Módulo1 - 1

```
Option Explicit

Public Const iMin = 1
Public Const iMax = 25
Public Const jMin = 1
Public Const jMax = 25

Type Block
    i As Integer
    j As Integer
    Value As Double
    BestPredID As Integer 'Indice del explorador que mejor predice el bloque
    IsSample As Boolean

End Type

Type Sample

    ID As String
    i As Integer
    j As Integer
    Value As Double

End Type

Type Explorer

    'Best: se mueve al bloque mejor predicho. Worst: se mueve al bloque peor predicho. Random: Se muev
e a un bloque al azar.
    'Tracker: se mueve entre la muestra de origen y la primera muestra encontrada, asignando un valor
en funcion de distancia a ambas.
    Caste As String
    Behavior As String 'Exploring, Returning

    IDExp As Integer

    'Birthplace
    i0 As Integer
    j0 As Integer

    'Current Location
    i As Integer
    j As Integer

    'Estas constantes suman 1
    cAVG As Double
    cMax As Double
    cMin As Double
    cSup As Double
    cInf As Double
    cAVGS As Double

    Predicted As Double

    dSampValue As Double 'Distancia entre valore predicho y valor de muestra real

    'Informacion de ultima muestra visitada
    LSi As Integer
    LSj As Integer
    LastSample As Variant 'Valor de última muestra visitada

End Type


Sub Anterpolator()

 Const Nblocks = iMax * jMax
 Const NSamples = 25

 Const MaxExplorers = Nblocks
 Dim Explorers(1 To MaxExplorers) As Explorer
 Dim ExpRanking(1 To MaxExplorers) As Integer
 Dim NExplorers As Integer
```

Módulo1 - 2

```
 Dim ColorBest As Variant
 Dim ColorWorst As Variant
 Dim ColorTracker As Variant
 Dim ColorRandom As Variant
 Dim ColorReturn As Variant

 Dim mMin As Integer
 Dim mMax As Integer
 Dim mN As Integer
 'Dim Muestras(iMin To iMax, jMin To jMax) As Block

 Dim RowMin As Integer
 Dim RowMax As Integer
 Dim ColMin As Integer
 Dim ColMax As Integer

 Dim bi As Integer
 Dim bj As Integer
 Dim Leyij As Variant
 Dim Leybibj As Double
 Dim RowExpIni As Integer
 Dim RowExpFin As Integer

 Dim i As Integer
 Dim j As Integer

 Dim Rowi As Integer

 Dim Map(iMin To iMax, jMin To jMax) As Variant

 Dim AvgBlocks As Double
 Dim BVij As Double
 Dim gk As Variant 'Integer
 Dim gl As Variant 'Integer

 Dim MaxNeighbour As Double
 Dim MinNeighbour As Double
 Dim Supremo As Double 'La menor de las leyes mayores
 Dim Infimo As Double 'La mayor de las leyes menores
 Dim AvgSamples As Double 'Promedio de las muestras adyacentes
 Dim NAdjSamples As Integer 'Número de muestras adyacentes
 Dim NNeighbours As Integer
 Dim ExpN As Integer
 Dim AntValueij As Double

 'Variables para generación de ponderadores:
 Dim cAvg1 As Double, cMax1 As Double, cMin1 As Double, cSup1 As Double, cInf1 As Double, cAvgs1 As Do
uble
 Dim cAvg2 As Double, cMax2 As Double, cMin2 As Double, cSup2 As Double, cInf2 As Double, cAvgs2 As Do
uble


 Dim Iterations As Integer
 Dim Iterationi As Integer

 Dim dFade As Double
 Dim Caping As Double

 Dim ShowExplorers As Boolean
 Dim StartOnSample As Boolean
 Dim BreedAnywhere As Boolean
 Dim KeepPos As Boolean

 mMin = 4
 mMax = 28

 RowMin = 4
 RowMax = 28

 ColMin = 2
 ColMax = 26

 RowExpIni = RowMin
```

Módulo1 - 3

```
Iterations = 5000
dFade = 0 '0.05
Caping = 0


BreedAnywhere = False 'True hace que se extingan todos excepto los best
KeepPos = True 'Explorador modificado mantiene su posicion
StartOnSample = False 'True: Exploradores siempre nacen en un bloque con muestra
ShowExplorers = True

ColorBest = vbGreen
ColorWorst = vbRed
ColorTracker = vbMagenta ' RGB(255, 0, 255)
ColorRandom = vbBlue
ColorReturn = vbWhite

Randomize

'Se inicializa el mapa:
Range("B" & RowMin & ":Z" & RowMax).Select
Selection.ClearContents
Selection.Font.Color = RGB(128, 128, 128)
Selection.Font.Bold = False
Selection.Borders.Color = RGB(128, 128, 128)
Selection.Borders.Weight = 2
Range("A1").Select

'Se genera arreglo de coordenadas y se añaden muestras:
Dim BlkCoords(1 To Nblocks) As Block
Dim Blocks(iMin To iMax, jMin To jMax) As Block
Dim Samples(1 To NSamples) As Sample
Dim n As Integer
Dim m As Integer

n = 1
For i = iMin To iMax
   For j = jMin To jMax

       Blocks(i, j).i = i
       Blocks(i, j).j = j

       Map(i, j) = GetLey(i, j)
       Blocks(i, j).BestPredID = -1
       Blocks(i, j).IsSample = (Map(i, j) <> "")

       If Map(i, j) <> "" Then
           Blocks(i, j).Value = Map(i, j)
           bi = i + (RowMin - 1)
           bj = j + (ColMin - 1)
           Worksheets("Mapa").Cells(bi, bj).Value = Map(i, j)
           Worksheets("Mapa").Cells(bi, bj).Font.Color = RGB(255, 255, 255)
       Else
           Blocks(i, j).Value = 0
       End If

       BlkCoords(n) = Blocks(i, j)

       n = n + 1

   Next j
Next i

'Se calcula seudovariograma de muestras:
'Recordar que el mapa esta rotado 90° hacia la derecha, la coordenada N es la horizontal y la E es la
vertical.
Dim xm As Integer, ym As Integer, xn As Integer, yn As Integer
Dim Leym As Double, Leyn As Double

Rowi = RowMin - 1
For m = 1 To NSamples

   xm = Worksheets("Mapa").Range("AD" & m + (RowMin - 1)).Value
```

```
    ym = Worksheets("Mapa").Range("AE" & m + (RowMin - 1)).Value
    Leym = Worksheets("Mapa").Range("AF" & m + (RowMin - 1)).Value

    For n = (m + 1) To NSamples

        'Rowi = RowMin + (m - 1) * (n - 2) + (n - 2)
        Rowi = Rowi + 1

'        If m = 3 And n = 8 Then
'            Beep
'        End If

        xn = Worksheets("Mapa").Range("AD" & n + (RowMin - 1)).Value
        yn = Worksheets("Mapa").Range("AE" & n + (RowMin - 1)).Value
        Leyn = Worksheets("Mapa").Range("AF" & n + (RowMin - 1)).Value

        Worksheets("Mapa").Range("BP" & Rowi).Value = "(" & m & ", " & n & ")" 'Rowi - (RowMin - 1)
        Worksheets("Mapa").Range("BQ" & Rowi).Value = Sqr((xm - xn) ^ 2 + (ym - yn) ^ 2) 'distancia en
tre m y n
        Worksheets("Mapa").Range("BR" & Rowi).Value = Abs(Leym - Leyn) 'diferencia de Ley
        Worksheets("Mapa").Range("BS" & Rowi).Value = Azimuth(xm, ym, xn, yn) * (180 / (4 * Atn(1))) '
Angulo


    Next n
 Next m

 DoEvents

 'Se randomiza el arreglo de coordenadas
 Dim nRand As Integer
 For n = 1 To iMax * jMax

    Dim blkAux As Block
    nRand = Int(Rnd * (iMax * jMax - (n - 1))) + n
    blkAux = BlkCoords(n)
    BlkCoords(n) = BlkCoords(nRand)
    BlkCoords(nRand) = blkAux

 Next n

 DoEvents

 'SE GENERAN LOS EXPLORADORES
 Dim fBest As Double, fWorst As Double, fRandom As Double
 Dim RndExp As Double

 NExplorers = 50 '0 '200 'Int((iMax * jMax) / 2)
 fBest = 1 / 4
 fWorst = 0 / 4
 fRandom = 2 / 4
 'fTracker = 1

 RowExpFin = RowExpIni + NExplorers - 1
 Range("AM" & RowMin & ":AW" & (RowMin + MaxExplorers - 1)).Select
 Selection.ClearContents

 Range("BC" & RowMin & ":BM" & (RowMin + MaxExplorers - 1)).Select
 Selection.ClearContents

 For Rowi = RowMin To (RowMin + NExplorers - 1)

    ExpN = Rowi - RowMin + 1
    Worksheets("Mapa").Range("AT" & 2) = ExpN

    RndExp = Rnd
    If RndExp <= fBest Then
        Explorers(ExpN).Caste = "Best"
    ElseIf RndExp <= fWorst Then
        Explorers(ExpN).Caste = "Worst"
    ElseIf RndExp <= fRandom Then
        Explorers(ExpN).Caste = "Random"
    Else
```

```
        Explorers(ExpN).Caste = "Tracker"
    End If

    'Tracker: siempre nace en una muestra, y sale a explorar. Cuando encuentra otra muestra, se devuel
ve a su lugar de origen
    'Marcando cada casilla con un valor que es función de la distancia a la muestra de origen y la últ
ima muestra visitada.
    'Al llegar a su origen, se repite el ciclo.
