Priority Algorithms for Makespan Minimization in the Subset Model

Oded Regev [*]

February 14, 2002

Keywords: algorithmical approximation, scheduling

Abstract

We continue the recent study of priority algorithms initiated by Borodin et al. [3]. The definition of a priority algorithm nicely captures the idea of a "greedy-like" type algorithm. While priority algorithms are applicable to many optimization problems, in this paper we consider the problem of makespan minimization in scheduling in the subset model. We show that by using a fixed priority algorithm one cannot achieve a considerable improvement over the approximation ratio given by the online greedy algorithm. Namely, we present an $\Omega(\frac{\log m}{\log \log m})$ lower bound on the approximation ratio of any fixed priority algorithm where m is the number of machines.

1 Introduction

In a recent paper [3], Borodin et al. defined the notion of a "greedy-like" algorithm. The definition is based on the observation that greedy-like algorithms satisfy the property that the output is constructed incrementally with each input item being considered once. Following this observation, they define a Fixed Priority algorithm as an algorithm that determines a total ordering on all possible input items without actually seeing the input items. Then the algorithm proceeds with processing one element of the input at a time following this order. It is important to note that the decisions taken

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

by the algorithm are irrevocable. In the same paper, they also define a more general class of algorithms called ADAPTIVE PRIORITY algorithms. An algorithm in this class is allowed to change the total ordering after seeing each item of the input. In this paper we will only consider the class of FIXED PRIORITY algorithms.

The notion of priority algorithms can be applied to many optimization problems. In this paper we will be concerned with scheduling problems or, more specifically, the makespan minimization problem in the subset model (e.g. Chapter 12 in [2]). The subset model, also known as the restricted machines model, is the scheduling model where each job has in addition to its processing time, a subset of the machines to which it can be assigned. An assignment assigns an admissible machine to each job. The goal is to minimize the makespan, i.e., the maximum total processing time over the machines.

Lenstra et al. [5] and Shmoys and Tardos [6] presented offline algorithms with an approximation ratio of 2 for scheduling in the subset model. The problem was also considered in the online model by Azar et al. [1]. They showed an online algorithm that achieves an $O(\log m)$ approximation ratio where m is the number of machines[1]. Note that as an online algorithm, their algorithm is also a fixed priority algorithm. They also showed a tight lower bound of $\Omega(\log m)$ on the approximation ratio of any online algorithm. As shown in [3], the lower bound can be modified to provide an $\Omega(\log m)$ lower bound for several specific total orderings of fixed priority algorithms, such as the ordering that gives the highest priority to jobs having the least number of allowable machines.

An indication that approximation ratios lower than $O(\log m)$ might be possible can be found in a paper by Broder et al. [4]. Their results imply that for certain instances of the problem consisting of unit jobs with two admissible machines, assuming a *random* arrival order results in a tight bound of $\Theta(\frac{\log m}{\log \log m})$ on the approximation ratio of an algorithm that schedules each job on the least loaded machine. Note that this is not a priority algorithm because of the assumption that jobs arrive in a random order.

The question of whether there exists a lower bound that holds for any priority algorithm was left as an open question in [3]. In this paper we show

[1]In the language of online algorithms, the term competitive ratio is usually used instead of approximation ratio.

that a lower bound of $\Omega(\frac{\log m}{\log\log m})$ holds for any fixed priority algorithm. The methods we use are different from methods used to show lower bounds for online algorithms. We hope that the methods presented in this paper can be used to show additional results for priority algorithms.

2 Definitions

An instance of a scheduling problem in the subset model is a set of m machines and a set of n jobs. Each job has its own processing time and a subset of admissible machines. An assignment chooses an admissible machine for each job. The goal is to minimize the makespan, or the maximum total processing time of a machine. The approximation ratio of an algorithm is said to be c if the makespan achieved by the algorithm on any input is at most c times the optimal makespan.

An algorithm is said to be a Fixed Priority algorithm if it determines a total ordering of all possible jobs and then, given an input S, repeats the process of taking the first job in S according to the ordering and deciding where to schedule it without looking at future jobs.

