Excel + VBA

Online-Algorithmen

Autor & Copyright: Dipl.-Ing. Harald Nahrstedt

Version: 2016 / 2019 / 2021 / 365

Erstellungsdatum: 28.04.2010

Überarbeitung: 01.12.2023

Quelle: Eigene Entwicklung

Beschreibung:

Im Bereich der klassischen kombinatorischen Optimierung unterscheiden wir zwischen Offline- und Online-Problemen. Bei Offline-Problemen liegen die Parameter und Ereignisse zur Optimierung vor und ein Algorithmus berechnet die (approximativ) optimale Lösung. Doch viele Praxisprobleme erfordern Entscheidungen, ohne das Wissen über zukünftige Ereignisse. Diese werden als Online-Probleme bezeichnet. Nachfolgend sind einige aufgezählt und anhand einer Aufzugsteuerung folgt ein einführendes Beispiel.

Anwendungs-Datei:

07-09-02_OnlineAlgorithmus.xlsm

In der Literatur finden sich einige Beispiele von Online-Problemen, wie

- das Skifahrproblem, wann Skier kaufen oder wann leasen
- das Bahncardproblem, wann eine Card kaufen bei unbekannten Reisetätigkeiten
- das Aufzugproblem, wie steuern ohne Fahraufträge zu kennen
- das Suchproblem, wo steht das geparkte Auto – links oder rechts
- das Pagingproblem, zur Speicherung in Ebenen
- und viele andere mehr …

Grundsätzlich unterscheidet man zwischen Offline-Algorithmen, die unabhängig von den Objekten und Beziehungen vorliegen und den Online-Algorithmen, die sich je nach den Objekten und Beziehungen anpassen.

1 Das Aufzugs-Problem

Ein klassisches Beispiel dazu ist das Aufzugsproblem. Es ist gut geeignet, um die Bedeutung von Offline- und Online- (Echtzeit-) Optimierung aufzuzeigen. Betrachten wir ein Szenario, wie es sich am Aufzug tagtäglich abspielt (Bild 1). Nehmen wir an, der Aufzug steht im 6-ten Stock. Im 8-ten Stock fordert A einen Transport in den 0-ten Stock (die unterste Etage). Im 2-ten Stock fordert B einen Transport in den 9-ten Stock und im 7-ten Stock fordert C einen Transport in den 2-ten Stock.

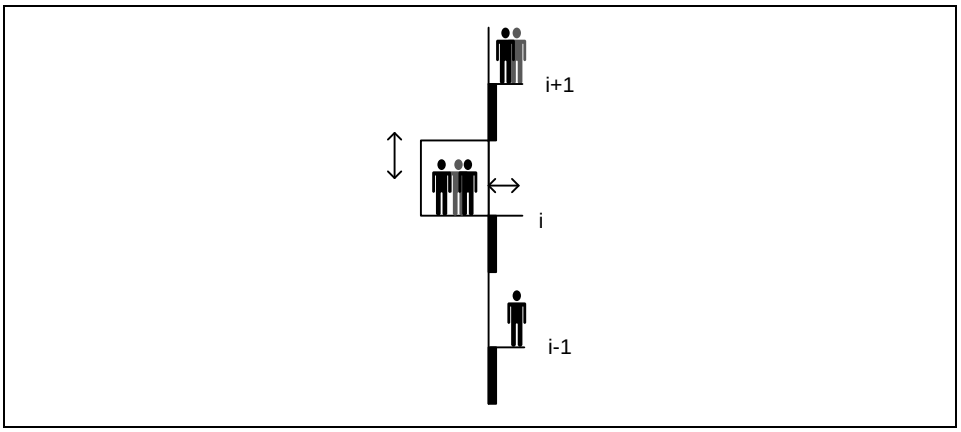

Bild 1. Aufzug-Schema

Es bleibt nicht viel Zeit für Überlegungen, denn die Aufrufenden sind schon ungeduldig und gerade als der Aufzug losfahren will, fordert D im 8-ten Stock einen Transport in den 3-ten Stock. Was ist zu tun, wenn bei der Fahrt zusätzlich zu den Personen in den Stockwerken noch andere Personen E, F, G, … hinzukommen, die wieder einen anderen Transportweg fordern. Damit sind wir mitten im Online-Geschehen des Aufzugproblems.

Nach klassischer Optimierungsmanier könnte man den Fahrplan für die kürzesten Wege der bekannten 3 Aufträge bestimmen. Das Problem ist jedoch, dass zukünftige Fahraufträge unbekannt sind und auch eine stetige Optimierung sicher nicht die kürzesten Wege liefert.

2 Offline-Algorithmus

Beginnen wir mit einem Offline-Algorithmus. Dieser ist ohne Kenntnis des tatsächlichen Geschehens nur in einer Form möglich. Alle Stockwerke werden vom Aufzug nacheinander angefahren. Dann hält der Aufzug eine gewisse Zeit für Ausstieg und Einstieg, und dann geht es weiter. In den Grenz-Etagen EMin und EMax erfolgt eine Umkehrung der Fahrtrichtung.

Für das nachfolgende Beispiel einer Probabilistischen Simulation gelten folgende Parameter:

F = Fahrtzähler

FMax = Fahrtzeit in Sekunden für ein Stockwerk / hier 5 Sekunden

H = Haltzähler

HMax = Haltezeit in Sekunden für ein Stockwerk / hier 4 Sekunden

E = Etagenzähler

EMin = unterstes Stockwerk / hier = 0

EMax = oberstes Stockwerk / hier = 9

R = Richtungsmerker / 0 = Ruhe / 1 = nach oben / 2 = nach unten

Die Wahrscheinlichkeit, mit der eine Person den Aufzug anfordert, ist für jede Etage unterschiedlich und durch Zählungen ermittelt worden. Die Angaben liegen als durchschnittliche Anzahl pro Minute vor:

0 = 4 / 1 = 1 / 2 = 0,5 / 3 = 0,2 / 4 = 0,1 / 5 = 0,5 / 6 = 0,8 / 7 = 0,1 / 8 = 0,05 / 9 = 0,06

Diese Angaben sind willkürlich. Der Betrachtungszeitraum (die Schrittweite der Simulation) beträgt 1 Sekunde. Betrachtet wird ein Zeitraum von einer Stunde. Zur Vereinfachung soll immer nur eine Person einen Auftrag auslösen und die Anzahl der Personen im Aufzug sei unbegrenzt.

Im ersten Schritt fertigen wir ein Zustands-Diagramm des Aufzugs für den Offline-Algorithmus an (Bild 2).

