CS 6150: Assignment 5
Due: Dec 12, 2014

This assignment has 5 questions, for a total of 100 points and 0 bonus points. Unless otherwise specified, complete and reasoned arguments will be expected for all answers.

Question	Points	Score
Biased and unbiased coins	20	
Pareto Optimality	20	
Good Chebyshev bounds	20	
Partition	20	
Approximate Graph Coloring	20	
Total:	100	

The following two questions are from Jeff Erickson's lecture on `Nuts and Bolts`

Question 1: Biased and unbiased coins . . . . **[20]**

Solve Question 6. The breakdown of points per subpart is $2 + 6 + 6 + 6$

Solution: (a)Let X denotes the value ONEINTHREE returns. if we want ONEINTHREE returns 1, then the function FAIRCOIN should return 1 firstly and this probability is $\frac{1}{2}$. Then the recursive part $1 -$ ONEINTHREE should return 0 with the probability that ONEINTHREE returns 0. Hence,

$$\begin{aligned} \mathbf{Pr}[X = 1] &= \mathbf{Pr}[\text{FAIRCOIN} = 1](1 - \mathbf{Pr}[X = 1]) = \frac{1}{2}(1 - \mathbf{Pr}[X = 1]) \\ \Rightarrow \mathbf{Pr}[X = 1] &= \frac{1}{3}. \end{aligned}$$

(b)If FAIRCOIN $= 0$, then this algorithm will stop and it only calls FAIRCOIN one time. Otherwise, it will recursively calls itself and hence calls FAIRCOIN. Let T denotes the calls number.

$$\begin{aligned} E(T) &= \frac{1}{2} * 1 + \frac{1}{2} * E(T) \\ \Rightarrow E(T) &= 1. \end{aligned}$$

Hence this algorithm is expected to call FAIRCOIN one time.
(c) FAIRCOIN:

```
if{ONEINTHREE = 1}
if{ONEINTHREE = 1}
return FAIRCOIN;
else
return 1 − ONEINTHREE;
```

Proof. Since ONEINTHREE only returns 0 or 1, it will return 0 with probability $\frac{2}{3}$. Let X denotes the returned value of FAIRCOIN. As the algorithm shows above,

$$\begin{aligned} \mathbf{Pr}[X = 1] &= \frac{1}{3} * \frac{1}{3} * \mathbf{Pr}[X = 1] + \frac{2}{3} * \frac{2}{3} \\ \mathbf{Pr}[X = 1] &= \frac{1}{2}. \end{aligned}$$

□

(d)let T denotes the time FAIRCOIN calls ONEINTHREE.p

$$\begin{aligned} E(T) &= \frac{1}{3}(1 + \frac{2}{3} * 1 + \frac{1}{3} * (1 + E(T))) + \frac{2}{3} * 2 \\ \Rightarrow E(T) &= \frac{9}{4}. \end{aligned}$$

Hence FAIRCOIN will call ONEINTHREE $\frac{9}{4}$ times in average.

Question 2: Pareto Optimality . . . . **[20]**

Solve Question 10. The breakdown of points per subpart is $6 + 14$.
(a)

Solution:

Algorithm 1: Compute The Pareto-optimal points

Require: The n points' Abscissas $X[1..n]$ and their Ordinates $Y[1..n]$;
Ensure: output the set of Pareto-optimal points;

```
1: sort $Y[1..n]$ decreasingly and get m different Ordinates;
2: and permute corresponding $X[1..n]$;
3: List $Ordinate[1..m] \leftarrow 0$, $cur \leftarrow 1$;
4: for $i \leftarrow 1$ to $n-1$ do
5:    if $Y[i] = Y[i+1]$ then
6:       add $i$ to List $Ordinate[cur]$;
7:    end if
8:    if $Y[i]! = Y[i+1]$ then
9:       add $i$ to List $Ordinate[cur]$ and $cur \leftarrow cur + 1$;
10:   end if
11: end for
12: add $n$ to List $Ordinate[cur]$;
13: $max \leftarrow 0$;
14: for $i \leftarrow 1$ to $m$ do
15:    $x \leftarrow max\{X[j]|j \in Ordinate[i]\}$;
16:    if $x \geq max$ then
17:       output $\{k|X[k] = x\}$;
18:       $max \leftarrow x$;
19:    end if
20: end for
```

