UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

FACULTAD CIENCIAS MATEMATICAS

E.A.P. DE..INVESTIGACIÓN OPERATIVA

Conceptos, algoritmo y aplicación al problema de las N – reinas

Capítulo4. Un ejemplo y su implementación computacional

MONOGRAFÍA

Para optar el Título de Licenciada de Investigación operativa

AUTOR

Alicia Cirila Riojas Cañari

LIMA – PERÚ
2005

CAPÍTULO 4 UN EJEMPLO Y SU IMPLEMENTACIÓN COMPUTACIONAL

Se presenta el problema de las n-reinas con fines de explicar el método de búsqueda tabú: sus componentes y su algoritmo.[4]

Se decidió usar este problema para fines didácticos. 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..."*[10]

Es decir, las restricciones típicas del problema de las n-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.

4.1. El Problema

El problema de las n-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 si, 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 si.

Se trata pues de encontrar una configuración – elegir las n celdas donde colocar a las reinas- que minimice 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

[4] Laguna, Manuel "A Guide to implementing Tabu search", *Investigación Operativa* v. 4, n. 1, pp. 5-25 Abril 1994

[10] Glover, 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

Por ejemplo una solución para n = 4 es (3,4,1,2)
Matricialmente se representa:

	Columna1	Columna2	Columna3	Columna4
Reina1			**ℜ1**	
Reina2				**ℜ2**
Reina3	**ℜ3**			
Reina4		**ℜ4**		

Hay 4 colisiones {(1,2) (3,4) (1,3) (2,4,)}

4.2. Formas de solución

Este problema puede resolverse de varias formas, por ejemplo enumerando todas las posibles alternativas y evaluando si se producen colisiones, en cuyo caso se tendría que evaluar factorial de n posibles soluciones. Cuando n es igual a 4 la cantidad de soluciones alternativas es de 24 y como se puede ver en el Anexo 4.1 hay 2 configuraciones que producen cero colisiones:

Configuración # 11

Reina	R1	R2	R3	R4
Ubicación en la columna	2	4	1	3

Configuración # 14

Reina	R1	R2	R3	R4
Ubicación en la columna	3	1	4	2

Observar que si se utiliza un método de búsqueda local, como el *greedy*, y suponiendo que como punto de partida se asigna la reina 1 a la columna 1 (ver las seis primeras alternativas en el anexo 4.1) aunque se permute exhaustivamente las otras 3 sólo se consigue un óptimo local, es decir el mínimo de colisiones posibles es una colisión y ya no se podría mejorar. (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.

Se tendría que volver a utilizar el método *greedy* tomando como punto de partida a la reina 2 en la columna 1 para encontrar cero colisiones en la solución (2,4,3,1).

Cabe anotar que en los métodos de búsqueda heurísticos no se dispone de teoremas como el de Khun Tucker para la programación matemática, con el que es posible determinar si una solución dada es óptima global.

En el siguiente cuadro se muestra la cantidad de soluciones alternativas que se tendrían que evaluar para diferentes tamaños de n:

Cantidad de reinas	**Evaluación exhaustiva de n! alternativas**
4	24
7	5,040
10	3,628,800
20	2,432,902,008,176,640,000

Cuando n es un número grande la evaluación exhaustiva de todas las alternativas factibles es intratable.

El problema de n reinas puede ser visto como un problema lineal de maximizar el número de reinas en un tablero de ajedrez sujeta a las restricciones de que en una fila sólo haya una reina, al igual que en cada columna y además que en cada diagonal haya una y sólo una reina.

Cuando n=4 el problema tiene 4 restricciones correspondiente a las filas, 4 restricciones para las columnas, (2*4-3) diagonales negativas y (2*4-3) diagonales positivas.
Las variables son: x_{ij} = la reina i ocupa la fila i y la columna j; i = 1...4; j=1....4.