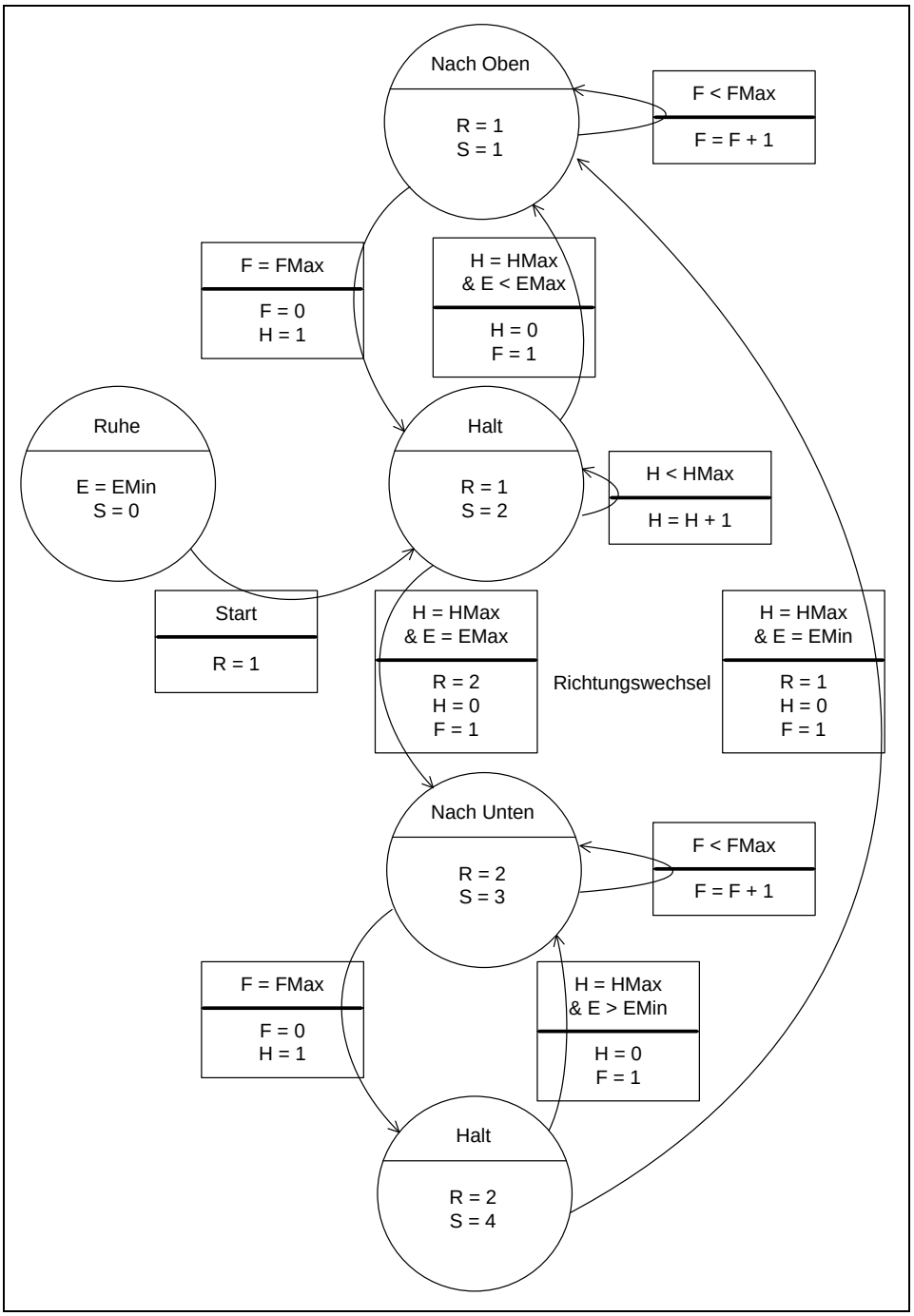

Bild 2. Zustands-Diagramm des Aufzugs für den Offline-Algorithmus

Es zeigt die möglichen Zustände des Aufzugs, wie Ruhe, Halt und Fahrt (nach oben oder unten). Die möglichen Übergänge von einem Zustand in den anderen, werden durch einen Pfeil gekennzeichnet. Sie sind mit einer Bedingung gekoppelt, die dem Pfeil zugeordnet ist. Es können auch mehrere UND- und ODER-Verknüpfungen zwischen Bedingungen vorliegen. UND-Verknüpfungen werden in einem Bedingungsfeld durch das Zeichen & dargestellt. ODER-Verknüpfungen durch mehrere Bedingungsfelder übereinander. Darunter stehen die Parameteränderungen, die mit dieser Zustandsänderung verbunden sind. Der Aufzug kann sich immer nur in einem Zustand befinden. Der Start ist im Ruhezustand in der Etage 0 vorgesehen und geht mit dem ersten Zeitschritt in den Haltezustand für eine Fahrt nach oben. Später werden wir den Start an ein Ereignis koppeln.

Die Realisierung sieht eine UserForm und Module für die unterschiedlichen Algorithmen vor (Bild 3).

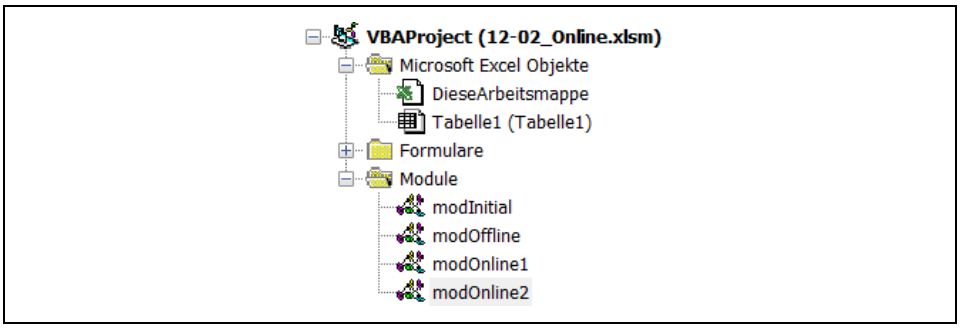

Bild 3. Objekte im Projekt-Explorer von Excel-VBA

Alle Algorithmen benutzen die gleiche Initialisierungsroutine im Modul *modInitial*.

Codeliste 1. Initialisierung

```
Option Explicit
Option Base 0

Public dWahr(9) As Double
Public dSumW As Double
Public Const MaxHalt As Integer = 4
Public Const MaxFahrt As Integer = 5

Public Function Initialisierung() As Integer
   Dim iC As Integer

   Initialisierung = 1

   dWahr(0) = 4 / 60
   dWahr(1) = 1 / 60
   dWahr(2) = 0.5 / 60
   dWahr(3) = 0.2 / 60
   dWahr(4) = 0.1 / 60
   dWahr(5) = 0.5 / 60
   dWahr(6) = 0.8 / 60
   dWahr(7) = 0.1 / 60
   dWahr(8) = 0.05 / 60
   dWahr(9) = 0.06 / 60
```

```
   dSumW = 0
   Cells.Delete
   Cells(1, 1) = "Etage"
   Cells(1, 2) = "w"
   Cells(1, 3) = "Auf"
   For iC = 0 To 9
      Cells(iC + 2, 1) = iC
      Cells(iC + 2, 2) = dWahr(iC)
      dSumW = dSumW + dWahr(iC)
   Next iC
End Function
```

Die entsprechenden Algorithmen befinden sich in weiteren Modulen.

