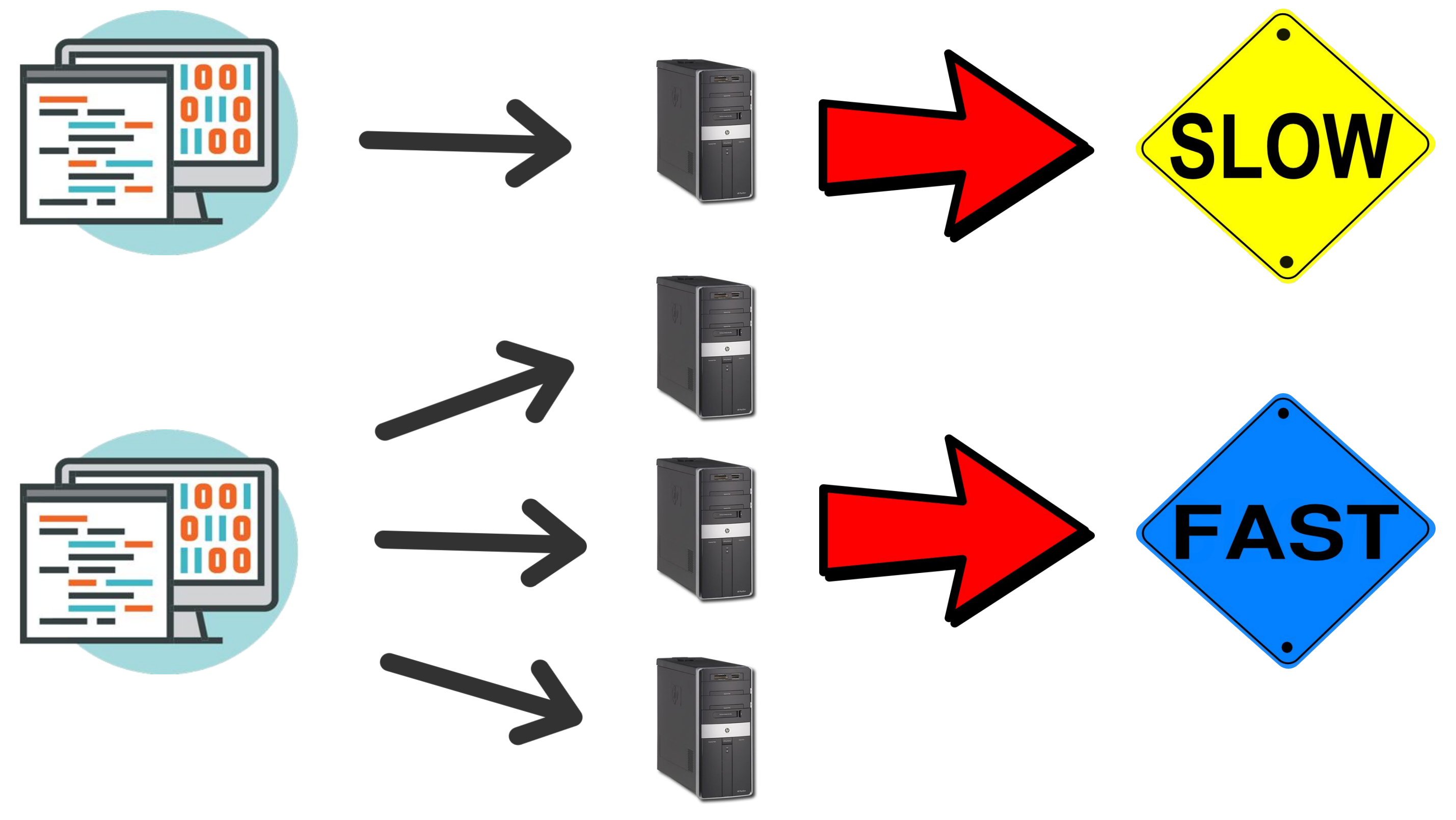
Asymptotic Lower Bounds for Online Scheduling

Loc Tran, Evan Lang, Khoi Le
Denison University, Department of Mathematics and Computer Science
Research Advisor: Nathaniel Kell

