On-Line Restricted Assignment of Temporary Tasks with Unknown Durations

Amitai Armon * Yossi Azar † Leah Epstein ‡ Oded Regev §

Keywords: on-line algorithms, competitiveness, temporary tasks

Abstract

We consider load balancing of temporary tasks on m machines in the restricted assignment model. It is known that the best competitive ratio for this problem is $\Theta(\sqrt{m})$. This bound is *not* achieved by the greedy algorithm whose competitive ratio is known to be $\Omega(m^{\frac{2}{3}})$. We give an alternative analysis to the greedy algorithm which is better than the known analysis for relatively small values of m. We also show that for a small number of machines, namely $m \leq 5$, the greedy algorithm is optimal.

1 Introduction

The problem: We study the following non-preemptive on-line load balancing problem. The input consists of a stream of temporary tasks (jobs). Each job j has an arrival time and a departure time (which is unknown at the arrival time and becomes known simply when the job departs), and a set of permitted machines $M(j)$. The job should be assigned to one machine immediately upon its arrival. A job j has a weight w_j. If it is assigned to machine i ($i \in M(J)$), it increases the load of machine i by w_j, for the duration of job j. The load on a machine at a certain time is the sum of the loads caused by the jobs assigned to it at that time. The goal is to minimize the maximum

*Department of Computer Science, Tel Aviv University, Tel-Aviv, 69978, Israel. E-Mail: armon@tau.ac.il.

†Department of Computer Science, Tel Aviv University, Tel-Aviv, 69978, Israel. E-Mail: azar@tau.ac.il. Research supported in part by the Israel Science Foundation and by a grant of the European Commission.

‡School of Computer and Media Sciences, The Interdisciplinary Center, Herzliya, Israel. E-Mail: epstein.leah@idc.ac.il.

§Institute for Advanced Study, Princeton, NJ. E-Mail: odedr@ias.edu. Research supported in part by NSF grant CCR-9987845

load over machines and time. Note that the load and the time are two separate axes of the problem. In this paper we are interested in the above problem for cases where m is a small constant.

Preliminaries: In this paper we consider a greedy algorithm "$GREEDY$". This is the algorithm which assigns each job to the minimum loaded admissible machine (in case of a tie - the machine with smallest index is selected).

The quality of an on-line algorithm, is measured by the *competitive ratio* that is the worst case ratio between the cost (maximum load over machines and time, in this paper) of the on-line algorithm (the on-line algorithm is denoted by ON) and the cost of an optimal off-line algorithm which knows the whole sequence in advance (the optimal off-line algorithm is denoted by OPT). The cost of an algorithm A is denoted by C_A.

Previous work: The results in [3] (and later in [8]) show a lower bound of $\Omega(\sqrt{m})$ on the competitive ratio that any on-line algorithm (deterministic or randomized) may have for restricted assignment of temporary jobs with unknown durations. An on-line algorithm for this problem with a competitive ratio of $O(\sqrt{m})$, was later presented in [5] (the "Robin-Hood" algorithm) thereby proving that the lower bound of $\Omega(\sqrt{m})$ is tight.

A more generalized load balancing problem is the case of unrelated machines. In that model a job j and a machine i have a weight $w_j(i)$ which the job j causes the machine i for the duration it is assigned to it. The restricted assignment model is the special case where all those values are either w_j or ∞. For this more general problem the best known algorithm achieves competitive ratio m. This Algorithm simply assigns a job to a machine with minimum $w_j(i)$. Surprisingly, a lower bound of $\Omega(m/\log m)$ given in [1] shows that this algorithm has almost optimal competitive ratio.

On the other hand, the problem of temporary tasks assignment on identical machines is a special case of our problem. In that case any job may be assigned to any machine. The greedy algorithm for identical machines and permanent jobs was presented and studied by Graham [7]. His analysis of $GREEDY$ holds for temporary jobs as well and gives competitive ratio of $2 - \frac{1}{m}$. [4] showed that this bound is tight. The more general versions (restricted assignment and unrelated machines) have both competitive ratio of $\Theta(\log m)$ for permanent jobs [6, 2]. For restricted assignment, this bound is achieved by $GREEDY$ [6].