We denote the set $\{1, 2, ..., k\}$ by $[k]$ and all logarithms are of base 2.

3 Lower Bound

Our lower bound consists of an adversarial sequence of jobs of processing time 1 with exactly two allowable machines. Therefore, given any ordering of all possible jobs we only have to consider its restriction to the total ordering of all $\binom{m}{2}$ such jobs. An alternative way to look at this restricted ordering is as an ordering of the edges of a complete graph of m vertices and this allows us to use the terminology of graph theory. Each vertex corresponds to a machine and each edge corresponds to a job. From now on, we use both definitions interchangeably. Also, since we are interested in asymptotic behavior, we choose the parameters in order to simplify the proof and assume that m is large enough. We begin with three simple technical lemmas.

Lemma 3.1 *Given a graph $G = (V, E)$ with an average degree d, the number of vertices whose degree is at least $\frac{d}{4}$ is at least $\sqrt{d|V|}/2$.*

Proof: Suppose otherwise and denote by V_1 the set $\{v \in V \mid d(v) \geq d/4\}$. The number of edges is $|E| = d|V|/2$. Remove all the edges incident to vertices in $V - V_1$. The number of edges removed is at most $d|V|/4$ and the number of remaining edges is therefore at least $d|V|/4$. Consider the graph G_1 induced by the vertices in V_1. Since all the edges left after the removal are between vertices in V_1, the number of edges in G_1 is at least $d|V|/4$. The number of vertices in G_1 is therefore at least $\sqrt{2d|V|/4} \geq \sqrt{d|V|}/2$. ■

Lemma 3.2 *Given a set A of n elements and k subsets each of size at least r, there are at least $\frac{r}{2}$ elements that are contained in at least $\frac{rk}{2n}$ of the subsets.*

Proof: Let a_i denote the number of subsets in which element i is contained. Notice that $\sum_i a_i \geq kr$ and for every i, $a_i \leq k$. Therefore, if there were less than $\frac{r}{2}$ elements for which $a_i \geq \frac{rk}{2n}$ then $\sum_i a_i < \frac{r}{2}k + n\frac{rk}{2n} = rk$. ■

Lemma 3.3 *Given a collection of k sets, $A_i, 1 \leq i \leq k$ and another set B of q elements such that every element of B is contained in at least s of the subsets A_i, there exists a set $C \subseteq B$ and a set of indices $I \subseteq [k]$ with $|I| = s$ such that every $c \in C$ is contained in A_i for every $i \in I$ and $|C| \geq q/\binom{k}{s}$.*

Proof: Let $I_a = \{i \in [k] \mid a \in A_i\}$ for every $a \in B$ be the set of indices indicating the containment of a in the A_i's. Note that $|I_a| \geq s$ so for every $a \in B$ let I'_a be any subset of I_a such that $|I'_a| = s$. Essentially, each element $a \in B$ is assigned to a subset of $[k]$ of size s. Since there are only $\binom{k}{s}$ such subsets, there must be a set $C \subseteq B$ with $|C| \geq q/\binom{k}{s}$ such that I'_a is the same for every $a \in C$. ■

The following lemma proves the existence of certain structures in any total ordering of the edges of the complete graphs. These structures will be used in building the lower bound. Roughly speaking, the lemma proves the existence of a big enough set V' of vertices and a sequence of sets of edges such that each vertex has many incident edges in each of the sets of edges.

