Delft University of Technology

FACULTY MECHANICAL, MARITIME AND MATERIALS ENGINEERING
Department Marine and Transport Technology

Mekelweg 2
2628 CD Delft
the Netherlands
Phone +31 (0)15-2782889
Fax +31 (0)15-2781397
www.mtt.tudelft.nl

Specialization: Transport Engineering and Logistics

Report number: 2011.TEL.7616

Title: **Solving the shortest path problem using Ant Colony Optimization and simulation**

Author: M.G. Kruiver BSc

Title (in Dutch) Het oplossen het kortste route probleem met behulp van Ant Colony Optimization en simulatie

Assignment: computer

Confidential: no

Initiator (university): dr.ir. J.A. Ottjes

Supervisor: dr.ir. J.A. Ottjes

Date: August 5, 2011

Student:	M.G. Kruiver BSc	Assignment type:	Computer
Supervisor (TUD):	dr.ir. J.A. Ottjes	Creditpoints (EC):	12
		Specialization:	TEL
		Report number:	2011.TEL.7616
		Confidential:	No

Subject: ***Ant Colony Optimization and simulation***

Ant colonies use indirect communication based on pheromone deposits to determine the shortest path between the nest and a food source. This method, called Ant Colony Optimization (ACO) is currently being used in several traffic models. Simulation is an ideally suitable tool to study this method.
This computer assignment consists of studying the ACO theory and developing and building a simulation model which can determine the shortest path between two points in any given (road) network, using ACO. In other words, artificial ants should solve the shortest-path problem on any undirected graph.

The value of several necessary parameters such as the pheromone evaporation rate needs to be determined using ACO theory and/or model and parameter tuning. A 2D animation depicting the course of the process in time is an additional requirement.

The report should comply with the guidelines of the section. Details can be found on the website.

The professor,

prof.dr.ir. G. Lodewijks

Summary

One of the most surprising behavioral patterns that ants exercise is the ability to find the shortest path to a food source. Biologists have shown experimentally that this is possible because ants communicate with each other using a chemical substance called pheromone that ants can sense and deposit. This kind of communication is a form of indirect communication mediated by modifications of the environment. In some ant species, a foraging ant deposits an amount of pheromone on the ground when travelling back to the nest from a food source. This increase in pheromone level will increase the probability that other ants will follow the same path, which will eventually lead them to the food. This probability increase, together with a continuous evaporation of deposited pheromones, will aid the ants in finding the shortest path between the nest and a food source. In this simulation study, an algorithm based on this ant behavior is created. The resulting Ant Colony Optimization (ACO) model is used to solve shortest path problems on several road networks.

The simulation model provided in this study uses artificial ants which will travel on undirected graphs, such as the map in Figure 1. The artificial ants will construct loop-free, feasible solution from the start node or nest (N1, in green) to the food source or destination node (N13, in red). The user can provide the model with any problem (map) they can think of.

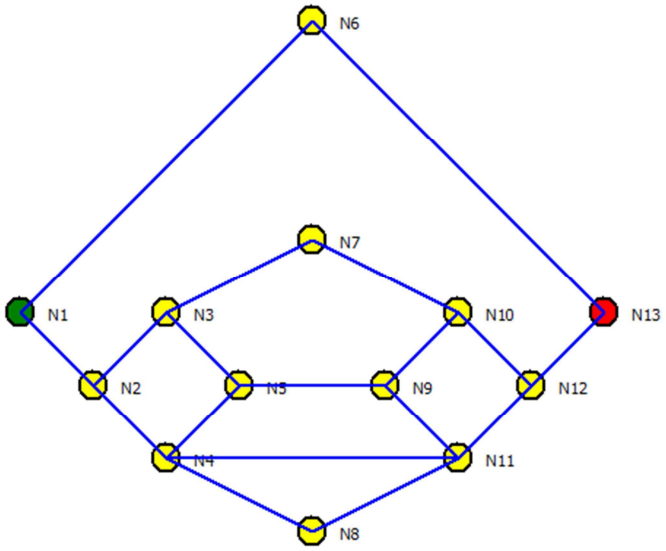

Figure 1 - Road network used in the ACO model

The behavior of the artificial ants is validated against real ant behavior and the model performance is observed using several test maps. The results of these tests are that the artificial ant behavior is valid and when the parameters such as the evaporation rate are set to the right value, the artificial ants are able to solve even more complex problems than real ants would be able to handle. The tuning of the parameters is map dependent, but in general increasing the colony size and enabling a modest pheromone evaporation rate will allow the ants to quickly converge to an optimal solution.

Contents

Preface and assignment description

The behavior of colonial animals like ants and bees has always attracted the attention of human beings. The type of collective behavior exhibited by animals of the same size which aggregate together is called swarm behavior or swarming. The intelligence that seems to exist in these swarms has inspired a considerable amount of researchers – mainly biologists – to study the behavior of these animals. These studies also led to a research field called 'Swarm Intelligence' introduced in 1989 *(Beni, G. 1989).* Part of the Swam Intelligence field is Ant Colony Optimization (ACO), in which the behavior of ants – searching for a path between their colony and a food source – is used to construct algorithms which can solve problems that can be reduced to finding paths in a graph. Initially proposed by Marco Dorigo in 1992 in his PhD thesis *(Dorigo, M. 1992)*, the first algorithm was aiming to search for an optimal path in a graph.

One way of using ACO is making ants search for the shortest path on a graph between A (nest) and B (food) and that is what this simulation study is about: Solving the Shortest Path problem using Ant Colony Optimization. The assignment description is as follows:

Shortest-path problem solving model using 'Ant Colony Optimization'

Ant colonies use indirect communication based on pheromone deposits to determine the shortest path between the nest and a food source. This method, called Ant Colony Optimization (ACO) is currently being used in several traffic models. Simulation is an ideally suitable tool to study this method.

This computer assignment consists of studying the ACO theory and developing and building a simulation model which can determine the shortest path between two points in any given (road) network, using ACO. In other words, artificial ants should solve the shortest-path problem on any undirected graph.

The value of several necessary parameters such as the pheromone evaporation rate needs to be determined using ACO theory and/or model and parameter tuning. A 2D animation depicting the course of the process in time is an additional requirement.

1. Introduction

One of the most surprising behavioral patterns that ants exercise is the ability to find the shortest path to a food source. Biologists have shown experimentally that this is possible because ants communicate with each other using a chemical substance called pheromone that ants can sense and deposit. This kind of communication is a form of indirect communication mediated by modifications of the environment. In some ant species, a foraging ant deposits an amount of pheromone on the ground when travelling back to the nest from a food source. This increase in pheromone level will increase the probability that other ants will follow the same path, which will eventually lead them to the food. This probability increase, together with a continuous evaporation of deposited pheromones, will aid the ants in finding the shortest path between the nest and a food source.

Consider an ant in Figure 2 that has started exploring its environment from the nest **(N)** and has travelled in direction 'a' until he discovered a food source **(F)**. Now that he has discovered a food source, he starts to return to the nest (direction 'b') leaving a trail of pheromone as he goes. This pheromone trail will be picked up by other ants and eventually they will also discover the food source. The map provided does offer the ants a number of crossings where they 'choose' to go either left or right. Because of the evaporation of pheromones over time, the shortest path will eventually reach a distinctive higher amount of pheromone and will be 'chosen' by most of the ants.

