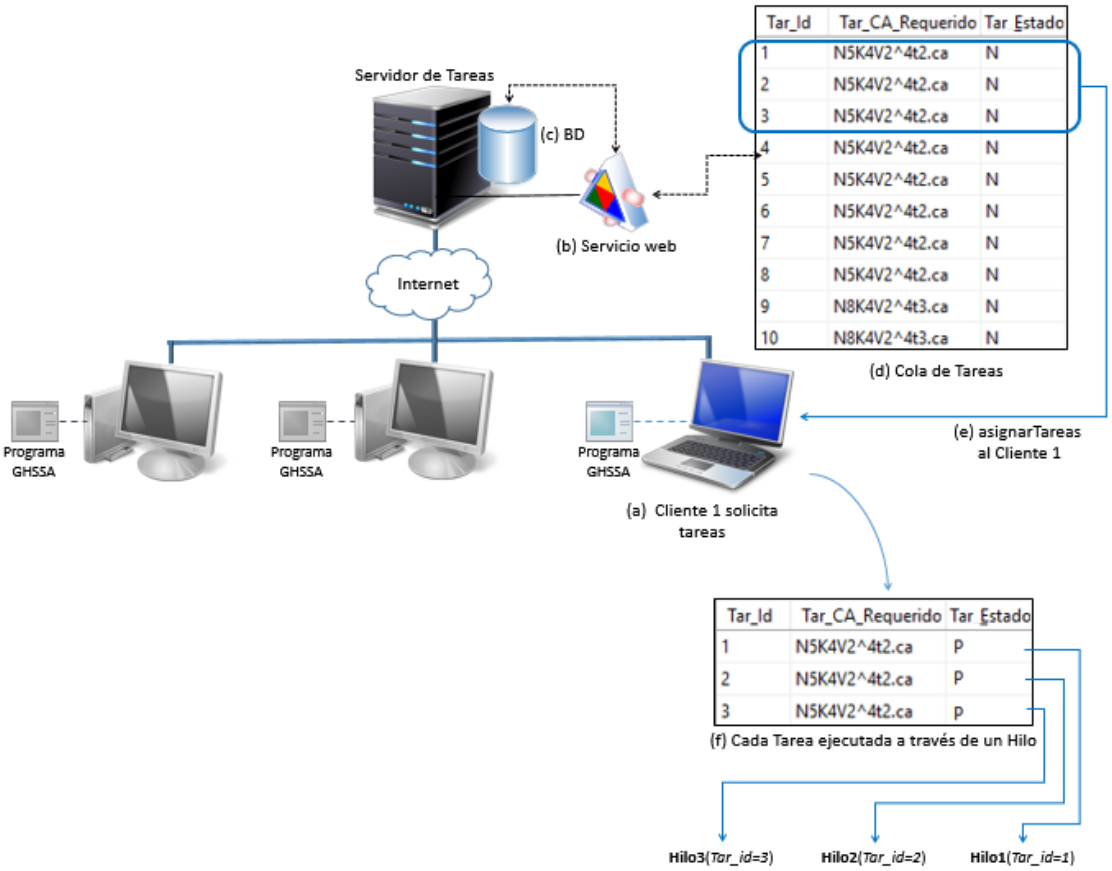
Figura 2 Conjunto de Tareas en la base de datos

A continuación se describe a través de un ejemplo, el proceso desde que un Cliente realiza una petición de asignación de Tarea hasta que ésta es finalmente terminada.

En la Figura 3(a) un equipo Cliente pone en funcionamiento el ejecutable del algoritmo GHSSA.exe, enviando una petición de asignación de tareas al Servidor de Tareas. La petición es atendida por el servicio web que se muestra en la Figura 3(b). El servicio web obtiene de la base de datos (Figura 3(c)), el conjunto de tareas con las configuraciones de Covering Arrays a construir y que estén en el estado de tarea no asignada o N. Para luego, organizarlas en una cola de tareas, como se muestra en la Figura 3(d) y desde ahí, asignarlas a cada uno de los equipos Clientes que han hecho una solicitud. En la figura en mención y solo por visualización se presentan tres columnas a saber: Tar_id, Tar_CA_Requerido y Tar_Estado. Sin embargo, en la cola de tareas, se carga la tarea completa con todos los elementos descritos previamente.

En la Figura 3(e), un número específico de tareas son asignadas al Cliente que hizo la petición. Para el ejemplo, al Cliente 1 le fueron asignadas 3 tareas.

Cada tarea asignada al Cliente 1 cambia su estado de tarea no asignada o N al estado tarea en proceso o P. La ejecución de cada tarea en el Cliente, se lleva a cabo a través de un hilo, como se muestra en la Figura 3(f).



