Resolución de Hitori mediante búsqueda informada

Inteligencia Artificial (IS) 2018/19 – Propuesta de trabajo

Francisco J. Martín Mateos

1. Introducción y objetivo

El *Hitori* (ひとりにしてくれ *Hitori ni shite kure*, literalmente “déjame solo”) es uno de los pasatiempos lógicos popularizados por la revista japonesa **Nikoli** [2]. El objetivo del juego consiste en, dada una cuadrícula con cifras, determinar cuáles hay que quitar para conseguir que no haya elementos repetidos ni en las filas ni en las columnas. De esta forma, el pasatiempo comienza con una configuración como la siguiente:

2	9	8	7	4	6	4	7	6
7	4	9	2	8	3	4	3	5
4	7	5	3	6	5	6	6	5
6	1	3	7	6	9	7	2	4
1	3	3	7	2	8	6	5	1
9	8	6	2	3	8	5	5	2
8	4	7	9	3	3	2	1	6
6	2	4	1	7	4	4	9	3
8	4	1	3	5	1	9	8	1

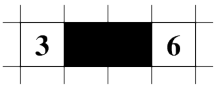
Y el objetivo consiste en eliminar algunas casillas, para obtener algo como:

2	9	8		4	6		7	
7		9	2	8		4	3	5
4	7		3		5		6	
	1	3		6	9	7	2	4
1	3		7	2	8	6	5	
9	8	6		3		5		2
8		7	9		3	2	1	6
6	2		1	7	4		9	3
	4	1		5		9	8	

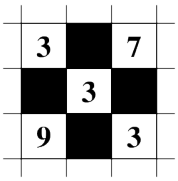
Como se puede observar, no quedan elementos repetidos en ninguna fila ni en ninguna columna.

Hay otras reglas que se deben cumplir a la hora de eliminar las casillas:

- No pueden eliminarse casillas colindantes horizontal o verticalmente. Por ejemplo, esta restricción impide que se puedan eliminar dos casillas consecutivas horizontalmente:



- Las casillas no eliminadas deben de formar una única componente conectada horizontal o verticalmente. Por ejemplo, esta restricción impide que se pueda aislar una casilla entre otras cuatro eliminadas:



También es importante tener en cuenta que el puzle tiene solución única.

La mejor forma de familiarizarse con el puzle es resolviéndolo. Existen muchas páginas web en las que se pueden resolver estos puzles de forma interactiva (http://www.menneske.no o http://www.hitoriconquest.com/), donde además se pueden aprender estrategias para resolverlos; e incluso aplicaciones para Android ('Real Hitori' en Google Play).

El **objetivo principal** de esta propuesta es la resolución de un puzle *hitori* mediante **técnicas de búsqueda**. Para ello se debe idear una representación de estos puzles como **problema de espacios de estado** e investigar cuál es la mejor forma de resolverlos. Esta investigación puede suponer definir funciones de **coste** y/o **heurística** que permitan aplicar algoritmos de búsqueda informada.

Para ello será necesario alcanzar los siguientes objetivos **específicos**:

1. Idear una representación de un puzle *hitori* como problema de espacios de estado usando la librería Python: `problema_espacio_estados.py`
2. Investigar cuál es la mejor forma de resolver este tipo de puzles, evaluando distintos algoritmos de búsqueda y funciones de coste y/o heurística para los algoritmos de búsqueda informada. Estos algoritmos se proporcionan en la librería Python: `búsqueda_espacio_estados.py`
3. Implementar una función que, a partir de una matriz de números naturales de tamaño MxN (no necesariamente cuadrada), construya el puzle *hitori* correspondiente a dicha matriz y lo resuelva mediante un algoritmo de búsqueda que también se pasa como argumento. Esta función debe devolver la matriz solución del puzle *hitori*. Por ejemplo[1],

```
In : resuelveHitori(bProfundidad,[[1,2,1],[2,2,1],[3,1,2]])
Out: [[1, 2, 0], [2, 0, 1], [3, 1, 2]]
```

4. Realizar pruebas de rendimiento del sistema para distintas configuraciones del puzle. Para ello se proporcionan dos ficheros de datos: `ejemplos_prueba.txt`, que incluye varios ejemplos de tamaños incrementales, 3x3, 4x4, 5x5 y 6x6, para realizar pruebas iniciales y

1 En este ejemplo el número 0 de la solución representa las casillas eliminadas.

evaluar la eficiencia de la solución propuesta; y `ejemplos_reto.txt`, que incluye 100 ejemplos de tamaño 9x9, cuya resolución se tendrá en cuenta para la evaluación.

