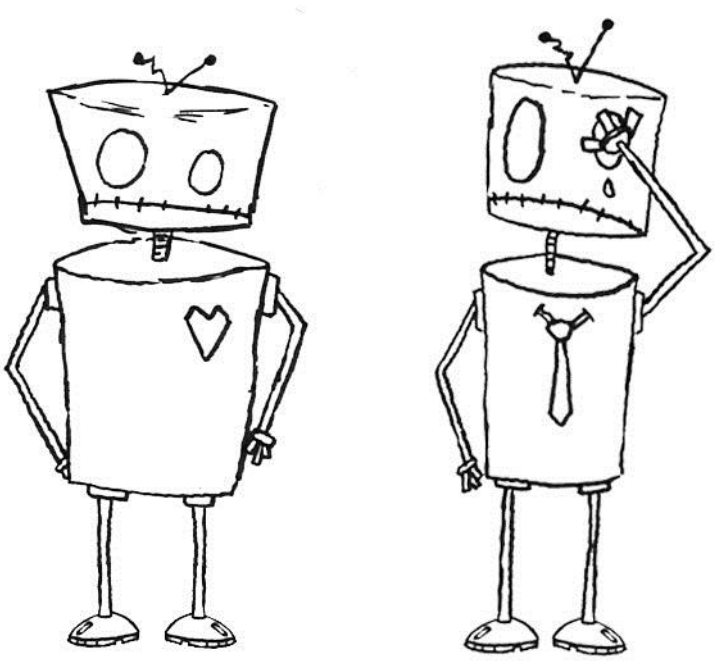
Projet

Max Halford

Michael Gisquet



Structures

Nous avons commencé par changer les structures. Premièrement il faut créer une population de robot. Nous définissons la taille plus tard (elle sera de 42).

```
Structure Population
        Dim taille As Integer
        Dim robots As Robot()
End Structure
Public pop As Population
```

Ensuite nous avons modifiés la structure robot en ajoutant la variable capacité, qui est la batterie du robot, ce qui est équivalent à la profondeur maximale des nœuds du robot:

```
Structure Robot
        Dim line As Integer
        Dim column As Integer
        Dim orientation As Direction
        Dim treeAsString As String
        Dim rootNode As Node
        Dim capacite As Integer
        Dim fitness As Double
End Structure
```

Enfin, la structure node reçoit une profondeur.

```
Structure Node
        Dim type As NodeType
        Dim successors As Node()
        Dim profondeur As Integer
End Structure
```

De plus, nous avons implantés les paramètres suivants :

```
'precisons le nombre de mouvements maximal que chaque robot peut faire
Public tempsImparti As Integer = 60
'precisons le nombre initial de mouvements pour chaque enchainement
Public mouvementsEnchainement As Integer = 6
```

Générer un robot aléatoirement

```
Function genererRobot() As Robot
        Dim newRobot As Robot
        newRobot.fitness = 0
        newRobot.capacite = mouvementsEnchainement
        newRobot.line = 16
        newRobot.column = 16
        newRobot.rootNode = New Node
        'Orientation aleatoire
        Dim p As Double = Rnd()
        If p >= 0 And p < 0.25 Then newRobot.orientation = Direction.North
        If p >= 0.25 And p < 0.5 Then newRobot.orientation = Direction.East
        If p >= 0.5 And p < 0.75 Then newRobot.orientation = Direction.South
        If p >= 0.75 And p < 1 Then newRobot.orientation = Direction.West
        'Noeud Racine aleatoire
        newRobot.rootNode.profondeur = 0
        Dim q As Double = Rnd()
        If q >= 0 And q < 0.25 Then newRobot.rootNode.type = NodeType.MoveOn
        If q >= 0.25 And q < 0.5 Then newRobot.rootNode.type = NodeType.Test
        If q >= 0.5 And q < 0.75 Then newRobot.rootNode.type = NodeType.TurnLeft
        If q >= 0.75 And q < 1 Then newRobot.rootNode.type = NodeType.TurnRight
        'Generons les noeuds successeurs
        genererSuccesseurs(newRobot.rootNode)
        'Modelisons l'arbre de decision du robot
        newRobot.treeAsString = ""
        newRobot.buildTreeAsString(newRobot.rootNode)
        Return newRobot
    End Function
```

