matemáticas: Una publicación del ICM – ESPOL
2008, Vol. 6, No. 1

APLICACIÓN DE LA BUSQUEDA TABÚ EN EL PROBLEMA DE LAS N-REINAS

Bustamante Johni[1]

Resumen: *El problema de las n-reinas, consiste en ubicar en un tablero cuadrado de ajedrez n-reinas tal que no se ataquen entre si. La técnica empleada para buscar la solución es el método de búsqueda tabú, y su implementación se realizo en Matemática 5.1. Además se lo realiza modularmente tal que estos procesos se puedan aplicar y calibrar a gusto y experiencia del investigador.*

Palabras claves: Búsqueda tabú, reinas, vecindad, intensificación, diversificación.

1.- INTRODUCCIÓN

El problema de la n-reinas consiste en ubicar n- reinas en una tablero de ajedrez de dimensión nxn cuadritos tal que todas entre si no se ataquen, esto implica que cada reina tendrá un corredor horizontal, uno vertical y uno diagonal para ella sola.

Este problema es de combinatoria, la cantidad de posibles formas de ubicar n reinas en un tablero de nxn es un número sumamente grande, es decir para un n=100 ninguna computadora podría evaluar exhaustivamente todas las posibles combinaciones, simplemente es imposible.

2. COMPLEJIDAD DEL PROBLEMA

A continuación analizamos el tamaño de todas las posibles formas de ubicar n reinas en un tablero de tamaño nxn.

Por tanto tenemos:

$$\binom{n^2}{n} = \frac{\left(n^2\right)!}{n!\left(n^2-n\right)!}$$

$$\binom{n^2}{n} = \left(n^2-2\right)!\left(n+1\right)$$

$n=5$

$Cantidad = (n^2-2)!(n+1)$

15511210043330985984000

$n=10$

$Cantidad = (n^2-2)!(n+1)$

10369579493771572520188804317362966721190663140486846829843662655024177777025546178771273775128502031805966435647469306472353912324096000000000000000000000000

Para n=100, la cifra es sumamente grande.

Para un tablero de 100x100 el cálculo exhaustivo implicaría una cantidad de operaciones inmensamente grande, incluso excluyendo casos muy obvios ésta cifra será muy grande.

Por tanto la cifra de posibilidades es n!, por tanto para:

n=5,

5!=120

Para: n=100,

100!=933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000000000

Observamos esta última cifra, es muy grande, y realizar esta cantidad de operaciones para un computador (súper computador) es simplemente imposible.

A continuación tenemos el planteamiento del problema como un problema de Programación Lineal Entera Binario, y la búsqueda de la solución por medio del método heurístico conocido como "Búsqueda Tabú".

3.- PLANTEAMIENTO DEL MODELO MATEMÁTICO

$$x_{i,j} = \begin{cases} 1 & si\ la\ reina\ se\ encuentra\ en\ la\ posición\ (i,j) \\ 0 & si\ no \end{cases}$$

Los conjuntos son

$I = \{1,2,3,..,n\}$

$J = \{1,2,3,..,n\}$

$IJ^{+} = \{2,3,..,(n-1)\}$

$IJ^{-} = \{-(n-1),...,-2,-1,0,1,2,..,n-1\}$

El problema queda planteado:

$$Max\ z = \sum_{\forall i,j} x_{i,j} \quad \text{(Función Objetivo)}$$

s.t.

$$\begin{cases} \sum_{i \in I} x_{i,j} = 1 & \forall j \\ \sum_{j \in J} x_{i,j} = 1 & \forall i \\ \sum_{i+j \in IJ^{+}} x_{i,j} = 1 & \forall i,j \\ \sum_{i-j \in IJ^{-}} x_{i,j} = 1 & \forall i,j \end{cases}$$

[1] *Bustamante Johni, Master, Profesor de la Escuela Superior Politécnica del Litoral (ESPOL); (e_mail: jobustam@espol.edu.ec)*

(Conjunto de factibilidad o población)
Las condiciones en orden son:
Una sola reina en la columna j
Una sola reina en la fila i
Una sola reina en la diagonal i+j
Una sola reina en la diagonal i-j
La búsqueda Tabú consiste en:
1.- Definir muy claramente la población.
2.- Definir muy claramente la vecindad.
3.- Definir la intensificación de la búsqueda.
4.- Definir la lista Tabú.
5.- Definir la Diversificación en función de la lista Tabú.