5. Documentar el trabajo en un fichero con formato de artículo científico, explicando con precisión las decisiones de diseño en la representación del problema, los resultados obtenidos en las pruebas de evaluación (número de ejemplos resueltos y tiempo de respuesta).
6. Realizar una presentación (PDF, PowerPoint o similar) de los resultados obtenidos en la defensa del trabajo.

Para que el trabajo pueda ser evaluado, se deben satisfacer TODOS los objetivos específicos al completo: el trabajo debe ser original, estar correctamente implementado y funcionar perfectamente, los experimentos se deben haber llevado a cabo y analizados razonadamente, el documento debe ser completo y contener entre 8 y 10 páginas (no más, no menos), y se debe realizar la defensa con una presentación de los resultados obtenidos.

2. Descripción del trabajo

A continuación se introduce la metodología a seguir para el correcto desarrollo del trabajo.

2.1. Configuración inicial

El trabajo debe implementarse de forma parametrizada, de forma que se pueda usar para puzles de cualquier tamaño (tanto en número de filas como de columnas, no necesariamente iguales) con números en sus casillas que varien entre 1 y el máximo entre el número de filas y el número de columnas del puzle.

2.2. Búsqueda de la solución

Parte del trabajo consiste en investigar la mejor manera de resolver el puzle mediante algoritmos de búsqueda. Es por esto que podría ser necesario evaluar distintas formas de representación, y funciones de coste y/o heurística, que proporcionen los mejores resultados en la búsqueda de la solución. En este proceso es admisible hacer modificaciones sobre los algoritmos de búsqueda implementados en la librería `búsqueda_espacio_estados.py`, o incluso el desarrollo de nuevos algoritmos de búsqueda.

2.3. Experimentación

La experimentación con puzles de tamaño pequeño puede proporcionar ideas sobre como mejorar el proceso de búsqueda. Es importante obtener buenos resultados con los ejemplos del fichero `ejemplos_prueba.txt` antes de probar con los del fichero `ejemplos_reto.txt`. Se deben registrar los resultados obtenidos con los puzles del fichero `ejemplos_prueba.txt`, con los distintos algoritmos de búsqueda evaluados, indicando el tiempo de respuesta (no las soluciones). Para los ejemplos del fichero `ejemplos_reto.txt`, se deben incluir los casos de éxito (no las soluciones) y el tiempo de respuesta para el algoritmo de búsqueda que haya sido escogido como el mejor en la fase anterior.

2.4. Documentación

En el fichero `plantilla-trabajo.doc` se muestra una sugerencia de estructura y formato de estilo artículo científico correspondiente a la documentación del trabajo. Este formato es el del *IEEE conference proceedings*, cuyo sitio web *guía para autores* [1] ofrece información más detallada y plantillas para Word y Latex.

El artículo deberá tener una extensión **entre 8 y 10 páginas**, y la estructura general del documento debe ser como sigue: en primer lugar realizar una **introducción** al trabajo explicando el objetivo fundamental, incluyendo un breve repaso de antecedentes en relación con la temática del trabajo y con los métodos empleados (mencionar referencias bibliográficas), a continuación describir la **estructura** del trabajo, las **decisiones de diseño** que se hayan tomado a lo largo de la elaboración del mismo, y la **metodología** seguida al implementarlo (nunca poner código, pero sí pseudocódigo), y seguidamente detallar los **experimentos** llevados a cabo, **analizando los resultados** obtenidos. Por último, el documento debe incluir una sección de **conclusiones**, y una **bibliografía** donde aparezcan no sólo las referencias citadas en la sección de introducción, sino cualquier documento consultado durante la realización del trabajo (incluidas las referencias web a páginas o repositorios).