Tar_Id	Tar_CA_Requerido	Tar_Estado
1	N5K4V2^4t2.ca	N
2	N5K4V2^4t2.ca	N
3	N5K4V2^4t2.ca	N
4	N5K4V2^4t2.ca	N
5	N5K4V2^4t2.ca	N
6	N5K4V2^4t2.ca	N
7	N5K4V2^4t2.ca	N
8	N5K4V2^4t2.ca	N
9	N8K4V2^4t3.ca	N
10	N8K4V2^4t3.ca	N

Tar_Id	Tar_CA_Requerido	Tar_Estado
1	N5K4V2^4t2.ca	P
2	N5K4V2^4t2.ca	P
3	N5K4V2^4t2.ca	P

Figura 3 Asignación de Tareas al Cliente 1

Cuando el Cliente 2 que se muestra en la Figura 4(a) solicita la asignación de tareas, el servicio web las asignará de acuerdo a las tareas disponibles en la cola de tareas y al número establecido de tareas que se debe asignar a cada equipo. Como se muestra en la Figura 4(b), 3 son el número de tareas a asignar en cada solicitud.

Una vez asignadas las tareas al Cliente 2, en la Figura 4(c) se muestra como cada una de ellas empezará su ejecución a través de un hilo de forma independiente. Es preciso observar que en esta caso, los dos clientes (Cliente 1 y Cliente 2) están buscando el mismo CA (N5K4V2^4t2.ca) pero cada una de las tareas tiene una semilla distinta, el primero que lo encuentre reporta al servicio web y hace que los otras tareas se interrumpan.

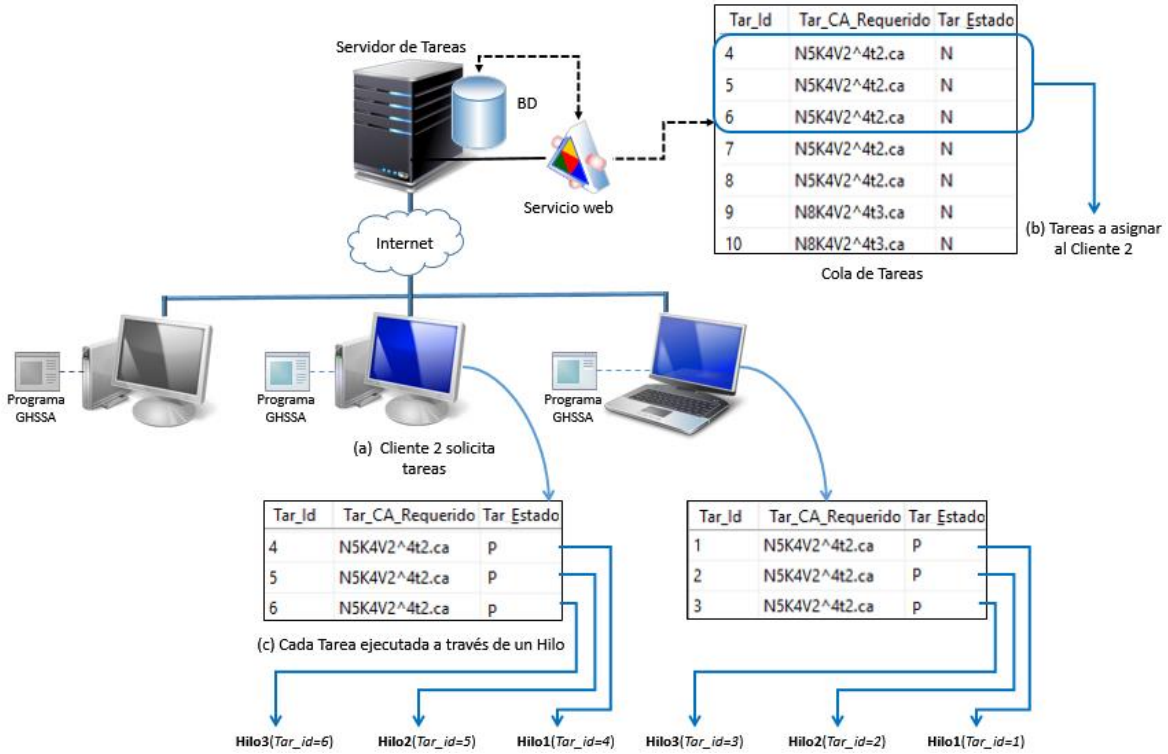


Figura 4 Asignación de Tareas al Cliente 2

Cuando el Cliente 3 que se muestra en la Figura 5(a) solicita asignación de tareas, el servicio web las asignará de acuerdo con las tareas disponibles en la cola de tareas y al número establecido para asignar. En la Figura 5(b), 3 son el número de tareas a asignar al Cliente 3 y de igual manera un hilo se encargará de la ejecución de cada tarea. Este cliente tiene 2 tareas (hilos) que buscan resolver el mismo CA de los clientes 1y 2 pero una tarea (hilo) adicional que busca resolver un CA diferente.