EXPLANATION: The elements in array $Ordinate[1..m]$ are lists. List $Ordinate[i]$ stores the points whose have the $i-th$ largest Ordinates. From Line 14 to Line 19, the algorithm traversals these Lists. For each list, find the points who have the largest Abscissas x among the points in the same list and then compare the largest Abscissas with the max Abscissas max of points who are above. Meanwhile, update the max Abscissas.

Proof. The algorithm can be proved by Induction Method
Base Case: The points in $Ordinate[1]$ who have the largest Abscissas are Pareto-optimal points since they have the largest Ordinates as well and no points are both above and to the right of them. The algorithm can find them correctly and then max is updated to the largest Abscissas.
Hypothesis: After the algorithm traversals $Ordinate[i-1]$, it can find the Pareto-optimal points among the points in first $i-1$ Lists correctly and max is the largest Abscissas of these points.
Induction: For $Ordinate[i]$, the algorithm will firstly find the points who have the largest Abscissas. For these points, if their Abscissas is larger than or equal to max, it implies points above don't have Abscissas larger than that of these points. Hence, no points above are in both above and right region of these points. Obviously, other points on the List have smaller Abscissas and the points below have smaller Ordinates, thus they can't appear in both above and right region of these points. So these points are the Pareto-optimal points. The algorithm can correctly find them. Then max is updated to the Abscissas of these points. Hence now max has the largest Abscissas of points in first i Lists. If the largest Abscissas of points in List $Ordinate[i]$ is smaller than max, it implies that there are points above in both above and right region of all points in List $Ordinate[i]$ and the algorithm doesn't output any point in this List. Hence the algorithm is correct. □

*Analysis.*The algorithm firstly sorts the n points by their Ordinates, and good algorithm can sort them in $O(nlogn)$ time. Then the algorithm adds points with the same Ordinate to the same List. Since the points have been sorted by Ordinates, this process just needs to traversal the n points. It costs $O(n)$ time. Finally, the algorithm searches for the Pareto-optimal points by traversing the m Lists. For each List, it just needs to find points with the largest Abscissas in one traversal and then compares the largest Abscissas with the largest Abscissas of points above. Hence for each List, its time cost is $O(k)$, here, k is the number of points in that List. So this process costs $O(n)$ in total. Hence, the algorithm costs $O(nlogn)$ in total. For the space cost, the algorithm uses List array $Oridinate[1..m]$ to store points in specific Lists. These Lists store the n points, so the total space cost is $O(n)$.

(b)

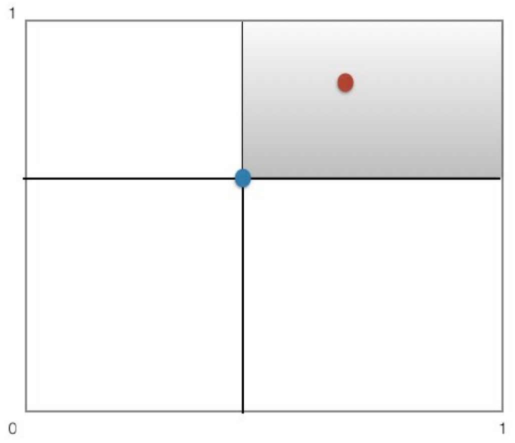

Figure 1

Solution:
solution 1:
For any n points in the plane, sort them by their Ordinates decreasingly firstly. For the first sorted point, since it has no point above it, it is a Pareto-optimal point. For the second point, it has no larger Ordinate than the first one, it is the Pareto-optimal point if and only if its Abscissas is no smaller than the first point. Since each point is chosen independently and uniformly at random form the square, the probability that second point's Abscissas is no smaller than the first one is $\frac{1}{2}$. Generally, for the i^{th} point, it's a Pareto-optimal point if and only if its Abscissas is no smaller than the first $i-1$ points, the probability is $\frac{1}{i}$. Let $Num(n)$ denotes the number of Pareto-optimal points

