

# Capítulo 1

## **HERRAMIENTA PARA IMPLICACIONES**

El PFC *Algoritmos para la manipulación de implicaciones en ACF* tiene como objetivo la implementación de algoritmos relacionados con cierres y cálculo de bases y bases directas y realizar comparativas entre ellos.

Por esto, surge la idea de desarrollar una herramienta que permita ejecutar los algoritmos implementados, medir su rendimiento y guardar los resultados y estadísticas de éstos permitiendo compararlos entre sí. A esta herramienta se le ha llamado IS Bench. Además, debido a que no existe un benchmark (banco de pruebas) al estilo de los que se utilizan en lógica para la comparación de algoritmos, se hace necesario un generador de conjuntos de implicaciones aleatorios que pueda servir de entrada a los algoritmos en cuestión.

## 1.1. IS Bench

*IS Bench* es una aplicación cuyo objetivo es ejecutar algoritmos sobre sistemas implicacionales, guardar sus resultados y compararlos posteriormente. Esta herramienta se pretende ofrecer a la comunidad FCA, para incorporarla en un futuro cercano a la librería de la Dra. K.Bertet para comparativas de algoritmos relacionados con las bases.

La información se organiza en *workspaces*, que no son más que directorios que contendrán los algoritmos y benchmarks registrados y ejecutados en él, así como los resultados (sistemas de salida, trazas y estadísticas) generados.

Un *benchmark* o *experimento* es un conjunto de algoritmos que se ejecutan con uno o varios sistemas implicacionales de entrada comunes.

Un *algoritmo* es una implementación Java que recibe como entrada un sistema implicacional y devuelve otro equivalente. Los algoritmos implementados inicialmente son algoritmos de cálculo de bases directas-optimales, pero la API no restringe su uso a este tipo de bases ya que la única condición es que su entrada sea un sistema implicacional al igual que la salida.

El menú principal consta de las opciones:

- Preferencias → Workspaces, para la configuración de workspaces.
- Help con la ayuda de usuario.

El funcionamiento básico de la aplicación se resume en registro benchmarks, ejecución de éstos y visualización de resultados. Por esto la aplicación se divide en tres zonas principales: *Inicio*, *Benchmarks* y *Resultados*.

Con el área *Inicio* se pretende que el usuario pueda recordar las últimas acciones en el workspace actual al entrar en la aplicación.

Se muestra un resumen de las últimas ejecuciones que contiene el nombre del benchmark, la fecha y hora de la última ejecución y un enlace para ejecutarlo de nuevo. La pestaña *Benchmarks*, es el área en el que se definen y ejecutan los benchmarks. Se divide a su vez en dos pestañas: *Nuevo* y *Ejecutar*. Como sus nombres indican, en la pestaña *Nuevo* se pueden registrar nuevos benchmarks y en *Ejecutar*, ejecutarlos.

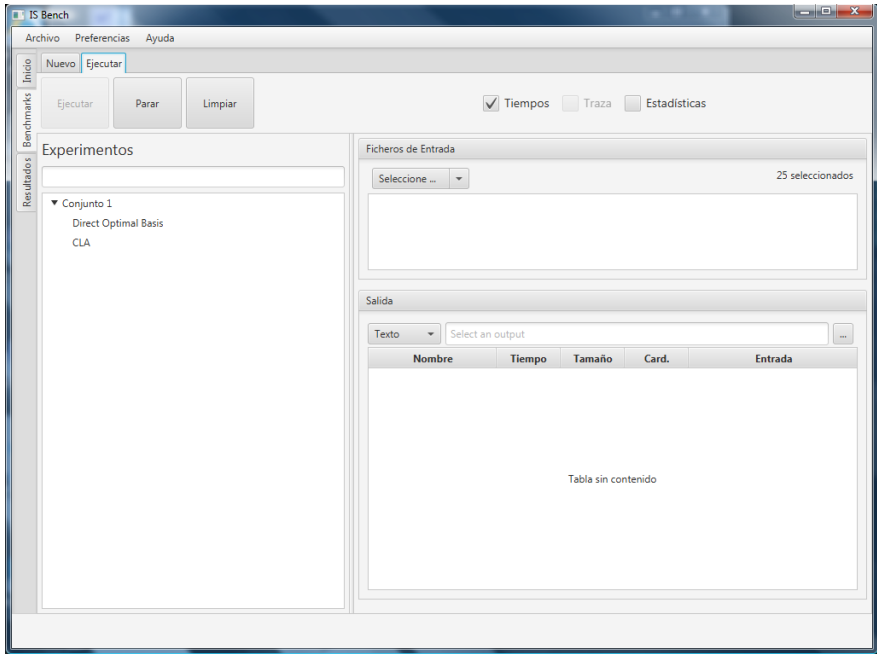


Figura 1.1: Pestaña Benchmarks

El área *Resultados*, contendrá los resultados detallados de los benchmarks del workspace actual, y será la zona que el usuario utilizará para realizar sus comparativas.

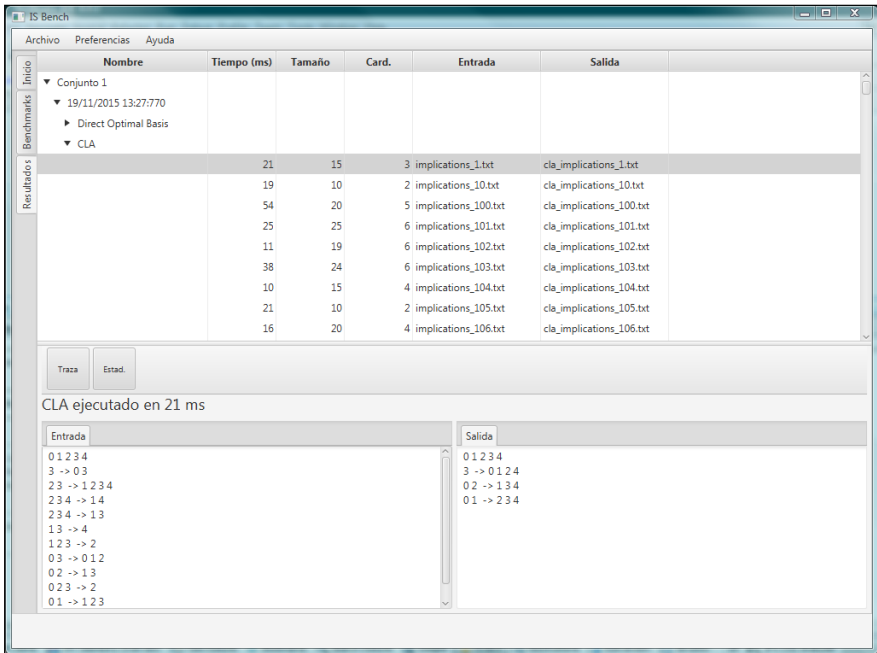


Figura 1.2: Pestaña Resultados

### 1.1.1. Workspaces

La configuración de la aplicación está organizada en *workspaces* que no son más que directorios en los que se guardan archivos relacionados para su uso en IS Bench.

Estos archivos pueden ser:

- Registro de benchmarks.
- Archivos con sistemas implicacionales de entrada para los algoritmos a ejecutar.
- Archivos con los sistemas implicacionales de salida de los algoritmos ejecutados.
- Resultados.
- Librerías de algoritmos.

