Aplicación de la metaheurística “Búsqueda tabú” al problema de las N-reinas

Mg. Alicia Riojas Cañari[1], Dra. María Álvarez Rivas[1]

[1]Facultad de Ciencias Matemáticas
Universidad Nacional Mayor de San Marcos

aliriojas@hotmail.com, malvarezr1@unmsm.edu.pe

RESUMEN

Se presenta los conceptos de la Búsqueda Tabú, su algoritmo base y su aplicación al problema combinatorio de las N- reinas para lo cual se desarrolló un programa en lenguaje c++.

Se presenta el problema de las -reinas con fines de explicar el método de búsqueda tabú: sus componentes y su algoritmo.

Se realizó 9 corridas del algoritmo con diferentes parámetros y se encontró 12 soluciones diferentes que proporcionan el óptimo (cero colisiones de las reinas).

Palabras clave: Metaheurísticas, Búsqueda Tabú, problema de -reinas

ABSTRACT

It presents the Tabu Search concepts, its algorithm, and a application to the N- Queen problem developing a program in c++ language

It presents the N- Queen problem for explaining the tabu search method

It presents the result of nine replications with different parameters, it was found 12 different solutions with zero collisions.

Keywords: Metaheuristics, Tabu search, N- Queen problem

1. INTRODUCCIÓN

Un área importante de la investigación operativa es la de la programación matemática, la cual puede estar definida en un dominio continuo o discreto. En el primero, el desarrollo de calculo diferencial y de los métodos exactos, como el simplex, proporcionan buenas herramientas para resolver los problemas de optimización, pero cuando las variables de decisión son discretas, especialmente si involucran una gran cantidad de variables, la búsqueda de soluciones exactas puede no ser posible porque, o no hay soluciones analíticas o no es viable su implementación en un medio computacional convencional.

En el seno de la investigación operativa ha surgido una serie de procedimientos heurísticos para resolver estos problemas, los cuales son flexibles al involucrar características específicas y permiten la interacción con el decisor para buscar aproximaciones a la solución ideal, sobre la base del desarrollo tecnológico en el área de la computación, pues generalmente los procedimientos heurísticos son iterativos y requieren de una gran cantidad de cálculos.

El propósito del presente artículo es presentar las características principales de la metaheurística Búsqueda Tabú *(TABU SEARCH),* sus conceptos, su metodología e implementar su algoritmo base en un programa computacional en c++ para la solución del problema combinatorio de las N-reinas *(N-queen problem)*. [5]

2. FUNDAMENTACIÓN TEÓRICA

El problema de las -reinas consiste en colocar n reinas en un tablero de ajedrez de n x n de tal manera que no sea posible que dos reinas se capturen entre sí, es decir, que no estén en la misma fila, ni en la misma columna, ni en la misma diagonal. Se dice que hay una colisión si hay dos reinas que se pueden capturar entre sí.

Se trata de elegir las n celdas donde colocar a las reinas, minimizando el número total de colisiones.

Una solución tiene la forma de un arreglo n-dimensional: $(r_1,r_2,r_3,.....r_n)$

Reina	R1	R2	R3	....	Rn
Ubicación en la columna	1	2	3	…	n

Por ejemplo una solución para n = 4 es (3,4,1,2)

Matricialmente se representa:

	Columna1	Columna2	Columna3	Columna4
Reina1			Reina1	
Reina2				Reina2
Reina3	Reina3			
Reina4		Reina4		

Observar que esta configuración presenta 4 colisiones: {(1,2) (3,4) (1,3) (2,4,)}

Aplicaciones del problema de las N-reinas

Se puede encontrar una aplicación del problema de las n-reinas al *"... problema de diseño de un material formado por un número de capas aislantes. El orden según el cual se planifican estas capas determina el valor de aislamiento total del material resultante"* [4] Es decir, las restricciones típicas del problema de las -reinas pueden ser modificadas de acuerdo a las características específicas de los materiales de las capas aislantes y estas variaciones se pueden introducir al algoritmo para adaptarlo a condiciones muy específicas.

