Theoretical Background

Graph coloring is the process of assigning distinct colors to each vertex of a graph $G = (V, E)$ such that no two adjacent vertices share the same color. A graph G is said to be *k-colorable* if it is possible to color its vertices using at most k different colors while satisfying this condition. The smallest k for which G is k-colorable is called the *chromatic number* of G, denoted by $\chi(G)$. The graph coloring problem seeks to determine the value of $\chi(G)$.

A *clique* in a graph $G = (V, E)$ is a subset of vertices such that every pair of vertices in the subset is connected by an edge. The *clique number* $\omega(G)$ is the size of the largest clique in G. Determining $\omega(G)$ is known as the *maximum clique problem.* Since each vertex in a clique must be assigned a unique color in any valid graph coloring, the relationship

$$\omega(G) \leq \chi(G)$$

always holds.

Description of the Project

The task is to develop an algorithm that finds the chromatic number of a graph G. The algorithm should be based on a branch-and-bound framework. For this purpose, the bounds should be determined as follows:

- Implement a heuristic to find the maximum clique in a given graph G. The cardinality of this maximum clique, denoted by $\mathrm{lb}(\omega(G))$, serves as a lower bound on the clique number $\omega(G)$ and thus the chromatic number $\chi(G)$, that is
$$\mathrm{lb}(\omega(G)) \leq \omega(G) \leq \chi(G).$$
- Implement a heuristic to find a valid coloring of a given graph G. The number of colors used in this coloring, denoted by $\mathrm{ub}(\chi(G))$, provides an upper bound on the chromatic number $\chi(G)$, i.e.,
$$\chi(G) \leq \mathrm{ub}(\chi(G)).$$

The branch-and-bound framework should then proceed as follows:

- At the root node, compute the bounds $\mathrm{lb}(\omega(G))$ and $\mathrm{ub}(\chi(G))$.
- If $\mathrm{lb}(\omega(G))$ and $\mathrm{ub}(\chi(G))$ do not coincide, identify two non-adjacent vertices in the graph. Create two branches:
  1. In the first branch, assume the two vertices are assigned the same color.
  2. In the second branch, assume the two vertices are assigned different colors.

  Continue branching and updating the bounds until the chromatic number of G is found.

Note that the number of nodes in the branch-and-bound tree strongly depends on the quality of the bounds found and the branching strategies used. Implement the computation of bounds efficiently and choose some clever branching strategies. Make inter-node parallelization of the branch-and-bound algorithm using MPI. Results obtained at each branc and bound node (lower and upper bounds and the supporting cliques/colorings) should be carefully stored in a log file, to prove the final result.

The code should have the option to quit after the time limit is reached. The final code should be run on the list of benchmark instances.

Tasks for Students

Each team will be composed of 4 students from two different universities. The workload is 64 hours per student.

Students 1-2

- **1h-8h:** Get acquainted with the graph coloring and the maximum clique problems.
- **9h-24h:** Create initial heuristics. Define branching strategies.
- **25h-32h:** Improve heuristics and branching strategies.
- **33h-40h:** Create project documentation.
- **41h-48h:** Help testing sequential and parallel solver on given benchmark instances.
- **49h-56h:** Document the results. Help students 3-4 in computations on benchmark dataset.
- **57h-64h:** Finalize the project documentation.

Students 3-4

- **1h-8h:** Get acquainted with the branch-and-bound framework.
- **9h-24h:** Program a basic (sequential) branch-and-bound algorithm.
- **25h-32h:** Make initial tests with heuristics and strategies developed by students 1-2. Communicate results with students 1-2 so that they can improve bounding and branching procedures.
- **33h-48h:** Parallelize the branch and bound code and make tests with improved heuristics and branching strategies developed by students 1-2.
- **49h-56h:** Finalise the computations on benchmark dataset. Work with students 1-2 on project documentation.
- **57h-64h:** Prepare the poster.

Supervisor

Prof. Janez Povh will be a general supervisor of the challenge - he will communicate with mentors and monitor the overall progress of the challenge.

Mentors

Each team will be guided by one mentor. Academic mentors can offer advice regarding design choices and assist students with technical issues that fall outside the scope of the project.

Evaluation Criteria

.

The evaluation will be based on:

- Number of instances solved to optimum in the time limit of 10.000 seconds (regardless how many computing cores were used)
- The quality and precision of the final documentation.

Given the extensive scope of the material, it is expected that no single individual will master everything within the allocated time. Group members will work in parallel, developing their individual research skills while cooperating in a structured team environment.

Timeline of the Challenge

- **Start Date:**
- **End Date:**
- **Evaluation Period:**
- **Results Announcement:**

Deliverables

Solutions of the problem instances

- The solutions are expected to be in the form of a table where each row will contain:
  - Unique problem instance name
  - Size of the problem instance
  - Best solution obtained in the time limit (wall time) of 10.000 seconds
  - Indicator if the solution is optimum
  - Total computation time (wall time) needed (including the input/output data processing)
  - Resources needed (number of cores and nodes)
- Dependence of the total computation time (including the input/output data processing) on the problem size and the resources used (the Amdhal's law) in a form of graph plot dependent on
  - The problem size
  - Resources needed (number of cores and nodes)
- Bonus: Try to estimate the speedup as a function of
  - The problem size
  - Resources needed (number of cores and nodes)

Code

- The code is expected to be delivered as a git repository in Github or Gitlab where the code complies with the common best practices such as
  - Docstrings in a standard format
  - Instruction how to use the code (README, INSTALL)
  - Having a release version tagged as vM.m.p (Major, minor, patch)
  - Bonus: Container image as Dockerfile for users to use the solver as the "Out of the box" solver in a containerized form.

Reference

Depolli, M., Konc, J., Rozman, K., Trobec, R., & Janezic, D. (2013). Exact parallel maximum clique algorithm for general and protein graphs. Journal of chemical information and modeling, 53(9), 2217-2228.