La primera vez que se arranca la aplicación, se crea el archivo `isbench.properties` en el directorio *isbench* del *home* del usuario. En Windows por ejemplo sería `C:\Users\usuario\isbench`.

Este archivo contiene las rutas de los workspaces registrados y el último workspace con el que el usuario entró a la aplicación. Por ejemplo:

```
default=C:\\Users\\Usuario\\isbench\\default  
workspace.current=default
```

Figura 1.3: `isbench.properties`

Como se ve en la Figura 1.4 la carpeta de un workspace contiene los benchmarks registrados en él, la carpeta *lib* donde se pueden añadir librerías con implementaciones de algoritmos externas y un archivo de preferencias. También contiene un directorio por benchmark registrado que almacenará las entradas y salidas de sus ejecuciones.

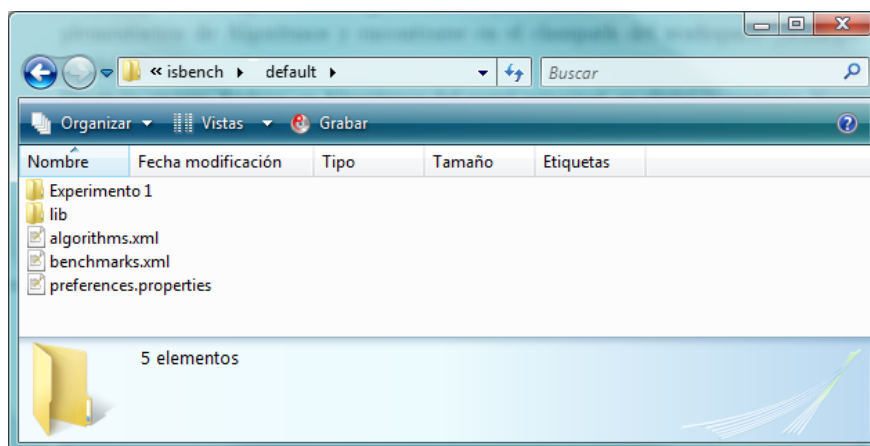


Figura 1.4: Contenido de un Workspace

La primera vez que el usuario arranca la aplicación, se establece un workspace por defecto en `[home_usuario]\isbench\default`. Desde la ventana *Workspaces*, que se puede acceder desde la opción de menú *Preferencias* → *Workspaces*, el usuario puede:

- Consultar el workspace actual y su configuración.
- Cambiar de workspace.
- Crear un nuevo workspace.

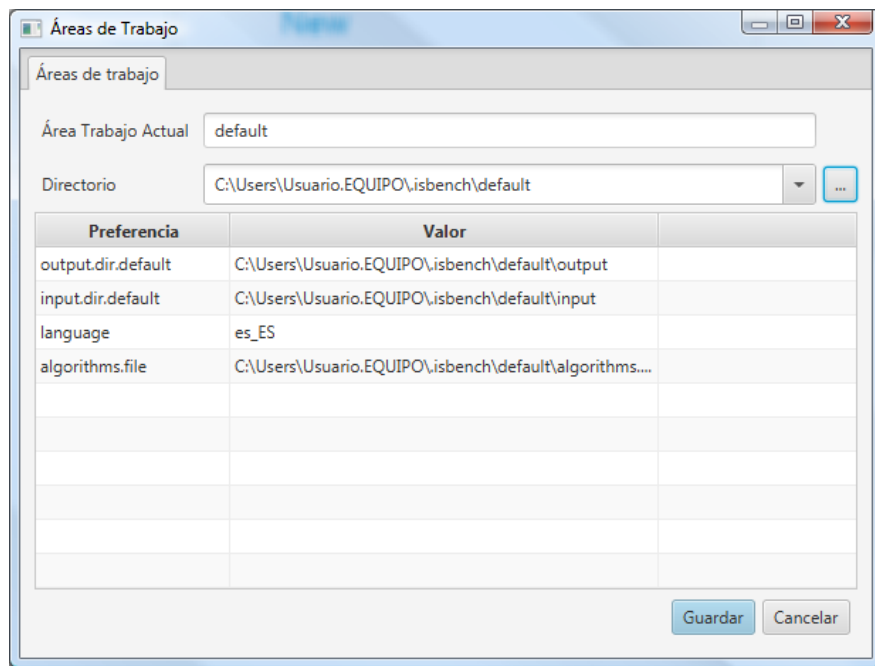



Figura 1.5: Ventana Áreas de Trabajo

El campo *Directorio* es un desplegable editable, que contiene como ítems los workspaces existentes.

Para cambiar de workspace, el usuario sólo tiene que seleccionar uno de los ítems del desplegable.

Para crear uno nuevo, sólo ha de introducir la ruta absoluta del directorio en el que quiere crearlo o seleccionarlo con el buscador pulsando en el botón .

### 1.1.2. Registrar Benchmarks

Un benchmark es la agrupación de un conjunto de algoritmos que se ejecutan con una misma entrada para la posterior comparación de sus resultados.

Para poder ejecutar un benchmark éste ha debido ser registrado en el workspace y esto se puede hacer desde la pestaña *Nuevo* del área *Benchmarks*.

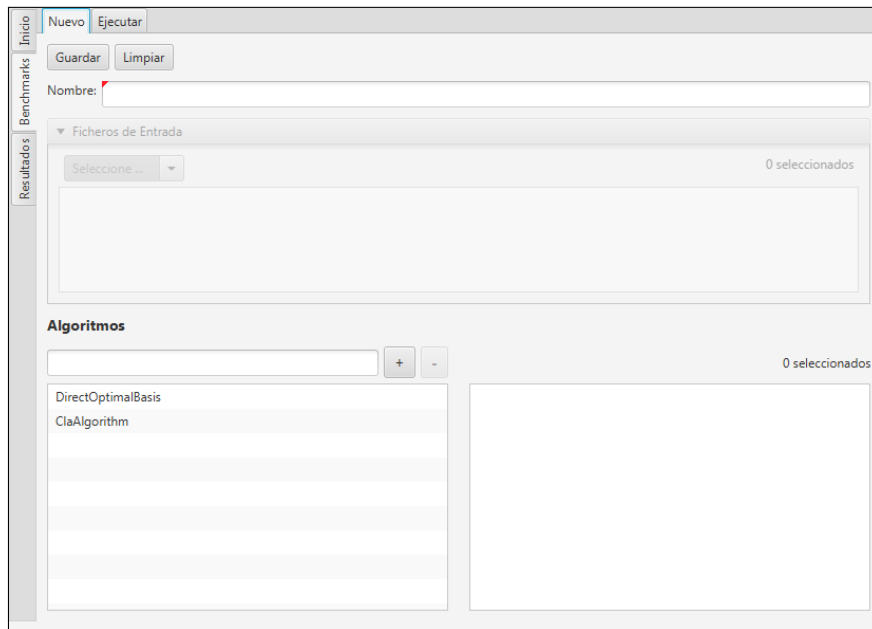




Figura 1.6: Pestaña Nuevo Benchmark

En ésta se muestra un formulario con los campos que se ven en la imagen:

- **Nombre:** Obligatorio y único. Nombre identificativo del benchmark.  
Es el nombre que se muestra cuando se carga en listas o tablas.
- **Ficheros de Entrada:** Lista de archivos que contienen los sistemas implicacionales de entrada.  
Este campo se puede introducir manualmente, seleccionándolo mediante el explorador (opción *File* del botón ...) o creándolo con el generador de implicaciones (opción *Random* del botón ...).
- **Algoritmos:** Obligatorio. Hay dos listas. La de la izquierda contiene los algoritmos encontrados en la carpeta *lib* del workspace y los algoritmos implementados en este proyecto, CLA y Direct Optimal Basis que se incluyen por defecto. En la de la derecha se irán añadiendo los algoritmos seleccionados con doble click en la primera lista.  
Para filtrar la lista de algoritmos disponibles se puede hacer uso del filtro situado en la parte superior de la primera lista.

