An Experimental Study of Online Scheduling Algorithms *

Susanne Albers† Bianca Schröder‡

Abstract

We present the first comprehensive experimental study of online algorithms for Graham's scheduling problem. Graham's scheduling problem is a fundamental problem in scheduling theory where a sequence of jobs has to be scheduled on m identical parallel machines so as to minimize the makespan. Graham gave an elegant algorithm that is $(2 - 1/m)$-competitive. Recently a number of new online algorithms were developed that achieve competitive ratios around 1.9. Since competitive analysis can only capture the worst case behavior of an algorithm a question often asked is: Are these new algorithms geared only towards a pathological case or do they perform better in practice, too?

We address this question by analyzing the algorithms on various job sequences. In our actual tests, we analyzed the algorithms (1) on real world jobs and (2) on jobs generated by probability distributions. It turns out that the performance of the algorithms depends heavily on the characteristics of the respective work load. On job sequences that are generated by standard probability distributions, Graham's strategy is clearly the best. However, on the real world jobs the new algorithms often outperform Graham's strategy. Our experimental study confirms theoretical results in the sense that there are also job sequences in practice on which the new online algorithms perform better. Our study can help to inform practitioners about the new scheduling strategies as an alternative to Graham's algorithm.

1 Introduction

During the last ten years *online scheduling* has received a lot of research interest, see for instance [1, 2, 18, 22, 23]. In online scheduling, a *sequence of jobs* $\sigma = J_1, J_2, \ldots, J_n$ has to be scheduled on a number of machines. The jobs arrive one by one; whenever a new job arrives, it has to be dispatched immediately to one of the machines, without knowledge of any future jobs. The goal is to optimize a given objective function. Many online algorithms for various scheduling problems have been proposed and evaluated using competitive analysis. However, an experimental evaluation of the algorithms was usually not presented. We remark here that there exist experimental studies for many scheduling strategies used in parallel supercomputers [11, 12]. However, these are strategies for scheduling jobs

*Important note: The figures presented in this paper make heavily use of color to distinguish between the performance of different algorithms. We recommend to print the paper on a color printer, if possible, or to preview it on a color screen.

†Institut für Informatik, Albert-Ludwigs-Universität, Georges-Köhler-Allee 79, 79110 Freiburg, Germany. salbers@informatik.uni-freiburg.de Part of this work was done while at the Max-Planck-Institut für Informatik, Saarbrücken, Germany.

‡Computer Science Department, 5000 Forbes Avenue, Carnegie Mellon University, Pittsburgh, PA 15213, U.S.A. bianca@cs.cmu.edu

that can span *more than one machine*, while in Graham's model each job has to be assigned to exactly *one* machine. Savelsbergh *et al.* [19] recently gave an experimental analysis of *offline* approximation algorithms for the problem of minimizing the weighted sum of completion times. Moreover, there are average-case analyses and related experimental studies of bin packing problems, see e.g. [4, 6].

In this paper we present an experimental study of algorithms for a fundamental problem in online scheduling. This problem is referred to as Graham's problem and has been investigated extensively from a theoretical point of view, see for instance [1, 2, 3, 5, 7, 8, 9, 10, 13, 14, 18, 21]. In Graham's problem, a sequence of jobs $\sigma = J_1, J_2, \ldots, J_n$ has to be scheduled on m identical parallel machines. Whenever a new job J_t, $1 \leq t \leq n$, arrives, its processing time p_t is known in advance. Each job has to be assigned immediately on one of the machines, without knowledge of any future jobs. The goal is to minimize the *makespan*, which is the completion time of the job that finishes last. This problem arises for instance in high performance and supercomputing environments. Here, it is often the case that either preemption is not supported by the system or the high memory requirements of the jobs make preemption prohibitively expensive. The runtimes of the jobs are known at least approximately since users are usually required to give an estimate for the CPU requirements of their jobs. The objective of minimizing the makespan in this setting aims at achieving a high utilization on the machines. In addition to its practical relevance, Graham's problem is important because it is the root of many problem variants where, for instance, preemption is allowed, precedence constraints exist among jobs, or machines run at different speeds.

In 1966 Graham gave an algorithm that is $(2 - 1/m)$-competitive. Following [20] we call an online scheduling algorithm *c-competitive* if, for all job sequences $\sigma = J_1, J_2, \ldots, J_n$, $A(\sigma) \leq c \cdot OPT(\sigma)$, where $A(\sigma)$ is the makespan of the schedule produced by A and $OPT(\sigma)$ is the makespan of an optimal schedule for σ. It was open for a long time whether an online algorithm can achieve a competitive ratio that is asymptotically smaller than 2, for all values of m. In the early nineties Bartal *et al.* [2] presented an algorithm that is 1.986-competitive. Karger *et al.* [18] generalized the algorithm and gave an upper bound of 1.945. Recently, Albers [1] presented an improved algorithm that is 1.923-competitive. An interesting question is whether these new techniques are geared only towards a pathological worst case or whether they also lead to better results in practice. In this paper we address this question and present the first comprehensive experimental study of online algorithms for Graham's scheduling problem.