Our results: In this paper we show that $GREEDY$ is an optimal on-line algorithm for the case when the number of machines is small: $m \leq 5$. This is in contrast to the known non-optimal performance of $GREEDY$ for a general m (The tight analysis of $GREEDY$ for large enough m yields the competitive ratio $\frac{(3n)^{2/3}}{2}(1 + o(1))$ [3], [5]).

We perform a new and simple analysis of $GREEDY$, which gives better upper bounds (better than the upper bounds given by "Robin-Hood" [5]) for small m ($m < 14$). Finally we show that $GREEDY$ is optimal for 3, 4, and 5 machines, by proving matching (deterministic) lower bounds for these three cases. We show that the values of the best possible competitive ratios in those three cases are $3, 3.5$ and 4 (respectively).

The best known lower bound for this problem for an arbitrary m is $\lfloor\sqrt{2m}\rfloor$ [3], and the best known upper bound for an arbitrary m is $2\sqrt{m}+1$ [5] (or trivially m, if m is smaller than that). For 3 machines, these bounds imply a lower bound of 2 and an upper bound of 3, for 4 machines a lower bound of 2 and an upper bound of 4, and for 5 machines a lower bound of 3 and an upper bound of 5. The results in this paper improve the lower bounds for the cases $m = 3, 4, 5$ and the upper bound for the cases $m = 4, 5$.

For the case of 2 machines, it is easily seen that $GREEDY$ is an optimal 2-competitive algorithm (follows directly from the known results). Adding this to our results proves that $GREEDY$ is optimal for $m \leq 5$.

2 Upper Bound

In this section we present a general result whcih is valid for any m. It has special interest for small values of m where it improves previous results.

Theorem 2.1 *The greedy algorithm achieves a competitive ratio of $\frac{1}{2}(m+3)$ for the problem of restricted assignment of temporary tasks with unknown durations.*

Proof: Given an input sequence for our problem, we denote the maximum load that OPT ever achieves for that sequence by C_{OPT}, and the maximum load that $GREEDY$ ever achieves for that sequence by C_{ON}. We will show that $C_{ON} \leq \frac{1}{2}(m+3)\cdot C_{OPT}$, and thus the required competitive ratio will be proven. Note that C_{OPT} is no smaller than the momentary load that OPT has at any certain time.

Let us consider the first moment in which $GREEDY$ reached its maximum load C_{ON}. We denote the machine on which $GREEDY$ achieved this load at that moment by machine i. Out of the active jobs on machine i at that moment, consider the last job that had more than one admissible machine, and denote it by job j (i.e. this is the last active job that could also be assigned elsewhere). Denote its weight by y. Obviously, $C_{OPT} \geq y$, since OPT had to assign job j to one of the machines. According to our definition of job j, all the jobs that $GREEDY$ assigned to machine i after job j and are still active at this moment had no other admissible machine. Let us denote the total load of these "no choice jobs" by z. It is obvious that OPT also had to assign all these

jobs to machine i, and they are all active at this certain moment, so $C_{OPT} \geq z$.

Now consider the moment in which job j was assigned to machine i. We denote the load on machine i at that moment by x. The fact that $GREEDY$ chose to assign job j to machine i, despite having another alternative, means that every other alternative had a load of at least x. This means that there was at least one more machine with a load of at least x. So at the moment in which job j was assigned, the total volume of active jobs was at least $2x + y$. The best that OPT could do was to equally divide this volume between the m machines. So we obtain: $C_{OPT} \geq (2x + y)/m$. Or: $m \cdot C_{OPT} \geq 2x + y$. Summing this last inequality with $C_{OPT} \geq y$, we obtain: $(m + 1) \cdot C_{OPT} \geq 2x + 2y$. As we defined it, clearly $C_{ON} \leq z + y + x$ (this is not an equality, since some of the jobs that were active when job j was assigned might have departed). So we obtain: $C_{ON} \leq z + y + x \leq C_{OPT} + \frac{1}{2}(m + 1) \cdot C_{OPT} = \frac{1}{2}(m + 3) \cdot C_{OPT}$, which completes the proof. ■

3 Lower Bounds

We continue by showing the tight lower bounds for 3, 4, and 5 machines, which match the performance that we proved for the greedy algorithm in these cases. We start with the lower bound for 3 machines, which is the simplest to prove, and later extend those ideas for 4 and 5 machines.