Con los botones  y , se pueden añadir y quitar librerías de algoritmos al workspace. Estos algoritmos deben cumplir las especificaciones que se indican en la sección 1.4.

Si se pulsa el botón *Guardar* con algún campo obligatorio sin completar, la aplicación

muestra un mensaje informando de ello y los campos *Nombre* y *Entrada* se marcan como erróneos.

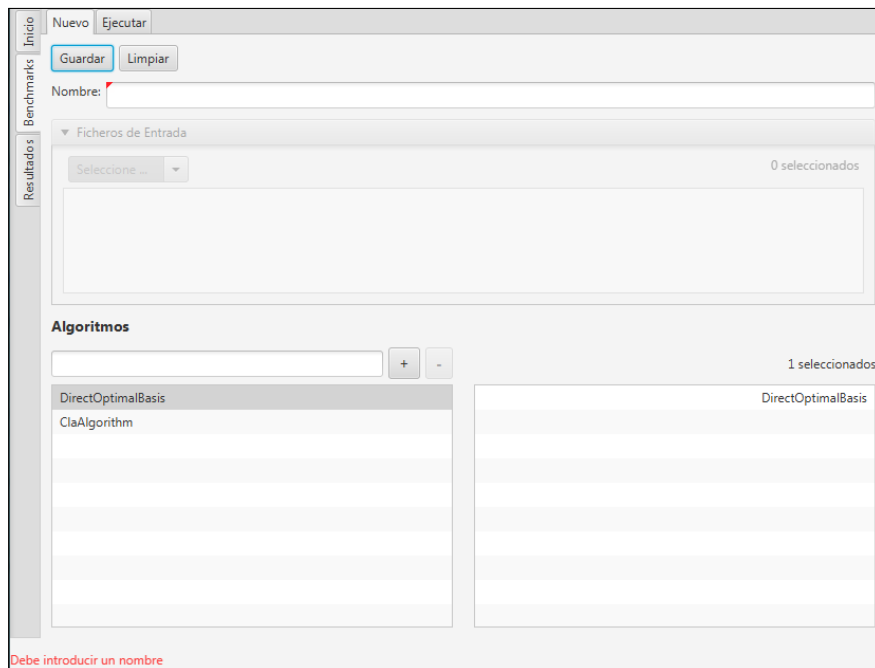


Figura 1.7: Error de validación del campo Nombre

Una vez introducidos todos los valores correctamente, al pulsar el botón *Guardar*, el benchmark se registra en el archivo `[workspace_dir]/benchmarks.xml`:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<benchmarks>
  <benchmark name="Experimento 1" workspace="C:\Users\Usuario.EQUIPO\.isbench\default">
    <inputsDir>C:\Users\Usuario.EQUIPO\.isbench\default\Experimento 1\input</inputsDir>
    <algorithms>
      <algorithm>
        <name>Direct Optimal Basis</name>
        <shortName>do</shortName>
        <type>es.uma.pfc.is.algorithms.optbasis.DirectOptimalBasis</type>
      </algorithm>
    </algorithms>
  </benchmark>
</benchmarks>
```

Figura 1.8: benchmarks.xml

### 1.1.3. Ejecutar algoritmos: Modos de ejecución

Recordemos que el objetivo de la ejecución de un algoritmo es, además de obtener un sistema implicacional de salida, obtener trazas y estadísticas que proporcionen información adicional para su comparativa.

Para lo que se distinguen tres modos de ejecución:

- **Tiempos:** Se mide el tiempo de ejecución del algoritmo y se registra.

- **Con traza:** En este modo, se genera el archivo *[nombre\_archivo\_salida]\_history.log*, que contiene la traza de la ejecución.

Esta traza será la que el desarrollador, en el momento de la implementación del algoritmo, considere que pueda ser de interés para el investigador.

*[nombre\_archivo\_salida]* es el nombre base del archivo seleccionado para la salida del algoritmo. P.e., si el archivo que se ha tomado como salida es *do\_output.txt*, el archivo de traza será *do\_output\_history.log*.

- **Con Estadísticas:** En este modo, se genera el archivo *[nombre\_archivo\_salida].csv*, en el que se guarda la evolución de los tamaños del sistema implicacional procesado. Esto se hace al final de la ejecución de cada algoritmo y se miden el tamaño y la cardinalidad del sistema obtenido.

*[nombre\_archivo\_salida]* es el nombre base del archivo seleccionado para la salida del algoritmo. Por ejemplo, si el archivo que se ha tomado como salida es *do\_output.txt*, el archivo con las estadísticas será *do\_output.csv*.

La información se guarda en archivos .csv para facilitar su visualización mediante tablas y gráficos.

Estos modos no son excluyentes y pueden combinarse como el usuario considere necesario.

#### 1.1.4. Ejecutar Benchmarks

La ejecución de un benchmark consiste en la ejecución secuencial del conjunto de algoritmos que lo componen con uno o varios sistemas implicacionales de entrada en común.

Se genera un archivo de salida y varios adicionales según el modo de ejecución, por algoritmo ejecutado, además de un resumen de dicha ejecución con:

- La fecha y hora de ejecución
- Algoritmo ejecutado
- Sistema implicacional de entrada
- Sistema implicacional de salida
- Rutas de trazas y estadísticas generadas

Esta información se usará para la consulta de resultados.

La ejecución de benchmarks se hace desde la pestaña *Ejecutar* del área *Benchmarks*.



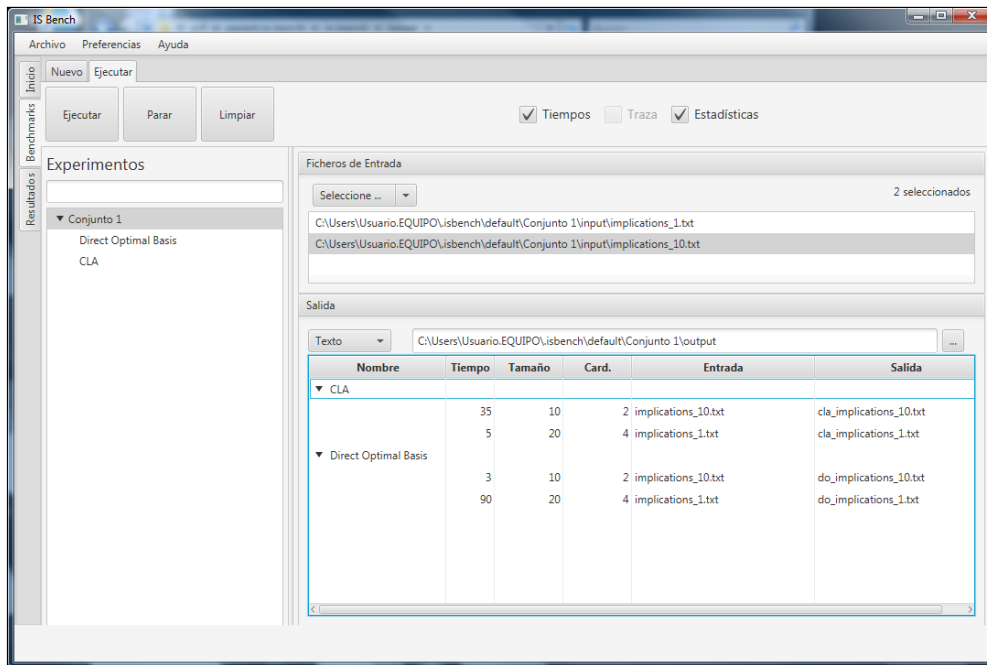


Figura 1.9: Pestaña Ejecutar Benchmark

Como se ve en la imagen, esta pestaña se compone de:

- Barra de herramientas
- Listado de Benchmarks
- Formulario para introducción de parámetros
- Visor de resultados


La barra de herramientas contiene tres botones *Ejecutar*, que ejecuta un benchmark o algoritmo seleccionado, *Parar* que para la ejecución en curso y *Limpiar* que limpia tanto la selección en el árbol como el formulario.

En la zona izquierda, se encuentra un árbol de dos niveles que contiene los benchmarks registrados en el workspace actual en el primer nivel, y los algoritmos que forman el benchmark en el segundo nivel. El contenido de éste árbol puede ser filtrado con el filtro situado en la parte superior de éste.

El formulario de introducción de parámetros consta de:

- **Modo de ejecución:** Tres modos posibles *Tiempos*, *Traza* y *Estadísticas* no excluyentes. Si se selecciona un benchmark en el árbol, el modo *Traza* se deshabilita.

Si se selecciona un algoritmo, los tres modos de ejecución están disponibles.

- **Ficheros de entrada:** Lista de los ficheros que contienen los sistemas implicacionales que servirán como entrada a los algoritmos a ejecutar. Para seleccionarlos, en el botón *Seleccionar Entrada* se presentan dos opciones: *Fichero* para seleccionar un fichero existente y *Aleatorio* para generar un sistema implicacional aleatorio con el Generador de Implicaciones.
- **Salida:** Directorio en el que se guardarán los archivos resultantes de la ejecución. La ruta se puede introducir manualmente o seleccionándolo pulsando el botón . La salida se puede generar en dos formatos: texto y prolog, que se pueden seleccionar en el desplegable que se encuentra a la izquierda del campo que contiene la ruta. El formato texto es la utilizada por defecto en la librería *java-lattice*.

Los archivos generados tomarán nombres por defecto:

- de salida: [abreviatura\_algoritmo]\_[nombre\_archivo\_entrada].txt
- traza: [abreviatura\_algoritmo]\_trace.log
- estadísticas: [nombre\_algoritmo]\_[fecha\_hora\_ejecucion].csv, donde [fecha\_hora\_ejecucion] es el momento de la ejecución con formato *yyyymmddhhMMss*.

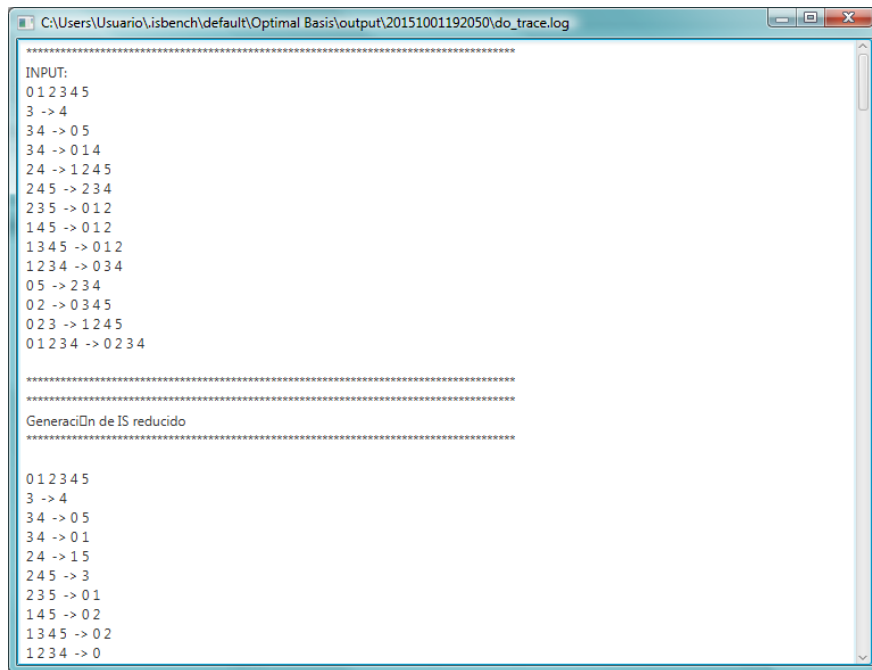
Bajo el campo *Salida*, se encuentra la tabla de resultados que contendrá los resultados de la ejecución actual agrupados por algoritmo. El contenido de cada una de las columnas es:

- **Nombre:** Sólo tiene valor en el primer nivel, y contiene el nombre del algoritmo.
- **Tiempo:** Tiempo de ejecución.
- **Tamaño:** Tamaño del sistema implicacional de salida.
- **Card.:** Cardinalidad del sistema implicacional de salida.
- **Entrada:** Nombre del archivo que contiene el sistema implicacional de entrada.
- **Entrada:** Nombre del archivo que contiene el sistema implicacional de salida.

Al hacer click con el botón derecho sobre alguno de los resultados, se muestra un menú contextual con las siguientes opciones:

- **Mostrar Log:** Muestra la traza generada si se ha ejecutado con el modo *Tiempos* activo como se ve en la Figura 1.10. También se podrá ver dicha traza haciendo doble click sobre el resultado.

- **Fichero de Estadísticas:** Abre el archivo de estadísticas generado si se ha ejecutado con el modo *Estadísticas* activo.



```

C:\Users\Usuario\isbench\default\Optimal Basis\output\20151001192050\do_trace.log

*****
INPUT:
0 1 2 3 4 5
3 -> 4
3 4 -> 0 5
3 4 -> 0 1 4
2 4 -> 1 2 4 5
2 4 5 -> 2 3 4
2 3 5 -> 0 1 2
1 4 5 -> 0 1 2
1 3 4 5 -> 0 1 2
1 2 3 4 -> 0 3 4
0 5 -> 2 3 4
0 2 -> 0 3 4 5
0 2 3 -> 1 2 4 5
0 1 2 3 4 -> 0 2 3 4

