# APLICACIÓN DE TÉCNICAS DE OPTIMIZACIÓN COMBINATORIAL A LA SOLUCIÓN DEL SUDOKU

## Combinatorial optimization techniques applied to Sudoku's solution

**RESUMEN**
Se presenta el Sudoku como un problema de optimización combinatorial de única solución y se propone un modelo matemático que lo representa. Se aplican las técnicas de optimización combinatorial Búsqueda Tabú y Algoritmos Genéticos en varios casos de prueba.

**PALABRAS CLAVES:** Sudoku, problema de optimización, optimización combinatorial, técnicas de optimización, Búsqueda Tabú, Algoritmos Genéticos.

**ABSTRACT**
*The Sudoku is presented as a unique solution NP-hard optimization problem and its mathematical model is proposed. Optimization Techniques Tabu Search and Genetic Algorithms are used to solve Sudoku's problem, showing results on several test cases*.

**KEYWORDS**: Sudoku, optimization problem, combinatorial optimization, optimization techniques, Tabu search, genetic algorithms.

**JOHN FREDY FRANCO B.**
Ingeniero Electricista, M. Sc.
Profesor Catedrático
Universidad Tecnológica de Pereira
jffb@utp.edu.co

**OSCAR GÓMEZ CARMONA**
Ingeniero Electricista, M. Sc.
Profesor Auxiliar
Universidad Tecnológica de Pereira
jr@utp.edu.co

**RAMON A. GALLEGO R.**
Ingeniero Electricista, Ph. D.
Profesor Titular
Universidad Tecnológica de Pereira
ragr@utp.edu.co

Grupo de Investigación en Planeamiento de Sistemas Eléctricos de Potencia

## 1. INTRODUCCIÓN

En las ciencias aplicadas al igual que en el conocimiento teórico, existen problemas de optimización lineal o no lineal con variables continuas o enteras. Cuando las dimensiones y la complejidad matemática del problema son grandes, aparece un fenómeno denominado explosión combinatorial, debido a la gran cantidad de soluciones factibles e infactibles que aparecen [1]. En la práctica, las técnicas exactas no puedan ser aplicables para su solución debido al gran tiempo de cómputo necesario. Estos problemas se conocen como problemas tipo NP (non-deterministic polynomial time) completo y no se puede encontrar un algoritmo que los resuelva en un tiempo de cómputo polinomial.

El Sudoku, es un pasatiempo de ubicación de números en una cuadricula dividida en filas, columnas y subcuadros que visto como un problema de optimización presenta el fenómeno de explosión combinatorial, sus variables no son continuas y en la mayoría de los casos tiene solución única. Estas características lo ubican en la categoría de los problemas NP completos.

Aunque este juego carece de utilidad práctica, hace parte de una clase de problemas de gran interés práctico conocidos como problemas de satisfacción de restricciones (*constraint satisfaction problems*) con explosión combinatorial y es un caso de prueba que permite verificar la eficiencia de los algoritmos exactos de optimización y métodos de computación blanda

En este artículo se resolverá el problema del Sudoku mediante técnicas de optimización combinatorial. Inicialmente, se presentará el Sudoku como problema de optimización y se planteará un modelo matemático, posteriormente, se utilizarán dos técnicas de optimización combinatorial, una técnica evolutiva (*algoritmos genéticos*) y una técnica basada en memoria adaptativa (*búsqueda tabú*) para encontrar su solución.

## 2. SUDOKU

Sudoku es un pasatiempo originario de Estados Unidos que se popularizó en Japón en 1986 y se dio a conocer en el ámbito internacional en el 2005. El objetivo es llenar una cuadrícula de 9 × 9 celdas (81 celdas), dividida en subcuadros de 3 × 3, con las cifras del 1 al 9, partiendo de algunos números ya dispuestos en algunas de las celdas. Además, cada número de la solución aparece sólo una vez en cada fila, columna y subcuadro, de ahí el "los números deben estar solos" que evoca el nombre del juego [2].

