How to program your way out of a paper bag

Frances Buontempo 2013
@fbuontempo
https://github.com/doctorlove/paperbag
overload@accu.org
Why do we ask this?

• “can't even ... out of a paper bag”
  – Couldn't find their way out of a paper bag with a map

• http://www.bored.com/findcliches/stupidpeople.htm
  – A few photons short of a hologram/holodeck
  – Couldn't hit water if he fell out of a boat
  – Doesn't know which side the toast is buttered on

• Angry? Surprised? By lack of ability or knowledge
Fizz Buzz

What is fizz buzz?

1, 2, fizz, 4, buzz, fizz, 7, 8, fizz, buzz, 11, fizz, 13, 14, fizzbuzz, ...

What has fizz buzz got to do with paper bags?
Fizz Buzz Test
24 Dec 2012 – The "Fizz-Buzz test" is an interview question designed to help filter out the ... who can't seem to program their way out of a wet paper bag.

Coding-Out-of-a-Wet-Paper-Bag/FizzBuzz at master · gregburek ...
https://github.com/gregburek/Coding...Paper-Bag/.../FizzBuzz
Coding-Out-of-a-Wet-Paper-Bag - As inspired by Jeff Atwood's post [1] about the inability of programmers to code and solve real world problems, this repo ...

clayton/fizzbuzz · GitHub
https://github.com/clayton/fizzbuzz
fizzbuzz. Contribute to fizzbuzz development by creating an account on GitHub. ... job candidates who can't seem to program their way out of a wet paper bag.

Coding Horror: Why Can't Programmers... Program?
www.codinghorror.com/.../why-cant-programmers-p...
by Jeff Atwood
26 Feb 2007 – An example of a Fizz-Buzz question is the following: ... that we're tired of talking to candidates who can't program their way out of a paper bag.
## Fizz buzz

<table>
<thead>
<tr>
<th>+</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can write code</td>
<td></td>
</tr>
<tr>
<td>Can talk through problem solving</td>
<td></td>
</tr>
<tr>
<td>Can spot edge cases</td>
<td>Sod all to do with paper-bags</td>
</tr>
<tr>
<td>Can demonstrate communication skills</td>
<td></td>
</tr>
<tr>
<td>Might be a good kata</td>
<td></td>
</tr>
</tbody>
</table>
Bring back the paper bag #1

How could we do this programmatically?

Let’s try drag and drop in html

DragAndDrop.html
DandD2.html
#fail

Have we *programmed* our way *out* of a paper bag?

1. No, we ended up in a paper bag
2. No, the user had to move the ant
Bring back the paper bag #2

Time for some ASCII art in C#

..\..\paperbag\expanding\expanding.exe
public void Go()
{
    Setup();
    while (Update())
    {
        Draw();
    }
    Draw();
    Console.WriteLine("\nDone");
}
private void Setup()
{
    _buffer = new char[_width*_width];

    for (int row = 0; row < _width; ++row)
    {
        if (row <= _edge || row > _edge + _bagWidth)
            FillEmptyRow(row);
        else if (row == _edge + _bagWidth)
            FillBagBase(row);
        else
            FillBagRow(row);
    }

    int centre = (_edge + _bagWidth / 2) * _width
                 + _edge + _bagWidth / 2;
    _buffer[centre] = '*';
    Draw();
}
private bool Update()
{
    bool breached = false;
    char[] newBuffer = _buffer.ToArray();
    for (int i = 0; i < _buffer.Length; ++i)
    {
        if (Above(i) == '*' || Below(i) == '*'
            || Left(i) == '*' || Right(i) == '*')
        {
            if (_buffer[i] == '|' || _buffer[i] == '-')
                breached = true;
            newBuffer[i] = '*';
        }
    }
    _buffer = newBuffer;
    return !breached;
}
private void Draw()
{
    int line = 0;
    Console.SetCursorPosition(0, line++);
    for (int i = 0; i < _buffer.Length; ++i)
    {
        Console.Write(_buffer[i]);
        if (i%_width == 0)
        {
            Console.SetCursorPosition(0, line++);
        }
    }
    Thread.Sleep(500);
}
Success?