*****
Generaci3n de IS reducido
*****

0 1 2 3 4 5
3 -> 4
3 4 -> 0 5
3 4 -> 0 1
2 4 -> 1 5
2 4 5 -> 3
2 3 5 -> 0 1
1 4 5 -> 0 2
1 3 4 5 -> 0 2
1 2 3 4 -> 0

```

Figura 1.10: Visor de traza

### 1.1.5. Consulta de Resultados

En el 3rea de *Resultados*, se pueden consultar los resultados de las ejecuciones realizadas en el workspace actual. 3stos se presentan en una tabla en forma de 3rbol agrupados por benchmark, fecha y hora de ejecuci3n y algoritmo. En el 3ltimo nivel, se muestra una fila por entrada procesada con el tiempo de ejecuci3n, el archivo de entrada y el archivo de salida como se ve en la siguiente imagen.

Nombre	Tiempo (ms)	Tamaño	Card.	Entrada	Salida
Conjunto 1					
19/11/2015 13:27:770					
Direct Optimal Basis					
CLA	21	15	3	implications_1.txt	cla_implications_1.txt
	19	10	2	implications_10.txt	cla_implications_10.txt
	54	20	5	implications_100.txt	cla_implications_100.txt
	25	25	6	implications_101.txt	cla_implications_101.txt
	11	19	6	implications_102.txt	cla_implications_102.txt
	38	24	6	implications_103.txt	cla_implications_103.txt
	10	15	4	implications_104.txt	cla_implications_104.txt
	21	10	2	implications_105.txt	cla_implications_105.txt
	16	20	4	implications_106.txt	cla_implications_106.txt

Traza Estad.

CLA ejecutado en 21 ms

Entrada

0 1 2 3 4  
3 -> 0 3  
2 3 -> 1 2 3 4  
2 3 4 -> 1 4  
2 3 4 -> 1 3  
1 3 -> 4  
1 2 3 -> 2  
0 3 -> 0 1 2  
0 2 -> 1 3  
0 2 3 -> 2  
0 1 -> 1 2 3

Salida

0 1 2 3 4  
3 -> 0 1 2 4  
0 2 -> 1 3 4  
0 1 -> 2 3 4

Figura 1.11: Resultados

Si se selecciona una de éstas filas, se muestra en la parte inferior sistemas implicacionales de entrada y salida. Además se puede consultar la traza generada y el archivo de estadísticas mediante los botones *Traza* y *Estadísticas*.

## 1.2. Generador de sistemas implicacionales aleatorios

El generador de sistemas implicacionales, permite a partir de un número de argumentos y un número de implicaciones, generar un sistema implicacional aleatorio. Se utiliza la librería *java-lattices* de la doctora Karell Bertet, para el cálculo del sistema a partir de estos parámetros.

Como se ve en la Figura 1.12, este generador es una herramienta sencilla que consta de una pantalla principal dividida en tres zonas bien diferenciadas:

- Barra de herramientas
- Panel izquierdo en el que se introducen los parámetros para la creación del sistema: número de atributos, número de implicaciones, tipo, máximo y mínimo número de atributos en premisas y conclusiones y número de sistemas.
- Panel derecho que contiene el campo *Salida* con la ruta absoluta del fichero en el que se guarda el sistema implicacional generado, y un visor en el que se muestra dicho sistema.

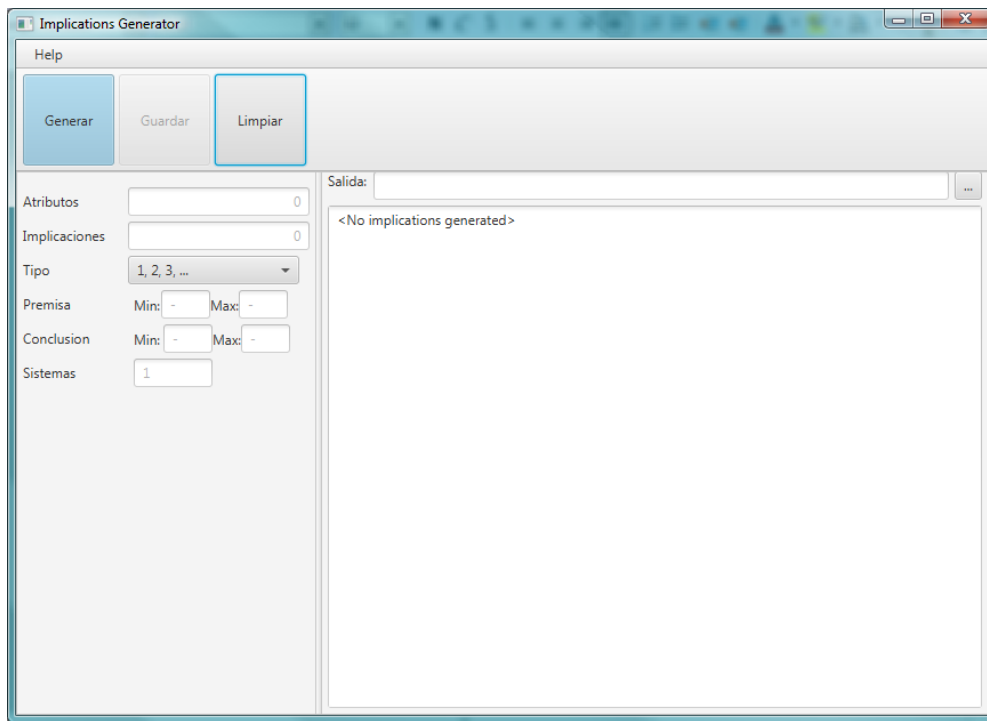


Figura 1.12: Generador de Implicaciones

### 1.2.1. Generar un sistema implicacional

Para generar un sistema implicacional se deben seguir los siguientes pasos:

1. Introducir los parámetros para el sistema.

- **Atributos:** Número de atributos.
- **Implicaciones:** Número de implicaciones.
- **Tipo:** Tipo de los atributos. Se puede seleccionar entre tres tipos: numérico (1, 2, 3,...) , alfabético (a, b, c, ...), alfanumérico (a1, a2, a3, ...).