Description of the experiments: We implemented the online algorithms by Graham, Bartal *et al.*, Karger *et al.* and Albers and tested them on (1) real world job sequences as well as on (2) job sequences generated by probability distributions. As for the real world jobs, we investigated data sets from three different machine configurations. The first data set consists of job sequences taken from the log files of three *MPP's* (Massively Parallel Processors) at three different supercomputing centers. The runtimes in the second data set were extracted from a log file of a 16 processor *vector machine* at the Pittsburgh Supercomputing Center. This environment resembles very much the model described above. The jobs in the third data set were obtained from a process accounting on a Sun Ultra *workstation*. This workstation is one of the main computing servers at the Max Planck Institute in Saarbrücken. We believe that an analysis of the algorithms' performance on real job traces gives the most meaningful results. However, we also evaluated the algorithms under job sequences generated by probability distributions. More specifically, we generated job sequences according to the uniform, exponential, Erlang, hyperexponential and Bounded Pareto distributions.

For each job sequence and each of the four algorithms, we determined the ratio *online makespan/optimum makespan* after each scheduling step, i.e. whenever a new job was scheduled, the ratio was re-computed. This allows us not only to compare the algorithms against each other but also gives us a measure for how far away the online algorithms are from the optimal offline solution at any given point of time. Since the problem of computing an optimal solution for Graham's problem is NP-hard, at any time the optimum makespan was approximated by a lower bound. Suppose that we have scheduled $J_1, \ldots, J_t$. Clearly, a lower bound on the optimum makespan for the subsequence $J_1, \ldots, J_t$ is given by the maximum of the following three expressions: (1) $\max_{1 \le s \le t} p_s$; (2) $\frac{1}{m} \sum_{s=1}^{t} p_s$; (3) twice the size of the $(m+1)$-st largest job in $J_1, \ldots, J_t$. It turns out that this lower bound is sufficient for a proper evaluation of the algorithms, i.e. we considered generalization of (3) but did not obtain any improved lower bounds. Finally, we also considered the algorithms' performance for different machine numbers and evaluated settings with 10, 50, 100 and 500 machines.

Summary of the experimental results: The results differ substantially depending on the workload characteristics. In the experiments with real world jobs, the ratios *online makespan/optimum makespan* fluctuate. We observe sudden increases and decreases, depending on the size of the last job that was scheduled. Whenever the processing time of a new job is in the order of the average load on the machines, the ratio goes up, with values up to 1.8–1.9. Whenever the processing time of a new job is very large compared to the average load on the machines, the ratio drops and approaches 1. Only after a large number of jobs has been scheduled do the ratios stabilize. An important result of the experiments is that some of the new algorithms suffer much less from these sudden increases than Graham's algorithm and therefore lead to a more predictable performance. They also often outperform Graham's algorithm. This makes the new algorithms also interesting from a practical point of view.

In the experiments with job sequences generated by one of the standard probability distributions, the ratios online makespan/optimum makespan converge quickly. Graham's algorithm outperforms the other three algorithms and achieves ratios close to 1. The ratios of the algorithm by Bartal *et al.* and Albers are slightly higher and converge to values between 1.2 and 1.3. The algorithm by Karger *et al.* performs worse, with ratios between 1.7 and 1.9. Surprisingly, these results hold for all standard probability distributions.

Our experimental study confirms and validates theoretical results in the sense that there are job sequences in practice where the new online algorithms outperform Graham's strategy. Our study can help that practitioners become aware of the new strategies and possibly use them as an alternative to Graham's algorithm.

Organization of the paper: In Section 2 we describe the online scheduling algorithms by Graham, Bartal *et al.*, Karger *et al.* and Albers. In Section 3 we give a detailed presentation of the experiments with real world jobs. A description of the tests with randomly generated jobs follows in Section 4. While the results in Section 3 and Section 4 are limited to the 10 machine case we discuss in Section 5 the results for experiments with larger machine numbers.

2 The Algorithms

In this section we describe the online algorithms that we will analyze experimentally. An algorithm is presented with a job sequence $\sigma = J_1, J_2, \ldots, J_n$. Let p_t denote the processing time of J_t, $1 \le t \le n$. At any time let the *load* of a machine be the sum of the processing times of the jobs already assigned to it. In the following, when describing the algorithms, we assume that an algorithm has already scheduled the first $t-1$ jobs $J_1, \ldots, J_{t-1}$. We specify how the next job J_t is scheduled.

Algorithm by Graham: Schedule J_t on the machine with the smallest load.