Quedando el problema matemático lineal con 16 variables y 19 restricciones:

max x11+ x12+ x13+ x14+ x21+ x22+ x23+ x24 +x31+ x32+ x33+ x34+ x41+ x42+ x43+ x44
Sujeto a
diagonales negativas

d-1) x12 + x21 <= 1
d-2) x13 + x22 + x31 <= 1
d-3) x14 + x23 + x32 + x41 <= 1
d-4) x24 + x33 + x42 <= 1
d-5) x34 + x43 <= 1

diagonales positivas

d+1) x13 + x24 <= 1
d+2) x12 + x23 + x34 <= 1
d+3) x11 + x22 + x33 + X44 <= 1
d+4) x21 + x32 + x43 <= 1
d+5) x31 + x42 <= 1

Una reina por fila

f1) x11 + x12 + x13 + x14 <= 1
f2) x21 + x22 + x23 + x24 <= 1
f3) x31 + x32 + x33 + x34 <= 1
f4) x41 + x42 + x43 + x44 <= 1

Una reina por columna

c1) x11 + x21 + x31 + x41 <= 1
c2) X12 + x22 + x32 + x42 <= 1
c3) x13 + x23 + x33 + x43 <= 1
c4) x14 + x24 + x34 + X44 <= 1

Sólo 4 reinas

solo4) x11+ x12+ x13+ x14+ x21+ x22+ x23+ x24 +x31+ x32+ x33+ x34+ x41+ x42+ x43+ x44 <= 4
Las variables son binarias.

Al procesar este modelo en el software LINDO se obtuvo:

OBJECTIVE FUNCTION VALUE 4.000000

VARIABLE	**VALUE**
X11	0
X12	0
X13	1
X14	0
X21	1
X22	0
X23	0
X24	0
X31	0
X32	0
X33	0
X34	1
X41	0
X42	1
X43	0
X44	0

La configuración óptima es:

La reina 1 en la columna 3

La reina 2 en la columna 1

La reina 3 en la columna 4

La reina 4 en la columna 2

Solución óptima:

		Columna			
		1	**2**	**3**	**4**
Reina i	**1**	**0**	**0**	**1**	**0**
	2	**1**	**0**	**0**	**0**
	3	**0**	**0**	**0**	**1**
	4	**0**	**1**	**0**	**0**

En el siguiente cuadro se muestra la cantidad de restricciones necesarias para diferentes valores de n:

Cantidad de reinas	Diagonales negativas	Diagonales positivas	Filas	Columnas	Incluir a todas	Total de restricciones
4	5	5	4	4	1	19
7	11	11	7	7	1	37
10	17	17	10	10	1	55
20	37	37	20	20	1	115

Como se puede observar este 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.

A continuación se resolverá el problema usando la metaheurística de búsqueda tabú para n = 7.

4.3. Identificación y/o definición de las componentes del algoritmo

Primero se identifican y/o definen las características descritas en la tabla 3.1.

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:

x =	6	1	2	7	5	3	4

representa la solución en la cual la reina 1 está ubicada en la columna 6, la reina 2 en la columna 1, ..., la reina 7 en la columna 4.

Para evaluar la solución x en la función objetivo se deben analizar todas las diagonales, primeramente se presenta en EXCEL por razones didácticas, posteriormente se presentará el programa en C++.

¿Cómo asignar las reinas en EXCEL?

Como se muestra en la figura 4.1, sólo se introduce valores en las celdas B5 a B11 y "automáticamente" aparece un 1 en la columna correspondiente, pues en cada celda se ha colocado una función condicional, Si {la reina *i* es asignada a la columna *j* } entonces colocar un 1 en la celda (i,j) y cero en las demás celdas de la fila *i.*

En el ejemplo la reina 5 es colocada en la columna 5, B9 es igual a J4 (en la celda J4 hay un 5) por lo tanto en la celda J9 aparece un 1, en la celda K9 aparece un cero pues B9 es diferente de K4.

Se observa que colisionan las reinas : (6 y 7) , (4 y 7) y la (2 y 3). La función objetivo vale 3.

Se define un ***movimiento*** como 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 sólo un intercambio de reinas.

x =	6	1	2	7	5	3	4

Por ejemplo sean w y z dos posibles soluciones:

w =	6	2	1	7	3	5	4

z =	4	1	2	7	5	3	6

z pertenece al vecindario de x, pues se ha permutado la reina 1 con la 7, pero w no pertenece al vecindario de x pues se han realizado dos permutaciones simultáneamente la 2 con la 3 y la 5 con la 6.

¿Cómo calcular las colisiones en EXCEL?

Para mostrar cómo puede evaluarse la función objetivo en EXCEL se forzará a una triple colisión, por ejemplo si dada x como solución actual se permuta la reina 5 con la 3, es decir la reina 5 pasa a ocupar la columna 2 y la reina 3 pasa al lugar que ocupaba la reina 5 (la columna 5), el movimiento se muestra en la figura 4.2:

Se calculan las 2n-1 diagonales positivas y las 2n-1 diagonales negativas. En realidad sólo es necesario calcular 2n-3 pues las diagonales de los extremos – la celda F5 y la celda L11 – no pueden tomar un valor mayor que 1, pues eso indicaría que se han colocado 2 reinas en una misma celda, análogamente las celdas L5 y F11.

