Implement the largest degree first heuristic and another heuristic of your choice to solve this problem. Give a brief description of the two heuristics and critically compare them.

One of the earliest ordering strategies is 'Largest Degree Ordering' (LDO), proposed by C. Avanthay, A.Hertz, and N. Zufferey [1]. Using this method, vertices are coloured in order of descending degree, with those nodes with the largest degree being coloured first. A vertex with *N* coloured neighbours will require at most *N+1* colours and can be implemented to run in polynomial time ($O(n^2)$) where the performance is directly proportional to the square of the size of the input data set due to the requirement of a nested iterator in the algorithm [2]. The LDO algorithm attempts to keep the maximum value of N as small as possible throughout the colouring process so there is a better chance of using only a small number of colours [3].

Another colouring strategy and one of the simplest [4] which runs in polynomial time [5] is First-Fit (FF). The colouring of the graph is driven by vertexes in the graph being selected in an unpredictable order and assigning the smallest possible integer as a colour to the current vertex that respects the constraints imposed by the intervals that have already been coloured [6]. Umland's two-step algorithm for FF colouring first builds then colours the graph [5].

1. Determine a list of all possible colours for v_i i.e. exclude those colours already used by vertices V_j, $J < I$ adjacent to V_i. This could be implemented using a Boolean array L_i – called the possibility list of vertices V_i – with the property $L_i[k] = FALSE \Leftrightarrow \exists V_j$ with $j < i$, $(V_i, V_j) \in E$ and $f(V_j) = k$. This step can be performed by a procedure $Build(L_i, Vj)$ which excludes the colour of vertex V_j from the possibility list L_i of vertex V_i.
2. Determine the smallest of all possible colours for vertex V_j, i.e. look for the smallest entry in L_i with $L_i[k] = TRUE$ and assign colour k to V_j. We put this step into a procedure $Colour(L_i, V_i)$ which colours vertex V_i in dependence on its possibility list L_i.

When comparing the performance of both heuristics in the solution for the timetabling problem (table 1) it was not possible to discern between the both which was the best performing with respect to the number of colours used, both produced solutions which required 4 colours. The time taken to complete the problem with each heuristic was also recorded. The FF heuristic completed the problem in half the time taken by the LDO algorithm, this is to be expected as the LDO ordering process incorporates an iterative loop thus increasing the amount time taken for the colouring problem in line with the expected time of $O(n^2)$ [2].

Heuristic	Colours Used	Graph Build Time	Graph Colour Time	Total Time
FF	4	0.6041ms	0.0230ms	0.6271ms
LDO	4	0.5006ms	0.0483ms	0.549ms

Table 1: Results from timetabling solution using both LDO & FF heuristics

In order to get a more complete view of the performance of both heuristics a number of independent studies into the performance of colouring algorithms were analysed [2][7][8]. Each of these studies implemented both heuristic graph colouring algorithms with more complex problems with the amount vertices and edges

reaching more than 4000 and 77000 respectively in one problem. Each study found that LDO out performed FF with respect to the number of colours used, this is an anticipated solution as LDO aims to keep the maximum value of I as small as possible throughout the computation [3]. In Aslan and Baykan's study [7] the amount of time taken for each heuristic to complete is recorded and produced results similar to those see in my own solution. Although the LDO algorithm consistently matched or improved upon the number of colours used by the FF algorithm, FF was considerably faster at reaching the solution, this speed difference became more pronounced as the complexity of the graphs increased.

References

[1] Avanthay, C., Hertz, A. and Zufferey, N., (2003) *A variable neighborhood search for graph coloring*. European Journal of Operational Research, 151(2), pp.379-388.

[2] Mansuri, A., Gupta, V. and Chandel, R.S., (2010) *Coloring Programs in Graph Theory*. Int. Journal of Math. Analysis, 4(50), pp.2473-2479.

[3] Allwright, J.R., Bordawekar, R., Coddington, P.D., Dincer, K. and Martin, C.L., (1995) *A comparison of parallel graph coloring algorithms*. SCCS-666, pp.1-19.

[4] Gyárfás, A. and Lehel, J., 1988. *On-line and first fit colorings of graphs*. Journal of Graph theory, 12(2), pp.217-227.

[5] Umland, T., 1998. Parallel graph coloring using java. Architectures: Languages and patterns.

[6] Narayanaswamy, N.S. and Babu, R.S., 2008. *A note on first-fit coloring of interval graphs*. Order, 25(1), pp.49-53.

[7] Aslan, M. and Baykan, N.A., 2018. *A performance comparison of graph coloring algorithms*. International Journal of Intelligent Systems and Applications in Engineering, pp.1-7.

[8] Al-Omari, H. and Sabri, K.E., 2006. *New graph coloring algorithms*. American Journal of Mathematics and Statistics, 2(4), pp.739-741.