All the other algorithms maintain a list of the machines sorted in non-decreasing order by current load. The goal is to always maintain some lightly loaded and some heavily loaded machines. Let M_i^{t-1} denote the machine with the i-th smallest load, $1 \le i \le m$, after exactly $t-1$ jobs have been scheduled. In particular, M_1^{t-1} is the machine with the smallest load and M_m^{t-1} is the machine with the largest load. We denote by l_i^{t-1} the load of machine M_i^{t-1}, $1 \le i \le m$. Note that the load l_m^{t-1} of the most loaded machine is always equal to the current makespan. Let A_i^{t-1} be the average load on the i smallest machines after $t-1$ jobs have been scheduled. The algorithm by Bartal *et al.* keeps about 44.5% of the machines lightly loaded.

Algorithm by Bartal *et al.*: Let $k = [0.445m]$ be the integer closest to $0.445m$ and $\epsilon = 1/70$. Schedule J_t on M_{k+1}^{t-1} if $l_{k+1}^{t-1} + p_t \le (2-\epsilon)A_k^{t-1}$ Otherwise schedule J_t on the machine with the smallest load.

The algorithm by Karger *et al.* maintains a full stair-pattern.

Algorithm by Karger *et al.*: Set $\alpha = 1.945$. Schedule J_t on the machine M_k^{t-1} with the largest load such that $l_k^{t-1} + p_t \le \alpha A_{k-1}^{t-1}$. If there is no such machine, then schedule J_t on the machine with the smallest load.

The algorithm by Albers keeps 50% of the machines lightly loaded.

Algorithm by Albers: Set $c = 1.923$, $k = \lfloor \frac{m}{2} \rfloor$ and $j = 0.29m$. Set $\alpha = \frac{(c-1)k - j/2}{(c-1)(m-k)}$. Let L_l be the sum of the loads on machines $M_1^t, \ldots, M_k^t$ if J_t is scheduled on the least loaded machine. Similarly, let L_h be the sum of the loads on machines $M_{k+1}^t, \ldots, M_m^t$ if J_t is scheduled on the least loaded machine. Let λ_m^t be the makespan, i.e. the load of the most loaded machine, if J_t is scheduled on the machine with the $(k+1)$-st smallest load. Recall that l_m^{t-1} is the makespan before the assignment of J_t. Schedule J_t on the least loaded machine if one of the following conditions holds: (a) $L_l \le \alpha L_h$; (b) $\lambda_m^t > l_m^{t-1}$ and $\lambda_m^t > c \cdot \frac{L_l + L_h}{m}$. Otherwise schedule J_t on the machine with the $(k+1)$-st smallest load.

3 Experiments with Real World Jobs

Before we discuss the results of the experiments we describe the experimental setup. The jobs used in the experiments come from three different types of systems. The first data set consists of job traces taken from *MPP's* (massively parallel processors) and were obtained from Feitelson's Parallel Workloads Archive. It includes a trace from a 512-node IBM-SP2 at Cornell Theory Center (CTC), a trace from a 100-node IBM-SP2 at the KTH in Sweden and a trace from a 128-node iPSC/860 at NASA Ames. The second data set consists of runtimes measured at the Pittsburgh Supercomputing Center's Cray C90, which is a *vector machine*. The jobs in the third data set were obtained from a process accounting on a Sun Ultra *workstation* with two 200 MHz processors and 1024 MB main memory. This workstation is one of the main computing servers at the Max Planck Institute in

Saarbrücken. The following table summarizes the main characteristics of the workloads. These will be crucial for the interpretation of the results.

System	Year	Number of Jobs	Mean Size (sec)	Min (sec)	Max (sec)	Squared Coefficient of Variation
CTC IBM-SP2	1996 - 1997	57290	2903.6	1	43138	2.72
KTH IBM-SP2	1996 - 1997	28490	8878.9	1	226709	5.48
NASA Ames iPSC/860	1993	42050	348.20	1	62643	27.21
PSC Cray C90	1997	54962	4562.6	1	2222749	43.16
MPI Sun Ultra	1998	300000	2.3	0.01	47565.4	7550.58

We split each job trace into job sequences containing 10000 jobs. We then ran the online algorithms on each job sequence and recorded the ratio *online makespan/optimum makespan* after each job. The machine numbers used in these experiments range from 10 to 500. The next two sections describe and analyze the experimental results. In Sections 3.1 and 3.2 we first present the results for 10 machines. The results for larger machine numbers are summarized in Section 5.

3.1 The Experimental Results

We begin with the results for the MPP data. The development of the ratios under the job sequences obtained from the CTC and the KTH traces was virtually identical. Figure 1 shows the typical development of the ratios of the online algorithms' makespans to the optimal makespans for these job sequences. We see that the ratios during the first 1000 jobs oscillate between values of 1.1 and 1.7.