Computational Job Scheduling

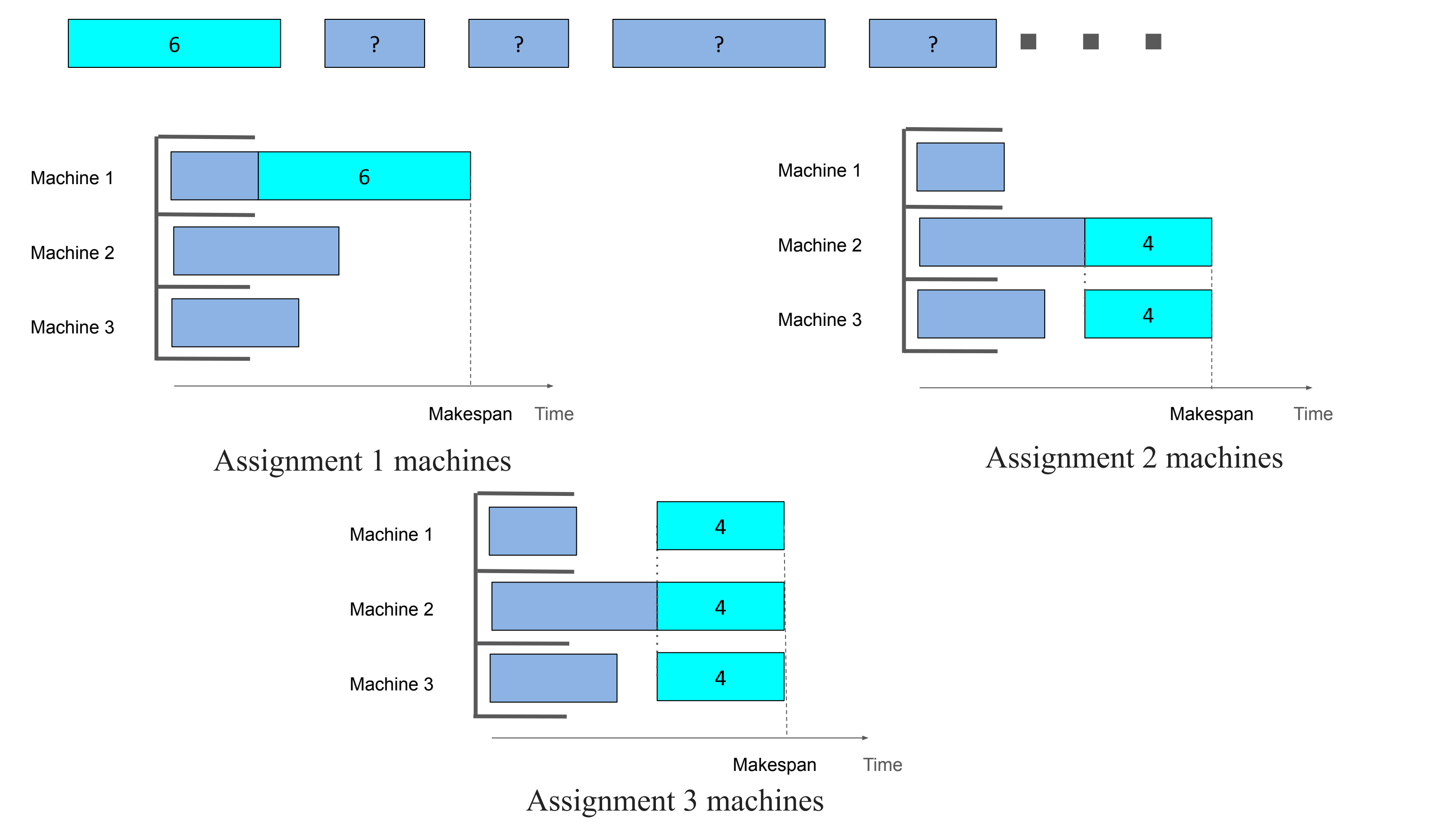
As cloud computing resources become more ubiquitous, it is increasingly important for data centers to be efficient when processing computational tasks from users. In particular, data centers must design algorithms that schedule incoming jobs on its machines such that the overall load is as balanced as possible.

Job Parallelization



Parallelization is a challenge that exists in practice. Data centers and cloud services use parallelization to process large amounts of data in real time. This speeds up the execution time, however, also requires additional overhead (preprocessing, inter-machine communication, etc.). A job scheduling algorithm must account for this tradeoff.

