A Survey on Approximation Algorithms for Scheduling with Machine Unavailability

Florian Diedrich[1,⋆,⋆⋆,⋆⋆⋆], Klaus Jansen[1,⋆⋆,⋆⋆⋆], Ulrich M. Schwarz[1,⋆⋆,⋆⋆⋆], and Denis Trystram[2,†]

[1] Institut für Informatik, Christian-Albrechts-Universität zu Kiel, Olshausenstr. 40, 24098 Kiel, Germany
{fdi,kj,ums}@informatik.uni-kiel.de
[2] LIG – Grenoble University, 51 avenue Jean Kuntzmann, 38330 Montbonnot Saint-Martin, France
denis.trystram@imag.fr

Abstract. In this chapter we present recent contributions in the field of sequential job scheduling on *network machines* which work in parallel; these are subject to temporary unavailability. This unavailability can be either unforeseeable (online models) or known a priori (offline models). For the online models we are mainly interested in preemptive schedules for problem formulations where the machine unavailability is given by a probabilistic model; objectives of interest here are the sum of completion times and the makespan. Here, the non-preemptive case is essentially intractable. For the offline models we are interested in non-preemptive schedules where we consider the makespan objective; we present approximation algorithms which are complemented by suitable inapproximability results. Here, the preemptive model is polynomial-time solvable for large classes of settings.

1 Introduction

One major application that would not be possible without large communication networks is distributed computing where machines work in parallel: in large-scale projects such as SETI@home [4], the Internet Mersenne Prime Project [63] or medical applications [5,6], processing power is donated by volunteers who receive subjobs transmitted over the internet, calculate and return the results. In this perspective, individual units constitute a dynamic distributed computational

⋆ Research supported in part by a grant "DAAD Doktorandenstipendium" of the German Academic Exchange Service. Part of this work was done while visiting the LIG, Grenoble University.

⋆⋆ Supported in part by EU research project AEOLUS, "Algorithmic Principles for Building Efficient Overlay Computers", EU contract number 015964.

⋆⋆⋆ Supported in part by DFG priority program 1126, "Algorithmics of Large and Complex Networks".

† Part of this work was supported by the "CoreGRID" Network of Excellence supported by the EU.

J. Lerner, D. Wagner, and K.A. Zweig (Eds.): Algorithmics, LNCS 5515, pp. 50–64, 2009.
© Springer-Verlag Berlin Heidelberg 2009

environment which is often termed as *overlay computing platform* or simply *grid*. Clearly it is not appropriate to make constraining demands in terms of processing power from those participants. Furthermore, in a dynamic network, reachability of the participants cannot be guaranteed. This practical demand leads to a host of new problems in parallel machine scheduling.

Hence, due to the dynamic nature of the computing platform, a crucial issue is to take intervals of machine unavailability into account. This restriction in availability of computational resources is also realistic due to periods of machine maintenance or because jobs of highest priority are present. In either case we obtain models capturing realistic industrial settings and scheduling problems in new parallel computing platforms.

This chapter is organized as follows. In Sect. 2 we discuss approaches and results for online models; here we first discuss existing results before presenting our contributions. Likewise, in Sect. 3 we give a survey on the literature concerning offline models and briefly sketch our new algorithmic techniques put into effect. Finally we conclude in Sect. 4 with open research problems.

2 Online Models

In this section, we give an overview of results for various online models, i.e. not all machine failures are known in advance; the jobs are given offline, i.e. all information concerning the jobs os known a priori. A common tool to obtain results is to think of an *adversary* who dictates external events to force the algorithm to obtain results as bad as possible. We shall start with models and results that can be found in the literature and outline more recent results in Sect. 2.2. Note that we do not consider here settings in which the job durations are random but rather refer the reader to [12,45,51] for recent results and surveys on this topic.

2.1 Preliminaries and Known Results

In the following, we let n the number of jobs, and we denote with m the number of machines. In most online settings, we assume that m is known a priori. The processing times of the jobs are denoted $p_1, \ldots, p_n$. A schedule is an assignment of jobs to machines such that at no time, a job is executed on more than one machine and no machine executes more than one job. In a *preemptive* schedule, processing of a job can be interrupted and resumed at a later time, even on a different machine, while in a *non-preemptive* schedule, this is not permitted. Different objectives can be considered, the most common are the latest completion time C_j of any job j, called *makespan*, and the *average completion time* (or, equivalently, the sum of completion times $\sum C_j$). If for each job, a deadline or due date d_j is given by which it should be completed, we can also define the *tardiness* of a job as the amount of time by which it misses its deadline.

Negative results and restricted models. It is perhaps not very surprising that there are no positive results that concern non-preemptive scheduling

with online failures: without further restriction, the adversary can prevent any algorithm even from scheduling jobs at all.