Have we *programmed* our way *out* of a paper bag?
1. Yes, we ended up out of a paper bag
2. Is changing our size cheating?
3. Is busting out of the side cheating?
4. Would the bag being wet make a difference?
Bring back the paper bag #3

• Let’s have more pictures, and a spot of JavaScript

• First an animation demo
  – canvas_doodle.html (uses canvas_doodle.js)
Animation in JavaScript

function action(x) {
    draw(x);
    x = update(x);
    if (x < 110) {
        id = setTimeout(function() {
            action(x);
        }, 100);
    }
    else {
        stop();
    }
}
function **draw** (x) {
    var canvas = document.getElementById('tutorial');
    if (canvas.getContext) {
        var ctx = canvas.getContext("2d");
        ctx.clearRect(0, 0, canvas.width, canvas.height);
        ctx.fillStyle = "rgb(169, 130, 19)";
        ctx.fillRect (10, 20, 100, 100);
        ctx.fillStyle = "rgba(0, 0, 0, 0.75)";
        ctx.fillRect (10 + x, 40, 25, 25);
    }
}

function **update** (x) {
    return x + 5;
}
Success?

Have we programmed our way out of a paper bag?

1. Yes 😊
2. But it’s a bit boring – it does the same thing every time
3. Let’s introduce some randomness
   – One beasty, several, a cluster, a heuristic
Basic algo

function init() {
    id = setTimeout(action, 100);
}

function action() {
    update();
    draw();
    if (in_bag()) {
        id = setTimeout(action, 100);
    }
}
Update

beast = beasties[index];

var new_x_move = bag_width * 0.2 * (-0.5 + Math.random());
var new_y_move = bag_width * 0.2 * (-0.5 + Math.random());
beast.x += new_x_move;
beast.y += new_y_move;

beasties[index] = beast;
K nearest neighbour

```javascript
function knn(items, index, k) {
    var results =[];
    for (var i=0; i<items.length; i++) {
        if (i !==index) {
            var neighbour = items[i];
            var distance =
                Math.sqrt(neighbour.x*neighbour.x
                + neighbour.y*neighbour.y);
            results.push( new distance_index(distance, i) );
        }
    }
    results.sort( function(a,b) {
        return a.distance - b.distance;
    });
    var top_k = Math.min(k, results.length);
    return results.slice(0, top_k);
}
```
Beasties

• **paperbag.html**
  – one random

• **paperbag_many.html**
  – all random

• **paperbag_many_follow.html**
  – k nearest neighbours (knn)

• **paperbag_many_follow_up.html**
  – heuristic = “go up”
Success?

Have we *programmed* our way *out* of a paper bag?

1. Yes, we ended up out of a paper bag
2. Yes, the program moved the “ants”
3. No, knn was a disaster, unsurprisingly

But, can they get better at it?
   We have a heuristic – go up

Time for some machine learning
Will this help us program our way out of a paper bag?
Overview

- Expert systems
- Statistical methods
- Artificial neural networks
- Inductive data mining
- “randomness”
Expert systems

• Human expert knowledge can be used
• Knowledge is transparent and causal
• New data cannot be used
• The output is often qualitative
• Different experts will often provide differing rules, so the knowledge is subjective
Example expert systems

• Dendral and MetaDendral

• DEREK by Lhasa
  – https://www.lhasalimited.org/derek_nexus/

• FxCop?

• Lint?

• Pex?
“An early motivation for our work was to explore the power of existing AI methods, such as heuristic search, for reasoning in difficult scientific problems. Another concern has been to exploit the AI methodology to understand better some fundamental questions in the philosophy of science, for example the processes by which explanatory hypotheses are discovered or judged adequate”

‘Dendral and Meta-Dendral: Their applications dimension’ Buchanan and Feigenbaum, 1978?

Statistical methods

• Data driven methods, so are more objective than expert systems.
• Quantitative predictions can be generated.
• The models are usually linear and sometimes black-box.
• Human knowledge cannot be used
**Regression**

![Graph showing a linear relationship between log(LC) and logP.]  

