EUMaster4HPC Challenge
Graph Vertex Coloring

Jan E. Marxen[1] Luca Lamperti[2] Mohamed Z.M. Mandour[2] Giacomo Pauletti[2]

[1]*Sorbonne Université* [2]*Politecnico di Milano*

Objective

Develop a parallel **branch-and-bound** algorithm to compute the **chromatic number** of graphs using the Vega supercomputer.

Vega architecture

CPU CLUSTER : 960 nodes, each with 2 AMD Rome 7H12 CPUs with 64 cores, for a total of 1,920 and 122,880 cores.

GPU CLUSTER : 60 nodes, each 4 Nvidia A100 GPUs and 2 AMD Rome 7h12 Cpus240 NVIDIA, for a total of 240 A100 GPUs

This project aims to leverage HPC capabilities to solve complex combinatorial optimization problems in graph theory.

Zykov algorithm

Given two non-adjacent vertices x, y ∈ V two new graphs can be defined:

- G′ xy where x and y are contracted or merged into one single vertex xy.
- G″ xy where the edge {x, y} has been added

A recursive algorithm, called Zykov's tree (Figure 1), can be built upon the following theorem:

Theorem 1 *The chromatic number of G is given by the recurrence*

$$\chi(G) = \min\{\chi(G'_{xy}), \chi(G''_{xy})\}$$

such that $x, y \in V(G)$ *and* $\{x, y\} \notin E(G)$

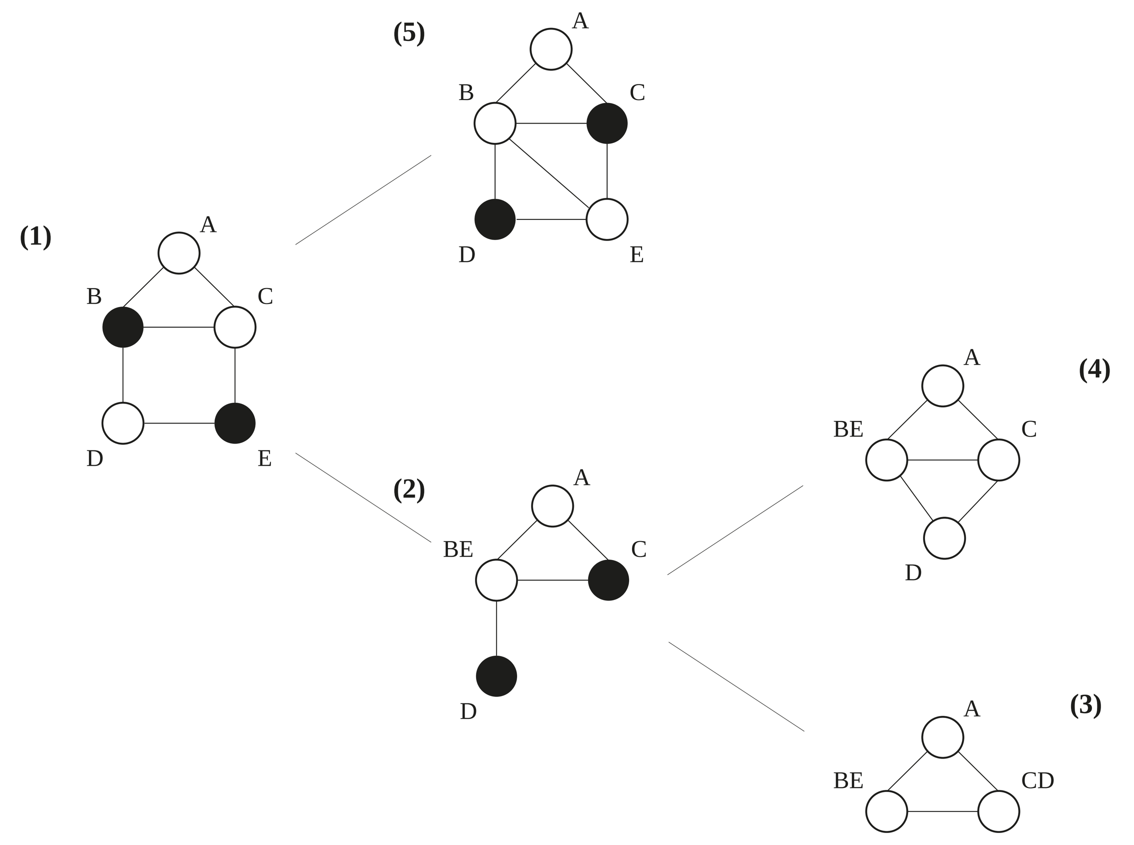


Figure 1: Zykov's tree

Leaf of the graph = complete graph - Size of **smallest leaf** = *chromatic number*.

At each node **lower bound** and **upper bound** computed:

$$lb(\chi) \leq \chi(G) \leq ub(\chi)$$

Prune the graph when: $best\_ub(\chi) \leq current\_lb(\chi)$

Coloring and clique heuristics

Color heuristics are used for calculating the upper bound. The most remarkable example is **DSatur**, as well as *Greedy* and *Recolor*.

