A Path Clustering Heuristic for Scheduling Task Graphs onto a Grid

Bittencourt, L. F.; Madeira, E. R. M.; Cicerre, F. R. L; Buzato, L. E.
Institute of Computing – State University of Campinas – Brazil
{luiz.bittencourt, edmundo, fcicerre, buzato}@ic.unicamp.br

INSTITUTE OF COMPUTING

INTRODUCTION

• Task scheduling is an NP-Complete problem and efficient scheduling is very important for achieving good performance.
• *Path Clustering Heuristic* (PHC): a heuristic for scheduling task graphs onto a grid infrastructure that supports dependent task execution.
  • Baseline: select a path from the DAG and schedule the nodes on this path onto the same processor.
• Developed based on Xavantes grid middleware and its programming model [1].

XAVANTES GRID MIDDLEWARE

• Arranges the resources in groups, placing in the same group processors and data repositories that are physically close.
• Each group has three components: one Group Manager (GM), at least one Process Manager (PM) and at least one Activity Manager.
  • GMs: maintain information about the resources of the group and high granularity information about resources in adjacent groups.
  • PMs: schedule and distribute the process elements, and mantain the execution and data state of them, for monitoring, coordination and recovering.
  • AMs: execute activities.

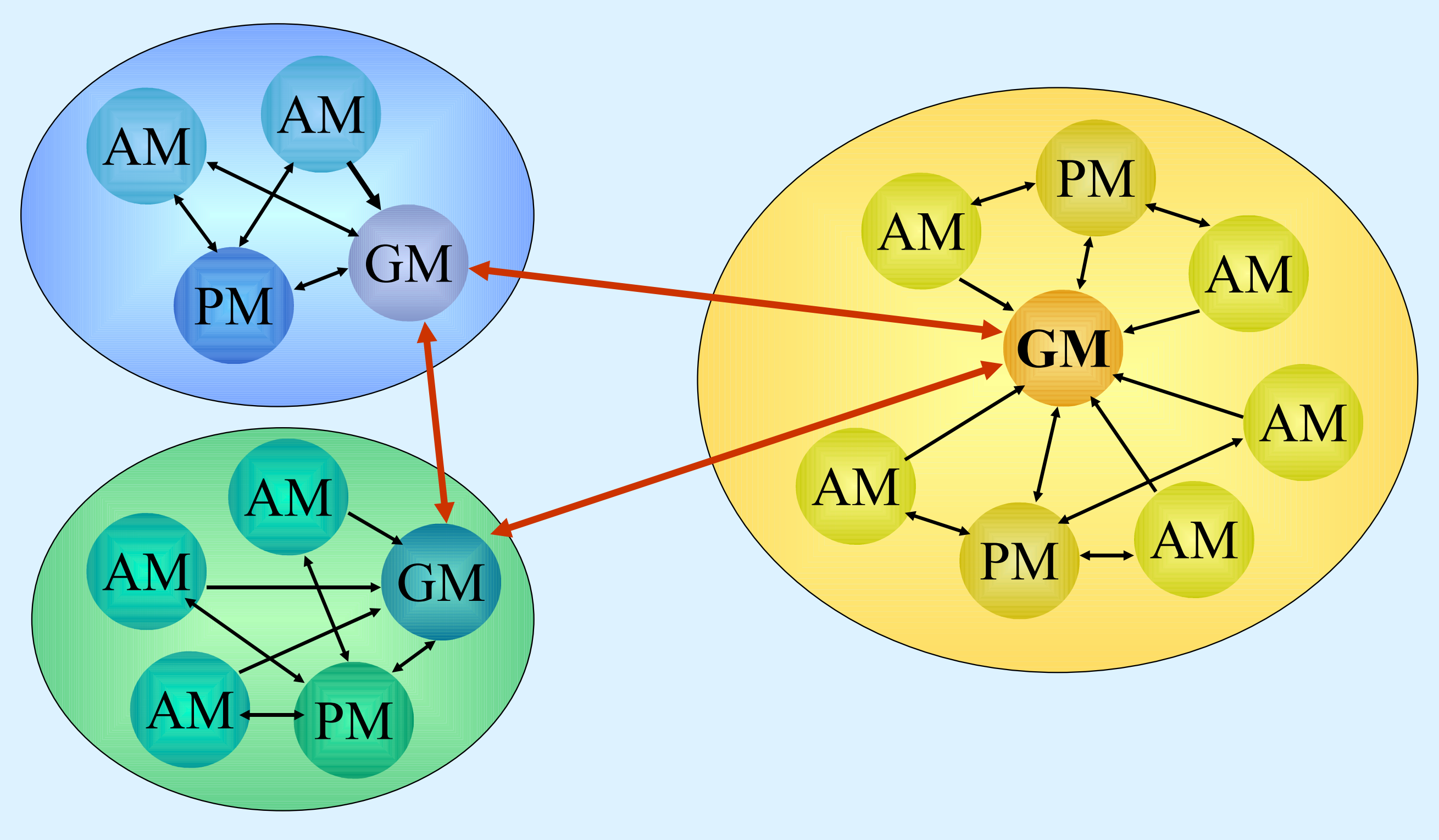

Figure 1: Xavantes architecture.

PROGRAMMING MODEL

• Allows to specify applications as structured processes, containing a hierarchy of control constructs to determine the control flow.
• A process is a set of subprocesses, controllers and activities.
• Controllers are control elements that specify the tasks execution order. They are also responsible for transmitting data between dependent tasks.
• Each task is subordinated to a controller. The controllers can be nested, allowing the hierarchical specification of the control, similarly to structured programming languages.
• Processes are represented by a Directed Acyclic Graph (DAG) and controllers are represented by rectangles.

