Promethee-II Algorithm

Variable Description:

detail[][]: a 2-D matrix used to store details of the objects being compared

number\_row: Number of objects being compared

number\_column: Number of constraints under consideration

max[]: a 1-D array to store maximum value under a constraint

min[]: a 1-D array to store mini mum value under a constraint

beneficial[]: a 1-D array to determine whether a given constraint is beneficial or non- beneficial

weight[] : a 1-D array to store the weights assigned to a constraint

normalized\_matrix[][] : a 2-D matrix to store normalized form of input data

difference matrix[][][]: An intermediate 3-D matrix to store intermediate processing data

aggregated[][]: a 2-D array to store row wise aggregate of an object from difference\_matrix[][][]

flow[][]: a 2-D array to store leaving and entering outranking flow

rank[]: a 1-D array to store the calculated final ranks corresponding to an object

Pseudocode:

promethee()

normalizing()

difference\_matrix()

calc\_rank()

normalizing()

for i 🡨 0 to number\_column do

low 🡨 detail[0][i]

large🡨 detail[0][i]

for j🡨 1 to number\_row do

if detail [j][i] > large

large=detail[j][i]

else if detail[j][i] < low

low = detail[j][i]

end loop

max[i]=large

min[i]=low

end loop

for i 🡨 1 to number\_column do

if beneficial[i] = 1

for j🡨 0 to number\_row do

normalized\_matrix[j][i] = (detail[j][i]–min[i]) / (max[i]-min [i])

end loop

else

for j🡨 0 to number\_row do

normalized\_matrix[j][i] = (max[i]-detail[j][i]) / (max[i]-min[i])

end loop

end loop

end func

difference\_matrix()

for i 🡨 0 to number\_row do

for j 🡨 0 to number\_column do

i1🡨0

for l🡨 0 to (number\_row-1) do

if i1!=i

difference\_matrix[i][l][j]=

normalized\_matrix[i][j]-normalized\_matrix[i1][j]

else

l 🡨l-1

i1🡨i1+1

end loop

end loop

end loop

for i 🡨 0 to number\_row do

for j🡨 0 to number\_column do

for l🡨 0 to number\_row-1 do

if difference\_matrix[i][l][j] < 0

difference\_matrix[i][l][j]=0

difference\_matrix[i][l][j]=difference\_matrix[i][l][j] \* weight[j];

end loop

end loop

end loop

tw 🡨 0

for i🡨0 to number\_column do

tw 🡨 weight[i] + tw

end loop

//declaring a temporary array to process data named t[] of size (number\_row-1) \* number\_row)

count 🡨 0

for i🡨 0 to number\_row do

for j🡨0 to (number\_row-1) do

rt 🡨 0

for l🡨 0 to number\_column do

rt 🡨 difference\_matrix[i][j][l] + rt

end loop

t[count] 🡨 rt

count🡨count+1

end loop

end loop

count 🡨 0

for i🡨 o to number\_row do

for j 🡨 0 to number\_row do

if i = j

aggregated[i][j] 🡨 -1

else

aggregated[i][j] 🡨 t[count]

count 🡨 count+1

end loop

end loop

end func

calc\_rank()

for i🡨 0 to number\_row do

row\_sum 🡨 0

column\_sum 🡨 0

for j🡨 0 to number\_row do

if i!=j

row\_sum=row\_sum+aggregated[i][j]

column\_sum=column\_sum+aggregated[j][i]

end loop

flow[i][0] = row\_sum / (number\_row – 1)

flow[i][1] = column\_sum / (number\_row – 1)

end loop

//declare a temporary array named te[] of size number\_row

for i🡨 0 to number\_row do

flow[i][2]=flow[i][0]-flow[i][1]

rank[i] 🡨 -1

te[i] 🡨 flow[i][2]

end loop

ranking 🡨 1

for i 🡨 0 to number\_row do

max\_index=max(te,r)

rank[max\_index]=ranking

ranking🡨 ranking + 1

te[max\_index]= -1

end loop

for i 🡨 0 to number\_row do

print “Object “+(i+1)+”: “+rank[i]

end loop

end func