Formas de solución

Este problema puede resolverse de varias formas:

1. Enumerando todas las posibles alternativas y evaluando si se producen colisiones, en cuyo caso se tendría que evaluar factorial de n posibles soluciones.
2. Usando un método de búsqueda local, como el *greedy*. Por ejemplo, si se asigna la reina 1 a la columna 1 aunque se permute exhaustivamente las otras 3 solo se consigue un óptimo local, es decir, el mínimo de colisiones posibles es una colisión y ya no se podría mejorar, (el algoritmo queda atrapado en un óptimo local), es decir, si se fija la reina 1 en la columna 1 nunca se encontrará una configuración con cero colisiones.
3. Modelándolo como un problema lineal de maximizar el número de reinas en un tablero de ajedrez sujeta a las restricciones de que en una fila solo haya una reina, al igual que en cada columna y, además que en cada diagonal haya una y solo una reina.

   Cuando n=4 el problema tiene 16 variables y 19 restricciones, pero si n= 20 el problema tiene 400 variables y 115 restricciones.

4. Usando una metaheurística, como en este caso la búsqueda tabú.

El problema de las N-reinas es un problema **P**, su complejidad es polinómica determinista, no se puede decir que este problema es intratable, pero se presenta por ser didáctico para la comprensión del método de búsqueda tabú y en cada caso es posible tener una solución analítica para contrastar la solución hallada heurísticamente.

3. MÉTODO

- Primero se hace una breve presentación de conceptos teóricos para contextualizar la aplicación de la búsqueda tabú.
- Luego se describe el algoritmo de la búsqueda Tabú y su implementación computacional
- A continuación se resuelve el problema de las N-reinas usando la metaheurística de búsqueda tabú para N = 7.

Metaheuristica Búsqueda Tabu

La búsqueda Tabú es un método de búsqueda "inteligente", la cual se caracteriza por utilizar una estrategia basada en el uso de estructuras de memoria para escapar de los óptimos locales en los que se puede caer al moverse de una solución a otra en el espacio de soluciones.[1]

El término tabú (*taboo*) procede de la Polinesia[3], donde es usado por los aborígenes de la isla Tonga para referirse a cosas que no pueden ser tocadas porque son sagradas, una acepción más moderna la define como *"Una prohibición impuesta por costumbres sociales como una medida de protección"*, también como "*marcada como que constituye un riesgo*", esta acepción es la que está más cerca de la esencia del método donde el riesgo a ser evitado es el de seguir un camino no productivo, incluyendo el de ser conducido a una trampa de la que no se puede salir (óptimo local).

La Búsqueda Tabú se caracteriza por:

- El uso de estructuras de memoria la cual puede ser de corto plazo (memoria reciente) y de largo plazo (memoria de frecuencias).
- La lista tabú y los mecanismos de selección del siguiente movimiento.
- Las estrategias de búsqueda: intensificación y diversificación.

La **lista tabú** es una lista donde se registran aquellas soluciones o atributos de soluciones que no deben volver a ser elegidas, por un tiempo.

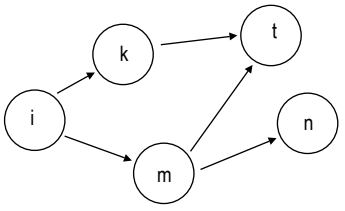
Una forma sencilla de construir una lista tabú consiste en que cada vez que se realiza un movimiento, se introduce el mismo en una lista circular, se considera que tras un cierto número de iteraciones la búsqueda está en una región distinta y puede liberarse del status tabú. Su objetivo es penalizar la búsqueda para evitar el ciclado.

El tamaño de la lista tabú (***tabu tenure***) es el tiempo o número de iteraciones que un elemento (movimiento o atributo) permanece en la lista tabú.

Las soluciones tabú pueden dejar de ser tabú, es decir, una solución o atributo puede salir de la lista tabú antes de que se cumpla su plazo. Esto se implementa a través del **criterio de aspiración,** que permite que un movimiento sea admisible aunque esté clasificado como tabú.

Se define un **movimiento s** como el mapeo definido en un subconjunto de la región factible.

Por ejemplo en el siguiente grafo, si el algoritmo está en el nodo i, (la solución factible i) los movimientos posibles son aquellos arcos que unen el nodo i con alguno de sus nodos adyacentes. (Otras posibles soluciones que se derivan de la solución i)



El vecindario o entorno de $x \in X$ (conjunto de soluciones) es el conjunto N(x), el cual contiene los movimientos $s \in S$ (conjunto de movimientos posibles) que pueden ser aplicados a la solución x.