Codeliste 2. Offline-Algorithmus im Modul modOffline

```
Option Explicit

Const FMax As Integer = 5
Const HMax As Integer = 4
Const EMin As Integer = 0
Const EMax As Integer = 9

Sub Offline()
   Load F1_Aufzug
   F1_Aufzug.Typ.Caption = "Online-Algorithmus 1"
   F1_Aufzug.Show
End Sub

Public Function Offline_AL()

'Zustandswerte
   Dim E As Integer
   Dim R As Integer
   Dim F As Integer
   Dim H As Integer
   Dim S As Integer

'Rechenwerte
   Dim x As Double
   Dim dSum As Double
   Dim Warte As Long
   Dim WMax As Integer
   Dim wz As Long
   Dim p As Long

'Zähler
   Dim iZ As Integer
   Dim iE As Integer
   Dim iW As Integer
   Dim iK As Integer
   Dim iL As Integer

   Dim iPers As Integer 'Anzahl erzeugter Personen
   Dim iStat(9, 2) As Integer

'Initialisierung
   E = 0
   R = 1
   H = 2
```

```
   S = 0
   WMax = 0
   Randomize (Timer) 'Start des Zufallsgenerators
   Initialisierung
   F1_Aufzug.AL.Clear
   F1_Aufzug.BL.Clear
   F1_Aufzug.SL.Clear
   For iZ = 0 To 9
      F1_Aufzug.SL.AddItem
      F1_Aufzug.SL.List(iZ, 0) = Right("     " & Str(iZ), 5)
   Next iZ
   F1_Aufzug.SL.AddItem


'eine Stunde Simulation
   For iZ = 1 To 3600
      F1_Aufzug.SL.List(E, 3) = ""

   'Wartezeiten
      For iK = 0 To F1_Aufzug.AL.ListCount - 1
         wz = Val(F1_Aufzug.AL.List(iK, 3)) + 1
         F1_Aufzug.AL.List(iK, 3) = Str(wz)
      Next iK

   'Erzeugung zufallsbedingter Aufträge
      For iE = 0 To 9
         x = Rnd(x)
         If x <= dWahr(iE) Then

         'Fahrtzielbestimmung
            Do
               x = Rnd(x)
               dSum = 0
               iW = -1
               Do
                  iW = iW + 1
                  dSum = dSum + dWahr(iW) / dSumW
               Loop While x > dSum
            Loop While iE = iW
            'verhindert, dass Etage und Ziel gleich sind

         'Übernahme in der Auftragsliste
            F1_Aufzug.AL.AddItem
            iK = F1_Aufzug.AL.ListCount - 1
            F1_Aufzug.AL.List(iK, 0) = Trim(Str(iZ))
            F1_Aufzug.AL.List(iK, 1) = Trim(Str(iE))
            F1_Aufzug.AL.List(iK, 2) = Trim(Str(iW))
            F1_Aufzug.AL.List(iK, 3) = "0"
            iPers = iPers + 1
            F1_Aufzug.SL.List(10, 1) = Str(iPers)
         End If
      Next iE

   'Statusanzeige
      F1_Aufzug.H = H
      F1_Aufzug.F = F
      F1_Aufzug.W = Warte
      F1_Aufzug.R = R
      F1_Aufzug.E = E
      F1_Aufzug.WMax = WMax
```

```
'Ereignisse
   If S = 0 Or S = 2 Or S = 4 Then
      'Aussteigen
      If F1_Aufzug.BL.ListCount > 0 Then
         iK = 0
         Do While (iK <= F1_Aufzug.BL.ListCount - 1)
            If Val(F1_Aufzug.BL.List(iK, 2)) = E Then
               F1_Aufzug.BL.RemoveItem (iK)
               iStat(E, 2) = iStat(E, 2) + 1
            Else
               iK = iK + 1
            End If
         Loop
      End If
      'Zusteigen, nur wenn Ziel in Fahrtrichtung
      If F1_Aufzug.AL.ListCount > 0 Then
         iK = 0
         Do While (iK <= F1_Aufzug.AL.ListCount - 1)
            If Val(F1_Aufzug.AL.List(iK, 1)) = E Then
               'Ziel liegt in Fahrtrichtung nach oben
               If Val(F1_Aufzug.AL.List(iK, 2)) > E Then
                  F1_Aufzug.BL.AddItem
                  iL = F1_Aufzug.BL.ListCount - 1
                  F1_Aufzug.BL.List(iL, 0) = _
                     F1_Aufzug.AL.List(iK, 0)
                  F1_Aufzug.BL.List(iL, 1) = _
                     F1_Aufzug.AL.List(iK, 1)
                  F1_Aufzug.BL.List(iL, 2) = _
                     F1_Aufzug.AL.List(iK, 2)
                  wz = Val(F1_Aufzug.AL.List(iK, 3))
                  Warte = Warte + wz
                  If wz > WMax Then WMax = wz
                  F1_Aufzug.AL.RemoveItem (iK)
                  iStat(E, 1) = iStat(E, 1) + 1
               Else
                  If Val(F1_Aufzug.AL.List(iK, 2)) < E Then
                     F1_Aufzug.BL.AddItem
                     iL = F1_Aufzug.BL.ListCount - 1
                     F1_Aufzug.BL.List(iL, 0) = _
                        F1_Aufzug.AL.List(iK, 0)
                     F1_Aufzug.BL.List(iL, 1) = _
                        F1_Aufzug.AL.List(iK, 1)
                     F1_Aufzug.BL.List(iL, 2) = _
                        F1_Aufzug.AL.List(iK, 2)
                     wz = Val(F1_Aufzug.AL.List(iK, 3))
                     Warte = Warte + wz
                     If wz > WMax Then WMax = wz
                     F1_Aufzug.AL.RemoveItem (iK)
                     iStat(E, 1) = iStat(E, 1) + 1
                  Else
                     iK = iK + 1
                  End If
               End If
            Else
               iK = iK + 1
            End If
         Loop
      End If
   End If
```

```
   'Status
      S = Status(R, E, F, H)

   'Statistik
      For iK = 0 To 9
         For iL = 1 To 2
            F1_Aufzug.SL.List(iK, iL) = _
               Right("     " & Str(iStat(iK, iL)), 5)
         Next iL
         If iK = E Then
            F1_Aufzug.SL.List(iK, 3) = "X"
         Else
            F1_Aufzug.SL.List(iK, 3) = " "
         End If
      Next iK

      F1_Aufzug.Repaint
   Next iZ

'Endberechnung
   F1_Aufzug.W = Warte
   F1_Aufzug.WMax = WMax
   p = Val(F1_Aufzug.SL.List(10, 1)) - F1_Aufzug.AL.ListCount
   x = Val(F1_Aufzug.W) / p
   F1_Aufzug.W = Format(x, "#0.0")
End Function

Private Function Status(R As Integer, _
   E As Integer, F As Integer, H As Integer) As Integer
   Dim AE As Boolean
   Dim BE As Boolean

   Select Case R
   Case 1 'aufwärts
      If F > 0 Then 'fahren
         If F < FMax Then
            F = F + 1
            Status = 1
         Else 'anhalten
            F = 0
            H = 1
            E = E + 1
            Status = 2
         End If
      Else 'halten
         If H < HMax Then 'halten
            H = H + 1
            Status = 2
         Else 'fahren H=HMax
            If E < EMax Then
               H = 0
               F = 1
               Status = 1
            Else
               H = 0
               F = 1
               R = 2 'Richtungsumkehr
               Status = 3
            End If
```

```
         End If
      End If
   Case 2 'abwärts
      If F > 0 Then 'fahren
         If F < FMax Then
            F = F + 1
            Status = 3
         Else 'anhalten
            F = 0
            H = 1
            E = E - 1
            Status = 4
         End If
      Else 'halten
         If H < HMax Then 'halten
            H = H + 1
            Status = 4
         Else 'fahren H=HMax
            If E > EMin Then
               H = 0
               F = 1
               Status = 3
            Else
               H = 0
               F = 1
               R = 1 'Richtungsumkehr
               Status = 1
            End If
         End If
      End If
   End Select
End Function
```

Alle Algorithmen benutzen das gleiche Dialogformular mit dem Inhalt von Codeliste 3.

Codeliste 3. Dialogformular im Formular frmAufzug

```
Option Explicit

Private Sub cStart_Click()
   Select Case Typ.Caption
   Case "Offline-Algorithmus"
      Offline_AL
   Case "Online-Algorithmus 1"
      Online_AL_1
   Case "Online-Algorithmus 2"
      Online_AL_2
   End Select
End Sub

Private Sub UserForm_Initialize()
'Auftragsliste
   With AL
      .ColumnCount = 4
      .ColumnWidths = "1cm;1cm;1cm;1cm"
      .ColumnHeads = False
   End With

'Bordliste
   With BL
```

```
        .ColumnCount = 3
        .ColumnWidths = "1cm;1cm;1cm"
        .ColumnHeads = False
    End With

'Statistikliste
    With SL
        .ColumnCount = 4
        .ColumnWidths = "1cm;1cm;1cm;1cm"
        .ColumnHeads = False
    End With
    cStart.SetFocus
End Sub
```

Das Dialogformular (Bild 4).

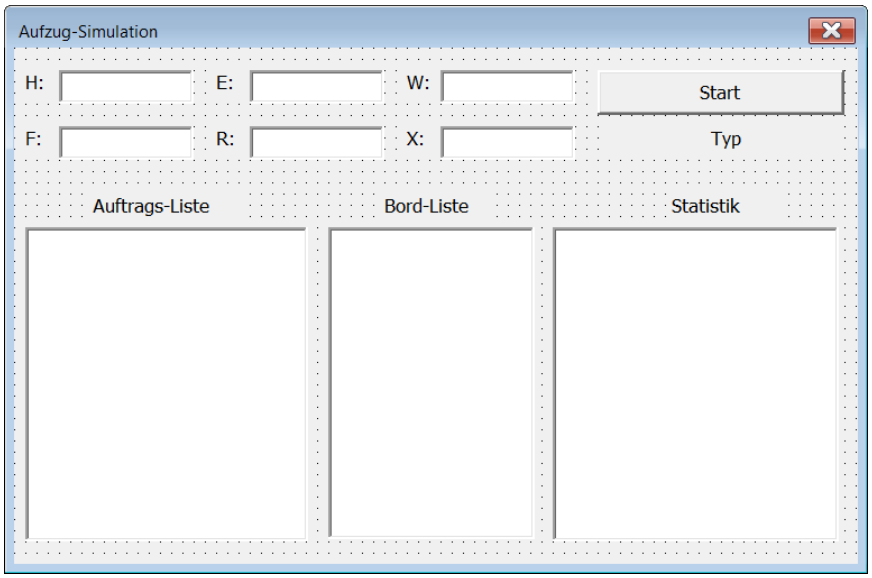

Bild 4. Dialogformular frmAufzug

Ein mögliches Ergebnis der Simulation zeigt Bild 5. Die Überprüfung der durch die Simulation erzielten Zufallswahrscheinlichkeiten ergibt natürlich Abweichungen. Also war das Simulationsintervall nicht hinreichend groß. Doch für eine einfache Betrachtung reicht es völlig aus.