Theorem 3.1 *Let ON be an on-line algorithm for restricted assignment of temporary tasks with unknown durations on 3 machines. Then the competitive ratio of ON is at least 3.*

Proof: We describe a sequence of unit jobs, such that ON must reach a maximum load of 3. We will show that an offline algorithm OFF can maintain a maximum load of 1 throughout that sequence. This will yield the required competitive ratio.

We start with two jobs, which are admissible to any machine. If ON assigns both of them to the same machine, w.l.o.g. machine 1, then OFF assigns them to machines $2, 3$ respectively. Now job 3 arrives, and can only be assigned to machine 1. Thus, ON reaches a load of 3 on machine 1. OFF has exactly one job on each machine, so it remains with a maximum load of 1, and the ratio is 3 as required. Therefore, ON must assign each of the first two jobs to a different machine. We can assume w.l.o.g. that ON assigns them to machines 1 and 2, respectively. Now job 3 arrives, and it is admissible only to machines 1 and 2. Again, we can assume w.l.o.g. that ON assigns it to machine 1. On the other hand, OFF assigns job 1 to machine 3, job 2 to machine 1 and job

3 to machine 2. Now job 2 departs (emptying machine 1 for OFF), and job 4 arrives, and can only be assigned to machine 1. So ON reaches a load of 3 on machine 1, while OFF still maintains a maximum load of 1. Thus the required competitive ratio has been proven. ■

Next we consider the case of four machines.

Theorem 3.2 *Let ON be an on-line algorithm for restricted assignment of temporary tasks with unknown durations on 4 machines. Then the competitive ratio of ON is at least 3.5.*

Proof: Our sequence will force ON to reach a maximum load of 7, while OFF maintains a maximum load of 2 throughout the sequence. We start with seven unit jobs (jobs $1, ..., 7$), which can be assigned to any of the machines. OFF can assign a maximum of two jobs to each machine and have a maximum load of 2. We divide the possible assignments of ON into two different cases.

Case 1: ON assigns the first seven jobs to less than 3 machines. Obviously, if ON assigns all the jobs to a single machine, then it immediately reaches a load of 7 while OFF maintains a maximum load of 2, and we are done. So we should only consider the case where ON assigns the jobs to two different machines. Again, there are two options.

Case 1.1: ON assigns 5 or 6 jobs to a single machine. We can assume w.l.o.g. that this single machine is machine 1, and that the rest of the jobs (one or two) were assigned to machine 2. OFF assigns to machines $2, 3$ and 4 the jobs which ON assigned to machine 1, without exceeding the maximum of two jobs on each machine. It assigns to machine 1 the one or two jobs that ON assigned to machine 2. Now the one or two jobs that ON assigned to machine 2 depart, and machine 1 becomes empty for OFF. Next arrives a job of weight 2 which can only be assigned to machine 1. Thus ON reaches a load of at least 7 on machine 1, while OFF maintains a maximum load of 2, and we are done.

Case 1.2: ON assigns a maximum of 4 jobs to each machine. This means that ON assigns 4 jobs to one machine, and 3 jobs to another machine. Without loss of generality, we assume that ON assigns jobs $1, ..., 4$ to machine 1 and jobs $5, 6$ and 7 to machine 2. Now job 1 departs from machine 1, and job 8 arrives. Job 8 has a weight of 2, and can only be assigned to machine 1 or machine 2. We assume w.l.o.g. that ON assigns it to machine 1 (which reaches a load of 5). OFF assigns jobs $2, 3$ to machine 3, jobs $4, 5$ to machine 4, and jobs $6, 7$ to machine 1. OFF

assigns job 1 (which departs before the arrival of job 8) to machine 2, so when job 8 arrives OFF assigns it to machine 2 without exceeding the maximum load of 2. Now jobs $6, 7$ depart, leaving machine 1 empty for OFF (and still with a load of 5 for ON). The next arriving job, job 9 has a weight of 2 and can only be assigned to machine 1. So ON reaches a load of 7 on machine 1, while OFF still maintains the maximum load of 2. Therefore, the ratio of maximum loads reaches 3.5 in this case as well.