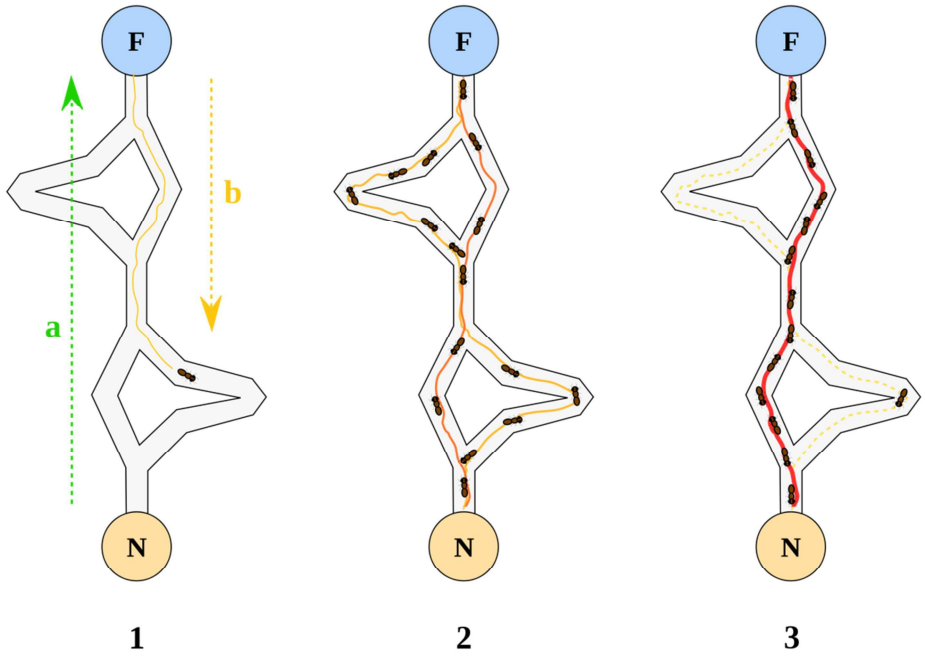

Figure 2 - Real ant behavior

2. The ACO model

The goal is to construct a simulation model using artificial ants which are able to solve shortest-path problems on any (road) network provided. This network will be represented by a static, connected graph $G = (N, A)$ consisting of a set of $n = |N|$ nodes and A is a set of undirected arcs (roads) connecting them, see Figure 3. The two points between which the shortest path is to be found are called start and end node, or in analogous to the real ants: nest and food source.

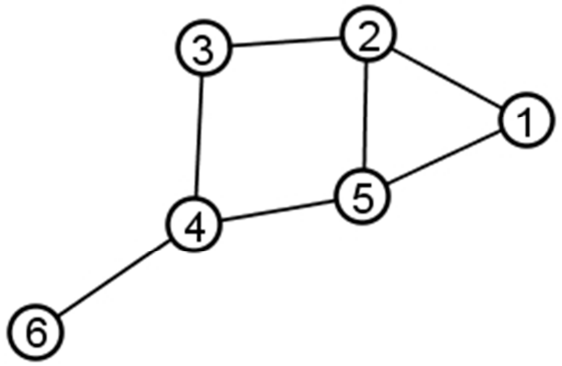

Figure 3 - Undirected graph (G)

Unfortunately, if one would try to solve the shortest-path problem on the graph G using artificial ants whose behavior is only a straightforward extension of real ant behavior, problems will arise. Real ants use both a forward and a backward pheromone trail updating mechanism and as a consequence, artificial ants may get trapped in a loop while generating a solution. Because of the pheromone updating mechanism, such a loop will become more and more attractive and most ants will start to follow it. This means that even if an ant escapes such a loop, the overall pheromone distributions (and thus, the probability distribution) will become such that the short paths are no longer favored and the whole mechanism doesn't work anymore. Because this problem arises due to the forward pheromone updating, the simplest solution seems to remove this forward updating and only rely on backward updating. As it turns out, the forward updating is a necessary behavior to obtain convergence of the ant colony to the shortest branch. If an ACO model is considered in which ants deposit pheromone only during either the forward or the backward trip, then the result is that the ant colony is unable to choose the shortest branch. Observations of real ant colonies have confirmed that ants that deposit pheromone only when returning to the nest are unable to find the shortest path between their nest and the food source *(Deneubourg, 2002)*.

Therefore, the capabilities of the artificial ants will need to be extended in such a way that, while they retain the most important characteristics of real ants, they are able to solve shortest-path problems on generic graphs. The solution to this problem is (in this case) to give the ants a limited form of memory in which they can store the path they have taken so far (from the start to where they are at a current moment) as well as the number of steps they have taken to get there. Using this memory, the ants can exercise some useful behavior that allows them to efficiently build feasible solutions to the shortest-path problem. This behavior consists of: (§2.1) probabilistic forward solution construction based on

pheromone levels; (§2.2) backward path tracking and pheromone updating and; (§2.3) evaluating the solution quality based on the number of steps taken and using this solution quality to determine the amount of pheromone that should be deposited while backtracking their path.

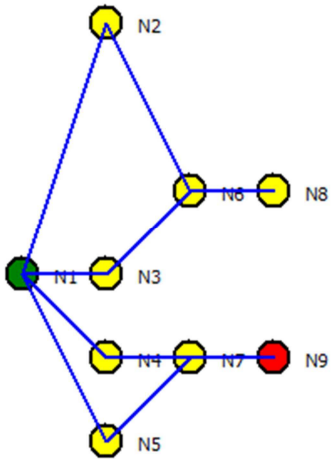

Figure 4 - Road network for the ACO model

2.1 Probabilistic forward solution creation

The ants in the ACO model devised for this simulation study effectively have two 'working modes': forward and backward mode. In forward mode, the ants travel the network from the start node (green in Figure 4) looking for the end node (red in Figure 4) using the current pheromone levels. An ant builds a solution to the shortest-path problem by applying a step-by-step decision policy embedded in the ant process. At each decision point – a node which is connected to more than one other node – the ant will read (sense) the local information stored on the outgoing roads and use this information in a stochastic way to 'choose' his next destination. This will be done by taking a sample from a destination distribution. This destination distribution is a probability which consists of all possible roads the ant can choose from; these are the roads leading to the connected nodes on the graph. The relative probability of each road is based on its pheromone level compared to the total amount of pheromones in the current destination distribution at that time. At the beginning of the search process, a constant amount of pheromone (e.g. $\tau_{ij} = 1, \forall (i,j) \in A$) is assigned to all the arcs. When located at node i ant k uses the pheromone trails τ_{ij} to compute the probability of choosing j as the next node:

$$P_{ij}^{k} = \begin{cases} \frac{\tau_{ij}}{\sum_{l \in N_i^k} \tau_{ij}}, & \text{if } j \in N_i^k; \\ 0, & \text{if } j \notin N_i^k; \end{cases}$$

Equation 1

where N_i^k is the neighborhood of ant k when in node i. In this ACO model the neighborhood of a node i contains all the nodes directly connected to node i in the graph $G = (N, A)$, except for the predecessor of node i (the last node the ant visited before moving to i). In this way, the ants avoid returning to the same node they visited immediately before node i. Only in the cases when either N_i^k is empty (which corresponds to a dead end) or when the ant is on node 1 (the start node), node i's predecessor is

included into N_i^k. Ants will hop from node to node using this decision policy until they come across the end node. Due to differences among the ants' paths, the time step at which the ants reach the destination may vary from other ants. Ants traveling on shorter paths will reach their destination earlier. Note that the probability P_{ij}^k (Equation 1) is influenced by the pheromone trails deposited on the graph by the ants. Ants that are moving forward (constructing solutions) do not deposit any pheromone while moving. This, together with carefully determined backward moving, helps avoiding the formation of self-reinforcing loops. In summary, in forward mode an ant is constructing a feasible solution to the shortest-path problem.

2.2 Backward path tracking and pheromone updates

The artificial ant stores the nodes visited and the number of steps taken in its memory. This memory allows an ant to retrace the path it has followed while searching for the end node. On top of this, it enables the artificial ants to improve their performance by using a loop elimination mechanism. When the end node is found, the ant will enter backward mode. In backward mode, the ant removes any loops from the path it memorized (§4.4) and starts retracing his path while depositing pheromone on all arcs (roads) it traverses. While traveling this loop-free path, ant k will deposit an amount $\Delta\tau^k$ of pheromone which changes the pheromone level τ_{ij} of the roads (i, j) the ant has traveled as follows:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau^k$$

Equation 2

These changes influence the environment in such a way that the probability that other ants will choose the same roads will increase. In the ACO model, the artificial ants will be all be objects of the same Ant class and follow the same ant process, influencing the environment as they deposit pheromones on the map.

2.3 Pheromone deposits based on solution quality

Because the ants memorize both the nodes they visit as well as the amount of steps taken (the cost of the arcs they traversed) they can evaluate the cost of the solution they constructed. Using this cost, ants can modulate the amount of pheromone $(\Delta\tau^k)$ they deposit while in backward mode. Making the pheromone drop amount dependent of the solution quality can help the ants in directing the colony more strongly toward better solutions. In the ACO model, this feature is implemented by making the pheromone drop amount a non-increasing function of the path length (the longer the path, the lower the amount of pheromone that is deposited).