En la i-ésima iteración, para evolucionar hacia otras soluciones, se selecciona éstas en un **vecindario reducido**: $(N(X_i) - \{\text{Lista Tabú}\})$, evaluando cada una de las soluciones y eligiendo la mejor.

Para realizar una búsqueda completa, es deseable que el tamaño del entorno no sea grande, en cuyo caso, con objeto de reducir el tiempo de computación, se puede realizar la búsqueda en un subconjunto tomado aleatoriamente.

La memoria basada en lo reciente es donde se almacenan los últimos movimientos realizados, y que puede ser utilizada para "recordar" aquellos movimientos que hacen caer de nuevo en soluciones ya exploradas [4].

La memoria basada en frecuencias proporciona un tipo de información que complementa la información proporcionada por la memoria basada en lo reciente, ampliando la base para seleccionar movimientos preferidos. En esta estructura de memoria se registra la frecuencia de ocurrencias de los movimientos, las soluciones o sus atributos.

Una alta frecuencia de transiciones de una solución puede indicar un ciclado y debe ser restringido para permitir diversidad.

Implementación computacional para el problema de las -reinas

Para resolver el problema se le adapta a la estructura del algoritmo, es decir, se identifica la función objetivo, el espacio de soluciones, el vecindario, los movimientos permitidos y el tamaño de la lista tabú.

Componentes del algoritmo

La *función objetivo* es minimizar la cantidad de colisiones.

El ***espacio de soluciones*** está dado por todos aquellos arreglos de 7 posiciones de los números {1,2,...7}

La estructura de ***la solución*** es la ubicación de las reinas

x =	R1	R2	R3	R4	R5	R6	R7

por ejemplo si:

x =	6	1	2	7	5	3	4

significa que la reina 1 está ubicada en la columna 6, la reina 2 en la columna 1, ..., la reina 7 en la columna 4.

Un ***movimiento*** es un intercambio de dos reinas.

El ***vecindario*** de x está formado por todas aquellas soluciones a las que se llega desde x al hacer un movimiento, es decir, en las cuales se ha realizado uno y solo un intercambio de reinas. Por ejemplo sea:

z =	4	1	2	7	5	3	6

z pertenece al vecindario de x, pues se ha permutado la reina 1 con la 7.

La condición de parada: el proceso se detendrá si se alcanza el máximo número de iteraciones permitida , que en este caso se asigna arbitrariamente como **MAXITER**=100, independientemente de la soluciones encontradas.

No se detendrá cuando se encuentre una solución que produce cero colisiones, pues se tratará de encontrar más de una solución con cero colisiones.

El tamaño de la lista tabú *(tabu tenure*), se define como 3, es decir, se penalizará hasta las tres últimas soluciones.

Selección de la solución inicial:

Puede ser el resultado de una heurística, de una selección aleatoria o de una asignación arbitraria realizada por el experto.

Sea la solución inicial:

x_0 =	4	5	3	6	7	1	2

La función objetivo vale 4. Colisionan las reinas: {(2,6) (6,7) (4,5) (1,2)}

Iteraciones para seleccionar la siguiente solución:

Se realizan los intercambios posibles (movimientos) mientras no se cumpla la condición de parada.

Iteración 1 El vecindario está conformado por las combinaciones de 7 elementos tomados de 2 en 2, es decir, 7!/(2!*5!) = 21. Se evalúan las 21 posibles soluciones:

# alternativa	intercambio		Colisiones
1	1	2	7
2	1	3	7
....			
21	6	7	3

Seleccionándose las 5 mejores

Lista de candidatos		
Intercambio		Colisiones
1	7	2
2	4	2
2	6	2
5	6	2
1	5	3

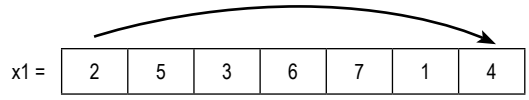
Hay 4 movimientos que producen 2 colisiones.

Cuando hay empates se puede utilizar un mecanismo aleatorio para seleccionar el mejor movimiento.

1	5	3

Se escoge la permutación (1,7)

La solución siguiente resulta de intercambiar las reinas 1 y 7 en x_0.