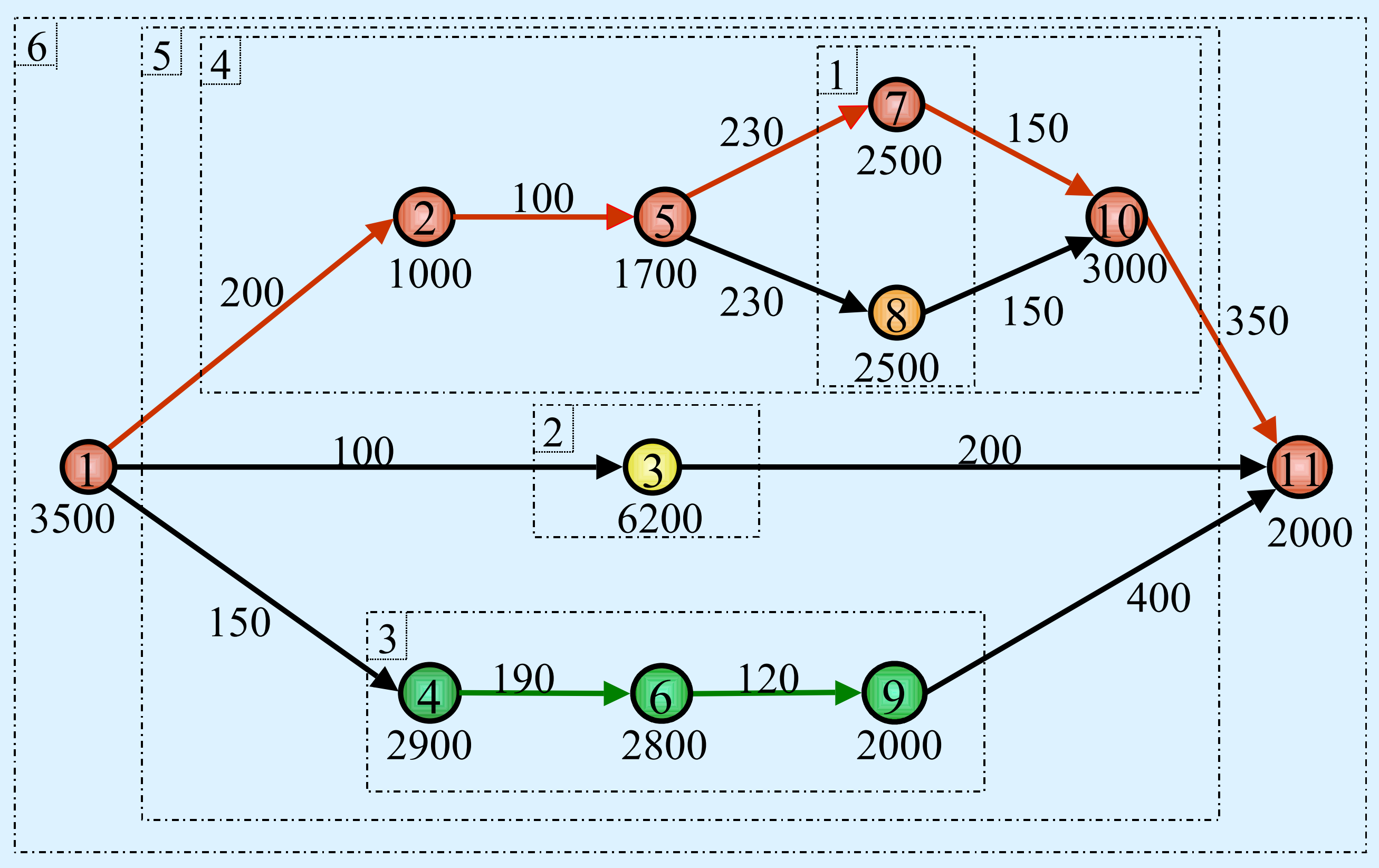

Figure 2: Process and controllers representation with the clusters produced by the PCH algorithm.

PROPOSED ALGORITHM

• Uses some attributes calculated for each task: *Priority*, *Weight* (Computation Cost), *Communication Cost*, *Earliest Start Time* (EST) and *Estimated Finish Time* (EFT).

• Computation and Communication Costs:

$$w_i = \frac{instructions_i}{power_p} \qquad c_{i,j} = \frac{data_{i,j}}{bandwidth_{r,t}}$$

• Priority:

$$P_i = w_j + \max_{n_j \in succ(n_i)} (c_{i,j} + P_j)$$

• Estimated Start Time:

$$EST(n_i, r_k) = \max\{time(r_k), \max_{n_h \in pred(n_i)} (EST_h + w_h + c_{h,i})\}$$

• Estimated Finish Time:

$$EFT(n_i, r_k) = EST(n_i, r_k) + \frac{instructions_i}{power_r}$$

CLUSTERING STEP

```
Algorithm 1 get_next_cluster
1: $n \Leftarrow$ unscheduled node with highest Priority
2: $cluster \Leftarrow cluster \cup n$
3: while ($n$ has unscheduled successors) do
4:     $n_{succ} \Leftarrow successor_i$ of $n$ with highest $P_i + EST_i$
5:     $cluster \Leftarrow cluster \cup n_{succ}$
6:     $n \Leftarrow n_{succ}$
7: return $cluster$
```

Algorithm 1: The clustering step.

• First cluster (red): nodes 1, 2, 5, 7 (or 8) and 10
• Second cluster (green): nodes 4, 6 and 9
• Third cluster (orange): node 8 (or 7)
• Fourth cluster (yellow): node 3.

PROCESSOR SELECTION STEP

```
Algorithm 2 get_best_resource
1: for all $r$ in $resources$ do
2:     $schedule \Leftarrow$ Insert $cluster$ in $schedule_r$
3:     calculate_EFT($schedule$);
4:     $time_r \Leftarrow$ calculate_EST(successor($cluster$))
5: return resource $r$ with the smallest $time_r$
```

Algorithm 2: The processor selection step.

CONTROLLER SCHEDULING

• Aims to minimize the communication of a controller with its nodes and with its subcontrollers.
• Clusters based on sequential tasks in the task graph helps in reducing the controller communication overhead.
• Baseline:
  • Makes a random choice among that controllers which have all subcontrollers already scheduled.
  • Send the controller to be executed on the resource that minimizes the communication between the controller and its tasks, and the controller and its subcontrollers.

