Project – n-coloring of a graph

# Purpose:

Compare the performance of parallel vs. distributed algorithms for solving the n-coloring of a graph problem, which is an NP-complete problem.

# Implementation:

There are 3 important classes implemented:

* ***ColorsGraph***: used to keep track of the n colors and their codes; each color is associated a number for easier manipulation
* ***DirectedGraph***: used to store a directed graph; the representation is done on a map, where the (key, value) pair is by the (vertex, list\_of\_outbound\_vertices) tuple
* ***GraphColoring***: contains the implemented algorithms

## Parallel algorithm:

In the parallel algorithm, the main idea is to implement a backtracking algorithm that simultaneously searches for the valid solution. The program stops when a solution is found or when all possibilities are exhausted.

Each time we add a new vertex to the partial solution, we continue the search in a new thread, if there are available threads. If there are no available threads left, the search is continued sequentially in the same thread, with basic backtracking. A thread starts by checking if the solution has already been found. If the solution was already obtained, the thread stops, otherwise it continues with the computations.

Synchronization is achieved by using a lock for modifying the solution and an atomic variable for keeping count of the available threads. When a partial solution is checked and it is valid, the variable storing the final solution (initialized with an empty list) is modified using a lock, in order to prevent unwanted access from other threads.

## Distributed algorithm:

For the distributed algorithm, the main idea remains the same – implementing a backtracking algorithm that simultaneously searches for the valid solution. The program stops when a solution is found or when all possibilities are exhausted.

Each time we add a new vertex to the partial solution, we continue the search in a new process, if there are available processes. If there are no available processes left, the search is continued sequentially in the same process, with basic backtracking.

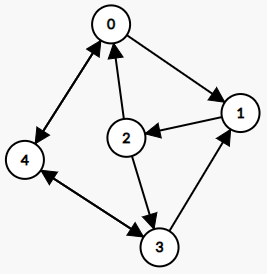
Communication between processes is established using MPI and is synchronous. Used methods are *Send(…)* and *Recv(…)*. The formula used for distributing the MPI processes is:

where:

* r = rank of current host
* m = no. of partial multiplications required by the alg. (3 or 4)
* p = power (increases as the level of recursion does), p= 0,1,..
* i = index of partial multiplication, iϵ[0,1,..,m)

# Performance:

For graph



and 3 colors available (red, green, blue):

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **No. of threads/processes** | **Time (µs)** |
| Parallel | 1 | 34852 |
| 4 | 25225 |
| 16 | 24493 |
| 64 | 26146 |
| Distributed | 1 | 3844 |
| 4 | 43076 |
| 16 | 26324 |
| 64 | 271200 |