In diesem Zeitraum werden 420 Personen an ihr Ziel gebracht. 1 Person steht noch vor dem Aufzug und 1 Person ist im Aufzug. Mit den angenommenen Wahrscheinlichkeiten ist der Aufzug also nicht überlastet. Doch uns interessiert die durchschnittliche Wartezeit vor dem Aufzug. Sie beträgt 61,1 Sekunden, also etwas mehr als eine Minute. Die maximale Wartezeit liegt mit 158 Sekunden bei gut 2,5 Minuten. Mit jeder Simulation ergeben sich Abweichungen, die um diese Werte schwanken.

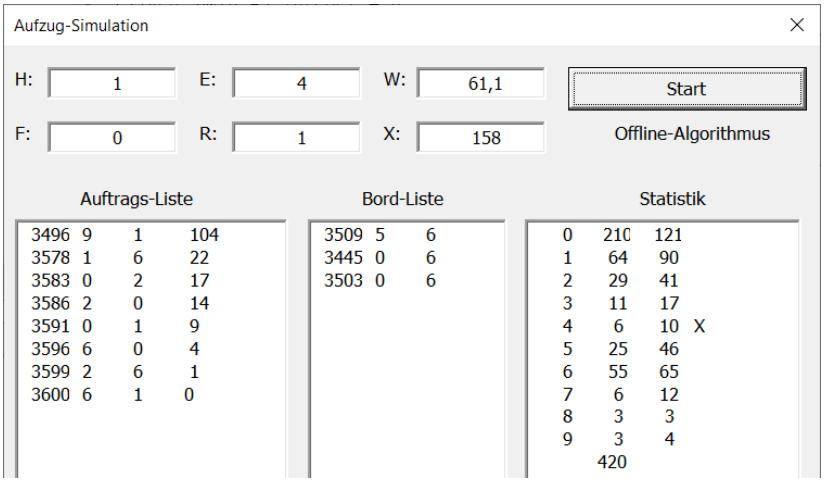

Bild 5. Ergebnis einer Simulation

3 Online-Algorithmus Nr. 1

Eine einfache erste Möglichkeit der Optimierung besteht nun darin, dass in Etagen, in denen weder eine Person aus- noch zusteigt, der Aufzug weiterfährt (Bild 6). Eine einfache Möglichkeit, auf die vorhandene Situation „online“ zu reagieren. Bild 7 zeigt das geänderte Zustands-Diagramm, bei dem die Bedingungen für die Übergänge von Halt nach Fahrt durch eine zweite ODER-verknüpfte Bedingung ergänzt wurden. Mit A(E) ist ein Ereignis definiert, dass mindestens eine Person aus der Auftragsliste in der aktuellen Etage zusteigt. Mit B(E) ist ein Ereignis definiert, dass wenigstens eine Person in der aktuellen Etage aussteigt.

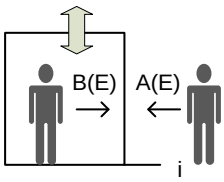

Bild 6. Aufzug-Ereignisse

Diese Ereignisse werden als Zähler-Funktionen konstruiert, da möglicherweise die Anzahl gleicher Ereignisse für spätere Online-Algorithmen eine wichtige Rolle spielen könnte.

Die Ereignisse werden bei der Ankunft auf einer Etage abgefragt. Haben beide Ereignisse den Wert Null, dann kann der Aufzug weiterfahren bzw. umkehren.