among n points and $isPar(i) = 1$ means point i is a Pareto-optimal point. Hence,

$$E(Num(n)) = \Sigma_{i=1}^{n}\mathbf{Pr}[isPar(i) = 1] * 1 = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \approx ln(n) + c.$$

solution 2:
Credit to the work on a webpage $http : //math.stackexchange.com/questions/206866/expected-number - of - pareto - optimal - points$.
Figure 1 shows an example that there are only two points. If the green point is not a Pareto-optimal point, the red point should be on the right-top corner of the green point as the grey area in Figure 1 shows. Hence, if the green point is a Pareto-optimal point, the red point should be on the other area other than the gray area. Since each point in S is chosen independently and uniformly at random from the unit square, hence the possibility that the green point is not a Pareto-optimal point is $\int_0^1\int_0^1[1-(1-x)(1-y)]dxdy$. Hence, if there are n points in total, a specific point is a Pareto-optimal point if and only if other $n-1$ points are all on the areas other than the top-right area of this specific point. Hence, the possibility that a specific is a Pareto-optimal point is $\int_0^1\int_0^1[1-(1-x)(1-y)]^{n-1}dxdy$. Define the event that if point i is a Pareto-optimal point, $I_i = 1$ and otherwise, $I_i = 0$. Hence,

$$P(I_i = 1) = \int_0^1\int_0^1[1-(1-x)(1-y)]^{n-1}dxdy = \frac{1}{n}(1 + \frac{1}{2} + ... + \frac{1}{n}).$$

Hence, the except number of Pareto-optimal points is $\sum_{i=1}^{n} P(I_i = 1) = 1 + \frac{1}{2} + ... + \frac{1}{n} \approx ln(n) + c$.

Question 3: Good Chebyshev bounds .... **[20]**
This is Exercise 4.9 from Probability And Computing, by Mitzenmacher and Upfal.

We'd like to get a good estimate of $E[X]$ for a random variable X by sampling it n times. We can sample $X_1, \ldots, X_n$ and try to use a tail bound to estimate how close the sample mean $\frac{1}{n}\sum_i X_i$ is to $E[X]$. But we don't know anything about the higher moments of X and we don't even know that it's a $0-1$ random variable. In fact, we might only have a bound on Var$[X]$. Let $r = \frac{sqrt\text{Var}[X]}{E[X]}$.

(a) **[5]** Use a Chebyshev bound to show that we can estimate $E[X]$ to within an additive error of $\varepsilon E[X]$ with probability $1-\delta$ using $O(\frac{r^2}{\varepsilon^2\delta})$ samples.

Solution: Suppose it needs n sample to estimate $E[X]$ to within an additive error of $\varepsilon E[X]$ with probability $1-\delta$. Hence,

$$\mathbf{Pr}[(|\frac{X_1 + X_2 + \cdots + X_n}{n} - E(X)| \leq \varepsilon E(X)] = 1 - \delta$$
$$\Rightarrow \mathbf{Pr}[(1-\varepsilon)E(X)n \leq X_1 + X_2 + \cdots + X_n \leq (1+\varepsilon)E(X)n] = 1 - \delta.$$