**EPA toxicity QSAR “ECOSAR” programme**  
- **N=2, r² = 1.0.**
Artificial neural networks

• Data driven methods.
• Quantitative predictions can be generated.
• The model is non-linear, and easy to set up and train.
• The model is largely a black-box.
• Human knowledge cannot be used.
• They cannot handle a large number of inputs e.g. training cases \( \leq \) input variables.
Feed-forward neural network

\[ y = f(w_0 + \sum w_i x_i) \]

Initial random weights
Choose f, say sigmoid function
Change weights in each epoch to minimise difference between predicted y and actual value
Sigmoid curve

\[ S(t) = \frac{1}{1 + e^{-t}} \]
Inductive data mining

• Data and human knowledge can be used simultaneously.
• The model automatically generates transparent and “causal” rules or trees.
• It can handle many inputs and noise.
• Results can be inaccurate, complicated, or not generalise well.
Top-down tree induction (starting with all the data)

- Entropy-based splitting
  - ID3
  - C4.5
- Statistical-based splitting
  - CHAID
  - CASE
  - CART
  - FIRM
  - LMTD
  - SCAM

Adaptive fuzzy partitioning
- Hand-crafted by trial and error
- Using genetic algorithms
- NEFPROX, etc.

Fuzzy inference systems
- Mamdani
- Takagi-Sugeno
- Grid-partition
- Subtractive clustering
- ANFIS

Others
- Bottom-up approaches (AQ family)
- Version Spaces
- ILP; Golem, etc.
- FOIL
- Decision forests
- Rules extracted from trained NN
Randomness

- E.g. Genetic algorithms, Monte-Carlo simulation, swarm “optimisation”
- Usually quantitative
- Data-driven
- Might need an a priori model or heuristic, and values for parameters
## Types of ML

<table>
<thead>
<tr>
<th>Name</th>
<th>Inputs</th>
<th>Learning</th>
<th>instance-based (lazy) learning</th>
<th>Randomness</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>regression</td>
<td>numeric</td>
<td>supervised</td>
<td>false</td>
<td>no</td>
<td>Numeric</td>
</tr>
<tr>
<td>k-th nearest neighbours</td>
<td>numeric</td>
<td>unsupervised</td>
<td>true</td>
<td>no</td>
<td>data points</td>
</tr>
<tr>
<td>kohonen neural network</td>
<td>numeric</td>
<td>unsupervised</td>
<td>false</td>
<td>no</td>
<td>Clusters</td>
</tr>
<tr>
<td>feedforward nn</td>
<td>numeric</td>
<td>supervised</td>
<td>false</td>
<td>no</td>
<td>Numeric</td>
</tr>
<tr>
<td>recurrent nn (eg Hopfield)</td>
<td>binary</td>
<td>reinforcement</td>
<td>true</td>
<td>no</td>
<td>State</td>
</tr>
<tr>
<td>C4.5 or See5</td>
<td>categoric</td>
<td>supervised</td>
<td>false</td>
<td>no</td>
<td>decision tree</td>
</tr>
<tr>
<td>CART</td>
<td>any</td>
<td>supervised</td>
<td>false</td>
<td>no</td>
<td>Tree</td>
</tr>
<tr>
<td>genetic algorithm</td>
<td>any</td>
<td>unsupervised</td>
<td>false</td>
<td>yes</td>
<td>Solution</td>
</tr>
<tr>
<td>dendral</td>
<td>numeric</td>
<td>hypothesis</td>
<td>formation</td>
<td>no</td>
<td>expert system (possible chemical structures)</td>
</tr>
<tr>
<td>ACO</td>
<td>spatial</td>
<td>unsupervised</td>
<td>false</td>
<td>yes</td>
<td>best 'path'</td>
</tr>
</tbody>
</table>
ML as tree using See5

• Which techniques are suitable for programming your way out of a paper bag?
• Can we make a decision tree of ML techniques?
• No – it’s supervised
• i.e. needs a target
How to make a tree

• Training data (rows)
• Inputs (columns: x values)
• Target output
• Choose an input to split on
  – Entropy
    • Info content = -∑frequency(class(j))/|S| * log₂(class(j)/|S|)
    • Compare info content set for potential splits
    • Which attributes give most information gain?
• Split the training data down each node
• Repeat
• Test
### Example (golf)

