Ronald Graham: Laying the Foundations of Online Optimization

Susanne Albers

Abstract. This chapter highlights fundamental contributions made by Ron Graham in the area of online optimization. In an online problem relevant input data is not completely known in advance but instead arrives incrementally over time. In two seminal papers on scheduling published in the 1960s, Ron Graham introduced the concept of *worst-case approximation* that allows one to evaluate solutions computed online. The concept became especially popular when the term *competitive analysis* was coined about 20 years later. The framework of approximation guarantees and competitive performance has been used in thousands of research papers in order to analyze (online) optimization problems in numerous applications.

2010 Mathematics Subject Classification: 68M20, 68Q25, 68R99, 90B35
Keywords and Phrases: Scheduling, makespan minimization, algorithm, competitive analysis

An architect of discrete mathematics

Born in 1935, Ron Graham entered university at age 15. Already at that time he was interested in a career in research. He first enrolled at the University of Chicago but later transferred to the University of California at Berkeley, where he majored in electrical engineering. During a four-year Air Force service in Alaska he completed a B.S. degree in physics at the University of Alaska, Fairbanks, in 1958. He moved back to the University of California at Berkeley where he was awarded a M.S. and a Ph.D. degree in mathematics in 1961 and 1962, respectively.

Immediately after graduation Ron Graham joined Bell Labs. Some friends were afraid that this could be the end of his research but, on the contrary, he built the labs into a world-class center of research in discrete mathematics and theoretical computer science. Ron Graham rose from Member of Technical Staff to Department Head and finally to Director of the Mathematics Center

Figure 1: Ron Graham at work and at leisure. Pictures taken in New Jersey in the late 1060s and mid 1970s, respectively. Printed with the permission of Ron Graham.

at Bell Labs. After establishment of AT&T Labs Research he served as the first Vice President of the Information Sciences Research Lab and later became the first Chief Scientist of AT&T Labs. After 37 years of dedicated service he retired from AT&T in 1999. Since then he has held the Jacobs Endowed Chair of Computer and Information Science at the University of California at San Diego.

Ron Graham is a brilliant mathematician. He has done outstanding work in Ramsey Theory, quasi-randomness, the theory of scheduling and packing and, last not least, computational geometry. The "Graham scan" algorithm for computing the convex hull of a finite set of points in the plane is standard material in algorithms courses. His creativity and productivity are witnessed by more than 300 papers and five books. Ron Graham was a very close friend of Paul Erdős and allowed to look not only after his mathematical papers but also his income. Together they have published almost 30 articles. Ron Graham is listed in the *Guinness Book of Records* for the use of the largest number that ever appeared in a mathematical proof. He has many interests outside mathematics and, in particular, a passion for juggling. It is worth noting that he served not only as President of the American Mathematical Society but also as President of the International Jugglers' Association.

Ron Graham has received numerous awards. He was one of the first recipients of the Pólya Prize awarded by the Society for Industrial and Applied Mathematics. In 2003 he won the Steele Prize for Lifetime Achievement awarded by the American Mathematical Society. The citation credits Ron Graham as "one of the principle architects of the rapid development worldwide of discrete mathematics in recent years" [2].

Scheduling and performance guarantees

The technical results presented in this chapter arose from extensive research on scheduling theory conducted at Bell Labs in the mid 1960s. Even today they exhibit some remarkable features: (1) They can be perfectly used to teach the concepts of provably good algorithms and performance guarantees to non-specialists, e.g., high school students or scientists from other disciplines. (2) The specific scheduling strategies are frequently used as subroutines to solve related scheduling problems. (3) The results stimulate ongoing research; some major problems are still unresolved.

Consider a sequence $\sigma = J_1, \ldots, J_n$ of jobs that must be scheduled on m identical machines operating in parallel. Job J_i has a processing time of p_i, $1 \leq i \leq n$. The jobs of σ arrive one by one. Each job J_i has to be assigned immediately and irrevocably to one of the machines without knowledge of any future jobs J_k, $k > i$. Machines process jobs non-preemptively: Once a machine starts a job, this job is executed without interruption. The goal is to minimize the makespan, i.e. the maximum completion time of any job in the schedule constructed for σ.

The scheduling problem defined above is an online optimization problem. The relevant input arrives incrementally. For each job J_i an algorithm has to make scheduling decisions not knowing any future jobs J_k with $k > i$. Despite this handicap, a strategy should construct good solutions. Graham [5] proposed a simple greedy algorithm. The algorithm is also called *List* scheduling, which refers to the fact that σ is a list of jobs.