2.4 Pheromone trail evaporation

In real ant colonies, pheromone intensity decreases over time because of evaporation. Even though in the experiments run by *Deneubourg et al (1990)* pheromone evaporation did not play any noticeable role because of the very slow evaporation rate, it can be very useful for artificial ants in ACO models to improve performance. Pheromone evaporation reduces the influence of the pheromones deposited in the early stages of a search, when ants do not yet have a clue which path to follow and might build solutions of very poor quality. So pheromone evaporation can be seen as an exploration mechanism that avoids quick convergence of the ants to suboptimal solutions. A decrease in overall pheromone intensity will promote the exploration of different paths during the whole search process. Even though the influence of evaporation on real ants' path finding seems to be unimportant, it could actually be very important for artificial ants due to the fact that the problems tackled with ACO models are much more complex than those that any real ant species can solve. A mechanism like evaporation allows a continuous improvement of the available solutions.
In this ACO model, pheromone evaporation is simulated by an evaporation rate that can be set to a constant value. After a set period of time, pheromone trails are all reduced (evaporated) by applying the following equation to all arcs:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij}, \qquad \forall (i,j) \in A,$$

Equation 3

Where $\rho \in (0,1]$ is the constant evaporation parameter that can be configured in the ACO model.

The whole ACO model described in the previous paragraph will be implemented in Delphi/Tomas. Before this is done, the model will first be described in pseudo code in the next chapter. This chapter is called the Program Descriptive Model (PDL), which will be translated into Delphi code.

3. PDL model

This chapter describes the entire simulation model in *program descriptive language*, or pseudo code. All of the object classes and their properties will be defined and processes will be described. This PDL model is implemented in Delphi/Tomas for this simulation study. For more details and the final implementation, refer to the ACO program Delphi/Tomas Source code.

The following classes and their attributes, functions and procedures will be defined: World, Node, Road, Node, Score, Ant, AntGenerator and UpdateClass.

World *{Global Settings}*

- NumberOfAnts — nAnt ∈ [1, ∞]
- AntPheromoneCapacity — α ∈ (0, ∞)
- PheromoneEvapRate — β ∈ (0, 1]
- AntSpeed — vAnt [m/s]
- UpdateRate — pheromo`ne update interval [ms]
- DisSeeder — Seed offset variable
- KeepSearching — Boolean: False when shortest path is found
- AllRoadsQ — TomasQ: All Roads
- AllNodesQ — TomasQ: All Nodes
- AllAntsQ — TomasQ: All Ants
- AntsinSystem — Current nr. of ants in system (for generator)
- TotalRoadLength — Length of all roads on the map (for pheromones)
- ScoreList — TList of all scores (length of ant solutions)

Node

- MyID — Unique number: arbitrary
- MyType — Node type: 1, 2 or 3
  //1=start 2=normal 3=end
- X — x-coordinate
- Y — y-coordinate
- RoadQ — TomasQ: All connected Roads

Road

- MyID — Unique number: arbitrary
- N1 — String, Name of First Connected Node
- N2 — String, Name of Second Connected Node
- MyN1 — Node, N1
- MyN2 — Node, N2
- PheromoneLevel — Current Pheremone level
- Length — [m]

Score

- Amount — Double

Initiation ***('Open file')***

- Input = Inputfile.txt

---------- **Procedure**

```
InputFile = TomasFile.Create(SelectInputFile.filename)

//Create World
      Map=World.Create('Map')
      Map.NumberOfAnts = AntsForm.NrAntsTrack.Position
      AntsForm.NrAntsLabel.Caption = 'Nr Ants ' +
        IntToStr(AntsForm.NrAntsTrack.Position)
      Map.AntsInSystem= 0
      Map.EvapRate = ( AntsForm.EvapTrack.Position / 100 )
      AntsForm.EvapRateLbl.Caption = 'EvapRate ' +
        IntToStr(AntsForm.EvapTrack.Position)
      Map.AntSpeed = 0.6
      Map.UpdateRate = 500
      Map.DistSeeder = 1
      Map.KeepSearching = true
      Map.AntPheromoneCapacity:=(((Map.TotalRoadLength)/1000));
      Map.AllNodesQ=TomasQueue.Create('AllNodesQ')
      Map.AllRoadsQ=TomasQueue.Create('AllRoadsQ')
      Map.AllAntsQ=TomasQueue.Create('AllAntsQ')
      Map.TotalRoadLength=0
      Map.ScoreList = TList<Score>.Create

NrOfNodes=InputFile.GetInteger //first number of input file = nr of Nodes
For i=1 To NrOfNodes Do

NewNodeName=InputFile.GetString
NewNode=Node.Create(NewNodeName)
NewNode.MyID=NewNodeName
NewNode.MyType=InputFile.GetInteger
NewNode.X=InputFile.GetInteger
NewNode.Y=InputFile.GetInteger

Map.AllNodesQ.AddToTail(NewNode)
NewNode.RoadQ=TomasQueue.Create(NewNode.MyID + '_RoadQ')

 //draw Nodes in OutputBox
 case NewNode.MyType of
   1:                    //startnode = green
     OutputBox.Canvas.Brush.Color=clGreen
   2:                    //normal node = yellow
     OutputBox.Canvas.Brush.Color=clYellow
   3:                    //endnode = red
     OutputBox.Canvas.Brush.Color=clRed
   else                  //type <> 1,2,3 = purple = error!
      OutputBox.Canvas.Brush.Color=clPurple
  End case

OutputBox.Canvas.Pen.Color=clBlack
OutputBox.Canvas.Pen.Width=2
```

```
        OutputBox.Canvas.Ellipse(Round(NewNode.X),Round(NewNode.Y),Round(NewNode.
        X)+20,Round(NewNode.Y)+20)
        OutputBox.Canvas.Brush.Color=clWhite
        OutputBox.Canvas.TextOut(NewNode.X+30, NewNode.Y+5, NewNode.MyID)

        //create Roads
        NrOfRoads=InputFile.GetInteger //second number of input file = nr of Roads

        For j=1 To NrOfRoads Do

        NewRoadName=InputFile.GetString
        NewRoad=Road.Create(NewRoadName)
        NewRoad.MyID=NewRoadName
        NewRoad.PheremoneLevel= 1

        NewRoad.N1=InputFile.GetString
        NewRoad.N2=InputFile.GetString

        NewRoad.MyN1=Map.AllNodesQ.ElementWithName(NewRoad.N1)
        NewRoad.MyN2=Map.AllNodesQ.ElementWithName(NewRoad.N2)

        Map.AllRoadsQ.AddToTail(NewRoad)
        NewRoad.MyN1.RoadQ.AddToTail(NewRoad)
        NewRoad.MyN2.RoadQ.AddToTail(NewRoad)

        //Calculate Road Length
        NewRoad.Length=round(sqrt(Power(NewRoad.MyN1.X-
                NewRoad.MyN2.X,2)+Power(NewRoad.MyN1.Y-NewRoad.MyN2.Y,2)))
        Map.TotalRoadLength=(Map.TotalRoadLength + NewRoad.Length)

        //draw Road
        OutputBox.Canvas.Pen.Color=clBlue
        OutputBox.Canvas.Pen.Width=2

        OutputBox.Canvas.MoveTo((NewRoad.MyN1.X+10), (NewRoad.MyN1.Y+10))
        OutputBox.Canvas.LineTo((NewRoad.MyN2.X+10), (NewRoad.MyN2.Y+10))

AntGen = AntGenerator.Create('AntGenerator')
AntGen.Start(TNow)

Updater:=UpdateClass.Create('Updater')
Updater.Start(TNow)
InputFile.Free
```

AntGenerator

---- Procedure

```
//Generate all ants at t=0

Repeat while Map.AllAntsQ.Length < NumberOfAnts
        New Ant ('Ant')
                Ant.ID = (TotalAntsQ.Length +1)
                Ant.PheromoneCapacity = AntPheromoneCapacity
                Ant.Speed = AntSpeed
                Ant.myNode = Node 'N1'          [NodeID = 1 = start node]
                Ant.VisitedNodesQ = Q = empty
                Ant.NextNode = Node = empty
                Ant.DestinationQ = Q = empty
                Ant.DestinationDistribution = Distribution = empty
                Ant.myRoad = RoadID = empty
                Ant.StepsTaken = double = 0
                Ant.EnterQue(Map.AllAntsQ)
                Ant.Start(TNow)

End Repeat
StopAndLeave
```