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

Figura 1. Sudoku

Fecha de Recepción: 07 Septiembre de 2007
Fecha de Aceptación: 16 Noviembre de 2007

En el Sudoku, la cuadrícula más común es de 9 x 9 con regiones de 3 × 3, aunque también se utilizan otros tamaños ($n$ x $n$ con regiones de $\sqrt{n}$ x $\sqrt{n}$,). Las regiones no tienen qué ser cuadradas, aunque generalmente lo son.

Normalmente el Sudoku se clasifica en niveles de dificultad, según la relevancia y posición de los números iniciales y no de la cantidad de estos. Efectivamente, la cantidad de números dados apenas afecta la dificultad del Sudoku e incluso puede no afectar en absoluto ya que un Sudoku con un mínimo de números iniciales puede ser muy fácil de resolver y uno con más números de la media puede ser extremadamente complicado de resolver. Además, se dice que un Sudoku está bien planteado si la solución es única.

Para los programadores es relativamente sencillo construir algoritmos de búsqueda por los métodos de backtracking [3], divide y vencerás, fuerza bruta, programación dinámica o algoritmo voraz. Aunque lejos de la eficiencia computacional, estos métodos encontrarían la solución si se permite infinito tiempo computacional, especialmente en juegos de alto orden ($n$=16, $n$=25). Algunos algoritmos usan los números dispuestos para implementar métodos de eliminación, llenando las celdas vacías con todos los posibles números que podrían ir [4]. Estos algoritmos, como cita el autor, sirven para la solución de Sudokus de nivel de dificultad sencillo y moderado.

Para la solución del problema han sido utilizados métodos más eficientes, como, "*Forward Checking (FC)*", "*Limited Discrepancy Search (LDS)*" [5], "*message-passing*" [6], "*SAT-based solver"* [7], *"ASP-Answer Set Programming"* [8], entre otros.

Las técnicas de optimización combinatorial han sido poco usadas para la solución del sudoku [9], aunque podrían ser aplicables que ya son técnicas que encuentran soluciones de buena calidad en tiempos de cómputo razonables. La dificultad en el uso de estas técnicas es que no aseguran el óptimo global en un problema que tiene solución única (caso del Sudoku), razón por la cual este problema se convierte en una forma de validar la eficiencia de las técnicas de optimización combinatorial.

### 2.1 Complejidad Matemática del Sudoku.

Aunque no se sabe a ciencia cierta los orígenes del Sudoku, se presume que su antecesor es el denominado "*Latin square*". El "*Latin square*", en un cuadrado de $n$ x $n$ que se llena con $n$ símbolos tal que cada símbolo aparece solo una vez en cada fila y en cada columna. El cuadro apareció en el siglo XVIII y es atribuido al matemático Leonhard Euler (1707 - 1783). *Latin square* de orden 1, existen 1, de orden 4, 576 pero de orden 9 existen 5.524.751.496.156.892.842.531.2254.600. [10]

La diferencia entre el Sudoku y el *Latin square* es la división en subcuadros de $\sqrt{n}$ x $\sqrt{n}$ del Sudoku y la respectiva restricción de presencia de símbolos en cada subcuadro. Esta restricción, hace que la cantidad de posibles Sudokus, sea lógicamente menor. Se ha estimado que la cantidad de Sudokus para el orden 1, es 1, para el orden 4, es 288 y para el orden 9, 6.670.903.752.021.072.936.960 [10,11].