<table>
<thead>
<tr>
<th>Inputs (attributes)</th>
<th>Output (target)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Outlook</strong></td>
<td><strong>Temp</strong></td>
</tr>
<tr>
<td>sunny</td>
<td>85</td>
</tr>
<tr>
<td>sunny</td>
<td>80</td>
</tr>
<tr>
<td>overcast</td>
<td>83</td>
</tr>
<tr>
<td>rain</td>
<td>70</td>
</tr>
<tr>
<td>rain</td>
<td>68</td>
</tr>
<tr>
<td>rain</td>
<td>65</td>
</tr>
<tr>
<td>overcast</td>
<td>64</td>
</tr>
<tr>
<td>sunny</td>
<td>72</td>
</tr>
<tr>
<td>sunny</td>
<td>69</td>
</tr>
<tr>
<td>rain</td>
<td>75</td>
</tr>
<tr>
<td>sunny</td>
<td>75</td>
</tr>
<tr>
<td>overcast</td>
<td>72</td>
</tr>
<tr>
<td>overcast</td>
<td>81</td>
</tr>
<tr>
<td>rain</td>
<td>71</td>
</tr>
</tbody>
</table>
By hand

• 14 training cases

• Play v. Don’t Play: Info(9/14, 5/14)
  \[(9/14 \times \log_2(9/14) + 5/14 \times \log_2(5/14)) = 0.94\]

• Outlook (sunny, 5), (overcast, 4), (rain, 5)
  – Always play when it’s overcast

• Try outlook
  \[(5/14 \times \text{Info(sunny)} + 4/14 \times \text{Info(overcast)} + 5/14 \times \text{info(rain)})) = 0.694\]
  Info gain = 0.94 − 0.694 = 0.246

• Try windy
  \[(8/14 \times \text{Info(not windy)} + 6/14 \times \text{info(windy)}) = 0.892\]
  Info gain = 0.94 − 0.892 = 0.048
Decision Tree

Outlook?

Sunny

Humidity?

≤75
Play (2)

> 75
Don’t Play (3)

Overcast

Play (4)

Rainy?

Rainy

True
Don’t Play (2)

False
Play (3)
# Target for See5

<table>
<thead>
<tr>
<th>Name</th>
<th>Inputs</th>
<th>Learning</th>
<th>instance-based (lazy) learning</th>
<th>Randomness</th>
<th>Output</th>
<th>Out of paper bag?</th>
</tr>
</thead>
<tbody>
<tr>
<td>regression</td>
<td>numeric</td>
<td>supervised</td>
<td>false</td>
<td>no</td>
<td>Numeric</td>
<td>no</td>
</tr>
<tr>
<td>k-th nearest neighbours</td>
<td>numeric</td>
<td>unsupervised</td>
<td>true</td>
<td>no</td>
<td>data points</td>
<td>no</td>
</tr>
<tr>
<td>kohonen neural network</td>
<td>numeric</td>
<td>unsupervised</td>
<td>false</td>
<td>no</td>
<td>Clusters</td>
<td>no</td>
</tr>
<tr>
<td>feedforward nn</td>
<td>numeric</td>
<td>supervised</td>
<td>false</td>
<td>no</td>
<td>Numeric</td>
<td>no</td>
</tr>
<tr>
<td>recurrent nn (eg Hopfield)</td>
<td>binary</td>
<td>reinforcement</td>
<td>true</td>
<td>no</td>
<td>State</td>
<td>no</td>
</tr>
<tr>
<td>C4.5 or See5</td>
<td>categoric</td>
<td>supervised</td>
<td>false</td>
<td>no</td>
<td>decision tree</td>
<td>no</td>
</tr>
<tr>
<td>CART</td>
<td>any</td>
<td>supervised</td>
<td>false</td>
<td>no</td>
<td>Tree</td>
<td>no</td>
</tr>
<tr>
<td>genetic algorithm</td>
<td>any</td>
<td>unsupervised</td>
<td>false</td>
<td>yes</td>
<td>Solution</td>
<td>yes</td>
</tr>
<tr>
<td>dendral</td>
<td>numeric</td>
<td>hypothesis formation</td>
<td>false</td>
<td>no</td>
<td>expert system</td>
<td>no</td>
</tr>
<tr>
<td>ACO</td>
<td>spatial</td>
<td>unsupervised</td>
<td>false</td>
<td>yes</td>
<td>best 'path'</td>
<td>yes</td>
</tr>
</tbody>
</table>
Which ML?

