The mystical power of twoness: in memoriam Eugene L. Lawler

Citation for published version (APA):

Document status and date:
Published: 01/01/1998

Publisher Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
Memorandum COSOR 98-05

The Mystical Power of Twoness:
In Memoriam Eugene L. Lawler

J.K. Lenstra


Eindhoven, March 1998
The Netherlands
The Mystical Power of Twoness:
In Memoriam Eugene L. Lawler

Jan Karel Lenstra
Eindhoven University of Technology; CWI, Amsterdam

Abstract. This paper reviews the work of Eugene L. Lawler, one of the early investigators of combinatorial optimization and the architect of deterministic machine scheduling theory. After sketching his research career and examining the development of his interests, I describe some of his seminal results in scheduling theory and recall one of his contributions to scientific journalism.

1. Introduction
Eugene L. Lawler (1933–1994) was one of the earliest researchers who concentrated on combinatorial optimization as a field of investigation. His textbook on networks and matroids is a classic, and several of his papers became benchmark references. His research in sequencing and scheduling was instrumental in stimulating and unifying an area that, prior to his work, was a rather unsystematic hodgepodge. He also made significant contributions to algorithmic graph theory, complexity theory, and computational biology. In addition, he was a phenomenal expositor, and he was the social conscience of the Computer Science Division at Berkeley.

In this paper I attempt to review his work, with an emphasis on his research in deterministic machine scheduling. The paper is based on a lecture given at the 16th International Symposium on Mathematical Programming in Lausanne on August 25, 1997. It supplements the personal reminiscences that he wrote in 1991 [23] and an obituary that appeared in 1994 in Optima, the newsletter of the Mathematical Programming Society [39]. A selection of his publications [1] and two issues of Mathematical Programming containing fourteen papers dedicated to his memory [30] will appear shortly.

2. An overview in six dimensions
Gene Lawler went to college in Tallahassee, Florida. He began graduate study at Harvard in 1954. After interruptions by law school, the army and employment at a real job shop, he obtained his PhD in Applied Mathematics
in 1962. He then joined the faculty of Electrical Engineering at the University of Michigan in Ann Arbor. In 1970 he moved to the EECS Department at the University of California in Berkeley. He died on September 2, 1994.

Figure 1 is a map of his scientific life. Time runs vertically. His career is represented by the vertical bar at the left-hand side. Figure 1(a) lists his publications, with algorithmic or analytical approaches indicated by capitals and problem areas by colors. Figure 1(b) classifies the publications according to style. Figure 1(c) gives a further partitioning of the papers on scheduling problems.

Some caveats are in order. I have applied a fair amount of judgemental rounding in classifying the publications, especially in deciding about the algorithmic approach taken. Several D papers also deal with B or E, and the last scheduling paper is not only A but also E. I have included only final publications and deleted prepublications as well as translations (into Czech, Dutch, Hungarian, and Japanese).

Figure 1(a) shows that Gene, after his early work in switching theory, became interested in combinatorial optimization. Most of his work is concerned with networks and matroids or sequencing and scheduling, perhaps with a gradual shift from the former area to the latter. In the 1990's, he turned to combinatorial problems in the new field of computational biology.

The algorithmic approaches also show a clear picture. He started by studying enumerative methods, first branch-and-bound and then dynamic programming. Under the influence of complexity theory, he became interested in polynomial-time optimization and, later and to a lesser extent, in performance guarantees for polynomial-time approximation algorithms. I will return to this development in Section 3.

Figure 1(b) highlights two books: his textbook on networks and matroids [17] and an edited collection of chapters on the traveling salesman problem [26]. Especially the first book had a pronounced impact and is as useful today as it was in 1976. He contributed a number of influential surveys, most notably on branch-and-bound (see Section 3), well-solvable cases of the TSP [8], and scheduling [10, 21, 27]. His lucid writing style and innate wit made him a superb scientific journalist. The Sputnik paper [20] will serve as an example in Section 6.

The classification given in Figure 1(c) will be discussed in Section 5.