Cada diagonal puede tomar 8 diferentes valores posibles :
0 si no hay ninguna reina en la diagonal
1 si sólo hay una reina en la diagonal
2 si hay dos reinas en una diagonal (1 colisión)
3 si hay 3 reinas en la diagonal (una triple colisión)
....
7 si las 7 reinas están en la diagonal.

En este caso hay:

- Una colisión entre las reinas 4 y 7, es decir una diagonal con colisión
- Una colisión entre las reinas 1 y 5, es decir una diagonal con colisión
- Una triple colisión {(5,6) (5,7) (6,7)}, es decir la combinación de tres elementos tomados de dos en dos:

$$\frac{3!}{(3-2)!2!} \quad \text{simplificando} \quad \frac{k*(k-1)}{2} = 3$$

Una diagonal con una triple colisión

- Si 4 reinas estuvieran colocadas en una misma diagonal, se tendrían 6 colisiones, es decir la combinación de cuatro elementos tomados de dos en dos:

$$\frac{4!}{(4-2)!2!} \quad \text{simplificando} \quad \frac{k*(k-1)}{2} = 6$$

... en general:

Cantidad de reinas en diagonal	Ocasionan colisiones $\frac{k*(k-1)}{2}$	
2	1	2 reinas ocasionan 1 colisión
3	3	....
4	6	4 reinas ocasionan 6 colisiones
5	10	
6	15	
7	21	7 reinas ocasionan 21 colisiones

En este ejemplo en que hay 2 diagonales con una colisión y 1 diagonal con 3 reinas colisionando se tiene:

Cant de reinas en diagonal	Ocasionan colisiones k*(k-1)/2
2	1
3	3
4	6
5	10
6	15
7	21

Cantidad de diagonales con colisiones	Cantidad de colisiones
2	2
1	3
0	0
0	0
0	0
0	0
Total de colisiones	**5**

Entonces, el valor de la función objetivo luego de haber intercambiado la reina 5 con la 3 aumentó (desmejoró) de 3 a 5.

¿Cuándo se detiene el proceso?

Se define la condición de parada: El algoritmo se detendrá si se alcanza el máximo número de iteraciones, que en este caso se asigna arbitrariamente como **MAXITER**=100.

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

4.4. Aplicación del algoritmo de la búsqueda tabú

Selección de la solución inicial:

El siguiente paso consiste en generar una solución inicial, ésta puede ser el resultado de una heurística, de una selección aleatoria o de una asignación arbitraria realizada por el experto (el que tiene el know how del problema).

En este caso sea la solución inicial :

x_0 =

4	5	3	6	7	1	2

Al evaluar en EXCEL se encuentra que la función objetivo vale 4.

Colisionan las reinas: {(2,6) (6,7) (4,5) (1,2)}

Se itera para seleccionar la siguiente solución:

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

Iteración 1 (**k=1**) El vecindario está conformado por todas las posibles combinaciones de 7 elementos tomados de 2 en 2, es decir, 7!/(2!*5!) = 21

Se evalúan las 21 posibles soluciones y se seleccionan las 5 mejores

# alternativa	Permutación		Colisiones
1	1	2	7
2	1	3	7
3	1	4	5
4	1	5	3
5	1	6	3
6	1	7	2
7	2	3	5
8	2	4	2
9	2	5	4
10	2	6	2
11	2	7	3
12	3	4	8
13	3	5	4
14	3	6	3
15	3	7	5
16	4	5	6
17	4	6	4
18	4	7	3
19	5	6	2
20	5	7	5
21	6	7	3

# alternat	Permutación		Colisiones
6	1	7	2
8	2	4	2
10	2	6	2
19	5	6	2
4	1	5	3

Hay 4 movimientos que producen 2 colisiones y 6 movimientos que producen 3 colisiones.

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

Se escoge la permutación 1,7

La solución siguiente resulta de intercambiar las reinas 1 y 7

x_1 =

2	5	3	6	7	1	4

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. El tamaño de la lista (*tabú tenure*) es tres.