Decision tree:

- randomness = yes: yes (2)
- randomness = no: no (8)

Randomness?

Suitable

- GA
- ACO

No suitable
So, why the ant before?
Aside

Machine learning and data mining frequently requires some form of pre-processing
Ant algorithm problem optimization colony migration edit ants solutions
Travelling salesperson problem

Start at A, chose shortest each time: \( A \rightarrow B \rightarrow C \rightarrow D = 1 + 1 + 30 = 32 \)

Start at A, think: \( A \rightarrow D \rightarrow B \rightarrow C = 3 + 2 + 1 = 6 \)

Start at C, think: \( C \rightarrow B \rightarrow A \rightarrow D = 1 + 1 + 3 = 5 \)
ACO for TSP

- Move some ants randomly, remembering the trail
- Lay pheromones along each trail
- For each epoch
  - Move the ants again, guided by the pheromones
    - E.g. roulette wheel selection
  - Update the pheromones
    - Evaporate a bit (subtract)
    - Emphasis on the better paths (add)
- Report the best path
ACO in C#

```csharp
int numCities = 60; int numAnts = 4; int maxTime = 1000;
int[][] dists = MakeGraphDistances(numCities);
int[][] ants = InitAnts(numAnts, numCities);
int[] bestTrail = BestTrail(ants, dists);
double bestLength = Length(bestTrail, dists);
double[][] pheromones = InitPheromones(numCities);
int time = 0;

while (time < maxTime) {
    UpdateAnts(ants, pheromones, dists);
    UpdatePheromones(pheromones, ants, dists);
    int[] currBestTrail = BestTrail(ants, dists);
    double currBestLength = Length(currBestTrail, dists);
    if (currBestLength < bestLength) {
        bestLength = currBestLength;
        bestTrail = currBestTrail;
    }
    ++time;
}
```


James McCaffery
Update ants

// For ant k at cityX
double[] probs = MoveProbs(k, cityX, visited,
    pheromones, dists);

// roulette wheel
double[] cumul = new double[probs.Length + 1];
for (int i = 0; i < probs.Length; ++i)
  cumul[i + 1] = cumul[i] + probs[i];

double p = random.NextDouble();

for (int i = 0; i < cumul.Length - 1; ++i)
  if (p >= cumul[i] && p < cumul[i + 1])
    return i;
Roulette wheels
double[] taueta = new double[numCities]; double sum = 0.0;
for (int i = 0; i < taueta.Length; ++i) {
    if ((i == cityX) || (visited[i] == true))
        // Prob of moving to self is zero
        // Prob of moving to a visited node is zero
        taueta[i] = 0.0;
    else {
        taueta[i] = Math.Pow(pheromones[cityX][i], alpha) * 
                    Math.Pow((1.0 / Distance(cityX, i, dists)), beta);
        // cap or floor if too big or too small
    } 
    sum += taueta[i];
}

// Normalise:  probs[i] = taueta[i] / sum;
Update Pheromones

double length = Length(ants[k], dists);
// length of ant k trail
double decrease = (1.0 - \rho) * pheromones[i][j];
double increase = 0.0;
if (EdgeInTrail(i, j, ants[k]) == true)
    increase = (Q / length);

pheromones[i][j] = decrease + increase;
// matrix of edges from city i to city j
Maths

• Probability
\[ p(K\text{-th ant moves from city } x \text{ to city } y) = \frac{\tau_\alpha^{xy} \eta^\beta_{xy}}{\sum \tau_\alpha^{xy} \eta^\beta_{xy}} \]

where \( \eta \) is attractiveness of move e.g. \( \frac{1}{\text{distance}(x,y)} \)