Resource	Power	Bandwidth			
0	133	∞	10	5	5
1	130	10	∞	5	5
2	118	5	5	∞	10
3	90	5	5	10	∞

Table 1: Resources used for the execution example.

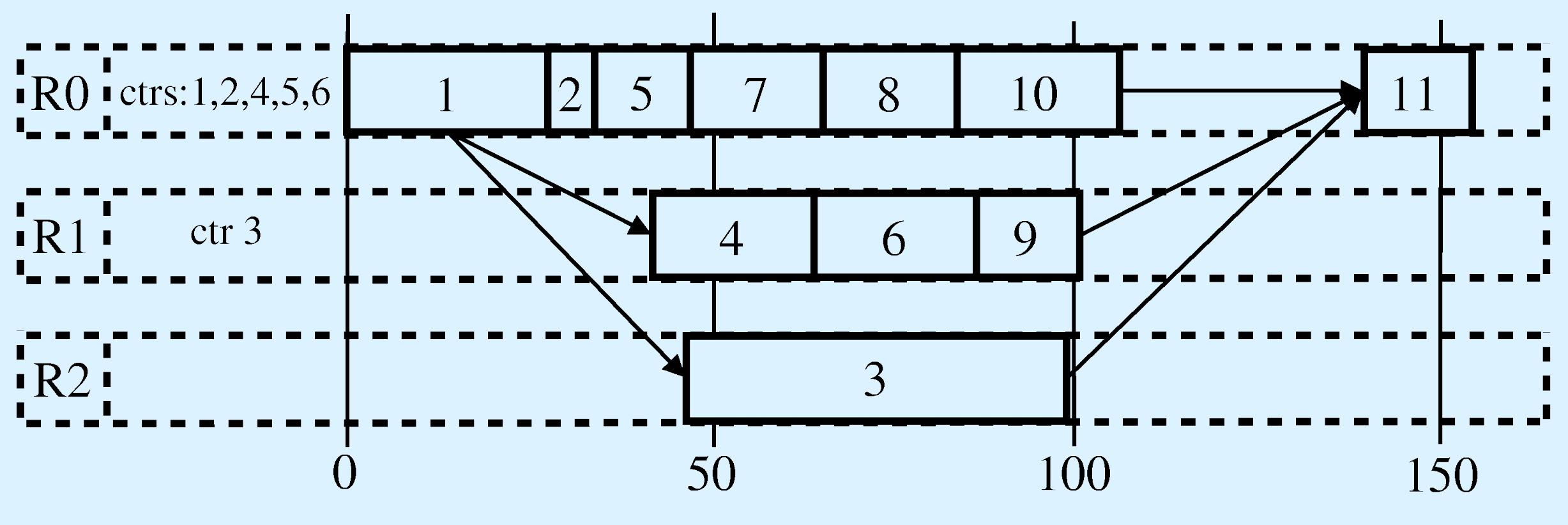

Figure 3: Resulting schedule of the process and its controllers.

EXPERIMENTAL RESULTS

We compared PCH algorithm with HEFT algorithm [2]. The comparison metrics are the number of best schedules of each algorithm, the Schedule Length Ratio (SLR) and the Speedup. The results show that PCH gives good results when controllers are considered, as expected and desired.

$$SLR = \frac{makespan}{\sum_{n_i \in CP} \frac{instructions_{n_i}}{power_{best}}} \qquad Speedup = \frac{\sum_{n_i \in V} \frac{instructions_{n_i}}{power_{best}}}{makespan}$$

