Algorithms Final Project Repor

組別： 第二組

成員:電機三b01901046謝沅廷0988595693

方彥鈞

熊浚佑

樊恩宇

一、分工內容

1.謝沅廷：

建出整個graph及edge，改寫之前提供的模版，整合與修改 code，實作窮舉法，研究基因演算法及實作基因演算法及跑實驗結果表一和表二。

2.方彥鈞：greedy algorithm實作及想法，包含一開始BFS的塗色。

3.熊浚佑：建出window以及group對應window的table還有關係。

4.樊恩宇：用dfs建出group，幫忙修改精簡code，以及研究openGL並能把圖形畫出來。

Makefile

樊謝：line 1~49

src/tm\_usage.h 和 src/tm\_usage.cpp是老師提供

src/graph.h: 4個人一起製作

src/graph.cpp: line 1~ 87 , line 178~302 之前作業

謝：line 87~157, 304~332, 796~985

方：line 351~519

熊：line 523~794

樊：line 87~157, 159~176, 334~349

src/main.cpp:

謝：line 1~190

GA/ga.h

謝：line 1~49

GA/ga.cpp

謝：line 1~241

doc/report

分工內容：謝方熊樊

實驗結果：謝

二、演算法

1.建graph：就是將每個shape的左下角座標sort，然後讓shape延伸長出上下右3個長方形，從左至右check有沒有overlap，有就要建edge，當左右距離超過alpha，或上下距離超過beta就停止，worst case仍是O(n^2)，但比純n取2的方法會快非常非常多。

2. 找出group: (樊恩宇打吧)

3. 找出box size及無法塗色的group： (方)

4. 建出window對應group的關係： (熊)

5. 窮舉塗色：用test case 大概估算出多少group和window以下，可以在不超過大會時間限制下，用窮舉來求最佳解，即是把每個group要不要invert都跑過一遍，複雜度O(2^n)。

6. GD color 演算法： (方)

7. 基因演算法：我找到一篇paper有關於adaptive GA，他是有點採用統計的方式，作法是兩個合起來，首先是MOGAR：

把每個group要不要invert的資訊，變成一串由01構成的序列，這就是GA中的染色體，我一開始依照group的大小會去生成一個row為100\*group，column為group的矩陣，所以每一個row就是每一個染色體，就是一種塗色的情況，每一次iteration中，會得到這些row的分數，將row按照分數排列，再去random產生數字，看數字有沒有比機率i/N-1小(i=在哪一列，N=總列數)，去看這行row要不要mutate，所以最前面的不會變動，而後面愈低分愈容易改變，而每一行要變動的entry會依照一個機率pjx=...... ，bj= 來選，也就是說，如果這一行很多1的話，那這一行就比較容易繼續是1，也就是因為分數高有較高機率被留下，所以這行裏面佔較多數的entry，就是這個loci比較好的結果。

再來是MOGAC：

這就是先選column，一樣random一個數字之後，去看有沒有比bj小，有的話就會去選最後面幾行的row讓他們在這個loci上突變。

我最後的結果是給定一個gamma=0.9，每次iteration再去隨機產生一個0~1的數字，當他比gamma小時，就用mogar，比gamma大時就用mogac。

8. 最後結果：當input case很小，會直接用窮舉去塗色，不然就會先用greedy去塗色，得到分數後，再用genetic algorithm去塗色。

三、實驗結果

1. Exhausted

2. Greedy Original

3. Genetic Algorithm

Each test 10 times, Genetic Algorithm takes average

Exhausted with small groups and windows

Genetic Algorithm original with 100\*groups rows and 200\*groups iteration

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Score table | Exhausted | GD lar. | GD max. | GD Ori. | GA Avg | GA highest | GA lowest | GA Hit  Highest |
| Test | 68.1605 | 68.1605 | 67.4938 | 67.4938 | 68.1605 | 68.1605 | 68.1605 | 100% |
| case1 | 69.464 | 59.8593 | 62.1051 | 62.3426 | 69.44 | 69.464 | 69.2265 | 90% |
| case2 | x | 64.612 | 64.3754 | 61.6641 | 64.7167 | 65.0668 | 64.6995 | 10% |
| case3 | x | 51.1934 | 58.6249 | 50.0752 | 50.0752 | 50.0752 | 50.0752 | 100% |
| case4 | 69.7816 | 65.1362 | 69.1468 | 69.6292 | 69.7471 | 69.7816 | 69.6292 | 80% |
| case5 | x | 68.5077 | 69.4174 | 68.3504 | 69.4174 | 69.4174 | 69.4174 | 100% |

Avg. runtime in 10 times

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Run time(second) | Exhausted | GD Ori. | GA Avg | Time improve  v.s. Exhausted |
| Test | 0.000653 | 0.0011 | 1.82 | INF |
| case1 | 0.0018 | 0.0009 | 1.98 | -100% |
| case2 | x | 0.0017 | 82.8016 | INF |
| case3 | x | 0.012 | 464 | INF |
| case4 | 227 | 0.0025 | 21.42 | 98% |
| case5 | x | 0.009 | >1000 | INF |