Sin importar la cantidad de Sudokus completos (que cumplen las restricciones de filas, columnas y sub-cuadros) que existen, la dificultad matemática radica en la explosión combinatorial que presenta un Sudoku con determinada cantidad de números iniciales (pistas), es decir, entre menos pistas tenga el Sudoku, mayor es el espacio de soluciones (factibles e infactibles) que existen (tabla 1), aunque como se expresó anteriormente, un Sudoku bien planteado debe tener solución única.

| Pistas | Cuadros en blanco | Espacio de soluciones |
|---|---|---|
| 75 | 6 | 5,3144E+05 |
| 70 | 11 | 3,1300E+10 |
| 65 | 16 | 1,8530E+15 |
| 60 | 21 | 1,0942E+20 |
| 55 | 26 | 6,4611E+39 |
| 50 | 31 | 3,8152E+44 |
| 45 | 36 | 2,2528E+49 |
| 40 | 41 | 1,3303E+54 |
| 35 | 46 | 7,8552E+58 |
| 30 | 51 | 4,6384E+63 |
| 25 | 56 | 2,7389E+68 |
| 20 | 61 | 1,6173E+73 |
| 15 | 66 | 9,5500E+77 |
| 10 | 71 | 5,6392E+82 |
| 5 | 76 | 3,3299E+87 |
| 0 | 81 | 1,9663E+92 |

Tabla 1. Espacio de soluciones según la cantidad de pistas para el Sudoku de orden 9.

Por lo tanto, dadas una cantidad de pistas ($Xp$), el espacio de soluciones ($Es$) es igual a:

$$Es = 9^{(81-Xp)} \qquad (1)$$

Como se observa en la ecuación, el problema del Sudoku presenta el fenómeno de explosión combinatorial, lo cual significa que un incremento en las variables de decisión, aumenta el espacio de soluciones y el esfuerzo computacional.

Se ha demostrado [12] que el problema de solucionar un Sudoku pertenece a la categoría NP completo, problemas cuya complejidad no es polinómica, por lo tanto (en teoría) no se puede construir un algoritmo que los resuelva en un tiempo polinómico.

En cuanto a la existencia de solución única, no ha sido posible demostrar cuál es la cantidad mínima de pistas que debe tener un Sudoku para garantizar solución única, aunque es fácil determinar que con 78, 79 y 80 pistas se

garantiza solución única. Respecto a 77, en [10] se muestra dos posibles soluciones para un Sudoku.

## 3. El SUDOKU COMO PROBLEMA DE OPTIMIZACIÓN

El Sudoku como problema de optimización, consiste en la minimización de la cantidad de números repetidos entre 1 y *n,* en filas, columnas y subcuadros de una matriz (*M*) representadas, dichas repeticiones, como los elementos del conjunto *A*. Por lo tanto, el objetivo es minimizar la cantidad de elementos presentes en ese conjunto (cardinalidad de A).

*minimizar Cardinalidad (A)*

$$A=\left\{\begin{array}{l}(i,j)/\quad M(f,i)=M(f,j)\wedge i\neq j\\ \vee\quad M(i,c)=M(j,c)\wedge i\neq j\\ \vee\left\{\begin{array}{l}M(i,j)=M(h,k)\wedge(i\neq h\vee j\neq k)\\ \wedge\ \left\lfloor (i-1)/\sqrt{n}\right\rfloor\cdot\sqrt{n}+\left\lfloor (j-1)/\sqrt{n}\right\rfloor=\left\lfloor (h-1)/\sqrt{n}\right\rfloor\cdot\sqrt{n}+\left\lfloor (k-1)/\sqrt{n}\right\rfloor\end{array}\right\}\\ i,j,f,c,h,k\in[1,\ldots,n]\quad M: \text{matriz de juego}\end{array}\right\} \quad (2)$$