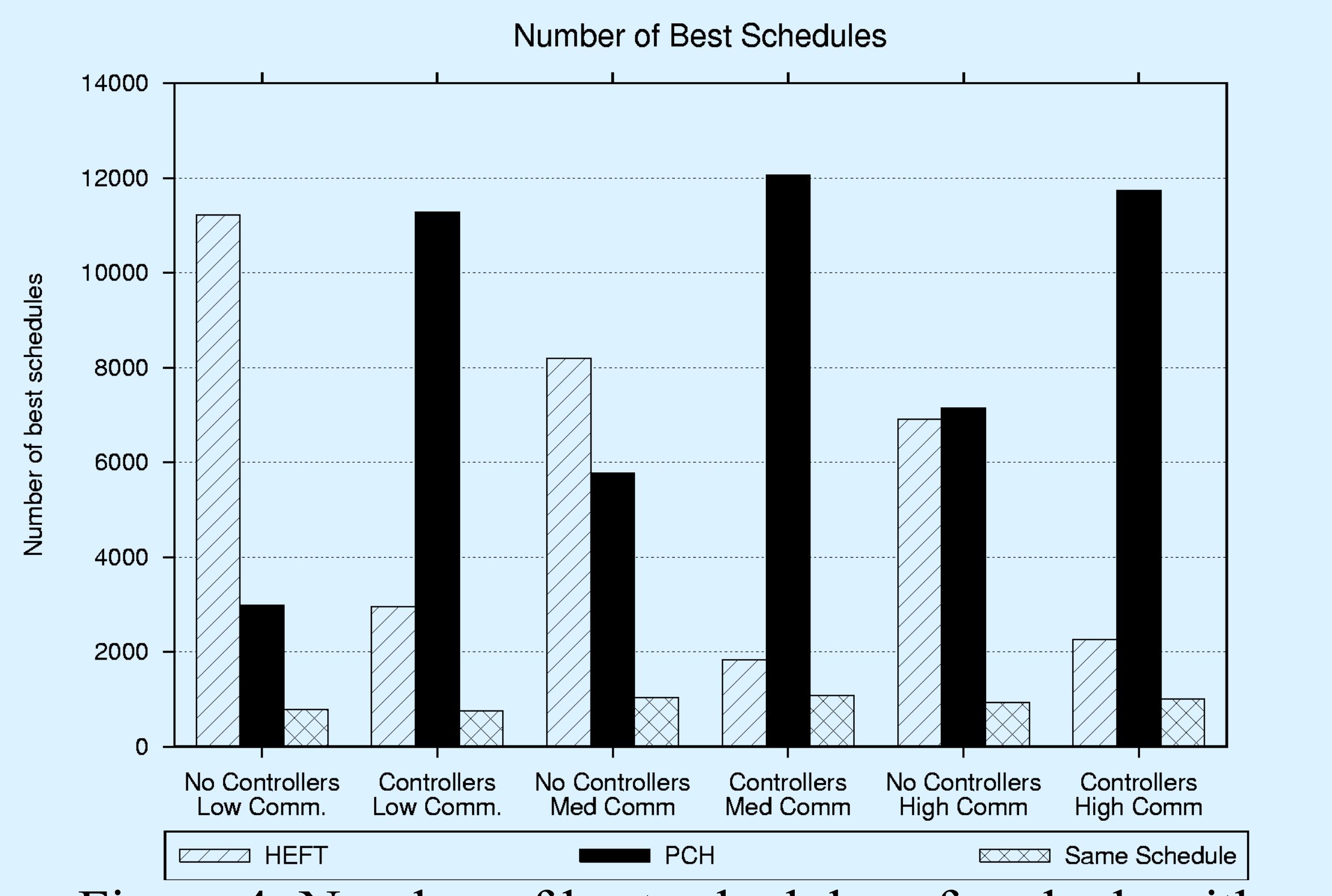

Figure 4: Number of best schedules of each algorithm.

