SCHEDULING

From Theory to Practice

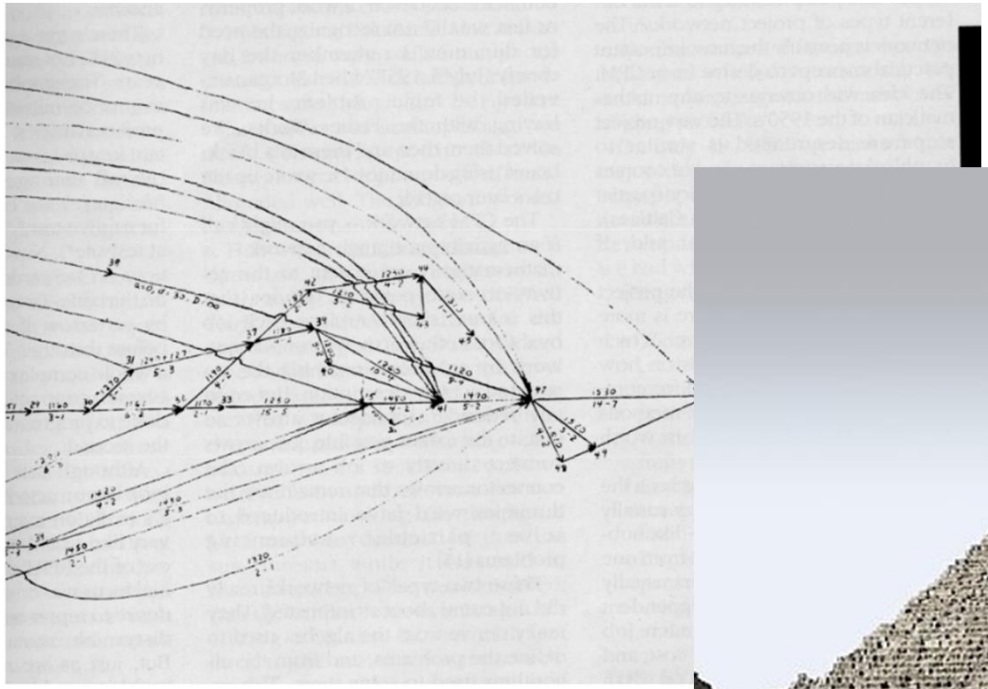
Günter Schmidt
Saarland University

guenter.schmidt@mpi-inf.mpg.de

http://people.mpi-inf.mpg.de/~gschmidt/

Metz, June 26, 2019

-3000, 1916, 1956



Machine/Shift	0	1	2	3	4
Machine 1	J_11	A	S		J_21
Machine 2	J_12		S	J_11	

What is scheduling ?

Theory has to discover, practice has to use discovery

What are we doing in theory ?

Epoche 1 Combinatorial Analysis

Flow shop with Johnson (1954);
single machine with Jackson (1955) (EDD/Lmax) and Smith (1956) (SPT/SUM Cj);
parallel machines with McNaughton (1959);
open shop twenty years later

J1	J2	J3	J4	J5	F2||Cmax = 21	J1, J5, J2, J3, J4
1	2	3	4	5	1||Lmax = 9	J5, J4, J3, J2, J1
6	5	4	3	2	1||SumCj = 15	J1, J2, J3, J4, J5

P2|pmtn|Cmax = 17,5

O2||Cmax = 21

Epoche 2 Restricting Search with Enumerative Algorithms (60th)

Branch & Bound and Dynamic Programming for practical problems with small instances applied to Single Machine, Flow Shop, Job Shop, ...

Branch & Bound

(a) a branching rule that indicates how solutions are partitioned into subsets;

(b) a lower bounding rule that computes a lower bound on the cost of any solution within the subset of solutions under consideration;

(c) optional features such as an upper bounding rule that constructs feasible solutions that can be evaluated to produce an upper bound, and dominance rules that can eliminate some subsets of solutions from further consideration.

There exist numerous results

Epoche 2 Restricting Search with Enumerative Algorithms

Branch & Bound and Dynamic Programming for practical problems with small instances applied to Single Machine, Flow Shop, Job Shop, ...

Branch & Bound
(a) a branching rule that indicates how solutions are partitioned into subsets;
(b) a lower bounding rule that computes a lower bound on the cost of any solution within the subset of solutions under consideration;
(c) optional features such as an upper bounding rule that constructs feasible solutions that can be evaluated to produce an upper bound, and dominance rules that can eliminate some subsets of solutions from further consideration.

