AI-Job Scheduling on Systems with Renewable Power Sources

—

Ganesh K. Nileshwar & Uwe Schwiegelshohn

Research Problem

The massive usage of AI training applications
→ increasing tendency of the total power consumption
→ improvements of operational efficiency
→ reductions in inefficient consumption of energy (energy loss & energy waste)
→ reduction in idle power
→ better scheduling of the computing jobs

Scheduling Algorithm

Acceptance

- **Greedy approach:** only rejects a new job if there is no schedule that completes this job on time without changing the allocation of any previously accepted job.
- **Threshold algorithm:** uses a deadline threshold and rejects a new job if its deadline is less than the threshold

Allocation

- **Load balancing:** allocates the new job to the necessary number of least loaded machines
- **BestFit strategy:** selects a set of machines with the largest possible total load that still allows a schedule without any deadline violation
- **MinIdle algorithm :** uses a set of jobs that produce the least amount of additionally enclosed idle time in a valid schedule (for **rigid jobs** with a **high** degree of **parallelism**)
- **Backfilling algorithm:** exploit some enclosed idle time in the schedule

Input Data

- Based on the workload traces from a Google cluster
- Characterized by 4 components:
  - Geometric mean ε
  - Geometric standard deviation σ
  - Original workload
  - Total number of available cores

$$\varepsilon_j = [(d_j - r_j)/p_j] - 1$$

r_j : submission time
p_j : consumed processing time
m_j : degree of parallelism
d_j : job deadline
ε_j : slack value

Algorithm 1 Greedy BestFit

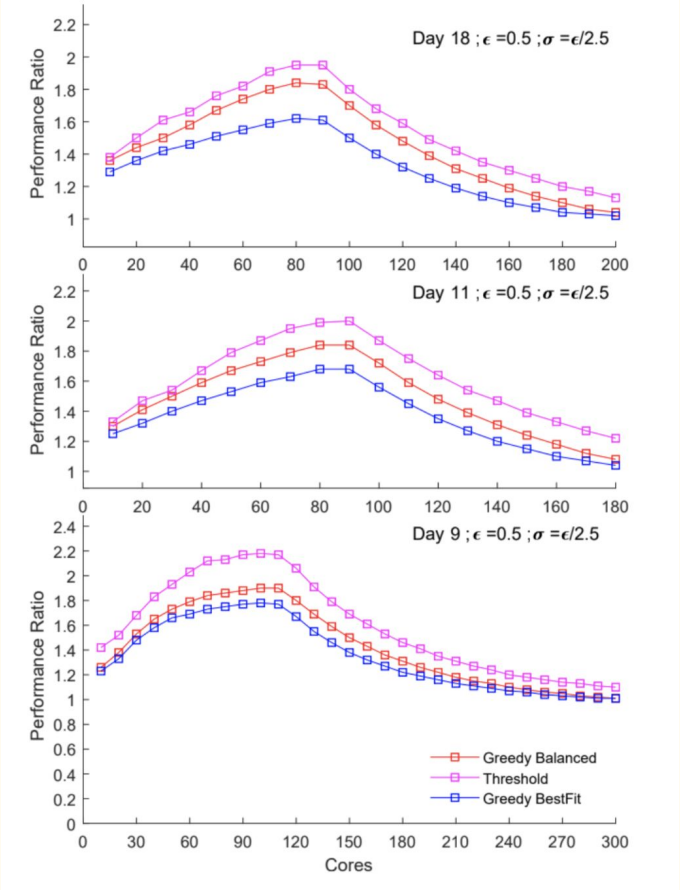
```
1: for the next job j do
2:     update the remaining load for all cores
3:     if the least loaded machine completes j on time then
4:         accept j
5:         determine the most loaded core that completes j on time
6:         start j on this core as early as possible
7:     else
8:         reject j
```