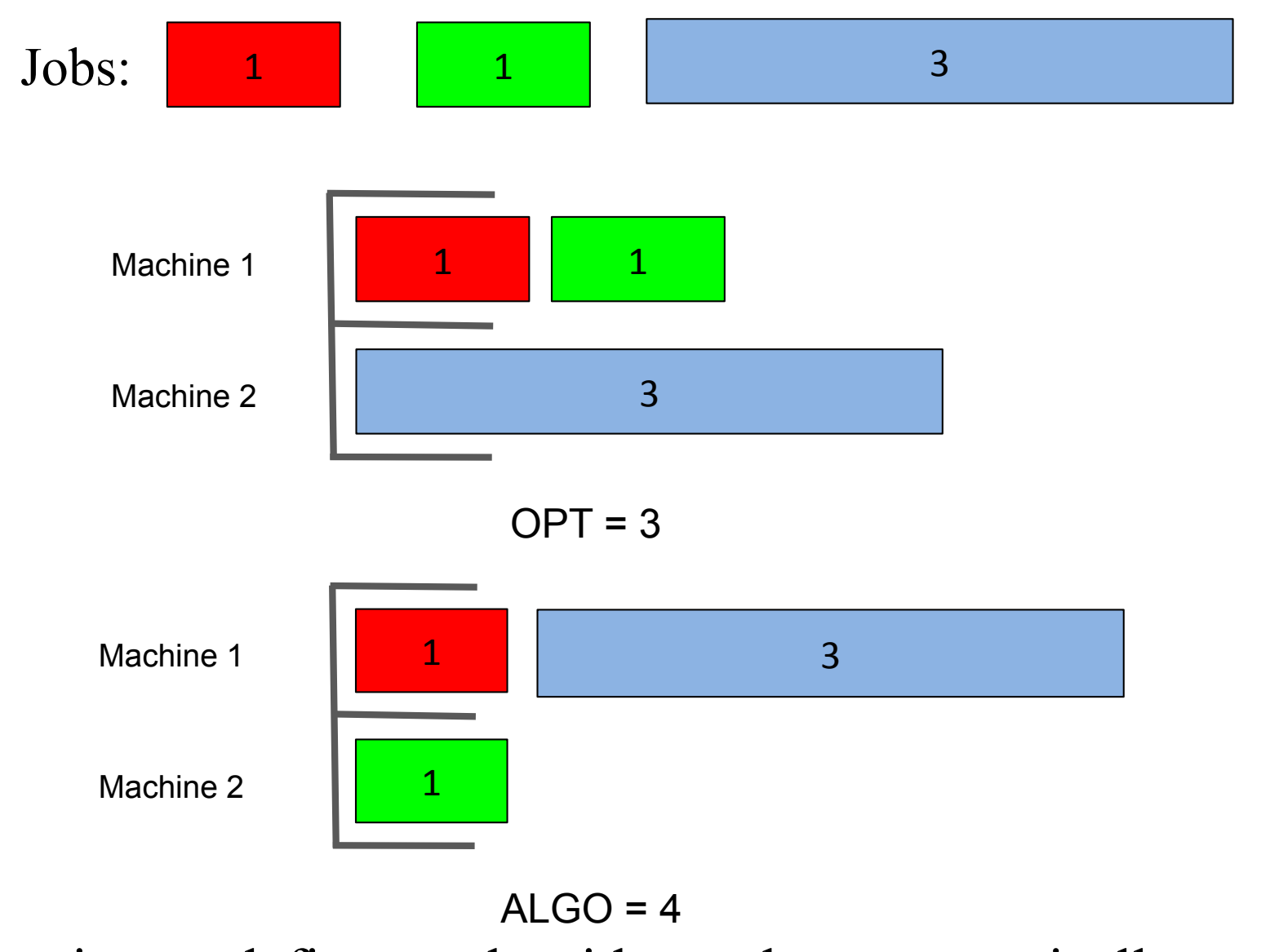
Model Definition



- Jobs are given in an *online* sequence, where the information about jobs in the sequence is unknown, but when a job j arrives, we know its processing time p_j.
- Every machine is identical, and each job is required to be either processed on a single machine or shared among multiple machines (*moldable*). If a job is shared among k_j machines, it must be processed in parallel among these machines and has execution time $p_j/k_j + (k_j-1)c$.
- After the schedule is finished, the *load* on each machine is the time that machine finishes its assigned jobs. The objective is to minimize the maximum load (*makespan*) over every machine.