3.1.- POBLACIÓN

Definimos como población todas las posibles formas de ubicar una cantidad de reinas en un tablero tales que no se ataquen entre ellas, es decir el conjunto de Factibilidad mostrado en el planteamiento del problema.

3.2.- FUNCIÓN OBJETIVO

El objetivo es alcanzar el mayor número de reinas de la población antes definidas, Sí esta solución existe será menor o igual a n.
Por ejemplo para un tablero de 2X2 es evidente que el máximo de reinas será 1.
Por ejemplo para un tablero de 10x10 se verifica en este trabajo que el máximo de reinas es 10.
A continuación el detalle de la construcción del algoritmo de Búsqueda Tabú.

4. IMPLEMENTACIÓN

4.1.- MÓDULO 1.- Crear una matriz.

```
fila=10;
Ajedrez=IdentityMatrix[fila];
For[i=1,i≤fila,
  {
   For[j=i,j≤fila,
     {
      Ajedrez[[i,j]]=10;
      Ajedrez[[j,i]]=10;
      If[j==fila,,
        {
         Ajedrez[[i,j+1]]=0;
         Ajedrez[[j+1,i]]=0;
        },
       ];
      };j=j+2];
  };i++];
Ajedrez//MatrixForm
```

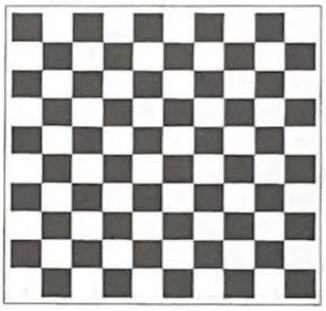
FIGURA 1
APLICACIÓN DE LA BUSQUEDA TABÚ EN EL PROBLEMA DE LAS N-REINAS
Matriz

$$\begin{pmatrix} 10 & 0 & 10 & 0 & 10 & 0 & 10 & 0 & 10 & 0 \\ 0 & 10 & 0 & 10 & 0 & 10 & 0 & 10 & 0 & 10 \\ 10 & 0 & 10 & 0 & 10 & 0 & 10 & 0 & 10 & 0 \\ 0 & 10 & 0 & 10 & 0 & 10 & 0 & 10 & 0 & 10 \\ 10 & 0 & 10 & 0 & 10 & 0 & 10 & 0 & 10 & 0 \\ 0 & 10 & 0 & 10 & 0 & 10 & 0 & 10 & 0 & 10 \\ 10 & 0 & 10 & 0 & 10 & 0 & 10 & 0 & 10 & 0 \\ 0 & 10 & 0 & 10 & 0 & 10 & 0 & 10 & 0 & 10 \\ 10 & 0 & 10 & 0 & 10 & 0 & 10 & 0 & 10 & 0 \\ 0 & 10 & 0 & 10 & 0 & 10 & 0 & 10 & 0 & 10 \end{pmatrix}$$

Esta matriz la podemos representar gráficamente como un tablero de ajedrez.

```
Print["Solucion
inicial",ArrayPlot[Ajedrez]];
```

FIGURA 2
APLICACIÓN DE LA BUSQUEDA TABÚ EN EL PROBLEMA DE LAS N-REINAS
Ajedrez



Es evidente este ultimo comando permitirá representar las soluciones gráficamente.

4.2.- MÓDULO 2

Obtención de la solución inicial
Idea.- Numeración del tablero de ajedrez:
Filas.- i=1, 2,3,…, fila.
Columnas.- j=1, 2,3,.., fila.
Diagonales derechas.- De izquierda a derecha y de abajo hacia arriba, estas diagonales tienen la siguiente característica, la suma de i+j es constante para todas las cuadriculas de esta diagonal,

i+j=2, 3,4,.., fila+1,..,2(fila).