Single core simulation

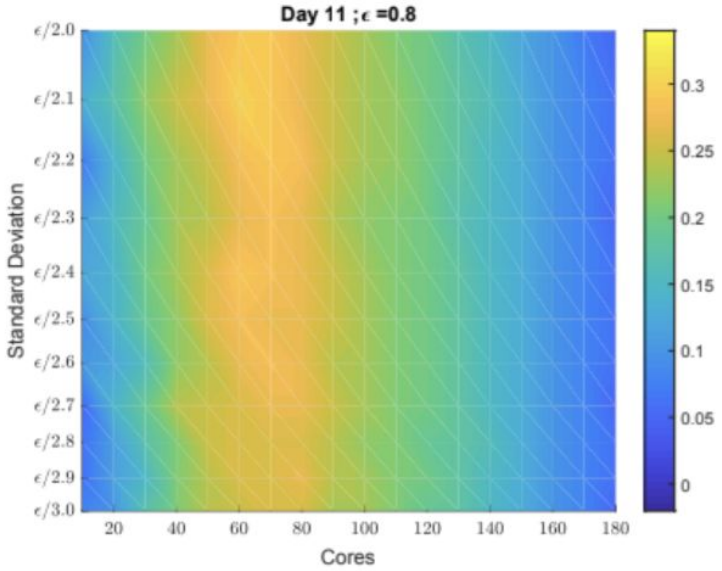
Greedy BestFit performs the best

Terms:

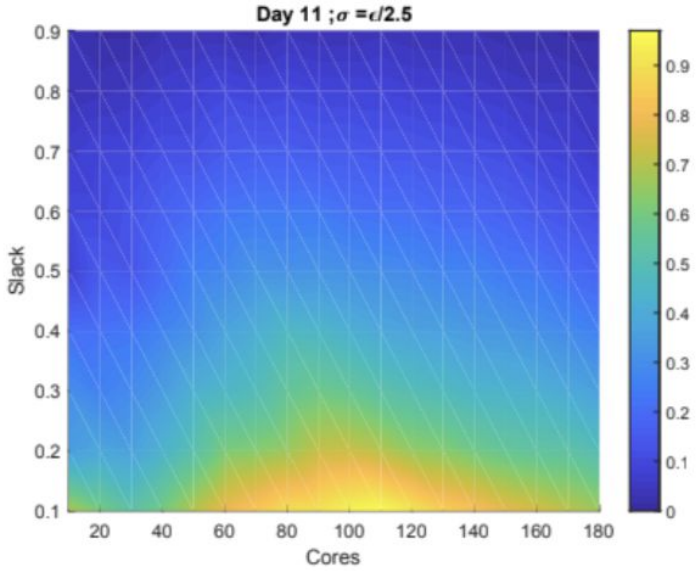
- Performance ratio: Acceptance/allocation (upper load limit/ total accepted load)
- The higher the performance ratio, the worse the performance.
- Parameters: slack value (ε), number of cores, standard deviation (σ)



Standard deviation has no effect on the performance
Greedy BestFit is not better than Threshold at high slack values