Now Define $Y = X_1 + X_2 + \cdots + X_n$. Since $X_1, X_2, ..., X_n$ are independent variables and $E(X_1) = E(X_2) = \cdots = E(X_n) = E(X)$, $E(Y) = E(X_1) + E(X_2) + \cdots + E(X_n) = nE(X)$. Hence,

$$\mathbf{Pr}[(1-\varepsilon)E(Y) \leq Y \leq (1+\varepsilon)E(Y)] = 1 - \delta$$
$$\Rightarrow \mathbf{Pr}[|Y - E(Y)| \leq \varepsilon E(Y)] = 1 - \delta.$$

The Chebyshev bound tells us that $\mathbf{Pr}[|Y - E(Y)| \leq k\sqrt{Var[Y]}] \geq 1 - \frac{1}{k^2}$. When $k\sqrt{Var[Y]} =$

$\varepsilon E(Y)$, $k = \frac{\varepsilon E[Y]}{\sqrt{Var[Y]}}$. Hence,

$$\mathbf{Pr}[|Y - E(Y)| \le \varepsilon E(Y)] = 1 - \delta \ge 1 - \frac{Var[Y]}{\varepsilon^2 E^2[Y]}$$
$$\Rightarrow 1 \le \frac{Var[Y]}{\varepsilon^2 E^2[Y]\delta}.$$

Since $X_1, X_2, \cdots, X_n$ are independent variables, $Var[Y] = Var[X_1] + Var[X_2] + \cdots + Var[X_n] = nVar[X]$. Hence,

$$1 \le \frac{nVar[X]}{\varepsilon^2 n^2 E^2[X]\delta}$$
$$\Rightarrow n \le \frac{Var[X]}{\varepsilon^2 E^2[X]\delta} = \frac{r^2}{\varepsilon^2\delta}.$$

So it needs $O(\frac{r^2}{\varepsilon^2\delta})$ samples to estimate $E(X)$ to within an additive error of $E(X)$ with probability $1-\delta$.

(b) [**5**] The previous bound gives a strong probability $(1-\delta)$ of being close to the mean. Suppose we only want the probability of error to be 1/4. In that case, show that $O(r^2/\varepsilon^2)$ samples suffice (call these *weak samples*)

Solution: When the probability of error becomes 1/4, so $\delta = 1/4$. According to conclusion of part(a), $n \le \frac{4r^2}{\varepsilon^2}$. So $O(\frac{4r^2}{\varepsilon^2}) = O(\frac{r^2}{\varepsilon^2})$ is enough.

(c) [**10**] Now let's take $\log(1/\delta)$ weak samples as in (b) above, and compute their median. Show that this gives us the desired error $(\varepsilon E[X])$ *and* the desired probability bound $(1-\delta)$. How does the *total* number of samples needed compare with the bound in part (a) ?

Solution: Define these samples are $Y_1, Y_2, ..., Y_m$. In addition, define $\overline{Y_1}, \overline{Y_2}, ..., \overline{Y_m}$ as their means. As part (b) shows, $Pr[|\overline{Y_i} - E[X]| \ge \varepsilon E[X]] = \frac{1}{4}$. Define the median of $\overline{Y_1}, \overline{Y_2}, ..., \overline{Y_m}$ is $\overline{Y_{med}}$. $|\overline{Y_{med}} - E[X]| \ge \varepsilon E[X]$ if and only if at least a half of the m weak samples satisfying $|\overline{Y_i} - E[X]| \ge \varepsilon E[X]$, $1 \le i \le m$. It can be proved:
(1)When $\overline{Y_{med}} - E[X] \ge \varepsilon E[X]$, at least a half of the weak samples' means are no smaller than $(1+\varepsilon)E[X]$, otherwise, $\overline{Y_{med}} - E[X] < \varepsilon E[X]$ since $\overline{Y_{med}}$ is the median of all the means.
(2)When $E[X] - \overline{Y_{med}} \ge \varepsilon E[X]$, at least a half of the weak samples' means are no larger than $(1-\varepsilon)E[X]$, otherwise, more than a half of the weak samples' means are larger than $(1-\varepsilon)E[X]$ including the median $\overline{Y_{med}}$ which is a contradiction.
Define the event $Pr[|\overline{Y_i} - E[X]| \ge \varepsilon E[X]] = \frac{1}{4}$ P and P= 1 means this event happens. Here, $i = 1, 2, ..., m$. Hence, $Pr[P = 1] = \frac{1}{4}$. Since $Y_1, Y_2, ..., Y_m$ are independent samples, hence the event p satisfies Binomial distribution. $Pr[|\overline{Y_{med}} - E[X]| \ge \varepsilon E[X]]$ is equal to the probability that p happened among at least a half of the m weak samples. According Hoeffding's inequality, we can get,