Lemma 3.4 *Let $G = (V, E)$ be the complete graph over m vertices with a total ordering of its edges. Also, let $s = \frac{\log m}{8 \log\log m}$. Then there exists a subset $V' \subseteq V$ and a sequence of disjoint sets of edges $E_1, E_2, ..., E_s$ such that:*

- $|V'| = \sqrt{m}$
- $\forall 1 \leq i < j \leq s, e_1 \in E_i, e_2 \in E_j$, *$e_1$ is before e_2 in the ordering*
- $\forall 1 \leq i \leq s$ *the edges of E_i are between V' and $V - V'$.*
- $\forall 1 \leq i \leq s, v \in V'$ *there are at least $\frac{m}{16 \log^2 m}$ edges in E_i incident to v.*
- $\forall 1 \leq i \leq s \; \forall \; V'' \subseteq V' \; \forall \; U \subseteq V - V'$, *such that $|V''| \geq m^{\frac{1}{4}}$ and $|V - V' - U| \leq \frac{m}{32 \log^2 m}$, there exists a vertex $v \in U$ that is connected by edges of E_i to at least* $\log m$ *vertices in V''.*

Proof: Assume without loss of generality that the total ordering is $e_1, e_2, ..., e_{\binom{m}{2}}$. We partition the set of $\binom{m}{2}$ edges into $\log^2 m$ equal parts. The set $\hat{E}_i$ for $i \in [\log^2 m]$ consists of the edges $e_{(i-1)\binom{m}{2}/\log^2 m+1}, ..., e_{i\binom{m}{2}/\log^2 m}$. Similarly, we define a partition of the complete graph G into $G_1, G_2, ..., G_{\log^2 m}$ $(G_i = (V, \hat{E}_i))$. Since the average degree in each G_i is $\frac{m-1}{\log^2 m} \geq \frac{m}{2 \log^2 m}$, Lemma 3.1 implies that in each G_i we can find at least $\frac{m}{4 \log m}$ vertices whose degree in G_i is at least $\frac{m}{8 \log^2 m}$. Denote these vertices by V_i.

Each of the $\log^2 m$ sets V_i contains at least $\frac{m}{4 \log m}$ vertices. According to Lemma 3.2, there exists a set of vertices $\hat{V}$ of size $\frac{m}{8 \log m}$ all of which are contained in $\frac{\log m}{8}$ of the V_i's.

We apply Lemma 3.3 with the collection of V_i's, the set $\hat{V}$ and the parameters $k = \log^2 m$, $q = \frac{m}{8 \log m}$ and $s = \frac{\log m}{8 \log \log m}$. Note that we can choose this value of s because $\frac{\log m}{8 \log \log m} \leq \frac{\log m}{8}$. Hence, there exists a set $V' \subseteq \hat{V}$ and a set of s indices I such that every vertex of V' is contained in V_i for all $i \in I$. The size of V' is at least $q/\binom{k}{s} \geq q \cdot k^{-s} = \frac{m}{8 \log m} \cdot 2^{-\frac{\log m}{4}} \geq \sqrt{m}$. Remove vertices from V' so that $|V'| = \sqrt{m}$.

Assume that $I = \{i_1, i_2, ..., i_s\}$ with $i_1 < i_2 < ... < i_s$ and define $E_j = \hat{E}_{i_j} \cap \{\{u, v\} \mid u \in V' \wedge v \in V - V'\}$, that is, the edges of $\hat{E}_{i_j}$ going between V' and $V - V'$. We claim that the set V' and the sequence $E_1, E_2, ..., E_s$ satisfy the required properties. The first three properties follow from the definition of the sequence and the size of V' mentioned above. The fourth property holds because for every j, $1 \leq j \leq s$ and $v \in V'$, v is in V_{i_j}. Therefore, the degree of v in $\hat{E}_{i_j}$ is at least $\frac{m}{8 \log^2 m}$. After removing at most $|V'|$ edges in $\hat{E}_{i_j}$ from v to vertices in V', the number of edges incident to v in E_j is at least $\frac{m}{8 \log^2 m} - \sqrt{m} \geq \frac{m}{16 \log^2 m}$ as required. The last property is a corollary of the fourth property. This is because each vertex of V'' is connected by edges of E_i to at least $\frac{m}{16\log^2 m}$ vertices in $V - V'$ and therefore to at least $\frac{m}{32\log^2 m}$ vertices of U. By Lemma 3.2, there exist at least $\frac{m}{64\log^2 m} \geq 1$ elements in U that are connected to at least $m^{\frac{1}{4}}\frac{m}{32\log^2 m}/(2m) \geq \log m$ elements in V''. ■