2.5. Mejoras

Aunque no sea obligatorio, sí se tendrá en cuenta en la calificación la incorporación de un interfaz o menú amigable que facilite la experimentación o la presentación de resultados. También podrán recibir puntuación adicional otras mejoras o añadidos que se incorporen al trabajo más allá de los requisitos mínimos que se mencionan en la lista de objetivos específicos.

2.6. Presentación y defensa

El día de la defensa se deberá realizar una pequeña presentación (PDF, PowerPoint o similar) de 10 minutos en la que participarán activamente todos los miembros del grupo que ha desarrollado el trabajo. Esta presentación seguirá a grandes rasgos la misma estructura que el documento, pero se deberá hacer especial mención a los resultados obtenidos y al análisis crítico de los mismos. Se podrá usar un portátil (personal del alumno), diapositivas y/o pizarra. En los siguientes 10 minutos de la defensa, el profesor procederá a realizar preguntas sobre el trabajo, que podrán ser tanto del documento como del código fuente.

3. Criterios de evaluación

Para que el trabajo pueda ser evaluado, se deberá satisfacer los objetivos concretos descritos en el apartado 1 (todos y cada uno de ellos, si no, el trabajo obtendrá una nota de suspenso). Uno de los alumnos del equipo deberá subir a través del formulario disponible en la página de la asignatura un fichero comprimido .zip, que contenga:

- **Una carpeta con el código fuente**. Dentro de dicha carpeta tiene que haber un fichero README.txt, que resuma la estructura del código fuente, e indique cómo usar la interfaz (si se ha implementado), o al menos cómo hacer pruebas con las funciones implementados, incluyendo ejemplos de uso. Asimismo se deberá indicar cómo reproducir los experimentos realizados. Es importante la coherencia de este fichero con la defensa.
- **El documento – artículo en formato PDF**. Deberá tener una extensión mínima de 8 páginas, y máxima de 10. Deberá incluir toda la bibliografía consultada (libros, artículos, technical reports, páginas web, códigos fuente, diapositivas, etc.) en el apartado de referencias, y mencionarlas a lo largo del documento.

Para la evaluación se tendrá en cuenta el siguiente criterio de valoración, considerando una nota máxima de 3 en total para el trabajo:

- **El código fuente (1 punto)**: se valorará la claridad y buen estilo de programación, corrección y eficiencia de la implementación, y calidad de los comentarios. La claridad del fichero README.txt también se valorará. En ningún caso se evaluará un trabajo con código copiado directamente de internet o de otros compañeros.
- **Retos resueltos (0.5 puntos)**: se valorará la cantidad y calidad (en tiempo de respuesta) de puzles resueltos del fichero `ejemplos_reto.txt`.
- **El documento – artículo científico (1 punto)**:
  - Se valorará el estilo general del documento (por ejemplo, el uso de la plantilla sugerida).
  - Se valorará la cantidad y calidad de los experimentos, los resultados alcanzados y el análisis crítico que se haga de los mismos.
  - Se valorará la claridad de las explicaciones, el razonamiento de las decisiones, el análisis y presentación de resultados, y el uso del lenguaje. Igualmente, no se evaluará el trabajo si se detecta cualquier copia del contenido.
- **La presentación y defensa (0,5 puntos)**: se valorará la claridad de la presentación y la buena explicación de los contenidos del trabajo, así como, especialmente, las respuestas a las preguntas realizadas por el profesor.
- **Mejoras**: Se valorarán hasta con 0.5 puntos extra sin superar el máximo de 3 puntos totales del trabajo.

IMPORTANTE: Cualquier **plagio**, **compartición de código** o uso de material que no sea original y del que no se cite convenientemente la fuente, significará automáticamente la **calificación de cero** en la asignatura para **todos los alumnos involucrados**. Por tanto, a estos alumnos **no se les conserva**, ni para la actual ni para futuras convocatorias, **ninguna nota** que hubiesen obtenido hasta el momento. Todo ello sin perjuicio de las correspondientes **medidas disciplinarias** que se pudieran tomar.

4. Referencias

[1] Plantilla IEEE. https://www.ieee.org/conferences_events/conferences/publishing/templates.html

[2] Revista japonesa Nikoli. https://www.nikoli.co.jp/en/index.html