Approximation Algorithms for Online Scheduling

Deterministic

Algorithm

```
Machine(label)
        addJob(nextJob)
                set startTime of nextJob to this machine's nextIdleTime
                add nextJob to this machine
                increment nextIdleTime by nextJob's duration
                set nextJob's endTime to this machine's nextIdleTime    //optional

GreedyScheduler(machineCount, jobsType)
        m = machineCount
        machines = [ ]
        jobs = [ ]

        initialiseMachines()
                for i = 0 → m
                        add "Machine(i+1)" to machines
        generateJobs(jobsType)
                for i = 0→jobsType.length
                        add new Job of Type jobsType[i] to jobs
        solve()
                Machine recipient = null
                for i = 0→machines.length       //the first m jobs
                        recipient = machines[i]
                        add jobs[i] to recipient
                for i = machines.length → jobs.length
                        recipient = getMachineWithLeastWork()
                        add jobs[i] to recipient

        getMachineWithLeastWork() {
                Machine result = machines [0]
                for int i = 1 → machines.length
                        Machine candidate = machines[i]
                        if candidate.nextIdleTime < result.nextIdleTime
                                result = candidate;
                return result;
```

Ratio

Let j be the index of a job with maximum completion time. The starting time of this job is at most $\frac{1}{m}\sum_i p_i \leq OPT$. Since $p_j \leq OPT$, we get that the processing times on all machines is bounded by $\frac{1}{m}\sum_i p_i + max_i\{p_i\} \leq 2\ OPT$.

Lower Bound

OPT: $\frac{1}{m}\sum_i p_i$ and $max_i\{p_i\}$

Proof

Let i be a machine with maximum completion time, and let j be the index of the last job assigned to i by the algorithm. Let $s_{i,j}$ be the sum of all times for jobs prior to j that are assigned to i. (This may be thought of as the time that job j begins on machine i). The algorithm assigned j to the machine with the least amount of work, hence all other machines i' at this point have larger finish time when t_j is added to it. Therefore $s_{i,j} \leq \frac{1}{m}\sum_{j=1}^{n} t_j$ i.e. $s_{i,j}$ is less 1/m of the total time of all jobs (recall m is the number of machines). Notice $B = \frac{1}{m}\sum_{j=1}^{n} t_j \leq p^*$, the completion time for an optimal solution, as the sum corresponds to the case where every machine takes exactly the same fraction of time to complete. Thus the completion time for machine i is $s_{i,j} + t_j \leq p^* + p^* = 2p^*$. So the maximum completion time is at most twice that of an optimal solution.

Probabilistic

Algorithm

```
Machine(label)
        addJob(nextJob)
                set startTime of nextJob to this machine's nextIdleTime
                add nextJob to this machine
                increment nextIdleTime by nextJob's duration
                set nextJob's endTime to this machine's nextIdleTime     //optional

ProbabilisticScheduler(machineCount, jobsType)
        m = machineCount
        processedSoFar = 0
        machines = [ ]
        jobs = [ ]

        initialiseMachines()
                for i = 0 → m
                        add "Machine(i+1)" to machines
        generateJobs(jobsType)
                for i = 0→jobsType.length
                        add new Job of Type jobsType[i] to jobs
        solve()
                for i = 0→jobs.length
                        Machine recipient = getRandomMachine(random limit)
                        add jobs[i] to recipient
                        increment processedSoFar by duration of jobs[i]
        getRandomMachine(random limit)
                Machine recipient = null
                do once and repeat until (candidateProbability >= random limit)
                        recipient = machines[get random integer 0….machines.length]
                        if processedSoFar is 0
                                candidateProbability = 0
                        else
                                candidateProbability = recipient.nextIdleTime /processedSoFar

                return recipient
```

Ratio

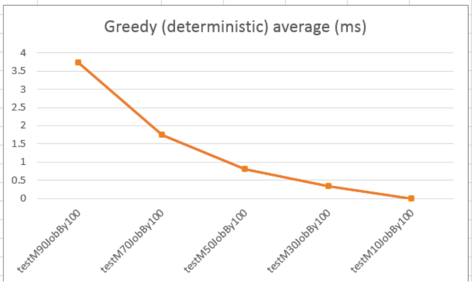
In the discrete case, when the set of time values is the set of non-negative integers, then the infimum in the definitions of the probabilistic makespan of s and the probabilistic minimum makespan is the same as minimum. We prove that there exists some solution which makes it sufficiently certain that all jobs finish by the probabilistic minimum makespan

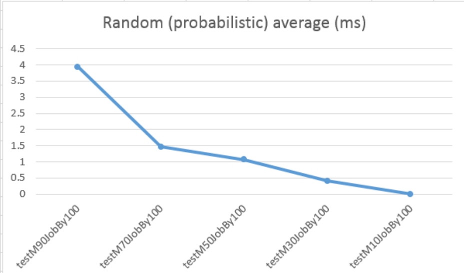
Lower Bound

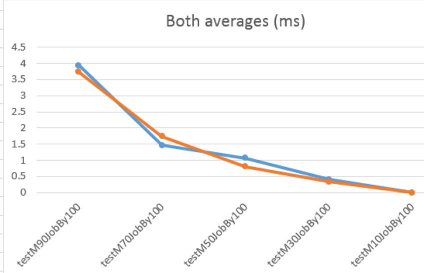
We show that a lower bound for the minimum make span can be found by solving a particular deterministic problem. A common approach is to generate a deterministic problem by replacing each random duration by the mean of the distribution. As we show, under certain conditions, the minimum makespan of this deterministic algorithm is a lower bound for the probabilistic minimum makespan. For instance, in the example, the minimum makespan of such a deterministic algorithm is 45, and the probabilistic minimum makespan is about 55. However, an obvious weakness with this approach is that it does not take into account the spreads of the distributions.

Results

Greedy																	
id	name	times															average (ms)
1	testM90JobBy100	19	3	3	3	3	4	3	2	2	2	2	1	3	3	3	3.733333
2	testM70JobBy100	2	2	2	2	1	2	1	2	1	2	1	2	2	2	2	1.733333
3	testM50JobBy100	1	1	0	0	1	1	1	1	1	1	0	1	1	1	1	0.8
4	testM30JobBy100	0	0	0	0	0	1	1	0	0	1	0	1	1	0	0	0.333333
5	testM10JobBy100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Random																	
id	name	times															average (ms)
1	testM90JobBy100	9	6	4	4	5	5	4	3	3	3	3	4	2	2	2	3.933333
2	testM70JobBy100	1	1	2	1	1	2	1	2	2	1	2	1	2	1	2	1.466667
3	testM50JobBy100	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1.066667
4	testM30JobBy100	1	1	0	0	0	0	0	0	0	1	0	0	1	1	1	0.4
5	testM10JobBy100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0







As we can see from the graphed results, the probabilistic algorithm slightly outperforms the deterministic algorithm when it comes to m = 70 and jobs = 7000. The random algorithm is also almost equal to the greedy algorithm, in terms of performance, on other parameter settings. This advantage is as expected that the probabilistic random algorithm outperforms the greedy algorithm.