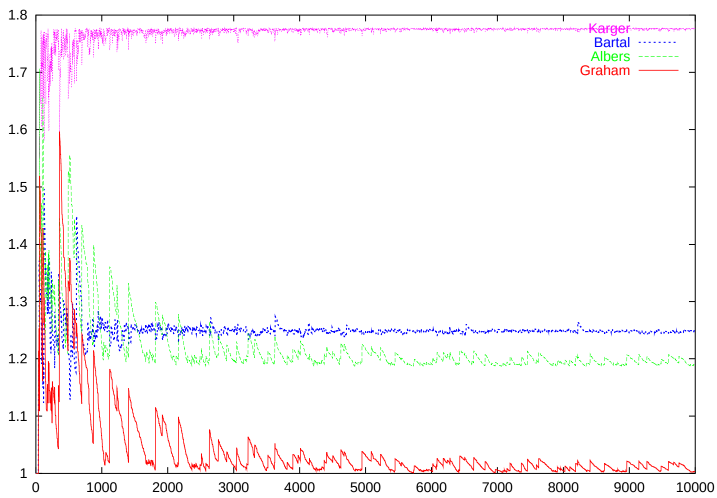

Figure 1: Performance of the online algorithms on the KTH data

The only exception are the ratios for Karger's algorithms which immediately approach a value of 1.8. After the first 1000 jobs the ratios of all algorithms stabilize. For Bartal's and Albers' algorithm they converge towards a value around 1.2 while the ratio for Graham's algorithm approaches 1. Figure 2 shows the results for the NASA jobs. The general trend in the development of the ratios is similar to that observed for the CTC and the KTH data. Initially, the ratios fluctuate until they finally converge to the same values as for the CTC/KTH data. In contrast to the results for the CTC and KTH jobs it

Figure 2: Performance of the online algorithm on the NASA data

takes much longer until the ratios stabilize. Under the PSC data the ratios are even more volatile (see Figure 3). Especially, the ratio for Graham's algorithm is extremely instable and goes frequently up to

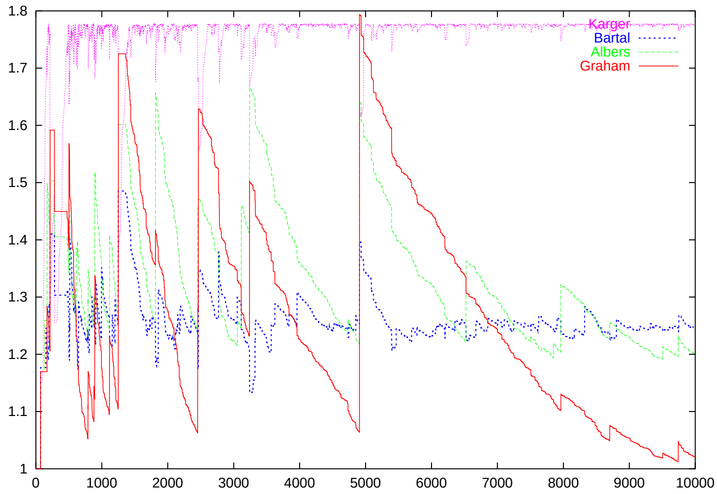

Figure 3: Performance of the online algorithms on the PSC data

values between 1.7 and 1.8. Bartal's algorithm, on the other hand, converges very early to a ratio close to 1.2. After around 9000 jobs have been scheduled the ratios approach the values that we observed for the previous traces. The workstation data set is the only one where the results were different for the various job sequences. They also differ from the results we have observed so far. Figure 4 shows two typical scenarios for job sequences extracted from the workstation trace. We see that the ratios again oscillate in the beginning, but this time they don't converge gradually to some value. Instead they drop very abruptly to 1 and don't change after that. This sudden drop in the ratios can occur very early as shown in Figure 4 (top) or later in the scheduling process as in Figure 4 (bottom).