    'En otras palabras, el comportamiento de una obrera en Ant Farm Simulator

    Explorers(ExpN).IDExp = ExpN

    Explorers(ExpN).i = Int(iMax * Rnd) + iMin
    Explorers(ExpN).j = Int(jMax * Rnd) + jMin

    If StartOnSample Or Explorers(ExpN).Caste = "Tracker" Then
        While Not EsMuestra(Explorers(ExpN).i, Explorers(ExpN).j)
            Explorers(ExpN).i = Int(iMax * Rnd) + iMin
            Explorers(ExpN).j = Int(jMax * Rnd) + jMin
        Wend
    End If

    'Posicion inicial de la muestra
    Explorers(ExpN).i0 = Explorers(ExpN).i
    Explorers(ExpN).j0 = Explorers(ExpN).j

    'Se inicializa ultima muestra visitada:
    Explorers(ExpN).LastSample = -1
    Explorers(ExpN).Behavior = "Birth"
    If EsMuestra(Explorers(ExpN).i, Explorers(ExpN).j) Then
        Explorers(ExpN).LSi = Explorers(ExpN).i
        Explorers(ExpN).LSj = Explorers(ExpN).j
        Explorers(ExpN).LastSample = Map(Explorers(ExpN).i, Explorers(ExpN).j)
    End If

    'Se generar ponderadores al azar:
    cAvg1 = Rnd
    cMax1 = Rnd * (1 - cAvg1)
    cMin1 = Rnd * (1 - (cAvg1 + cMax1))
    cSup1 = Rnd * (1 - (cAvg1 + cMax1 + cMin1))
'    cInf1 = Rnd * (1 - (cAvg1 + cMax1 + cMin1 + cSup1))
'    cAvgs1 = (1 - (cAvg1 + cMax1 + cMin1 + cSup1 + cInf1))
    cInf1 = (1 - (cAvg1 + cMax1 + cMin1 + cSup1))
    cAvgs1 = 0


    cAvgs2 = 0
    cInf2 = Rnd
'    cAvgs2 = Rnd
'    cInf2 = Rnd * (1 - cAvgs2)
    cSup2 = Rnd * (1 - (cAvgs2 + cInf2))
    cMin2 = Rnd * (1 - (cAvgs2 + cInf2 + cSup2))
    cMax2 = Rnd * (1 - (cAvgs2 + cInf2 + cSup2 + cMin2))
    cAvg2 = 1 - (cAvgs2 + cInf2 + cSup2 + cMin2 + cMax2)

    Explorers(ExpN).cAVG = 0.5 * (cAvg1 + cAvg2)
    Explorers(ExpN).cMax = 0.5 * (cMax1 + cMax2)
    Explorers(ExpN).cMin = 0.5 * (cMin1 + cMin2)
    Explorers(ExpN).cSup = 0.5 * (cSup1 + cSup2)
    Explorers(ExpN).cInf = 0.5 * (cInf1 + cInf2)
    Explorers(ExpN).cAVGS = 0.5 * (cAvgs1 + cAvgs2)

    Explorers(ExpN).dSampValue = GetPredicted(Explorers(ExpN), Map, Explorers(ExpN).i, Explorers(ExpN)
.j) '100

    ExpRanking(ExpN) = ExpN

    'Se actualiza la informacion de los exploradores en pantalla:
    Worksheets("Mapa").Range("AM" & Rowi) = Explorers(ExpN).IDExp
    Worksheets("Mapa").Range("AN" & Rowi) = Explorers(ExpN).i
    Worksheets("Mapa").Range("AO" & Rowi) = Explorers(ExpN).j
    Worksheets("Mapa").Range("AP" & Rowi) = Explorers(ExpN).dSampValue
    Worksheets("Mapa").Range("AQ" & Rowi) = Explorers(ExpN).cAVG
```

```
    Worksheets("Mapa").Range("AR" & Rowi) = Explorers(ExpN).cAVGS
    Worksheets("Mapa").Range("AS" & Rowi) = Explorers(ExpN).cMin
    Worksheets("Mapa").Range("AT" & Rowi) = Explorers(ExpN).cMax
    Worksheets("Mapa").Range("AU" & Rowi) = Explorers(ExpN).cInf
    Worksheets("Mapa").Range("AV" & Rowi) = Explorers(ExpN).cSup
    Worksheets("Mapa").Range("AW" & Rowi) = Explorers(ExpN).Caste
    Worksheets("Mapa").Range("AX" & Rowi) = Explorers(ExpN).Behavior

    With Explorers(ExpN)

        If ShowExplorers Then
            bi = RowMin + .i - 1
            bj = ColMin + .j - 1
            Worksheets("Mapa").Cells(bi, bj).Borders.Weight = 3
            If .Caste = "Best" Then Worksheets("Mapa").Cells(bi, bj).Borders.Color = ColorBest
            If .Caste = "Worst" Then Worksheets("Mapa").Cells(bi, bj).Borders.Color = ColorWorst
            If .Caste = "Tracker" Then Worksheets("Mapa").Cells(bi, bj).Borders.Color = ColorTracker
            If .Caste = "Random" Then Worksheets("Mapa").Cells(bi, bj).Borders.Color = ColorRandom
        End If

    End With

    'Se registra el que inicialmente mejor predice cada muestra:
    i = Explorers(ExpN).i
    j = Explorers(ExpN).j
    If Blocks(i, j).IsSample Then
        If Blocks(i, j).BestPredID = -1 Then
            Blocks(i, j).BestPredID = ExpN
        Else
            If Explorers(ExpN).dSampValue < Explorers(Blocks(i, j).BestPredID).dSampValue Then
                Blocks(i, j).BestPredID = ExpN
            End If
        End If
    End If

Next Rowi

DoEvents

'SE ACTUALIZAN LOS BLOQUES:
Iterationi = 1
While Iterationi <= Iterations

    Worksheets("Mapa").Range("AT" & 2) = Iterationi

    'Desvanecimiento:
    If dFade > 0 Then
        For bi = RowMin To RowMax
            For bj = ColMin To ColMax

                i = bi - RowMin + 1
                j = bj - ColMin + 1

                'Los bloques con Muestas no se modifican
                i = Worksheets("Mapa").Range("A" & bi)
                j = Worksheets("Mapa").Range(Split(Cells(, bj).Address, "$")(1) & "3")
                Leyij = Map(i, j)

                If Leyij <> "" And Not Blocks(i, j).IsSample Then
                    Map(i, j) = Max(Map(i, j) - dFade, 0) 'No hay que actualiza blocks aqui tambien?
                    Blocks(i, j).Value = Map(i, j) 'Esta bien esto?
                    If Map(i, j) = 0 Then Map(i, j) = ""
                End If

            Next bj
        Next bi

        'Se actualiza información de bloques en pantalla:
        Worksheets("Mapa").Range("B" & RowMin & ":Z" & RowMax).Value = Map

    End If

    'LOS EXPLORADORES EVALUAN LOS BLOQUES:
```

```
    For ExpN = 1 To NExplorers

        If Explorers(ExpN).Behavior = "Birth" Then Explorers(ExpN).Behavior = "Exploring"

        i = Explorers(ExpN).i
        j = Explorers(ExpN).j

        If Map(i, j) = "" Then
            Leyij = 0
        Else
            Leyij = Map(i, j)
        End If

        AvgBlocks = 0
        MaxNeighbour = -1
        MinNeighbour = -1
        Supremo = 1000
        Infimo = -1
        AvgSamples = 0
        NAdjSamples = 0
        NNeighbours = 0

        Dim bvkl As Variant 'valor del bloque en la posición (i+k, j + l)

        For gk = -1 To 1
            For gl = -1 To 1

                'If (gk <> 0 Or gl <> 0) Then 'Si esto esta descomentado solo busca los vecinos, no la
 casilla actual

                    If (i + gk < iMin Or i + gk > iMax) Or _
                       (j + gl < jMin Or j + gl > jMax) Then
                        bvkl = 0
                    Else

                        bvkl = Map(i + gk, j + gl)
                        If bvkl = "" Then
                            bvkl = 0
                        End If

                        If MaxNeighbour = -1 Then MaxNeighbour = bvkl
                        If MinNeighbour = -1 Then MinNeighbour = bvkl

                        NNeighbours = NNeighbours + 1

                        'Maximo y Minimo
                        If bvkl > MaxNeighbour Then
                            MaxNeighbour = bvkl
                        End If
                        If bvkl < MinNeighbour Then
                            MinNeighbour = bvkl
                        End If


                        'Infimo (mayor de las cotas inferiores) y Supremo (menor de las cotas superior
es)
                        'Infimo y Supremo definen la direccion de menor varianza?

                        If bvkl > Leyij And bvkl < Supremo Then
                            Supremo = bvkl
                        End If
                        If bvkl < Leyij And bvkl > Infimo Then
                            Infimo = bvkl
                        End If

                        If EsMuestra(i + gk, j + gl) Then
                            AvgSamples = AvgSamples + bvkl
                            NAdjSamples = NAdjSamples + 1
                        End If

                        AvgBlocks = AvgBlocks + bvkl
```

```
                    End If

                'End If

            Next gl
        Next gk

        'AvgBlocks = AvgBlocks / NNeighbours 'Promedio de las casillas adyacentes [NO BORRAR]
        AvgBlocks = (AvgBlocks + Leyij) / (NNeighbours + 1) 'Promedio de actual y adyacentes [NO BORRA
R]
        If NAdjSamples > 0 Then AvgSamples = AvgSamples / NAdjSamples 'Promedio de las muestras adyace
ntes
        If (Infimo > Leyij Or Infimo = -1) Then Infimo = Leyij
        If (Supremo < Leyij Or Supremo = 1000) Then Supremo = Leyij

        With Explorers(ExpN)

            Dim SqrdDistSrc As Integer
            Dim SqrdDistLS As Integer
            Dim fLeyij As Double

            fLeyij = 0 '.cAVG '0.5

            If .Caste = "Tracker" Then

                If Not (Blocks(.i, .j).IsSample) Then

                    SqrdDistSrc = GetSqrdDistance(.i, .j, .i0, .j0)
                    SqrdDistLS = GetSqrdDistance(.i, .j, .LSi, .LSj)

                    If .LSi <> .i0 Or .LSj <> .j0 Then
                        .Behavior = "Returning"
                    Else
                        .Behavior = "Exploring"
                    End If

                    If .Behavior = "Returning" Then

                        'Promedio ponderado por inverso de la distancia a muestra de origen y de desti
no
                        .Predicted = fLeyij * Leyij + (1 - fLeyij) * ((Blocks(.i0, .j0).Value * 1 / Sq
rdDistSrc) + _
                                                    (Blocks(.LSi, .LSj).Value * 1 / SqrdDistLS)) _
                                                    / (1 / SqrdDistSrc + 1 / SqrdDistLS)
                    Else
                        .Predicted = .cAVG * AvgBlocks + .cAVGS * AvgSamples + .cInf * Infimo + .cMax
* MaxNeighbour + _
                                .cMin * MinNeighbour + .cSup * Supremo
                    End If

                ElseIf Blocks(i, j).IsSample Then

                    'Si vuelve a la muestra de origen después de haber visitado otra muestra
                    If (.i = .i0 And .j = .j0) Then 'And .LSi <> .i0 And .LSj <> .j0) Then

                        '.Behavior = "Returned"

'                        .i0 = .LSi
'                        .j0 = .LSj
                        '.LastSample = Blocks(.i, .j).Value

                        .Behavior = "Exploring"

                    Else

                        .Behavior = "Returning"

                    End If

                    .LSi = .i
                    .LSj = .j
                    .LastSample = Blocks(i, j).Value
```

```
                End If

            Else

                'sumconsts = .cAVG + .cAVGS + .cInf + .cMax + .cMin + .cSup