3. Early work
I will explore the development of Gene's research interests by examining nine publications from the period 1962–1975.
A approximation algorithms / B branch-and-bound / C complexity
D dynamic programming / E easy: polynomial-time optimization
F full: most of A-E / X none

<table>
<thead>
<tr>
<th>switching theory</th>
<th>book</th>
<th>scheduling</th>
<th>comp. biology</th>
</tr>
</thead>
<tbody>
<tr>
<td>networks + matroids</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>other comb. (opt.)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1. A research map.

E.L. Lawler. The quadratic assignment problem. *Management Science* 9 (1963), 586-599. – Gene’s first paper in combinatorial optimization is still an important reference. Some later papers on special cases of the quadratic assignment problem propose lower bounds that are dominated by the bound developed here.

E.L. Lawler, D.E. Wood. Branch-and-bound methods: a survey. *Operations Research* 14 (1966), 699-719. – This is a great survey. It unifies approaches that originated in distinct areas. The LP-based integer programming methods of Land, Doig and Dakin, Balas’ additive algorithm for the same problem, the synthesis and covering algorithms for switching circuits proposed by Roth, Karp and Lawler, the game trees from artificial intelligence, and the backtrack methods from combinatorics are all put in the branch-and-bound framework. The paper became a citation classic [22].

At this point, Gene had paid his dues to branch-and-bound. His research was moving to more structured forms of enumeration. I remember that, at Bernard Roy’s 1974 Versailles Summer School on Combinatorial Programming, where we first met, he tallied the papers presented and commented that there was too much branch-and-bound to his liking. It is not that he came to dislike the approach. He admired his younger colleagues who turned it from an art into a science and who used Lagrangian relaxation and polyhedral combinatorics in replacing branch-and-bound by continue-to-bound-and-avoid-to-branch. But he followed his own gifts and redirected his interests to pseudopolynomial and truly polynomial algorithms.

E.L. Lawler, J.M. Moore. A functional equation and its application to resource allocation and sequencing problems. *Management Science* 16 (1969), 77-84. – Gene became the wizard of finite-state dynamic programming recursions. Among his many papers on the approach, the most salient ones are the Lawler-Moore paper and his 1977 paper on a pseudopolynomial-time algorithm for minimizing total tardiness on a single machine [18].


E.L. Lawler. Matroid intersection algorithms. *Mathematical Programming* 9 (1975), 31-56. – As Gene has commented elsewhere [23], his work on matroid optimization was inspired by the insights of Jack Edmonds. It culminated in this landmark paper, which gave a polynomial-time algorithm for finding an optimum intersection of two matroids. It is undoubtedly the most important “mystical 2” in his work. Gene was not the only one who found out that often 2 is easy while 3 seems undoable.

R.M. Karp. Reducibility among combinatorial problems. In: R.E. Miller, J.W. Thatcher (editors). *Complexity of Computer Computations*. Plenum Press, New York, NY (1972), 85-103. – An explanation was provided by Dick Karp’s famous paper on computational complexity, in which he established NP-completeness of 21 fundamental problems. Dick attributes two reductions to Gene: from Vertex Cover to Directed Hamiltonian Circuit and from Exact Cover to 3-Dimensional Matching. The latter result gives another mystical 2: 2-dimensional (or bipartite) matching is easy but 3-dimensional matching is hard. Gene would invariably comment that this is why a world with two sexes has been devised.

A quite different mystical 2 in his life was his habit of having second thoughts about the topic of his talks. His preferred subject was what he had been working on the night before. At the 1974 Versailles conference he was asked to talk about the quadratic assignment problem. He suggested complexity but was not allowed any second thoughts. He then gave, after hours, an illicit talk about NP-completeness.

Eventually, Gene was not primarily interested in what makes a problem hard but in what makes it well solvable, or approximable within a decent bound. In other words, he was concerned with the mystical 2 rather than the magical 3. At the same time, he knew the limitations of worst-case guarantees on running time or solution quality. Probabilistic analysis may
not have been his favorite approach, but he was fully convinced of the im-
portance and computational power of superpolynomial techniques such as
branch-and-bound and local search.

4. Scheduling theory is something of a jungle
There is a bewildering variety of problem types in deterministic machine
scheduling theory. The investigation of their complexity status is not as
simple as "2 is easy, 3 is hard," and may surprise and confuse the analyst.
Let me give an example.