Il n'y a rien de très savant, cela ressemble au TP/5. On génère une direction aléatoire, un nœud racine aléatoire et on génère des successeurs pour ce nœud racine. Tous les robots commenceront en (16,16), ceci n'est pas important, ça aurait pu être aléatoire, mais comme on est sur un tore tous les emplacements sont équivalent, il n'y pas de « centre » ni de « coins » sur le plateau.

Générer des successeurs à un nœud

La fonction `genererRobot()` fait appel à `genererSuccesseurs(newRobot.rootNode)`, que nous allons maintenant créer :

```
Sub genererSuccesseurs(ByRef noeud As Node)
    ReDim noeud.successors(0)
'Si le noeud sur lequel on se base est de type "Test" il faut deux successeurs,        mais pas de
successeur "Test"
    If noeud.type = NodeType.Test Then
        ReDim noeud.successors(1)
        Dim tmpNoeud1 As Node = New Node
        tmpNoeud1.profondeur = noeud.profondeur + 1
        ReDim tmpNoeud1.successors(0)
        Dim r As Double = Rnd()
        If r >= 0 And r < 1 / 3 Then tmpNoeud1.type = NodeType.MoveOn
        If r >= 1 / 3 And r < 2 / 3 Then tmpNoeud1.type = NodeType.TurnLeft
        If r >= 2 / 3 And r < 1 Then tmpNoeud1.type = NodeType.TurnRight
        Dim tmpNoeud2 As Node = New Node
        tmpNoeud2.profondeur = noeud.profondeur + 1
        tmpNoeud2.type = NodeType.MoveOn
        ReDim tmpNoeud2.successors(0)
'Si on a atteint le nombre de movements maximal, le noeud successeur est de type "EndDecision"
        If tmpNoeud1.profondeur = mouvementsEnchainement Then
        tmpNoeud1.type = NodeType.EndDecision : tmpNoeud1.successors = Nothing
        If tmpNoeud2.profondeur = mouvementsEnchainement Then
        tmpNoeud2.type = NodeType.EndDecision : tmpNoeud2.successors = Nothing
        noeud.successors(0) = tmpNoeud1
        noeud.successors(1) = tmpNoeud2
    Else : Dim tmpNoeud As Node = New Node
        tmpNoeud.profondeur = noeud.profondeur + 1
        ReDim tmpNoeud.successors(0)
        Dim s As Double = Rnd()
        If s >= 0 And s < 1 / 4 Then tmpNoeud.type = NodeType.MoveOn
        If s >= 1 / 4 And s < 2 / 4 Then tmpNoeud.type = NodeType.Test
        If s >= 2 / 4 And s < 3 / 4 Then tmpNoeud.type = NodeType.TurnLeft
        If s >= 3 / 4 And s < 1 Then tmpNoeud.type = NodeType.TurnRight
        If tmpNoeud.profondeur = mouvementsEnchainement Then
        tmpNoeud.type = NodeType.EndDecision : tmpNoeud.successors = Nothing
        noeud.successors(0) = tmpNoeud
    End If
    'Recursivite
    For i = 0 To noeud.successors.Length - 1
       If noeud.successors(i).type <> NodeType.EndDecision Then
       genererSuccesseurs(noeud.successors(i))
    Next
End Sub
```

De même, les nœuds successeurs sont générer aléatoirement, cependant il faut faire une distinction si le nœud pour lequel on génère des successeurs aléatoires est de type « test » ou pas, s'il l'est on ne va pas générer un nœud successeur de type « test », ce serait contre-productif. On ajoute les profondeurs en fonction du nœud parent.