                .Predicted = .cAVG * AvgBlocks + .cAVGS * AvgSamples + .cInf * Infimo + .cMax * MaxNei
ghbour + _
                             .cMin * MinNeighbour + .cSup * Supremo
            End If
        End With

        'Se actualizan los bloques con los valores predichos. Los bloques con Muestras no se modifican
        Dim IsTracker As Boolean
        IsTracker = Explorers(ExpN).Caste = "Tracker"
        If Not Blocks(i, j).IsSample Then 'And Not IsTracker Then

            If Explorers(ExpN).Predicted <> 0 Then
                Map(i, j) = Explorers(ExpN).Predicted
                If Map(i, j) = 0 Then
                    Map(i, j) = ""
                End If
            End If

            'Caping:
            If Map(i, j) < Caping Then Map(i, j) = ""

        End If

        Leyij = Map(i, j)
        If Leyij = "" Then Leyij = 0

        If Blocks(i, j).IsSample Then
            'ExpN actualiza la última muestra visitada
            Explorers(ExpN).LSi = i
            Explorers(ExpN).LSj = j
            Explorers(ExpN).LastSample = Leyij


            'Se calcula diferencia entre valor de la muestra y valor predicho por el explorador
            Explorers(ExpN).dSampValue = Abs(Explorers(ExpN).Predicted - Leyij)

            'Se registra el ID del explorador que mejor predice la muestra:
            Dim bpId As Integer
            bpId = Blocks(i, j).BestPredID

            If bpId = -1 Then

                Blocks(i, j).BestPredID = ExpN

            ElseIf Explorers(ExpN).dSampValue < Explorers(bpId).dSampValue Then

                Blocks(i, j).BestPredID = ExpN 'Se registra que ExpN es el que mejor predice el valor
de la muestra

            ElseIf ExpN <> bpId Then

                'ExpN es influenciado por BestPredID:
                'If Explorers(bpId).Caste <> "Tracker" And Explorers(ExpN).Caste <> "Tracker" Then
                If True Then
                    Explorers(ExpN).cAVG = 0.5 * (Explorers(ExpN).cAVG + Explorers(bpId).cAVG)
                    Explorers(ExpN).cAVGS = 0.5 * (Explorers(ExpN).cAVGS + Explorers(bpId).cAVGS)
                    Explorers(ExpN).cInf = 0.5 * (Explorers(ExpN).cInf + Explorers(bpId).cInf)
                    Explorers(ExpN).cMax = 0.5 * (Explorers(ExpN).cMax + Explorers(bpId).cMax)
                    Explorers(ExpN).cMin = 0.5 * (Explorers(ExpN).cMin + Explorers(bpId).cMin)
                    Explorers(ExpN).cSup = 0.5 * (Explorers(ExpN).cSup + Explorers(bpId).cSup)
                End If

                If Explorers(bpId).Caste = "Tracker" And Explorers(ExpN).Caste = "Tracker" Then
                    Explorers(ExpN).LSi = Explorers(bpId).LSi
                    Explorers(ExpN).LSj = Explorers(bpId).LSj
                    Explorers(ExpN).LastSample = Explorers(bpId).LastSample
                End If
```

```
                With Explorers(ExpN)

                    If .Caste = "Tracker" Then

                        If Not (Blocks(.i, .j).IsSample) Then

                             SqrdDistSrc = GetSqrdDistance(.i, .j, .i0, .j0)
                             SqrdDistLS = GetSqrdDistance(.i, .j, .LSi, .LSj)

                             If .LSi <> .i0 Or .LSj <> .j0 Then
                                .Behavior = "Returning"
                             Else
                                .Behavior = "Exploring"
                             End If

                             If .Behavior = "Returning" Then
                                'Promedio ponderado por inverso de la distancia a muestra de origen y
de destino

                                .Predicted = fLeyij * Leyij + (1 - fLeyij) * ((Blocks(.i0, .j0).Value
* 1 / SqrdDistSrc) + _
                                                      (Blocks(.LSi, .LSj).Value * 1 / SqrdDistLS)) _
                                                      / (1 / SqrdDistSrc + 1 / SqrdDistLS)
                             Else
                                .Predicted = .cAVG * AvgBlocks + .cAVGS * AvgSamples + .cInf * Infimo
+ .cMax * MaxNeighbour + _
                                    .cMin * MinNeighbour + .cSup * Supremo
                             End If

                        ElseIf Blocks(i, j).IsSample Then

                            'Si vuelve a la muestra de origen después de haber visitado otra muestra
                            If (.i = .i0 And .j = .j0) Then ' And .LSi <> .i0 And .LSj <> .j0) Then

'                                .i0 = .LSi
'                                .j0 = .LSj

                                .Behavior = "Exploring"

                            Else

                                .Behavior = "Returning"

                            End If

                            .LSi = .i
                            .LSj = .j
                            .LastSample = Blocks(i, j).Value
                            '.LastSample = -1

                        End If

                    Else

                        .Predicted = .cAVG * AvgBlocks + .cAVGS * AvgSamples + .cInf * Infimo + .cMax
* MaxNeighbour + _
                                    .cMin * MinNeighbour + .cSup * Supremo

                    End If
                    .dSampValue = Abs(.Predicted - Leyij)

                End With

            End If

    '         Worksheets("Mapa").Range("AM" & (ExpN + RowMin - 1)) = Explorers(ExpN).IDExp
    '         Worksheets("Mapa").Range("AN" & (ExpN + RowMin - 1)) = Explorers(ExpN).i
    '         Worksheets("Mapa").Range("AO" & (ExpN + RowMin - 1)) = Explorers(ExpN).j
            Worksheets("Mapa").Range("AP" & (ExpN + RowMin - 1)) = Explorers(ExpN).dSampValue
            Worksheets("Mapa").Range("AQ" & (ExpN + RowMin - 1)) = Explorers(ExpN).cAVG
            Worksheets("Mapa").Range("AR" & (ExpN + RowMin - 1)) = Explorers(ExpN).cAVGS
            Worksheets("Mapa").Range("AS" & (ExpN + RowMin - 1)) = Explorers(ExpN).cMin
            Worksheets("Mapa").Range("AT" & (ExpN + RowMin - 1)) = Explorers(ExpN).cMax
```

```
            Worksheets("Mapa").Range("AU" & (ExpN + RowMin - 1)) = Explorers(ExpN).cInf
            Worksheets("Mapa").Range("AV" & (ExpN + RowMin - 1)) = Explorers(ExpN).cSup
'            Worksheets("Mapa").Range("AW" & (ExpN + RowMin - 1)) = Explorers(ExpN).Caste
            Worksheets("Mapa").Range("AX" & (ExpN + RowMin - 1)) = Explorers(ExpN).Behavior

        End If

    Next ExpN

    Worksheets("Mapa").Range("B" & RowMin & ":Z" & RowMax).Value = Map

    Dim mutate As Boolean
    Dim CellsRanking As Variant

    mutate = True
    If mutate Then
        'Se ordenan por valor predicho (Ranking):
        SortByRank (RowMin + NExplorers - 1)
        CellsRanking = Worksheets("Mapa").Range("AM4:AW" & (RowMin + NExplorers - 1))
        Worksheets("Mapa").Range("BC4:BM" & (RowMin + NExplorers - 1)) = CellsRanking
        For ExpN = 1 To NExplorers

            ExpRanking(ExpN) = Worksheets("Mapa").Range("AM" & (ExpN + RowMin - 1))

        Next ExpN

        'Seleccion Natural
        'Los parametros mutados de los mejor rankeados sobrescriben a los últimos
        'Solo deberían sobreescribirse los que estan cerca, no cualquiera
        Dim IDExp As Integer
        Dim IdExpNew As Integer
        Dim inew As Integer, jnew As Integer
        For ExpN = 1 To Int(NExplorers / 10)


            i = Explorers(ExpN).i
            j = Explorers(ExpN).j

            IDExp = ExpRanking(ExpN)

            IsTracker = (Explorers(IDExp).Caste = "Tracker")

            If Not IsTracker And (BreedAnywhere Or Blocks(Explorers(IDExp).i, Explorers(IDExp).j).IsSa
mple) Then

                'Reproduccion:
                IdExpNew = ExpRanking(NExplorers - ExpN + 1)
                inew = Explorers(IdExpNew).i
                jnew = Explorers(IdExpNew).j

                'Los trackers no son sobreescritos:
                Dim DistReplace As Integer
                DistReplace = 2
                If (Not Explorers(IdExpNew).Caste = "Tracker" And GetSqrdDistance(i, j, inew, jnew) <=
 DistReplace) Then

                    Explorers(IdExpNew) = Explorers(IDExp)
                    Explorers(IdExpNew).IDExp = IdExpNew

                    'Se resetea la última muestra visitada
                    Explorers(IdExpNew).LastSample = -1

                    If KeepPos Then
                        Explorers(IdExpNew).i = inew
                        Explorers(IdExpNew).j = jnew
                    End If

                    'Mutacion:
                    Dim consts(1 To 6) As Double
                    Dim mut As Double
                    Dim c1 As Integer, c2 As Integer
                    Dim SumConsts As Double
```

```
                    With Explorers(IdExpNew)

                        mut = ((-1) ^ (Int(Rnd * 2))) * (Rnd * 0.02)
                        c1 = Int(Rnd * 6) + 1
                        While c1 = 2
                            c1 = Int(Rnd * 6) + 1
                        Wend
                        c2 = Int(Rnd * 6) + 1
                        While c2 = c1 Or c2 = 2
                            c2 = Int(Rnd * 6) + 1
                        Wend

                        If c1 = 1 Then
                            .cAVG = .cAVG + mut
                        ElseIf c1 = 2 Then
                            .cAVGS = .cAVGS + mut
                        ElseIf c1 = 3 Then
                            .cInf = .cInf + mut
                        ElseIf c1 = 4 Then
                            .cMax = .cMax + mut
                        ElseIf c1 = 5 Then
                            .cMin = .cMin + mut
                        Else: .cSup = .cSup + mut
                        End If

                        If c2 = 1 Then
                            .cAVG = .cAVG - mut
                        ElseIf c2 = 2 Then
                            .cAVGS = .cAVGS - mut
                        ElseIf c2 = 3 Then
                            .cInf = .cInf - mut
                        ElseIf c2 = 4 Then
                            .cMax = .cMax - mut
                        ElseIf c2 = 5 Then
                            .cMin = .cMin - mut