A single machine needs to process a finite set $N$ of jobs. The machine is
available from time 0 onwards and can process at most one job at a time.
Each job $j \in N$ requires processing during an uninterrupted period of length
$p_j$; it is convenient to write $P = \sum_{j \in N} p_j$. We wish to find a schedule of
minimum length. This is trivial: any schedule without machine idle time
has length $P$ and is optimal. When we introduce a second machine, the
problem becomes NP-hard: a schedule of length $P/2$ exists if and only if
the set of processing times can be split into two subsets of equal sum. We
now allow job preemption; a job may be split but cannot be processed by
both machines at the same time. The problem becomes easy again: if there
is a job that is longer than the others together, it gets a machine on its
own and the other jobs go on the second machine; otherwise, we schedule
the jobs in arbitrary order on one machine, at time $P/2$ we interrupt the
job that the machine is working on, we start its remainder at time 0 on the
second machine, and continue with the jobs that are left. Returning to the
single-machine problem, we now introduce job release dates and stipulate
that a job cannot start before it is released. The problem remains easy: just
schedule the jobs in order of nondecreasing release dates. We then add job
deadlines, by which the jobs must be finished. A simple application of the
equal-split argument tells us that the problem has become NP-hard.

We discard release dates and deadlines but change the objective. We
wish to find a schedule that minimizes the sum of the job completion times,
not their maximum. This problem is solved by the SPT rule: put the jobs
in order of nondecreasing processing times. When we add a second machine,
the problem remains easy (SPT still works), but when we add release dates
instead, it becomes NP-hard (by a less obvious reduction). This is the
reverse of what we saw for the makespan objective.

Figure 2 depicts the situation in English and in what Gene called Sched-
ulese, the language proposed by Graham et al. [10] as a revision of the
classification system devised by Conway, Maxwell and Miller in their book
Figure 2. Reducibility among scheduling problems.
The above example illustrates a game that Dick Karp, Ben Lageweg, Gene and I played on an afternoon in September 1975 in the old Mathematisch Centrum in Amsterdam, amid its characteristic smell of overworked copying machines and stale beer. (The Amstel brewery was next door.) We soon realized that we needed the aid of a computer. Ben wrote a program that generated 4,536 problem types, involving a single machine, identical, uniform and unrelated parallel machines, open shops, flow shops, job shops, various sorts of job characteristics such as preemption, precedence constraints, release dates and deadlines, and eight optimality criteria. With the known results as input, the program classified each problem as solvable in polynomial time, NP-hard, or open. The most useful part of the output was given by the borderlines: the maximal easy problems, the minimal and maximal open ones, and the minimal hard ones. Each new listing of open problems contained targets that we had overlooked in the previous round.

The computer-aided game started when we decided on a surprise for Alexander Rinnooy Kan on the occasion of his thesis defense. A listing of all scheduling problems with their complexity status seemed a good complement to his thesis. The surprise to us was that Ben’s program proved to be a much more useful research tool than we had anticipated [15, 16]. This does not imply that all of those 4,536 problems are for real. Many of them are quite artificial and exist only because a computer has generated them.

The automatic classification system is presently maintained by Peter Brucker and Sigrid Knust at the Universität Osnabrück for an extended class of deterministic machine scheduling problems; see http://www.mathematik.uni-osnabrueck.de/research/OR/class/.

5. Precedence constraints and preemption
When we were generating random walks in scheduling space in 1975, Gene’s main interest was not in NP-hardness proofs but in polynomial-time algorithms. His principal concern was the algorithmic treatment of precedence constraints and preemption. Figure 1(c) supports this observation: almost 40% of his scheduling work dealt with precedence constraints and over 40% with preemption. Gene’s last manuscript on scheduling [24] considers precedence constraints with communication delays; see Section 7. His last research discussion was about staircase algorithms for handling preemption on parallel processors.

In Section 3 his Management Science paper of 1973 was cited. It gives a solution to the following problem. A finite set $N$ of jobs is to be processed
on a single machine. Each job \( j \in N \) has a processing time \( p_j \) and a nondecreasing cost function \( f_j \); if job \( j \) is completed at time \( t \), a cost \( f_j(t) \) is incurred. Furthermore, a digraph \( G \) with vertex set \( N \) is given; if \( G \) contains a path from job \( j \) to job \( k \), then \( k \) cannot start before \( j \) has been completed. We wish to find a schedule that satisfies the precedence constraints and minimizes the maximum job completion cost.