Updater

- AllPheromones [numeric value]
-

---- Process

```
Repeat
Hold (Map.Updaterate)

AllPheromones = 0

//Update pheromone values
        For All Roads in AllRoads.Q
                Road[i].PheromoneLevel =
                        Road[i].PheromoneLevel * Map.EvapRate
                AllPheromones  =
                        AllPheromones + Road[i].PheromoneLevel
                Next Road
        End For


//Draw roads according to relative pheromone level
        For All Roads in AllRoads.Q
                Draw line between Road[i].N1.X,Y & Road[i].N2.X,Y
                        With thickness =
                        (Road[i].PheromoneLevel / AllPheremones) * 50 pixels
                Next Road
        End For
```

Ant

- myID — String, Unique name
- PheromoneCapacity = α — (Read from World data)
- Speed = vAnt — (Read from World data)
- MyNode — Node, Ant's current location
- MyRoad — Road, Current road
- StepsTaken — Steps taken on forward cycle (=solution cost)
- StepsBack — Steps taken on backward cycle (=score)
- MyScore — Score, StepsBack when at Start Node
- Stepnumber — Current iteration number since Start Node
- VisitedList — TList of Node: Nodes Already visited (forward cycle)
- BackW — Array of Node: Nodes Already visited (backward cycle)
- NextRoad — Road, Next Road to travel on
- NextNode — Node, Next Node (destination)
- DestinationQ — TomasQ, all *possible* destinations (connected Roads)
- DestinationDistribution — TTableDistribution, Tomas distribution of destinations
- TotalDestinationLevel — Pheromone level of ALL possible destinations
- DropAmount — Amount of pheromone to be dropped on a road
- FoundTarget — Boolean, TRUE if Ant found End Node
- FoundStart — Boolean, TRUE if Ant returned at Start Node
- AverageScore — Current average score of all ants, used for sorting
- DisplayScore — Ant' score that should be displayed in ScoreList
- DeletedSomething — Boolean, used for loop removal

--- **Function** Ant.NodeFromRoad (MyNode:Node; MyRoad:Road): returns Next Node

```
    //Assigns NextNode from Road identifier (forward mode)

    {If N1 equals the current Node, N2 is the next Node}
    If MyRoad.N1 = MyNode then
            Next Node = N2
    Else {N1 is the next Node}
            Next Node = N1
```

--- **Function** Ant.RoadFromNode (DestinationQ:TomasQ; NextNode:Node): returns Road

```
    //Assigns NextRoad when NextNode is known (backward mode)
    //NextNode and MyNode are known, MyNode.RoadQ equals DestinationQ
    //check entire DestinationQ for a match

    For I = 0 to DestinationQ.length-1
            NextR:=DestinationQ.Element(i)
            If NextR.MyN1 = Nextnode then  {NextNode is connected to THIS NextR}
                    Next Road = NextR
            ElseIf NextR.MyN2 = NextNode then
                    Next Road = NextR
            {Else: check next Road}
```

--- **Procedure:** Ant.FillDestinationDistribution (DestinationQ:TomasQ)

```
//Fill the table distribution with probabilities based on pheromone levels

{calculate total pheromone level)
For I = 0 to DestinationQ.length-1
        CurrentRoad = DestinationQ.Element(i)
        Totallevel = totallevel + CurrentRoad.Pheremonelevel

{add Roads to DestinationDistribution}
For J = 0 to DestinationQ.length-1
        CurrentRoad = DestinationQ.Element(j)
        X = CurrentRoad.MyID
        F = CurrentRoad.PheremoneLevel / Totallevel
        DestinationDist.AddValue(X,F)
```

--- **Process:** Ant.Process

```
FoundTarget, FoundStart = False
StepsTaken, StepNumber = 0

Repeat while KeepSearching is True

//Ant starts in forward mode and will be at the Start Node

VisitedList.Count = Stepnumber+1
{Add current Node to VistedList}
VisitedList.Items[Stepnumber] = MyNode
DestinationQ = MyNode.RoadQ

//Forward mode: 1 option
{If there is only one option (road) to take: take it}
If DestinationQ.Length = 1 then
MyRoad = DestinationQ.FirstElement  //First element is also the only element
NextNode = NodeFromRoad (MyNode, MyRoad)  //Returns Next Node

//Forward mode: 2 options (while NOT @ Start Node)
//If there are two options: Don't take the previous Road
//Note: This is NOT valid @ Start Node
If DestinationQ.Length = 2 AND MyNode.Mytype = 2 then

        if DestinationQ.Element(0) = MyRoad then
        //element 1 = current road, next road is element 2
                MyRoad:=DestinationQ.Element(1);
                NextNode:=NodeFromRoad (MyNode, MyRoad);
              else
        //element 2 = current road, next road is element 1
                MyRoad:=DestinationQ.Element(0);
                NextNode:=NodeFromRoad (MyNode, MyRoad);
```

```
        //Forward mode: >2 options (or >1 option @ start)
        Else //There are more than two options: make choice based on pheromone levels

        Read optional seed offset --> j

        {grab a seed from TomasSeeds}
        i:=Seed(Map.DistSeeder+j);

        //create and fill a new destination distribution
        DestinationDist = TTableDistribution.Create(i, discrete)
        FillDestinationDist(DestinationQ)

        //choose destination (sample from destdist)
        NextRoad = (DestinationDist.sample)
        MyRoad = DestinationQ.Elementwithname(floattostr(NextRoad))
        NextNode = NodeFromRoad (MyNode, MyRoad)  //Returns Next Node

//NextNode and NextRoad have now been defined for all possible scenarios
//Now the Ant walks to the Next Node

Hold (myRoad.Length / Speed)
myNode = NextNode
StepNumber = Stepnumber +1

//We arrived at the Next Node and check if we already found the End Node
Case MyNode.MyType OF:
        2: FoundTarget = False //haven't found the End Node yet

        3: FoundTarget = True //Found the End Node!
           FoundStart = False //Start to 'look' for the Start Node again

{If the ant found the target it starts the backward mode}
{First, it removes any loops from its path}

DeletedSomething = False
P,R,T = 0

While P < VisitedList.Count-1 do
        CheckNode = VisitedList.Items[P]
        For Q = P+1 to VisitedList.Count-1 do
        If VisitedList.Items[Q] = CheckNode then
                //a match is found which corresponds to a loop
                //a range of nodes will be deleted from the VisitedList to remote the loop
                VisitedList.DeleteRange(P, Q-P)
                StepNumber = Stepnumber – (Q-P)
                DeletedSomething = TRUE
                Break   //exit THIS loop
        If DeletedSomething = True then
                //restart the search
                P = 0, DeletedSomething = False
        Else //Nothing was deleted (no loop detected)
                P = P+1 {Check next Node for matches}
```

```
//The ant now has a loop free path which it will follow back to the Start Node (nest)

While FoundStart = False do

NextNode = VisitedList.Items[(StepNumber-1)]
DestinationQ = MyNode.RoadQ
MyRoad = RoadFromNode(DestinationQ, NextNode, MyNode)

Hold((MyRoad.Length) / Speed)
Stepsback  = (Stepsback+MyRoad.Length)
DropAmount = Map.AntPheromoneCapacity

//Drop Amount dependent of StepsTaken?
if DropAmountDependent = TRUE then
DropAmount = (Map.AntPheromoneCapacity / (StepsTaken))

MyRoad.PheremoneLevel = MyRoad.PheremoneLevel + DropAmount

      MyNode= NextNode
      StepNumber=StepNumber-1

        Case MyNode.MyType OF {Type 1 (start), 2 (normal) or 3 (end)}
            1: //Ant is @ Start Node (found goal)
              FoundStart = True
              MyScore= Score.Create(MyId + 'AntScore')
              MyScore.Amount= StepsBack

              Map.ScoreList.Add(MyScore)
              Map.ScoreList.Sort(Scores)

              // Clear ScoreMemo (GUI)
              AntsForm.ScoreMemo.Clear
              // Display 25 best scores only (truncate scorelist for speed/handling purposes)
              if Map.ScoreList.Count < 25 then
                 MaxScoreNr=Map.ScoreList.Count-1
              else
                 MaxScoreNr=24 {0 to 24 = 25 items}

              //Reset average top25 score
              AverageScore=0

              // And display the ScoreList in ScoreMemo (GUI)
              for s = 0 to MaxScoreNr do
               if Antsform.ScoreCheckBox.Checked = TRUE then
               AntsForm.ScoreMemo.Lines.Add(floattostr(Map.ScoreList.Items[s].Amount))

              //calculate average top25 score
                AverageScore= ( AverageScore + (Map.ScoreList.Items[s].Amount) )
              DisplayScore = Round((AverageScore))
              DisplayScore = Round(Displayscore / (MaxScoreNr+1))
              AntsForm.Label7.Caption= inttostr(DisplayScore)
```

```
                    //Display iteration number (GUI)
                    if Antsform.IterationsCheckbox.Checked = True then
                          Antsform.IterationsLabel.Caption=Inttostr(round(Map.ScoreList.Count/
                          Map.NumberOfAnts))

                    //end condition 1:
                    //compare avg top25 score with total score after min. iterations per ant
                    if Map.ScoreList.Count > (AntsForm.MinRunsTrack.Position *
                       Map.NumberOfAnts) then
                        //If average score equals best score, the ants are considered converged
                        if DisplayScore = Round(Map.ScoreList.Items[0].Amount) then
                          Map.KeepSearching = False {End the simulation}
                          Antsform.ConvergedPanel.Color = clGreen
                          Antsform.ConvergedPanel.Caption = 'yes'

                    //end condition 2:
                    //stop after max. iterations per ant
                    if Map.ScoreList.Count > (Antsform.MaxRunsTrack.Position *
                       Map.NumberOfAnts) then

                      Map.KeepSearching = False
                      Antsform.ConvergedPanel.Color = clGreen
                      Antsform.ConvergedPanel.Caption = 'yes'

                    //Reset steps/scores and continue
                    StepsTaken=0
                    StepsBack=0
                    StepNumber=0
                    continue

                2: //Ant is @ a normal node: continue to follow path
                      FoundStart = False


                3:  //ant is back @ end node (shouldn't happen!)
                      FoundStart = False
                      showmessage('I failed… sorry!')

            End Case

//Stop if it's time (global condition)
If Map.KeepSearching = False then
        InterruptSimulation //Interrupt the simulation

End of Ant.Process;
```