## 4. SOLUCIÓN AL SUDOKU MEDIANTE BÚSQUEDA TABÚ

Búsqueda Tabú [13], es una técnica de optimización combinatorial que proviene de la inteligencia artificial y usa conceptos de memoria adaptativa y exploración sensible. Esta técnica resuelve problemas del tipo:

$$\begin{array}{l}min\quad f(x)\\ s.a.\quad x\in X\end{array} \quad (3)$$

donde *f* es una función general, lineal o no lineal, y *X* es un conjunto de restricciones lineales o no lineales. Las variables *x* pueden ser de naturaleza continua, entera o mixta. La exploración sensible de Búsqueda tabú se basa en la idea de que una mala decisión hecha por una estrategia produce más información que una buena selección hecha de forma aleatoria. Búsqueda Tabú implementa una estrategia de búsqueda local para explorar de forma eficiente el espacio alrededor de una configuración. Cada solución tiene asociado un conjunto de soluciones vecinas llamada vecindad. Con la estructura de vecindad dada, la búsqueda local permite pasar a la mejor configuración vecina, o en el caso de que la configuración actual sea la mejor, se pasa a la menos peor, donde se aplica nuevamente búsqueda local sobre la nueva vecindad en busca del óptimo. Debido a que en cada paso se deben analizar todos los vecinos (o un conjunto reducido que sigue siendo grande), Búsqueda Tabú es un método de alto costo computacional.

Un algoritmo de Búsqueda Tabú completo utiliza técnicas de exploración y de memoria avanzadas, como: memoria de corto y largo plazo, estrategias de intensificación, diversificación, oscilación estratégica, path relinking y lista de configuraciones de élite, entre otras [13].

### 4.1 Algoritmo de Búsqueda Tabú Aplicado a la Solución del Sudoku

La configuración inicial se genera llenando cada columna con los números faltantes, de tal manera que por columna no se repitan números. Esta configuración inicial normalmente no cumple con las condiciones por filas y subcuadros.

El vecindario para una configuración se define según las siguientes estrategias:

- Cambiar un número repetido en una columna por un número que falte en esa columna.
- Cambiar un número repetido en una fila por un número que falte en esa fila.
- Cambiar un número repetido en un subcuadro por un número que falte en ese subcuadro.
- Intercambiar los números de dos celdas en una misma columna.
- Intercambiar los números de dos celdas en una misma fila.

En lugar de evaluar la función objetivo recorriendo el cuadro completamente para analizar el número de repeticiones por filas, columnas o subcuadros, se realiza una estimación de la variación de la función objetivo para un vecino partiendo de la función objetivo de la configuración actual, con lo que se consigue un gran ahorro en tiempo computacional. La estimación de la función objetivo se efectúa de acuerdo al tipo de variación hecha para llegar a la configuración vecina.

## 5. SOLUCIÓN AL SUDOKU MEDIANTE ALGORITMOS GENÉTICOS

Los algoritmos genéticos son una técnica de búsqueda a través de configuraciones (espacio de soluciones del problema), que inicialmente fue idealizado usando los mecanismos de la evolución y de la genética natural [1].

Este algoritmo, en su forma básica, inicia con la selección de una población inicial (conjunto de configuraciones iniciales). Luego se realiza un proceso de selección entre las configuraciones para darles el derecho de participar en la generación de nuevos descendientes, posteriormente, se forman parejas para ser sometidas a un proceso de recombinación (crossover) y finalmente se realiza un proceso de mutación que altera los elementos de algunas configuraciones. Este proceso tiene como finalidad generar diversidad en la población y está controlado por un parámetro denominado tasa de mutación el cual determina cuantos cambios se realizarán sobre la población. Después de este proceso, se tienen los elementos de la nueva generación. El buen desempeño del algoritmo genético depende en gran medida, de la selección adecuada de los parámetros de control (tamaño de la población inicial, tasa de recombinación y tasa de mutación) y la forma en que se realizará la selección, recombinación y mutación.