Diagonales izquierdas.- De izquierda a derecha y de arriba hacia abajo, estas diagonales tienen la siguiente característica i-j es constante para todas las cuadriculas que se encuentran en la diagonal.

i-j=fila-1,..,3,2,1,0,-1,-2,-3,..,-fila+1.

Estas características de las cuadriculas nos permiten verificar que cuando se ubique una reina en una cuadricula ésta se encuentre solo en una de las filas, columna y diagonales.

4.3.- ALGORITMO GLOTÓN.

4.3.1.- Ponemos una reina en una posición inicial a la cual llamamos pivote. Ejemplo (1,4), esto es en la fila 1, columna 4, diagonal derecha 5 y diagonal izquierda -3, es decir la posición es (1, 4, 5,-3).

4.3.2.- Comenzamos desde la posición de las filas {2, 3, 4,5,..fila}, columnas {1,2,3,5,6,..,fila}, diagonales derechas {2,3,4,6,7,..,2fila}, diagonales izquierdas {fila-1,..,5,4,3,2,1,0,-1,-2,-4,-5,..,-fila+1}, obsérvese que en estos conjuntos no se encuentra las respectivas fila, columnas, y diagonales del elemento pivote, esto nos garantiza que la reina siguiente se ubique tal que no se atacan entre ellas, luego de ubicar la siguiente reina entonces eliminamos las posiciones de esta nueva reina.

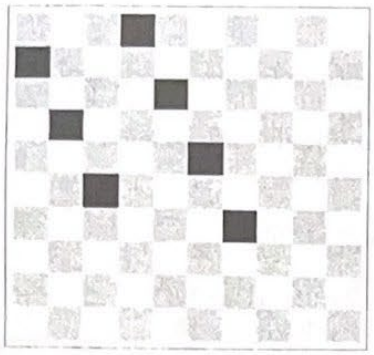
4.3.3.- Se repite el paso dos hasta que los conjuntos de filas y columnas estén vacíos.

Obsérvese que este proceso realiza una cantidad de iteraciones menor a (fila)(fila+1)/2.

```
Cf={};
Cc={};
Dder={};
Dizq={};
Contador=1;
Sol={};

pivote={1,4};
Print["La Semilla tomada es : ",pivote];
Sol=Union[Sol,{{pivote[[1]],pivote[[2]]
}}];
Cf=Union[Cf,{pivote[[1]]}];
Cc=Union[Cc,{pivote[[2]]}];
Dder=Union[Dder,{pivote[[1]]+pivote[
[2]]}];
Dizq=Union[Dizq,{pivote[[1]]-
pivote[[2]]}];
nf=1;
nc=1;
ndd=1;
ndi=1;
Ctotal=Range[fila];
Cfcom=Complement[Ctotal,Intersection[
Ctotal,Cf]];
Cccom=Complement[Ctotal,Intersection
[Ctotal,Cc]];
Nfila=Length[Cfcom];
Ncolu=Length[Cccom];
For[j=1,j□Ncolu,
  {
    For[i=1,i□Nfila,
      {
        Ni=Cfcom[[i]];
        Nj=Cccom[[j]];
        pivote={Ni,Nj};

If[Length[Union[Cf,{pivote[[1]]}]]==nf
+1,
      {

If[Length[Union[Cc,{pivote[[2]]}]]==n
c+1,
        {

If[Length[Union[Dder,{pivote[[1]]+piv
ote[[2]]}]]==ndd+1,
          {

If[Length[Union[Dizq,{pivote[[1]]-
pivote[[2]]}]]==ndi+1,
            {

Sol=Union[Sol,{{pivote[[1]],pivote[[2]]
}}];

NNsol=Length[Sol];

Cf=Union[Cf,{pivote[[1]]}];

Cc=Union[Cc,{pivote[[2]]}];

Dder=Union[Dder,{pivote[[1]]+pivote[
[2]]}];

Dizq=Union[Dizq,{pivote[[1]]-
pivote[[2]]}];
              nf=nf+1;
              nc=nc+1;
              ndd=ndd+1;
              ndi=ndi+1;
              }];
            }];
          }];
        }];
      };i++];
  };j++];
Print["Solucion Inicial es : ",Sol];
Print["En un tablero de ",fila," casilleros
tenemos ",Length[Sol]," reinas"];
k=Length[Sol];
For[i=1,i□k,Ajedrez[[Sol[[i,1]],Sol[[i,2
]]]]=30;i++];
Ajedrez//MatrixForm;
Print["Solucion
inicial",ArrayPlot[Ajedrez]];
CuantasIntersecciones=0;
```