There exist numerous results

Epoche 3 Complexity and Classification: P vs NP

Edmonds (1965) argues that a "good" algorithm is one whose running time depends polynomially on the length of the input (or size) of the problem.

Since a computer uses the binary representation of the numbers, the length L of the input is essentially bounded by the product of the number of inputs: e.g. the length of input for Rm||SUM wjCj is bounded by L = nmlog(max{pij}) + nlog(max{wj}).

An algorithm that requires O(Lk) time, where k is a constant that does not depend on L, is called a polynomial-time (or simply a polynomial) algorithm. Those algorithms that require polynomial time under the unary encoding are called pseudopolynomial; they are of some interest too, but less practical than polynomial algorithms and their behaviour strongly depends on the size of input parameters.

Epoche 3 Complexity and Classification: case of preemption

Mostly it is assumed that preemption, could simplify the problem but this is not always true.

R||SUM Cj is polynomially solvable.

R2|pmtn|SUM Cj is NP-hard in the strong sense; see Sitters (2005).

The interest in this result is that, unlike for essentially all other scheduling problems, the preemptive version appears to be harder than its polynomially solvable non-preemptive counterpart.

The same is true for bi-directional conversion, a problem which is related to scheduling (Danoura and Sakurai (1996), Schmidt (2017)).

Epoche 4 Approximate Solutions

Approximation versus heuristic algorithms: for a scheduling problem with the goal of minimizing a function F(S) over all feasible schedules S, let SO denote an optimal schedule.

A polynomial-time algorithm that finds a feasible solution SF such that F(SF) is at most ρ (where $\rho \geq 1$) times the optimal value F(SO) is called a ρ-approximation algorithm; the value of ρ is called a worst-case ratio bound.

If a problem admits a ρ-approximation algorithm, then it is said to be approximable within a factor ρ. A family of ρ-approximation algorithms is called a polynomial-time approximation scheme (PTAS) if $\rho = 1 + \varepsilon$ for any fixed $\varepsilon > 0$; if additionally the running time is polynomial with respect to both the length of the problem input and $1/\varepsilon$, then the scheme is called a fully polynomial-time approximation scheme (FPTAS).

Epoche 4 Approximate Solutions: Parallel Machines

Graham (1969) shows that the LPT rule solves P||Cmax with $\rho = 4/3 - 1/3m$.

Braun und Schmidt (2003) compare for P|pmtn|Cmax the makespan of preemptive and *i*-preemptive schedules, $0 \leq i \leq m-1$. They show that

$Cmax(i) \leq (2-2/(m/(i+1)+1))Cmax$.

Assume Cmax = SUM pj/m for m=2 and i=1 then

$Cmax(1) \leq 2 - 4/(m+2) = Cmax$, i.e. $\rho = 1$.

It is known that for m=2 an optimal schedule with at most one preemption exists.

Epoche 4 Approximate Solutions: Flow Shop

Model the original problem by an artificial F2 shop, and convert the solution to the original problem.

No algorithm of this type is known to deliver a ratio better than /m/2\ (see Gonzalez and Sahni (1978), Röck and Schmidt (1983)).

Note that for m = 3 and /m/2\ = 2 it is easy to design a 2-approximation algorithm:

Schedule the first two machines optimally by Johnson's algorithm and concatenate the resulting schedule with a block of operations on the third machine.

For m = 3, an improved (5/3)-approximation algorithm is due to Chen et al. (1996).

The best possible approximation result for Fm||Cmax is a PTAS developed by Hall (1998).

It remains unknown whether the problem with a variable number of machines is approximable within a constant factor ρ that does not depend on m.

Epoche 4 Approximate Solutions : Open Shop

The ideas from flow shops can also be applied to open shops.

Sevastianov and Woeginger (1998) show the existence of a polynomial time approximation scheme for Om||Cmax.

For the variant of the problem where the number of machines is part of the input, it is known that the existence of an approximation scheme would imply P = NP. Hence, the result draws a precise separating line between approximable cases (i.e., with m fixed) and non-approximable cases (i.e., with m part of the input) of this shop problem.

Epoche 4 Approximate Solutions: Local Search

Simulated Annealing, Tabu Search, Genetic Algorithms,
An important development in job shop scheduling with a wide applicability, is the shifting bottleneck procedure proposed by Adams, Balas and Zawack (1988).