Algorithm 1 DSatur coloring

procedure DSATURCOLOR(G)
  Initialize $max\_color \leftarrow 0$
  while G not empty **do**
    $v \leftarrow$ GETMAXSATDEGREE(G)
    for $i = 1 \rightarrow max\_color$ **do**
      if can be assigned color i **then**
        $color[v] \leftarrow i$ break
      end if
    end for
    if not assigned **then**
      $max\_color \leftarrow max\_color + 1$
      $color[v] \leftarrow max\_color$
    end if
  end while
end procedure

Clique heuristics are used for calculating upper bound; we adopted **FastWClq** algorithm.

Branching strategy

At each step verteices (u, v) are chosen such that **merging minimizes the graph**:

$$(u, v) = \arg \min_{(u,v) \in V} |N(u) \cap N(v)|$$

where $N(u)$ is the set of neighbors of u

Graph Representation

Operation	Adjacency Matrix	Edge List	CSR	Adjacency List
Add Edge	$O(1)$	$O(1)$	$O(\lvert E\rvert)$	$O(1)$
Remove Edge	$O(1)$	$O(\lvert E\rvert)$	$O(\lvert E\rvert)$	$O(k)$
Add Vertex	$O(\lvert E\rvert)$	$O(1)$	$O(1)$	$O(1)$
Remove Vertex	$O(\lvert E\rvert)$	$O(\lvert E\rvert)$	$O(\lvert E\rvert)$	$O(\lvert E\rvert)^1$
Merge Vertices	$O(\lvert E\rvert)$	$O(\lvert E\rvert)$	$O(\lvert E\rvert)$	$O(\lvert E\rvert)^1$
Get Neighbors	$O(\lvert V\rvert)$	$O(\lvert E\rvert)$	$O(1)$	$O(1)$

[1]More precisely $O((avg\_neighbors)^2)$, which tipically it is much less

We used **Adjacency list** representation since it is the most flexible

MPI & OpenMPI

Parallelize execution using **MPI** and **OpenMP**.

Work Distribution Models

- Simple Approach: Master orchestrates tasks.
- Scalable Approach 1 and 2: Each process explores its own search space, sharing solutions from time to time.

Multi-Threaded Processing

- Terminator: Monitors execution time and stops processes if needed.
- Gatherer: Collects solutions and updates global best results.
- Employer: Manages work-stealing (Idle workers take over unfinished tasks to balance the load)