• Pheromone
\[ \tau_{xy} = (1 - \rho)\tau_{xy} + \sum_k \Delta\tau^k_{xy} \]

with \( \Delta\tau^k_{xy} = \begin{cases} \frac{Q}{L_k} & \text{if ant } k \text{ uses edge } xy \\ 0 & \text{otherwise} \end{cases} \)
ACO for TSP results

• No pictures 😞

• ACO.exe

• Change the code for our pathological case
  static int[][][] MakeGraphDistances(int numCities)
  {
    int[][][] dists = new int[numCities][][];
    dists[0] = new int[] { 0, 1, 20, 3 };
    dists[1] = new int[] { 1, 0, 1, 2 };
    dists[2] = new int[] { 20, 1, 0, 30 };
    dists[3] = new int[] { 3, 2, 30, 0 };
    return dists;
  }

• ACOPathological.exe
**Graphviz**

Best trail found:
2 1 0 3

*Change to:*
digraph G {2->1; 1->0; 0->3;}

*Run it through dot:*
> dot.exe -Tpng
digraph G {2->1; 1->0; 0->3;}
> TSPACO.png
Observations

• Cheating! Just reports the best path ever
  
  ```csharp
  int[] currBestTrail = BestTrail(ants, dists);
  double currBestLength = Length(currBestTrail, dists);
  if (currBestLength < bestLength) {
    bestLength = currBestLength;
    bestTrail = currBestTrail;
    Console.WriteLine("New best length of " +
                      bestLength.ToString("F1");
  }
  ```

• Do the worst ones get any better?

• Would this work for escaping a paper bag?
  – Let’s make the ants move nearby rather than jumping anywhere
ACO for escaping a paper bag

• Pictures 😊

• Change the distance metric, $\eta$?
  – We have a heuristic – “go up”, so use $y$

• Our problem is really continuous: start inside the bag and stop at the top

• Why don’t the ants update the pheromones as they move? (Another day...)
ACO in JavaScript

Pseudo-algorithm
Let n ants start in the bottom of the bag
In each epoch
All ants step up/down/left/right
  guided by pheromones
  ‘til they come out the top
Lay pheromones
Draw best trail
Shall we have some unit tests?

• “All ants step up/down/left/right”
  – And should not burst out of the bottom of the bag
Passing 18 specs

next_pos
    should not be below bag
    should not be beside bag
    should not return to a previous point

random_trail
    should start at bottom of bag
    should end above bag

make_trails
    should return same number of trails as ants

find_best
    should find the only trail if there is just one
    should find the shortest trail when it is first
    should find the shortest trail when it is last

contains
    should not contain a item when it is empty
    should contain a item when it is the only item

add_new_pheromones
    should contain each point in a new trail

nearest_pheromones
    should find the only item if just one exists
    should report -1 if none are near enough

cumulative_probability
    should give the sum of tau eta when there is one point
    should give the sum of tau eta when there is a non-zero weight and the rest are zero

roulette_wheel_choice
    should return a position
    should go to best pheromone point if all other points have zero probability
describe("next_pos", function() {

  it("should not be below bag", function() {
    var width = 4;
    for (var i = 0; i < width; ++i) {
      var pos = { x: i, y :0 };
      var next = next_pos(width, pos, []);
      expect(next.y >= 0).toBe(true);
    }
  });

});
function update(pheromones, height, width) {
    var trail, i;
    var updated = evapourate(pheromones);

    for( i = 0; i < trails.length; ++i) {
        trail = trails[i];
        pheromones =
            add_new_pheromones(height, pheromones, trail);
    }

    trails = new_trails(pheromones, height, width, ants);
}
Recap

\[ \tau_{xy} = (1 - \rho)\tau_{xy} + \sum_k \Delta\tau^k_{xy} \]

\[ = (\text{evapourate old}) + (\text{lay new}) \]

with \( \Delta\tau^k_{xy} = \begin{cases} \frac{Q}{L_k} & \text{if ant } k \text{ uses edge } xy \\ 0 & \text{otherwise} \end{cases} \)
Pheromone evapouration

```javascript
function evapourate(pheromones) {
    var evapouration = 0.75;
    var updated = [], new_pos;

    for(i = 0; i < pheromones.length; ++i) {
        new_pos = {x: pheromones[i].x, y: pheromones[i].y,
                   weight: evapouration * pheromones[i].weight};
        updated.push( new_pos );
    }
    return updated;
}
```
function add_new_pheromones(height, pheromones, trail) {
    var i, pos, new_pos;
    var Q = 2.0 * height;
    var L = Q/trail.length;