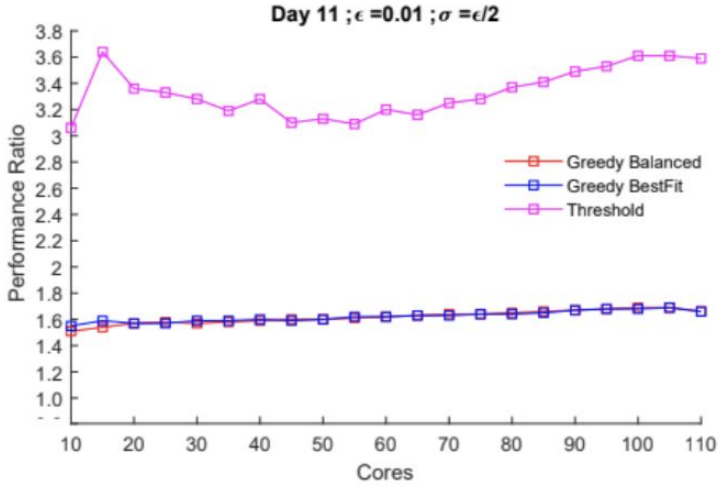
(a) Greedy Balanced - Greedy BestFit



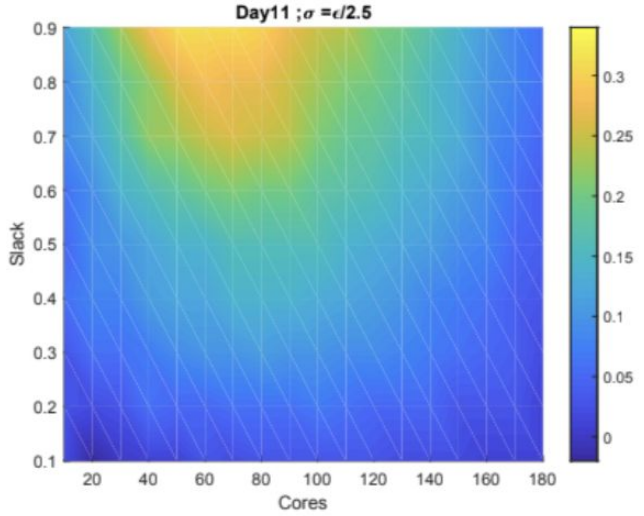
(b) Threshold - Greedy BestFit

Fig. 4: Performance ratio against Greedy BestFit for geometric standard deviation σ and number of cores on Day 11 using target slack $\varepsilon = 0.8$

The performance of Greedy BestFit at low and high core numbers is not better than Greedy Balance



(a) Target slack $\varepsilon = 0.01$ and deviation $\sigma = \varepsilon/2.0$

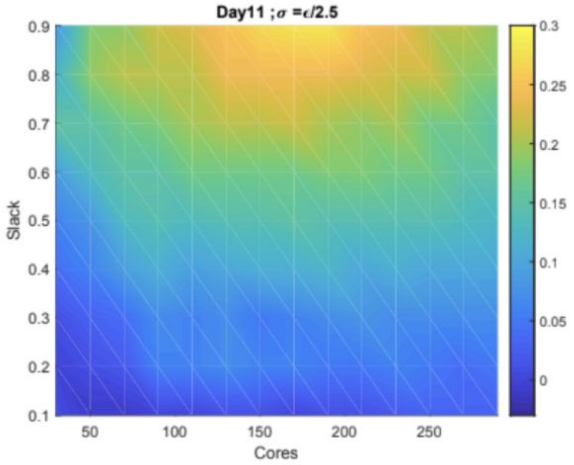


(b) Greedy Balanced - Greedy BestFit for slack ε and deviation $\sigma = \varepsilon/2.5$

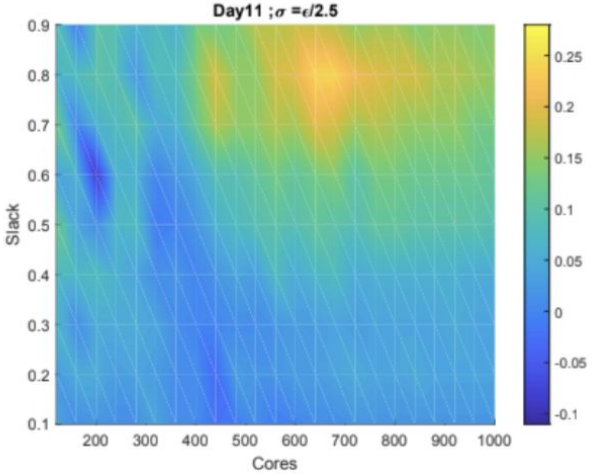
Fig. 5: Performance ratios Threshold, Greedy Balanced, Greedy BestFit on Day 11