Déplacer le robot

```
‘tourner a droite
Sub turnRight(ByRef robot As Robot)
        If robot.orientation = Direction.North Then
            robot.orientation = Direction.East
        ElseIf robot.orientation = Direction.East Then
            robot.orientation = Direction.South
        ElseIf robot.orientation = Direction.South Then
            robot.orientation = Direction.West
        ElseIf robot.orientation = Direction.West Then
            robot.orientation = Direction.North
        End If
    End Sub
‘tourner a gauche
Sub turnLeft(ByRef robot As Robot)
        If robot.orientation = Direction.North Then
            robot.orientation = Direction.West
        ElseIf robot.orientation = Direction.East Then
            robot.orientation = Direction.North
        ElseIf robot.orientation = Direction.South Then
            robot.orientation = Direction.East
        ElseIf robot.orientation = Direction.West Then
            robot.orientation = Direction.South
        End If
End Sub
‘avancer
Sub moveOn(ByRef robot As Robot, ByVal board As Board)
        Select Case robot.orientation
            Case Direction.North
                If robot.line = 0 Then
                    robot.line = Math.Sqrt(board.matrix.Length) - 1
                else : robot.line -= 1
                End If
            Case Direction.East
                If robot.column = Math.Sqrt(board.matrix.Length) - 1 Then
                    robot.column = 0
                Else : robot.column += 1
                End If
            Case Direction.South
                If robot.line = Math.Sqrt(board.matrix.Length) - 1 Then
                    robot.line = 0
                Else : robot.line += 1
                End If
            Case Direction.West
                If robot.column = 0 Then
                    robot.column = Math.Sqrt(board.matrix.Length) - 1
                Else : robot.column -= 1
                End If
        End Select
End Sub
```

Tourner à gauche ou à droite revient tout simplement à changer d’orientation. Avancer revient à augmenter ou à diminuer les coordonnées du robot. Attention, les coordonnées fonctionnent comme celles d’une matrice, et non pas d’un repère cartésien « classique » :

Nord-Ouest / (0,0)	Nord / (0,1)	Nord-Est / (0,2)
Ouest / (1,0)	Centre / (1,1)	Est / (1,2)
Sud-Ouest / (2,0)	Sud / (2,1)	Sud-Est / (2,2)

« Lire » le robot

Maintenant que nous pouvons déplacer le robot, il faut savoir quels mouvements faire, on doit donc lire les nœuds du robot choisi et agir en conséquence :

```
Sub letsgo(ByVal n As Node, ByRef robot As Robot, ByRef board As Board)
        pause(180)
        If n.type <> NodeType.EndDecision Then
            Select Case n.type
                Case NodeType.MoveOn
                    moveOn(robot, board)
                    'assignons la fitness
                    If board.matrix(robot.line, robot.column) = Square.Paint Then
             robot.fitness += 1
                    drawRobot(robot)
                    board.matrix(robot.line, robot.column) = Square.Empty
                    updateSquareColor(robot.line, robot.column, Square.Empty)
                    letsgo(n.successors(0), robot, board)
                Case NodeType.Test
                    If test(robot, board) = "paint" Then
                        letsgo(n.successors(1), robot, board)
                    Else : letsgo(n.successors(0), robot, board)
                    End If
                Case NodeType.TurnLeft
                    turnLeft(robot)
                    drawRobot(robot)
                    letsgo(n.successors(0), robot, board)
                Case NodeType.TurnRight
                    turnRight(robot)
                    drawRobot(robot)
                    letsgo(n.successors(0), robot, board)
            End Select
        End If
End Sub
```

On lit le robot tant que l'on ne tombe pas sur un nœud de type « enddecision ». Les cases de la matrice sont seulement assujetties à un changement si on avance, de même que la fitness des robots. Tourner à gauche ou à droite nécessite seulement de redessiner le robot selon la nouvelle orientation.

