DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico

Sudoku

28 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

1. Introducción

Sudoku es un juego de lógica cuyo objetivo es rellenar una cuadricula 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 cuadricula inicial respetando las reglas del juego. A pesar de las simpleza del objetivo y las reglas del juego, hay 6670903752021072936960 formas posibles de ser rellenada una cuadricula. 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:

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

La formulación como programación lineal entera es la siguiente:

$$\begin{array}{lll} \min & \text{FuncionCosto(X)} & \\ \text{sujeto a} & \sum_{i=1}^{9} X_{ijk} = 1, j = 1:9, k = 1:9 & (1) \\ & \sum_{j=1}^{9} X_{ijk} = 1, i = 1:9, k = 1:9 & (2) \\ & \sum_{j=3q-2}^{3q} \sum_{i=3p-2}^{3p} X_{ijk} = 1, k = 1:9, p = 1:3, q = 1:3 & (3) \\ & \sum_{k=1}^{9} X_{ijk} = 1, i = 1:9, j = 1:9 & (4) \\ & X_{ijk} = 1 \forall (i,j,k) \in INICIALES & (5) \\ & X_{ijk} \in 0, 1 & \end{array}$$

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 numero 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 martiz esten rellenas. La restricción (5) fuerza que las variables fijas de la instancia permanezcas sin alterar.

2.1. Hormigas

Nuestra segunda metaheuristica no es exactamente una colonia de hormigas ya que esta solución cuando la aplicamos tal cual no nos daba buenos resultados. Por eso probamos readaptarla de forma tal que mejoren lo mas posibles las soluciones. A continuación explicaremos el pseudo código de nuestra adaptación de colonia de hormigas.

function HORMIGAS(Inicio)
    $feromonas \leftarrow$ new Fermonas()
    For 1 to 400
        $sudoku \leftarrow$ new Sudoku(inicio)
        $hormiga \leftarrow$ new Hormiga(sudoku,feromonas,probaSeguirFeromona)
        Si iteración es multiplo de 6
            $feromonas$-¿evaporar()
        Fin Si
        Si *hormiga*-¿resolver()
            Devolver *hormiga*-¿solucion
        Fin Si
end function

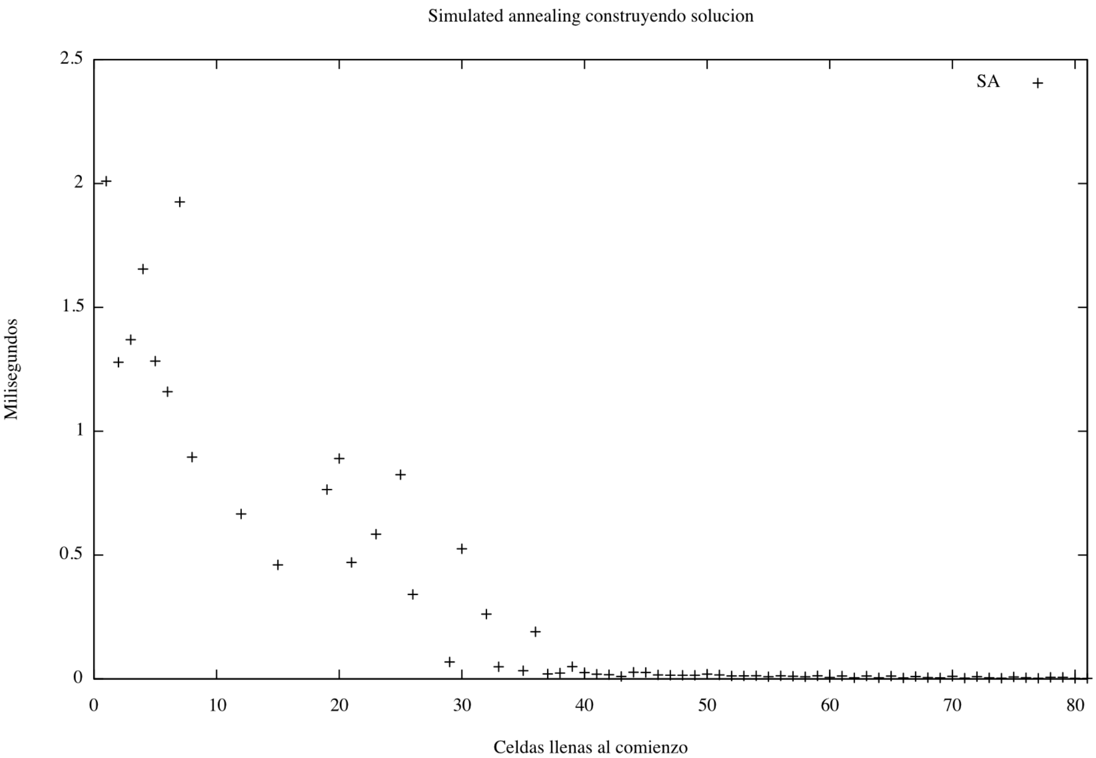
	Simulated Annealing	Colonia de Hormigas
Escenarios probados	115	
Solucionados	70 (60 %)	
Dificultad baja	71 %	
Dificultad moderada	40 %	
Dificultad alta	16 %	