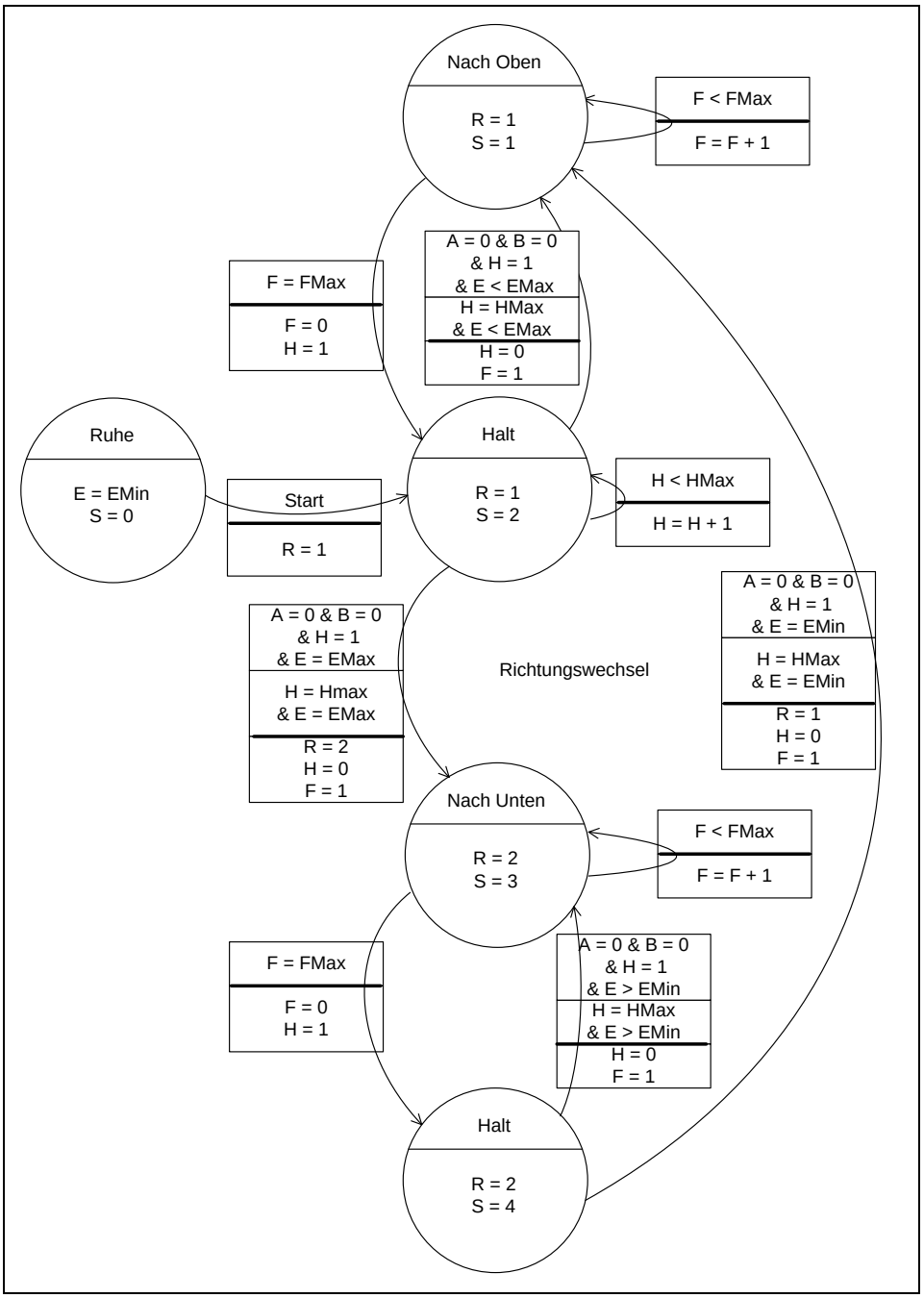

Bild 7. Zustands-Diagramm des Aufzugs für den Online-Algorithmus 1

Codeliste 4. Online-Algorithmus 1 im Modul modOnline1

```
Option Explicit

Const FMax As Integer = 5
Const HMax As Integer = 4
Const EMin As Integer = 0
Const EMax As Integer = 9

Sub Online_1()
   Load F1_Aufzug
   F1_Aufzug.Typ.Caption = "Online-Algorithmus 1"
   F1_Aufzug.Show
End Sub

Public Function Online_AL_1()

'Zustandswerte
   Dim E As Integer
   Dim R As Integer
   Dim A As Integer
   Dim B As Integer
   Dim F As Integer
   Dim H As Integer
   Dim S As Integer

'Rechenwerte
   Dim x As Double
   Dim dSum As Double
   Dim Warte As Long
   Dim WMax As Integer
   Dim wz As Long
   Dim p As Long

'Zähler
   Dim iZ As Integer
   Dim iE As Integer
   Dim iW As Integer
   Dim iK As Integer
   Dim iL As Integer

   Dim iPers As Integer 'Anzahl erzeugter Personen
   Dim iStat(9, 2) As Integer

'Initialisierung
   E = 0
   R = 1
   H = 2
   S = 0
   WMax = 0
   Randomize (Timer) 'Start des Zufallsgenerators
   Initialisierung
   F1_Aufzug.AL.Clear
   F1_Aufzug.BL.Clear
   F1_Aufzug.SL.Clear
   For iZ = 0 To 9
      F1_Aufzug.SL.AddItem
      F1_Aufzug.SL.List(iZ, 0) = Right("     " & Str(iZ), 5)
   Next iZ
   F1_Aufzug.SL.AddItem
```

```
'eine Stunde Simulation
   For iZ = 1 To 3600
      F1_Aufzug.SL.List(E, 3) = ""

   'Wartezeiten
      For iK = 0 To F1_Aufzug.AL.ListCount - 1
         wz = Val(F1_Aufzug.AL.List(iK, 3)) + 1
         F1_Aufzug.AL.List(iK, 3) = Str(wz)
      Next iK

   'Erzeugung zufallsbedingter Aufträge
      For iE = 0 To 9
         x = Rnd(x)
         If x <= dWahr(iE) Then

         'Fahrtzielbestimmung
            Do
               x = Rnd(x)
               dSum = 0
               iW = -1
               Do
                  iW = iW + 1
                  dSum = dSum + dWahr(iW) / dSumW
               Loop While x > dSum
            Loop While iE = iW 'verhindert, Etage und Ziel gleich

         'Übernahme in der Auftragsliste
            F1_Aufzug.AL.AddItem
            iK = F1_Aufzug.AL.ListCount - 1
            F1_Aufzug.AL.List(iK, 0) = Trim(Str(iZ))
            F1_Aufzug.AL.List(iK, 1) = Trim(Str(iE))
            F1_Aufzug.AL.List(iK, 2) = Trim(Str(iW))
            F1_Aufzug.AL.List(iK, 3) = "0"
            iPers = iPers + 1
            F1_Aufzug.SL.List(10, 1) = Str(iPers)
         End If
      Next iE

   'Statusanzeige
      F1_Aufzug.H = H
      F1_Aufzug.F = F
      F1_Aufzug.W = Warte
      F1_Aufzug.R = R
      F1_Aufzug.E = E
      F1_Aufzug.WMax = WMax

   'Ereignisse
      If S = 0 Or S = 2 Or S = 4 Then
         'Aussteigen
         If F1_Aufzug.BL.ListCount > 0 Then
            iK = 0
            Do While (iK <= F1_Aufzug.BL.ListCount - 1)
               If Val(F1_Aufzug.BL.List(iK, 2)) = E Then
                  F1_Aufzug.BL.RemoveItem (iK)
                  iStat(E, 2) = iStat(E, 2) + 1
               Else
                  iK = iK + 1
               End If
```

```
            Loop
         End If
         'Zusteigen, nur wenn Ziel in Fahrtrichtung
         If F1_Aufzug.AL.ListCount > 0 Then
            iK = 0
            Do While (iK <= F1_Aufzug.AL.ListCount - 1)
               If Val(F1_Aufzug.AL.List(iK, 1)) = E Then
                  'Ziel liegt in Fahrtrichtung nach oben
                  If Val(F1_Aufzug.AL.List(iK, 2)) > E Then
                     F1_Aufzug.BL.AddItem
                     iL = F1_Aufzug.BL.ListCount - 1
                     F1_Aufzug.BL.List(iL, 0) = _
                        F1_Aufzug.AL.List(iK, 0)
                     F1_Aufzug.BL.List(iL, 1) = _
                        F1_Aufzug.AL.List(iK, 1)
                     F1_Aufzug.BL.List(iL, 2) = _
                        F1_Aufzug.AL.List(iK, 2)
                     wz = Val(F1_Aufzug.AL.List(iK, 3))
                     Warte = Warte + wz
                     If wz > WMax Then WMax = wz
                     F1_Aufzug.AL.RemoveItem (iK)
                     iStat(E, 1) = iStat(E, 1) + 1
                  Else
                     If Val(F1_Aufzug.AL.List(iK, 2)) < E Then
                        F1_Aufzug.BL.AddItem
                        iL = F1_Aufzug.BL.ListCount - 1
                        F1_Aufzug.BL.List(iL, 0) = _
                           F1_Aufzug.AL.List(iK, 0)
                        F1_Aufzug.BL.List(iL, 1) = _
                           F1_Aufzug.AL.List(iK, 1)
                        F1_Aufzug.BL.List(iL, 2) = _
                           F1_Aufzug.AL.List(iK, 2)
                        wz = Val(F1_Aufzug.AL.List(iK, 3))
                        Warte = Warte + wz
                        If wz > WMax Then WMax = wz
                        F1_Aufzug.AL.RemoveItem (iK)
                        iStat(E, 1) = iStat(E, 1) + 1
                     Else
                        iK = iK + 1
                     End If
                  End If
               Else
                  iK = iK + 1
               End If
            Loop
         End If
      End If

   'Status
      S = Status(R, E, F, H)

   'Statistik
      For iK = 0 To 9
         For iL = 1 To 2
            F1_Aufzug.SL.List(iK, iL) = _
               Right("     " & Str(iStat(iK, iL)), 5)
         Next iL
         If iK = E Then
            F1_Aufzug.SL.List(iK, 3) = "X"
         Else
```

```
            F1_Aufzug.SL.List(iK, 3) = " "
         End If
      Next iK

      F1_Aufzug.Repaint
   Next iZ

'Endberechnung
   F1_Aufzug.W = Warte
   F1_Aufzug.WMax = WMax
   p = Val(F1_Aufzug.SL.List(10, 1)) - F1_Aufzug.AL.ListCount
   x = Val(F1_Aufzug.W) / p
   F1_Aufzug.W = Format(x, "#0.0")
End Function

Private Function Event_A(E As Integer) As Integer
   Dim iK As Integer

   Event_A = 0
   For iK = 0 To F1_Aufzug.AL.ListCount - 1
      If Val(F1_Aufzug.AL.List(iK, 1)) = E Then _
         Event_A = Event_A + 1
   Next iK
End Function

Private Function Event_B(E As Integer) As Integer
   Dim iK As Integer

   Event_B = 0
   For iK = 0 To F1_Aufzug.BL.ListCount - 1
      If Val(F1_Aufzug.BL.List(iK, 2)) = E Then _
         Event_B = Event_B + 1
   Next iK
End Function

Private Function Status(R As Integer, _
   E As Integer, F As Integer, H As Integer) As Integer
   Dim A As Integer
   Dim B As Integer


   Select Case R
   Case 1 'aufwärts
      If F > 0 Then 'fahren
         If F < FMax Then
            F = F + 1
            Status = 1
         Else 'anhalten
            F = 0
            H = 1
            E = E + 1
            Status = 2
            A = Event_A(E)
            B = Event_B(E)
            If A = 0 And B = 0 Then
               If E < EMax Then
                  H = 0
                  F = 1
                  Status = 1
               Else
```

```
                H = 0
                F = 1
                R = 2 'Richtungsumkehr
                Status = 3
             End If
          End If
       End If
    Else 'halten
       If H < HMax Then 'halten
          H = H + 1
          Status = 2
       Else 'fahren H=HMax
          If E < EMax Then
             H = 0
             F = 1
             Status = 1
          Else
             H = 0
             F = 1
             R = 2 'Richtungsumkehr
             Status = 3
          End If
       End If
    End If
 Case 2 'abwärts
    If F > 0 Then 'fahren
       If F < FMax Then
          F = F + 1
          Status = 3
       Else 'anhalten
          F = 0
          H = 1
          E = E - 1
          Status = 4
          A = Event_A(E)
          B = Event_B(E)
          If A = 0 And B = 0 Then
             If E > EMin Then
                H = 0
                F = 1
                Status = 3
             Else
                H = 0
                F = 1
                R = 1 'Richtungsumkehr
                Status = 1
             End If
          End If
       End If
    Else 'halten
       If H < HMax Then 'halten
          H = H + 1
          Status = 4
       Else 'fahren H=HMax
          If E > EMin Then
             H = 0
             F = 1
             Status = 3
          Else
             H = 0
```

```
                F = 1
                R = 1 'Richtungsumkehr
                Status = 1
            End If
        End If
    End If
  End Select
End Function
```