Example 1. Consider two identical machines M_1, M_2, two jobs with processing time $p_1 = 1$, $p_2 = 2$, and an arbitrary algorithm A. At time $t = 0$, both machines are available. Let t_2 be the time at which A starts the second job J_2, w.l.o.g. on machine M_1. The adversary now shuts off M_1 at time $t_2 + 1$ and M_2 at time $t_2 + 2$. It is obvious that J_2 cannot terminate under A, and an offline optimal algorithm could schedule J_1 on M_1 and J_2 on M_2 at time t_2.

In total, this means that for both objectives $C_{\max}$ and $\sum C_j$ under consideration, no bounded competitive ratio can be achieved. Even if we allow preemptions, there are two major settings to distinguish here: in the "classical" preemptive setting, jobs can continue from the point they were interrupted, without any loss or penalty due to interruption. A stronger, but more realistic, setting requires that an interrupted job is *restarted* from the beginning. It is easy to see that many examples for the non-preemptive setting will carry over to the model with restarts. Therefore, for online problems, we concentrate on the truly preemptive setting.

As the following example from [3] shows, shutting off all machines is very powerful, even when the algorithm is permitted preemptions:

Example 2. Consider an instance with m identical machines $M_1, \ldots, M_m$, and m jobs $J_1, \ldots, J_m$ of unit processing time. Initially, only M_1 is available. Let t the first time when some job, w.l.o.g. J_1, is started by some algorithm A, and t' the time when J_1 first ceases to run, either by termination or by preemption. At time t', machines $M_2, \ldots, M_m$ become available, and all machines will become unavailable at time $t' + 1 - (t' - t)/m$. Clearly, A cannot complete all jobs by time $t' + 1 - (t' - t)/m$. However, an offline optimal algorithm could share the interval $[t, t')$ on machine M_1 between all jobs, thus being able to complete all jobs.

Note that in this example, the algorithms will terminate if we force an unlimited amount of processing time to be available, but their competetive ratio will be arbitrarily bad if there are large intervals of time where no machine is available. Hence, most settings will require that at all times, at least one machine is available. This guarantees that even simple one-machine algorithms are $\mathcal{O}(m)$-competitive.

A different restriction limits the adversary's power to change machine availability suddenly: such changes must be announced beforehand, either in the way that the time of the next change of availability is known (lookahead) or that all changes for a fixed window into the future are known (rolling horizon). These two settings are in many cases closely related: consider a look-ahead algorithm in a rolling horizon setting. Clearly, the algorithm can know the time of the next event if it is within the horizon. To make sure that there is always an event within the current planning horizon, we insert dummy events, for example, a machine leaves and immediately recovers, at regular intervals. (This is not a restriction in settings that do not penalize preemption.)

On the other hand, consider a rolling-horizon algorithm in a look-ahead setting. Depending on the circumstances, it is often possible to change the length of the horizon on the fly, as long as its length is lower-bounded to guarantee termination. Now, we can present the time interval from the present time to the next event as a horizon to the algorithm and run the algorithm. After this, we are told the next event and again present a fixed horizon, of possibly different length, to the algorithm and so on.

One final restriction that can be made concerns the selection of machines that can be disabled or enabled. Of particular interest in the online setting are three patterns:

1. In a *zig-zag* pattern, the number of machines at time t, $m(t)$, is always m or $m-1$ for some suitable constant $m \in \mathbb{N}$.
2. In an *increasing* pattern, for all two points in time t, t' with $t < t'$, we have $m(t) \leq m(t')$, i.e. the number of available machines never decreases.
3. In an *increasing zig-zag* pattern, for $t < t'$, we have $m(t) - 1 \leq m(t')$. Intuitively, if we have $m(t)$ machines now, we will always have at least $m(t) - 1$ machines in the future.

Known positive results. Not many results are known for the online setting, mostly considering the makespan and some only for the single-machine case. Kasap et al. [33] have shown that for the single-machine problem, the "Longest First" heuristic is optimal in expectation if the uptime distribution (i.e. the time between failures) is convex. Adiri et al. [1] have given asymptotically optimal algorithms to minimize the expected sum of completion times if the breakdowns are exponentially distributed. These settings are not preemptive in our sense, since an interrupted job must be started anew. (In the single-machine case, results with resumable jobs are generally not harder than their corresponding counterparts without unavailability constraints.) Li & Cao [46] have considered a setting with two different kinds of breakdowns: one kind which allows resumption, and one kind which forces restart, possibly with entirely different job characteristics. They give results for weighted completion time if all random variables are exponentially distributed.