4. Implementation and issues

For this simulation study, the PDL model is implemented in Delphi/Tomas. During implementation of the PDL, some issues have occurred which have been ironed out in the progress and have been documented below. Some issues have resulted in a change, update or refinement of the PDL model, these changes have already been incorporated in the PDL model of Chapter three, such that this Chapter provides a complete and correct PDL model.

4.1 Distributions & seeds

When using probability distributions such as the TomasDistributions, one needs a (random) seed to initiate the process of taking samples from such a distribution. Issues here are that a unique seed is required to generate unique and random samples. On the other hand, using the same unique seed values, runs become reproducible which is preferable for experimental and demonstrational purposes. The approach in this particular simulation study is to use a user-controlled seed offset value and a Delphi unit called 'ACO Seeds' containing a large number of seed values. This seed offset value determines both which seed value is retrieved from the ACO seeds unit and in which way the following seed values will be retrieved. The result is that while the model can produce several *different* unique and valid runs, these runs are also reproducible when the parameters are set to the exact same values. This allows for unique experiments and reproducible runs for demonstrational purposes.

The second issue that occurred also seemed to do with the TomasDistributions, but turned out to be an error in the Ant process where ants were not allowed to travel on the branch they just had followed to arrive at the decision point. This behaviour caused problems in networks with two or more branches originating from the start node. When an ant returned at the start node traveling on the shortest path, it wasn't allowed back in the network using its previous branch, which actually would have been the 'best choice'. This error presented itself as weird ant behaviour and seemed to be caused by invalid samples from the DestinationDistribution. When the filling and sampling from the TomasDistribution was tested step-by-step and proved to be valid, attention was redirected to the Ant process itself. When this process was thoroughly reviewed, the decision process was found to be faulty since it would not allow ants to re-enter the network on the branch they travelled to reach the start node. This decision policy has been revised and is now in the form that can be found in the PDL model. This revised decision policy does allow ants to travel on the branch they have taken to reach the decision point, if this is necessary.

4.2 The VisitedNodesQ and backtracking the route

When ants visited a particular node more than once on their route from start to end, something weird happened when the ants started tracking back their route to the start. Say an ant visited the following sequence of nodes: 1-3-5-4-5-7, where 1 is the start node and 7 is the end node. When this ant started tracking back its path, it would repeatedly select the previous node from its VisitedNodesQ and visited all the nodes in reversed order. So we would expect the route from the end node to the start node to be: 7-5-4-5-3-1, the reversed order of the forward sequence. But in fact, this ant took the following sequence on its way back to the start: 7-5-5-4-3-1. The ant visited node 5 two times in a row, instead of visiting node 4 in between. This issue finally turned out to be a kind of sorting behavior of TomasQueues which cannot indifferently handle objects with the same name without sorting them (and augmenting a sub-identifier: Node5.1, Node 5.2, etc.). This problem has been mitigated by using an Array to store the visited nodes (VisitedNodesArray) instead of a TomasQueue (VisitedNodesQueue) which was originally planned. The ant process itself hasn't undergone any procedural changes to correct this behavior and the ants will now backtrack their path in the same way as in forward mode.

4.3 Scorekeeping and end conditions

Another issue was that of scorekeeping and end conditions. From an experimental point of view the model should allow the ants' performance to be measured. This performance will be registered using Scores which consists of the number of steps an ant has taken to reach the goal. Besides scorekeeping, the simulation should also be able to stop after some end conditions are met. These end condition include the following: minimal and maximal number of iterations per ant, average score and the top 25 scores.

The scores of the ants are measured in the number of steps they take on their way back from the food to the nest (this equals the cost of the solution). This means that the lowest score represents the best performance and best found solution so far. These scores are all stored in a Delphi element called Generic TList. This TList element allows scores to be stored and easily ranked from low to high. Using the best 25 scores in the List, the Average Score value is computed.

Using the Score values and the recorded number of iterations per ant, the end conditions are implemented as follows. 1: When the condition of maximum number of iterations per ant is met, the simulation will be interrupted. 2: When the condition of minimum iterations per ant is met *and* the value of the Average Score equals the value of the nr.1 Score, the model is considered to be converged to a solution and the simulation will be interrupted.

4.4 Self-Reinforcing Loops and how to eliminate them

The last problem encountered in a later stage was that of self-reinforcing loops. In (large) networks where it is possible for an ant to create suboptimal paths by making loops, these loops would receive a lot of pheromones and attracted a lot of ants. This finally caused ants to get infinitely stuck in these loops because they were self-reinforcing, as expected and described in Chapter 2.

The loop elimination is implemented by iteratively scanning the node identifying numbers (Node ID's) position by position, starting from the Nest node. For the node at the i-th position, the path is scanned from the end node until the first occurrence of that node is encountered at position j (it always holds that $i \leq j$ because the scanning process stops at position i at the latest). When $j > i$, the subpath from position $i + 1$to position j corresponds to a loop and can be removed from the path. This loop elimination procedure removes loops in the same order as they are created, and doesn't necessarily remove the longest loops but does provide the ants a loop-free path. This loop elimination process is depicted in Figure 5 and its process can also be found in the Ant Process.

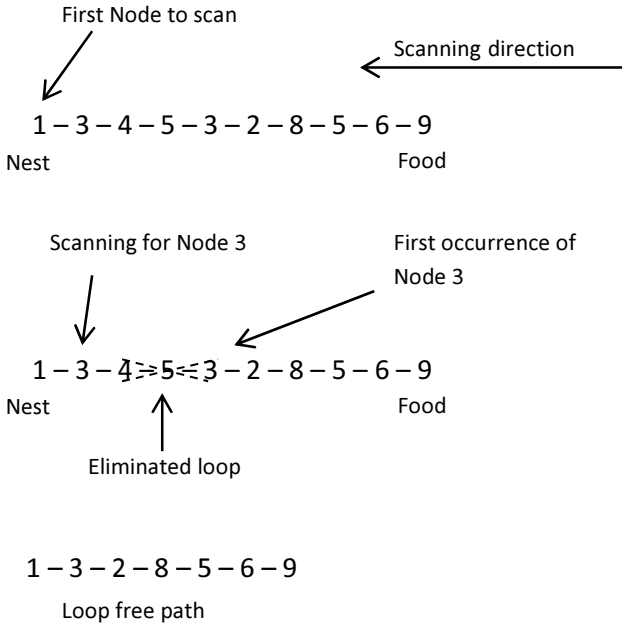

Figure 5 - Loop elimination process

5. Model validation

5.1 The double bridge experiment

To validate the simulation model and the behavior of the ants, the model was run on several networks (representing road networks). The first set of maps is called the double bridge experiment, see Figure 6. In this experiment, ants start on the green start node (N1) and will try to find the shortest path to the red end node (N4).