Performance Metric: Competitive Ratio

We say an algorithm is *OPT* if it is the optimal makespan of a sequence of jobs, and *ALGO* be the algorithm's makespan on that same sequence of jobs. We say an algorithm is *strongly α-competitive* if, for every sequence of job, the quotient *ALGO/OPT* never exceeds α. In other words, $ALGO \leq \alpha \cdot OPT$.



Similar to the strongly competitive ratio, we define an algorithm to be asymptotically α-competitive if $ALGO \leq \alpha \cdot OPT + constant$. Since this is a constant value, it becomes less and less important to the runtime as the number of jobs increases, which is very common to occur in a practical setting.

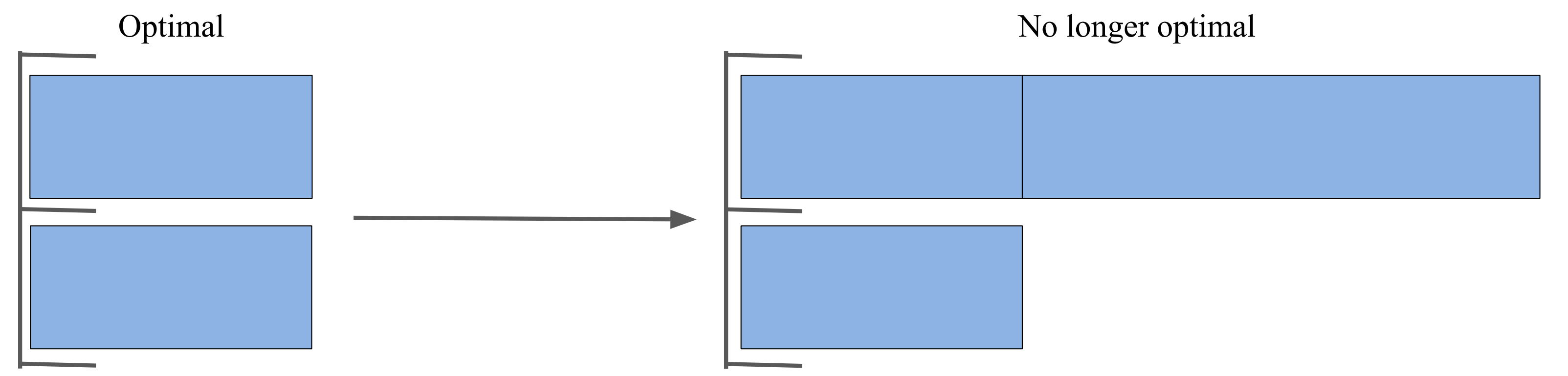
Results

	m = 2		m = 3		m → ∞
Case	**Lower Bound**	**Upper Bound**	**Lower Bound**	**Upper Bound**	**Upper Bound**
Asymptotical	7/6	4/3 [3]	?	5/3 [2]	4 [1]

We worked on the moldable job scheduling problem for the case of two and three machines. For the two machines case, we simply tried to find a lower bound of the asymptotical competitive ratio. In other words, we found a sequence of jobs that for every algorithm, the competitive ratio cannot be lower than a specific number, which we found out to be 7/6.

Online Lower Bound

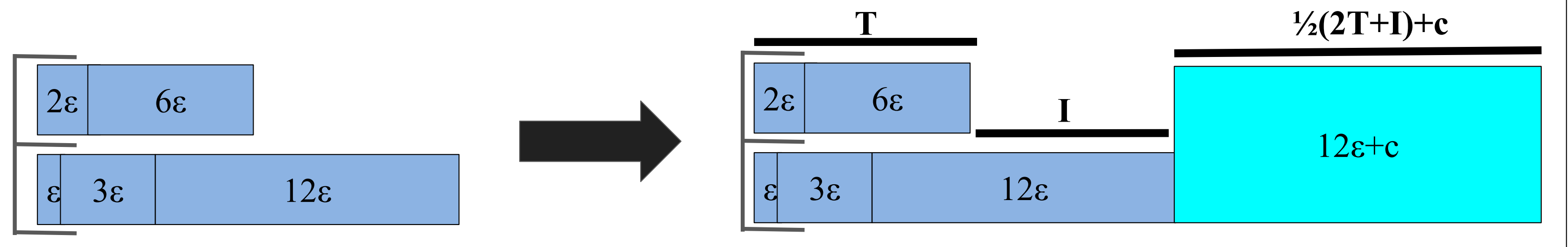
Given that we are in the online model, it is usually impossible for an algorithm to schedule jobs optimally.