Solución actual x_1

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						

Mejores 5 candidatos

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

Si se intercambian las reinas 2 y 4 la función objetivo valdrá 1, sólo hay una colisión {(1,4)}

Se itera una vez más, **k=2**

Solución actual x_2

2	6	3	5	7	1	4

El movimiento (1,7) está penalizado durante 2 iteraciones y el movimiento (2,4) durante 3 iteraciones

Estructura de la lista tabú

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

Mejores 5 candidatos

cambio		colisiones
1	3	1
1	2	2
1	4	2
1	6	2
1	7	2**T**

Entre los 21 vecinos hay 1 movimiento que produce una colisión y 8 movimientos que producen 2 colisiones, se pueden seleccionar al azar o sistemáticamente tomar los primeros, en este caso el vecino (1,7) está marcado como movimiento tabú, por lo tanto aún si fuera el mejor no sería elegido para la siguiente iteración.

Se selecciona la permutación (1,3), es decir se intercambian las reinas 1 y 3; la función objetivo no mejora, hay una colisión: {(1,5)}

Se itera una vez más, **k = 3**

Solución actual x_3

3	6	2	5	7	1	4

El movimiento (1,7) está penalizado durante 1 iteración y el movimiento (2,4) durante 2 iteraciones y el (1,3) durante 3 iteraciones

Estructura de la lista tabú

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

Mejores 5 candidatos

cambio		colisiones
1	3	1**T**
1	7	1**T**
5	7	2
6	7	2
1	2	3

Las permutaciones (1,3) o (1,7) son las mejores en esta iteración, pero como son tabú no pueden ser elegidas. La permutación de las reinas 5 y 7 a pesar de que produce 2 colisiones y la función objetivo desmejora es la elegida para ser la solución en la siguiente iteración. Las colisiones que se producen son: {(3,5) (4,5)}

k = 4

Solución actual x_4

3	6	2	5	4	1	7

Observar como decrecen los tiempos de permanencia en la lista Tabú

Estructura de la lista tabú

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

Mejores 5 candidatos

cambio		colisiones
4	7	1
5	7	1**T**
1	5	2
2	5	2
2	4	3**T**

La elección del movimiento (4,7) produce una sola colisión y no es tabú.

k=5

Solución actual x5

3	6	2	7	4	1	5

Estructura de la lista tabú

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

Mejores 5 candidatos

cambio		colisiones
1	3	0**T**
5	6	1
5	7	1**T**
1	6	2
1	7	2

El movimiento (1,3) es el mejor en la iteración k = 5, pero es tabú, sin embargo realizar este movimiento produce el mejor valor encontrado hasta el momento: cero colisiones, es por eso que se recurre al ***criterio de aspiración*** para sobrepasar la condición de tabú y permitir que la siguiente solución sea (2; 6; 3; 7; 4; 1; 5) con la cual se obtiene una función objetivo igual a cero, es decir es una solución de elite. Por la naturaleza del problema no hay otras soluciones mejores, (no hay colisiones negativas) pero el proceso continúa para tratar de encontrar otras soluciones.

Si el objetivo es encontrar una solución con cero colisiones, el criterio de aspiración consistente en quitarle la restricción tabú a un movimiento que produce el "óptimo" es válido, 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."*[4]

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

Podría ser deseable incrementar el porcentaje de movimientos tabú, esto se puede lograr ya sea incrementando el tamaño de la lista tabú, (***tabu tenure***) el número de iteraciones que un movimiento está en la lista tabú o cambiando la restricción tabú.

Como el tamaño de la lista es un número finito, en algún momento un movimiento saldrá de la lista tabú y podrá ser elegido, con lo que se corre el riesgo de ciclado – una misma solución vuelve a ocurrir repetidas veces cada cierto tiempo - , para diversificar la búsqueda se usará **la memoria de largo plazo**, en este caso, la estructura que registra la frecuencia de ocurrencia de los movimientos.

Supongamos que no se está considerando ningún criterio de aspiración y se han realizado ya 75 iteraciones

Solución actual x_{75}

5	3	1	6	4	2	7

Frecuencias

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	

Mejores 5 candidatos

cambio		**FO**	**freq**
4	7	1	7
5	7	1**T**	5
1	2	2	7
1	4	2**T**	4
1	6	2	3

La lista tabú es:

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

La solución x_{75} es (5, 3, 1, 6, 4, 2, 7) y 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 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 hay regiones no exploradas 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.

Este movimiento llevará el proceso de búsqueda a regiones menos exploradas.
El procedimiento se repite hasta superar el máximo número de iteraciones permitidas.