La Semilla tomada es: {1,4}
Solución Inicial es: {{1,4}, {2,1}, {3,5}, {4,2}, {5,6}, {6,3}, {7,7}}
En un tablero de 10 casilleros tenemos 7 reinas

FIGURA 2
APLICACIÓN DE LA BUSQUEDA TABÚ EN EL PROBLEMA DE LAS N-REINAS
Tablero 10 casilleros, 7 reinas



4.4.- MÓDULO 3

Creación del vecino.

4.4.1.- DEFINICIÓN DE LA VECINDAD.

En este método la vecindad se define de la siguiente forma:

Eliminamos la reina de la primera fila y las otras reinas suben una fila, dejando libre posiciones en las cuales se ubican las nuevas reinas y así sucesivamente, La cantidad de vecinos a evaluar se puede calibrar a deseos del usuario, de estos vecinos se escoge el mejor (Se tiene que definir el mejor vecino).

4.4.2.- CANTIDAD DE ELEMENTOS EN LA VECINDAD.

Existen varias formas de determinar la cantidad de vecinos en la vecindad, entre ellas tenemos:

4.4.2.a.- Explorar los vecinos hasta que los vecinos desmejoren en tres reinas. (Aquí la cantidad de vecinos en la vecindad es una cantidad variable, para cada vecindad)

Por ejemplo:

Cantidad de reinas: 7,8,7,6,7,6,7,6,5 se detiene, por cuanto el máximo es 8 y desmejoro de 8 a 5 (se detiene por cuanto la desmejora es en 3 reinas), se puede poner condición de desmejora de 2 reinas o 5 reinas, lo cual queda a gusto del usuario.

En este ejemplo tenemos que la vecindad tiene 9 vecinos.

4.4.2.b.- En mi propuesta yo tomo una cantidad fija de vecinos, dada por el usuario en proporción de la cantidad de filas, es decir hasta (intensidad*fila), si la intensidad es de 2 entonces se tiene 2*fila vecinos.

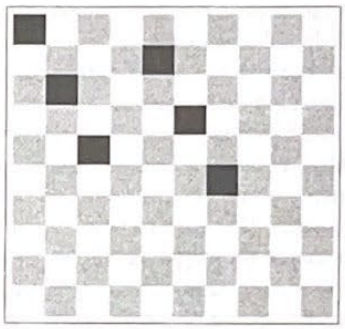
De esta forma la experiencia me indica cuando es mayor la intensidad obtenemos la solución optima.

4.4.2.c.- Programa para obtener el primer vecino.

```
Tt={Sol[[1]]};
Sol=Complement[Sol,Intersection[Sol,Tt
]];
n=Length[Sol];
For[i=1,i<n,
 {
  Sol[[i,1]]=Sol[[i,1]]-1;
  };i++];
k=Length[Sol];
For[i=1,i<k,Ajedrez[[Sol[[i,1]],Sol[[i,2
]]]]=30;i++];
Print["Vecino",ArrayPlot[Ajedrez]];
```

Para representar esto gráficamente, primero aplicamos el algoritmo para crear la matriz (enceramos la matriz) y luego el modulo 3.2.c Obtenemos lo siguiente:

FIGURA 3
APLICACIÓN DE LA BUSQUEDA TABÚ EN EL PROBLEMA DE LAS N-REINAS
Matriz encerada



En este ejemplo tenemos:

Sol= {{1,1}, {2,5}, {3,2}, {4,6}, {5,3}, {6,7}}

Aún el vecino no esta valorado, por cuanto al aplicar este modulo tenemos una reina menos, y más filas y comunas libres las cuales permiten ubicar nuevas reinas.