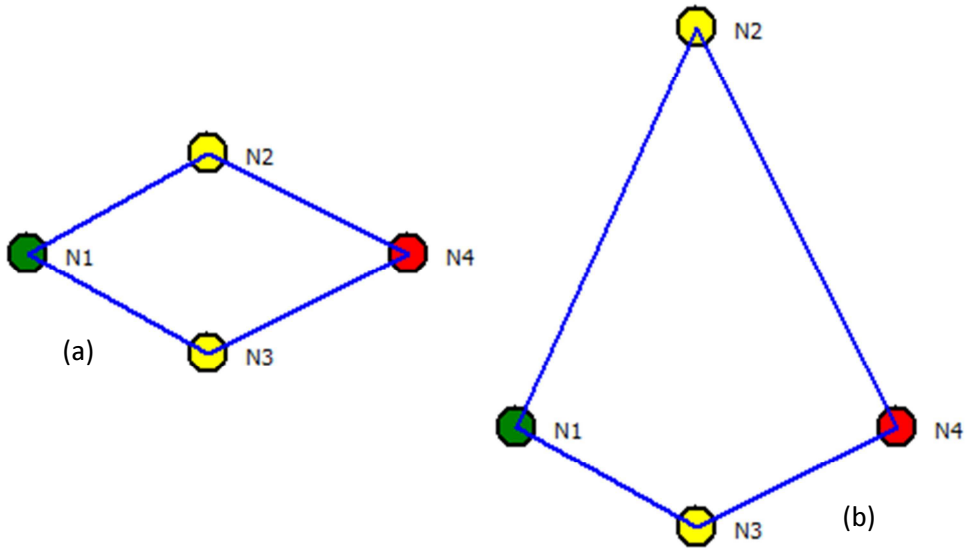

Figure 6 - Double bridge experiment, equal length (a) and different length (b) networks

On the first map (left side) the upper and lower routes are of the same length and the resulting path used by the ants is a trivial choice. The expected result after a number of experiments is that the ants use one branch or the other approximately the same number of trials.
A comparable experiment has been conducted with real ants (*Iridomytmex Humulis)* by *Goss et al. (1989).* It was already known that the foraging behavior of ants is based on indirect communication mediated by pheromones. The pheromone trail-laying and -following behavior of some ant species has been investigated in controlled experiments by several researchers. They ran various experiments varying the ratio $r = l_l/l_s$ between the lengths of the two branches of the double bridge, where l_l is the length of the longer branch and l_s is the length of the shorter one.
In the first experiment the bridge had two branches of equal length (r=1). At the start, ants were introduced in the system and were allowed to freely move between the nest and the food source and the percentage of ants that chose one or the other of the two branches were observed over time. The outcome was that, although some random choices occurred in the initial phase, eventually all ants used the same branch. This result can be explained as follows. When a trial starts there are no pheromones in the system. Hence, the ants do not (yet) have a preference and they will select any of the two branches with the same probability. Yet, because of random fluctuations, a few more ants will select one branch over the other. Because ants deposit pheromone while walking, a larger number of ants on a branch

results in a higher pheromone level. This higher pheromone level in turn stimulates more ants to choose that branch again, and so on until finally the ants converge to one single path.

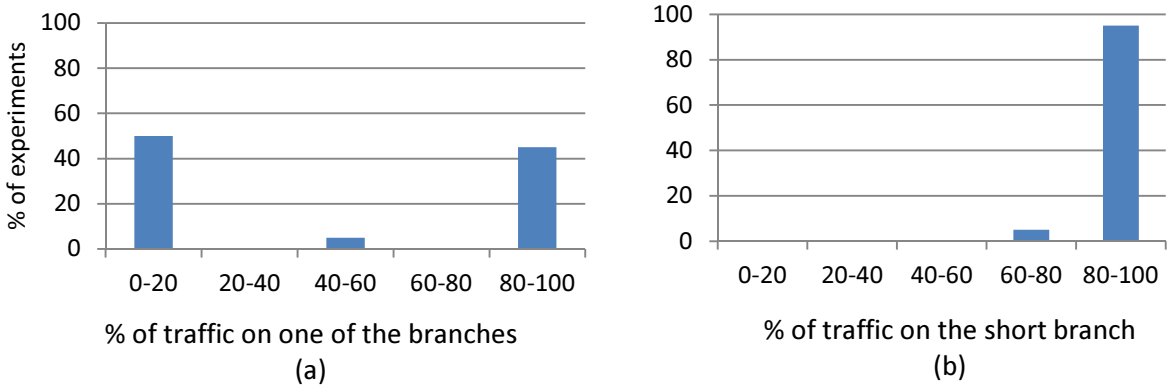

Figure 7 - Results of the double bridge experiment using real *I. Humilis ants*. (a) Results for the case with equal branches (r=1). (b) Results for the case where one branch is twice as long as the other (r=2).

In the second experiment, the length ratio between the two branches was set to r=2, so that the long branch was twice as long as the short one (see Figure 6 b). Ants are released on the map as in the first experiment, and they are allowed to explore the environment. At the decision point, the two branches appear to be identical to the ants so they choose randomly. Therefore it is expected that half of the ants will travel on the short branch and the other half on the long one, although stochastic oscillations may occasionally favor one branch over the other. Because one branch is shorter than the other, the ants traveling on the short branch are the first to find the food and return to the nest. But then, when they again reach the decision point, the higher level of pheromone on the short branch will bias their decision in its favor. Therefore, pheromone starts to accumulate faster on the short branch, which will eventually be used by all the ants.

An interesting case was to see what happened when the ant colony was offered a new shorter connection from nest to food, when they already converged to one branch. This case was studied in an additional experiment in which initially only the long branch was offered to the colony and after 30 minutes, the short branch was added. In this case, the short branch was only selected sporadically and the colony was effectively 'trapped' on the long branch. This can be explained by the high pheromone concentration of the long branch and the low pheromone evaporation rate. This also indicates that the ant's foraging method is susceptible for getting trapped in loops or on longer routes once they have been travelled by a large number of ants and thus have received a high pheromone concentration. So in fact, most of the ants keep choosing the long branch because of its high pheromone concentration and this behavior keeps reinforcing the use of the long branch.

To test the validity of the ACO model that was developed for this simulation study, the artificial ant behavior will be compared with the actual ant behavior that was determined using the experiments described above.

Twenty runs with 10 ants and an evaporation rate of 0.9 were run on both maps and the results were recorded. Each run received a new, unique seed number (seed offset 0-19) to ensure unique, reproducible stochastic behavior.

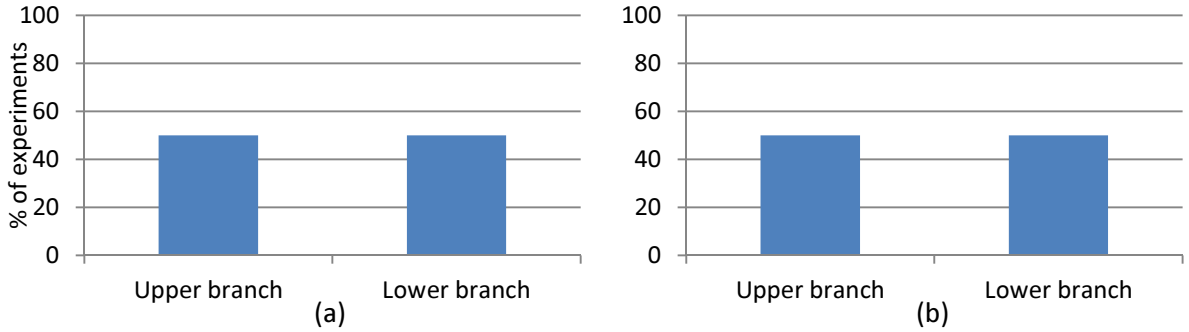

Figure 8 - Results of the even bridge experiment with ACO. (a) Constant pheromone drop amount $\Delta\tau^k$ and (b) pheromone drop amount $\Delta\tau^k$ dependent of solution quality.