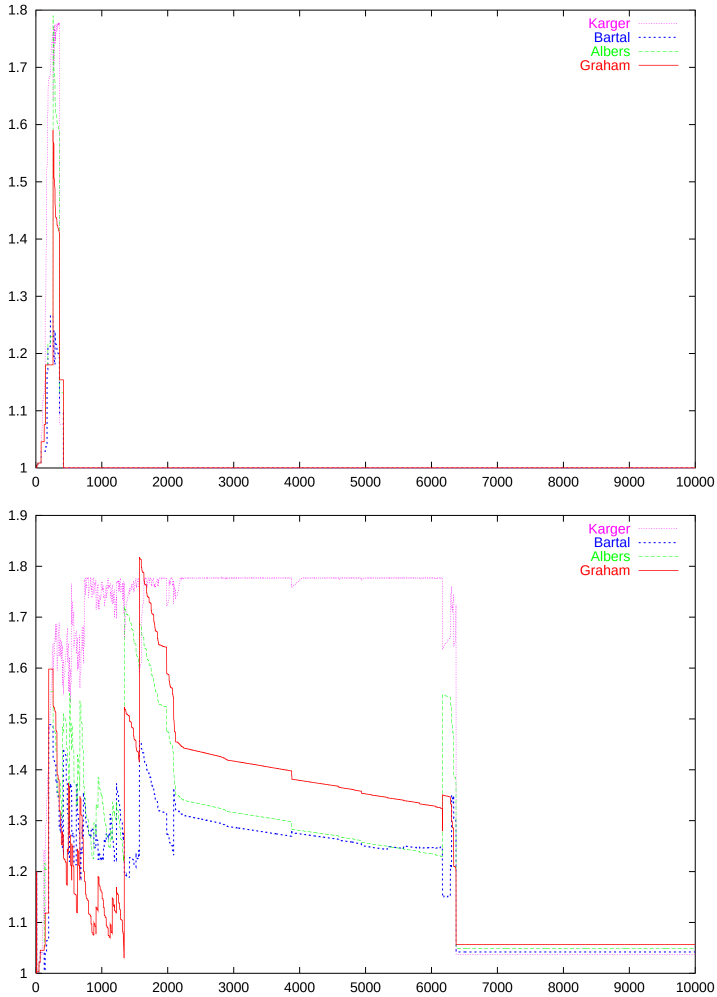

Figure 4: Typical results for experiments with workstation jobs

3.2 Analysis of the Results

To interpret the experimental results it is helpful to understand in which way a new job can affect the ratio *online makespan/optimal makespan*. Depending on its size compared to the average load on the machines, a job can have one of the following three effects.

1. If the size of an arriving job is small compared to the average load on the machines, the job will neither significantly affect the optimum makespan nor the online makespans. Therefore, the ratio *online makespan/optimal makespan* will remain almost unchanged.

2. If the size of an arriving job is in the order of the average load on the machines the ratio *online makespan/optimal makespan* will increase. The reason is that all algorithms have to maintain a certain balance between the load on the machines to prevent the makespan from growing too large. Therefore, they will have to assign the arriving job to a machine that contains already an amount of load close to the average load. The optimal offline strategy would have been to reserve one machine almost entirely for this job. Therefore, if the size of a new job is approximately that

of the average load on the machines, the ratio *online makespan/optimal makespan* will increase and in the worst case approach 2.

3. If the size of the new job is extremely large compared to the average load on the machines, the new job will completely dominate the optimal makespan, as well as the makespan of an online algorithm. This leads to almost the same makespan for the optimal and the online algorithm's solutions. As a result, the ratio *online makespan/optimal makespan* will approach 1.

In the following we will refer to these three effects as effect 1, 2, and 3, respectively. Note at this point that a sequence of small jobs (effect 1) followed by a job triggering effect 2 is the worst case scenario for Graham's algorithm. This is because Graham will distribute the small jobs completely evenly over the machines and therefore has to assign the "effect 2 job" to a machine that contains already a lot of load. All the other algorithms try to alleviate this problem by keeping some of the machines lightly loaded and hence reserving some space for "effect 2 jobs" that might arrive in the future.

How likely the occurrence of each of the three effects is and how pronounced the effect will be, depends on the characteristics of the workload and the scheduling algorithm. If the variability in the job sizes is low effect 2 and 3 are very unlikely to occur. The reason is that a low variability in the job size distribution means that the jobs are relatively similar in size. Therefore, the probability that a new job has a size similar to that of all the jobs at one machine combined is very low. Looking at the table with the characteristics of the traces we see that the CTC and the KTH traces have a very low squared coefficient of variation, which indicates a low variability in the job sizes. This explains why the ratios converged so quickly in the experiments with these traces: the low variability in the job sizes makes the arrival of an "effect 2" or "effect 3" job very unlikely. It also explains why the performance of the three new algorithms is worse than that of Graham's algorithm (except for the first jobs). The new algorithms reserve some space for large jobs that never arrive and therefore have higher makespans. For the NASA and the PSC trace the squared coefficient of variation is much higher than for the CTC and the KTH traces indicating a higher variability in the job sizes. Therefore, effect 3 and in particular effect 2 are likely to happen, even after many jobs have been scheduled. This leads to the fluctuation of the *online makespan/optimal makespan* that we observed in Figure 2 and 3. We also see that in this case the strategy of keeping some machines lightly loaded can pay off. The ratios for Bartal's algorithm, for instance, are in many cases much lower than the ratios of Graham's algorithm. Moreover, the ratio under Bartal's algorithm converges quickly leading to a more predictable performance than the heavily oscillating ratio of Graham's algorithm. In the workstation trace the variability is extremely high meaning that some jobs have a size that is extremely large compared to that of an average job. Typically, in workstation traces the largest 1 percent of all jobs make up half of the total load (a property sometimes referred to as *heavy-tailed property*). As soon as one of these extremely large jobs arrives, it completely dominates both the optimal and the online makespan. This leads to the drop of the ratios to 1 that we see in Figure 4.

To sum it up, the development of the ratios for our real world data depends almost exclusively on the occurrences of the large and particularly the extremely large jobs. The most important quantity is the proportion of the large jobs to the average load on the machines. Please note at this point, that the high variability that we observed in our traces is not a weirdness in these particular traces. The property that the largest jobs are extremely large compared to the average size has been observed in many systems and (as mentioned above) distributions like this are often called heavy-tailed. See for example [16] for more on heavy-tailed workloads.