Das Ergebnis (Bild 8) ist nun schon besser. Die durchschnittliche Wartezeit liegt bei 46,6 Sekunden und die längste Wartezeit bei 133 Sekunden.

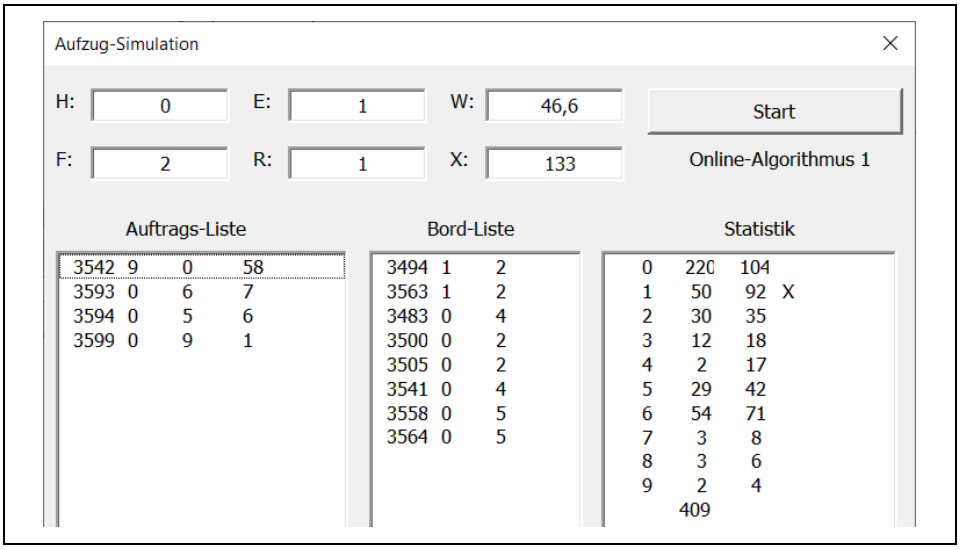

Bild 8. Ergebnisse einer Simulation mit Online-AL 1

4 Online-Algorithmus Nr. 2

In einem weiteren Schritt kann der Aufzug frühzeitig seine Richtung ändern, wenn für ihn in Fahrtrichtung kein Ereignis vorliegt (Bild 9).

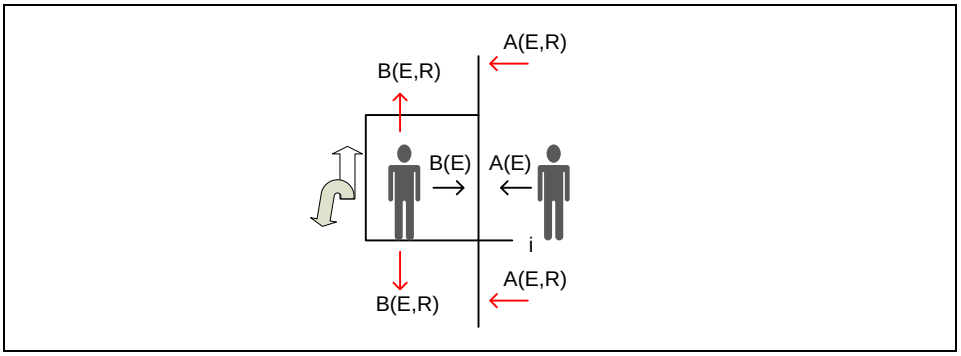

Bild 9. Aufzug-Ereignisse beim Online-AL 2

Diese Ereignisse sind sowohl von der aktuellen Etage, als auch von der Fahrtrichtung abhängig. Daher nennen wir die Ereignisse A und B in AE und BE um und führen zwei neue Ereignisse AR und BR ein. Sie geben an, ob in Fahrtrichtung noch Ereignisse zu erwarten sind. Das Zustandsdiagramm bekommt weitere Ergänzungen (Bild 10).