It is a decomposition approach for problems with multiple machines. The shifting bottleneck procedure for job shop scheduling works with the disjunction graph representation coupled with a single machine problem with release and delivery times is solved (typically using the algorithm of Carlier (1982)). The bottleneck machine is the next machine to be selected and is defined as the one having the largest objective function value among the solutions of the single machine problems.

Among the local search algorithms discussed above, the tabu search algorithm of Nowicki and Smutnicki (1996a) and the guided local search algorithm employing the shifting bottleneck procedure of Balas and Vazacopoulos (1998) are the most effective.

Epoche 5 Enhanced Scheduling Models: non-availability scheduling

Starting Schmidt (1984), numerous new results, many survey papers, e.g. Ma et al. (2010), Schmidt (2000), Sanlaville and Schmidt (1998)

Parallel machines:

For P2 with one non-availability after time t a FPTAS exist, see Kacem, Sahnoune and Schmidt (2016)

Flow shop:

Blazewicz et al. (2002) investigate F2|NC|Cmax and analyze constructive and local search based heuristic algorithms. Computational results show that the algorithms perform well.

Braun et al. (2002) investigate F2|NC|Cmax and prove sufficient conditions for the optimality of Johnson's permutation in the case of given $w \geq 1$ non-availability intervals. They apply stability analysis, which answers the question of how robust an optimal schedule is if there are independent changes in the processing times of the jobs.

Epoche 5 Enhanced Scheduling Models: non-availability scheduling

Open shop:

Breit et al. (2001) and (2003) investigate O2|NC, non-pmtn|Cmax, show that the problem is NP-hard and present an approximation algorithm with a worst-case ratio of 4/3 for the problem with a single non-availability interval.

Kubzin et al. (2005) give for O2|NC, resume|Cmax two PTAS, one of which for the problem with one non-availability interval on each machine and the other for the problem with several non-availability intervals on one of the machines.

Problems with a more general structure of the non-availability intervals are not approximable in polynomial time within a constant factor, unless P=NP-

Epoche 5 Enhanced Scheduling Models: non-availability scheduling

Breakdowns:

Albers and Schmidt (2001) investigate the online version of P|NC, pmtn|Cmax. Machines can be added, break down and recover at arbitrary points of time *not known in advance*. Results are that no online algorithm (1) can construct optimal schedules, (2) can achieve a bounded competitive ratio if there may be time intervals where no machine is available and (3) an online algorithm constructs schedules with an optimal makespan if a *lookahead* of one is given, i.e., the next point in time when the set of available machines changes is known.

Kacem and Kellerer (2016) investigate the online version of 1|NC, {dj,rj} |{Cmax, Lmax} and propose approximation algorithms for Cmax with or without release dates. They show that the competitive ratio in both cases is 3/2 and that this bound is best possible for Cmax. For Lmax they propose a ≈ 1.70-approximation algorithm to solve the problem with delivery times but without release dates. This ratio is tight for the proposed algorithm and allows to establish a precise window for the best possible ratio, which is ≈[1.50, 1.70].

Epoche 5 Enhanced Scheduling Models: lotsize scheduling:

Pattloch, Schmidt and Kovalyov (2001) investigate a problem which appears in many industries. The question is how tasks of n > 2 different job types can be scheduled on machines in lots such that the number of changeovers is minimized. These changeovers occur if two tasks of different types are scheduled in sequence on a machine. Heuristics for the single and parallel machine case are presented.

The same authors (2002) investigate the problem of scheduling unit time tasks of n = 2 job types on *m* parallel identical machines. For each job type, given numbers of tasks are required to be completed by K specified deadlines. The in-process inventory capacities are given. The objective is to minimize the number of changeovers occurring between the tasks of different types.

Earlier Pattloch and Schmidt [Discrete Appl. Math. 65 (1996) 409–419] give an $O(mH)$ algorithm to solve this problem where H is the latest deadline. Here a modification of this algorithm with $O(K.\min\{K,m\})$ time complexity is given.

Epoche 5 Enhanced Scheduling Models: imprecise data

Kacem and Kellerer (2018) consider the problem of scheduling independent jobs with a common due date on a single machine with the objective of maximizing the number of early jobs.The processing times are uncertain and take any value from a certain job dependent interval. For measuring the quality of an algorithm for that problem with imprecise data they use the concept of minimizing the maximum regret. They present complexity results and dominance properties.

Schmidt (2017) invetigates a similar problem in the area of financial scheduling. In an online conversion problem a player is converting one asset into another based on a finite sequence of unknown future conversion rates taking any value from an interval [m, M] of upper and lower bounds. When a new rate is announced the player must decide when to convert all current wealth. Conversion is over when the last rate has appeared. The objective is to maximize terminal wealth after conversion.