### 5.1 Algoritmo Genético Aplicado a la Solución del Sudoku

El algoritmo genético en su forma básica aplicado a la solución del Sudoku considera las siguientes etapas:

#### *A. Codificación del problema*

Cada configuración del problema es una matriz con números entre 1 y *n*, como se muestra en la figura 2.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 7 | **2** | **4** | 8 | 3 | **7** | 2 | 6 | 5 |
| **6** | 8 | 8 | 9 | 1 | 3 | 5 | 2 | 4 |
| 4 | 2 | **3** | **6** | **8** | 1 | **4** | **1** | **5** |
| **4** | **3** | **1** | 5 | 1 | **5** | 7 | 4 | 3 |
| **5** | 1 | 2 | 9 | 9 | 1 | 4 | **3** | **2** |
| **7** | **9** | 5 | 3 | 9 | 8 | 1 | **6** | 3 |
| **2** | 2 | **9** | **7** | **1** | 6 | **8** | 9 | 2 |
| 2 | **4** | 2 | 6 | **9** | **3** | 2 | 9 | 3 |
| **3** | **1** | 2 | 4 | 7 | **4** | **7** | **5** | 2 |

Figura 2. Codificación del problema

#### *B. Población inicial*

La población inicial es generada de forma aleatoria. Para cada celda en blanco del problema, se genera un número aleatorio entre 1 y *n*. El proceso se repite para cada configuración de la población.

#### *C. Calculo de la función objetivo*

La función objetivo de cada configuración, debe reflejar la cantidad de números repetidos en cada fila, columna y subcuadro, dado que computacionalmente es ineficiente contar repeticiones, la función objetivo se calculó determinando los números que faltan en filas, columnas o subcuadros (figura 3). La función objetivo de cada configuración es la suma de los faltantes en filas columnas y subcuadros (*Fo* = 79). De esta forma se determina la mejor configuración encontrada y se almacena como incumbente.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 7 | **2** | **4** | 8 | 3 | **7** | 2 | 6 | 5 | 2 |
| | **6** | 8 | 8 | 9 | 1 | 3 | 5 | 2 | 4 | 1 |
| | 4 | 2 | **3** | **6** | **8** | 1 | **4** | **1** | **5** | 2 |
| | **4** | **3** | **1** | 5 | 1 | **5** | 7 | 4 | 3 | 4 |
| | **5** | 1 | 2 | 9 | 9 | 1 | 4 | **3** | **2** | 3 |
| | **7** | **9** | 5 | 3 | 9 | 8 | 1 | **6** | 3 | 2 |
| | **2** | 2 | **9** | **7** | **1** | 6 | **8** | 9 | 2 | 3 |
| | 2 | **4** | 2 | 6 | **9** | **3** | 2 | 9 | 3 | 4 |
| | **3** | **1** | 2 | 4 | 7 | **4** | **7** | **5** | 2 | 3 |
| | | | | | | | | | | |
| Faltan | 3 | 3 | 2 | 2 | 4 | 2 | 3 | 2 | 5 | |
| Faltan SubC | 3 | 3 | 4 | 2 | 4 | 3 | 4 | 3 | 3 | |

Figura 3. Cálculo de la función objetivo.

#### *D. Proceso de selección*

El proceso de selección es proporcional y realizado mediante el método de la ruleta, para esto se utiliza una función de adaptación que garantiza selectividad y transforma el valor de función objetivo original de minimización a maximización.

$$Fa_i = max(\overline{Fo}) \cdot k - Fo_i \qquad (4)$$

donde:

$Fa_i$: Función de adaptación del individuo i-ésimo.

$\overline{Fo}$: Vector función objetivo de los individuos.

$Fa_i$: Función objetivo del individuo i-ésimo.

$k$: Tasa de adaptación ($k$ >1).

#### *E. Proceso de recombinación*

El proceso de recombinación se realizó de tal manera que se conservaran las mejores características de los padres. Debido que en el proceso la población se mantiene constante, cada pareja de padres tendrá dos hijos: el primero, tendrá los mejores subcuadros de los padres, es decir, aquellos subcuadros que presentan menos repeticiones; el segundo tendrá las mejores filas o columnas de los dos padres, seleccionado de forma aleatoria con igual probabilidad de ocurrencia.