- **Premisa:** Número mínimo y máximo de atributos en la premisa.
- **Conclusion:** Número mínimo y máximo de atributos en la conclusión.
- **Sistemas:** Número de sistemas implicacionales a generar. Por defecto, 1.

2. Pulsar el botón *Generar*. Al pulsar este botón se genera el sistema con los parámetros introducidos y se muestra en el visor.

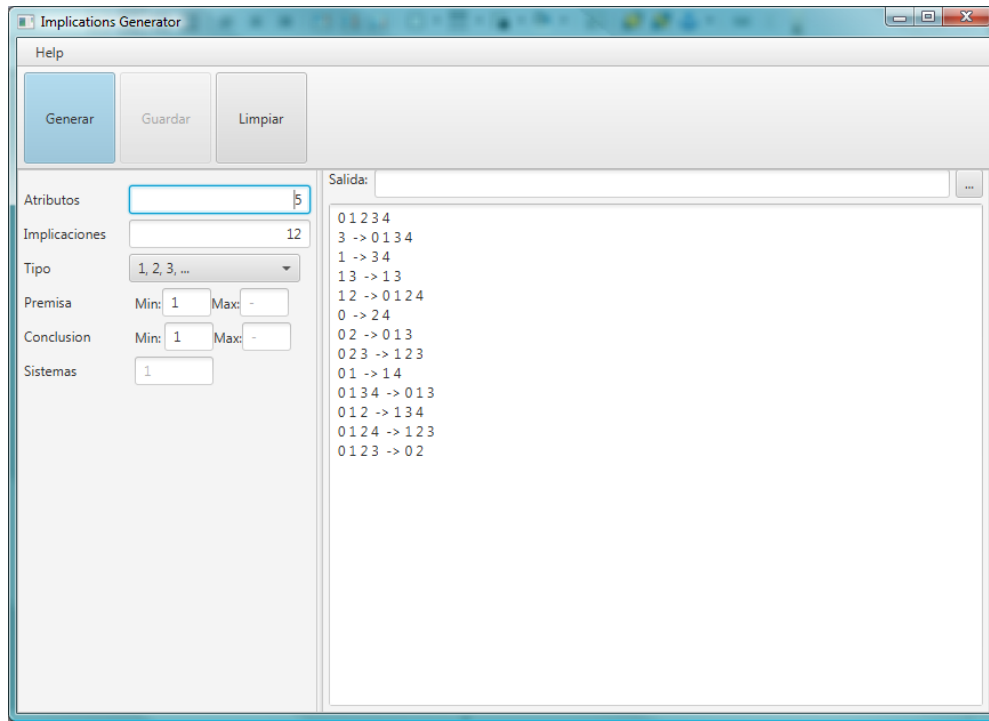



Figura 1.13: Generación de sistema aleatorio

### 3. Guardar el sistema en un fichero si se desea.

Para guardar el sistema, se debe introducir previamente la ruta absoluta del fichero en el que se desea guardar. Se puede hacer introduciéndolo directamente en el campo de texto, o buscándolo en el explorador, haciendo click en el botón *Examinar* (  ) del campo *Salida*.

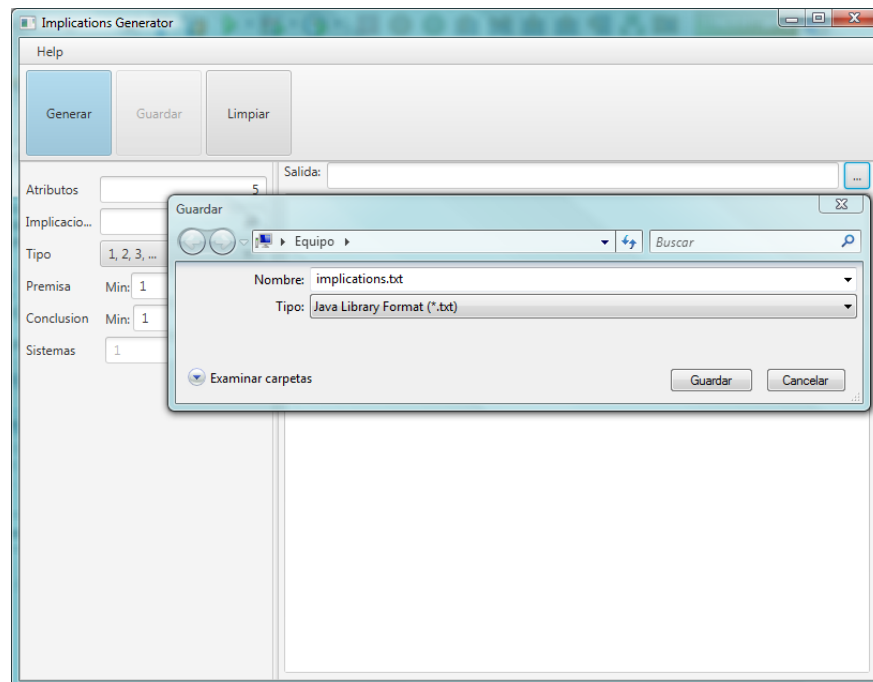


Figura 1.14: Selección de archivo de salida

El sistema se puede guardar en dos formatos: archivo de texto (txt) y archivo Prolog (pl).

Una vez introducida la ruta del archivo, el botón *Guardar* se habilitará y al pulsar sobre él se creará o actualizará el archivo con el sistema implicacional generado.

### 1.2.2. Generar $n$ sistemas implicacionales

Como se ha comentado anteriormente, es posible generar  $n$  sistemas implicacionales con los mismos parámetros. Para ello, sólo se ha de introducir en el campo *Sistemas* el número de conjuntos de implicaciones a generar. En este modo de ejecución, los conjuntos de implicaciones generados no se previsualizan y se guardan directamente en fichero. Por ello, es obligatorio que el campo *Salida* contenga la ruta absoluta del fichero base, de otra forma, el botón *Generar* no se habilitará.

A partir de esta ruta, se guardarán los  $n$  conjuntos de implicaciones generados en  $n$  archivos, tomando su nombre a partir del introducido en el campo *Salida* y añadiéndole un índice.

Por ejemplo, si el campo *Salida* contiene la ruta C:\implications\system.txt y se van a generar tres conjuntos de implicaciones, se crearan los archivos:

- C:\implications\system\_0.txt
- C:\implications\system\_1.txt
- C:\implications\system\_2.txt

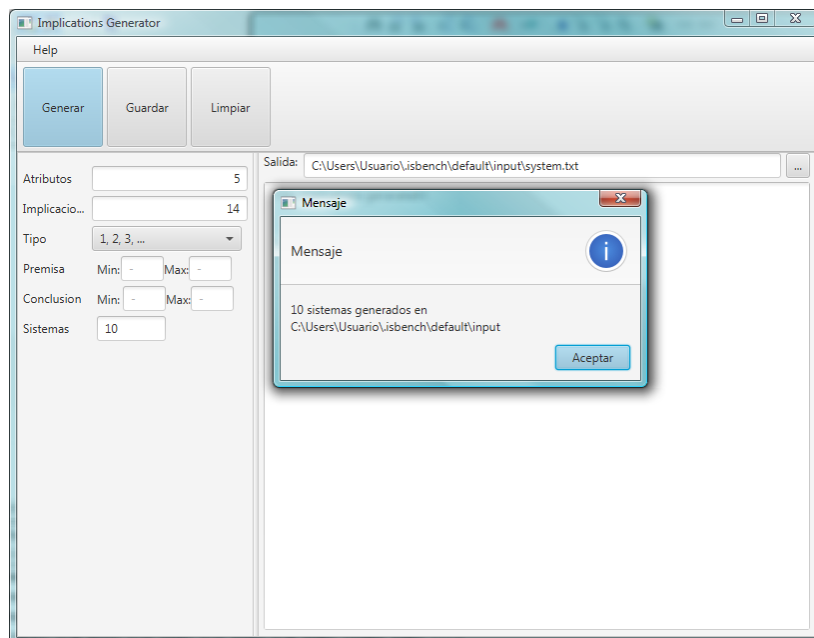


Figura 1.15: Generar  $n$  sistemas implicacionales

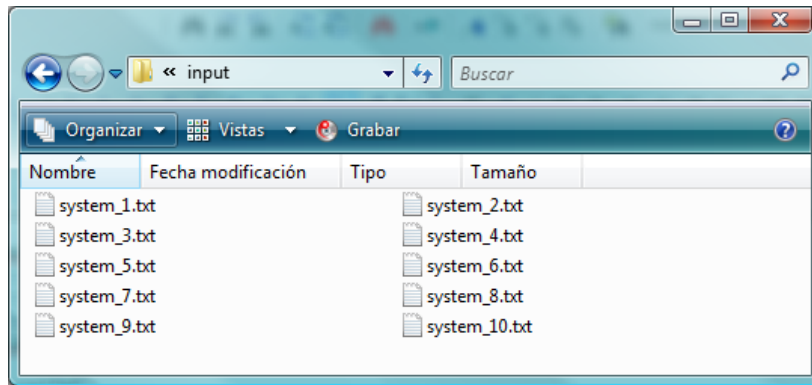


Figura 1.16: Sistemas implicacionales generados

Desde IS Bench se puede acceder al Generador de Implicaciones, para seleccionar las entradas de los benchmarks y algoritmos. Se puede hacer en:

1. el registro de Benchmarks (*Benchmarks*  $\rightarrow$  *Nuevo*).



Figura 1.17: IS aleatorio para el registro de benchmarks

2. la ejecución de benchmarks o algoritmos (*Benchmarks*  $\rightarrow$  *Ejecutar*).

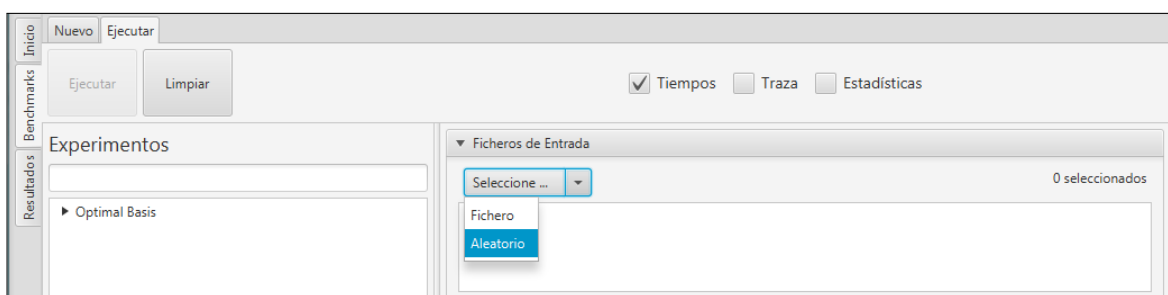


Figura 1.18: IS aleatorio para la ejecución de benchmarks o algoritmos

### 1.3. Ejemplo de uso

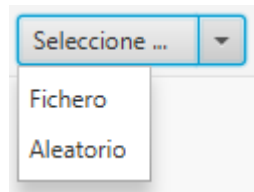
En esta sección se muestra un ejemplo de uso de la herramienta, desde que se registra el benchmark hasta la consulta de resultados, pasando por su ejecución. El ejemplo



