Guide #2
Different Approaches to the Multicolored Clique Problem

The **Multicolored Clique Problem (MCP)** is defined as follows: Given an undirected graph where each vertex is assigned one of k colors, find a maximum clique (complete subgraph) where all vertices have different colors.

Exercise A (40 points) Solve the (MCP) using a Branch-and-Bound Algorithm. Adapt the algorithm presented in class for the Maximum Clique Problem (MaxClique) to this problem. You will have to take the following points into consideration.

- Change the *Choice* function, to take into account the fact that you want all members of the clique to have different colors.
- Change the *size-bound* bounding function presented in class for the MaxClique to adapt it to the MCP.
- Change the backtracking with bounds algorithm presented in class, so that it is in accordance with the algorithm for branch-and-bound we studied.
- Test your algorithm with different random graphs with density 0.5. You can use the random graph generator provided by the book, but you will have to complete each graph, by randomly assigning k colors to the nodes.
- Empirically determine the largest value $nmax$, for n the number of nodes, for which your algorithm can solve the problem in less than a minute, when the number of allowed colors in $k = n/2$.
- Find 10 different graphs with $nmax$ nodes, for which your branch-and-bound algorithm needs approximately 1 minute to solve the MCP. Save the 10 graphs and respective solutions for exercise B and C.

Exercise B (30 points) Solve the MCP using a Hill Climbing Algorithm. You will have to take the following points into consideration.

- Clearly define you neighborhood function N and your local search function h_N.
- Include a randomized initialization.
- Do not use a c_{max} value to stop the algorithm, let the algorithm converge to the optimal solution. Run re-tries for half a minute, using the 10 graphs found in exercise A. Report, in average, how close does the algorithm get to the optimal solution found in A. Report *min*, *max*, *avg* and the number of times the optimal solution was obtained, as shown in class.

Exercise C (30 points) Solve the MCP using a Simulated Annealing algorithm. You will have to take the following points into consideration.

- Clearly define your neighborhood function N and your local search function h_N. Make sure that they now allow down-hill moves.
- Include a randomized initialization.
- Empirically determine suitable values to define a cooling schedule. I.e., test different values for T_0 and α to allow for sufficient exploration, but trying to keep running times low.
- Do not use a c_{max} value to stop the algorithm, let the algorithm converge to the optimal solution. Run re-tries for half a minute, using the 10 graphs found in exercise A. Report, in average, how close does the algorithm get to the optimal solution found in A. Report *min*, *max*, *avg* and the number of times the optimal solution was obtained, as shown in class.