Finding the fractional clique cover number and the Shannon entropy for graphs

Abstract

In this study we carried out optimisation for the given linear programming problems using PuLP python library to find the optimal values: the fractional clique cover number and the Shannon entropy, given various constraints. After modifying the LPs and constraints to the appropriate form, we successfully optimised the problems and derived the optimal value and the corresponding vector for each case.

1 Introduction

For any graph G with vertices V and edges E, we have *G(E,V)*. For each subset S of V, we have a value x_S. The fractional clique cover number and Shannon entropy are the optimal values of $Minimise \sum x_S$ over all possible S and $Maximise\ x_V$, respectively, subject to various constraints. Since the constraints initially were given in a mathematical form, we solved the LPs on paper for some very simple graphs to better understand how to derive the constraints for the computer. In our program we used the python library PuLP to optimise the Linear Programs.

2 The fractional clique cover number: $\pi^*(G)$

PuLP requires us to identify the variables that appear in the constraints of the problem. These variables are all x_S such that $S \subseteq V$.

According to the constraints all x_S had to be greater or equal to 0, moreover we were looking to minimise the sum, thus we set the initial value of all x_S to 0. Then we derived $K(G)$, the collection of sets of $G(V, E)$ which are cliques of G too. The constraints were the following:

$$\sum_{S:v\in S} x_S \geq 1 \qquad \forall v \in V, \tag{1}$$

$$x_S = 0 \qquad \forall S \notin K(G). \tag{2}$$

Since x_S for all non-clique S has to be zero, we modified the above constraints so that it is only relevant to the cliques of G, and we left the x_S for every non-clique S to be zero:

$$\sum_{S\in K} x_S \geq 1 \qquad \forall v \in V. \tag{3}$$

In fact, this means we can simplify our objective function to the following:

$$Minimise \quad \sum_{S\subseteq K} x_S. \tag{4}$$

The only one constraint that is left to put in our system is Equation 3. To do so, for each v in V we created a list which contained the relevant x_S variables, and then included this in another list, thus we created a list of lists called *constraints*, where the first dimension corresponds to the given vertex, and the second dimension contains the x_S values for subsets $S \in K$ which contains the given v. Next for each v we added the constraint to the LP problem: that the sum of the values in the second dimension of *constraints* for a given v must be greater or equal to 1. For example, if $K = \{\{\emptyset\}, \{a\}, \{a, b\}, \{b\}\}$, then $constraints = [[a, ab], [b, ab]]$ and we have:

$$x_{\{a\}} + x_{\{a,b\}} \geq 1 \quad and \quad x_{\{b\}} + x_{\{a,b\}} \geq 1, \tag{5}$$

hence to minimise the Equation 4 we have $x_{\{a\}} = x_{\{b\}} = 0$ and $x_{\{a,b\}} = 1$.

3 The Shannon entropy: $\eta(G)$

First, similarly to the LP above, we added the basic constraints to the system i.e. $x_{\{\emptyset\}} = 0$, $x_{\{v\}} \leq 1 \quad \forall v \in V$ and $x_S \geq 0 \quad \forall S \subseteq V$. For the remaining three constraints, several modification were required. For

$$x_{N(v)\cup\{v\}} - x_{N(v)} = 0 \qquad \forall v \in V \tag{6}$$

we created a 2d list, *con*1, where the first dimension list corresponded to the vs, and for each v we had a list containing 2 elements in the second dimension of *con*1: the x value of the subset N of V, where N is the neighbourhood of vertex v, and the other element is the x value for the same neighbourhood containing the vertex v itself. Then we put this list in a loop and added the constraint Equation 6 to the LP solver for each v.