La función objetivo vale 2, colisionan las reinas: {(2,6) (4,5)}

*La **lista tabú*** contiene los movimientos considerados prohibidos, en este caso se registran los atributos de las permutaciones (el intercambio de reinas).

Para prevenir que las reinas vuelvan a su lugar anterior, se registrará en la lista tabú los 3 últimos movimientos.

Solución actual x1

2	5	3	6	7	1	4

El movimiento (1,7) está penalizado durante 3 iteraciones

Estructura de la lista tabú

	2	3	4	5	6	7
1						3
2						
3						
4						
5						
6						

Iteración 2

Mejores 5 candidatos		
cambio		colisiones
2	4	1
1	6	2
2	5	2
3	5	2
3	6	2

Iteración 2: se intercambian las reinas 2 y 4 en x_1 la función objetivo valdrá 1, solo hay una colisión {(1,4)}. Se continúa iterando hasta llegar a la condición de parada.

Si la mejor permutación es una solución que está en la lista tabú, se desestima y se toma la siguiente "mejor", sin embargo, si el objetivo es encontrar una solución con cero colisiones, se puede utilizar el **criterio de aspiración** a un movimiento que produce cero colisiones, pero como se verá cuando se procese en un programa computacional, esto ocasiona que no se diversifique la búsqueda y no se encuentren más soluciones.

> *"tradicionalmente en la literatura la noción de mejor movimiento corresponde a aquel que lleva a un mejor cambio en la función objetivo y frecuentemente se asume por convención. Sin embargo, la filosofía de la búsqueda tabú ve "el mejor" en el contexto, teniendo en cuenta una variedad de dimensiones además del cambio en la función objetivo [2].*

Dado que el *tabu tenure* es un número finito, en algún momento un movimiento saldrá de la lista tabú y podrá ser elegido nuevamente, por lo que siempre se corre el riesgo de ciclado.

Para diversificar la búsqueda se usó **la memoria de largo plazo**, en este caso, la frecuencia de ocurrencia de los movimientos. Supongamos que no se está considerando ningún criterio de aspiración y se han realizado ya 75 iteraciones.

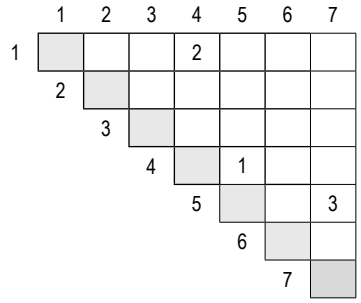
Suponer que la solución actual x75

5	3	1	6	4	2	7

Memoria de frecuencias

	1	2	3	4	5	6	7
1							
2	7						
3	8	7					
4	4	5	0				
5	4	0	0	5			
6	3	4	0	4	4		
7	4	0	0	7	5	4	

La lista tabú en la iteración 75 es:



Mejores 5 candidatos para la iteración 76			
Intercambio		FO	frecuencia
4	7	1	7
5	7	1T	5
1	2	2	7
1	4	2T	4
1	6	2	3

T significa que la permutación está penalizada en la lista tabú.

Al evaluar los 5 mejores candidatos para pasar a x_{76} se observa que el elegido debería ser el movimiento 4,7 pues da un valor para la función objetivo igual a una colisión, pero, al revisar la tabla de frecuencias (**memoria a largo plazo**) se observa que dicho movimiento ha ocurrido 7 veces en el pasado, hay otros candidatos no tabú con menor frecuencia y además hay otros movimientos que tienen frecuencia cero, lo que implica que se puede explorar otras regiones que podrían contener otras soluciones con menos colisiones.

- Por lo tanto el movimiento 4,7 a pesar de no ser tabú debe ser penalizado para que no vuelva a ocurrir hasta que la frecuencia de los otros lo supere.
- El candidato (5, 7) no puede ser elegido por ser tabú.
- El candidato (1, 2) es no tabú, pero entre los otros candidatos no tabú hay uno que tiene menor frecuencia de ocurrencias en el pasado.
- Por lo tanto el elegido para conformar la siguiente solución debería ser el movimiento (1, 6) para diversificar la búsqueda.

El procedimiento se repite hasta superar el máximo número de iteraciones permitidas.