Case 2: ON assigns the first seven jobs to at least 3 machines. Without loss of generality, we can assume that machines $1, 2$, and 3 have at least one job each. Now four jobs depart, and only one job on each of the first 3 machines remains active. We can assume that these 3 jobs are jobs $1, 2$, and 3, respectively. Next, jobs 8 and 9 arrive. Both of them have a weight of 2 and can be assigned to any of the machines $1, 2$, and 3. There are two cases:

Case 2.1: ON assigns jobs 8 and 9 to the same machine. This brings us to a case similar to **case 1.1** discussed above: ON has a machine with a load of 5 (w.l.o.g. machine 1), while OFF can assign the jobs which ON assigned to machine 1 to the other machines, without exceeding the load of 2. OFF assigns jobs 2 and 3 (which ON assigned to machines 2 and 3) to machine 1. It assigns job 1 to machine 4, job 8 to machine 2, and job 9 to machine 3. Now jobs 2 and 3 depart, leaving machine 1 empty for OFF (and still with a load of 5 for ON). Next, a job of weight 2 which can only be assigned to machine 1 arrives. This makes the load of machine 1 equal 7 for ON, while OFF maintains a maximum load of 2, and the ratio is 3.5 as required.

Case 2.2: ON assigns jobs 8 and 9 to two different machines. We assume, w.l.o.g., that they are assigned to machines 1 and 2 respectively, which now both reach a load of 3. Now job 3 departs (from machine 3). This brings us to a situation similar to the one we had in **case 1.2** (two machines with a load of 3 each), but here we have assignment restrictions that we did not have in the above case, so we have to make sure that OFF can still maintain a maximum load of 2. Similarly to case 1.2, job 10 now arrives, having a weight of 2, and it can only be assigned to machines 1 and 2. We can assume w.l.o.g. that ON assigns it to machine 1. Then job 9 departs from machine 2 and job 11 arrives, having a weight of 2. Job 11 can only be assigned to machine 1, and thus machine 1 reaches a load of 7. Now let us check what OFF does. OFF assigns jobs 1 and 2 to machine 4, and assigns job

3 to machine 2. It assigns job 8 to machine 3 and job 9 to machine 1. After the departure of job 3, machine 2 becomes empty for OFF, and job 10 is assigned to it. After the departure of job 9, machine 1 becomes empty for OFF and job 11 is assigned to it. OFF has a load of exactly 2 on each machine, and it maintained a maximum load of 2 throughout the sequence. Therefore, the ratio of maximum loads is 3.5 in this case as well.

This completes the proof of the lower bound. ■

We continue by proving the lower bound of 4 for the case of 5 machines.

Theorem 3.3 *Let ON be an on-line algorithm for restricted assignment of temporary tasks with unknown durations on 5 machines. Then the competitive ratio of ON is at least 4.*

Proof: Our input sequence consists only of unit jobs, and we prove that OFF maintains a maximum load of 1 while ON reaches a load of 4. Our sequence starts with five jobs (jobs $1, ..., 5$),which are admissible to all the machines. Obviously, if ON assigns four of them to the same machine, then OFF assigns each job to a different machine and the ratio is immediately 4. We divide the other possibilities that ON has into three cases.

Case 1: ON **assigns three of the jobs to the same machine.** We can assume that these three jobs are assigned to machine 1. Then OFF assigns one job to each machine, such that the job it assigns to machine 1 is not one of the 3 jobs that ON assigned to machine 1. Now the job that OFF assigned to machine 1 departs, and a new job which is only admissible to that machine arrives. Both algorithms must assign it to machine 1, so ON reaches a load of 4 on that machine, while OFF only has one job on each machine, and the ratio is 4.

Case 2: ON **assigns four of the jobs to two machines, two jobs each**. The fifth job is assigned to another machine. For instance, ON assigns jobs 1 and 2 to machine 1, jobs 3 and 4 to machine 2, and job 5 to machine 3. W.l.o.g., we can assume that this is its exact assignment in this case. First, job 5 departs. Now job 6 arrives, and it is only admissible to machines 1 and 2. Assume w.l.o.g. that ON assigns it to machine 2. OFF assigns jobs $1, ..., 4$ to machines $2, ..., 5$, respectively, and assigns job 5 to machine 1 (so that the machine which ON does not choose for job 6 becomes empty after job 5 leaves). Then OFF assigns job 6 to machine 1. Now job 1 departs, leaving machine 2 empty for OFF (and still with a load of 3 for ON). Job 7 now arrives, and is only admissible to machine 2. This makes the

maximum load of ON equal 4, while OFF maintains a maximum load of 1. The ratio is again 4.