En el ejemplo tenemos los conjuntos filas y columnas no ocupadas:

Cfcom={7,8,9,10}
Cccom={4,8,9,10}

4.5.- MÓDULO 4

Exploración (valoración de los vecinos).

Este módulo tiene tres momentos importantes

4.5.1.- INTENSIDAD

Con estos conjuntos aplicamos el modulo 2 (Glotón) para ubicar las nuevas reinas (quedando valorado el nuevo vecino):

Este proceso realizamos (intensidad)*(fila) veces. (Toda la vecindad)

4.5.2.- TABÚ

Cuando exploramos los vecinos tenemos nuevas soluciones factibles en nuestra vecindad, para no caer en la exploración de vecinos ya explorados introducimos la lista tabú, en nuestro ejemplo la lista tabú es de longitud 10

Quedando el tamaño de esta lista a gusto del usuario, y en esta lista yo guardo solamente la posición de la reina eliminada (no de todas las reinas en cada solución, para hacer el programa menos pesado, obteniendo buenos resultados).
Por lo general en este proceso es posible que aparezca muy rápido una reina repetida en la lista tabú, siendo no así el restante de las reinas de estas soluciones para ello introduzco un nuevo parámetro (en el ejemplo el parámetro 3), este parámetro hace que, después de la tercera aparición de la reina en la lista tabú entonces se detiene la búsqueda, mediante este parámetro existe la posibilidad de que el proceso se cicle o sea muy extenso, para ello se introduce otro parámetro de parada (la experiencia me indica que no es necesario este nuevo parámetro de parada, pero lo ponemos).

4.5.3.- DIVERSIFICACIÓN

Luego de que el proceso de intensificación se realizo (se detuvo por los criterios de parada) sin obtener la solución optima, entonces procedemos a diversificar la búsqueda (yo lo llamo "Cambio de la dirección de búsqueda").
Esto lo realizamos siguiendo el siguiente criterio.
Tomamos el mejor vecino antes explorado, o cualquier vecino cuya solución no sea muy desmejorada. (La experiencia me indica que es mucho mejor tomar un vecino cuya solución fue desmejorada en dos reinas, y a este vecino lo clasificamos como el mejor vecino para continuar la exploración).
Luego de tomar el mejor vecino, el proceso de diversificación lo realizamos quitando dos filas (Aplicamos dos veces 3.2.c y luego aplicamos el glotón), nótese que parece ser la misma forma de elección del vecino antes descrita, no siendo así por cuanto, al eliminar dos reinas de las dos primeras filas y subiendo las reinas restantes a las filas dos filas anteriores, tenemos mas dos filas libres y mas dos columnas libres.

4.5.3.1.- CONTINUAR LA INTENSIFICACIÓN.

Luego de realizar la diversificación, tenemos la lista tabú nueva vacía y procedemos a repetir este modulo.

4.6.- CRITERIO DE PARADA FINAL

Si en el proceso de intensificación no se ha obtenido la solución optima entonces se aplica el criterio de diversificación, y como ultimo criterio de parada tenemos que la diversificación la realizamos una cantidad fija de veces, en el ejemplo tengo 8 (es decir el proceso máximo pasará 8 veces el criterio de diversificación), siendo este parámetro a criterio del usuario (La experiencia empírica me dice que con 8 es suficiente para obtener la solución óptima).

4.7.- IMPLEMENTACIÓN TOTAL.