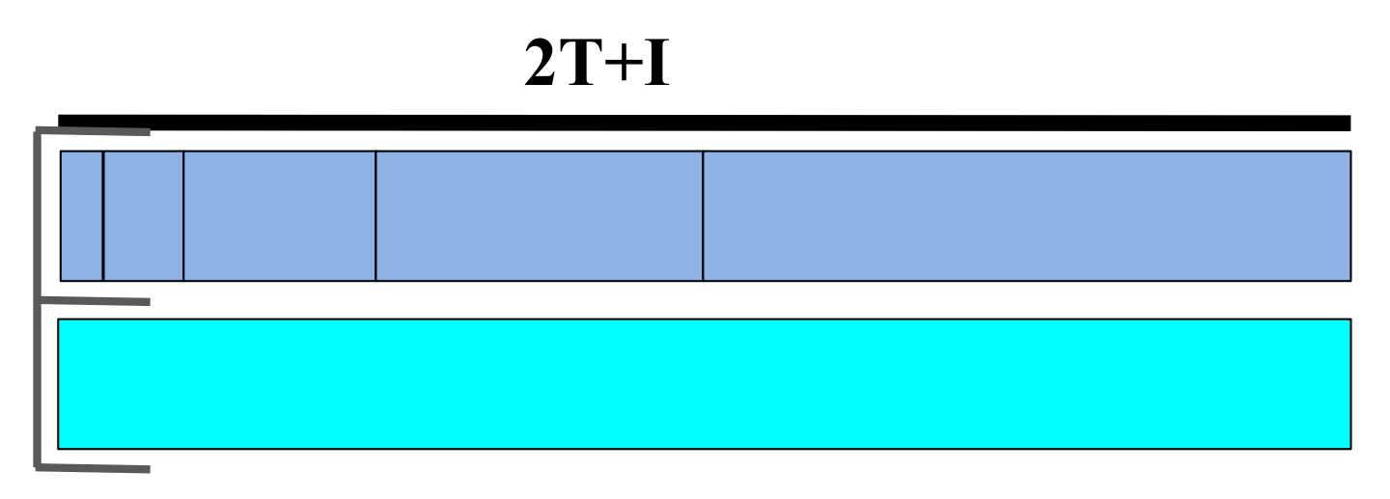
We have found in our research that the best any algorithm can do (the lower bound) must be 4/3 in the standard case, or 7/6 in the moldable case

Proof Sketch

We can consider a game being played between us, an adversary to the algorithm, trying to make the algorithm as inefficient as possible, and the algorithm itself, which can assign any job to either machine, or split. To make the adversary, we will first assigns jobs much smaller than the additive "constant c", which we will call ε. To make the algorithm as inefficient as possible, we then assign jobs that are the size of all jobs before it. Doing this, if the algorithm makes the most efficient choice at any given moment, then the block will always have a 2-1 ratio between the machines, until the algorithm decides to split a job.



As the adversary, we can react to the algorithm splitting a job by simply starting the process over, with a job of size ε. We will then get, through this job scheduling, a series of "blocks" that consist of jobs assigned to 1 machine, before finally being assigned to both machines. If we can then show a lower bounded estimate for all of these blocks, we will have an estimate for the entire schedule.



We can note that, since our initial input ε is much smaller than our splitting cost c, that if the algorithm splits before we reach the possible optimal solution above, that we will always be worse than a 7/6 approximation. In our other cases, we end up with an approximation of (T+I+½(2T+I)+c)/(2T+I), which simplifies to be 7/6+(c/3T) > 7/6.

References

[1] Jessen T. Havill and Weizhen Mao. Competitive online scheduling of perfectly malleable jobs with setup times. European Journal of Operational Research, 187(3):1126–1142, 2008.
[2] Jessen T. Havill. Online malleable job scheduling for m ≤ 3. Information Processing Letters, 111(1):31–35, 2010.
[3] Nathaniel Kell and Jessen Havill. Improved upper bounds for online malleable job scheduling. Journal of Scheduling, 18(4):393–410, Aug 2015.
[4] Debasis Dwibedy and Rakesh Mohanty. Online scheduling with makespan minimization: State of the art results, research challenges and open problems. CoRR, abs/2001.04698, 2020.

Acknowledgements

This project was funded by the J. Reid & Polly Anderson Endowment and Dr. Joan Krone Summer Scholars Endowment at Denison University.