4 Experiments with Jobs Generated by Probability Distributions

We also analyzed the performance of the scheduling algorithms on job sequences generated by the following probability distributions: (a) the uniform distribution; (b) the exponential distribution; (c) the Erlang distribution; (d) the hyperexponential distribution; and (e) the Bounded Pareto distribution. For a definition of these distributions, see e.g. [17, 24]. When choosing the parameters of the distributions from which the numbers where generated we tried on the one hand to cover a great range and on the other hand to use parameters similar to that in tests presented in [11] and [12]. The distributions commonly used to model service times of computer systems are the exponential, hyperexponential and the Erlang distribution [17]. For the sake of completeness we also included the uniform distribution. The experimental results for these four standard distributions are discussed in Section 4.1. The Bounded Pareto distribution is discussed in Section4.2.

4.1 The Standard Distributions

Surprisingly, the results did not differ significantly for the various standard distributions. Even more surprisingly, the results were similar for all parameters. Figure 5 shows the development of the ratio *online makespan/optimum makespan* on 10 machines for exponentially distributed job sizes, but also represents the results for the other distributions quite well. We observe that the curves fluctuate to a much smaller degree than under the real work loads. They converge to the same values as in the case of real job sequences, but they do so much faster. The reason is that the variability in the job sizes is much lower for these distributions. The exponential distribution has a mean squared coefficient of variation of 1 independently of its mean. The Erlang distribution and the uniform distribution always have a squared coefficient of variation less than or equal to 1, independently of how their parameters are chosen. For the hyperexponential distribution it is theoretically possible to choose the parameters as to match the mean and the squared coefficient of variation of any distribution. However, to achieve squared coefficients of variations as observed for the more variable real world traces one would have to set the parameters to very extreme values.

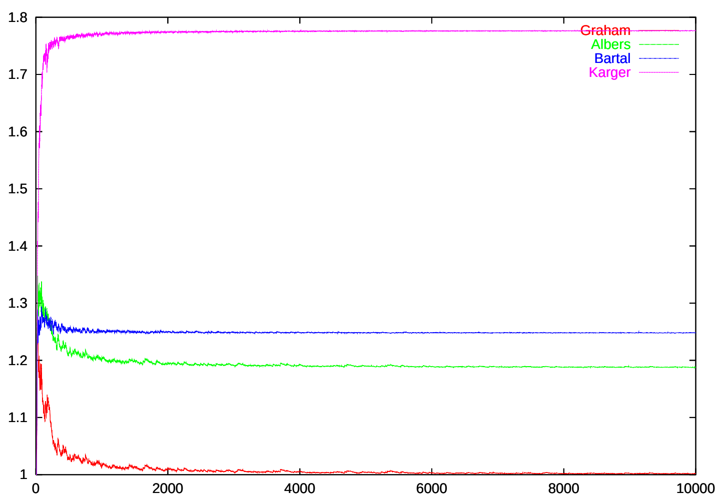

Figure 5: The performance of the algorithms with 10 machines under an exponential workload.

4.2 The Bounded Pareto Distribution

In contrast to the standard distributions, the Bounded Pareto distribution can be easily fit to observed data. We chose the parameters for this distribution so as to match the mean job sizes in the various job traces and to create different degrees of variability in the job sizes. It turned out that for a very low variability the results were virtually identical to those for the CTC and the KTH data as shown in Figure 1. For medium variability the results looked very similar to those for the PSC data in Figure 3. For extremely variable job sizes the results matched those for the workstation traces (see Figure 4). This confirms our theory from Section 3 that the variability in the job sizes is the crucial factor for the performance of the algorithms.

5 Results for Larger Machine Numbers

All results shown so far are for simulations with 10 machines. We repeated all experiments for all job sequences with 50, 100 and 500 machines, to study the effect of larger machine numbers.

It turns out that the performance of the algorithms for larger machine numbers and jobs generated from *standard distributions* can be predicted pretty well from their performance on 10 machines. The development of the ratios for large machine numbers is similar to that in the 10 machine case in that the ratios finally converge to similar values. Figure 6 shows the ratios for the experimental results with 500 machines on the same job sequence used for the 10 machine experiment plotted in Figure 5. Graham's performance ratio gets close to 1 while Karger's ratio of approximately 1.9 is by far the

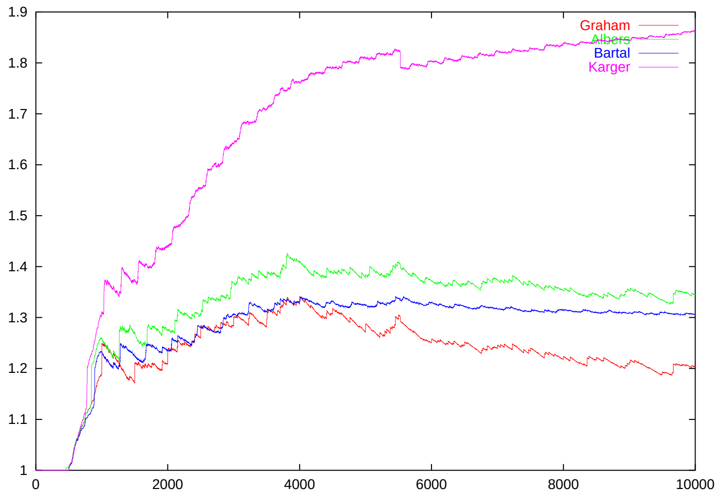

Figure 6: The performance of the algorithms under an exponential workload and 500 machines