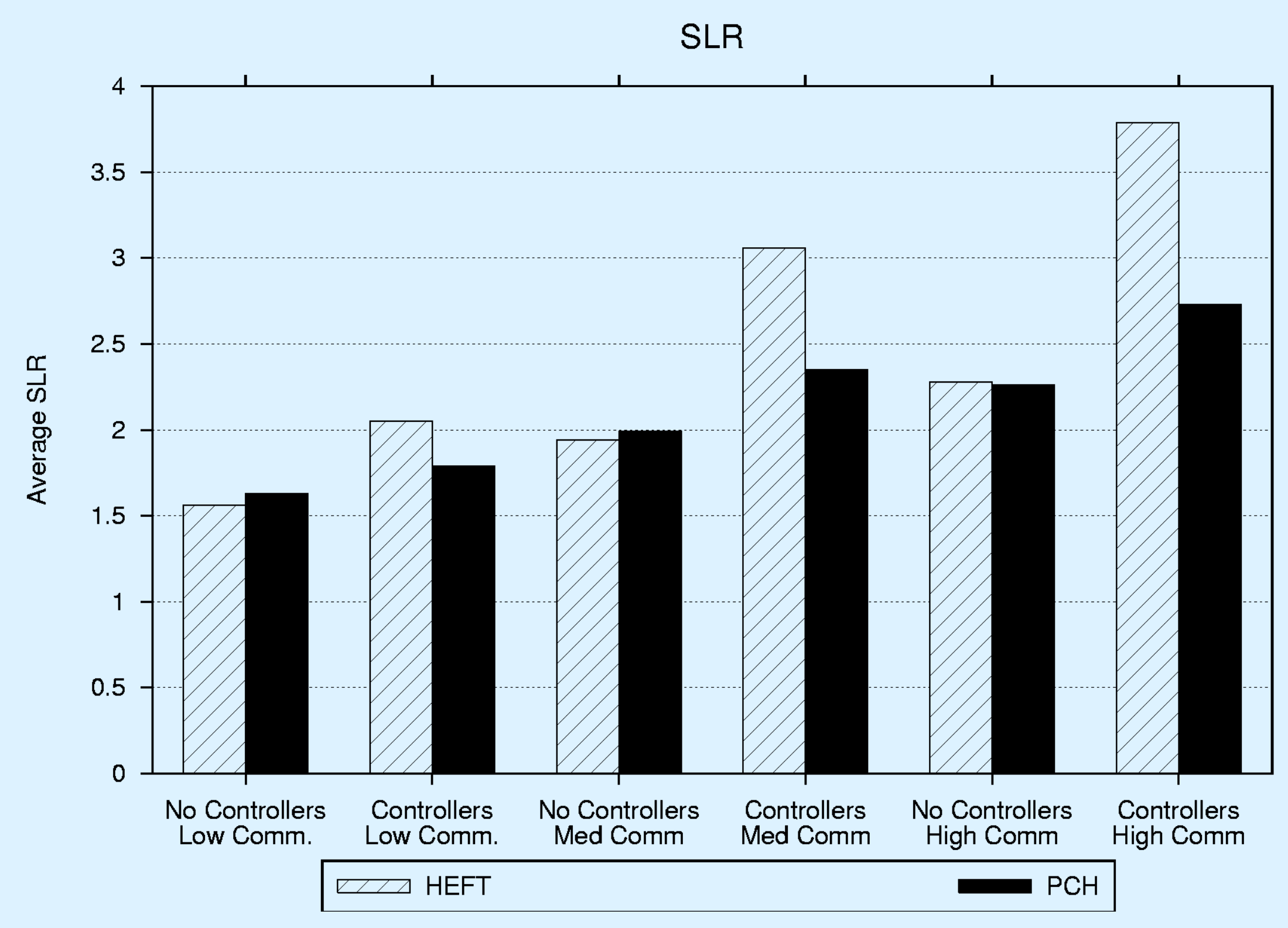

Figure 5: Results of average SLR.