                        Else: .cSup = .cSup - mut
                        End If

                        SumConsts = .cAVG + .cAVGS + .cInf + .cMax + .cMin + .cSup

                    End With

                    'Se actualiza en pantalla la celda mutada
                    Dim ExpRow As Integer
                    ExpRow = NExplorers - ExpN + 1
                    Worksheets("Mapa").Range("AM" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).IDExp
                    Worksheets("Mapa").Range("AN" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).i
                    Worksheets("Mapa").Range("AO" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).j
                    Worksheets("Mapa").Range("AP" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).dSamp
Value
                    Worksheets("Mapa").Range("AQ" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).cAVG
                    Worksheets("Mapa").Range("AR" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).cAVGS
                    Worksheets("Mapa").Range("AS" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).cMin
                    Worksheets("Mapa").Range("AT" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).cMax
                    Worksheets("Mapa").Range("AU" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).cInf
                    Worksheets("Mapa").Range("AV" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).cSup
                    Worksheets("Mapa").Range("AW" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).Caste
                    Worksheets("Mapa").Range("AX" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).Behav
ior

                End If 'Not Explorers(IdExpNew).Caste = "Tracker"

            End If

        Next ExpN

        'Se ordenan por IDExp
        SortByID (RowMin + NExplorers - 1)

    End If 'Mutate

    'Se desplazan los exploradores a un bloque adyacente:
    For ExpN = 1 To NExplorers
```

```
    If ShowExplorers Then
        bi = RowMin + Explorers(ExpN).i - 1
        bj = ColMin + Explorers(ExpN).j - 1
        Worksheets("Mapa").Cells(bi, bj).Borders.Weight = 2
        Worksheets("Mapa").Cells(bi, bj).Borders.Color = RGB(128, 128, 128)
    End If

    If (Explorers(ExpN).Caste = "Best") Or (Explorers(ExpN).Caste = "Worst") Then

        'Se busca bloque adyacente mejor y peor predicho:
        i = Explorers(ExpN).i
        j = Explorers(ExpN).j

        Dim iBest As Integer, jBest As Integer
        Dim iworst As Integer, jWorst As Integer

        Dim Predicted As Double

        iBest = 0
        jBest = 0
        iworst = 0
        jWorst = 0

        Dim mapikjl As Variant
        Dim mapibjb As Variant
        Dim mapiwjw As Variant

        While (iBest = 0 And jBest = 0)
            iBest = Max(Min(i + Int(3 * Rnd) + (-1), iMax), iMin)
            jBest = Max(Min(j + Int(3 * Rnd) + (-1), jMax), jMin)
            iworst = iBest
            jWorst = jBest
        Wend

        For gk = -1 To 1
            For gl = -1 To 1

                If (gk <> 0 Or gl <> 0) Then

                    If (i + gk >= iMin And i + gk <= iMax) And _
                       (j + gl >= jMin And j + gl <= jMax) Then
                        Predicted = GetPredicted(Explorers(ExpN), Map, i + gk, j + gl)

                        mapikjl = Map(i + gk, j + gl)
                        If mapikjl = "" Then mapikjl = 0
                        mapibjb = Map(iBest, jBest)
                        If mapibjb = "" Then mapibjb = 0
                        mapiwjw = Map(iworst, jWorst)
                        If mapiwjw = "" Then mapiwjw = 0

                        If Abs(Predicted - mapikjl) < Abs(Predicted - mapibjb) Then
                            iBest = i + gk
                            jBest = j + gl
                        End If

                        If Abs(Predicted - mapikjl) > Abs(Predicted - mapiwjw) Then
                            iworst = i + gk
                            jWorst = j + gl
                        End If

                    End If

                End If

            Next gl
        Next gk

        With Explorers(ExpN)
            If .Caste = "Best" Then
                .i = iBest
                .j = jBest
```

```
                If ShowExplorers Then
                    bi = RowMin + Explorers(ExpN).i - 1
                    bj = ColMin + Explorers(ExpN).j - 1
                    Worksheets("Mapa").Cells(bi, bj).Borders.Weight = 3
                    Worksheets("Mapa").Cells(bi, bj).Borders.Color = ColorBest
                End If

            End If

            If .Caste = "Worst" Then
                .i = iworst
                .j = jWorst

                If ShowExplorers Then
                    bi = RowMin + Explorers(ExpN).i - 1
                    bj = ColMin + Explorers(ExpN).j - 1
                    Worksheets("Mapa").Cells(bi, bj).Borders.Weight = 3
                    Worksheets("Mapa").Cells(bi, bj).Borders.Color = ColorWorst
                End If

            End If

            Worksheets("Mapa").Range("AN" & (ExpN + RowMin - 1)) = .i
            Worksheets("Mapa").Range("AO" & (ExpN + RowMin - 1)) = .j
        End With
    ElseIf Explorers(ExpN).Caste = "Tracker" Then

        'Si no ha visitado una muestra distinta a su lugar de nacimiento
'         If (Explorers(ExpN).LastSample = -1 Or _
'             (Explorers(ExpN).i0 = Explorers(ExpN).LSi And Explorers(ExpN).j0 = Explorers(ExpN).LS
j)) Then

        If Explorers(ExpN).Behavior = "Exploring" Then

            'Se mueve a una posicion adyacente al azar
            gk = 0
            gl = 0

            While (gk = 0 And gl = 0)
                gk = Int(3 * Rnd) + (-1)
                gl = Int(3 * Rnd) + (-1)
            Wend

        ElseIf Explorers(ExpN).Behavior = "Returning" Then

            Dim DMink As Integer, DMinl As Integer
            Dim i0 As Integer, j0 As Integer

            'If on its way to the original sample it finds a different sample, stops returning
            If Blocks(i, j).IsSample And (i <> i0 And j <> j0) Then

               'Explorers(ExpN).Behavior = "Exploring"
               Explorers(ExpN).i0 = Explorers(ExpN).LSi
               Explorers(ExpN).j0 = Explorers(ExpN).LSj
               Explorers(ExpN).LSi = i
               Explorers(ExpN).LSj = j
               Explorers(ExpN).LastSample = Blocks(i, j).Value


            Else

                i = Explorers(ExpN).i
                j = Explorers(ExpN).j
                i0 = Explorers(ExpN).i0
                j0 = Explorers(ExpN).j0
                DMink = 0
                DMinl = 0

                'Se mueve a la casilla más cercana a (i0,j0)
                For gk = -1 To 1
                    For gl = -1 To 1

                        If (gk <> 0 Or gl <> 0) Then
```

```
                                If (i + gk >= iMin And i + gk <= iMax) And _
                                   (j + gl >= jMin And j + gl <= jMax) Then

                                    If (GetSqrdDistance(i + gk, j + gl, i0, j0) < GetSqrdDistance(i +
DMink, j + DMinl, i0, j0)) Then

                                        DMink = gk
                                        DMinl = gl

                                    End If
                                End If

                            End If

                        Next gl
                    Next gk

                    gk = DMink
                    gl = DMinl
                End If

            End If 'Explorers(ExpN).LastSample = -1 or...

            With Explorers(ExpN)
                .i = Max(Min((.i + gk), 25), 1)
                .j = Max(Min((.j + gl), 25), 1)

'                'Se supone que esto hace que olvide la ruta cuando vuelve a la muestra de origen:
'                If DMink = 1 And DMink = 1 Then
'                    .LastSample = -1
'                End If

                Worksheets("Mapa").Range("AN" & (ExpN + RowMin - 1)) = .i
                Worksheets("Mapa").Range("AO" & (ExpN + RowMin - 1)) = .j

                If ShowExplorers Then
                    bi = RowMin + Explorers(ExpN).i - 1
                    bj = ColMin + Explorers(ExpN).j - 1
                    Worksheets("Mapa").Cells(bi, bj).Borders.Weight = 3

                    If .Behavior = "Exploring" Then
                        Worksheets("Mapa").Cells(bi, bj).Borders.Color = ColorTracker
                    ElseIf .Behavior = "Returning" Then
                        Worksheets("Mapa").Cells(bi, bj).Borders.Color = ColorReturn
                    End If

                End If

            End With

        ElseIf Explorers(ExpN).Caste = "Random" Then

            gk = 0
            gl = 0
            While (gk = 0 And gl = 0)
                gk = Int(3 * Rnd) + (-1)
                gl = Int(3 * Rnd) + (-1)
            Wend

            With Explorers(ExpN)
                .i = Max(Min((.i + gk), 25), 1)
                .j = Max(Min((.j + gl), 25), 1)

                Worksheets("Mapa").Range("AN" & (ExpN + RowMin - 1)) = .i
                Worksheets("Mapa").Range("AO" & (ExpN + RowMin - 1)) = .j

                If ShowExplorers Then
                    bi = RowMin + Explorers(ExpN).i - 1
                    bj = ColMin + Explorers(ExpN).j - 1
                    Worksheets("Mapa").Cells(bi, bj).Borders.Weight = 3
                    Worksheets("Mapa").Cells(bi, bj).Borders.Color = ColorRandom
                End If
```

```
            End With
        End If

    Next ExpN

    'Worksheets("Mapa").Range("B" & RowMin & ":Z" & RowMax).Value = Map

    DoEvents

    Iterationi = Iterationi + 1

 Wend

 SortByRank (RowMin + NExplorers - 1)
 'Worksheets("Mapa").Range("B" & RowMin & ":Z" & RowMax).Value = Map

End Sub

Function Min(x As Double, y As Double) As Double

    Min = WorksheetFunction.Min(x, y)

End Function

Function Max(x As Double, y As Double) As Double

    Max = WorksheetFunction.Max(x, y)

End Function
Sub BorrarMapa()
'
' BorrarMapa Macro
'

'
    Range("B4:Z28").Select
    Selection.ClearContents
End Sub