será de un benchmark en el que se compararán los algoritmos CLA y Direct Optimal Basis con un conjunto de implicaciones aleatorias.

1. Registrar un nuevo benchmark en la pestaña *Nuevo* del área *Benchmarks*.

- a) Introducir el nombre en el campo *Nombre*.
- b) Seleccionar la opción *Aleatorio* del desplegable *Selección Entrada*.



- c) Se abre el generador de implicaciones. Introducir el número de atributos, implicaciones y conjuntos a generar en sus correspondientes campos y pulsar el botón *Generar*.
- d) Al finalizar la generación, se muestra un mensaje informando de ello. Aceptar el mensaje y cerrar el generador. Las rutas de las entradas generadas se cargan en la lista de entradas.

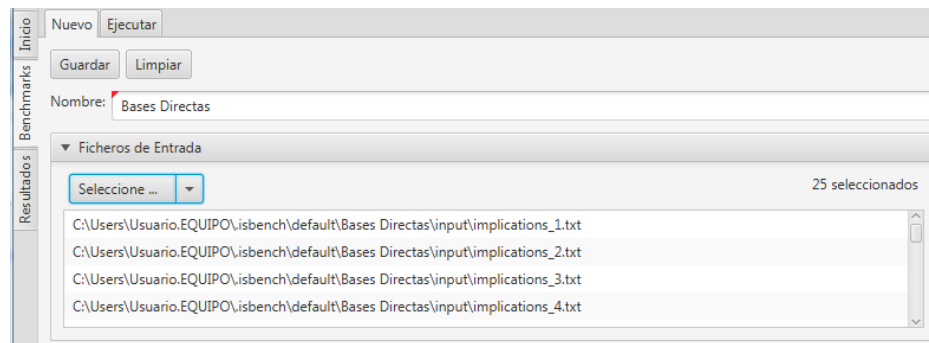


Figura 1.19: Entadas insertadas desde el Generador

- e) Seleccionar los algoritmos CLA y Direct Optimal Basis de la lista de algoritmos con doble click. Éstos se añadirán a la lista de la derecha.
  - f) Hacer click en *Guardar*. Si todo ha ido bien, se mostrará un mensaje al pie informando de ello.
2. Ejecutar el benchmark creado desde la pestaña *Ejecutar*.

- a) El benchmark registrado aparece en el árbol de benchmarks a la izquierda. Seleccionarlo haciendo click sobre él.

- b) La lista *Ficheros de Entrada* se inicializa con las entradas definidas.
- c) El campo *Salida*, se inicializa con la ruta  $[workspace\_dir]/[nombre\_benchmark]/output$ . En este caso la ruta es un directorio, ya que se generarán  $n$  salidas, una o más por algoritmo ejecutado.
- d) El modo de ejecución por defecto es *Tiempos*. Seleccionar además el modo *Estadísticas* para que calcule los tamaños y cardinalidades de los sistemas implicacionales de salida.
- e) Pulsar el botón *Run*. Aparece un indicador que estará visible durante la ejecución.

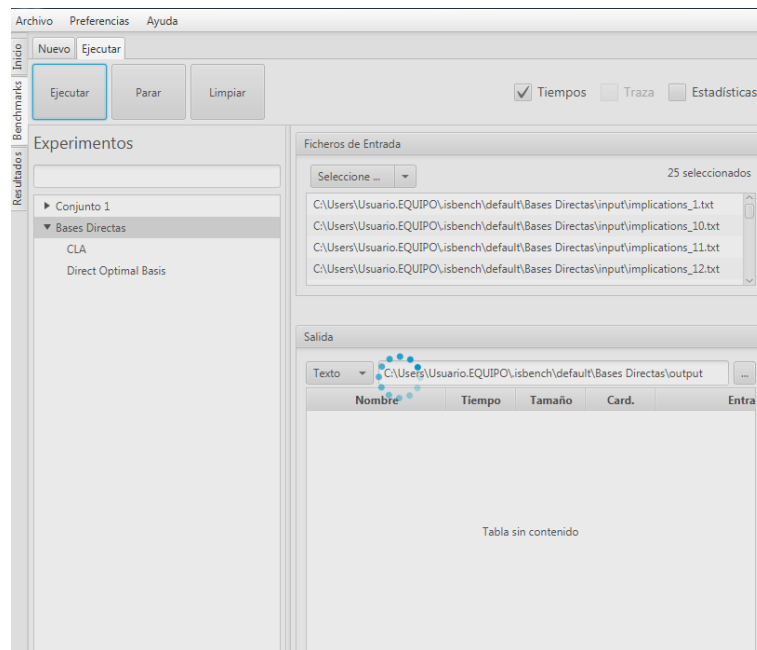


Figura 1.20: Ejecución de un benchmark

- f) Cuando finaliza la ejecución, el indicador desaparece y los resultados se muestran en la tabla de resultados.

Salida					
Texto		Select an output			
Nombre	Tiempo	Tamaño	Card.	Entrada	Salida
► Direct Optimal Basis					
▼ CLA					
	94,237	15	3	implications_1.txt	cla_implications_1.txt
	101,402	25	5	implications_10.txt	cla_implications_10.txt
	98,975	15	3	implications_11.txt	cla_implications_11.txt
	128,701	20	4	implications_12.txt	cla_implications_12.txt
	24,095	10	2	implications_13.txt	cla_implications_13.txt
	114,197	15	3	implications_14.txt	cla_implications_14.txt
	207,472	15	3	implications_15.txt	cla_implications_15.txt
	101,444	20	4	implications_16.txt	cla_implications_16.txt
	262,652	20	4	implications_17.txt	cla_implications_17.txt

Figura 1.21: Resultados de la ejecución

## 1.4. Implementación de Algoritmos

Para que terceros puedan implementar algoritmos que puedan ser ejecutados por IS Bench, se ha creado una API con las interfaces y clases básicas para este fin que se incluye en el la librería *is-algorithms*.

El requisito mínimo para que un algoritmo pueda ser ejecutado desde IS Bench, es que implemente la interfaz `es.uma.pfc.is.algorithms.Algorithm` incluida en la librería **is-algorithms-1.0.0.jar**.

La dependencia a incluir en proyectos Maven es:

```
<dependency>
    <groupId>es.uma.pfc</groupId>
    <artifactId>is-algorithms</artifactId>
    <version>1.0.0</version>
</dependency>
```

La aplicación ejecuta el método

`execute(input : ImplicationalSystem) : ImplicationalSystem` por lo que éste será en el que se implemente el algoritmo en cuestión.

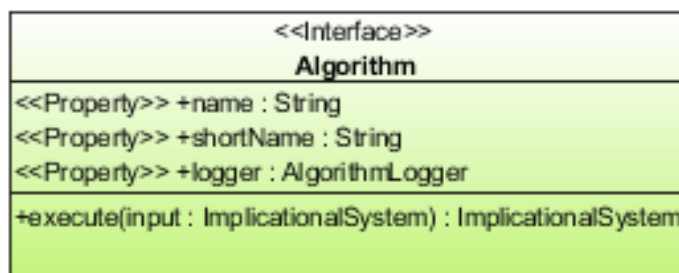


Figura 1.22: Interfaz Algorithm

La API proporciona una implementación abstracta de esta interfaz, `es.uma.pfc.is.algorithms.GenericAlgorithm`, que implementa los *getters* y *setters* obligatorios, así como utilidades para la sustitución, adición y eliminación de implicaciones del sistema. El método `execute(input : ImplicationalSystem) : ImplicationalSystem` no se implementa, ya que esto se deberá hacer en la implementación final.

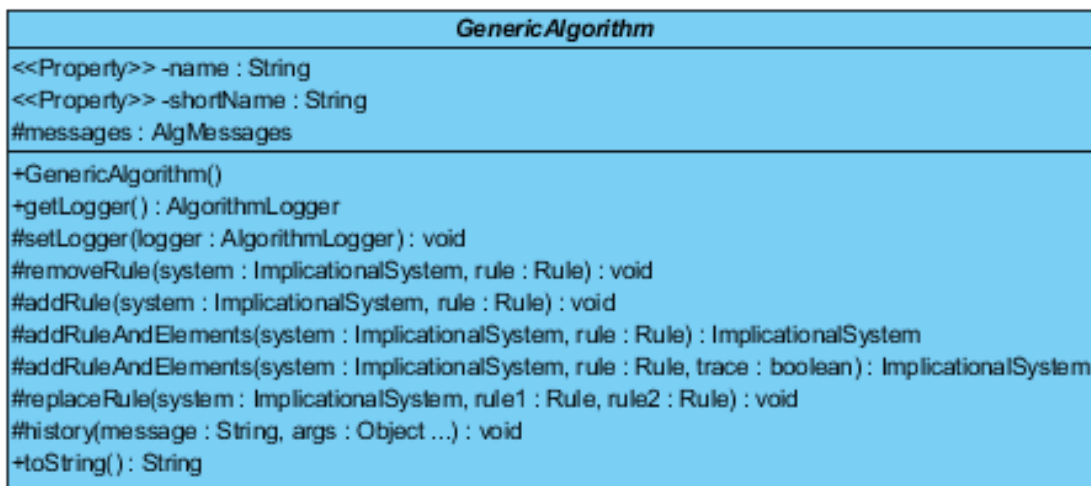



Figura 1.23: Clase abstracta GenericAlgorithm

En el apartado ?? se incluyen los detalles de esta API.

A continuación se muestra un ejemplo de un algoritmo, que devuelve un sistema equivalente al de entrada cuyas conclusiones sólo contienen un atributo.