The results of the even bridge experiment (Figure 8) show that the artificial ants seem to behave the same as the real ants did in the experiments of Goss et al. These results are reproducible with the ACO model provided. The runs were done both with the pheromone drop amount independent of the number of steps taken (a) and dependent of the steps taken (b). This shows that the option "Drop amount depends on steps taken" doesn't seem to invalidate the ant behavior.

While the even bridge experiment shows the expected results, the ACO performance at the same settings on the uneven bridge experiment were different, see Figure 9.

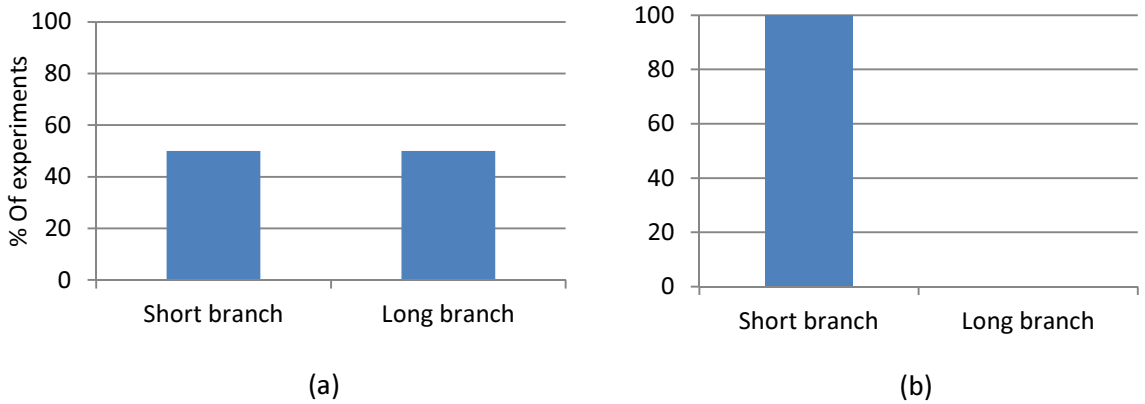

Figure 9 - Results of the uneven bridge experiment with ACO. (a) Constant pheromone drop amount $\Delta\tau^k$ and (b) pheromone drop amount $\Delta\tau^k$ dependent of solution quality.

These results show that when the drop amount is independent of the amount of steps taken (a), the current model parameters (10 ants, pheromone evaporation rate of 0.9) are inadequate to converge to the shortest path in all or at least most of the trials. On the other hand, when the drop amount is made dependent of the steps taken (b) the results are very clear: all trials converge to the shortest path. These results call for more experiments on parameter tuning, see the next chapter.

6. Parameter tuning

When running the ACO model with any network, some model parameters are to be set before starting the simulation. These parameters are: the number of ants, pheromone evaporation rate, setting the pheromone drop amount $\Delta\tau^k$ (in)dependent of the path length, allowing ants to remove loops from their path and setting the minimum and maximum number of iterations per ant. The parameter 'seed offset' allows the model to create unique and reproducible runs.

The first two interesting parameters are the number of ants and making the pheromone drop amount dependent of the path length. In the double bridge experiment (chapter 5) it was established that the artificial ant behavior is not invalidated by basing the pheromone update function on solution quality (path length). This experiment also hinted that the artificial ants might actually rely on this solution quality based pheromone updating, since ten ants weren't able to choose the shortest branch in more than 50% of the runs. This pheromone deposit based on solution quality is also present in real ant species, such as the *Lasius Niger.* It was observed by *Beckers et al. (1993)* that ants of this species returning to the nest from rich food sources, tend to drop more pheromone than ants returning from poorer food sources.

6.1 Pheromone deposits based on solution quality: double bridge experiment

The parameter 'Drop Amount dependent of path length' will influence the choice of $\Delta\tau^k$, the amount of pheromone an ant deposits on an arc it traverses while in backward mode. When this is left unchecked, $\Delta\tau^k$ has the same constant value for all ants. In this case, only the 'differential path length' mechanism will work in favor of the detection of short paths: ants that have constructed a shorter path can deposit pheromones earlier than ants traveling on a longer path. In addition to this mechanism, ants may also deposit an amount of pheromone which is a function of the path length – the shorter the path, the higher the amount of pheromone deposited. So, when checked, the ants will divide their pheromone capacity by the number of steps taken on the way from start to end (i.e., $\Delta\tau^k = 1 \,/\, L^k$). The influence of this parameter will be investigated using the double bridge experiment with the uneven map. The results of two experiments will be discussed:

1. Drop amount **disabled**: Run ACO with different values for the number of ants, while keeping the drop amount $\Delta\tau^k$ at a constant value
2. Drop amount **enabled**: Same as in 1. above, except that the ants deposit an amount $\Delta\tau^k$ which is proportional to the solution quality

All runs were done with 20 different seed offsets creating 20 unique runs for each setting. The outcome has been recorded as either 'short' or 'long' when one of the two branches received at least twice as much traffic (pheromones) as the other branch. If this discrepancy wasn't achieved, the run is recorded as having an even outcome (after at least 500 iterations per ant). The results can be found in Table 1 on the next page.

Table 1 - Percentage of trials in which the ACO model converged to the long path

Drop amount dependent of path length	number of ants	1	2	4	8	16	32
Disabled	% of traffic on the long branch	55	30	15	10	10	10
Enabled		0	0	0	0	0	0

The results of the double bridge experiment in Table 1 show a massive difference between enabling and disabling the drop amount being dependent of the path length. There are several observations to be made from these results. First of all, it shows that increasing the size of the artificial ant colony will also increase the performance of the colony. But this increase will only hold up to a certain point where the colony size doesn't seem to influence the performance anymore, since 8 ants came up with the same result as 32 ants. The second observation is that on this particular map (the uneven double bridge), enabling the drop amount to be dependent of the path length (i.e., $\Delta\tau^k = 1 \,/\, L^k$) massively influences and increases the model performance. Just one ant managed to converge to the shortest path in 100% of the trials. So when drop amount was enabled, increasing the colony size didn't further influence the models' performance. Another observation made is that for some specific seed offset values, the model seemed to lean towards the longer branch, especially when drop amount was disabled. Setting the seed offset to 18 for instance resulted in the colony converging to the long path, regardless of the colony size. This result indicates that in this case, the initial random fluctuations made the probabilities to shift in favor of the longer path and because of the lack of pheromone evaporation in these experiments, the ants became 'stuck' on this longer path. So, to investigate the influence of the pheromone evaporation rate, additional experiments are described in the following section.

6.2 Pheromone evaporation: extended double bridge experiment

In this set of experiments, the influence of pheromone trail evaporation on the convergence behavior of the ACO model is studied. To study this influence, the experiments are run using the extended double bridge map, see Figure 10.

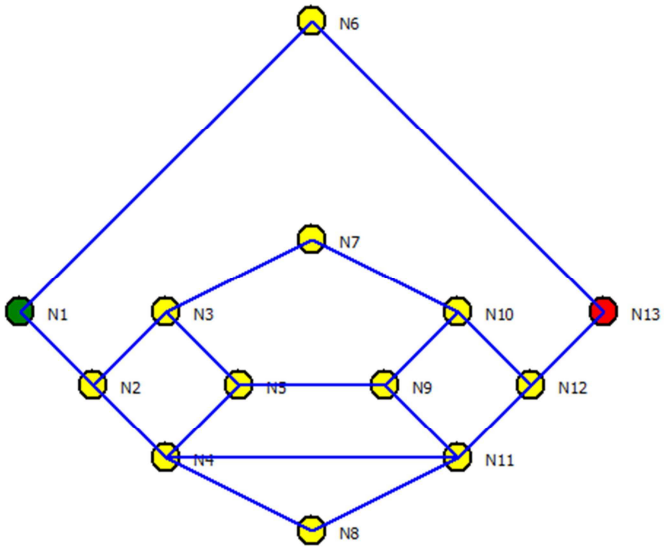

Figure 10 - Extended Double Bridge experiment

An ant starting at the green start node (N1) can choose between the upper and lower part of the graph. The upper branch consists of a single path with length of 894 steps leading directly to the end node. The lower part of the graph consists of a set of paths (of which many paths are shorter than 894 steps) and the ant has many decisions to make before reaching its destination. Therefore, an ant choosing the upper part will always find a path length of 894 steps, while ants choosing the lower part of the graph may find paths shorter than that, but also may enter loops or create other inefficient solutions and thus generate very long paths. The optimal solution can be generated in two different ways and consists of 484 steps. This all means that converging to the minimum cost path is not a trivial task for the algorithm. The ants have to make a number of 'correct choices' and if some of these choices are wrong, the ant generates sub-optimal paths which can be a lot longer than when the upper branch is chosen. There is a trade-off between converging to the use of an 'easy' but sub-optimal path, and searching for the optimal path in a search space were suboptimal paths can easily be generated.