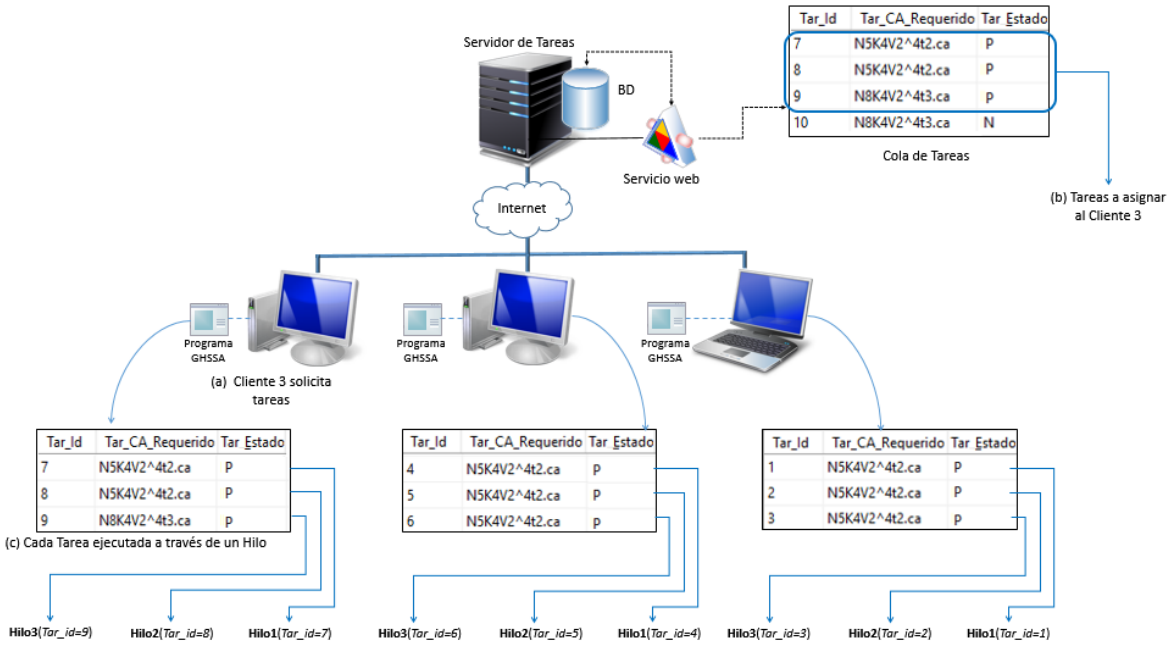


Figura 5 Asignación de Tareas al Cliente 3

En resumen, en la Figura 5, se puede observar que el CA N5K4V2^4t2.ca está siendo ejecutado paralelamente en 3 equipos Cliente diferentes y en 8 tareas independientes con Tar_id de 1 hasta 8. Además, el CA N8K4V2^4t2.ca se está buscando sólo en el Cliente 3 con 1 tarea (Tar_id = 9) o hilo de ejecución.

En la Figura 6, el hilo2 es el encargado de correr el algoritmo GHSSA con toda la configuración especificada para el Tar_id = 8. Durante la ejecución interna del algoritmo, constantemente se está preguntando si el fitness del CA requerido es cero. Si es cero y como se muestra en la Figura 6(a), se envía una petición al servicio web para que se registre un estado de éxito para la tarea y termina la ejecución. De la misma manera y como se muestra en la Figura 6(b), constantemente se está preguntado si otro proceso que esté corriendo la misma configuración del CA ya encontró el CA requerido. Si es así, se envía una petición al servicio web para que se registre un estado de interrupción para la tarea y se termina la ejecución.