 Function EsMuestra(i As Integer, j As Integer) As Boolean

'    Dim i As Integer
'    Dim j As Integer
    Dim Leyij As Variant

'    i = Worksheets("Mapa").Range("A" & bi)
'    j = Worksheets("Mapa").Range(Split(Cells(, bj).Address, "$")(1) & "3")
    Leyij = ""
    On Error Resume Next
        Leyij = Application.WorksheetFunction.VLookup(i & ":" & j, Worksheets("Mapa").Range("AC:AF"),
4, False)
    On Error GoTo 0
    EsMuestra = Leyij <> ""

 End Function

Function GetLey(i As Integer, j As Integer) As Variant

    'Los bloques con Muestras no se modifican
'    i = Worksheets("Mapa").Range("A" & bi)
'    j = Worksheets("Mapa").Range(Split(Cells(, bj).Address, "$")(1) & "3")
    Dim Leyij As Variant
    Leyij = ""
    On Error Resume Next
        Leyij = Application.WorksheetFunction.VLookup(i & ":" & j, Worksheets("Mapa").Range("AC:AF"),
4, False)
    On Error GoTo 0

    GetLey = Leyij

End Function

Function GetDistance(i1 As Integer, j1 As Integer, i2 As Integer, j2 As Integer) As Integer
```

```
    GetDistance = CInt(Sqr((i1 - i2) ^ 2 + (j1 - j2) ^ 2))

End Function

Function GetSqrdDistance(i1 As Integer, j1 As Integer, i2 As Integer, j2 As Integer) As Integer

    GetSqrdDistance = CInt((i1 - i2) ^ 2 + (j1 - j2) ^ 2)

End Function

Function Azimuth(x1 As Integer, y1 As Integer, x2 As Integer, y2 As Integer) As Double

    Dim Pi As Double
    Pi = 4 * Atn(1)

    If x2 > x1 And y2 > y1 Then
        Azimuth = Atn((x2 - x1) / (y2 - y1))
    ElseIf x2 > x1 And y2 < y1 Then
        Azimuth = Pi / 2 + Atn(Abs(y2 - y1) / (x2 - x1))
    ElseIf x2 < x1 And y2 < y1 Then
        Azimuth = Pi + Atn((x2 - x1) / (y2 - y1)) 'Abs no se necesita pues ambos son negativos
    ElseIf x2 < x1 And y2 > y1 Then
        Azimuth = (3 / 2) * Pi + Atn((y2 - y1) / Abs(x2 - x1))
    ElseIf x2 = x1 Then
        Azimuth = 0
    End If

End Function

Function GetPredicted(Explorer As Explorer, Map() As Variant, i As Integer, j As Integer) As Double

    Dim Leyij As Double
    Dim bvkl As Variant
    Dim AvgBlocks As Double
    Dim MaxNeighbour As Double
    Dim MinNeighbour As Double
    Dim Supremo As Double
    Dim Infimo As Double
    Dim AvgSamples As Double
    Dim NAdjSamples As Integer
    Dim NNeighbours As Integer

    i = Explorer.i
    j = Explorer.j

    If Map(i, j) = "" Then
        Leyij = 0
    Else
        Leyij = Map(i, j)
    End If

    AvgBlocks = 0
    MaxNeighbour = Leyij
    MinNeighbour = Leyij
    Supremo = 1000
    Infimo = -1
    AvgSamples = 0
    NAdjSamples = 0
    NNeighbours = 0

    Dim gk As Integer, gl As Integer

    For gk = -1 To 1
        For gl = -1 To 1

            'If (gk <> 0 Or gl <> 0) Then

                If (i + gk < iMin Or i + gk > iMax) Or _
                   (j + gl < jMin Or j + gl > jMax) Then
                    bvkl = 0
                Else
                    bvkl = Map(i + gk, j + gl)
                    If bvkl = "" Then
```

```
                        bvkl = 0
                    End If
                    NNeighbours = NNeighbours + 1
                End If

                If bvkl > MaxNeighbour Then
                    MaxNeighbour = bvkl
                End If
                If bvkl < MinNeighbour Then
                    MinNeighbour = bvkl
                End If

                If bvkl > Leyij And bvkl < Supremo Then
                    Supremo = bvkl
                End If
                If bvkl < Leyij And bvkl > Infimo Then
                    Infimo = bvkl
                End If

                If EsMuestra(i + gk, j + gl) Then
                    AvgSamples = AvgSamples + bvkl
                    NAdjSamples = NAdjSamples + 1
                End If

                AvgBlocks = AvgBlocks + bvkl
            'End If

        Next gl
    Next gk

    AvgBlocks = AvgBlocks / NNeighbours 'Promedio de las casillas adyacentes
    If NAdjSamples > 0 Then AvgSamples = AvgSamples / NAdjSamples 'Promedio de las muestras adyacentes
    If Infimo = -1 Then Infimo = 0
    If Supremo = 1000 Then Supremo = 0

    With Explorer
            'sumconsts = .cAVG + .cAVGS + .cInf + .cMax + .cMin + .cSup
            GetPredicted = .cAVG * AvgBlocks + .cAVGS * AvgSamples + .cInf * Infimo + .cMax * MaxNeigh
bour + _
                        .cMin * MinNeighbour + .cSup * Supremo
    End With

End Function



Sub SortByRank(LastRow As Integer)
'
' SortExplorers Macro
'

'
    Range("AM4:AW4").Select
    Range(Selection, Selection.End(xlDown)).Select
    ActiveWorkbook.Worksheets("Mapa").Sort.SortFields.Clear
    ActiveWorkbook.Worksheets("Mapa").Sort.SortFields.Add2 Key:=Range("AP4:AP" & LastRow _
        ), SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
    With ActiveWorkbook.Worksheets("Mapa").Sort
        .SetRange Range("AM3:AW" & LastRow)
        .Header = xlYes
        .MatchCase = False
        .Orientation = xlTopToBottom
        .SortMethod = xlPinYin
        .Apply
    End With
End Sub

Sub SortByID(LastRow As Integer)
'
' SortExplorers Macro
'

'
```

```
    Range("AM4:AW4").Select
    Range(Selection, Selection.End(xlDown)).Select
    ActiveWorkbook.Worksheets("Mapa").Sort.SortFields.Clear
    ActiveWorkbook.Worksheets("Mapa").Sort.SortFields.Add2 Key:=Range("Am4:Am" & LastRow _
        ), SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
    With ActiveWorkbook.Worksheets("Mapa").Sort
        .SetRange Range("AM3:AW" & LastRow)
        .Header = xlYes
        .MatchCase = False
        .Orientation = xlTopToBottom
        .SortMethod = xlPinYin
        .Apply
    End With
End Sub
Sub Anterpolator_OLD()

 Const Nblocks = iMax * jMax
 Const NSamples = 25

 Const MaxExplorers = Nblocks
 Dim Explorers(1 To MaxExplorers) As Explorer
 Dim ExpRanking(1 To MaxExplorers) As Integer
 Dim NExplorers As Integer
 Dim ColorTracker As Variant

 Dim mMin As Integer
 Dim mMax As Integer
 Dim mN As Integer
 'Dim Muestras(iMin To iMax, jMin To jMax) As Block

 Dim RowMin As Integer
 Dim RowMax As Integer
 Dim ColMin As Integer
 Dim ColMax As Integer

 Dim bi As Integer
 Dim bj As Integer
 Dim Leyij As Variant
 Dim Leybibj As Double
 Dim RowExpIni As Integer
 Dim RowExpFin As Integer

 Dim i As Integer
 Dim j As Integer

 Dim Rowi As Integer

 Dim Map(iMin To iMax, jMin To jMax) As Variant

 Dim AvgBlocks As Double
 Dim BVij As Double
 Dim gk As Variant 'Integer
 Dim gl As Variant 'Integer

 Dim MaxNeighbour As Double
 Dim MinNeighbour As Double
 Dim Supremo As Double 'La menor de las leyes mayores
 Dim Infimo As Double 'La mayor de las leyes menores
 Dim AvgSamples As Double 'Promedio de las muestras adyacentes
 Dim NAdjSamples As Integer 'Número de muestras adyacentes
 Dim NNeighbours As Integer
 Dim ExpN As Integer
 Dim AntValueij As Double

 'Variables para generación de ponderadores:
 Dim cAvg1 As Double, cMax1 As Double, cMin1 As Double, cSup1 As Double, cInf1 As Double, cAvgs1 As Do
uble
 Dim cAvg2 As Double, cMax2 As Double, cMin2 As Double, cSup2 As Double, cInf2 As Double, cAvgs2 As Do
uble


 Dim Iterations As Integer
 Dim Iterationi As Integer
```

Módulo1 - 20

```
Dim dFade As Double
Dim Caping As Double

Dim ShowExplorers As Boolean
Dim StartOnSample As Boolean
Dim BreedAnywhere As Boolean
Dim KeepPos As Boolean

mMin = 4
mMax = 28

RowMin = 4
RowMax = 28

ColMin = 2
ColMax = 26

RowExpIni = RowMin

Iterations = 5000
dFade = 0.05
Caping = 0
NExplorers = 10 '200 'Int((iMax * jMax) / 2)

BreedAnywhere = False 'True hace que se extingan todos excepto los best
KeepPos = True 'Explorador modificado mantiene su posicion
StartOnSample = False 'True: Exploradores siempre nacen en un bloque con muestra
ShowExplorers = True

ColorTracker = RGB(255, 0, 255)

Randomize

'Se inicializa el mapa:
Range("B" & RowMin & ":Z" & RowMax).Select
Selection.ClearContents
Selection.Font.Color = RGB(128, 128, 128)
Selection.Font.Bold = False
Selection.Borders.Color = RGB(128, 128, 128)
Selection.Borders.Weight = 2
Range("A1").Select

'Se genera arreglo de coordenadas y se añaden muestras:
Dim BlkCoords(1 To Nblocks) As Block
Dim Blocks(iMin To iMax, jMin To jMax) As Block
Dim n As Integer

n = 1
For i = iMin To iMax
   For j = jMin To jMax

       Blocks(i, j).i = i
       Blocks(i, j).j = j

       Map(i, j) = GetLey(i, j)
       Blocks(i, j).BestPredID = -1
       Blocks(i, j).IsSample = (Map(i, j) <> "")

       If Map(i, j) <> "" Then
           Blocks(i, j).Value = Map(i, j)
           bi = i + (RowMin - 1)
           bj = j + (ColMin - 1)
           Worksheets("Mapa").Cells(bi, bj).Font.Color = RGB(255, 255, 255)
       Else
           Blocks(i, j).Value = 0
       End If

       BlkCoords(n) = Blocks(i, j)

       n = n + 1

   Next j
```

```
 Next i
 'Se randomiza el arreglo de coordenadas
 Dim nRand As Integer
 For n = 1 To iMax * jMax

    Dim blkAux As Block
    nRand = Int(Rnd * (iMax * jMax - (n - 1))) + n
    blkAux = BlkCoords(n)
    BlkCoords(n) = BlkCoords(nRand)
    BlkCoords(nRand) = blkAux

 Next n

 DoEvents

 'Se generan los exploradores
 Dim fBest As Double, fWorst As Double, fRandom As Double
 Dim RndExp As Double

 RowExpFin = RowExpIni + NExplorers - 1
 'Range("AM" & RowMin & ":AW" & (RowMin + NExplorers - 1)).Select
 Range("AM" & RowMin & ":AW" & (RowMin + MaxExplorers - 1)).Select
 Selection.ClearContents

 Range("BC" & RowMin & ":BM" & (RowMin + MaxExplorers - 1)).Select
 Selection.ClearContents

 fBest = 0 / 4
 fWorst = 0 / 4
 fRandom = 0 / 4
 'fTracker = 1


 For Rowi = RowMin To (RowMin + NExplorers - 1)

    ExpN = Rowi - RowMin + 1
    Worksheets("Mapa").Range("AT" & 2) = ExpN

    RndExp = Rnd
    If RndExp <= fBest Then
        Explorers(ExpN).Caste = "Best"
    ElseIf RndExp <= fWorst Then
        Explorers(ExpN).Caste = "Worst"
    ElseIf RndExp <= fRandom Then
        Explorers(ExpN).Caste = "Random"
    Else
        Explorers(ExpN).Caste = "Tracker"
    End If

    'Tracker: siempre nace en una muestra, y sale a explorar. Cuando encuentra otra muestra, se devuel
ve a su lugar de origen
    'Marcando cada casilla con un valor que es proporcional a la distancia a la muestra de origen y la
 última muestra visitada.