```
public class UnaryAlgorithm extends GenericAlgorithm {
    @Override
    public ImplicationalSystem execute(ImplicationalSystem system) {
        ImplicationalSystem outputSystem = new ImplicationalSystem(system);
        outputSystem.makeUnary();
        return outputSystem;
    }
}
```

Por lo que, para implementar un algoritmo y ejecutarlo desde IS Bench:

1. Crear un proyecto Java (JDK 1.8\_u65+) que tenga como dependencia la librería *is-algorithms*.
2. Crear una clase que implemente la interfaz `es.uma.pfc.is.algorithms.Algorithm` o extienda de `es.uma.pfc.is.algorithms.GenericAlgorithm`.
3. En el método `execute(input : ImplicationalSystem) : ImplicationalSystem` implementar el algoritmo en cuestión.
4. Compilar el proyecto y generar el JAR correspondiente.
5. Añadir la librería al workspace desde la pestaña *Nuevo* del área *Benchmarks*, con el botón .

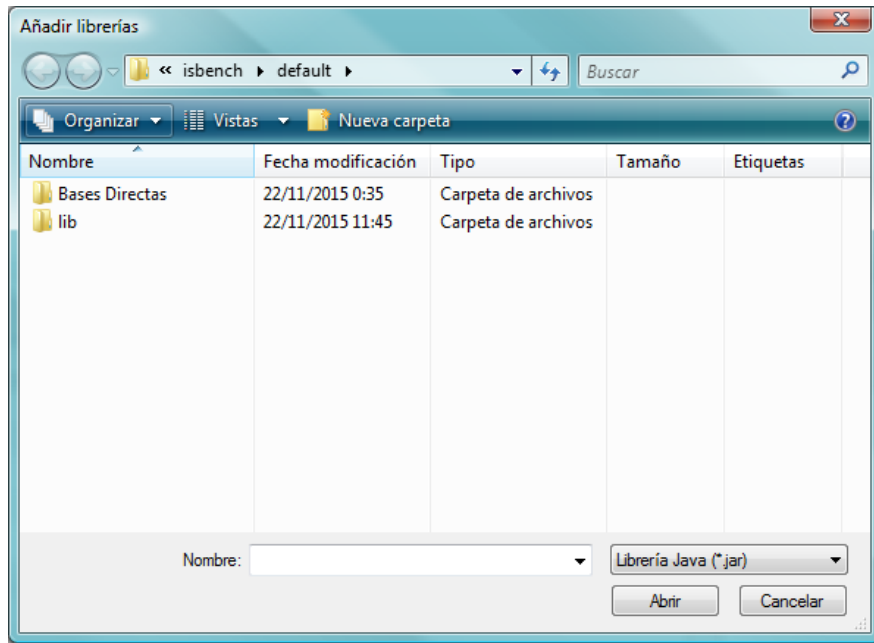


Figura 1.24: Añadir librerías

6. Incluir el algoritmo en un *benchmark* como se explica en el apartado 1.1.2, si no se ha hecho ya en el punto anterior.
7. Una vez incluido en el Benchmark, se podrá ejecutar el algoritmo bien individualmente o bien a través del Benchmark desde la pestaña *Benchmarks / Ejecución*, como se explica en 1.1.4.