#### *F. Proceso de mutación*

El proceso de mutación se realizó para generar diversidad ya que la solución normalmente es única y se debe realizar una buena exploración del espacio de soluciones. Cuatro tipos de mutación fueron realizadas, controladas por probabilidades, según el problema: intercambio de dos elementos en una fila, intercambio de dos elementos en una columna, intercambio de dos elementos en un subcuadro y reemplazo de un número del individuo por otro número generado de forma aleatoria entre 1 y *n* (figura 4).

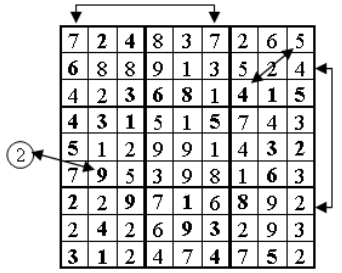


Figura 4. Tipos de mutación

Adicionalmente, para no perder el rastro de búsqueda, si en la nueva población no hay un individuo mejor o igual a la incumbente, se reemplaza el peor individuo de la población por la mejor solución encontrada. En caso de que el algoritmo genético no encuentre la solución, el proceso para cuando un número predefinido de iteraciones (generaciones) es alcanzado.

## 6. CASOS DE PRUEBA Y RESULTADOS

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

Caso 1

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

Caso 2

Figura 5. Casos de prueba

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

Caso 3

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

Caso 4

Figura 6. Casos de prueba

Las metodologías propuestas se probaron en cuatro sistemas de prueba con "diferente nivel de dificultad".

### 6.1 Solución mediante la técnica Búsqueda Tabú

Se implementó una estructura de memoria de corto plazo para prohibir movimientos que involucraran celdas cambiadas recientemente. Para cada configuración se creaba la lista de mejores movimientos a configuraciones vecinas, escogiéndose la mejor que no estuviera prohibida (estado tabú).

El mejor resultado se obtuvo cuando se definía el vecindario usando un número de iteraciones determinado, primero para cambios de números repetidos en filas, columnas y subcuadros, luego un número de iteraciones efectuando intercambios por columnas y finalmente haciendo intercambios por filas.

| **Parámetros Búsqueda Tabú** | |
|---|---|
| Tamaño lista tabú | 30 |
| Valor tabú | 20 |
| Iter. # repetidos | 1 |
| Iter. intercambios columnas | 10 |
| Iter. intercambios filas | 10 |

Tabla 2. Parámetros para el algoritmo Búsqueda Tabú.

El número de iteraciones necesarias para encontrar la solución fue: caso 1, 637; caso 2, 782; caso 3, 1001; caso 4, 424.

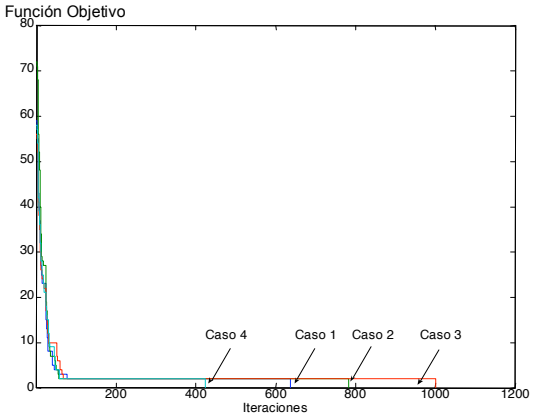


Figura 7. Convergencia del algoritmo Búsqueda Tabú.

### 6.2 Solución mediante el Algoritmos Genéticos

Se presenta la solución a los casos de prueba de dos formas:

***A. Parámetros iguales para los cuatro casos***

Se ejecutó el algoritmo descrito con los siguientes parámetros:

| | |
|---|---|
| Semilla | 13 |
| Población Inicial | 16 |
| Taza de cruzamiento | 90% |
| Cantidad de mutaciones | 15 |
| Mutación en filas | 0% |
| Mutación en columnas | 10% |
| Mutación en cuadros | 50% |
| Mutación de elementos | 40% |

Tabla 3. Parámetros para el algoritmo genético.