Next we have

$$x_S - x_P \geq 0 \qquad \forall P \subseteq S \subseteq V. \tag{7}$$

For each S we created a list of its subsets: the possible sets P. Then for each x_S we have a list ,*con*2, of x_P: a 2d list. The way we represented this is that the first element of in the second dimension lists is the corresponding x_S and this is then followed by the relevant x_P values. For instance for $V = \{a, b\}$ we have $S = [x_\emptyset, x_a, x_b, x_{a,b}]$ and thus $P_{x_\emptyset} = [x_\emptyset]$, $P_{x_a} = [x_\emptyset, x_a]$, $P_{x_b} = [x_\emptyset, x_b]$ and $P_{x_{a,b}} = [x_\emptyset, x_a, x_b, x_{a,b}]$. Therefore $con2 = [[x_\emptyset, x_\emptyset], [x_a, x_\emptyset, x_a], [x_b, x_\emptyset, x_b], [x_{a,b}, x_\emptyset, x_a, x_b, x_{a,b}]]$.

Once we established the constraints lists and rules, it was straightforward to use nested loops to satisfy Equation 7: for each element of *con*2, we subtracted the all but the first element of the sublist from the first element one by one and adding them to the LP solver as constraints. In the previous example it gives $x_{a,b} - x_a \geq 0$ and $x_{a,b} - x_b \geq 0$. Note that in each sublist of *con*2, the first and last x derived from P (that is the second and last element of the sublist) can be neglected due to the previous constraints. By doing so, we successfully improved performance.

Our last constraint states

$$x_S + x_T - x_{S\cup T} - x_{S\cap T} \geq 0 \qquad \forall S, T \subseteq V. \tag{8}$$

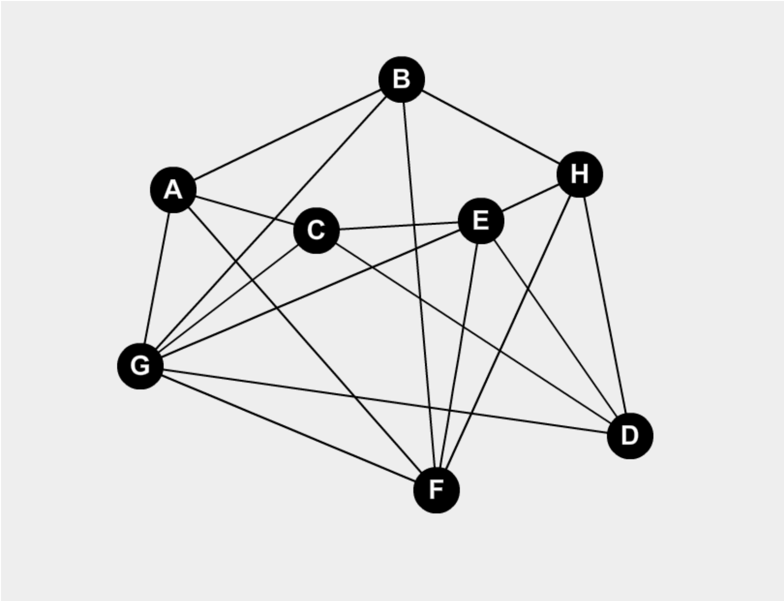
By using nested loops we produced the values $[x_S, x_T, x_{S\cup T}, x_{S\cap T}]$ for each combination of S and T, obtaining a 2d list, *con*3, of size $|S|^2 \times 4$. Note that if $x_S = x_{S\cup T}$ *and* $x_T = x_{S\cap T}$ or $x_S = x_{S\cap T}$ *and* $x_T = x_{S\cup T}$, then this gives 0, thus we neglected these options which resulted in better performance. Finally, using another loop on *con*3 we added the constraints to the LP solver according to Equation 8.

4 Program execution

To run the program, make sure PuLP python library is installed and *optimisation.py* and *rational.py* are in the same directory. Open *optimisation.py*: to input your graph, modify V (vertices) and E (edges) on the top of the script, so that it matches your graph's properties. Make sure the vertices are single-valued letters e.g. 'a', 'b', 'c', ..., and the edges are the concatenation of the relevant vertices e.g. 'ab', 'bc', 'ca', etc. Next, execute the program, which will produce a text file *optimal_solution.txt* which tells us if the found solution is optimal, its value and the corresponding vector for both LPs, in rational format.

