Inteligencia Artificial I - CI5437
Profesor: Blai Bonet
Integrantes:
Ronald Becerra, 12-10706
Fabio Suárez, 12-10578
Constanza Abarca, 13-10000

Proyecto III Nonogramas

Especificaciones del entorno de prueba:

- **Memoria:** 32 GB RAM
- **Procesamiento:** Intel(R) Core(TM) i7-2920 CPU @ 2.50GHz

Especificaciones de uso

El proyecto consiste en 3 códigos principales escritos en Python, los cuales deben estar en un mismo directorio.

1. **resolverNonograma.py:** Es el programa principal que llama a las funciones de los otros archivos, y el cual es el que debe invocar el usuario pasando como parámetro el nombre del archivo de entrada. Por ejemplo, si el archivo de entrada es "100.non", el usuario debe escribir por la terminal:

   "python3 resolverNonograma.py 100.non"

El código asume que cualquier archivo que se le pase como entrada se encuentra dentro de una carpeta llamada "**nonogramas**", la cual debe encontrarse en el mismo directorio de este código.

2. **funcionesParaSATsolver.py:** Contiene las funciones que le permiten al programa interactuar con el SAT solver Glucose, pasándole un archivo como entrada con las restricciones del nonograma escritas en forma normal conjuntiva con formato Dimacs, leyendo los resultados arrojados y generando las salidas correspondientes, como la imagen .pbm que muestra las casillas en blanco y las casillas en negro.

3. **crearRestriccionesYRutas.py:** Allí se encuentra la clase "globales", que contiene todas las variables usadas tanto por este código como por los otros, como son el contador de variables del problema, el contador de cláusulas y las rutas a cada uno de los directorios empleados para ir almacenando los archivos que se van generando, y que explicaremos más abajo. También se encuentra una función que genera todas las restricciones del nonograma correspondiente y las devuelve como una única cadena de caracteres. Las restricciones de "a lo sumo 1" fueron estructuradas como aparece en las láminas de clase, tomando todos los posibles pares de variables correspondientes a la restricción.

Con la finalidad de que el usuario no tenga que intervenir durante la ejecución del sino que automáticamente se invoque al programa Glucose una vez creado el archivo con

las restricciones en forma normal conjuntiva, tuvimos que asumir que la carpeta de Glucose se encuentra en el mismo directorio que los códigos y que ésta es siempre la versión 4.1

De esta manera, al inicio son dos las carpetas que deben estar junto a los códigos: (1) "**nonogramas**", que debe incluir los casos de prueba, y (2) "**glucose-syrup-4.1**". Hay que tener cuidado al descargar esta última carpeta desde la página oficial y descomprimirla, puesto que dependiendo del sistema operativo la descompresión podría generar una subcarpeta llamada también "glucose-syrup-4.1", en la que finalmente se encuentren las carpetas necesarias. Esa subcarpeta duplicada no está incluida en la ruta considerada dentro del proyecto.

Una vez que se tenga esa carpeta descomprimida, se debe ingresar a la subcarpeta "simp" y allí ejecutar el comando "make" desde la terminal. Asumimos que el ejecutable que se crea tiene el nombre "glucose".

De todos modos, si se desea modificar estos parámetros, basta con ingresar al archivo "crearRestriccionesYRutas.py" y dentro de la clase "globales", que está casi al inicio del código, modificar la línea que dice:

```
rutaEjecutableGlucosa = rutaActual + "/glucose-syrup-4.1/simp/glucose" # >>>> Esta ruta posiblemente haya que modificarla
```

Así, podría cambiarse en esta ruta la dirección en que está el programa Glucose, además de la versión de éste en caso de usarse otra.

El programa también crea unas nuevas carpetas en el mismo directorio actual. Si ya existen, las deja como están. Dichas carpetas son:

1. **Entrada Para Glucose:** Allí se almacenan los archivos en formato .txt que serán leídos por el programa Glucose. Dichos archivos son generados por la función "generarArchivoGlucosa" que se encuentra dentro del código "funcionesParaSATsolver.py".
2. **Salida SAT Glucose:** Almacena los archivos en formato .txt con los resultados de los valores que deben tomar las variables para satisfacer las condiciones del problema. Éstos fueron generados por el programa Glucose, el cual fue invocado por la función "correrGlucose" que está dentro del código "funcionesParaSATsolver.py". Si aparece un número en negativo, es porque la variable correspondiente debe tomar el valor False; y si lo aparece en positivo, la variable debe tomar el valor True.
3. **Estadisticas-Glucose:** El SAT solver, además de dar los valores de las variables, también devuelve las estadísticas de ejecución, como el número de reinicios, el número de conflictos, el tiempo requerido, etc. Por la forma como está configurado nuestro programa, esas estadísticas no se muestran en la terminal sino que se guardan en un archivo en este directorio.
4. **txt-salida-matriz-PBM:** Una vez que el SAT solver ha determinado los valores de las variables, el programa se encarga de leerlos y generar una matriz de 0's y 1's, dependiendo de si la variable respectiva toma el valor False o True, respectivamente.Ello

se guarda en un archivo en formato .txt en este directorio. Esa matriz es la que luego permite crear la imagen con cuadros blancos y negros.

5. **PBMs:** Allí es donde finalmente estarán las salidas deseadas, que son las imágenes en formato .pbm que muestran las casillas pintadas de blanco o negro, según el valor de la variable sea False o True, respectivamente.

Resultados

No pudimos generar las estadísticas de todos los casos de prueba, ya que algunos tardaban bastante y tuvimos que parar la ejecución para continuar con los otros. Un ejemplo de ello es el archivo “meow.non”. La carpeta “Entrada Para Glucose” aparece en el repositorio pero la dejamos con un solo archivo porque todos ocupaban mucho espacio en total. Decidimos dejarla allí para que se vea que forma parte de la estructura.

Las imágenes generadas se pueden visualizar, aunque pequeñas, cuando se descargan al sistema operativo, pero no se ven como tal desde el mismo repositorio. Eso es así porque el repositorio no logra mostrar como imagen el formato .pbm, y tampoco el formato .md muestra como imagen los ceros y unos que se generaron.

A continuación colocamos como ejemplo algunas de las imágenes creadas. También subimos al repositorio una carpeta adicional llamada “JPGs” con algunas imágenes expandidas en formato .jpg.

Analizando las estadísticas obtenidas, podemos ver que para las imágenes generadas se obtuvo “SAT” lo que indica que el solver pudo hallar una solución. Mientras que las imágenes no generadas estaban tomando mucho tiempo de procesamiento debido al alto número de cláusulas y variables. Por ejemplo, para el caso del nonograma meow.non se tienen 150500 variables y 21088448 cláusulas, y tras horas de procesamiento no se había alcanzado una solución. Uno de los nonogramas más complejos resueltos tenía 4126157 cláusulas y 48572 variables, y se resolvió en 942.633 segundos de CPU.