Initialiser la population

On initialise grâce au bouton « Initialiser » :

```
Private Sub Button_Initialiser_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button_Initialiser.Click
        ListBox_Individus.Items.Clear()
        genererPopulation()
        For i = 0 To pop.taille - 1
            For j = 0 To 5
                'Plateau
                Dim myBoard As Board = loadTrail(j)
                draw(TabPage2, myBoard)
                'Robot
                Dim myRobot As Robot = pop.robots(i)
                drawRobot(myRobot)
                If myBoard.matrix(myRobot.line, myRobot.column) = Square.Paint Then
                 myRobot.fitness += 1
                myBoard.matrix(myRobot.line, myRobot.column) = Square.Empty
                updateSquareColor(myRobot.column, myRobot.line, Square.Empty)
                'Let's go!
                Dim nbEnchainements As Integer = Math.Floor(tempsImparti / myRobot.capacite)
                For k = 1 To nbEnchainements
                    letsgoBis(myRobot.rootNode, myRobot, myBoard)
                Next
                pop.robots(i).fitness = myRobot.fitness
            Next
            pop.robots(i).fitness = Math.Round(pop.robots(i).fitness / 6, 2)
            pop.robots(i).save(i)
            ListBox_Individus.Items.Add(pop.robots(i).treeAsString & " " & pop.robots(i).fitness)
        Next

        For i = 0 To pop.taille - 1
            If pop.robots(i).fitness > Label6.Text Then Label6.Text = pop.robots(i).fitness
        Next
        For i = 0 To pop.taille - 1
            Label7.Text = Label7.Text + pop.robots(i).fitness
        Next
        Label7.Text = Label7.Text / pop.taille
        Label8.Text = pop.robots(0).fitness
        For i = 0 To pop.taille - 1
            If pop.robots(i).fitness < Label8.Text Then Label8.Text = pop.robots(i).fitness
        Next
        Label9.Text += 1
    End Sub
```

`genererPopulation()` est une procédure qui crée 42 robots avec `genererRobot()` et les place dans le tableau pop.robots().
On va mesurer la fitness moyenne des robots sur 6 pistes. Il ne faut pas oublier d'augmenter la fitness du robot si son emplacement initial est une case de type « peinture ».
`nbEnchainements` correspond au nombre de fois que le robot peut effectuer son enchainement, ici le nombre de mouvements maximal est 60. Le robot parcourt donc son arbre round(60/capacité) fois.
Le reste est seulement du calcul de maximum, minimum et moyenne.

NB. letsgoBis est pareil que letsgo mais sans les représentations graphiques, cela permet de diminuer le temps de calcul.

Simuler le parcours d'un robot

```
Private Sub Button6_Click(sender As System.Object, e As System.EventArgs) Handles Button6.Click
        'Plateau
        Dim myBoard As Board = loadTrail(ComboBox1.Text - 1)
        draw(TabPage2, myBoard)
        Dim nbPaint As Integer = 0
        For i = 0 To Math.Sqrt(myBoard.matrix.Length) - 1
            For j = 0 To Math.Sqrt(myBoard.matrix.Length) - 1
                If myBoard.matrix(i, j) = Square.Paint Then nbPaint += 1
            Next
        Next
        'Robot
        Dim myRobot As Robot = pop.robots(ComboBox2.Text - 1)
        myRobot.fitness = 0
        drawRobot(myRobot)
        If myBoard.matrix(myRobot.line, myRobot.column) = Square.Paint Then myRobot.fitness += 1
        myBoard.matrix(myRobot.line, myRobot.column) = Square.Empty
        updateSquareColor(myRobot.column, myRobot.line, Square.Empty)
        'Let's go!
        Dim nbEnchainements As Integer = Math.Floor(tempsImparti / myRobot.capacite)
        For i = 1 To nbEnchainements
            letsgo(myRobot.rootNode, myRobot, myBoard)
        Next
        Label11.Text = myRobot.fitness
        Label13.Text = Math.Round(100 * (myRobot.fitness / nbPaint), 2)
    End Sub
```

Le principe est le même qu'avec la génération de la population, sauf qu'on utilise letsgo (affichage graphique) et on simule seulement un robot. On met les fitness et la performance dans les labelboxs correspondantes.