    'Al llegar a su origen, se repite el ciclo.
    'En otras palabras, el comportamiento de una obrera en Ant Farm Simulator

    Explorers(ExpN).IDExp = ExpN

    Explorers(ExpN).i = Int(iMax * Rnd) + iMin
    Explorers(ExpN).j = Int(jMax * Rnd) + jMin

    If StartOnSample Or Explorers(ExpN).Caste = "Tracker" Then
        While Not EsMuestra(Explorers(ExpN).i, Explorers(ExpN).j)
            Explorers(ExpN).i = Int(iMax * Rnd) + iMin
            Explorers(ExpN).j = Int(jMax * Rnd) + jMin
        Wend
    End If

    'Posicion inicial de la muestra
    Explorers(ExpN).i0 = Explorers(ExpN).i
    Explorers(ExpN).j0 = Explorers(ExpN).j

    'Se inicializa ultima muestra visitada:
```

```
    Explorers(ExpN).LastSample = -1
    If EsMuestra(Explorers(ExpN).i, Explorers(ExpN).j) Then
        Explorers(ExpN).LSi = i
        Explorers(ExpN).LSj = j
        Explorers(ExpN).LastSample = Map(Explorers(ExpN).i, Explorers(ExpN).j)
    End If

    'Se generar ponderadores al azar:
    cAvg1 = Rnd
    cMax1 = Rnd * (1 - cAvg1)
    cMin1 = Rnd * (1 - (cAvg1 + cMax1))
    cSup1 = Rnd * (1 - (cAvg1 + cMax1 + cMin1))
    cInf1 = (1 - (cAvg1 + cMax1 + cMin1 + cSup1))
    cAvgs1 = 0

    cAvgs2 = 0
    cInf2 = Rnd
    cSup2 = Rnd * (1 - (cAvgs2 + cInf2))
    cMin2 = Rnd * (1 - (cAvgs2 + cInf2 + cSup2))
    cMax2 = Rnd * (1 - (cAvgs2 + cInf2 + cSup2 + cMin2))
    cAvg2 = 1 - (cAvgs2 + cInf2 + cSup2 + cMin2 + cMax2)

    Explorers(ExpN).cAVG = 0.5 * (cAvg1 + cAvg2)
    Explorers(ExpN).cMax = 0.5 * (cMax1 + cMax2)
    Explorers(ExpN).cMin = 0.5 * (cMin1 + cMin2)
    Explorers(ExpN).cSup = 0.5 * (cSup1 + cSup2)
    Explorers(ExpN).cInf = 0.5 * (cInf1 + cInf2)
    Explorers(ExpN).cAVGS = 0.5 * (cAvgs1 + cAvgs2)

    Explorers(ExpN).dSampValue = 100

    ExpRanking(ExpN) = ExpN

    Worksheets("Mapa").Range("AM" & Rowi) = Explorers(ExpN).IDExp
    Worksheets("Mapa").Range("AN" & Rowi) = Explorers(ExpN).i
    Worksheets("Mapa").Range("AO" & Rowi) = Explorers(ExpN).j
    Worksheets("Mapa").Range("AP" & Rowi) = Explorers(ExpN).dSampValue
    Worksheets("Mapa").Range("AQ" & Rowi) = Explorers(ExpN).cAVG
    Worksheets("Mapa").Range("AR" & Rowi) = Explorers(ExpN).cAVGS
    Worksheets("Mapa").Range("AS" & Rowi) = Explorers(ExpN).cMin
    Worksheets("Mapa").Range("AT" & Rowi) = Explorers(ExpN).cMax
    Worksheets("Mapa").Range("AU" & Rowi) = Explorers(ExpN).cInf
    Worksheets("Mapa").Range("AV" & Rowi) = Explorers(ExpN).cSup
    Worksheets("Mapa").Range("AW" & Rowi) = Explorers(ExpN).Caste

    With Explorers(ExpN)

        If ShowExplorers Then
            bi = RowMin + .i - 1
            bj = ColMin + .j - 1
            Worksheets("Mapa").Cells(bi, bj).Borders.Weight = 3
            If .Caste = "Best" Then Worksheets("Mapa").Cells(bi, bj).Borders.Color = vbGreen
            If .Caste = "Worst" Then Worksheets("Mapa").Cells(bi, bj).Borders.Color = vbRed
            If .Caste = "Tracker" Then Worksheets("Mapa").Cells(bi, bj).Borders.Color = ColorTracker
            If .Caste = "Random" Then Worksheets("Mapa").Cells(bi, bj).Borders.Color = vbBlue
        End If

    End With

    i = Explorers(ExpN).i
    j = Explorers(ExpN).j
    If Blocks(i, j).IsSample Then
        If Blocks(i, j).BestPredID = -1 Then
            Blocks(i, j).BestPredID = ExpN
        Else
            If Explorers(ExpN).dSampValue < Explorers(Blocks(i, j).BestPredID).dSampValue Then
                Blocks(i, j).BestPredID = ExpN
            End If
        End If
    End If

Next Rowi
```

```
 DoEvents

 'SE ACTUALIZAN LOS BLOQUES:
 Iterationi = 1
 While Iterationi <= Iterations

    Worksheets("Mapa").Range("AT" & 2) = Iterationi

    'Desvanecimiento:
    If dFade > 0 Then
        For bi = RowMin To RowMax
            For bj = ColMin To ColMax

                i = bi - RowMin + 1
                j = bj - ColMin + 1

                'Los bloques con Muestas no se modifican
                i = Worksheets("Mapa").Range("A" & bi)
                j = Worksheets("Mapa").Range(Split(Cells(, bj).Address, "$")(1) & "3")
                Leyij = Map(i, j)

                If Leyij <> "" And Not Blocks(i, j).IsSample Then
                    Map(i, j) = Max(Map(i, j) - dFade, 0) 'No hay que actualiza blocks aqui tambien?
                    If Map(i, j) = 0 Then Map(i, j) = ""
                End If

            Next bj
        Next bi

        Worksheets("Mapa").Range("B" & RowMin & ":Z" & RowMax).Value = Map

    End If

    'Los Exploradores evaluan los bloques:
    For ExpN = 1 To NExplorers

        i = Explorers(ExpN).i
        j = Explorers(ExpN).j

        If Map(i, j) = "" Then
            Leyij = 0
        Else
            Leyij = Map(i, j)
        End If

        AvgBlocks = 0
        MaxNeighbour = -1
        MinNeighbour = -1
        Supremo = 1000
        Infimo = -1
        AvgSamples = 0
        NAdjSamples = 0
        NNeighbours = 0

        Dim bvkl As Variant 'valor del bloke en la posición (i+k, j + l)

        For gk = -1 To 1
            For gl = -1 To 1

                If (gk <> 0 Or gl <> 0) Then

                    If (i + gk < iMin Or i + gk > iMax) Or _
                       (j + gl < jMin Or j + gl > jMax) Then
                        bvkl = 0
                    Else

                        bvkl = Map(i + gk, j + gl)
                        If bvkl = "" Then
                            bvkl = 0
                        End If

                        If MaxNeighbour = -1 Then MaxNeighbour = bvkl
                        If MinNeighbour = -1 Then MinNeighbour = bvkl
```

```
                        NNeighbours = NNeighbours + 1

                        If bvkl > MaxNeighbour Then
                            MaxNeighbour = bvkl
                        End If
                        If bvkl < MinNeighbour Then
                            MinNeighbour = bvkl
                        End If

                        If bvkl > Leyij And bvkl < Supremo Then
                            Supremo = bvkl
                        End If
                        If bvkl < Leyij And bvkl > Infimo Then
                            Infimo = bvkl
                        End If

                        If EsMuestra(i + gk, j + gl) Then
                            AvgSamples = AvgSamples + bvkl
                            NAdjSamples = NAdjSamples + 1
                        End If

                        AvgBlocks = AvgBlocks + bvkl

                    End If

                End If

            Next gl
        Next gk

        AvgBlocks = AvgBlocks / NNeighbours 'Promedio de las casillas adyacentes
        If NAdjSamples > 0 Then AvgSamples = AvgSamples / NAdjSamples 'Promedio de las muestras adyace
ntes
        If (Infimo > Leyij Or Infimo = -1) Then Infimo = Leyij
        If (Supremo < Leyij Or Supremo = 1000) Then Supremo = Leyij

        With Explorers(ExpN)

            If .Caste = "Tracker" Then


                If .LastSample <> -1 And Not (Blocks(.i, .j).IsSample) Then

                    '.Predicted = 0.5 * (.LastSample + Leyij)
                    'Promedio ponderado por distancia a muestra de origen y de destino
                    .Predicted = ((Blocks(.i0, .j0).Value * GetDistance(.i, .j, .i0, .j0)) + _
                                  (Blocks(.LSi, .LSj).Value * GetDistance(.i, .j, .LSi, .LSj))
) _
                                  / (GetDistance(.i, .j, .i0, .j0) + GetDistance(.i, .j, .LSi,
 .LSj))
'                    Else
'                        .Predicted = Leyij
                End If

            Else

                'sumconsts = .cAVG + .cAVGS + .cInf + .cMax + .cMin + .cSup
                .Predicted = .cAVG * AvgBlocks + .cAVGS * AvgSamples + .cInf * Infimo + .cMax * MaxNei
ghbour + _
                             .cMin * MinNeighbour + .cSup * Supremo
            End If
        End With

        'Se actualizan los bloques con los valores predichos. Los bloques con Muestras no se modifican
        Dim IsTracker As Boolean
        IsTracker = Explorers(ExpN).Caste = "Tracker"
        If Not Blocks(i, j).IsSample Then 'And Not IsTracker Then

            If Explorers(ExpN).Predicted <> 0 Then
                Map(i, j) = Explorers(ExpN).Predicted
                If Map(i, j) = 0 Then
                    Map(i, j) = ""
```

```
            End If
        End If

        'Caping:
        If Map(i, j) < Caping Then Map(i, j) = ""

    End If

    Leyij = Map(i, j)
    If Leyij = "" Then Leyij = 0

    If Blocks(i, j).IsSample Then
        'ExpN actualiza la última muestra visitada
        Explorers(ExpN).LSi = i
        Explorers(ExpN).LSj = j
        Explorers(ExpN).LastSample = Leyij


        'Se calcula diferencia entre valor de la muestra y valor predicho por el explorador
        Explorers(ExpN).dSampValue = Abs(Explorers(ExpN).Predicted - Leyij)

        Dim bpId As Integer

        bpId = Blocks(i, j).BestPredID

        If bpId = -1 Then

            Blocks(i, j).BestPredID = ExpN

        ElseIf Explorers(ExpN).dSampValue < Explorers(bpId).dSampValue Then

            Blocks(i, j).BestPredID = ExpN 'Se registra que ExpN es el que mejor predice el valor
de la muestra

        ElseIf ExpN <> bpId Then

            'ExpN es influenciado por BestPredID:
            If Explorers(bpId).Caste <> "Tracker" And Explorers(ExpN).Caste <> "Tracker" Then
                Explorers(ExpN).cAVG = 0.5 * (Explorers(ExpN).cAVG + Explorers(bpId).cAVG)
                Explorers(ExpN).cAVGS = 0.5 * (Explorers(ExpN).cAVGS + Explorers(bpId).cAVGS)
                Explorers(ExpN).cInf = 0.5 * (Explorers(ExpN).cInf + Explorers(bpId).cInf)
                Explorers(ExpN).cMax = 0.5 * (Explorers(ExpN).cMax + Explorers(bpId).cMax)
                Explorers(ExpN).cMin = 0.5 * (Explorers(ExpN).cMin + Explorers(bpId).cMin)
                Explorers(ExpN).cSup = 0.5 * (Explorers(ExpN).cSup + Explorers(bpId).cSup)
            End If

            If Explorers(bpId).Caste = "Tracker" And Explorers(ExpN).Caste = "Tracker" Then