$$\delta = Pr[|\overline{Y_{med}} - E[X]| \ge \varepsilon E[X]] = Pr[H(m) \ge \frac{1}{2}m] = Pr[H(m) \ge (Pr[P=1] + \frac{1}{2})m] \le e^{-2(\frac{1}{4})^2 m}$$
$$\Rightarrow \delta \le e^{-\frac{1}{8}m} \Rightarrow m \le \frac{8\log\frac{1}{\delta}}{\log e} = O(\log\frac{1}{\delta}).$$

It implies that take $O(log\frac{1}{\delta})$ weak samples and compute the median, then $Pr[|\overline{Y_{med}} - E[X]| \leq \varepsilon E[X]] \leq 1-\delta$.
For this method, we need $O(\log(\frac{1}{\delta})\frac{r^2}{\varepsilon^2})$ samples in total. Define $t = \frac{1}{\delta}$ and $k = \log(t) - t$. It's obviously $t > 1$, then,

$$\frac{dk}{dt} = \frac{1}{t} - 1 < 0 \Rightarrow k \textit{ decreases as } t \textit{ increases} \Rightarrow k < \log(1) - 1 = -1 < 0 \Rightarrow \log(t) < t.$$

Hence, $O(\log(\frac{1}{\delta})\frac{r^2}{\varepsilon^2}) < O(\frac{r^2}{\varepsilon^2\delta})$. This method needs fewer samples.

The following two questions are from Jeff Erickson's lecture on `Approximation Algorithms`

Question 4: Partition .... **[20]**
Solve Question 5. *Hint: look at the algorithm for scheduling in the notes.*

Solution: Discussed with Liang Zhang and do by myself.
The approximation ratio of the algorithm is 2. Let OPT denotes the optimal solution.

Proof. OPT is the maximum of the two sets and we can assume $\Sigma A \geq \Sigma B$(it doesn't effect the result) in the optimal solution. So $\Sigma A + \Sigma A \geq \Sigma_{i=1}^{n} X[i]$ and $OPT \geq \frac{1}{2}\Sigma_{i=1}^{n} X[i]$. Assume ΣA has the maximum again, and $X[j]$ is the last integer added to it. $X[j]$ is added to set A implies set A has smaller sum than set B for the first $j-1$ integers, i.e, $\Sigma_{i=1}^{m} A < \Sigma_{i=1}^{k} B (m + k = j - 1)$. Moreover, $j = n$. Otherwise, it implies $X[j+1]$ will be added to set B since $X[j]$ is the last integer of set A. X is sorted in a increasing order, so $X[j] < X[j+1]$. Then $\Sigma_{i=1}^{m+1} A = \Sigma_{i=1}^{m} A + X[j] < \Sigma_{i=1}^{k} B + X[j+1] = \Sigma_{i=1}^{k+1} B$ and it implies set B is maximum and it's a contradiction. So, the last integer in the set with larger sum is just $X[n]$. Hence $\Sigma A - X[n] < \Sigma B$ and $\Sigma A - X[j] < \Sigma A$, i.e., $\Sigma A - X[n] < \frac{1}{2}(\Sigma A + \Sigma B) = \frac{1}{2}\Sigma_{i=1}^{n} X[i] \leq OPT$. Now define $tA = \Sigma A - X[n], tB = \Sigma B$ and $\frac{X[n]}{tA} = k$. It's obvious that $X[n] \leq OPT$, hence $ktA \leq OPT$. In addition, $tA \leq OPT$ as proved above. What's more, since $\Sigma_{i=1}^{n} \leq 2OPT$, then $tA + tB + X[n] \leq 2OPT$. Since $tB > tA$, then,