Visualiser individu

```
Private Sub Button_VisualiserIndividu_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button_VisualiserIndividu.Click
        Dim orientation As String = ""
        If pop.robots(ListBox_Individus.SelectedIndex).orientation = 0 Then orientation = "north"
        If pop.robots(ListBox_Individus.SelectedIndex).orientation = 1 Then orientation = "west"
        If pop.robots(ListBox_Individus.SelectedIndex).orientation = 2 Then orientation = "south"
        If pop.robots(ListBox_Individus.SelectedIndex).orientation = 3 Then orientation = "east"
        MsgBox("Arbre: " & pop.robots(ListBox_Individus.SelectedIndex).treeAsString & "      Capacite: "
& pop.robots(ListBox_Individus.SelectedIndex).capacite & "      Orientation: " & orientation & "
Fitness: " & pop.robots(ListBox_Individus.SelectedIndex).fitness)
End Sub
```

Rien de très compliqué. On utilise à peu près la même chose dans tabpage2 sauf que l'utilise quel robot et quel piste afficher grâce à des comboboxs.

Croiser

```
Function croiser(ByVal sujets As Robot()) As Robot()
        Dim tableauNoeuds1 As Node() : ReDim tableauNoeuds1(0) : tableauNoeuds1(0) = sujets(0).rootNode
        Dim tableauNoeuds2 As Node() : ReDim tableauNoeuds2(0) : tableauNoeuds2(0) = sujets(1).rootNode
        treeToarray(tableauNoeuds1(0), tableauNoeuds1)
        treeToarray(tableauNoeuds2(0), tableauNoeuds2)

        Dim proba1 As Double = Math.Floor(Rnd() * tableauNoeuds1.Length)
        Dim proba2 As Double = Math.Floor(Rnd() * tableauNoeuds2.Length)

        Dim NouveauRobotUn As Robot = genererRobot() : NouveauRobotUn.rootNode = sujets(0).rootNode :
        NouveauRobotUn.treeAsString = sujets(0).treeAsString
        Dim NouveauRobotDeux As Robot = genererRobot() : NouveauRobotDeux.rootNode = sujets(1).rootNode :
        NouveauRobotDeux.treeAsString = sujets(1).treeAsString

        Dim counterUn As Integer = 1
        Dim counterDeux As Integer = 1

        changeNode(tableauNoeuds2(proba2), NouveauRobotUn.rootNode, counterUn, proba1)
        changeNode(tableauNoeuds1(proba1), NouveauRobotDeux.rootNode, counterDeux, proba2)

        Dim nouveauxRobots As Robot() : ReDim nouveauxRobots(1)
        nouveauxRobots(0) = NouveauRobotUn : nouveauxRobots(1) = NouveauRobotDeux
        nouveauxRobots(0).rootNode.profondeur = 0 : nouveauxRobots(1).rootNode.profondeur = 0
      nouveauxRobots(0).capacite =
      getMaxDepth(nouveauxRobots(0).rootNode,nouveauxRobots(0).rootNode.profondeur) :
      nouveauxRobots(1).capacite
      getMaxDepth(nouveauxRobots(1).rootNode, nouveauxRobots(1).rootNode.profondeur)
        assignDepth(nouveauxRobots(0).rootNode) : assignDepth(nouveauxRobots(1).rootNode)

        Return nouveauxRobots
    End Function
```

Une fois que les robots sont sélectionnés (même méthode que le TP/5, il faut croiser. Nous avons décidé de transformer l'arbre du robot en tableau, pour faciliter la sélection aléatoire, grâce à la méthode `treeToarray()`. Il ne faut pas oublier d'assigner les profondeurs et une capacité à chaque robot, grâce à `assignDepth()` et `getMaxDepth()`.

Ensuite, il suffit, comme dans le TP/5, de créer une nouvelle population, qui reçoit deux à deux les nouveaux robots. Cette nouvelle population devient alors la population principale, sur laquelle on peut réappliquer toutes nos méthodes.