                Explorers(ExpN).LSi = Explorers(bpId).LSi
                Explorers(ExpN).LSj = Explorers(bpId).LSj
                Explorers(ExpN).LastSample = Explorers(bpId).LastSample
            End If

            With Explorers(ExpN)

                If .Caste = "Tracker" Then

                    If .LastSample <> -1 And Not (Blocks(.i, .j).IsSample) Then
                        '.Predicted = 0.5 * (.LastSample + Leyij)
                        .Predicted = ((Blocks(.i0, .j0).Value * GetDistance(.i, .j, .i0, .j0)) + _
                                      (Blocks(.LSi, .LSj).Value * GetDistance(.i, .j, .LSi, .LSj))
) _
                                      / (GetDistance(.i, .j, .i0, .j0) + GetDistance(.i, .j, .LSi,
 .LSj))
                    End If

                Else

                    .Predicted = .cAVG * AvgBlocks + .cAVGS * AvgSamples + .cInf * Infimo + .cMax
* MaxNeighbour + _
                                 .cMin * MinNeighbour + .cSup * Supremo

                End If
                .dSampValue = Abs(.Predicted - Leyij)
```

```
                End With

            End If

    '         Worksheets("Mapa").Range("AM" & (ExpN + RowMin - 1)) = Explorers(ExpN).IDExp
    '         Worksheets("Mapa").Range("AN" & (ExpN + RowMin - 1)) = Explorers(ExpN).i
    '         Worksheets("Mapa").Range("AO" & (ExpN + RowMin - 1)) = Explorers(ExpN).j
            Worksheets("Mapa").Range("AP" & (ExpN + RowMin - 1)) = Explorers(ExpN).dSampValue
            Worksheets("Mapa").Range("AQ" & (ExpN + RowMin - 1)) = Explorers(ExpN).cAVG
            Worksheets("Mapa").Range("AR" & (ExpN + RowMin - 1)) = Explorers(ExpN).cAVGS
            Worksheets("Mapa").Range("AS" & (ExpN + RowMin - 1)) = Explorers(ExpN).cMin
            Worksheets("Mapa").Range("AT" & (ExpN + RowMin - 1)) = Explorers(ExpN).cMax
            Worksheets("Mapa").Range("AU" & (ExpN + RowMin - 1)) = Explorers(ExpN).cInf
            Worksheets("Mapa").Range("AV" & (ExpN + RowMin - 1)) = Explorers(ExpN).cSup

        End If

    Next ExpN

    Worksheets("Mapa").Range("B" & RowMin & ":Z" & RowMax).Value = Map

    Dim mutate As Boolean
    Dim CellsRanking As Variant

    mutate = True
    If mutate Then
        'Se ordenan por valor predicho (Ranking):
        SortByRank (RowMin + NExplorers - 1)
        CellsRanking = Worksheets("Mapa").Range("AM4:AW" & (RowMin + NExplorers - 1))
        Worksheets("Mapa").Range("BC4:BM" & (RowMin + NExplorers - 1)) = CellsRanking
        For ExpN = 1 To NExplorers

            ExpRanking(ExpN) = Worksheets("Mapa").Range("AM" & (ExpN + RowMin - 1))

        Next ExpN

        'Seleccion Natural
        'Los parametros mutados de los mejor rankeados sobrescriben a los últimos
        Dim IDExp As Integer
        Dim IdExpNew As Integer
        Dim inew As Integer, jnew As Integer
        For ExpN = 1 To Int(NExplorers / 10)


            IDExp = ExpRanking(ExpN)

            IsTracker = (Explorers(IDExp).Caste = "Tracker")

            If Not IsTracker And (BreedAnywhere Or Blocks(Explorers(IDExp).i, Explorers(IDExp).j).IsSa
mple) Then

                'Reproduccion:
                IdExpNew = ExpRanking(NExplorers - ExpN + 1)
                inew = Explorers(IdExpNew).i
                jnew = Explorers(IdExpNew).j
                Explorers(IdExpNew) = Explorers(IDExp)
                Explorers(IdExpNew).IDExp = IdExpNew

                'Se resetea la última muestra visitada
                Explorers(IdExpNew).LastSample = -1

                If KeepPos Then
                    Explorers(IdExpNew).i = inew
                    Explorers(IdExpNew).j = jnew
                End If

                'Mutacion:
                Dim consts(1 To 6) As Double
                Dim mut As Double
                Dim c1 As Integer, c2 As Integer
                Dim SumConsts As Double
```

```
            With Explorers(IdExpNew)

                mut = ((-1) ^ (Int(Rnd * 2))) * (Rnd * 0.02)
                c1 = Int(Rnd * 6) + 1
                While c1 = 2
                    c1 = Int(Rnd * 6) + 1
                Wend
                c2 = Int(Rnd * 6) + 1
                While c2 = c1 Or c2 = 2
                    c2 = Int(Rnd * 6) + 1
                Wend

                If c1 = 1 Then
                    .cAVG = .cAVG + mut
                ElseIf c1 = 2 Then
                    .cAVGS = .cAVGS + mut
                ElseIf c1 = 3 Then
                    .cInf = .cInf + mut
                ElseIf c1 = 4 Then
                    .cMax = .cMax + mut
                ElseIf c1 = 5 Then
                    .cMin = .cMin + mut
                Else: .cSup = .cSup + mut
                End If

                If c2 = 1 Then
                    .cAVG = .cAVG - mut
                ElseIf c2 = 2 Then
                    .cAVGS = .cAVGS - mut
                ElseIf c2 = 3 Then
                    .cInf = .cInf - mut
                ElseIf c2 = 4 Then
                    .cMax = .cMax - mut
                ElseIf c2 = 5 Then
                    .cMin = .cMin - mut
                Else: .cSup = .cSup - mut
                End If

                SumConsts = .cAVG + .cAVGS + .cInf + .cMax + .cMin + .cSup

            End With

            'Se actualiza en pantalla la celda mutada
            Dim ExpRow As Integer
            ExpRow = NExplorers - ExpN + 1
            Worksheets("Mapa").Range("AM" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).IDExp
            Worksheets("Mapa").Range("AN" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).i
            Worksheets("Mapa").Range("AO" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).j
            Worksheets("Mapa").Range("AP" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).dSampValu
e
            Worksheets("Mapa").Range("AQ" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).cAVG
            Worksheets("Mapa").Range("AR" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).cAVGS
            Worksheets("Mapa").Range("AS" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).cMin
            Worksheets("Mapa").Range("AT" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).cMax
            Worksheets("Mapa").Range("AU" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).cInf
            Worksheets("Mapa").Range("AV" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).cSup
            Worksheets("Mapa").Range("AW" & (ExpRow + RowMin - 1)) = Explorers(IdExpNew).Caste


        End If

    Next ExpN

    'Se ordenan por IDExp
    SortByID (RowMin + NExplorers - 1)

End If 'Mutate

'Se desplazan los exploradores a un bloque adyacente:
For ExpN = 1 To NExplorers

    If ShowExplorers Then
        bi = RowMin + Explorers(ExpN).i - 1
```

```
        bj = ColMin + Explorers(ExpN).j - 1
        Worksheets("Mapa").Cells(bi, bj).Borders.Weight = 2
        Worksheets("Mapa").Cells(bi, bj).Borders.Color = RGB(128, 128, 128)
    End If

    If (Explorers(ExpN).Caste = "Best") Or (Explorers(ExpN).Caste = "Worst") Then

        'Se busca bloque adyacente mejor y peor predicho:
        i = Explorers(ExpN).i
        j = Explorers(ExpN).j

        Dim iBest As Integer, jBest As Integer
        Dim iworst As Integer, jWorst As Integer

        Dim Predicted As Double

        iBest = 0
        jBest = 0
        iworst = 0
        jWorst = 0

        Dim mapikjl As Variant
        Dim mapibjb As Variant
        Dim mapiwjw As Variant

        While (iBest = 0 And jBest = 0)
            iBest = Max(Min(i + Int(3 * Rnd) + (-1), iMax), iMin)
            jBest = Max(Min(j + Int(3 * Rnd) + (-1), jMax), jMin)
            iworst = iBest
            jWorst = jBest
        Wend

        For gk = -1 To 1
            For gl = -1 To 1

                If (gk <> 0 Or gl <> 0) Then

                    If (i + gk >= iMin And i + gk <= iMax) And _
                       (j + gl >= jMin And j + gl <= jMax) Then
                        Predicted = GetPredicted(Explorers(ExpN), Map, i + gk, j + gl)

                        mapikjl = Map(i + gk, j + gl)
                        If mapikjl = "" Then mapikjl = 0
                        mapibjb = Map(iBest, jBest)
                        If mapibjb = "" Then mapibjb = 0
                        mapiwjw = Map(iworst, jWorst)
                        If mapiwjw = "" Then mapiwjw = 0

                        If Abs(Predicted - mapikjl) < Abs(Predicted - mapibjb) Then
                            iBest = i + gk
                            jBest = j + gl
                        End If

                        If Abs(Predicted - mapikjl) > Abs(Predicted - mapiwjw) Then
                            iworst = i + gk
                            jWorst = j + gl
                        End If

                    End If

                End If

            Next gl
        Next gk

        With Explorers(ExpN)
            If .Caste = "Best" Then
                .i = iBest
                .j = jBest

                If ShowExplorers Then
                    bi = RowMin + Explorers(ExpN).i - 1
                    bj = ColMin + Explorers(ExpN).j - 1
```

```
                    Worksheets("Mapa").Cells(bi, bj).Borders.Weight = 3
                    Worksheets("Mapa").Cells(bi, bj).Borders.Color = vbGreen
                End If

            End If

            If .Caste = "Worst" Then
                .i = iworst
                .j = jWorst

                If ShowExplorers Then
                    bi = RowMin + Explorers(ExpN).i - 1
                    bj = ColMin + Explorers(ExpN).j - 1
                    Worksheets("Mapa").Cells(bi, bj).Borders.Weight = 3
                    Worksheets("Mapa").Cells(bi, bj).Borders.Color = vbRed
                End If

            End If

            Worksheets("Mapa").Range("AN" & (ExpN + RowMin - 1)) = .i
            Worksheets("Mapa").Range("AO" & (ExpN + RowMin - 1)) = .j
        End With
    ElseIf Explorers(ExpN).Caste = "Tracker" Then

        Dim iNSup As Integer, jNSup As Integer
        Dim iNInf As Integer, jNInf As Integer