5 Results

As an output the program produces the optimal values for $\pi^*(G)$ and $\eta(G)$ with their corresponding vectors of xs: $\overline{x}$. Furthermore, it successfully verified the results for some basic simple graphs that we calculated earlier on paper and fulfilled the theorems that for a complete graph K_n with n vertices we have $\pi^*(K_n) = 1$ and $\eta(K_n) = n - 1$, and that for any G_n we have $\pi^*(G_n) + \eta(G_n) \geq n$. Figure 1 shows the diagram and the results obtained from a graph with




```
V=['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
E=['fg', 'ef', 'ce', 'ag', 'cg', 'bf', 'de', 'bg', 'af', 'ac', 'eh', 'ab', 'bh', 'fh', 'dg', 'dh', 'cd', 'eg']

LP1 Status:Optimal
fractional clique cover number = 5/2
and the corresponding vector x:
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1/2,0,0,0,0,0,0,0,0,0
,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1/2,0,1/2,0,0,0,0,0,0,0,0,0,0,0,1/2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1/2,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

LP2 Status:Optimal
Shannon entropy =11/2
and the corresponding vector x:
[0,1,1,1,1,1,1,1,1,2,3/2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,5/2,3,3,3,3,3,5/2,5/2,5/2,5/2,5/2,3,3
,
3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,5/2,3,3,3,3,3,3,3,3,3,3,3,3,3,5/2,3,3,3,3,3,3,3,7/2,7/2,7/2,7/2,7/2,4,4,
4,4,4,4,4,7/2,7/2,4,7/2,7/2,7/2,7/2,7/2,7/2,7/2,7/2,7/2,7/2,4,4,7/2,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,7/2,4,4,4,7/2,4,
7/2,4,4,7/2,4,7/2,4,7/2,7/2,4,4,4,4,4,4,4,4,7/2,7/2,4,4,9/2,9/2,9/2,9/2,9/2,9/2,9/2,4,4,9/2,5,5,9/2,9/2,9/2,5,9/2,
9/2,5,4,9/2,4,4,9/2,9/2,9/2,9/2,9/2,9/2,9/2,5,9/2,9/2,5,5,5,9/2,9/2,5,9/2,5,5,9/2,5,9/2,5,4,9/2,9/2,9/2,9/2,9/2,4,
5,5,9/2,11/2,5,5,5,5,11/2,5,5,11/2,9/2,11/2,5,11/2,5,5,5,5,9/2,11/2,11/2,11/2,11/2,5,5,11/2,11/2,5,5,11/2,11/2,11/
2,11/2,11/2,11/2,11/2,11/2,11/2]
```

Figure 1: The top figure shows a graphical representation of the problem and the bottom one shows the results obtained from such a graph G.

8 vertices. Clearly, $\pi^*(G) = 5/2$ and $\eta(G) = 11/2$ sums up to $8 = n$, so the program respected the theorem from earlier. Moreover, one can see that all of the obtained values are in rational format, unless they are integers.

5.1 Limitations

For our script to work properly, we need to have single-valued letters as vertices, meaning that the program cannot take more than 26 vertices (assuming English alphabet). Furthermore we found it hard to test the accuracy of the system as there was not much material to compare the results with. The obstacles throughout the process were to first understand what the problem was and what the constraints meant through a real example, and then to translate the mathematical constraints to computer language. During programming we also struggled assigning PuLP variables to the x values in a convenient way, because doing so changed the datatype of these xs and thus we could no longer use them as list elements. For this we created a copy of the vector $\overline{x}$ which we dynamically updated with the correct values while maintaining its datatype as string.