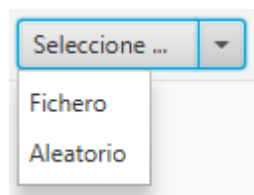


# 1. Ejemplo de uso

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.

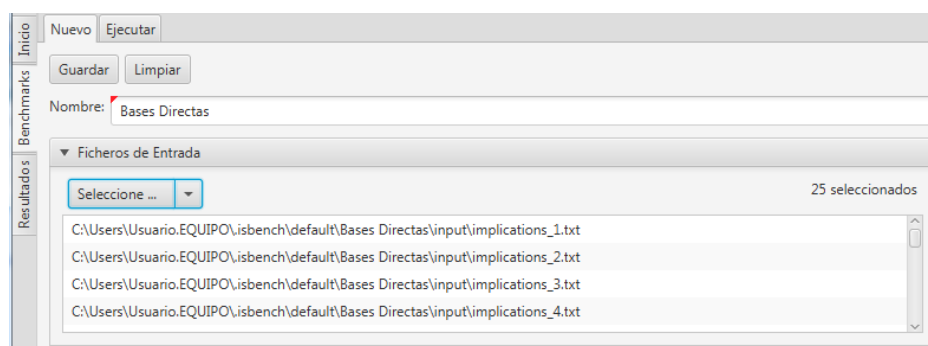


Figura 1: Entadas insertadas desde el Generador

- 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.

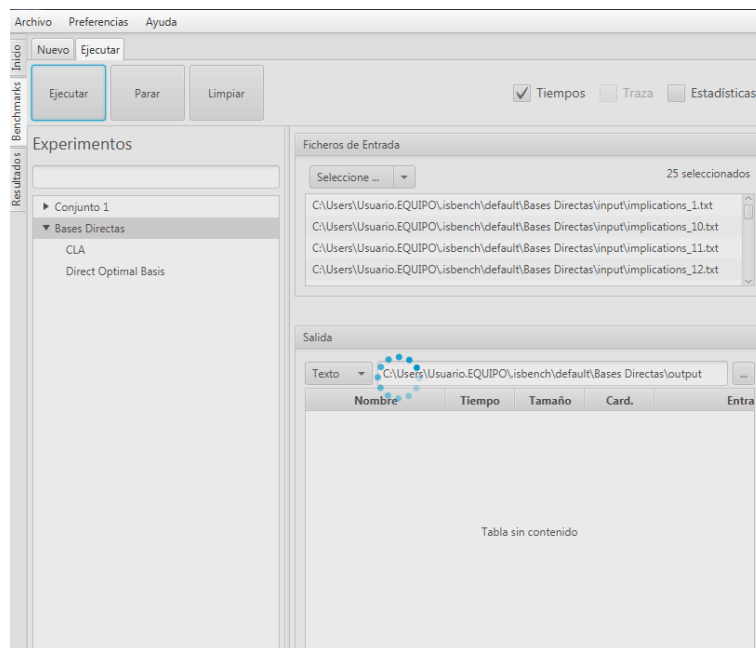


Figura 2: Ejecución de un benchmark

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

Salida					
<div> <div>Texto</div> <div>Select an output</div> <div>...</div> </div>					
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 3: Resultados de la ejecución

## 2. 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.

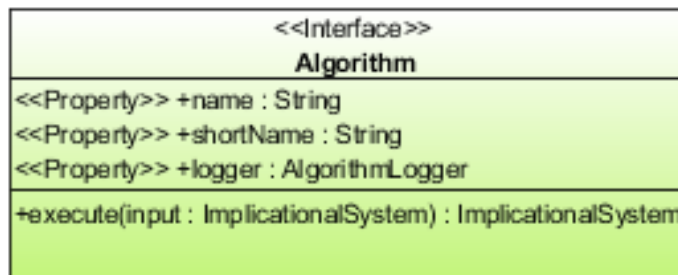


Figura 4: Interfaz Algorithm

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.