    for (i = 0; i < trail.length; ++i) {
        pos = trail[i];
        index = nearest_pheromone(pheromones, pos);
        if ( index !== -1 ) {
            pheromones[index] = {x: pheromones[index].x,
                                y: pheromones[index].y, weight: pheromones[index].weight + L};
        } else {
            pheromones.push( {x: pos.x, y: pos.y, weight: L});
        }
    }
    return pheromones;
}
Make new trails

// For each ant, with var trails = [];
// trails.push
//   (pheromone_trail(width, height, pheromones));

function pheromone_trail(height, width, pheromones) {
  var trail = [], pos = start_pos(width);
  trail.push(pos);

  while (pos.y < height) {
    pos = roulette_wheel_choice(width, pos, trail, pheromones);
    trail.push(pos);
  }
  return trail;
}
function roulette_wheel_choice(width, pos, trail, pheromones) {
    var p=0;
    var possible = allowed_positions(width, pos, trail);
    var cumulative = cumulative_probability(possible, pheromones);
    var total = cumulative[cumulative.length-1];
    if (total === 0) {
        p = Math.floor(Math.random() * possible.length);
        return possible[p];
    }
    p = Math.random() * total;
    for (i = 0; i < cumulative.length - 1; ++i) {
        if (p >= cumulative[i] && p <= cumulative[i+1]) {
            //the first place where it is in range, with 1 is in [1,1]
            return possible[i];
        }
    }
}

Roulette wheel

function roulette_wheel_choice(width, pos, trail, pheromones) {
    var p=0;
    var possible = allowed_positions(width, pos, trail);
    var cumulative = cumulative_probability(possible, pheromones);
    var total = cumulative[cumulative.length-1];
    if (total === 0) {
        p = Math.floor(Math.random() * possible.length);
        return possible[p];
    }
    p = Math.random() * total;
    for (i = 0; i < cumulative.length - 1; ++i) {
        if (p >= cumulative[i] && p <= cumulative[i+1]) {
            //the first place where it is in range, with 1 is in [1,1]
            return possible[i];
        }
    }
}
allowed_positions

function allowed_positions(width, pos, trail) {
    var possible = possible_positions(width, pos);
    var allowed = [];
    var i = 0;
    for (i = 0; i < possible.length; ++i) {
        if (!contains(trail, possible[i])) {
            allowed.push(possible[i]);
        }
    }
    if (allowed.length === 0) {
        allowed = possible;
    }
    return allowed;
}
Recap

tau eta is \( \frac{\tau^\alpha_{xy} \eta^\beta_{xy}}{\sum \tau^\alpha_{xy} \eta^\beta_{xy}} \)

where

- \( \tau \) is the pheromone
- \( \eta \) is attractiveness of move e.g. \( \frac{1}{\text{distance}(x,y)} \)
- \( \alpha, \beta \) are parameters
  - (numbers picked out of the air and experimented with)
cumulative_probability

function cumulative_probability(possible, pheromones){
  var total = 0.0, index;
  var cumulative = [total];
  for (i = 0; i < possible.length; ++i) {
    index = nearest_pheromone(pheromones, possible[i]);
    if (index !== -1) {
      total = total + taueta(pheromones[index].weight, pheromones[index].y);
    }
    cumulative.push(total);
  }
  return cumulative;
  //not in [0, 1] but choosing random(0, total) is same as dividing by total here
}
function taueta(weight, y) {
    var alpha = 1.0;
    var beta = 3.0;
    return Math.pow(weight, alpha) + Math.pow(y, beta);
}
Success?

Learning: lighter dots worst, darker dots best

Finished: worst tending to be a bit closer to best
What we have learnt so far

• Fizz buzz doesn’t have a paper bag
• A simple JavaScript animation can involve a paper bag we program our way out of
• Machine learning covers many ideas, some of which are suitable for the problem at hand
• We saw an implementation of an ant colony optimisation
  – We could try other pheromone updating schemes
What else can we do?