> Algorithm List: Schedule each job J_i on a machine that currently has the smallest load.

The load of a machine is the sum of the processing times of the jobs presently assigned to it.

A natural question is, what is the quality of the solutions computed by *List*. Here Graham introduced the concept of worst-case approximation. For any job sequence σ, compare the makespan of the schedule constructed by *List* to that of an optimal schedule for σ. How large can this ratio grow, for any σ? Formally, let *List(σ)* denote the makespan of *List*'s schedule for σ. Furthermore, let *OPT(σ)* be the makespan of an optimal schedule for σ. We would like to determine

$$c = \sup_{\sigma} \frac{List(\sigma)}{OPT(\sigma)},$$

which gives a worst-case performance guarantee for *List*. In online optimization such a guarantee is called *competitive ratio*. Following Sleator and Tarjan [8], an online algorithm A is c-competitive if, for any input, the cost of the solution computed by A is at most c times that of an optimal solution for that input.

Graham [5] gave an elegant proof that *List* is $(2 - 1/m)$-competitive, i.e. remarkably *List* achieves a small constant performance ratio. For the proof, fix an arbitrary job sequence σ and consider the schedule computed by *List*. Without

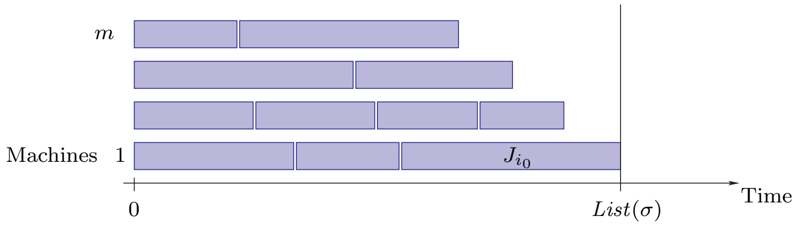

Figure 2: Analysis of *List*

loss of generality, number the machines in order of non-increasing load. Hence machine 1 is one having the highest load and defines the makespan. Figure 2 depicts an example. In the time interval $[0, List(\sigma))$ machine 1 continuously processes jobs. Any other machine j, $2 \leq j \leq m$, first processes jobs and then may be idle for some time. Let J_{i_0} be the job scheduled last on machine 1. We observe that in *List*'s schedule J_{i_0} does not start later than the finishing time of any machine j, $2 \leq j \leq m$, because *List* assigns each job to a least loaded machine. This implies that the idle time on any machine j, $2 \leq j \leq m$, cannot be higher than p_{i_0}, the processing time of J_{i_0}. Considering the time interval $[0, List(\sigma))$ on all the m machines we obtain

$$mList(\sigma) \leq \sum_{i=1}^{n} p_i + (m-1)p_{i_0}.$$

Dividing by m and taking into account that $p_{i_0} \leq \max_{1 \leq i \leq n} p_i$, we obtain

$$List(\sigma) \leq \frac{1}{m} \sum_{i=1}^{n} p_i + (1 - \frac{1}{m}) \max_{1 \leq i \leq n} p_i.$$

A final argument is that the optimum makespan $OPT(\sigma)$ cannot be smaller than $\frac{1}{m}\sum_{i=1}^{n} p_i$, which is the average load on the m machines. Moreover, obviously $OPT(\sigma) \geq \max_{1 \leq i \leq n} p_i$. We conclude that $List(\sigma) \leq OPT(\sigma) + (1 - 1/m)OPT(\sigma) = (2 - 1/m)OPT(\sigma)$.

Graham [5] also showed that the above analysis is tight. *List* does not achieve a competitive ratio smaller than $2 - 1/m$. Consider the specific job sequence σ consisting of $m(m-1)$ jobs of processing time 1 followed by a large job having a processing time of m. *List* distributes the small jobs evenly among the m machines so that the final job cause a makespan of $m - 1 + m = 2m - 1$. On the other hand the optimum makespan is m because an optimal schedule will reserve one machine for the large job and distribute the small jobs evenly among the remaining $m - 1$ machines. Figure 3 shows the schedules by *List* and *OPT*.