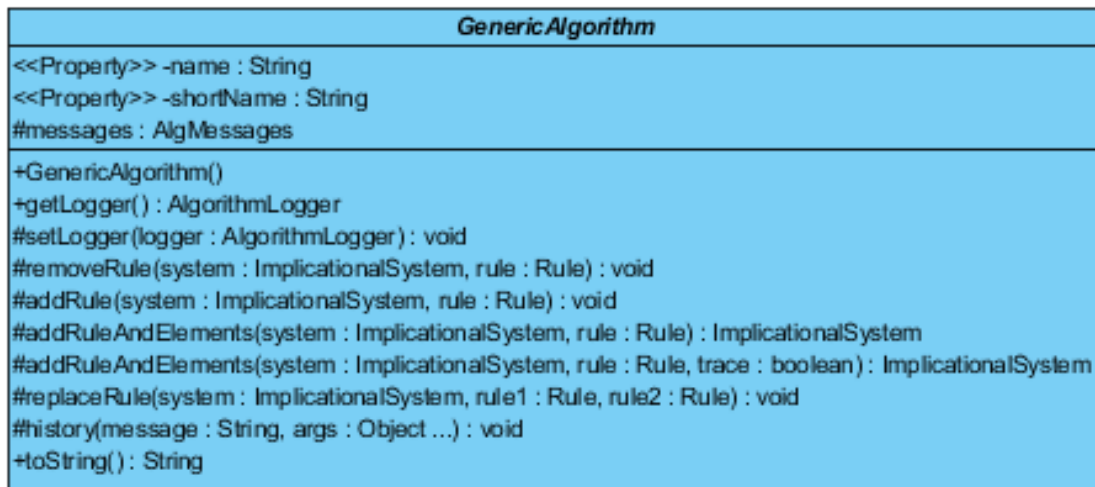



Figura 5: Clase abstracta GenericAlgorithm

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 .

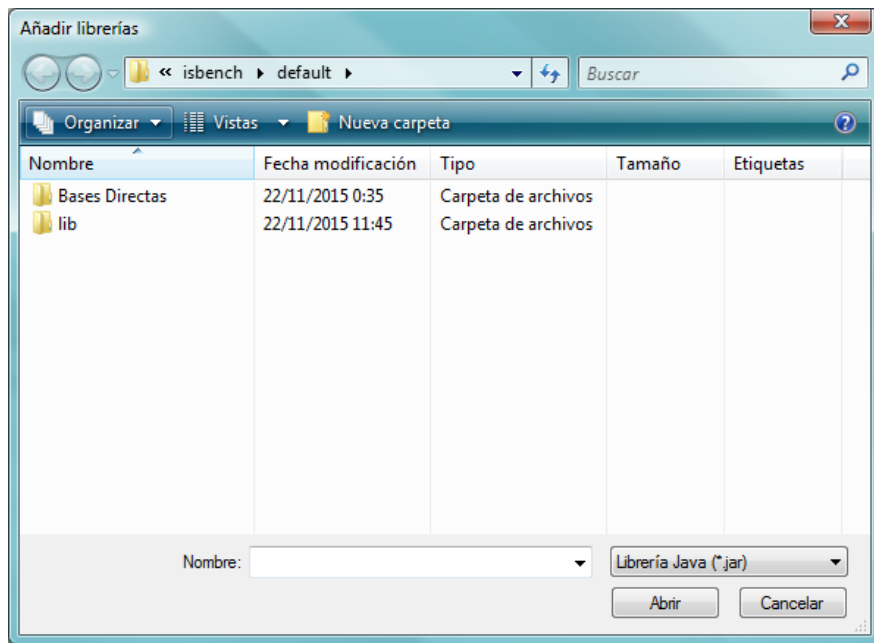


Figura 6: Añadir librerías

6. Incluir el algoritmo en un *benchmark* como se explica en el apartado ??, 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 ??.