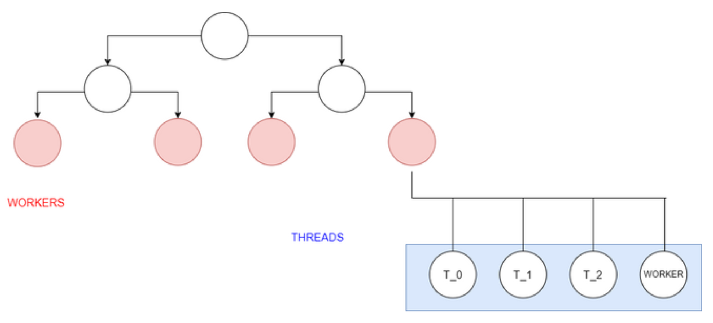


Figure 2: balanced algorithm

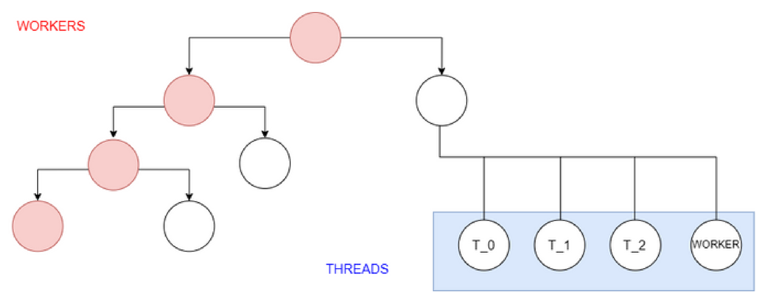


Figure 3: unbalanced algorithm

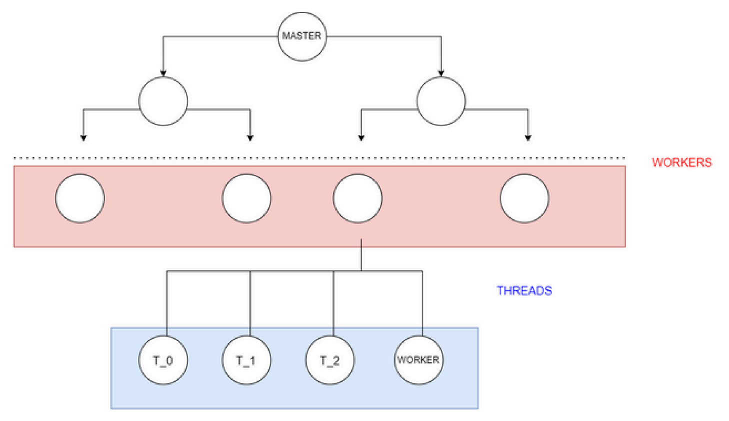
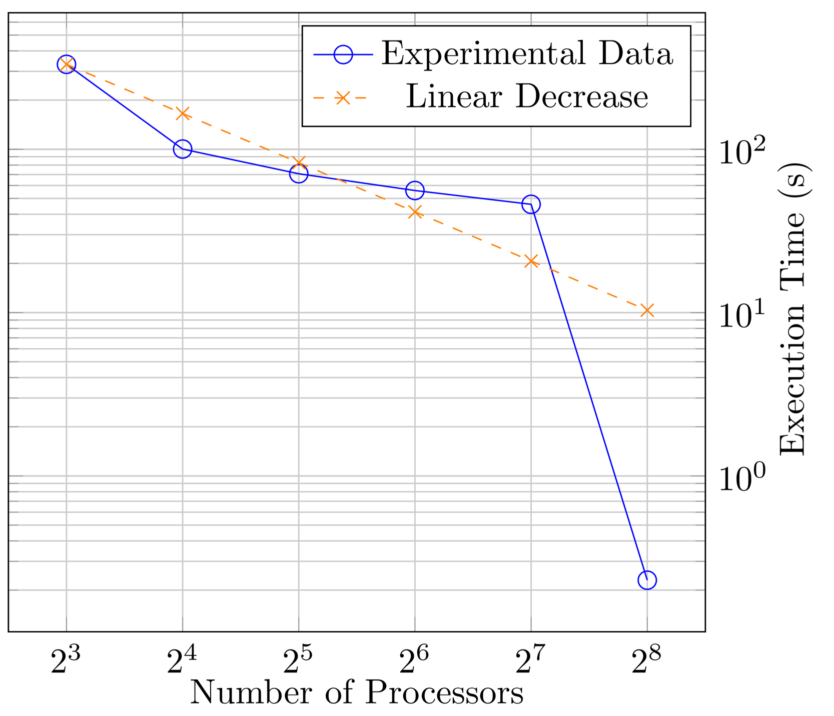


Figure 4: simple algorithm

Results

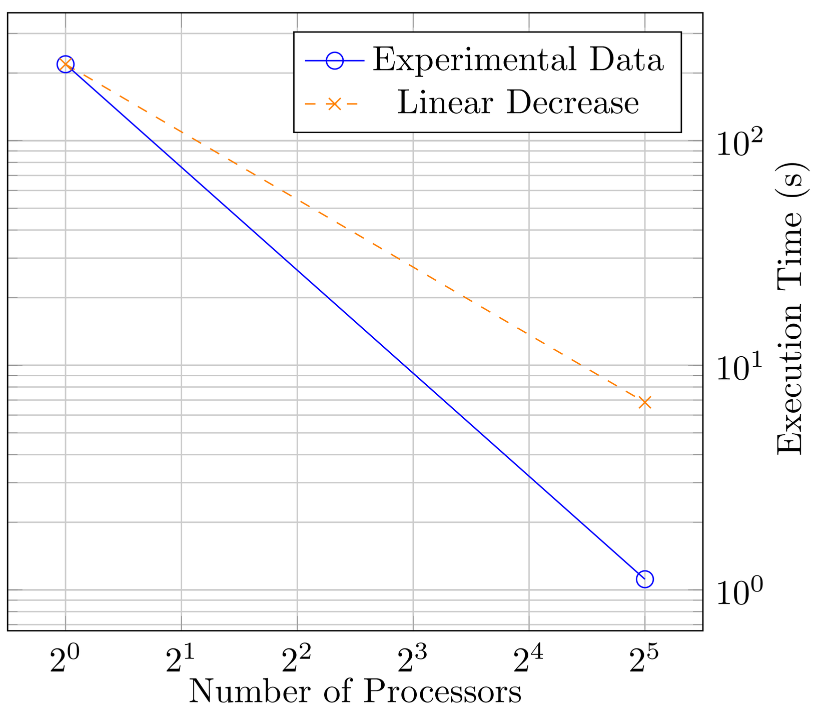
MPI proc	Cores	Time (s)
1	1	> 10000
8	32	330.91
16	64	100.22
32	128	70.78
64	256	55.89
128	512	46.01
256	1024	0.23

Strong scaling (queen7_7.col): **superlinear speedup** due to more pruning



MPI proc	Cores	Time (s)
1	1	218.86
32	128	1.12

Strong scaling (queen9_9.col). superlinear speedup due to more pruning. Plateau reached with 32 cores.



Conclusions

The proposed parallel branch-and-bound algorithm demonstrates **strong scalability**. Further optimizations for sparse graphs remain an open direction

Acknowledgements

The authors thank Dr. Arnaud Renard for his expert guidance and the Vega supercomputer team not only for their excellent support but also for the provision of the computing resources.

This project has received funding from the European High-Performance Computing Joint Undertaking under grant agreement No. 101051997

EUROHPC SUMMIT 2024, Antwerp — EUMaster4HPC See more: https://github.com/Rudolfovoorg/EuroHPC_Student_Challenge_2025_Team_2

jan.marxen@gmail.com l.lamperti2002@gmail.com mohamedzeyad.mandour@mail.polimi.it giacomo.pauletti@mail.polimi.it