Minimum Colours Grouping

Markus Sass
[22548890]

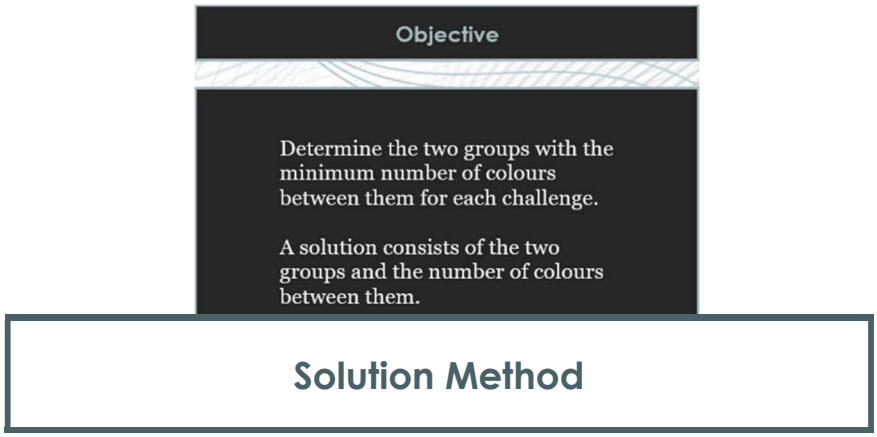
26/5/22

—

OR314

—

Prof Visagie



Solution Method

Algorithm Explanation

My model is based on swapping nodes between two groups based on a scoring method, which will be explained.

Initially the total nodes are equally distributed into two separate groups, namely group1 and group2. Then it iterates through each column of each group(1) and then row, looking for a leading value of one in that column. Once found, it then checks horizontally the remaining rows for another value of one. For each non-zero value found, it means that each is a colour edge within its own group. Hence, it adds a -1 score to its group scoring. The reason for this is that we do not want nodes with many colour connections within their groups to be swapped to another group, or at least minimize it.

Within that same initial column loop, it now does the same as above, but now to group2. It now iterates through group2 looking for related nodes and edges, however, instead of subtracting 1, it now adds 1, since we want those edges that are crossing to the other groups to be swapped as much as possible, to minimize the colour crossings.

Once completed, the scores are combined resulting in a net exchange of the nodes which should or should not be swapped. Those with higher values are favoured, hence, the maximum score and node is calculated, and that node is then transferred to the opposing group(2).

Now, we have an extra item in the other group(2), hence, we need to repeat it, but now using group(2) as the column iterations.

Ultimately, group2 scores are calculated and the max net score is swapped. This leaves the groups having equality (size of n/2 in each).

Following, the group scores are recalculated within their new positions. If all scores are less than one, meaning that no move will bring anymore benefit, the algorithm terminates with those groups. Else, it continues. This completes one iteration. There is also a check to prevent infinite looping, by storing a previous version of group1 from a couple iterations back and then comparing it further down the line, if no improvements were made, the algorithm also finds termination as it has found a stalemate. This could be investigated further for improvements.

Pseudo Algorithm

1. Split into equal groups
2. Loop until terminated
3. Calculate scores of each group
   respective to their colour edges to and within groups.
4. Find index of max score in group1
5. Transfer to group2
6. Find index of max score in group2
7. Transfer to group1
8. Recalculate scores
9. If max score of both < 1, terminate
   Else:
   Continue until either max score <1 or the groups are in stalemate