We need some notation: \( L \subseteq N \) is the set of jobs without successors in \( G \), \( P = \sum_{j \in N} p_j \) is the schedule length, and \( f^*(S) \) is the cost of an optimal schedule for the job set \( S \subseteq N \). The least cost last rule selects a job \( l \in L \) satisfying \( f_l(P) = \min_{j \in L} f_j(P) \), puts job \( l \) in the last position of the schedule, and repeats. Why does this work? We clearly have \( f^*(N) \geq f_l(P) \) and \( f^*(N) \geq f^*(N \setminus \{l\}) \), and hence

\[
 f^*(N) \geq \max\{f_l(P), f^*(N \setminus \{l\})\}.
\]

But the right-hand side of this inequality is precisely the cost of an optimal schedule under the condition that \( l \) is scheduled last.

The original proof uses a job interchange argument. Gene’s above more elegant proof extends to the case that there are release dates and preemption is allowed [3]. Without release dates, there is no use to preemption; with release dates, the nonpreemptive case is NP-hard.

A problem that occupies a central position in Gene’s work is the minimization of total weighted completion time on a single machine subject to precedence constraints. Each job \( j \in N \) now also has a weight \( w_j \) and, if \( C_j \) denotes its completion time, we wish to find a schedule minimizing \( \sum_{j \in N} w_j C_j \). In the absence of precedence constraints, Smith’s ratio rule [35] uses a simple job interchange relation to show that the jobs must be scheduled in order of nondecreasing values \( p_j/w_j \). Various special types of precedence constraints can still be solved in \( O(n \log n) \) time, typically by a decomposition approach that is based on a string interchange relation. Horn [11] solved the case of tree-type constraints, Lawler [19] and Monma and Sidney [32, 33] handled series-parallel constraints. Very recently, Goe­mans and Williamson [9] gave a new proof for Gene’s algorithm, using a linear programming formulation of the precedence-constrained single-machine problem due to Queyranne and Wang [34], which completely describes the scheduling polyhedron in the case of series-parallel constraints. Gene’s NP-hardness proof for general precedence constraints [19] was a breakthrough at the time.

Series-parallel digraphs already occur in Gene’s very first paper [29]. A digraph is series-parallel if its transitive closure is transitive series-parallel,
• is tsp
 Artikel for tsp 
 Artikel for tsp 
 Artikel for tsp 
 Artikel for tsp 
 Artikel for tsp 
 Artikel for tsp 
 Artikel for tsp 
 Artikel for tsp 
 Artikel for tsp 

(a) Composition of transitive series-parallel (tsp) digraph.

(b) Decomposition of series-parallel digraph.

(c) Smallest non-series-parallel digraph.

Figure 3. Series-parallel precedence constraints.

which means that it can be recursively composed as follows (cf. Figure 3(a)): the digraph with a single vertex and no arcs is transitive series-parallel, and if the digraphs \((V_1, A_1)\) and \((V_2, A_2)\) are transitive series-parallel with \(V_1 \cap V_2 = \emptyset\), then so are the series composition \((V_1 \cup V_2, A_1 \cup A_2 \cup (V_1 \times V_2))\) and the parallel composition \((V_1 \cup V_2, A_1 \cup A_2)\). Series-parallel digraphs can be recognized in linear time [37]; the smallest digraph that is not series-parallel is the Z-digraph shown in Figure 3(c).

The structure of a series-parallel digraph is displayed by a decomposition tree (cf. Figure 3(b)). Its leaves are the vertices, a P-node represents parallel composition, and an S-node represents series composition with the convention that left precedes right. The scheduling algorithms work their way up the tree and construct an optimal schedule for a parent node out of
Let us now turn to preemption. Gene often cited an experimental law, which states that a preemptive problem should not be harder that its non-preemptive counterpart. Indeed, in many cases, the former problem is easier than the latter, and the distinction has the same flavor as that between linear and integer programming. But experimental laws have experimental exceptions. Here I know of 1.5 exceptions.

