



DEPARTAMENTO  
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

# Trabajo Práctico

## Sudoku

14 de diciembre de 2015

Metaheurísticas  
2do Cuatrimestre de 2015

Integrante	LU	Correo electrónico
Kujawski, Kevin	459/10	kevinkuja@gmail.com
Ortiz de Zarate, Juan Manuel	45/10	jmanuoz@gmail.com

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		



**Facultad de Ciencias Exactas y Naturales**  
Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2160 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (54 11) 4576-3359

<http://www.fcen.uba.ar>

## Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Descripción y formulación matemática</b>	<b>2</b>
<b>3. Descripción de la Solución</b>	<b>3</b>
3.1. Simulated annealing . . . . .	3
3.1.1. Definición . . . . .	3
3.1.2. Implementación para resolver Sudoku . . . . .	3
<b>4. Decisiones tomadas y su justificación</b>	<b>5</b>
<b>5. Experimentos</b>	<b>6</b>
<b>6. Modo de uso y Requerimientos</b>	<b>7</b>
<b>7. Código fuente del programa</b>	<b>8</b>
<b>8. Conclusiones</b>	<b>9</b>
<b>9. Referencias</b>	<b>9</b>

## 1. Introducción

Sudoku es un juego de lógica cuyo objetivo es rellenar una cuadrícula de tamaño 9x9, divididas a su vez en 9 cajas de 3x3, estas últimas llamadas cajas. Inicialmente contiene cierta cantidad de celdas con valores fijos pre-definidos y válidos, es decir, sus valores están entre [1,9], no hay repetidos por filas, columnas o cajas.

Las reglas del juego son:

1. Cada fila debe contener todos los números en el intervalo [1,9]
2. Cada columna debe contener todos los números en el intervalo [1,9]
3. Cada caja debe contener todos los números en el intervalo [1,9]

El objetivo del juego es rellenar las celdas en blanco de la cuadrícula inicial respetando las reglas del juego. A pesar de la simpleza del objetivo y las reglas del juego, hay 6670903752021072936960 formas posibles de ser rellenada una cuadrícula. Buscar una solución intentando todas las formas posibles es claramente imposible, y eso incentiva la utilización de una metaheurística para solucionar el juego.

En el presente trabajo, presentaremos una posible formulación matemática del Sudoku como un problema de optimización e dos propuestas de implementación utilizando dos metaheurísticas: Simulated annealing y Colonia de hormigas

## 2. Descripción y formulación matemática

Nuestra formulación matemática es una adaptación de (2). En esa formulación, las variables  $X_{ijk}$  son variables de decisión, definidas de la siguiente manera:

$$X_{ijk} = \begin{cases} 1 & \text{si el elemento (i,j) de la cuadrícula contiene el valor k} \\ 0 & \text{en caso contrario} \end{cases}$$

La formulación como programación lineal entera es la siguiente:

$$\begin{aligned} \min \quad & 0 \\ \text{sujeto a} \quad & \sum_{i=1}^9 X_{ijk} = 1, j = 1 : 9, k = 1 : 9 \end{aligned} \quad (1)$$

$$\sum_{j=1}^9 X_{ijk} = 1, i = 1 : 9, k = 1 : 9 \quad (2)$$

$$\sum_{j=3q-2}^{3q} \sum_{i=3p-2}^{3p} X_{ijk} = 1, k = 1 : 9, p = 1 : 3, q = 1 : 3 \quad (3)$$

$$\sum_{k=1}^9 X_{ijk} = 1, i = 1 : 9, j = 1 : 9 \quad (4)$$

$$X_{ijk} = 1 \forall (i, j, k) \in \text{INICIALES} \quad (5)$$

$$X_{ijk} \in \{0, 1\}$$

Hay que notar que como se trata de un problema de satisfacibilidad, la formulación no necesita de una función objetivo, por eso la definimos como 0. Las restricciones (1),(2) y (3) garantizan que cada número en el intervalo posible de la instancia solo aparezca una vez en cada columna, fila, y caja, respectivamente. La restricción (4) garantiza que todas las posiciones de la matriz estén rellenas. La restricción (5) fuerza que las variables fijas de la instancia permanezcan sin alterar.