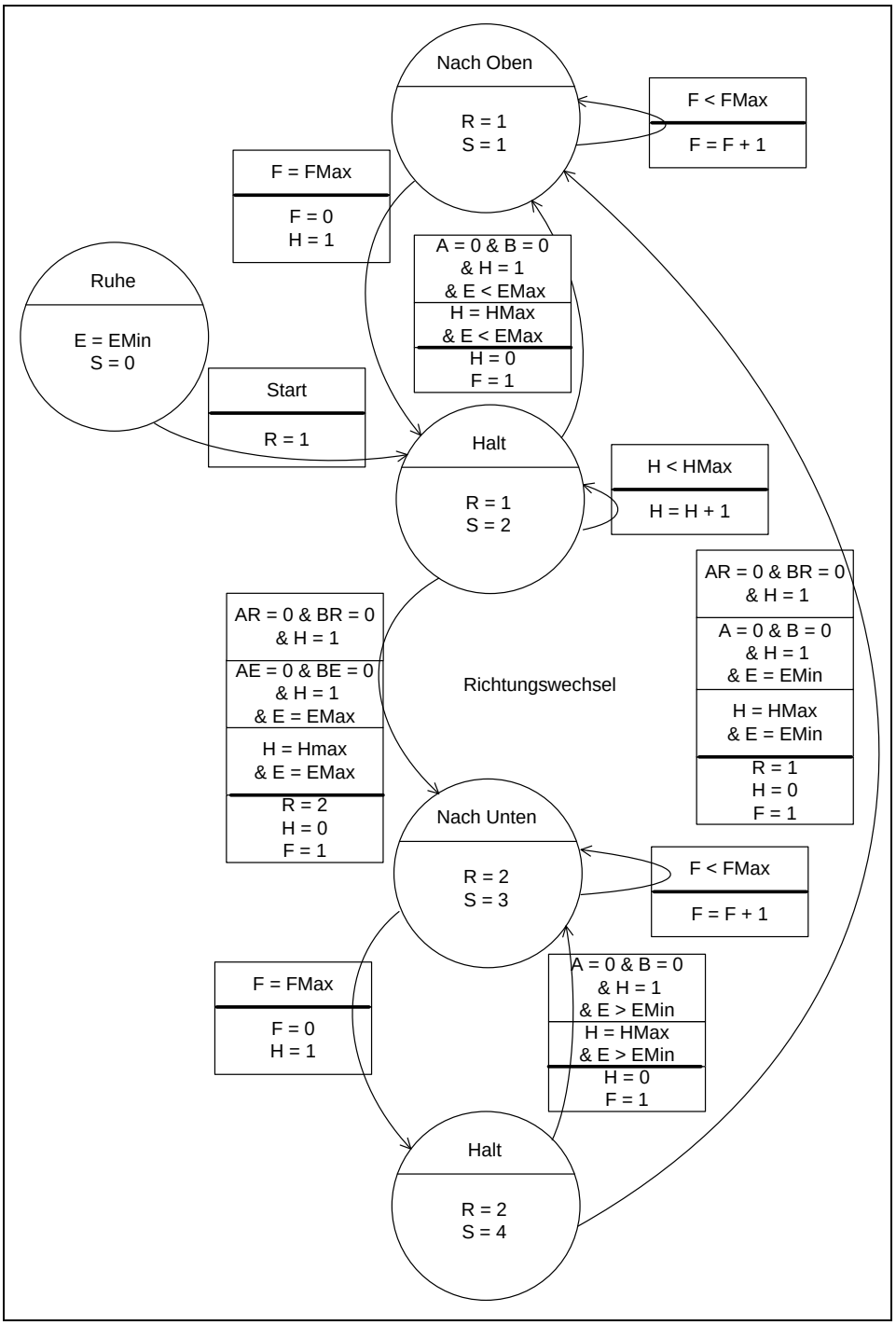

Bild 10. Zustands-Diagramm des Aufzugs für den Online-Algorithmus 2

Codeliste 5. Online-Algorithmus 2 im Modul modOnline2

```
Option Explicit

Const FMax As Integer = 5
Const HMax As Integer = 4
Const EMin As Integer = 0
Const EMax As Integer = 9


Sub Online_2()
   Load F1_Aufzug
   F1_Aufzug.Typ.Caption = "Online-Algorithmus 2"
   F1_Aufzug.Show
End Sub

Public Function Online_AL_2()

'Zustandswerte
   Dim E As Integer
   Dim R As Integer
   Dim AE As Integer
   Dim BE As Integer
   Dim F As Integer
   Dim H As Integer
   Dim S As Integer

'Rechenwerte
   Dim x As Double
   Dim dSum As Double
   Dim Warte As Long
   Dim WMax As Integer
   Dim wz As Long
   Dim p As Long

'Zähler
   Dim iZ As Integer
   Dim iE As Integer
   Dim iW As Integer
   Dim iK As Integer
   Dim iL As Integer

   Dim iPers As Integer 'Anzahl erzeugter Personen
   Dim iStat(9, 2) As Integer

'Initialisierung
   E = 0
   R = 0
   H = 2
   S = 0
   WMax = 0
   Randomize (Timer) 'Start des Zufallsgenerators
   Initialisierung
   F1_Aufzug.AL.Clear
   F1_Aufzug.BL.Clear
   F1_Aufzug.SL.Clear
   For iZ = 0 To 9
      F1_Aufzug.SL.AddItem
      F1_Aufzug.SL.List(iZ, 0) = Right("     " & Str(iZ), 5)
   Next iZ
```

```
   F1_Aufzug.SL.AddItem


'eine Stunde Simulation
   For iZ = 1 To 3600
      F1_Aufzug.SL.List(E, 3) = ""

   'Wartezeiten
      For iK = 0 To F1_Aufzug.AL.ListCount - 1
         wz = Val(F1_Aufzug.AL.List(iK, 3)) + 1
         F1_Aufzug.AL.List(iK, 3) = Str(wz)
      Next iK

   'Erzeugung zufallsbedingter Aufträge
      For iE = 0 To 9
         x = Rnd(x)
         If x <= dWahr(iE) Then

         'Fahrtzielbestimmung
            Do
               x = Rnd(x)
               dSum = 0
               iW = -1
               Do
                  iW = iW + 1
                  dSum = dSum + dWahr(iW) / dSumW
               Loop While x > dSum
            Loop While iE = iW
            'verhindert, dass Etage und Ziel gleich sind

         'Übernahme in der Auftragsliste
            F1_Aufzug.AL.AddItem
            iK = F1_Aufzug.AL.ListCount - 1
            F1_Aufzug.AL.List(iK, 0) = Trim(Str(iZ))
            F1_Aufzug.AL.List(iK, 1) = Trim(Str(iE))
            F1_Aufzug.AL.List(iK, 2) = Trim(Str(iW))
            F1_Aufzug.AL.List(iK, 3) = "0"
            iPers = iPers + 1
            F1_Aufzug.SL.List(10, 1) = Str(iPers)
         End If
      Next iE

   'Statusanzeige
      F1_Aufzug.H = H
      F1_Aufzug.F = F
      F1_Aufzug.W = Warte
      F1_Aufzug.R = R
      F1_Aufzug.E = E
      F1_Aufzug.WMax = WMax

   'Ereignisse
      If S = 2 Or S = 4 Then
         'Aussteigen
         If F1_Aufzug.BL.ListCount > 0 Then
            iK = 0
            Do While (iK <= F1_Aufzug.BL.ListCount - 1)
               If Val(F1_Aufzug.BL.List(iK, 2)) = E Then
                  F1_Aufzug.BL.RemoveItem (iK)
                  iStat(E, 2) = iStat(E, 2) + 1
               Else
```

```
                iK = iK + 1
             End If
          Loop
       End If
       'Zusteigen, nur wenn Ziel in Fahrtrichtung
       If F1_Aufzug.AL.ListCount > 0 Then
          iK = 0
          Do While (iK <= F1_Aufzug.AL.ListCount - 1)
             If Val(F1_Aufzug.AL.List(iK, 1)) = E Then
                'Ziel liegt in Fahrtrichtung nach oben
                If Val(F1_Aufzug.AL.List(iK, 2)) > E Then
                   F1_Aufzug.BL.AddItem
                   iL = F1_Aufzug.BL.ListCount - 1
                   F1_Aufzug.BL.List(iL, 0) = _
                      F1_Aufzug.AL.List(iK, 0)
                   F1_Aufzug.BL.List(iL, 1) = _
                      F1_Aufzug.AL.List(iK, 1)
                   F1_Aufzug.BL.List(iL, 2) = _
                      F1_Aufzug.AL.List(iK, 2)
                   wz = Val(F1_Aufzug.AL.List(iK, 3))
                   Warte = Warte + wz
                   If wz > WMax Then WMax = wz
                   F1_Aufzug.AL.RemoveItem (iK)
                   iStat(E, 1) = iStat(E, 1) + 1
                Else
                   If Val(F1_Aufzug.AL.List(iK, 2)) < E Then
                      F1_Aufzug.BL.AddItem
                      iL = F1_Aufzug.BL.ListCount - 1
                      F1_Aufzug.BL.List(iL, 0) = _
                         F1_Aufzug.AL.List(iK, 0)
                      F1_Aufzug.BL.List(iL, 1) = _
                         F1_Aufzug.AL.List(iK, 1)
                      F1_Aufzug.BL.List(iL, 2) = _
                         F1_Aufzug.AL.List(iK, 2)
                      wz = Val(F1_Aufzug.AL.List(iK, 3))
                      Warte = Warte + wz
                      If wz > WMax Then WMax = wz
                      F1_Aufzug.AL.RemoveItem (iK)
                      iStat(E, 1) = iStat(E, 1) + 1
                   Else
                      iK = iK + 1
                   End If
                End If
             Else
                iK = iK + 1
             End If
          Loop
       End If
    End If

 'Status
    S = Status(R, E, F, H)

 'Statistik
    For iK = 0 To 9
       For iL = 1 To 2
          F1_Aufzug.SL.List(iK, iL) = _
             Right("     " & Str(iStat(iK, iL)), 5)
       Next iL
       If iK = E Then
```

```
            F1_Aufzug.SL.List(iK, 3) = "X"
         Else
            F1_Aufzug.SL.List(iK, 3) = " "
         End If
      Next iK

      F1_Aufzug.Repaint
   Next iZ

'Endberechnung
   F1_Aufzug.W = Warte
   F1_Aufzug.WMax = WMax
   p = Val(F1_Aufzug.SL.List(10, 1)) - F1_Aufzug.AL.ListCount
   x = Val(F1_Aufzug.W) / p
   F1_Aufzug.W = Format(x, "#0.0")
End Function

Private Function Event_AE(E As Integer) As Integer
   Dim iK As Integer

   Event_AE = 0
   For iK = 0 To F1_Aufzug.AL.ListCount - 1
      If Val(F1_Aufzug.AL.List(iK, 1)) = E Then _
         Event_AE = Event_AE + 1
   Next iK
End Function

Private Function Event_BE(E As Integer) As Integer
   Dim iK As Integer

   Event_BE = 0
   For iK = 0 To F1_Aufzug.BL.ListCount - 1
      If Val(F1_Aufzug.BL.List(iK, 2)) = E Then _
         Event_BE = Event_BE + 1
   Next iK
End Function

Private Function Event_AR(E As Integer, R As Integer) As Integer
   Dim iK As Integer

   Event_AR = 0
   Select Case R
   Case 1
      For iK = 0 To F1_Aufzug.AL.ListCount - 1
         If Val(F1_Aufzug.AL.List(iK, 1)) > E Then _
            Event_AR = Event_AR + 1
      Next iK
   Case 2
      For iK = 0 To F1_Aufzug.AL.ListCount - 1
         If Val(F1_Aufzug.AL.List(iK, 1)) < E Then _
            Event_AR = Event_AR + 1
      Next iK
   End Select
End Function

Private Function Event_BR(E As Integer, R As Integer) As Integer
   Dim iK As Integer

   Event_BR = 0
   Select Case R
```

```
   Case 1
      For iK = 0 To F1_Aufzug.BL.ListCount - 1
         If Val(F1_Aufzug.BL.List(iK, 2)) > E Then _
            Event_BR = Event_BR + 1
      Next iK
   Case 2
      For iK = 0 To F1_Aufzug.BL.ListCount - 1
         If Val(F1_Aufzug.BL.List(iK, 2)) < E Then _
            Event_BR = Event_BR + 1
      Next iK
   End Select
End Function

Private Function Status(R As Integer, _
   E As Integer, F As Integer, H As Integer) As Integer
   Dim AE As Integer
   Dim BE As Integer
   Dim AR As Integer
   Dim BR As Integer

   Select Case R
   Case 0 'ruhend
      AR = Event_AR(E, 1)
      AE = Event_AE(E)
      If AE > 0 Then
         R = 1
         H = 1
         Status = 2
      End If
   Case 1 'aufwärts
      If F > 0 Then 'fahren
         If F < FMax Then
            F = F + 1
            Status = 1
         Else 'anhalten
            F = 0
            H = 1
            E = E + 1
            Status = 2
            AE = Event_AE(E)
            BE = Event_BE(E)
            If AE = 0 And BE = 0 Then
               If E < EMax Then
                  H = 0
                  F = 1
                  Status = 1
                  AR = Event_AR(E, R)
                  BR = Event_BR(E, R)
                  If AR = 0 And BR = 0 Then
                     H = 0
                     F = 1
                     R = 2 'Richtungswechsel
                     Status = 3
                  End If
               Else
                  H = 0
                  F = 1
                  R = 2 'Richtungsumkehr
                  Status = 3
               End If
```

```
            End If
         End If
      Else 'halten
         If H < HMax Then 'halten
            H = H + 1
            Status = 2
         Else 'fahren H=HMax
            If E < EMax Then
               H = 0
               F = 1
               Status = 1
               AR = Event_AR(E, R)
               BR = Event_BR(E, R)
               If AR = 0 And BR = 0 Then
                  H = 0
                  F = 1
                  R = 2 'Richtungswechsel
                  Status = 3
               End If
            Else
               H = 0
               F = 1
               R = 2 'Richtungsumkehr
               Status = 3
            End If
         End If
      End If
   Case 2 'abwärts
      If F > 0 Then 'fahren
         If F < FMax Then
            F = F + 1
            Status = 3
         Else 'anhalten
            F = 0
            H = 1
            E = E - 1
            Status = 4
            AE = Event_AE(E)
            BE = Event_BE(E)
            If AE = 0 And BE = 0 Then
               If E > EMin Then
                  H = 0
                  F = 1
                  Status = 3
                  AR = Event_AR(E, R)
                  BR = Event_BR(E, R)
                  If AR = 0 And BR = 0 Then
                     H = 0
                     F = 1
                     R = 1 'Richtungswechsel
                     Status = 1
                  End If
               Else
                  H = 0
                  F = 1
                  R = 1 'Richtungsumkehr
                  Status = 1
               End If
            End If
         End If
```

```
      Else 'halten
         If H < HMax Then 'halten
            H = H + 1
            Status = 4
         Else 'fahren H=HMax
            If E > EMin Then
               H = 0
               F = 1
               Status = 3
               AR = Event_AR(E, R)
               BR = Event_BR(E, R)
               If AR = 0 And BR = 0 Then
                  H = 0
                  F = 1
                  R = 1 'Richtungswechsel
                  Status = 3
               End If
            Else
               H = 0
               F = 1
               R = 1 'Richtungsumkehr
               Status = 1
            End If
         End If
      End If
   End Select
End Function
```