worst. The curves of the algorithms by Albers and Bartal *et al.* are quite similar lying between 1.2 and 1.3 with an advantage for Albers' algorithm that grows as the number of jobs gets larger. However, it is noticeable that for larger machine numbers the convergence of the ratios is much slower, since it takes more jobs to "fill" all the machines and reach schedules whose makespan is stable with respect to competitiveness.

If we, however, look at job sequences with length proportional to the number of machines the curves look very similar even for different machine numbers. Figure 7 shows the development of the ratios for the 10 machine experiment for only the first 200 jobs. These curves resemble very much those for 500 machines and 10000 jobs shown in Figure 6. For the Bounded Pareto distribution, depending on the chosen parameters, we observe the same results as in the case of the standard distributions or the real world data presented below.

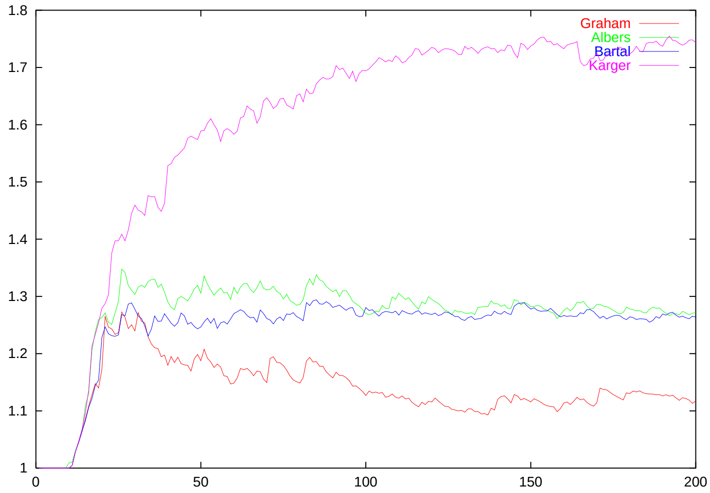

Figure 7: The performance of the algorithms under an exponential workload and 10 machines for the first 200 jobs

For the job sequences taken from the *real world data* the results don't scale in the above sense. Recall that these job sequences exhibit a higher variability in the job sizes and that the results mainly depend on how strong the effects of medium and very big jobs are. The important observation in the experiments with these sequences and larger machine numbers is that the more machines we use the smaller is the influence of effect 2. Since the average load on the machines grows more slowly for a long time the medium jobs cause effect 3 rather than effect 2. By the time the average load on the machines is on the order that makes effect 2 more likely there is with high probability already one of the extremely large jobs on the machines that dominates the makespan. As a result the ratios of the makespans for all algorithms are much lower for larger machine numbers. Figure 8 shows the results for 100 machines on the same workstation job sequence that was used for the 10 machine experiment in Figure 4 (top). The curve of Graham's algorithm is identical to that of Albers.

6 Conclusion

We saw that the performance of scheduling algorithms depends heavily on the workload characteristics. For workloads with a low variability the simple greedy algorithm by Graham has the best performance. For highly variable real workloads, however, the new algorithms often outperform Graham's algorithm. Our results also show the importance of choosing the right workload when evaluating scheduling algorithms experimentally. In particular, we observed that standard probability distributions do not often capture important characteristics of real workloads very well.

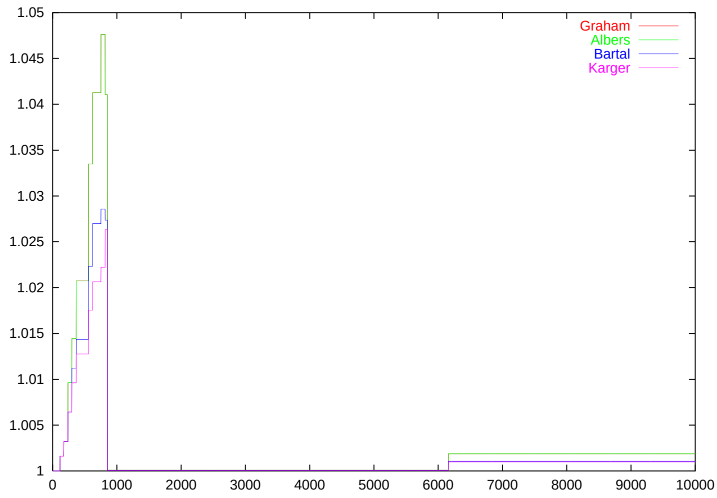

Figure 8: Typical results for experiments with workstation jobs for 100 machines

7 Acknowledgements

We would like to thank Chad Vizino and Mark Levine at the PSC for providing us with access to logs from their Cray J90's and their Cray C90 and helping us to interpret these logs. Thanks also to Dror Feitelson for making public the CTC, NASA and KTH traces. Furthermore, we would like to thank Arne Wichman at MPI for collecting the workstation jobs for us. Finally we thank two anonymous referees for helpful comments.