### 3. Descripción de la Solución

#### 3.1. Simulated annealing

##### 3.1.1. Definición

Simulated annealing (Recocido Simulado en español), respecto a las ciencias de la computación, es una metaheurística para problemas de optimización global, aplicándose a la heurística de Búsqueda Local Aleatoria. La técnica fue presentada por Kirkpatrick, Gelatt y Vecchi (1982) y ha sido muy empleado para resolver problemas combinatorios. La idea surge del proceso físico conocido como recocido en el cual, se eleva la temperatura de un sólido hasta el punto que se vuelve líquido, a continuación la temperatura se disminuye de forma paulatina para obtener una estructura cristalina sin defectos y que puede considerarse como un estado de mínima energía. Cada descenso de temperatura debe ser lo suficientemente pequeño para que el sistema no adquiera una estructura cristalina con defectos, además el sistema debe permanecer un tiempo suficiente a una misma temperatura para permitir alcanzar un estado estacionario, en otras palabras, que las partículas vuelvan a acomodarse.

Se parte de una solución inicial posible  $S_i$ , luego se selecciona una solución posible  $S_j$  dentro de una vecindad, enseguida se evalúa la calidad de la solución posible empleando una función de costo  $f(x)$  asociada a cada posible solución  $S$  del problema.

Si la nueva solución posible  $S_j$  es mejor que la actual (de acuerdo al costo), se acepta, de lo contrario se selecciona de acuerdo a una probabilidad, para Simulated annealing dicha probabilidad de selección está dada por la ecuación (1) que se conoce como el criterio de Metropolis; a  $temp$  se le conoce como el parámetro de control; este valor se inicia con un valor suficientemente grande para que cualquier solución tenga una probabilidad alta de ser seleccionada, a medida que  $temp$  disminuye, la probabilidad de aceptar soluciones factibles de mala calidad disminuye. El enfriamiento se realiza empleando el sistema geométrico dado por la ecuación (2), donde  $\alpha$  es un parámetro fijo.

$$criterio(S_i, S_j) = \exp((costo(S_i) - costo(S_j))/temp) \quad (1)$$

$$temp_{k+1} = temp_k * \alpha \quad (2)$$

##### 3.1.2. Implementación para resolver Sudoku

Antes de ir directo al algoritmo hay que tener en cuenta las diferentes elecciones que hicimos para encontrar una solución inicial, mejorarla y decidir cuando una solución es mejor que otra.

**Solución inicial** Como el algoritmo así lo requiere, debemos partir de una solución inicial para intentar mejorarla y en el mejor de los casos poder resolverla. Una primera opción que se puede pensar es, a partir de las celdas iniciales, rellenar todas las celdas vacías con valores aleatorios directamente. Sin descartar totalmente esta idea fuimos un poco más allá. Básicamente intentamos resolver el Sudoku lo más que podamos, estando siempre seguro de que los valores que pongamos en las celdas son los correctos. Para esto analizamos los posibles valores que pueden ir en cada celda, eliminando de esas posibilidades valores que estén en la misma fila, columna o caja. Luego de esto, en caso de detectar que una celda tiene un solo valor posible, se rellenará la celda con ese valor y se lo considerará correcto ya que fue deducido de la configuración inicial. En caso de que no haya valores posibles para esa celda, indicará que esa configuración inicial no tiene solución posible. Por cada celda que se rellene se deberá recalcular todo el proceso de nuevo, hasta que en algún ciclo no haya modificaciones.

Una vez que se terminó de deducir lo máximo posible será el momento de rellenar las celdas que quedaron vacías. Para este punto intentaremos ser lo menos aleatorio posible, y para tal caso seleccionaremos entre los posibles valores que pueden tener la celda, que ya habíamos calculado

en el punto anterior. Y en caso de que para una celda ya no haya posibles valores, es el momento en que se le asignará uno aleatorio, siempre respetando la formulación (3) que se menciona en la sección anterior, es decir, se respetará en todo momento que no haya celdas inválidas en las cajas.

**Función de costo** El costo (o penalización) va a depender de la cantidad de celdas ilegales” que tenga la cuadrícula. Es decir, cuando haya una celda que se repita en la fila, columna o caja se sumará una penalidad. El caso ideal, que indica que el tablero está solucionado, es cuando el costo devuelve 0.

**Soluciones vecinas**

**Algoritmo**

#### 4. Decisiones tomadas y su justificación

## 5. Experimentos

## 6. Modo de uso y Requerimientos



## **7. Código fuente del programa**

A continuación incluimos el código fuente del programa desarrollado.

## 8. Conclusiones

## 9. Referencias

1. B. Felgenhauer, e F. Jarvis. (2006, January). Mathematics of Sudoku I.
2. A. Bartlett, T. Chartier, A. N. Langville, e T. Rankin. (2008). An Integer Programming Model for the Sudoku Problem. *Journal of Online Mathematics and its Applications MAA*, (8):1-14.