Con los módulos 1, 2.3, 3.2.c y la implementación de los parámetros de intensificación, diversificación y los criterios de parada obtenemos el siguiente programa:

```
Print["_____Dete
rmincacion de la Solucion inicial"];
fila=100;
Ajedrez=IdentityMatrix[fila];
For[i=1,i□fila,
  {
   For[j=i,j□fila,
     {
      Ajedrez[[i,j]]=10;
      Ajedrez[[j,i]]=10;
      If[j==fila,,
       {
        Ajedrez[[i,j+1]]=0;
        Ajedrez[[j+1,i]]=0;
        },
       ];
      };j=j+2];
  };i++];
Ajedrez//MatrixForm;
Cf={};
Cc={};
Dder={};
Dizq={};
Contador=1;
Sol={};
pivote={1,4};
Print["La Semilla tomada es : ",pivote];
Sol=Union[Sol,{{pivote[[1]],pivote[[2]]}}];
Cf=Union[Cf,{pivote[[1]]}];
Cc=Union[Cc,{pivote[[2]]}];
Dder=Union[Dder,{pivote[[1]]+pivote[[2]]}];
Dizq=Union[Dizq,{pivote[[1]]-pivote[[2]]}];
nf=1;
nc=1;
ndd=1;
ndi=1;
Ctotal=Range[fila];
Cfcom=Complement[Ctotal,Intersection[Ctotal,C
f]];
Cccom=Complement[Ctotal,Intersection[Ctotal,C
c]];
Nfila=Length[Cfcom];
Ncolu=Length[Cccom];
For[j=1,j□Ncolu,
  {
   For[i=1,i□Nfila,
     {
      Ni=Cfcom[[i]];
      Nj=Cccom[[j]];
      pivote={Ni,Nj};

If[Length[Union[Cf,{pivote[[1]]}]]==nf+1,
       {
```

```
If[Length[Union[Cc,{pivote[[2]]}]]==nc+1,
        {
If[Length[Union[Dder,{pivote[[1]]+pivote[[2]]}]]
==ndd+1,
            {
If[Length[Union[Dizq,{pivote[[1]]-
pivote[[2]]}]]==ndi+1,
                {
Sol=Union[Sol,{{pivote[[1]],pivote[[2]]}}];
                NNsol=Length[Sol];
                Cf=Union[Cf,{pivote[[1]]}];
Cc=Union[Cc,{pivote[[2]]}];
Dder=Union[Dder,{pivote[[1]]+pivote[[2]]}];
Dizq=Union[Dizq,{pivote[[1]]-pivote[[2]]}];
                nf=nf+1;
                nc=nc+1;
                ndd=ndd+1;
                ndi=ndi+1;
                }];
            }];
        }];
    };i++];
  };j++];
Print["Solucion Inicial es : ",Sol];
Print["En un tablero de ",fila," casilleros tenemos
",Length[Sol]," reinas"];
k=Length[Sol];
For[i=1,i□k,Ajedrez[[Sol[[i,1]],Sol[[i,2]]]]=30;i+
+];
Ajedrez//MatrixForm;
Print["Solucion inicial",ArrayPlot[Ajedrez]];
CuantasIntersecciones=0;
q=1;
While[q□20 &&Length[Sol]!=fila,
  {
   Print["_____Busqueda
tabu: ",q];
   q=q+1;
   longitudTabu=IntegerPart[fila/10]-1;
   Intensidad=5;
   Diversificacion=8;
   tabu=Intensidad*fila;
   RestriccionTabu={};
   GeneralTabu={};
   status=0;
   Ctotal=Range[fila];
   diversificar=0;
   NNsol=Length[Sol];
   MMayor=NNsol;
   ListaTabu={};
   elementoQuitado=Part[Sol,{1}];
   CuantasIntersecciones=0;
   While[NNsol<fila &&status□tabu&&
diversificar□Diversificacion&&CuantasIntersecci
ones□10,
     {
      status=status+1;
RestriccionTabu=Union[RestriccionTabu,{status-
>NNsol->{Sol}}];
      Cf={};
      Cc={};
      Dder={};
      Dizq={};
      Tt={Sol[[1]]};
      ListaTabu=Union[ListaTabu,Tt];
GeneralTabu=Union[GeneralTabu,{status□{Lista
Tabu}}];
      If[Length[ListaTabu]>=longitudTabu,
       {
ListaTabu=Complement[ListaTabu,Intersection[L
istaTabu,{ListaTabu[[1]]}]]
       }
      ];