Multiple cores simulation

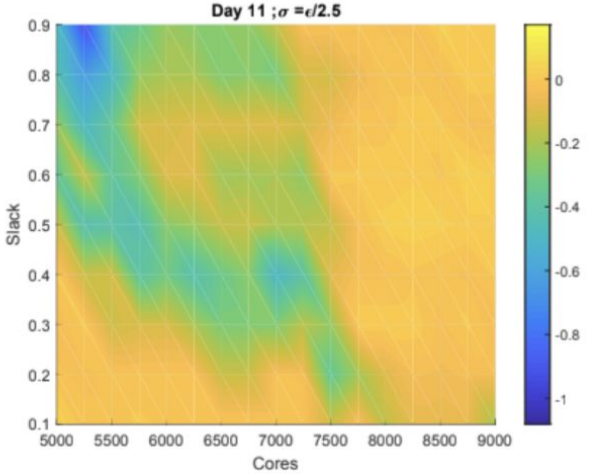
Performance of Greedy BestFit is better than Greedy Balance with parallelism limit of 30 and 120 cores
At 5000 cores, Greedy Balance is better than Greedy BestFit



Parallelism limit 30 cores

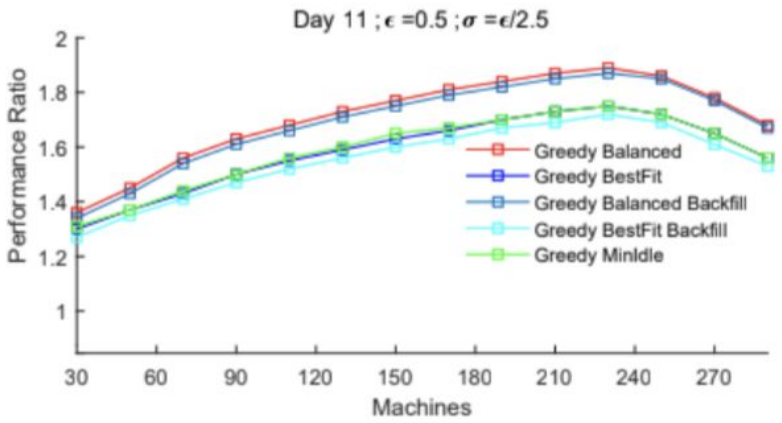


Parallelism limit 120 cores

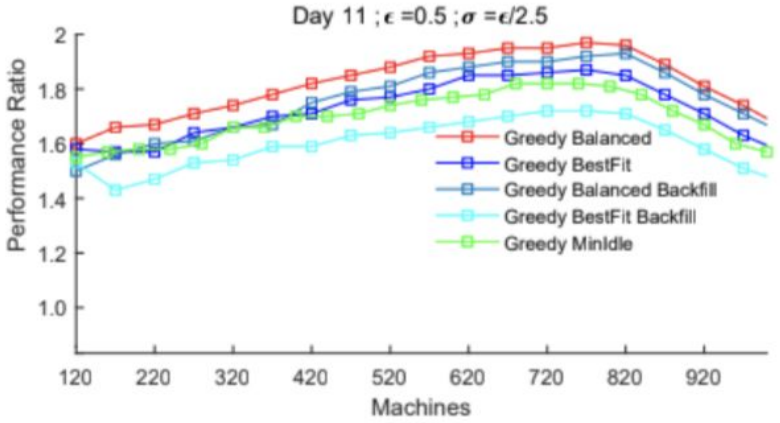


Parallelism limit 5000 cores

Backfilling improves the performance



(a) Parallelism limit 30 cores



(b) Parallelism limit 120 cores

Fig. 8: Performance ratio of Greedy Best Fit and Greedy Balanced with and without backfilling for Day 11, target slack $\varepsilon = 0.5$, and deviation $\sigma = \varepsilon/2.5$.

Results

1. The ***acceptance*** method is important for ***small*** slack values; the ***allocation*** method is important ***large*** slack values.

2. ***Greedy*** approach outperforms threshold algorithm for ***small*** slack values.

3. ***BestFit*** is better than load balancing for jobs required ***low parallelism***.

4. Conservative backfilling & MinIdle perform better than simple allocation methods for rigid jobs with high parallelism but are computationally expensive.

THANK YOU!

References

- Ganesh K. Nileshwar, Uwe Schwiegelshohn. "AI-Job Scheduling on Systems with Renewable Power Sources".