Implementación computacional en c++

Se realizó un programa en lenguaje c++ versión 3.0 Borland International Inc. 1990 1992. Este programa es específico para el problema de las N reinas, en este caso se han contado las colisiones en las diagonales.

El programa cuenta con 2 rutinas, inicio y el proceso hasta alcanzar la condición de parada. Los pasos más importantes de las cuales se describen brevemente a continuación:

Rutina de inicio

- Se asignan los parámetros de la corrida: Iteración a partir de la cual se considera el criterio de aspiración. Iteración a partir de la cual se considera la memoria de largo plazo. Número máximo de iteraciones permitidas. Cantidad de reinas ($n \leq 10$). La solución inicial.
- Se calcula la cantidad de vecinos: n tomados de 2 en 2, (para $n \leq 10$ el máximo es de 45 vecinos).
- Se construye el vecindario: una matriz de 4 columnas y la cantidad de filas depende de la cantidad de vecinos donde (#, i, j, k) significa intercambiar a la reina i con la reina j , k es el valor de la FO luego del intercambio y # es el número de orden del vecino (esta variable se registró solo para verificar los resultados contra los elaborados manualmente).
- Se inicializan con ceros la lista tabú y la tabla de frecuencias.
- Se calcula la función objetivo de la solución inicial.

Proceso hasta alcanzar la condición de parada

Repetir mientras el número de iteraciones sea menor que MAXITER

- Copiar la solución actual en una transitoria (para preservar la solución actual).

Para cada vecino:

- Hacer el intercambio correspondiente en la solución transitoria. Evaluar la solución, es decir, calcular las diagonales positivas superiores e inferiores y las diagonales negativas superiores e inferiores y calcular la cantidad de colisiones (la función objetivo).
- Seleccionar los “c” mejores candidatos (c<10) la estructura de los candidatos tiene 6 columnas:

(1)	(2)	(3)	(4)	(5)	(6)
# de orden en el vecindario	Reina que intercambia	Reina que intercambia	Valor de la función objetivo	Condición de tabú o no tabú	frecuencia de ocurrencias

Se considera como mejores a los que tienen menor valor en la función objetivo.

- Seleccionar al mejor no tabú
- Dependiendo de los parámetros se usa el criterio de aspiración y/o la memoria de largo plazo para modificar la selección dada en el paso anterior
- Se actualizan las estructuras tabú, las de frecuencias y la nueva solución actual

4. RESULTADOS

Se realizaron 9 corridas con diferentes parámetros para hacer un análisis de la influencia de la memoria de corto y largo plazo en el proceso, así como también de la conveniencia de usar como criterio de aspiración el encontrar una solución con cero colisiones.

Los parámetros comunes para todas las corridas son:

Cantidad de reinas en el tablero	n = 7
Solución inicial	x_0 = 4,5,3,6,7,1,2
Máximo número de iteraciones	MAXITER = 100
Tamaño de lista tabú	tabu_tenure = 2
Cantidad de candidatos	c = 5

Los resultados se resumen en la siguiente tabla

Corrida	Parámetros		Resultados	
	Nivel de aspiración	Memoria de largo plazo	Cantidad de soluciones sin colisiones	Soluciones diferentes
Sólo lista tabú	100	100	30	2
Lista tabú + criterio de aspiración desde el comienzo	1	100	46	1
Lista tabú + criterio de aspiración desde iteración 20	20	100	43	2
Lista tabú + criterio de aspiración desde iteración 40	40	100	40	2
Lista tabú + criterio de aspiración desde iteración 40 +Memoria largo plazo desde iteración 40	40	40	21	6
Lista tabú + criterio de aspiración desde iteración 40 + Memoria largo plazo desde iteración 70	40	70	31	3
Lista tabú + criterio de aspiración desde iteración 70 +Memoria largo plazo desde iteración 40	70	40	21	5
Lista tabú + Memoria largo plazo desde iteración 20, sin criterio de aspiración	100	20	17	6
Lista tabú + Memoria largo plazo desde iteración 20 + criterio de aspiración desde iteración 70	70	20	19	5

Se considera que dos soluciones son iguales cuando:

- La solución i es la solución j desplazada, por ejemplo:

Solución i	5	1	4	7	3	6	2
Solución j	2	5	1	4	7	3	6

- Es la misma solución, pero en orden inverso, por ejemplo:

Solución i	6	1	3	5	7	2	4
Solución j	4	2	7	5	3	1	6

- Una combinación de las dos anteriores, por ejemplo:

Solución i	2	5	1	4	7	3	6
Solución j	2	6	3	7	4	1	5

5. DISCUSIÓN

Se produce una mayor cantidad de soluciones diferentes cuando se utiliza el criterio de aspiración más tarde, por ejemplo en la iteración 70.