        Dim iNMin As Integer, jNMin As Integer
        Dim iNMax As Integer, jNMax As Integer
        Dim iNSamp As Integer, jNSamp As Integer
        Dim iNRand As Integer, jNRand As Integer

        Dim Leykl As Variant
        Dim LeyMax As Variant
        Dim LeyMin As Variant
        Dim LeySup As Variant
        Dim LeyInf As Variant
        Dim LeySamp As Variant

        i = Explorers(ExpN).i
        j = Explorers(ExpN).j

        Leyij = Map(i, j)
        If Leyij = "" Then Leyij = 0

        iNMin = Explorers(ExpN).i
        jNMin = Explorers(ExpN).j
        iNMax = Explorers(ExpN).i
        jNMax = Explorers(ExpN).j
        iNSamp = Explorers(ExpN).i
        jNSamp = Explorers(ExpN).j
        iNRand = -1
        jNRand = -1

        LeySup = 1000
        LeyInf = -1
        LeySamp = 0

        If (Explorers(ExpN).LastSample = -1 Or _
            (Explorers(ExpN).i0 = Explorers(ExpN).LSi And Explorers(ExpN).j0 = Explorers(ExpN).LSj
)) Then
            'Se mueve a una posicion adyacente al azar
            gk = 0
            gl = 0
            While (gk = 0 And gl = 0)
                gk = Int(3 * Rnd) + (-1)
                gl = Int(3 * Rnd) + (-1)
            Wend

            With Explorers(ExpN)
                iNRand = Max(Min((.i + gk), 25), 1)
                jNRand = Max(Min((.j + gl), 25), 1)
                iNSamp = iNRand
                jNSamp = jNRand
```

```
            End With
        Else

            '<Hay que recorrerlos al azar>
            Dim colk(1 To 3) As Integer
            Dim coll(1 To 3) As Integer

            colk(1) = -1
            colk(2) = 0
            colk(3) = 1

            coll(1) = -1
            coll(2) = 0
            coll(3) = 1


            Dim p As Integer
            Dim q As Integer
            Dim Aux As Integer

            For p = 1 To 3

                q = Int(3 * Rnd) + (1)
                Aux = colk(p)
                colk(p) = colk(q)
                colk(q) = Aux

                q = Int(3 * Rnd) + (1)
                Aux = coll(p)
                coll(p) = coll(q)
                coll(q) = Aux

            Next p


'               For gk = -1 To 1
'                   For gl = -1 To 1
            For Each gk In colk
                For Each gl In coll

                    If (gk <> 0 Or gl <> 0) Then

                        If (i + gk >= iMin And i + gk <= iMax) And _
                           (j + gl >= jMin And j + gl <= jMax) Then

                            Leykl = Map(i + gk, j + gl)
                            If Leykl = "" Then Leykl = 0

                            'Para debugging en una coordenada especifica
'                            If i = 12 And j = 10 Then
'                                Beep
'                            End If

                            'Valor más cercano a ultima muestra visitada
                            Dim dSample As Double

                            Dim klEsMuestra As Boolean
                            Dim klMasCercano As Boolean

                            dSample = (10 * dFade)

                            klEsMuestra = EsMuestra(i + gk, j + gl)
                            'klEsMuestra = True

                            klMasCercano = (Abs(Leykl - Explorers(ExpN).LastSample) <= Abs(LeySamp
 - Explorers(ExpN).LastSample) And _
                                           (Abs(Leykl - Leyij) > dSample) And _
                                           (Abs(Leykl - Explorers(ExpN).LastSample) > dSample)) An
d _
                                            Explorers(ExpN).LastSample <> -1

                            If klEsMuestra Or klMasCercano Then
```

```
                                    'Random para que en caso de empate siempre se elija la ultima casi
lla
                                    Randomize

                                    If (LeySamp <> Leykl) Or (LeySamp = Leykl And Rnd < 0.5) Then
                                        LeySamp = Leykl
                                        iNSamp = i + gk
                                        jNSamp = j + gl

                                        'If klEsMuestra Then Explorers(ExpN).LastSample = LeySamp

                                    End If

                                End If

                                'Maximo
                                LeyMax = Map(iNMax, jNMax)
                                If LeyMax = "" Then LeyMax = 0
                                If Leykl >= LeyMax Then
                                    iNMax = i + gk
                                    jNMax = j + gl
                                    LeyMax = Leykl
                                End If

                                'Minimo
                                LeyMin = Map(iNMin, jNMin)
                                If LeyMin = "" Then LeyMin = 0
                                If Leykl <= LeyMin Then
                                    iNMin = i + gk
                                    jNMin = j + gl
                                    LeyMin = Leykl
                                End If

                                'Supremo
                                If Leykl >= Leyij And Leykl <= LeySup Then
                                    iNSup = i + gk
                                    jNSup = j + gl
                                    LeySup = Leykl
                                End If

                                'Infimo
                                If Leykl <= Leyij And Leykl >= LeyInf Then
                                    iNInf = i + gk
                                    jNInf = j + gl
                                    LeyInf = Leykl
                                End If

                            End If

                        End If

                    Next gl
                Next gk
            End If 'Explorers(ExpN).LastSample = -1

            If iNInf = 0 And jNInf = 0 Then
                iNInf = iNMin
                jNInf = jNMin
                LeyInf = LeyMin
            End If

            If iNSup = 0 And jNSup = 0 Then
                iNSup = iNMax
                jNSup = jNMax
                LeySup = LeyMax
            End If

            If iNSamp = 0 And jNSamp = 0 Then
                Beep
            End If

            'Tracker Alternativa 2:
```

```
            Explorers(ExpN).i = iNSamp
            Explorers(ExpN).j = jNSamp
            If EsMuestra(iNSamp, jNSamp) Then
                Explorers(ExpN).LSi = iNSamp
                Explorers(ExpN).LSj = jNSamp
                Explorers(ExpN).LastSample = Map(iNSamp, jNSamp)
            End If

'            If Not EsMuestra(iNSamp, jNSamp) Then
'                If (Explorers(ExpN).LastSample <> -1) Then
'                    Map(iNSamp, jNSamp) = 0.5 * (Explorers(ExpN).LastSample + LeySamp)
'                End If
'            End If


            'Tracker Alternativa 1:
'            If Leyij >= LeyMax Then 'Leyij es Maximo local
'                Explorers(ExpN).i = iNInf
'                Explorers(ExpN).j = jNInf
'                If Not EsMuestra(iNInf, jNInf) Then
'                    Map(iNInf, jNInf) = 0.5 * (Leyij + LeyInf) 'Max(Leyij - dFade, 0)
'                    If Map(iNInf, jNInf) = 0 Then Map(iNInf, jNInf) = ""
'                End If
'            ElseIf Leyij <= LeyMin Then 'Leyij es Minimo local
'                Explorers(ExpN).i = iNSup
'                Explorers(ExpN).j = jNSup
'                If Not EsMuestra(iNSup, jNSup) Then
'                    Map(iNSup, jNSup) = 0.5 * (Leyij + LeySup) 'Max(Leyij + dFade, 72.4)
'                End If
'            Else
'
'                If Abs(Leyij - LeySup) < Abs(Leyij - LeyInf) Then
'
'                    Explorers(ExpN).i = iNInf
'                    Explorers(ExpN).j = jNInf
'                    If Not EsMuestra(iNInf, jNInf) Then
'                        Map(iNInf, jNInf) = Leyij '0.5 * (Leyij + LeyInf) 'Max(Leyij - dFade, 0)
'                        If Map(iNInf, jNInf) = 0 Then Map(iNInf, jNInf) = ""
'                    End If
'                Else
'
'                    Explorers(ExpN).i = iNSup
'                    Explorers(ExpN).j = jNSup
'                    If Not EsMuestra(iNSup, jNSup) Then
'                        Map(iNSup, jNSup) = Leyij '0.5 * (Leyij + LeySup) 'Max(Leyij + dFade, 72.4)
'                    End If
'
'                End If
'
'            End If


            If ShowExplorers Then
                bi = RowMin + Explorers(ExpN).i - 1
                bj = ColMin + Explorers(ExpN).j - 1
                Worksheets("Mapa").Cells(bi, bj).Borders.Weight = 3
                Worksheets("Mapa").Cells(bi, bj).Borders.Color = ColorTracker
            End If

        ElseIf Explorers(ExpN).Caste = "Random" Then

            gk = 0
            gl = 0
            While (gk = 0 And gl = 0)
                gk = Int(3 * Rnd) + (-1)
                gl = Int(3 * Rnd) + (-1)
            Wend

            With Explorers(ExpN)
                .i = Max(Min((.i + gk), 25), 1)
                .j = Max(Min((.j + gl), 25), 1)

    '            Worksheets("Mapa").Range("AM" & (ExpN + RowMin - 1)) = .IDExp
```

```
                Worksheets("Mapa").Range("AN" & (ExpN + RowMin - 1)) = .i
                Worksheets("Mapa").Range("AO" & (ExpN + RowMin - 1)) = .j
    '            Worksheets("Mapa").Range("AP" & (ExpN + RowMin - 1)) = .dSampValue

                If ShowExplorers Then
                    bi = RowMin + Explorers(ExpN).i - 1
                    bj = ColMin + Explorers(ExpN).j - 1
                    Worksheets("Mapa").Cells(bi, bj).Borders.Weight = 3
                    Worksheets("Mapa").Cells(bi, bj).Borders.Color = vbBlue
                End If

            End With
        End If

    Next ExpN

    'Worksheets("Mapa").Range("B" & RowMin & ":Z" & RowMax).Value = Map

    DoEvents

    Iterationi = Iterationi + 1

 Wend

 SortByRank (RowMin + NExplorers - 1)
 'Worksheets("Mapa").Range("B" & RowMin & ":Z" & RowMax).Value = Map

End Sub
```