The above nemesis job sequence motivated Graham to formulate a second algorithm. Obviously *List*'s performance can degrade if large jobs arrive at

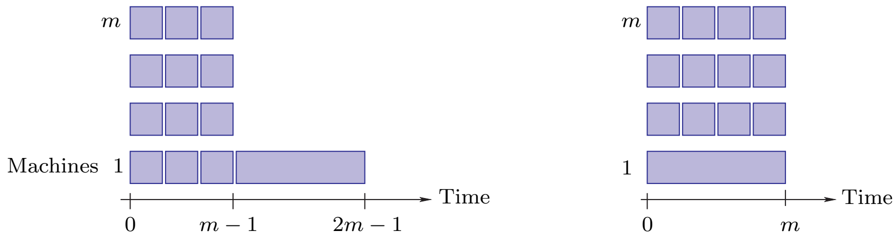

Figure 3: The worst-case performance of *List*. Online schedule (left) and an optimal schedule (right).

the end of the input sequence. Why not sort the jobs initially? Graham [6] proposed a *Sorted List* algorithm that first sorts the jobs in order of non-increasing processing time and then applies *List* scheduling. Of course *Sorted List* is not an online algorithm because the entire job sequence must be known and rearranged in advance.

Graham [6] proved that *Sorted List* achieves a worst-case approximation ratio of $4/3 - 1/(3m)$. The analysis is more involved than that of *List* but the global idea can be described in one paragraph: Consider an arbitrary sorted job sequence σ and assume without loss of generality that the last job of σ defines *Sorted List*'s makespan. If this is not the case, then one can consider the prefix sequence σ' such that the last job of σ' defines *Sorted List*'s makespan for σ' and σ. It suffices to consider two cases. (1) If the last job J_n of σ has a processing time p_n of at most $OPT(\sigma)/3$, then using the same arguments as above one can establish a performance factor of $4/3 - 1/(3m)$. (2) If $p_n > OPT(\sigma)/3$, then all jobs of σ have a processing time greater than $OPT(\sigma)/3$. Hence in an optimal schedule each machine can contain at most two jobs and $n \leq 2m$. Assume for simplicity $n = 2m$. One can show that there exists an optimal schedule that pairs the largest with the smallest job, the second largest with the second smallest job and so on. That is, the pairing on the m machines is $(J_1, J_{2m}), (J_2, J_{2m-1}), \ldots, (J_m, J_{m+1})$. If $n = 2m - k$, for some $k \geq 1$, then there is an optimal schedule that is identical to the latter pairing except that $J_1, \ldots, J_k$ are not combined with any other job. *Sorted List* produces a schedule that is no worse than this optimal assignment, i.e., in this case the performance ratio is equal to 1.

The above results led to a considerable body of further research. It was open for quite some time if online algorithms for makespan minimization can attain a competitive ratio smaller than $2 - 1/m$. It turned out that this is indeed possible. Over the past 20 years the best competitiveness of deterministic online strategies was narrowed down to $[1.88, 1.9201]$. More specifically, there exists a deterministic online algorithm that is 1.9201-competitive, and no deterministic online strategy can attain a competitive ratio smaller than 1.88. If job preemption is allowed, i.e., the processing of a job may be stopped and resumed later, the best competitiveness drops to $e/(e-1) \approx 1.58$. The book chapter [7] contains a good survey of results.

During the last few years researchers have explored settings where an online algorithm is given extra information or ability to serve the job sequence. For instance, on online algorithm might be able to migrate a limited number of jobs or alternatively might know the total processing time of all jobs in σ. In these scenarios significantly improved performance guarantees can be achieved. Using limited job migration, the competitiveness reduces to approximately 1.46. The recent manuscript [1] points to literature for these extended problem settings. Nonetheless a major question is still unresolved. What is the best competitive ratio that can be achieved by randomized online algorithms? It is known that no randomized online strategy can attain a competitiveness smaller than $e/(e-1)$. However, despite considerable research interest, no randomized online algorithm that provably beats deterministic ones, for general m, has been devised so far.

Finally, as mentioned above, the design and analysis of online algorithms has become a very active area of research. We refer the reader to two classical books [3, 4] in this field.

References

[1] S. Albers and M. Hellwig. On the value of job migration in online makespan minimization. *Proc. 20th European Symposium on Algorithms*, Springer LNCS 7501, 84–95, 2012.

[2] AMS document about the 2003 Steele Prize. Accessible at `http://en.wikipedia.org/wiki/Ronald_Graham`.

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

[4] A. Fiat and G.J. Woeginger (eds). *Online Algorithms: The State of the Art.* Springer LNCS 1442, 1998.

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

[6] R.L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal of Applied Mathematics*, 17(2):416–429, 1969.

[7] K. Pruhs, J. Sgall and E. Torng. Online scheduling. *Handbook on Scheduling*, edited by J. Y-T. Leung. Chapman & Hall / CRC. Chapter 15, 2004.

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

Susanne Albers
Department of Computer Science
Humboldt-Universität zu Berlin
Unter den Linden 6
10099 Berlin
Germany
albers@informatik.hu-berlin.de