Less is known about the multi-machine case: Lee & Yu [44] consider the case in which all machines will fail at the same time, but the length of the disruption is unknown. They present pseudopolynomial algorithms to minimize the weighted sum of completion times for both the resumable and restart settings. In the case that there is a lookahead to the next event, Albers & Schmidt [3] have shown that the optimal makespan can be achieved by a simple "Longest Remaining First" heuristic. In [60], this result is extended to related machines and non-zero release times by using the corresponding extension "Longest Remaining First on Fastest Machine". One can remove the need for lookahead and still obtain a makespan of $\mathrm{OPT} + \epsilon$ for any fixed $\epsilon > 0$ for identical machines. This, too, can be extended to related machines [59], however, the number of preemptions will then depend polynomially on the ratio of maximal to minimal machine speed.

Earlier results of Sanlaville [57] are concerned with the maximum tardiness objective. (This is a more general case than the makespan, since we may add

due dates of 0 for all jobs.) If additionally all the release times are 0, then the *shortest laxity first* heuristic yields optimal results for zig-zag availability. In this heuristic, the priority of a job is higher if its remaining processing time is "high" and its deadline is "soon" (i.e. we consider the values $d_i - \mathrm{rem}_i(t)$ for a time t and select the jobs of smallest difference). If the release times are not all equal, this heuristic still yields a schedule of value at most $\mathrm{OPT} + \max p_j$. Some results are known if there are precedence constraints on the jobs: if the constraints form chains, the optimal makespan is found by a "Longest Path First" heuristic, as shown by Liu & Sanlaville [48]. This algorithm is still optimal for outforests (each job has at most one immediate predecessor) and decreasing zig-zag patterns (and vice versa for inforests and increasing zig-zag). Note that such heuristics are suitable for online algorithms since all jobs are known from the beginning, and only the machine availability is online.

These results only hold for the case that preemption and migration are permitted, i.e. we may interrupt execution of jobs at any time and resume it – even on another machine – with no penalty. In particular, this means that a machine failure will only negatively affect the future. Kalyanasundaram & Pruhs [32] have considered the restart model: whenever a machine fails, the progress it has made on the job it was executing is discarded completely. To overcome this problem, the same job is executed several times, possibly in parallel, in hope that one of the copies will eventually succeed. In particular, they show:

Theorem 1. *If each machine fails independently with probability f and machines do not recover, the competitive ratio of a deterministic online algorithm for the $C_{\max}$ objective is $\Omega(\log m/(\log\log m))$; for the $(1/n)\sum C_j$ objective, it is $\Omega(1)$. Both these bounds are tight in the sense that there exist algorithms of competitive ratios $\mathcal{O}(\log m/(\log\log m))$ resp. $\mathcal{O}(1)$.*

Note that here the failure probability is a parameter dependent on the time step; the modelization of uptime distribution from data of actual systems is an interesting topic of its own [53].

2.2 New Results

Some results were recently found by the authors in [62], mostly concerning the sum of completion times for identical machines. The main techniques used are rearrangment of preempted portions of jobs and a suitable reduction to a one-machine schedule by means of regarding the schedule as a queue. In particular, they show the following result:

Theorem 2. *The heuristic SRPT, which at any time executes those jobs which have the shortest remaining processing time, minimizes $\sum C_j$ for increasing zig-zag availability pattern.*

Corollary 1. *SRPT minimizes $\sum C_j$ on two machines, if one of them is always available.*

However, SRPT is not a good heuristic for arbitrary patterns:

Example 3. Consider an instance with m machines (where m is even) and $m + m/2$ jobs. $J_1, \ldots, J_m$ have length 1, $J_{m+1}, \ldots, J_{m+m/2}$ have length 2. During the interval $[0, 2)$, all machines $M_1, \ldots, M_m$ are available, and after that, only M_1 is available.

SRPT will schedule all short jobs immediately and hence will not complete all jobs by time 2. The optimal solution is to start all long jobs immediately and fill empty machines with short jobs. It is easily verified that the ratio of the sums of completion times is then $\Omega(m)$.

Indeed, we can show [61]:

Theorem 3. No *online algorithm can be* $(2 - \epsilon)$*-competitive for* $\sum C_j$*, for any* $\epsilon > 0$*, even if all job lengths are in* $\{1, 2\}$*.*

Examination of the bad instances we have given here reveals that they depend heavily on the ability to shut off many (usually: all but one) machines at the same time. This leads to a different performance measure of the algorithms where the probability of machine failure influences the quality.

If we consider the setting where each of the m machines fails independently of the others with some probability f that is independent of the machine, the following result can be shown [62]:

Theorem 4. *Given a scheduling problem on identical machines which fail with probability* f*, and an* α*-approximate algorithm for the offline setting without failures, there is an online algorithm with asymptotic competitive ratio* $\alpha/(1-f)$*.*

This holds for both makespan and weighted sum of completion times, even if the instance also has release times and precedence constraints.

Table 1 lists some suitable offline algorithms and the setting they pertain to.