      Sol=Complement[Sol,Intersection[Sol,Tt]];
      n=Length[Sol];
      For[i=1,i□n,
       {
        Sol[[i,1]]=Sol[[i,1]]-1;
        };i++];
      elementoQuitado={Sol[[1]]};
      For[i=1,i□n,
       {
        Cf=Union[Cf,{Sol[[i,1]]}];
        Cc=Union[Cc,{Sol[[i,2]]}];
Dder=Union[Dder,{Sol[[i,1]]+Sol[[i,2]]}];
        Dizq=Union[Dizq,{Sol[[i,1]]-
Sol[[i,2]]}];
        };i++];
      nf=nf-1;
      nc=nc-1;
      ndd=ndd-1;
      ndi=ndi-1;
Cfcom=Complement[Ctotal,Intersection[Ctotal,C
f]];
Cccom=Complement[Ctotal,Intersection[Ctotal,C
c]];
      Nfila=Length[Cfcom];
      Ncolu=Length[Cccom];
      For[j=1,j□Ncolu,
       {
        For[i=1,i□Nfila,
          {
           Ni=Cfcom[[i]];
           Nj=Cccom[[j]];
           pivote={Ni,Nj};
```

```
If[Length[Union[Cf,{pivote[[1]]}]]==nf+1,
            {

If[Length[Union[Cc,{pivote[[2]]}]]==nc+1,
                {

If[Length[Union[Dder,{pivote[[1]]+pivote[[2]]}]]
==ndd+1,
                    {

If[Length[Union[Dizq,{pivote[[1]]-
pivote[[2]]}]]==ndi+1,
                        {

Sol=Union[Sol,{{pivote[[1]],pivote[[2]]}}];
                            NNsol=Length[Sol];

Cf=Union[Cf,{pivote[[1]]}];

Cc=Union[Cc,{pivote[[2]]}];

Dder=Union[Dder,{pivote[[1]]+pivote[[2]]}];

Dizq=Union[Dizq,{pivote[[1]]-pivote[[2]]}];
                            nf=nf+1;
                            nc=nc+1;
                            ndd=ndd+1;
                            ndi=ndi+1;
                            }];
                        }];
                    }];
                }];
            };i++];
        };j++];
      If[MMayor!=NNsol,MMayor=NNsol];
      diversificar=MMayor-NNsol;

If[Length[Intersection[ListaTabu,elementoQuitad
o]]!=0,CuantasIntersecciones=CuantasInterseccio
nes+1];
      }];

RestriccionTabu=Union[RestriccionTabu,{status+
1->Length[Sol]->{Sol}}];
    Print["tablero de ",fila," Filas con ",
Length[Sol] ," Reinas"];

Print["_____Diversificaci
on o cambio de la direccion de la busqueda"];
    direccion=3;
    For[k=1,k!=direccion,
     {
      If[Length[Sol]!=fila,
        {
         For[i=1,i!=Length[Sol],
          {
           Sol[[i,2]]=Sol[[i,2]]-1;
           };i++];
         pariad=Position[Sol,0];
         Print["Ceros ",pariad];
         If[pariad!={},
          {
           Tt={Sol[[pariad[[1,1]]]]};

Sol=Complement[Sol,Intersection[Sol,Tt]];
           nf=nf-1;
           nc=nc-1;
           ndd=ndd-1;
           ndi=ndi-1;
           }];
        }];
      };k++];

If[Length[Intersection[ListaTabu,elementoQuitad
o]]!=0,CuantasIntersecciones=CuantasInterseccio
nes+1];
    }];
RestriccionTabu=Union[RestriccionTabu,{status+
1->Length[Sol]->{Sol}}];
Print["tablero de ",fila," Filas con ", Length[Sol]
," Reinas"];
Print["_____Presentac
ion resultados "];
RestriccionTabu=Union[RestriccionTabu,{status+
1->Length[Sol]->{Sol}}];
Print["Solucion Inicial: ",RestriccionTabu[[1]]];
Print["tablero de ",fila," Filas con ", Length[Sol]
," Reinas"];
Print["Solucion : ",Sol];
```