Code in R

```
Mat #imported data as matrix from excel
no_of_edges_per_color=rowSums(mat)
no_of_colors_per_node=colSums(mat)

#split nodes equally into g1 and g2
group1<-NULL
group2<-NULL
no_of_nodes=length(no_of_colors_per_node)
no_of_colours=length(no_of_edges_per_color)


for (i in 1:no_of_nodes){
 if (i<=no_of_nodes/2){
  group1[length(group1)+1]<-i
 } else {
  group2[length(group2)+1]<-i
 }
}

max=no_of_nodes
b=1
bb=0
change=1
count=0
cnt=0
old_group1=group1
while ((b==1)&&(count<=max)) {
 count=count+1
 #g1

 if (cnt>2){ #after 3 iterations no change, then terminate
  no=0
  for (i in 1:length(group1)){
   if (old_group1[i]==group1[i]){
    no=no+1
   }
  }
  if (no==length(group1)){
   b=0
   break
  }
  old_group1=group1
  cnt=0

 }
 cnt=cnt+1
```

```
group1_scores<-NULL
group2_scores<-NULL
for (y in group1){
 score=0
 for (x in 1:no_of_colours){
  if (mat[x,y]==1){
   for (yy in group1){
    if(yy!=y){#skips first since know =1
     if(mat[x,yy]==1){
      score=score-1
      break #in color to its group
     }
    }

   }
  }
 }
 #check score of other group
 for (v in group2){
  score2=0
  for (w in 1:no_of_colours){
   if (mat[w,v]==1){
    for (vv in group2){
     if (vv!=v){
      if(mat[w,vv]==1){#found another color to g2
       score2=score2+1
       break #out color to other group = more reson to swap
      }
     }
    }
   }
  }
 }
 score=score+score2
 #add score of each col for group 1
 group1_scores[length(group1_scores)+1]<-score #score of in/out degrees
}
#max value of score must exit and be added to g2
index_max_score_group1=-1
max_score_group1=-100
for (i in 1:length(group1_scores)){#finds max score and index
 if (group1_scores[i]>max_score_group1){
  max_score_group1=group1_scores[i]
  index_max_score_group1=i
 }
}
```

```
 if (change==1){#just recalculating not changing
  group2[length(group2)+1]<-group1[index_max_score_group1] #add max
score of g1 to group2
  group1=group1[-index_max_score_group1] #remove that col that was added
to g2
 }

 ###Group 2 has extra item, now find largest to equal again
 #same as first just inverse groups and scoring

 for (y in group2){
  score=0
  for (x in 1:no_of_colours){
   if (mat[x,y]==1){
    for (yy in group2){
     if(yy!=y){#skips first since know =1
      if(mat[x,yy]==1){
       score=score-1
       break #in color to its group
      }
     }

    }
   }
  }
  #check score of other group
  for (v in group1){
   score2=0
   for (w in 1:no_of_colours){
    if (mat[w,v]==1){
     for (vv in group1){
      if (vv!=v){
       if(mat[w,vv]==1){#found another color to g2
        score2=score2+1
        break #out color to other group = more reson to swap
       }
      }
     }
    }
   }
  }
  score=score+score2
  #add score of each col for group 1
  group2_scores[length(group2_scores)+1]<-score #score of in/out degrees
 }
 #max value of score must exit and be added to g2
 index_max_score_group2=-1
 max_score_group2=-100
 for (i in 1:length(group2_scores)){#finds max score and index
  if (group2_scores[i]>max_score_group2){
```

```
    max_score_group2=group2_scores[i]
    index_max_score_group2=i
   }
  }
  if(change==1){
   group1[length(group1)+1]<-group2[index_max_score_group2] #add max
score of g2 to group1
   group2=group2[-index_max_score_group2] #remove that col that was added
to g1
  }

  ##Iteration complete groups are in equality
  #now recalculate score to see if termination is fit
  if(bb==1){
   b=0
  }

  if (change==1){
   change = 0
  } else {
   #check termination if not then change=1
   sum_groupScores=1
   for (i in 1:length(group1)){
    if(group1_scores[i]>0){
     sum_groupScores=1
     break
    }
    if (group2_scores[i]>0){
     sum_groupScores=1
     break
    }
    sum_groupScores=-1
   }
   if (sum_groupScores<=0) {
    #termination satisfies
    bb=1
    change=1
   } else {
    #not optimal
    change=1
   }
  }

}
#group1 contains nodes
print(group1)
#group2 contains nodes
print(group2)
```