Case 3: ON **assigns the jobs to at least four machines.** This is the only option left for ON. This way no machine has a load greater than 2, and there is at most one machine with a load of 2. Now one job departs, so that ON remains with 4 jobs, each one on a different machine. We can assume w.l.o.g. that these jobs are jobs $1, ..., 4$ and they are on machines $1, ..., 4$, respectively. Our strategy is simple. We will bring jobs that are admissible to two machines, and make sure that the machine that ON does not choose is empty for OFF. This way OFF will be able to assign the jobs to different machines than the one on which ON will build a "tower" of height 3. Next, a job which will only be admissible to that "tower" will arrive, and ON will have a maximum load of 4, while OFF maintains a maximum load of 1.

The next arrival is job 6, and it is only admissible to machines 1 and 2. Assume that ON assigns it to machine 1. Now job 2 departs, and job 7 arrives. Job 7 is only admissible to machines 3 and 4. Assume that ON chooses machine 4. This is followed by the departure of job 3, and the arrival of job 8, which is only admissible to machines 1 and 4 (which both have a load of 2). W.l.o.g., ON chooses machine 1. Now job 4 departs, and job 9 arrives. Job 9 is only admissible to machine 1, so it makes the load of ON equal 4 on that machine (jobs $1, 6, 8$, and 9 are all on machine 1 now).

Let us examine what OFF does in this case, according to our strategy mentioned above. At the beginning, OFF assigns job 5 to machine 2 (so that this machine will be empty when job 6 arrives), job 2 is assigned to machine 3 (so that machine 3 will be empty when job 7 arrives), job 3 is assigned to machine 4 (so that it will be empty when job 8 arrives), job 4 is assigned to machine 1 (so that machine 1 will be empty when job 9 arrives), and job 1 is assigned to machine 5 (and will remain there for all the rest of the sequence). Now, OFF assigns each of the jobs $6, 7$ and 8 to the machine that ON does not choose. Job 6 is assigned to machine 2, job 7 to machine 3, and job 8 to machine 4. Our assignment of the first 4 jobs assures us that these machines are empty when jobs $6, 7$ and 8 are assigned to them, and a maximum load of 1 is maintained. Now job 9 arrives and OFF assigns it to machine 1 (which is empty because job 4 has just left). So OFF still maintains a maximum load of 1. Therefore, the ratio is 4 in this case as well.

In this the proof is completed. ■

This completes our tight analysis of the case of $m \leq 5$, as we have shown the lower bounds which match the upper bound given in section 2.

4 Conclusions

We showed that GREEDY is optimal for $m \leq 5$, but it is known that GREEDY is not optimal for large enough m. A natural open question would be: What is the smallest value of m for which GREEDY is not optimal?

References

[1] A. Armon, Y. Azar, L. Epstein, and O. Regev. Temporary tasks assignment resolved. In *Proceedings of 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2002.

[2] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *Journal of the ACM*, 44(3):486–504, 1997. Also in *Proc. 25th ACM STOC*, 1993, pp. 623-631.

[3] Y. Azar, A. Z. Broder, and A. R. Karlin. On-line load balancing. *Theoretical Computer Science*, 130(1):73–84, 1994. Also in *Proc. 33rd IEEE FOCS*, 1992, pp. 218-225.

[4] Y. Azar and L. Epstein. On-line load balancing of temporary tasks on identical machines. In *5th Israeli Symp. on Theory of Computing and Systems*, pages 119–125, 1997.

[5] Y. Azar, B. Kalyanasundaram, S. Plotkin, Kirk R. Pruhs, and Orli Waarts. On-line load balancing of temporary tasks. *Journal of Algorithms*, 22(1):93–110, 1997. Also in *Proc. WADS'93*, pp. 119-130.

[6] Y. Azar, J. Naor, and R. Rom. The competitiveness of on-line assignments. *Journal of Algorithms*, 18(2):221–237, 1995. Also in *Proc. 3rd ACM-SIAM SODA*, 1992, pp. 203-210.

[7] R.L. Graham. Bounds for certain multiprocessor anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

[8] S. Plotkin and Y. Ma. An improved lower bound for load balancing of tasks with unknown duration. *Inform. Process. Lett.*, 62:301–303, 1997.