AscendToLocalMax.m

```
function [SolutionValue, bestSolution, solutionsOverTime] =
AscendToLocalMax(x, NoPots, PotAl, PotFe, PotsPerCrucible, NoCrucibles,
NoQualities, QualityMinAl, QualityMaxFe, QualityValue)
numSolutions = 1;
solutionsOverTime = [];
maxSpread = 11;
while true
    SolutionValue = CalcSolutionValue(x, PotAl , PotFe, PotsPerCrucible,
NoCrucibles, NoQualities, QualityMinAl, QualityMaxFe, QualityValue);
    bestSolution = x;
    for i=1:NoPots;
        for j=i+1:NoPots;
            y = x;
            iCrucible = fix(i/PotsPerCrucible-1) + 1;
            iIndex = rem(i, PotsPerCrucible) + 1;
            jCrucible = fix(j/PotsPerCrucible-1) + 1;
            jIndex = rem(j, PotsPerCrucible) + 1;

            y(jCrucible,jIndex) = x(iCrucible, iIndex);
            y(iCrucible, iIndex) = x(jCrucible, jIndex);

            iOld = CalcCrucibleValueForIndex(x, maxSpread, iCrucible,
PotAl , PotFe, NoQualities, QualityMinAl, QualityMaxFe, QualityValue);
            iNew = CalcCrucibleValueForIndex(y, maxSpread, iCrucible,
PotAl , PotFe, NoQualities, QualityMinAl, QualityMaxFe, QualityValue);

            jOld = CalcCrucibleValueForIndex(x, maxSpread, jCrucible,
PotAl , PotFe, NoQualities, QualityMinAl, QualityMaxFe, QualityValue);
            jNew = CalcCrucibleValueForIndex(y, maxSpread, jCrucible,
PotAl , PotFe, NoQualities, QualityMinAl, QualityMaxFe, QualityValue);

            increaseInValue = iNew + jNew - iOld - jOld;
            if increaseInValue >= 0
                x = y;
            end
            solutionsOverTime(numSolutions) = SolutionValue +
increaseInValue;
            numSolutions = numSolutions + 1;
        end
    end
    newPrice = CalcSolutionValue(x, PotAl , PotFe, PotsPerCrucible,
NoCrucibles, NoQualities, QualityMinAl, QualityMaxFe, QualityValue);
    if newPrice > SolutionValue
        SolutionValue = newPrice;
        bestSolution = x;
    else
        break
    end
end
end
```

CalcCrucibleValue.m

```
function Value = CalcCrucibleValue(CrucibleAl, CrucibleFe, NoQualities,
QualityMinAl, QualityMaxFe, QualityValue)
% Lookup the value of a product that has the given Al and Fe percents
% Uses a tolerance of 0.00001 to avoid rounding errors
% Find the last quality range (assuming they are sorted worst to best)
that we fit into
  Value = 0;
  for i = NoQualities:-1:1
    if CrucibleAl>=QualityMinAl(i)-0.00001
      if CrucibleFe <= QualityMaxFe(i)+0.00001
         Value = QualityValue(i);
         break;
      end
    end
  end
end
```

CalcCrucibleValueForIndex.m

```
function CrucibleValue = CalcCrucibleValueForIndex(x, maxSpread,
crucible, PotAl , PotFe, NoQualities, QualityMinAl, QualityMaxFe,
QualityValue)
CrucibleAl  = FastMean(PotAl(x(crucible,:)));
CrucibleFe  = FastMean(PotFe(x(crucible,:)));
CrucibleValue =
CalcCrucibleValue(CrucibleAl,CrucibleFe,NoQualities,QualityMinAl,Quality
MaxFe,QualityValue);
spread = max(x(crucible,:)) - min(x(crucible,:));
if spread >= maxSpread
    CrucibleValue = CrucibleValue * (maxSpread / spread);
end
end
```

CalcSolutionValue.m

```
function [SolutionValue, CrucibleValue] = CalcSolutionValue(x, PotAl ,
PotFe, PotsPerCrucible, NoCrucibles, NoQualities, QualityMinAl,
QualityMaxFe, QualityValue)
  for c = 1:NoCrucibles
    CrucibleAl = mean(PotAl(x(c,:)));
    CrucibleFe = mean(PotFe(x(c,:)));
    CrucibleValue(c)=
CalcCrucibleValue(CrucibleAl,CrucibleFe,NoQualities,QualityMinAl,Quality
MaxFe,QualityValue);
  end

  SolutionValue = sum(CrucibleValue);

end
```

CalcVal.m

```
function Value = CalcCrucibleValue(crucible)
% Lookup the value of a product that has the given Al and Fe percents
% Uses a tolerance of 0.00001 to avoid rounding errors
% Find the last quality range (assuming they are sorted worst to best)
that we fit into
  Value = 0;
  for i = NoQualities:-1:1
    if CrucibleAl>=QualityMinAl(i)-0.00001
      if CrucibleFe <= QualityMaxFe(i)+0.00001
         Value = QualityValue(i);
         break;
      end
    end
  end
end
```

FastMean.m

```
function mean = FastMean(data)
mean = sum(data) / size(data, 2);
end
```

GenRandom.m

```
function x = GenRandom(NoPots, NoCrucibles, PotsPerCrucible)
  solution = randperm(NoPots);
  x = reshape(solution,[NoCrucibles,PotsPerCrucible]);
end
```