Para cada caso se encontró la solución, con la siguiente cantidad de iteraciones: caso 1, 4457; caso 2, 6979; caso 3, 29659; caso 4, 9268. En la figura 8 se observa el proceso de convergencia para cada caso.

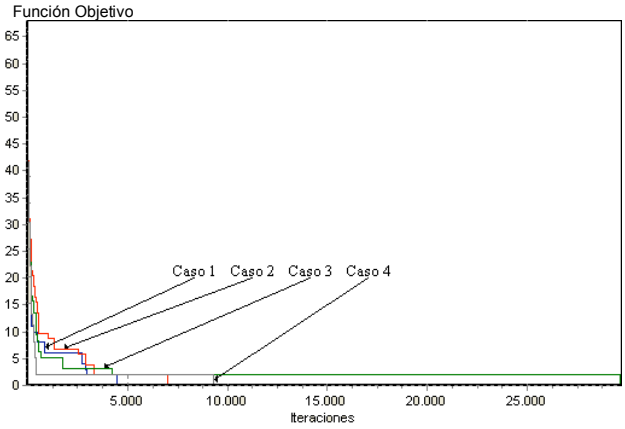


Figura 8. Convergencia del algoritmo genético

***B. Parámetros especiales para cada caso***

Una de las principales características de los algoritmos genéticos es que para cada problema, sus parámetros deben ser ajustados, de tal manera que sea eficiente en la solución.

La solución del caso 1 se obtiene de forma más eficiente (1792 iteraciones) con los siguientes parámetros:

| | |
|---|---|
| Semilla | 1 |
| Población Inicial | 10 |
| Taza de cruzamiento | 90% |
| Cantidad de mutaciones | 15 |
| Mutación en filas | 0% |
| Mutación en columnas | 33% |
| Mutación en cuadros | 33% |
| Mutación de elementos | 34% |

Tabla 4. Parámetros para el algoritmo genético

Con los anteriores parámetros, modificando solo la semilla (Semilla = 8), la solución del caso 3 se obtiene en 3490 iteraciones:

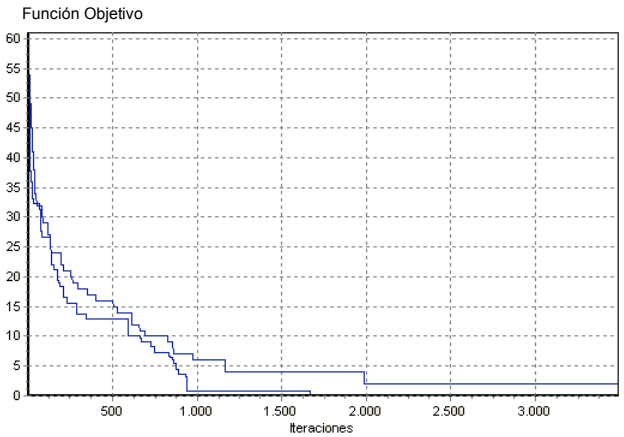

Figura 9. Solución a los casos 1 y 3 con parámetros particulares

## 7. CONCLUSIONES

Se plantea el sudoku, como un problema de optimización que presenta características de alta complejidad matemática como la explosión combinatorial, la presencia de variables enteras y la existencia de solución única y se propone un modelo matemático.

La estrategia de búsqueda local que implementa Búsqueda Tabú permite salir de configuraciones que tienen pocas repeticiones pero en las que pequeños cambios muestras un empeoramiento en la solución (óptimos locales). Al prohibir movimientos en celdas modificadas recientemente se consigue escapar de esos óptimos locales.

El mejor desempeño de la solución con Búsqueda Tabú se obtuvo al implementar varias definiciones para el vecindario, usando cada una para generar nuevas configuraciones durante un número de iteraciones y luego cambiándola por otra; con esto se consiguió resolver problemas por columnas, filas o subcuadros concentrándose en uno a la vez.