In these experiments the ants deposit an amount of pheromone that is dependent on their path length (i.e., $\Delta\tau^k = 1 \,/\, L^k$) and before depositing this pheromone, the ants remove any loops from their path using the procedure describe in §4.4. Note that while this removes the (largest) loops from the ants' path, it still allows sub-optimal solutions to be created. The experiments are all run with 32 ants and different settings for the evaporation rate $\rho \in \{0, 0.01, 0.1\}$. If $\rho = 0$, no pheromone evaporation takes place, 0.01 will result in low evaporation and $\rho = 0.01$ will result in a rather large evaporation rate, since the evaporation takes place at every update cycle, each 500ms.
To evaluate the behavior and performance of the algorithm, the average path length found by the ants over time will be observed for 200 iterations per ant (6400 iterations in total).

The average path length is recorded after each ten iterations and displayed in the chart below, in Figure 11.

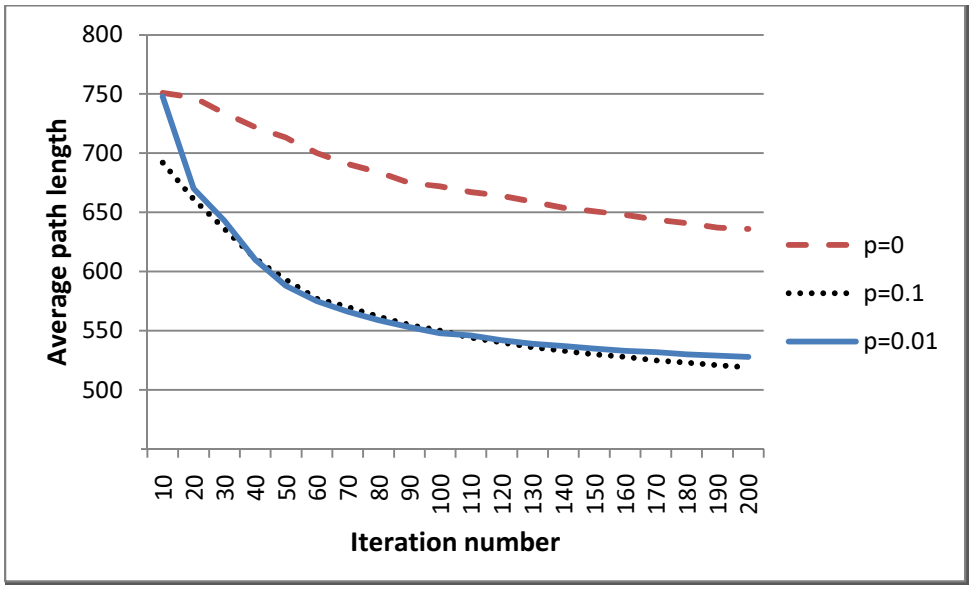

Figure 11 - Average path length vs number of iterations

The behavior observed is representative for the typical model behavior. If no evaporation is used, the best solution for this map isn't found (or at least not in 6400 iterations). With pheromone evaporation enabled, the observed model behavior is significantly different. The average path length decreases a lot faster than without evaporation. The lowest average path length was achieved using low evaporation ($\rho = 0.1$), and at closer examination it turned out that the ants had converged to the shortest possible option of 484 steps in both cases when evaporation was enabled. On top of these results, the following observations were made in these and additional experiments:

- Without pheromone updates based on solution quality (Drop Amount disabled) the performance was much worse. The most observed behavior was that the ants converged to the suboptimal upper branch.
- The pheromone evaporation rate ρ can be a critical parameter. In particular when the evaporation rate is set too high, the model converged to suboptimal paths. The tuning of this parameter can be different for different maps, but in general using evaporation results in better solutions.

7. Conclusion and Recommendations

Considering the assignment description '*This computer assignment consists of studying the ACO theory and developing and building a simulation model which can determine the shortest path between two points in any given (road) network, using ACO.'* It can be concluded that these requirements and the additional 2D-animation requirement have been met. This report describes the ACO theory and the ACO route-finding algorithm that has been developed. The Delphi/Tomas code included in this report on CD is the result of the PDL model described in Chapter three. The models' behavior has been validated against real ant behavior and has been found to be valid as described in Chapter five. It can be used to find short paths in virtually any network that it is provided with.

When the model behavior was tested, several observations have been made. First of all, the model performance increases when the size of the virtual ant colony is increased. This is an expected result, since more 'agents' will provide more solutions and thus allow the search space to be checked more thoroughly at a higher speed. Other parameters like enabling pheromone evaporation and making the pheromone deposits dependent of the solution quality also (greatly) improved the performance in the experiments. Unfortunately, there is no general optimal setting for all of these parameters since they are map dependent. As the problems presented become more complex, the parameter settings become increasingly important to converge to the optimal solution. The experiments conducted in Chapters five and six support the following conclusions:

1. The effect of differential path length, although important, is not enough to allow large optimization problems to be solved effectively
2. Pheromone deposits based on solution quality are important for fast convergence
3. The larger the ant colony size, the better the convergence behavior of the algorithm, although this comes at the cost of longer simulation times
4. Pheromone evaporation is important, especially when trying to solve more complex problems

Considering the amount of calculation power available today, running several experiments with different parameters for each map is a viable option. While keeping an eye on the average score, the best settings for each map could easily be determined.

Some recommendations for further development the ACO model provided in this study are the following. One thing that would be a very useful thing to implement is a map editor in which a user can easily generate maps using a graphical user interface. This map editor could be expanded with a feature that imports data from external (open) sources, such as traffic congestion data or map data from OpenSteetMap. This would allow the ACO model to be used to calculate shortest paths in actual road networks. On top of this, the artificial ant intelligence could be improved and there may be room for overall code optimizations in the current model implementation.

References

Beckers, R., Deneubourg, J.-L., Goss, S. *Modulation of trail laying in the ant Lasius niger and its role in the collective selection of a food source,* Journal of Insect Behavior, 6, 751-759, 1993

Bell, J.E., McMullen, P.R. *Ant colony optimization techniques for the vehicle routing problem,* Advanced Engineering Informatics, 18, 41-48, 2004

Beni, G., Wang, J. *Swarm Intelligence in Cellular Robotic Systems*, Proceedings NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy, June 26–30, 1989

Colorni, A., Dorigo, M., Maniezzo, V. *Distributed Optimization by Ant Colonies*, actes de la première conférence européenne sur la vie artificielle, Paris, Elsevier Publishing, 134-142, 1991

Deneubourg, J.-L. *Personal communication.* Université Libre de Bruxelles, Brussels, 2002

Dorigo, M. *Optimization, Learning and Natural Algorithms*, PhD thesis, Politecnico di Milano, Italy, 1992

Dorigo, M., Gambardella, L.M. *Ant colony system: A cooperative learning approach to the traveling salesman problem,* IEEE Transactions on Evolutionary Computation, 1, 53-66, 1997

Goss, S., Aron, S., Deneubourg, J.-L., Pasteels, J.-M *Self-organized shortcuts in the Argentine ant*, Naturwissenschaften, volume 76, pages 579-581, 1989

Reizzoli, A.E., Montemanni, R., Lucibello, E. Gambardella, L.M., *Ant colony optimization for real-world vehicle routing problems,* Swarm Intell, 1, 135-151, 2007

Wei, G. *New Computational Model from Ant Colony*, Proceedings IEEE International Conference on Granular Computing, 2007

Zeimpekis, V., Tarantalis, C.D., Giaglis, G.M., Minis, I. *Dynamic fleet management – concepts, systems, algorithms & case studies,* Springer, Berlin, 2007