One exception occurs in job shop scheduling. If no preemption is allowed, there are two nice mystical 2's: the 2-job problem is easy [2], the 2-machine problem with a fixed number of jobs is easy [4], and the 3-machine 3-job problem is NP-hard [36]. If preemption is allowed, the 2-machine 3-job problem turns out to be NP-hard [5], a surprising and unsettling result that oversteps the law.

Half an exception occurs in scheduling unrelated parallel machines. Here each of \( n \) jobs is to be processed by one of \( m \) machines, and putting job \( j \) on machine \( i \) takes time \( p_{ij} \). If we wish to minimize schedule length, everything is as it should be. The nonpreemptive case is NP-hard, because it is a generalized packing problem; the preemptive case can be solved by an LP-based algorithm due to Lawler and Labetoulle [25]. If the objective is the sum of the completion times, the nonpreemptive case is a bipartite matching problem: allocating job \( j \) to the \( k \)th last position on machine \( i \) gives a contribution of \( kp_{ij} \) to the objective [12, 6]. The preemptive total completion time problem, however, is one of the more vexing open questions.

Gene made important contributions to preemptive scheduling: the LP-based algorithm cited above, the staircase algorithm for minimizing makespan on uniform parallel machines subject to release dates with Jacques Labetoulle and others [14], and the polymatroidal flow algorithm for the same problem with release dates and deadlines with Chip Martel [28].

6. The great mathematical Sputnik of 1979

The Sputnik paper, which appeared in The Sciences and The Mathematical Intelligencer in 1980 [20], is Gene's most memorable foray into scientific journalism. It deals with the advent of the ellipsoid method, but also with the way in which professional journalists handle their material.

In January 1979, Rainer Burkard brought a Doklady paper to Oberwolfach, which he had obtained through a Polish colleague. The author was Leonid Khachiyan, the result was that the ellipsoid method solved linear programming problems in polynomial time, and the proofs were lacking. Gene brought the paper to Amsterdam, made a rough translation with the help
of Milan Vlach, a visitor from Prague, and sent it around. Within weeks, the result had been verified, and the West knew about it before the Doklady issue had made it to the libraries. Science published a well-researched article about Khachiyan's achievement. A New York Times journalist managed to grossly misinterpret the Science article and reported on the front page that the TSP had been solved. All over the world, headlines appeared of the sort "A Soviet Discovery Rocks World of Mathematics," "Soviet Mathematician Is Obscure No More," "Discovery Rocks World of Math, Computers," and "The Russian Genius Who Has Rocked the Computer World." It took some time before the dust settled. At the end, the Times printed a half-baked retraction.

The following is a quotation from the Sputnik paper.

The Times story appears to have been based on certain unshakable preconceptions of its writer, Malcolm W. Browne. Browne telephoned George Dantzig of Stanford University, a great pioneering authority on linear programming, and tried to force him into various admissions. Dantzig’s version of the interview bears repeating. “What about the Traveling Salesman Problem?” asked Browne. “If there is a connection, I don’t know what it is,” said Dantzig. (“The Russian discovery proposed an approach for [solving] a class of problems related to the ‘Traveling Salesman Problem,’” reported Browne.) “What about cryptography?” asked Browne. “If there is a connection, I don’t know what it is,” said Dantzig. (“The theory of codes could eventually be affected,” reported Browne.) “Is the Russian method practical?” asked Browne. “No,” said Dantzig. (“Mathematicians describe the discovery . . . as a method by which computers can find solutions to a class of very hard problems that has hitherto been attacked on a hit-or-miss basis,” reported Browne.)

7. Communication delays

In the past decade, much attention has been paid to multiprocessor scheduling subject to precedence constraints with communication delays. With each arc \((j, k)\) in the precedence digraph, a delay \(c_{jk}\) is associated; if jobs \(j\) and \(k\) are allocated to different machines, then \(k\) can only start \(c_{jk}\) time units after the completion of \(j\).

The presence of positive communication delays implies that, among all the direct successors of a job, at most one, its favored child, can start at
its completion time; both jobs must then be allocated to the same machine. All the other successors will incur a delay. Similarly, among the direct predecessors of a job there is at most one favored parent. The need to choose favored children and parents causes an additional combinatorial difficulty, which implies that minimizing makespan is already NP-hard in case \( m \geq n \), i.e., in the absence of machine capacity constraints.