Se consiguió ajustar los parámetros de Búsqueda Tabú para resolver los distintos casos presentados, de tal forma que fueron resueltos en un número relativamente bajo de iteraciones.

El proceso de cruzamiento empleado por el algoritmo genético, da prioridad a las mejores configuraciones y extrae sus mejores atributos para generar la nueva población. Este comportamiento hace que el método no se desvíe y siempre trate de mejorar la solución durante el proceso de búsqueda

Dado que el problema presenta un gran espacio de soluciones, los diferentes procesos de mutación realizados permiten una amplia exploración sin perder el rastro del óptimo global.

Los procesos de recombinación y mutación empleados en el algoritmo genético presentan una relación apropiada entre diversidad e intensificación en la búsqueda de la solución.
Debido a la naturaleza aleatoria del algoritmo genético y la inicialización realizada para este problema (iniciación aleatoria), la semilla que genera la secuencia de números aleatorios se convierte en un parámetro más del algoritmo ya que en el proceso de inicialización es la que determina el punto de partida para el proceso de optimización.

Aunque es posible determinar parámetros para que el algoritmo genético soluciones casos diferentes, el mejor desempeño se obtiene cuando el algoritmo es parametrizado para cada caso.

Se comprobó que las técnicas combinatoriales son herramientas que mediante diferentes estrategias o procesos de exploración, selección e intensificación tienen la capacidad de encontrar óptimos globales.

## 8. BIBLIOGRAFÍA

[1] Gallego R. Ramón A., Escobar Z. Antonio, Romero L. Rubén A. "Técnicas de Optimización Combinatorial", Primera edición. grupo de Investigación en Planeamiento de Sistemas Eléctricos. Universidad Tecnológica de Pereira. Pereira abril de 2006.
[2] Wikipedia. Sudoku. Wikipedia, the free encyclopedia. Available. online at http://en.wikipedia.org/wiki/Sudoku.
[3] Ruiz Jose María, Orandes Jose Pedro. “Hal 9000 contra los Sudokus mutantes”. Desarrollo Python. http://www.linux-magazine.es.
[4] Luis R. Morera. “Solución de un Sudoku”. www.articuloweb.com/articles.php?art_id=331&start=1. August 23, 2007, 5:57 pm
[5] Tristan Cazenave. “A search based Sudoku solver”. Labo IA Dept. Informatique Universit´e Paris 8, 93526, Saint-Denis, France.
[6] Jacob Goldberger. “Solving Sudoku Using Combined Message Passing Algorithms”. School of Engineering, Bar-Ilan University.
[7] Tjark Weber. “A SAT-based Sudoku Solver”. Institut für Informatik, Technische Universität München Boltzmannstr. 3, D-85748 Garching b. München, Germany.
[8] Jiménez, José A. “Solución declarativa del Sudoku mediante ASP”. Grupo de Lógica Computacional. Dpto. de Ciencias de la Computación e Inteligencia Artificial, Universidad de Sevilla Sevilla, 17 de Junio de 2006.
[9] Rhyd Lewis. “Metaheuristics can solve Sudoku puzzles”, SpringerLink, Springer Science+Business Media, LLC 2007. Published online: 1 May 2007
[10] Delahaye Jean-Paul. “The science behind Sudoku”. Scientific American. 2006. Pg. 80-87.
[11] Bertram Felgenhauer, Frazer Jarvis. “Enumerating possible Sudoku grids”. www.afjarvis.staff.shef.ac.uk/sudoku/sudoku.pdf June 20, 2005
[12] Takayuki YATO, Takahiro SETA, “Complexity and Completeness of Finding Another Solution and Its Application to Puzzles”, IEICE Trans. Fundam. E 86-A(5), 1052-1060. 2003
[13] GLOVER, F., “Tabu Search Fundamentals and Uses”, University of Colorado, Boulder, Colorado, April 1995.