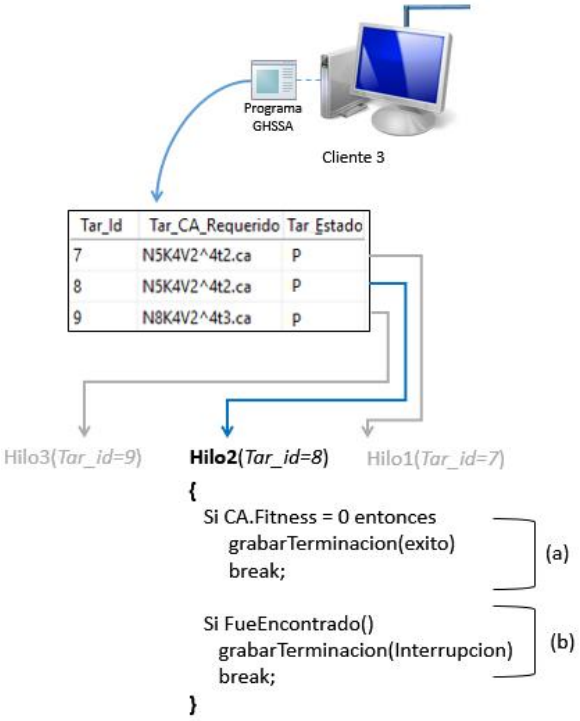


Figura 6 Ejecución de un Hilo para el CA de configuración N5K4V2^4t2.ca

Cuando una tarea específica encuentra que el fitness del CA a construir es cero, se notifica al servicio web que el estado de la tarea es exitoso y de esta manera el servicio web actualizará en la base de datos los elementos de la tarea a saber: el Tar_Estado, el Tar_CA_Encontrado, Tar_Fecha, Tar_Faltantes, Tar_Tiempo. El resto de tareas que estaban corriendo para construir la misma configuración de CA quedan con estado Interrumpido. Como la configuración del CA N5K4V2^4t2.ca es un CA pequeño y fácil de construir, cada uno de los hilos que ejecuta el algoritmo GHSSA con dicha configuración encuentra en paralelo la solución y todos reportan estado exitoso. En la Figura 7 se muestra lo anteriormente descrito.

Tar_Id	Tar_CA_Reque...	Tar_Algoritmo	Tar_Parametros	Tar_Semilla	Tar_Estado	Tar_CA_Encontrado	Tar_Fecha	Tar_Faltantes	Tar_tiempo
1	N5K4V2^4t2.ca	GHSSA	5000 5000	1	S	1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1	2017-05-29 11:22:13.000	0	78,132500
2	N5K4V2^4t2.ca	GHSSA	5000 5000	2	S	1 1 1 0 1 0 0 1 0 1 1 1 0 1 0 0 0 0 1 0	2017-05-29 11:22:13.000	0	62,507900
3	N5K4V2^4t2.ca	GHSSA	5000 5000	3	S	1 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 0 0 0	2017-05-29 11:22:13.000	0	62,507900
4	N5K4V2^4t2.ca	GHSSA	5000 5000	4	S	1 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 0 1 0	2017-05-29 11:22:13.000	0	62,507900
5	N5K4V2^4t2.ca	GHSSA	5000 5000	5	S	1 1 1 1 1 1 0 0 1 0 1 0 0 1 1 0 0 0 0 1	2017-05-29 11:22:13.000	0	22,133200
6	N5K4V2^4t2.ca	GHSSA	5000 5000	6	S	1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1	2017-05-29 11:22:13.000	0	22,133200
7	N5K4V2^4t2.ca	GHSSA	5000 5000	7	S	1 1 0 1 1 0 1 1 1 0 0 0 0 1 1 0 0 0 0 1	2017-05-29 11:22:13.000	0	19,631700
8	N5K4V2^4t2.ca	GHSSA	5000 5000	8	S	1 1 0 1 1 0 1 0 0 1 1 0 0 0 1 1 0 0 0 0	2017-05-29 11:22:13.000	0	15,629300

Figura 7 Resultados en la base de datos para el CA de configuración N5K4V2^4t2.ca

Anexo B

- Artículo publicado en una revista indexada categoría A2 según PUBLINDEX de Colciencias con la siguiente referencia: J.Timana-Peña, C.Cobos-Lozada, J.Torres-Jimenez."Metaheuristic algorithms for building Covering Arrays: A review".Revista Facultad de Ingeniería (Rev. Fac. Ing.) Vol. 25 (43), pp. 31-45. Septiembre-Diciembre, 2016. Tunja-Boyacá, Colombia. ISSN Impreso 0121-1129,ISSN Online 2357-5328.

- Artículo titulado: “Algoritmo Híbrido para construir Covering Arrays de fuerza variable basado en la Mejor Búsqueda Armónica Global y Recocido Simulado” el cual resume el desarrollo de toda la investigación y los resultados obtenidos, el cuál será presentado en el evento “European Conference on the Applications of Evolutionary Computation EvoApplications – LNCS” a desarrollarse en Parma, Italia en el mes de abril de 2018.

Anexo C

Código fuente del algoritmo GHHSA desarrollado en:

- Entorno de Programación Microsoft Visual Studio
  Microsoft Visual Studio Enterprise 2015
  Versión 14.0.25431.01 Update 3
  Microsoft .NET Framework
  Versión 4.7.02053

Base de datos implementada en:

- Microsoft SQL Server 2014.

El código fuente se anexa en el CD.