GenStart.m

```
function x = GenStart(NoPots, NoCrucibles, PotsPerCrucible)
% Generate a (boring) starting solution
  Pot = 1;
  for c=1:NoCrucibles;
    for i = 1:PotsPerCrucible;
      x(c,i) = Pot;
      Pot = Pot + 1;
    end
  end
end
```

InitProb.m

```
function [PotAl, PotFe] = InitProb
% Set up particular values of today's pots
  PotAl=[99.79, 99.23, 99.64, 99.88, 99.55, 99.87, 99.55, 99.19, 99.76,
99.70, ...
         99.26, 99.60, 99.05, 99.49, 99.69, 99.48, 99.60, 99.89, 99.39,
99.48, ...
         99.77, 99.57, 99.48, 99.85, 99.09, 99.64, 99.71, 99.59, 99.14,
99.87, ...
         99.38, 99.56, 99.32, 99.55, 99.61, 99.57, 99.75, 99.63, 99.17,
99.97, ...
         99.74, 99.49, 99.75, 99.40, 99.72, 99.95, 99.31, 99.55, 99.29,
99.09, 99.20];
  PotFe=[0.01, 0.68, 0.25, 0.61, 0.13, 0.77, 0.48, 0.18, 0.66, 0.43, ...
         0.13, 0.87, 0.96, 0.47, 0.51, 0.73, 0.04, 0.76, 0.89, 0.90, ...
         0.96, 0.73, 0.88, 0.43, 0.60, 0.37, 0.51, 0.26, 0.30, 0.46, ...
         0.21, 0.77, 0.43, 0.52, 0.63, 0.76, 0.02, 0.75, 0.90, 0.53, ...
         0.14, 0.10, 0.31, 0.20, 0.45, 0.67, 0.56, 0.24, 0.72, 0.56,
0.01];
end
```

InitQual.m

```
function [NoCrucibles,NoPots,PotsPerCrucible,NoQualities, ...
          QualityMinAl, QualityMaxFe, QualityValue] = Init()
% Init all the main constants with the Comalco specific data
  NoCrucibles = 17;
  NoPots = 51;
  PotsPerCrucible = 3;

  NoQualities = 11;

  QualityMinAl = [95.00, 99.10, 99.10, 99.20, 99.25, 99.35, 99.50,
99.65, 99.75, 99.85, 99.90];
  QualityMaxFe = [ 5.00,  0.81,  0.79,  0.79,  0.76,  0.72,  0.53,
 0.50,  0.46,  0.33,  0.30];
  QualityValue = [   10, 21.25, 26.95, 36.25, 41.53, 44.53, 48.71,
52.44, 57.35, 68.21, 72.56];

end
```

Main.m

```
function main()
% This is where it all happens

% Initialise the data
  [NoCrucibles,NoPots,PotsPerCrucible,NoQualities, ...
          QualityMinAl, QualityMaxFe, QualityValue] = InitQual;
  [PotAl, PotFe] = InitProb;

  %TestAscendToLocalMax(PotAl, PotFe,
NoCrucibles,NoPots,PotsPerCrucible,NoQualities, QualityMinAl,
QualityMaxFe, QualityValue)
  DoRepeatedAscents(200, NoPots, PotAl, PotFe, PotsPerCrucible,
NoCrucibles, NoQualities, QualityMinAl, QualityMaxFe, QualityValue);

end
```

TestAscendToLocalMax.m

```
function [] = TestAscendToLocalMax(PotAl, PotFe,
NoCrucibles,NoPots,PotsPerCrucible,NoQualities, QualityMinAl,
QualityMaxFe, QualityValue)
x = GenStart(NoPots, NoCrucibles, PotsPerCrucible);
% Do the local search here...
[SolutionValue, solution, solutionsOverTime] = AscendToLocalMax(x,
NoPots, PotAl, PotFe, PotsPerCrucible, NoCrucibles, NoQualities,
QualityMinAl, QualityMaxFe, QualityValue);
plot(solutionsOverTime);

disp('---------------------------')
% View the solution (double checking its objective function)
ViewSoln(solution, PotAl, PotFe, NoCrucibles, NoQualities, QualityMinAl,
QualityMaxFe, QualityValue);
end
```

ViewSoln.m

```
function totalValue = ViewSoln(x, PotAl, PotFe, NoCrucibles, ...
                  NoQualities, QualityMinAl, QualityMaxFe, QualityValue)
% Print current solution in x to the screen.
% The solution objective & spread is calculated from scratch and
printed.
  maxSpread = 0;
  for c = 1:NoCrucibles
    spread = max(x(c,:)) - min(x(c,:));
    maxSpread = max(maxSpread,spread);
    CrucibleAl = mean(PotAl(x(c,:)));
    CrucibleFe = mean(PotFe(x(c,:)));
    CrucibleValue(c)=
CalcCrucibleValue(CrucibleAl,CrucibleFe,NoQualities,QualityMinAl,Quality
MaxFe,QualityValue);
    fprintf('%2d [%3d %3d %3d] %5.2fAl  %4.2fFe %6.2f %2d\n',c,
x(c,:),...
            CrucibleAl , CrucibleFe, CrucibleValue(c), spread);
  end
  fprintf('                         Sum,Max=
%6.2f,%2d\n',sum(CrucibleValue), maxSpread)
  totalValue = sum(CrucibleValue);
end
```