Consideraciones previas al análisis de los resultados

- Es claro que la implementación computacional del algoritmo es imprescindible para encontrar soluciones.

- Es evidente también que el algoritmo está en función del tipo de problema, en este caso el movimiento es un intercambio de reinas, en el problema del agente viajero o en un problema de secuenciación un movimiento podría ser insertar un nodo y desplazar los otros, etc.

- En líneas generales lo central del método es la búsqueda inteligente, alternando estrategias de intensificación con estrategias de diversificación, penalizando la ocurrencia de un movimiento en algunos casos y propiciándola en otros casos, todo sobre la base de la memoria.

- El algoritmo no es una camisa de fuerza sino una pauta para disminuir el riego de caer en óptimos locales y en concordancia con la naturaleza del problema se le pueden introducir modificaciones producto del conocimiento de los expertos o de los resultados encontrados en corridas exploratorias

Variaciones que se pueden introducir al algoritmo de búsqueda tabú

Otra forma de producir la **diversificación** es la de generar una asignación aleatoria cada cierto número de iteraciones.

Para **intensificar la búsqueda** se puede usar la estructura llamada memoria intermedia[4], la idea general es identificar estructuras de solución que ocasionan "buenas soluciones" y se propicia que las componentes sean fijadas de tal modo que sólo varíen los otros atributos e intensificar así la búsqueda alrededor de las características fijadas. En este ejemplo podría ser que "es bueno que estén juntas las reinas 4 y 7" entonces se puede fijar esas reinas e intercambiar sólo las demás.

Para introducir la **oscilación estratégica** podrían hacerse dos intercambios a la vez, esto ocasionaría que la solución nueva no pertenezca al vecindario de la actual solución, pero enviaría el proceso a otras regiones.

Para introducir la estrategia de **reencadenamiento (*path relinking*)** se pueden tomar dos soluciones que ocasionan cero colisiones, por ejemplo:
Las soluciones de elite :

A =	6	3	1	4	7	5	2	la FO de A es cero, A es la solución inicial
B =	2	5	7	4	1	3	6	La FO de B es cero, B es la **solución guía**

[4] Laguna, Manuel "A Guide to implementing Tabu search", *Investigación Operativa* v. 4, n. 1, pp. 21 Abril 1994

Se tratará de llegar a B partiendo de A, no mediante el método de búsqueda tabú sino transformando A en B, introduciendo las características de B en la solución A.

En este caso A tiene a la reina 1 en la columna 6 y B la tiene en la 2, entonces en la solución A permutamos a la reina 1 de tal manera que quede en la columna 2, evaluamos, y continuamos los intercambios hasta llegar a la solución B. Esto se registra en la siguiente tabla:

	R1	R2	R3	R4	R5	R6	R7	FO
A	6	3	1	4	7	5	2	0
s1	2	3	1	4	7	5	6	2
s2	2	5	1	4	7	3	6	0
B	2	5	7	4	1	3	6	0

Como se puede observar se ha llegado de A a B en 3 pasos y en el camino se ha encontrado una nueva solución de elite (la solución s2 que también da cero colisiones).

En la figura 4.3 se representa en línea continua el camino que se siguió según el algoritmo tabú para llegar de A a B. En línea punteada se representa el reencadenamiento tomando a A como punto inicial y a B como guía.

El seleccionar a A como inicio y a B como guía es arbitrario (por eso se llaman referenciales) si se realiza el camino inverso se observa que se encuentra otra configuración que da cero colisiones.

El reencadenamiento no garantiza que se encuentren soluciones mejores, es sólo una forma de **intensificar la búsqueda** en los vecindarios de soluciones que ya probaron ser buenas (las soluciones de elite).

En todos los casos debe decidirse previamente si los cambios de estrategia serán sistemáticos o si habrá una interactividad entre el usuario y el programa computacional.

4.5. Implementación computacional en c++

Se realizó un programa en lenguaje c++ versión 3.0 Borland International Inc. 1990 1992 (ver anexo 4.2), el cual es específico para el problema presentado en este capítulo, aunque puede ser adaptado para ser utilizado en otro problema cambiando el concepto de movimiento, la construcción del vecindario y la forma de la función objetivo – en este caso se han contado las colisiones en las diagonales.

En líneas generales el programa cuenta con 3 rutinas:

I) Rutina de inicio
II) Rutina de iteraciones hasta llegar al máximo número de iteraciones permitido
III) Rutina de finalización.