(1) Practice R|in-tree|SUM Cj

(1) R|in-tree|SUM Cj

P2|chains|SUM Cj is NP-hard in the strong sense,

R||SUM Cj can be solved in $O(n^3)$ time according to Bruno et al. (1974) as shown in Blazewicz et al. (2001 p170).

Heuristic Algorithm for R|in-tree|SUM Cj

For t = 0, …, T determine the set of tasks without predecessors St
If some machine i becomes available at time t
(1) schedule St with a variant of Bruno et al. (1974) considering idle time
(2.1) choose job j from St with minimum remaining processing time
(2.2) should j start on i at t (upper bound for Cj) or wait for i' until t' (lower bound for Cj)

Choose the schedule with minimum SUM Cj

(2) Practice: R||MIN SetUp

(2) R||MIN SetUp

R|sequence/machine dependent setup times/costs, deadlines|MIN #SetUps can be formulated with MathProgramming;

The problem is NP-hard in the strong sense; a related problem is investigated in Pattloch, Schmidt and Kovalyov (2001) and solved by a heuristic
(1) solve the problem using the heuristic of Pattloch, Schmidt and Kovalyov (2001)
(2) solve R|deadlines| without setups; for setups solve the one machine problems

CONCLUSIONS

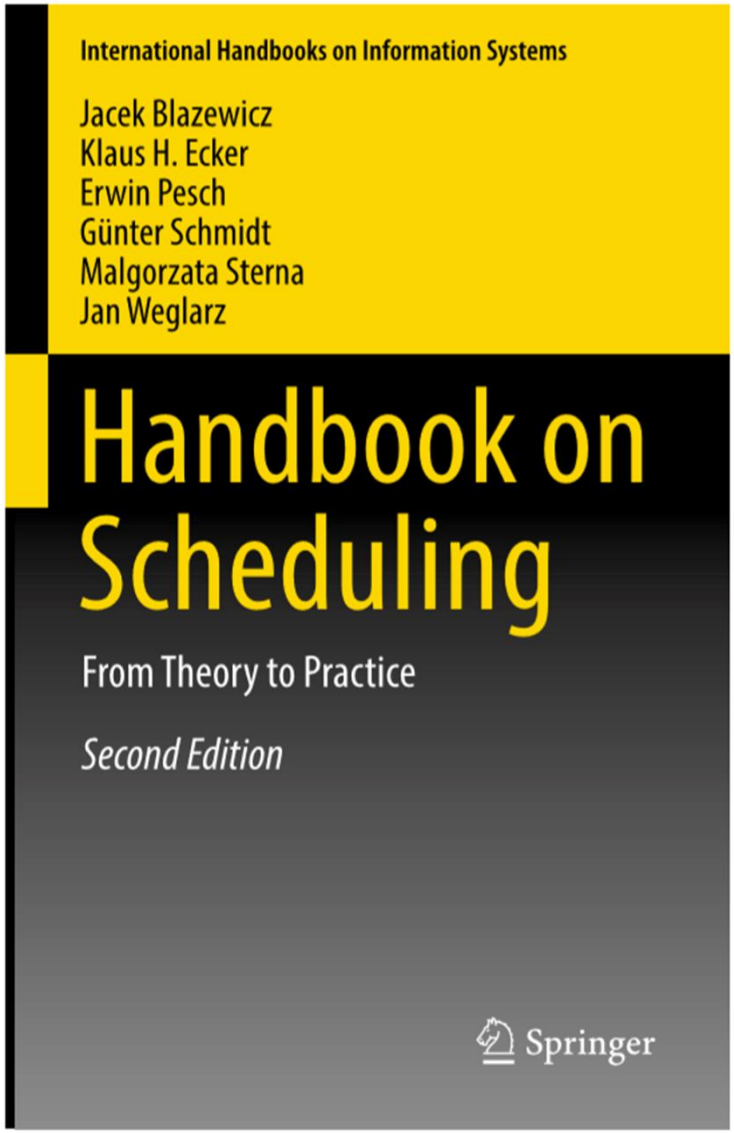
Scheduling

from theory to practice

and

from usage to discovery

!


International Handbooks on Information Systems
Jacek Blazewicz
Klaus H. Ecker
Erwin Pesch
Günter Schmidt
Malgorzata Sterna
Jan Weglarz
Handbook on Scheduling
From Theory to Practice
Second Edition
Springer