Wieder gibt es eine Verbesserung zu verzeichnen. Nun beträgt die durchschnittliche Wartezeit 44,1 Sekunden (Bild 11). Eine Verbesserung der längsten Wartezeit wird ebenfalls erzielt, ist aber nicht unbedingt zu erwarten, da es trotz der verkürzten Wege auch immer wieder Fahrten über alle Etagen gibt.

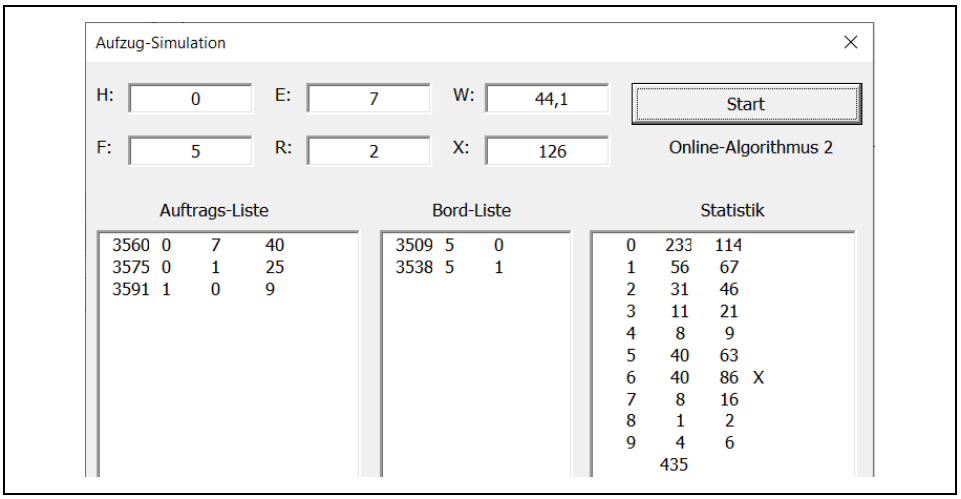

Bild 11. Ergebnisse einer Simulation mit Online-AL 2

5 Online-Algorithmus Nr. 3

In einem weiteren Schritt kann der Aufzug frühzeitig seine Richtung ändern, wenn für ihn die Mehrzahl der Ereignisse in der anderen Fahrtrichtung liegt (Bild 12).

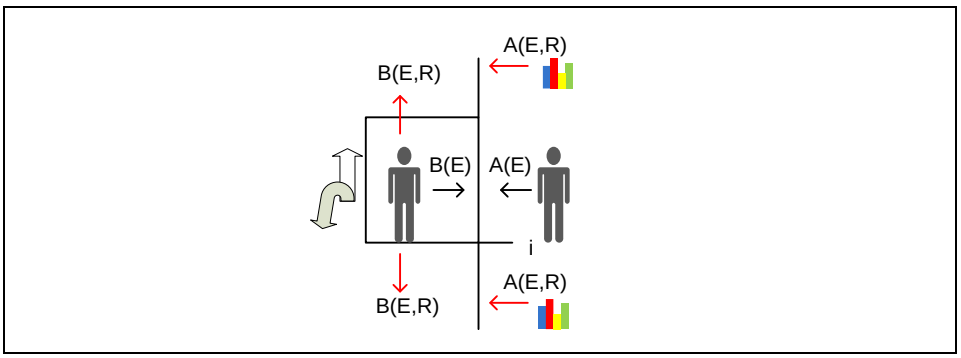

Bild 12. Aufzug-Ereignisse beim Online-AL 3

In einem nächsten Schritt kann dies auch noch mit einer Gewichtung der Entfernung zum aktuellen Standort passieren. Doch dieses Betätigungsfeld überlasse ich zunächst dem interessierten Leser.