I) 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 selecciona la opción de imprimir paso a paso o no
- 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 sólo para verificar los resultados contra los elaborados manualmente)

- Se inicializan con ceros la lista tabu y la tabla de frecuencias
- Se calcula la función objetivo de la solución inicial
- Se imprime la solución inicial
- Se graba en un archivo de disco con formato .txt la solución inicial y el valor de su función objetivo.

Se puede cambiar el tamaño de la lista tabú y el número de candidatos modificando las dos primeras líneas del programa.
También se puede cambiar al número máximo de reinas, que en este caso es 10, cambiando la dimensión del arreglo colisiones que sólo acepta 10 reinas en una diagonal y la dimensión del arreglo vecinos, que en este caso acepta como máximo 45 (combinaciones de 10 reinas tomadas de dos en dos).

II) Rutina de iteraciones (hasta llegar al máximo número de iteraciones permitido)

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:

    Calcular las diagonales positivas superiores e inferiores y las diagonales negativas superiores e inferiores

    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 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
- Se graba en un archivo de disco con formato .txt el número de iteración, el movimiento, la nueva solución y su función objetivo
- Se imprime en pantalla los detalles de la iteración si se seleccionó imprimir

III) Rutina de finalización

Se imprime en pantalla el nombre del archivo de texto donde se han grabado todas las iteraciones. Se cierra el archivo de disco.

Discusión de los hallazgos

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.

No se ha realizado un diseño del experimento estadístico en el cual se deben agotar exhaustivamente todos los posibles rangos de variación para los parámetros sino un diseño exploratorio guiado por los resultados obtenidos en cada paso.

Los resultados se resumen en la tabla 4.1

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

Tabla 4.1

Corrida	Parámetros		Resultados	
	Nivel de aspiración	Memoria de largo plazo	Cantidad de soluciones sin colisiones	Soluciones diferentes
0) Sólo lista tabú	100	100	30	2
1) Lista tabú + criterio de aspiración desde el comienzo	1	100	46	1
2) Lista tabú + criterio de aspiración desde iteración 20	20	100	43	2
3) Lista tabú + criterio de aspiración desde iteración 40	40	100	40	2
4) Lista tabú + criterio de aspiración desde iteración 40 +Memoria largo plazo desde iteración 40	40	40	21	6
5) Lista tabú + criterio de aspiración desde iteración 40 + Memoria largo plazo desde iteración 70	40	70	31	3
6) Lista tabú + criterio de aspiración desde iteración 70 +Memoria largo plazo desde iteración 40	70	40	21	5
7) Lista tabú + Memoria largo plazo desde iteración 20, sin criterio de aspiración	100	20	17	6
8) 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

Al usar sólo la estructura de la lista tabú (memoria de corto plazo) se observa que, si bien es cierto se obtienen dos soluciones diferentes que hacen que la cantidad de colisiones sea igual a cero, a partir de la iteración # 21 se produce un ciclado entre los movimientos (1,5) (6,7) y (2,4), pues como la *tenure* es igual a 2 cada tercera iteración se regresa al mismo movimiento.

Mientras más tarde se utiliza el criterio de aspiración: “si la solución produce cero colisiones considerarla aunque sea tabú”, se producen más soluciones diferentes.

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”, es decir cuando se diversifica más temprano.

En la tabla 4.2 se muestran las soluciones diferentes que se encontraron en las 9 corridas.

Tabla 4.2

Soluciones diferentes								Apareció en la corrida:								
	R1	R2	R3	R4	R5	R6	R7	0	1	2	3	4	5	6	7	8
Sol 1	6	1	3	5	7	2	4	1	1	1	1	1	1	1	1	1
Sol 2	7	4	1	5	2	6	3	1		1	1	1	1	1	1	1
Sol 3	3	1	6	2	5	7	4					1				
Sol 4	6	2	5	1	4	7	3					1				
Sol 5	6	4	7	1	3	5	2					1		1		
Sol 6	6	3	1	4	7	5	2					1	1			
Sol 7	2	4	1	7	5	3	6							1		
Sol 8	3	7	2	4	6	1	5							1	1	1
Sol 9	6	3	1	4	7	5	2								1	
Sol 10	4	1	3	6	2	7	5								1	1
Sol 11	2	5	1	4	7	3	6								1	
Sol 12	2	5	7	4	1	3	6									1
Total de soluciones diferentes en cada corrida :								2	1	2	2	6	3	5	6	5