The counting of colour crossing algorithm

```
no_of_colours=length(rowSums(mat))
no_of_colours

val_g1=0
val_g2=0
mat
#splits groups into their matrices
mat_g1<-c(mat[,group1[1]])
mat_g2<-c(mat[,group2[1]])
mat_g1
mat_g2
mat

for (i in 2:length(group1)){
  mat_g1=cbind(mat_g1, mat[,group1[i]])
  mat_g2=cbind(mat_g2, mat[,group2[i]])
}
mat_g1
mat_g2

#calculates crossings by basically summing rows and counting those
non-zero entries between the two groups
colour_crossing=0
for (i in 1:length(rowSums(mat_g1))){
  val_g1<-rowSums(mat_g1)
  val_g2<-rowSums(mat_g2)
  if (val_g1[i]>0){
    if (val_g2[i]>0){
    colour_crossing=colour_crossing+1
    }
  }
}
print(val_g1)
print(val_g2)
print(colour_crossing)
```

Solutions:

Example Problem.
> #group1 contains nodes
> print(group1)
[1] 4 6 5 2
> #group2 contains nodes
> print(group2)
[1] 7 8 1 3
> print(colour_crossing)
[1] 1

1. 5_8
> #group1 contains nodes
> print(group1)
[1] 4 6 5 3
> #group2 contains nodes
> print(group2)
[1] 7 8 1 2
> print(colour_crossing)
[1] 1

2. 30_10

> #group1 contains nodes
> print(group1)
[1] 1 2 5 7 3
> #group2 contains nodes
> print(group2)
[1] 6 8 9 10 4
> print(colour_crossing)
[1] 22

3. 90_40

> #group1 contains nodes
> print(group1)
[1] 1 2 3 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
> #group2 contains nodes
> print(group2)
[1] 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 4
> print(colour_crossing)
[1] 78

4. 150_50

```
> #group1 contains nodes
> print(group1)
 [1]  1  2  3  4  5  6  7  8  9 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 10
> #group2 contains nodes
> print(group2)
 [1] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
> print(colour_crossing)
[1] 125
```

5. 250_60

```
> #group1 contains nodes
> print(group1)
 [1]  2  3  4  5  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
37  6
> #group2 contains nodes
> print(group2)
 [1] 31 33 34 35 36 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
58 59 60  1 32
> print(colour_crossing)
[1] 235
```

6. 500_60

```
> #group1 contains nodes
> print(group1)
 [1]  1  2  3  4  5  6  7  8 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
30 48
> #group2 contains nodes
> print(group2)
 [1] 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 49 50 51 52 53 54 55 56
57 58 59 60  9
> print(colour_crossing)
[1] 487
```

Discussion:

This was my fourth and final attempt of an algorithm, the previous algorithms being either incorrectly calculated or were found to be inconsistent.

After many explorations (and sleepless nights) and use of different programming languages, I resulted to R and ventured towards this method of thinking. This proved to result in consistent grouping even when the groups were split up differently, always resulting back to the same solution. By no means is this optimal, however, consistency was an attractive attribute.

The termination as mentioned, has rooms for improvement as well as efficiency of the looping structures. I know for example, based off my Computer Science knowledge, I can improve performance greatly by just changing those outer for loops to row-major instead of column-major. Not much time was spent on efficiency.

However, given not much information of what and how an optimal solution would look like, I am satisfised with the outcome of this method. For small, scaled matrices that can be worked out by hand, it resulting it successfully completing those.

For the larger ones, I cannot comment confidently on their outcomes. However, judging by the outcomes of the larger matrices, optimal looks very unlikely. I would be interested in viewing other methods to compare and be exposed to different thought processes or be provided with visual representations of the solutions.