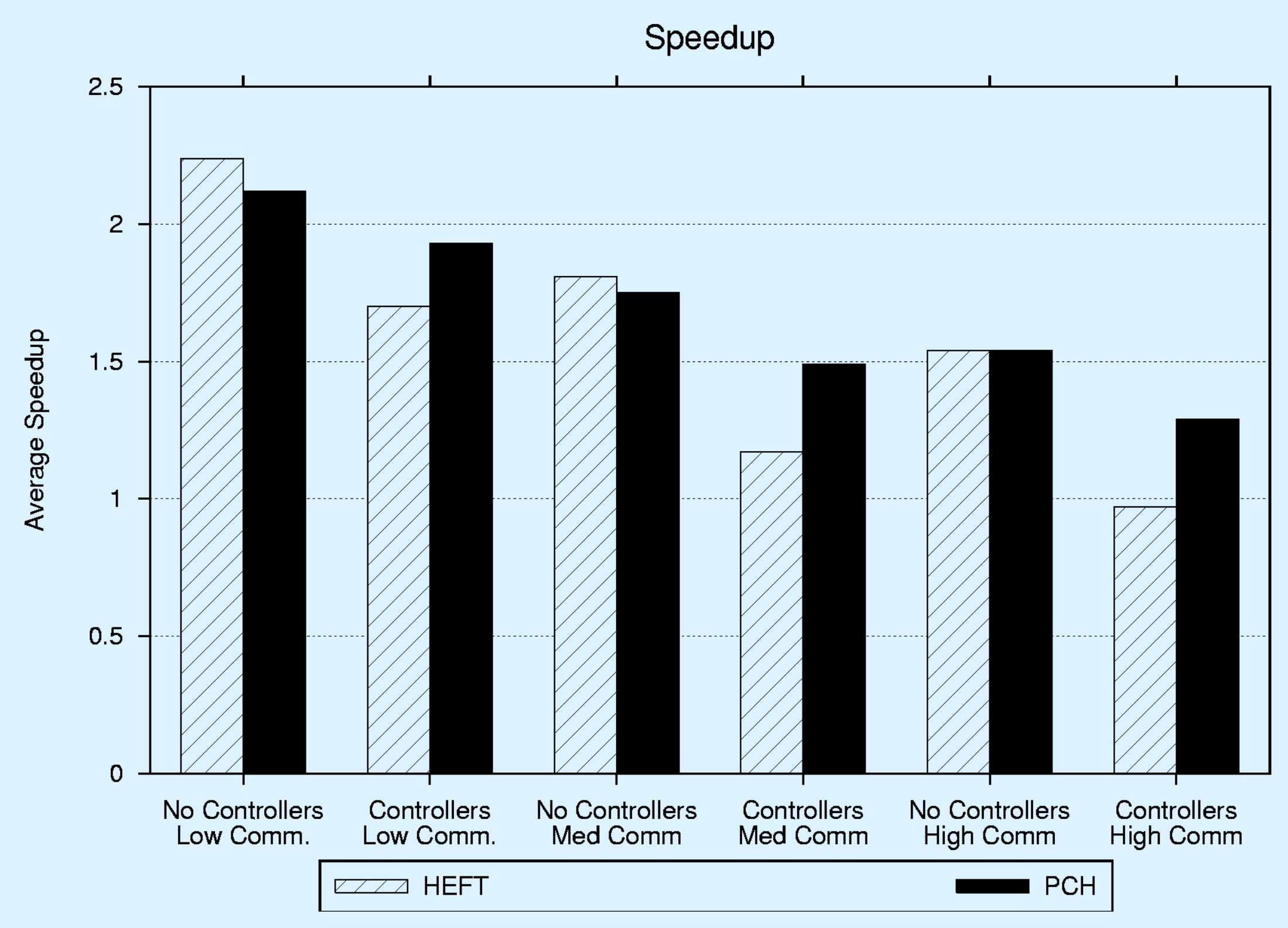

Figure 6: Results of average Speedup.

CONCLUSION

We present a heuristic for task scheduling in the Xavantes grid middleware. For the task graphs supported by Xavantes, the strategy gives good performance when controllers are considered with time complexity $O(pv^3)$.

The authors thank CNPq, and WebMaps and Agroflow projects for the financial support.

REFERENCES

[1] F. R. L. Cicerre, E. R. M. Madeira, and L. E. Buzato. A hierarchical process execution support for grid computing. *To appear in Concurrency and Computation: Practice and Experience*, 2005.
[2] H. Topcuoglu, S. Hariri, and M.-Y. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans. Parallel Distrib. Syst.*, 13(3):260–274, 2002.