Theorem 3.5 *For the makespan problem on the subset model, any fixed priority algorithm has an approximation ratio $\Omega(\frac{\log m}{\log\log m})$. Moreover, this also holds when all the jobs have unit processing time and the number of allowable machines per job is exactly two.*

Proof: Given any fixed priority algorithm, consider the restriction of its total ordering to the edges of the complete graph over m vertices. Lemma 3.4 shows the existence of a subset of vertices V' and sets of edges $E_1, E_2, ..., E_s$ with certain properties. The adversary proceeds in s stages where in stage i jobs from E_i are released. The jobs are released according to the total ordering. After each stage, the makespan of the priority algorithm increases by one while the makespan of an optimal assignment remains one. Notice that this proves the theorem since $s = \Theta(\frac{\log m}{\log\log m})$. We assume by contradiction that the approximation ratio of the priority algorithm is less than s.

Let $V_1 = V'$ and $U_1 = V - V'$. At the end of stage i we define the sets $V_{i+1} \subseteq V_i$ and $U_{i+1} \subseteq U_i$ based on the decisions of the priority algorithm. The size of the sets V_i decreases by a factor of $O(\log m)$ in every stage and the sizes of the sets U_i decrease only slightly. Stage i is composed of edges of E_i between the set V_i and U_i. According to the last property in Lemma 3.4, there exists a vertex $v \in U_i$ that is connected by edges of E_i to $\log m$ vertices in V_i. Provide as input to the priority algorithm these $\log m$ edges and remove the $\log m$ vertices from V_i. Continue with the remaining vertices in V_i as long as there are at least $m^{\frac{1}{4}}$ vertices in V_i. Note that the set of $O(|V_i|)$ edges of stage i described above should be given to the priority algorithm according to the total ordering.

Consider a set of $\log m$ edges incident to a vertex $v \in U_i$ that are given to the priority algorithm in stage i. The priority algorithm cannot assign all the $\log m$ edges to vertex v or its load will be too high. Therefore, at least one edge must be assigned to a vertex in V_i. From every set of $\log m$ vertices in V_i, the set V_{i+1} contains one vertex to which the priority algorithm assigned an edge in stage i. The set U_{i+1} contains the vertices of U_i except the vertices that are incident to an edge that was given as input to the priority algorithm in stage i. Note that $|U_i - U_{i+1}| \leq |V_i| \leq \sqrt{m}$ and that as long as $|V_i| \geq 2m^{\frac{1}{4}}$ the size of V_{i+1} is at least $|V_i|/(2\log m)$. By a straight forward calculation, it can be seen that the conditions of the last property in Lemma 3.4 hold after s stages. Note that the makespan of the priority algorithm increases by one after each stage. Also, the optimal assignment that assigns $\log m - 1$ edges of every set of $\log m$ edges to vertices in $V_i - V_{i+1}$ and the remaining edge to the common neighbor in U_i is legal and its makespan is one. ■

4 Conclusion

The above lower bound shows that a fixed priority algorithm cannot do much better than an online algorithm in the subset model. A natural open question is to consider the power of adaptive priority algorithms or that of randomization.

Acknowledgements

I thank Yossi Azar and Allan Borodin for helpful discussions and comments.

References

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

[2] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.

[3] A. Borodin, M. N. Nielsen, and C. Rackoff. (Incremental) Priority Algorithms. In *Proc. 13th ACM-SIAM Symp. on Discrete Algorithms*, pages 752–761, 2002.

[4] A. Z. Broder, A. Frieze, C. Lund, S. Phillips, and N. Reingold. Balanced allocations for tree-like inputs. *Information Processing Letters*, 55(6):329–332, 1995.

[5] J.K. Lenstra, D.B. Shmoys, and E. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Prog.*, 46:259–271, 1990.

[6] D. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming A*, 62:461–474, 1993. Also in the proceeding of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms, 1993.