$$\begin{aligned} tA + tB + X[n] &= (1+k)tA + tB \geq (2+k)tA, \\ &\Rightarrow (2+k)tA \leq 2OPT. \end{aligned}$$

Define $\theta = \frac{tA}{OPT}$ and $tA + X[n] = (1+k)tA \leq hOPT$, then,

$$\begin{aligned} &\theta \leq 1 \\ &\theta \leq \frac{1}{k} \\ &\theta \leq \frac{2}{2+k} \\ &h \geq (1+k)\theta. \end{aligned}$$

The target is to figure the minimum of h satisfying $h \geq (1+k)\theta$ for all k and θ.
If $\frac{2}{2+k} \geq \frac{1}{k}$, then $k \geq 2$ and $\frac{1}{k} \leq \frac{1}{2}$. Hence θ just needs to satisfy $\theta \leq \frac{1}{k}$. Hence $(1+k)\theta \leq \frac{1+k}{k} \leq \frac{3}{2}$. So the feasible minimum of h is $\frac{2}{3}$ when $k \geq 2$.
If $\frac{2}{2+k} \leq \frac{1}{k}$, then $k \leq 2$. Since $k > 0$, $\frac{2}{2+k} \leq 1$. So θ just needs to satisfy $\theta \leq \frac{2}{2+k}$. Hence $(1+k)\theta \leq \frac{2(1+k)}{2+k} \leq \frac{3}{2}$. So the feasible minimum of h is $\frac{2}{3}$ when $k \leq 2$.
In conclusion, $tA + X[n] = \Sigma A \leq \frac{3}{2}OPT$. Similarly, when $\Sigma B \geq \Sigma A$, $\Sigma B \leq \frac{3}{2}OPT$. Hence, the approximation ratio is $\frac{3}{2}$. □

Question 5: Approximate Graph Coloring . . . . **[20]**

Solve Question 6. *Hint: You need to modify the input graph in some manner before presenting it to the claimed approximation algorithm.*

Credit to the idea of Zhao Chang and do by myself.