3. Experimentos

3.1. Simulate annealing

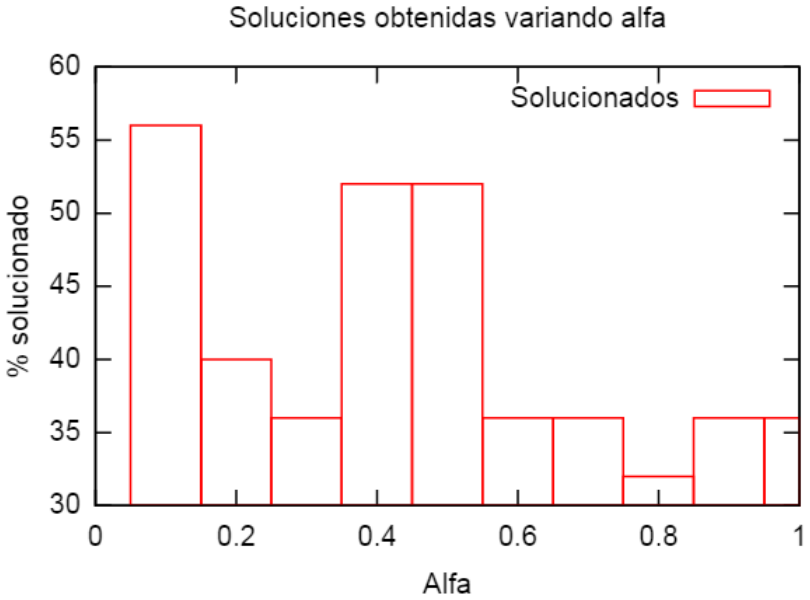
Para los experimentos elegimos un factor de enfriamiento en 0.5 y una temperatura inicial de 1. La justificación es en base a los experimientos realizados y que mostraremos en esta sección.

El siguiente grafico es el resultado del tiempo que toma solucionar un Sudoku en donde partiendo de una solución existente blanqueamos celdas random y observamos como lo soluciona el algoritmo:



Clarmente se puede observar que a medida que el tablero esta mas lleno el tiempo es menor, eso es debido a la suma de varios factores, como que inicialmente infiere más celdas, son menos las vacias (que se convierte en menos iteraciones) y menos búsqueda de soluciones vecinas.

A continuación corrimos el algoritmo con 25 soluciones de diferentes niveles de dificultad, variando el factor de enfriamiento (o alfa):



Si bien con valores intermedios (aprox 0.5) es donde se concentran las mejores soluciones, y que la mayor cantidad se logra con un 0.1, creemos en base a anteriores pruebas que el el factor de enfriamiento debería ser siempre mayor de 0.5 para que sea algo ”lentoz no siempre elija ir por soluciones vecinas pero que tampoco las restrinja totalmente.

4. Conclusiones

5. 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 Pro- gramming Model for the Sudoku Problem. Journal of Online Mathematics and its Applications MAA, (8):1-14.

3. Algunas soluciones sacadas de http://es.websudoku.com