References

[1] S. Albers. Better bounds for online scheduling. In *Proc. 29th Annual ACM Symposium on Theory of Computing*, pages 130–139, 1997.

[2] Y. Bartal, A. Fiat, H. Karloff and R. Vohra. New algorithms for an ancient scheduling problem. *Journal of Computer and System Sciences*, 51:359–366, 1995.

[3] Y. Bartal, H. Karloff and Y. Rabani. A better lower bound for on-line scheduling. *Information Processing Letters*, 50:113–116, 1994.

[4] J. L. Bentley, D. S. Johnson, F. T. Leighton and C. C. McGeoch. An experimental study of bin packing. In *Proc. 21st Annual Allerton Conference on Communication, Control, and Computing*, pages 51–60, 1983.

[5] B. Chen, A. van Vliet and G.J. Woeginger. Lower bounds for randomized online scheduling. *Information Processing Letters*, 51:219–222, 1994.

[6] E. G. Coffman Jr., C. A. Courcoubetis, M. R. Garey, D. S. Johnson, L. A. McGeoch, P. W. Shor, R. R. Weber and M. Yannakakis. Fundamental discrepancies between average-case analyses under discrete and continuous distributions: A bin packing case study. In *Proc. 23rd Annual ACM Symposium on Theory of Computing*, pages 230–40, 1991.

[7] E.G. Coffman, L. Flatto, M.R. Garey and R.R. Weber. Minimizing expected makespans on uniform processor systems, Advances in Applied Probability, 19:177–201, 1987.

[8] E.G. Coffman, L. Flatto and G.S. Lueker. Expected makespans for largest-first multiprocessor scheduling, *Performance '84*, Elsevier Science Publishers, pages 491–506, 1984.

[9] E.G. Coffman Jr., G.N. Frederickson and G.S. Lueker. Expected makespans for largest-first sequences of independent tasks on two processors, *Math. Oper. Res.*, 9:260-266, 1984.

[10] U. Faigle, W. Kern and G. Turan. On the performance of on-line algorithms for particular problems. *Acta Cybernetica*, 9:107–119, 1989.

[11] D.G. Feitelson and L. Rudolph, editors. *Job Scheduling Strategies for Parallel Processing (IPPS 95). Workshop, Santa Barbara, CA, USA, 25. April, 1995: Proceedings*, *Springer Lecture Notes in Computer Science*, Volume 949, 1995.

[12] D.G. Feitelson and L. Rudolph, editors. *Job Scheduling Strategies for Parallel Processing (IPPS 96). Workshop, Honolulu, Hawaii, April 16, 1996: Proceedings*, *Springer Lecture Notes in Computer Science*, Volume 1162, 1996.

[13] G. Galambos and G. Woeginger. An on-line scheduling heuristic with better worst case ratio than Graham's list scheduling. *SIAM Journal on Computing*, 22:349–355, 1993.

[14] R.L. Graham. Bounds for certain multi-processing anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

[15] L.A. Hall, D.B. Shmoys and J. Wein. Scheduling to minimize average completion time: Off-line an on-line algorithms. In *Proc. 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 142–151, 1996.

[16] M. Harchol-Balter. The Effect of Heavy-Tailed Job Size Distributions on Computer System Design. *Proceedings of ASA-IMS Conference on Applications of Heavy Tailed Distributions in Economics, Engineering and Statistics*, June 1999.

[17] R. Jain. *The Art of Computer Systems Performance Analysis*, Wiley, 1991.

[18] D.R. Karger, S.J. Phillips and E. Torng. A better algorithm for an ancient scheduling problem. *Journal of Algorithms*, 20:400–430, 1996.

[19] M.W.P. Savelsbergh, R.N. Uma and J. Wein. An experimental study of LP-based approximation algorithms for scheduling problems. In *Proc. 8th ACM-SIAM Symposium on Discrete Algorithms*, pages 453–462, 1998.

[20] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–208, 1985.

[21] J. Sgall. A lower bound for randomized on-line multiprocessor scheduling, *Information Processing Letters*, 63:51–55, 1997.

[22] J. Sgall. On-line scheduling. In *Online Algorithms: The State of the Art*, A. Fiat and G.J. Woeginger. Springer Lecture Notes in Computer Science, Volume 1224, pages 196–231, 1998.

[23] D. Shmoys, J. Wein and D.P. Williamson. Scheduling parallel machines on-line. *SIAM Journal on Computing*, 24:1313–1331, 1995.

[24] D. von Seggern. *CRC Standard Curves and Surfaces*. CRC Press, 1993.