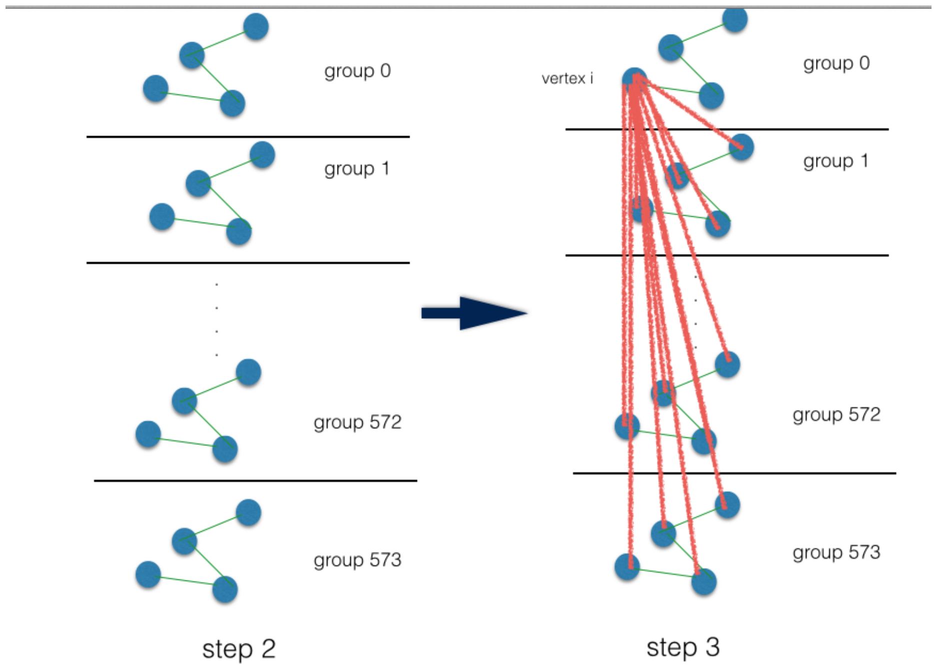

Figure 2

Proof. Firstly, create a new graph according to the original $G = (E, V)$, where $V = \{v1, v2, ..., vi, ..., vm\}$ as the following: (1)For each vertex vi in the graph G, produce 573 copies of it, i.e., $vi_1, vi_2, ..., vi_{573}$;
(2)For $(k = 1$ to $573)$
consider $v1_k, v2_k, ..., vm_k$ as group k, connect the vertices $v1_k, v2_k, ..., vm_k$ as the same way as $v1, v2, ..., vm$, i.e., if there is an edge between vi and vj, then connect vi_k and vj_k. In addition, consider $v1, v2, ..., vm$ as group 0;
(3)For $(k = 0$ to $573)$
For $(i = 1$ to $m)$
connect vi_k with vertices in all groups except group k.
Now all the vertices and edges described above form the new graph G'.
The following figure is an example. The left-hand side represents the 2nd step and the right-hand side represents the 3rd step. Group 0 is the original graph G. Connect each vertex with all the other vertices except the ones in the same group just as vertex i on the right-hand side shows. Then a new graph G' is a constructed.

If $\chi(G)$ is the chromatic number of graph G, then the chromatic number of graph G' $Y(G)$ is $574\chi(G)$, it can be proved:

Proof. In each group of G', $\chi(G)$ is obviously the chromatic number since each group has the same con structure with G . Now, for any vertex, if we use the same approaches to color the vertices within its group as G with $\chi(G)$ colors, it won't have same colors with vertices that connects with. For different groups, we use totally different $\chi(G)$ colors, i.e., for any vertex $i \in$ group k1, and any vertex $j \in$ group k2, they won't have same colors. Now, for any vertex, its color won't conflict with the vertexes on other

groups since different groups have totally different colors. Hence, $574\chi(G)$ satisfy the requirement that every edge have endpoints with different colors.
In addition, $574\chi(G)$ is the minimum number of colors required to color the vertices of G'. If the chromatic number is smaller than $574\chi(G)$, it can be considered as at least one color disappears among these colors since at least two vertices with different colors now have the same color. Obviously, for any vertex i and any vertex j in different groups, they are connected in G' and they can't use the same color. Hence at least two vertices in the same group with different colors use same color now and this causes the one color disappears in that group. It's impossible since$\chi(G)$ is the chromatic number of that group. Hence, $574\chi(G)$ is the chromatic number of G'. □

Hence, the chromatic number of graph G' is the multiple of 574. Hence there is only one number between $Y(G)$ and $Y(G)+573$ that is the multiple of 574, that is $Y(G)$. If there is an algorithm that can return an integer between $Y(G)$ and $Y(G)+573$ within polynomial time, for example $Y(G)+y$, then we just needs to test whether it's the multiple of 574, if not, subtract it by one and test again until it succeeds and this process can be finished in polynomial time. Hence, it implies that we can find the chromatic number of G' in polynomial time. Hence, if finding the approximated solution can be solved in polynomial time, finding the chromatic number of G' can be solved within polynomial time, however, it's a NP-HARD problem, hence, if it's solved in polynomial time, all NP problems can solved. It implies that if returning a number between $Y(G)$ and $Y(G)$ is solved in polynomial time, then all NP problems can be solved. Hence it's a NP-HARD problem as well. □