• When faced with a difficult problem
  – make a cup of tea
  – break your problems into parts and handle them one part at a time
  – remember, “All the greatest and most important problems of life are fundamentally insolvable. They can never be solved, but only outgrown.” Carl Jung
  – try to transform it into a known problem and solve that instead
“The mere formulation of a problem is far more essential than its solution, which may be merely a matter of mathematical or experimental skills. To raise new questions, new possibilities, *to regard old problems from a new angle* requires *creative imagination* and marks real advances in science.”

Albert Einstein
Transform it into a known problem and solve that instead
Mazes

- Let’s be lazy and assume we already have a maze

```python
Z = numpy.zeros(shape, dtype=numpy.int32)
# for rand x, y
Z[y, x] = 1  # make a wall
for j in range(complexity):
    neighbours = []
    if x > 1:
        neighbours.append((y, x - 2))
    if x < shape[1] - 2:
        neighbours.append((y, x + 2))
    if y > 1:
        neighbours.append((y - 2, x))
    if y < shape[0] - 2:
        neighbours.append((y + 2, x))
    if len(neighbours):
        y_, x_ = neighbours[rand(0, len(neighbours) - 1)]
        if Z[y_, x_] == 0:
            # if it's not a wall
            Z[y_, x_] = 1
            Z[y_ + (y - y_) // 2, x_ + (x - x_) // 2] = 1
            x, y = x_, y_
```
Left-wall follower

Need to track direction and find next position that isn’t a wall
#find start and append next_move til we come out the end

facing, row, col = next_move(facing, row, col, maze)
path.append( (row, col) )

... 

def next_move(facing, row, col, maze):
    l = potential_moves(facing)
    index = 0
    rows = maze.shape[0]
cols = maze.shape[1]
    while index < len(l):
        if l[index] == 'U' and (row - 1 >= 0) and (maze[row - 1, col] == 0):
            return 'U', row - 1, col
        elif l[index] == 'R' and (col + 1 < cols) and (maze[row, col + 1] == 0):
            return 'R', row, col + 1
        elif l[index] == 'D' and (row + 1 < rows) and (maze[row + 1, col] == 0):
            return 'D', row + 1, col
        elif l[index] == 'L' and (col - 1 >= 0) and (maze[row, col - 1] == 0):
            return 'L', row, col - 1
        index = index + 1
Other ideas

• Cellular automata
  – Langton’s ant ([http://en.wikipedia.org/wiki/Langton%27s_ant](http://en.wikipedia.org/wiki/Langton%27s_ant))
    Squares on a plane are coloured variously either black or white. One square is the "ant". The ant moves according to these rules:
    • At a white square, turn 90° right, flip the colour of the square, move forward one unit
    • At a black square, turn 90° left, flip the colour of the square, move forward one unit
    Cells are alive or dead. At each time step
    • Any live cell with fewer than two live neighbours dies, as if caused by under-population.
    • Any live cell with two or three live neighbours lives on to the next generation.
    • Any live cell with more than three live neighbours dies, as if by overcrowding.
    • Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.
Other ideas...

• Other swarm optimisation algorithms
• Monte-Carlo simulations
• Anything that moves in a plane, or space
Is this a software problem?

- All our paper bags are 2D
- Can they move?
- What colour are they?
- Does it make a difference if the bag is wet?
- Lots of maze solving projects on the internet involve robots...
Is it a software problem?
Is this a hardware problem?
(or am I a hardware engineer?)

• My soldering was rubbish
• Hexabug often gets stuck in the bag
Other hardware ideas

• Raspberry Pi

• Learn to build robots
Other ideas

http://upload.wikimedia.org/wikipedia/commons/thumb/c/c7/Explosions.jpg/800px-Explosions.jpg
Conclusion

• Fizz buzz doesn’t involve paper bags
• Drag and drop wasn’t enough fun
• It’s a hard problem: “difficult scientific problem”
• Machine learning provides many ideas
• Is it actually a software problem?
• Can you program your way out of a paper bag?
• Email your attempts to overload@accu.org
This is to certify that Frances Buontempo can program her way out of a paper bag.