Let us assume that we have \( m \) identical parallel machines, \( n \) unit-time jobs, and unit-time communication delays. Without delays, the case of tree-type constraints and the case of general precedence constraints and 2 machines are both solvable in linear time. With delays, the former problem is NP-hard [31] and the latter problem is open (constituting an annoying 2 rather than a mystical 2). Gene's contribution [24, 38] is a linear-time algorithm for tree-type constraints that computes a schedule whose length exceeds the optimum by no more than \( m - 2 \) time units; hence, it solves the 2-machine problem to optimality.

I will give an outline of Gene's approach. Let the precedence constraints be given in the form of an outforest \( F \). A schedule is said to have the favored child property if every parent has exactly one child scheduled earlier than the others. An interchange argument shows that there exists an optimal schedule with the favored child property.

Given a choice of favored children, it is easy to find a minimum-length schedule. First, replace each arc \((j, l) \in F\) by an arc \((k, l)\) whenever \( j \) has a favored child \( k \neq l \); call the new outforest \( \hat{F} \). Next, ignore the delays and schedule \( \hat{F} \) in linear time, e.g., by Hu's algorithm [13].

It is apparently hard to make an optimal choice of favored children. It is easy, however, to make a choice of favored children that yields an outforest \( \hat{F}^* \) of minimum depth, and thereby solves the problem in case \( m \geq n \). For each job \( j \), the depth \( d_j \) of the subtree of \( \hat{F}^* \) rooted at \( j \) is given by

\[
\begin{align*}
    d_j &= 1 \quad \text{when } j \text{ is a leaf,} \\
    d_j &= \max\{1 + \max_{(j, k) \in F} d_k, 2 + \max_{2(j, k) \in F} d_k\} \quad \text{when } j \text{ is not a leaf,}
\end{align*}
\]

where the max 2-term denotes the highest \( d \)-value among the children of \( j \) after a child with highest \( d \)-value has been deleted. For any parent job in \( F \), any child with maximum \( d \)-value is chosen as its favored child.

This choice is not too bad when \( m \) is given as part of the problem instance. It requires a careful argument to show that it has the absolute performance guarantee mentioned above, so that it is optimal for \( m = 2 \).
8. Closing remarks
Life is more than scientific achievement. I find it difficult to imagine anyone who enjoys life more than Gene did. He was a remarkable man, not only because of his inquiring mind, but also because of his expository gift and his personal commitment.

His expository talent was, to a large extent, related to his style of research. He usually preferred to do things his own way, by going back to the original question, identifying what he saw as the essential difficulty, and eventually achieving a deeper insight as a result. He would apply the same principle when he had difficulties in understanding cumbersome presentations of other people’s work. By taking a fresh look at things, he often arrived at a simpler proof of a more general result.

His commitment was in no way restricted to topics of research. He was ready to approach any issue, scientific or not, in an intelligent and thought-provoking way. Throughout his years at Berkeley, he did everything imaginable to make the university a more humane and more stimulating place to study. He helped the individual student fight the bureaucracy, and reformed what the university taught and to whom it taught. He showed a true and pure interest in other people. It made him the most honest and inspiring friend one could wish to have.

Acknowledgements. I wish to thank Karen Aardal, Francesco Maffioli and David Shmoys for their help in taking care of Gene’s scientific legacy, and for their comments on a draft of this paper. I am grateful to Tom Liebling, who invited me to give the talk on which this paper is based, to Edmund Burke, who asked me to contribute to the first issue of the new Journal of Scheduling, and to Petra Schuurman, who provided technical assistance. This research was partially supported by NATO Collaborative Research Grant 971550.

References


22. E.L. Lawler. This week’s citation classic. *Current Contents (Arts & Humanities; Social & Behavioral Sciences; Engineering, Technology & Applied Sciences; Physical, Chemical & Earth Sciences)* 2 (January 12, 1987), 16.


---

Jan Karel Lenstra  
Department of Mathematics and Computing Science  
Eindhoven University of Technology  
P.O. Box 513  
5600 MB Eindhoven  
The Netherlands  
jkl@win.tue.nl