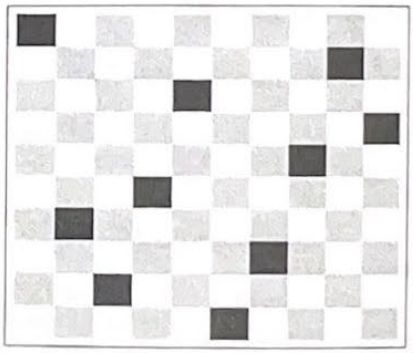
5.- RESULTADOS

Corriendo el programa con n=10 tenemos:

FIGURA 4

APLICACIÓN DE LA BUSQUEDA TABÚ EN EL PROBLEMA DE LAS N-REINAS

Matriz Resultante



Solución:
{{1,1},{2,9},{3,5},{4,10},{5,8},{6,4},{7,2},{8,},{9,3},{10,6}}
Y el tiempo de resolución es de 0.03 seg. es decir 3 centésimas de segundos.

A continuación presento una tabla en la que se puede apreciar el tiempo de ejecución en función del tamaño del tablero de ajedrez.

TABLA I

APLICACIÓN DE LA BUSQUEDA TABÚ EN EL PROBLEMA DE LAS N-REINAS

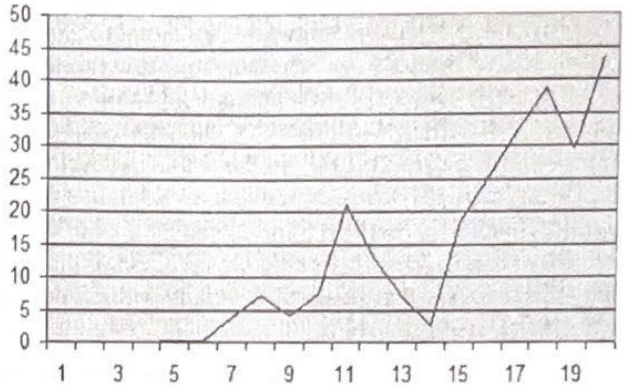
Tiempo de Ejecución y Tamaño del Tablero

n	Tiempo
5	0.04
10	0.03
15	0.08
20	0.11
25	0.141
30	0.35
35	3.806
40	7.3
45	4.146
50	7.491
55	20.94
60	12.828
65	6.99
70	2.574
75	18.577
80	25.296
85	32.687
90	38.806
95	29.482
100	43.683

GRÁFICO I

APLICACIÓN DE LA BUSQUEDA TABÚ EN EL PROBLEMA DE LAS N-REINAS

Tiempo de Ejecución y Tamaño del Tablero



Representación grafica tiempo en segundo en función del tamaño del tablero.

REFERENCIAS BIBLIOGRÁFICAS

[1]. **GONZÁLEZ A., CALDERÓN S., HIDALGO R., CABALLERO R., COBO A., ORDÓÑEZ J.** *"MATHEMATICA Programación matemática en la economía y la Empresa"*. Departamento de Economía Aplicada (Matemáticas), Facultad de Ciencias Económicas y Empresariales y Escuela Universitaria de Estudios Empresariales. Universidad de Málaga. Ed. RA-MA, España, 1999.

[2]. **DIAZ A., GLOVER F., GHAZIRI H., GONZALEZ J., LAGUNA M., MOSCATO P., TSENG F.** *"Optimización Heurística y Redes Neuronales"*, Ed. Paraninfo, España, 1996.