Las corridas que produjeron más soluciones diferentes fueron la 4 y la 7, en las cuales se implementa la memoria de largo plazo "más temprano", (las iteraciones 40 y 20 respectivamente) mientras que cuando se utilizó la memoria de largo plazo a partir de la iteración 70, sólo se obtuvo 3 soluciones diferentes, es decir, si se diversifica más temprano se visita regiones diferentes y en consecuencia se obtiene más soluciones diferentes, pero que logran cero colisiones.

En la siguiente tabla se muestran las soluciones diferentes que se encontraron en las 9 corridas de prueba.

Soluciones diferentes:

	R1	R2	R3	R4	R5	R6	R7
Sol 1	6	1	3	5	7	2	4
Sol 2	7	4	1	5	2	6	3
Sol 3	3	1	6	2	5	7	4
Sol 4	6	2	5	1	4	7	3
Sol 5	6	4	7	1	3	5	2
Sol 6	6	3	1	4	7	5	2
Sol 7	2	4	1	7	5	3	6
Sol 8	3	7	2	4	6	1	5
Sol 9	6	3	1	4	7	5	2
Sol 10	4	1	3	6	2	7	5
Sol 11	2	5	1	4	7	3	6
Sol 12	2	5	7	4	1	3	6

Apareció en la corrida:

0	1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1	1
1		1	1	1	1	1	1	1
				1				
				1				
				1		1		
				1	1			
						1		
						1	1	1
							1	
							1	1
							1	
								1
2	1	2	2	6	3	5	6	5

Total de soluciones diferentes en cada corrida :

6. CONCLUSIONES

- Para el problema de las N-reinas con valores pequeños de n (como en este caso n=7), se puede formular como un programa lineal y encontrar una solución usando algún software, sin embargo haciendo un programa computacional para el algoritmo de búsqueda tabú se pueden encontrar varias soluciones a un costo menor, pues siempre hay lenguajes de programación de propósito general libres de costo en el mercado. Además, se puede introducir modificaciones al algoritmo básico de tal modo que el usuario pueda aportar su conocimiento de experto.
- El criterio de aspiración utilizado desde las primeras iteraciones produce que el algoritmo se concentre en óptimos locales y se genera una cantidad de soluciones óptimas, pero iguales, mientras que cuando se utiliza dicho criterio luego de realizar una gran cantidad de iteraciones o cuando no se utiliza, se genera más soluciones diferentes
- El uso de la memoria de largo plazo permite la diversificación, es decir se visita regiones diferentes y en consecuencia se obtiene más soluciones diferentes que logran cero colisiones.

7. BIBLIOGRAFÍA

[1] [DIAZ 1996] Díaz A, Glover F, Ghaziri HM et al. 1996. *Optimización Heurística y Redes Neuronales.* Madrid, Paraninfo.

[2] [LAGUNA1994] Laguna 1994. "A Guide to implementing Tabu search", *Investigación Operativa* v. 4, n. 1, pp. 17.

[3] [GLOVER 2005] Glover Fred & Laguna Manuel "Tabu search". http://leeds-faculty.colorado.edu/laguna/articles/ts2.pdf (consultado en noviembre 2005).

[4] [GLOVER 2003] lover Fred y Melián Belén. "Búsqueda tabú" *Revista Iberoamericana de Inteligencia Artificial*. N.19 pp. 29-48. ISSN: 1137-3601. © AEPIA (2003). http://www.aepia.org/revista (consultado en noviembre 2005).

[5] [RIOJAS 2005] RiojasA. (2005) "BÚSQUEDA TABÚ: Conceptos, algoritmo y aplicación al problema de las N-reinas http://sisbib.unmsm.edu.pe/bibvirtualdata/monografias/basic/riojas_ca/