Table 1. Some offline results that can be adapted to online settings. First column describes scheduling problems in 3-field notation, cf. [45,54].

Setting	Source		Approximation ratio
$\mathrm{P}\|pmtn\|\sum C_j$	McNaughton	[50]	1
$\mathrm{P}\|\|\sum w_j C_j$	Kawaguchi & Kyan	[34]	$(1+\sqrt{2})/2$
$\mathrm{P}\|r_j, pmtn\|\sum w_j C_j$	Afrati et al.	[2]	PTAS
$\mathrm{P}\|r_j, prec, pmtn\|\sum w_j C_j$	Hall et al.	[22]	3

3 Offline Models

Here we discuss recent offline models where the job characteristics as well as the periods of machine unavailability are given in advance. We give a brief survey on approximation algorithms complemented by inapproximability results; these can be summarized as in Tab. 2; we exclusively study the makespan objective for our problems here.

Table 2. Complexity results for offline problems in 3-field notation, cf. [45,54]

Problem	$m = 1$	$m = 2$	$m \geq 3$
$Pm\|nr\text{-}a\|C_{\max}$ arbitrary reservations	no polynomial time algorithm with constant approximation ratio unless $\mathsf{P} = \mathsf{NP}$		
$Pm\|nr\text{-}a\|C_{\max}$ at most one reservation per machine	NP-hard, FPTAS	no polynomial time algorithm with constant approximation ratio unless $\mathsf{P} = \mathsf{NP}$	
$Pm, 1up\|nr\text{-}a\|C_{\max}$ arbitrary reservations	P ($r = 0$)	strongly NP-hard, PTAS, no FPTAS unless $\mathsf{P} = \mathsf{NP}$	
$Pm, 1up\|nr\text{-}a\|C_{\max}$ at most one reservation per machine	P ($r = 0$)	NP-hard, FPTAS	strongly NP-hard, PTAS, no FPTAS unless $\mathsf{P} = \mathsf{NP}$
$P, 1up\|nr\text{-}a\|C_{\max}$ at most one reservation per machine	no polynomial time algorithm with approximation ratio better than 3/2 unless $\mathsf{P} = \mathsf{NP}$		
$P, 1up\|nr\text{-}a\|C_{\max}$ where a constant percentage of machines is assumed to be permanently available	no polynomial time algorithm with approximation ratio better than 3/2 unless $\mathsf{P} = \mathsf{NP}$, approximation algorithm with a PTAS-like running time and approximation ratio $3/2 + \epsilon$ for any $\epsilon \in (0, 1/2]$		

This section is organized as follows. In Subsect. 3.1, we discuss preliminaries and related work. In Subsect. 3.2, we present some inapproximability results which are complemented in Subsect. 3.3 by suitable approximation algorithms. Techniques used to obtain our new results [15,17] are mainly dual approximation via binary search over a suitable target makespan, a PTAS for Multiple Subset Sum, definition of configurations, linear grouping and rounding known from state-of-the-art approximation schemes for Bin Packing, and an interesting cyclic shifting argument which permits usage of a network flow model for assignment of large jobs.

3.1 Preliminaries and Known Results

Our problem can be formally defined as follows. Let $m \in \mathbb{N}^*$ denote the number of machines which is assumed to be either constant or part of the input; this results in different formulations, however. Besides the n jobs, an instance I consists of r reservations $R_1, \ldots, R_r$. For each $k \in \{1, \ldots, r\}$, $R_k = (i_k, s_k, t_k)$ indicates unavailability of machine M_{i_k} in the time interval $[s_k, t_k)$, where we have $s_k, t_k \in \mathbb{N}, i_k \in \{1, \ldots, m\}$ and $s_k < t_k$. We suppose that for reservations on the same machine there is no overlap; for two reservations $R_k, R_{k'}$ such that $i_k = i_{k'}$ holds, we have $[s_k, t_k) \cap [s_{k'}, t_{k'}) = \emptyset$. For each $i \in \{1, \ldots, m\}$ let $R'_i := \{R_k \in I | i_k = i\}$

denote the set of reservations for machine M_i. Our goal is to compute a non-preemptive schedule of the jobs such that no job is scheduled on a machine that is unavailable and on each machine at most one job runs at a given time; the objective is to minimize the makespan. (We note that in contrast to the online setting, preemptions allow very large classes of problems to be solved exactly in polynomial time with LP and network flow methods as in [11,14].)

Using the 3-field notation, we denote the corresponding model by $\mathrm{P}m|nr\text{-}a|C_{\max}$ if m is a constant and by $\mathrm{P}|nr\text{-}a|C_{\max}$ if m is considered part of the input. One might argue that denoting the presence of unavailability intervals in the job field of the classification scheme might be counter-intuitive; however, reservations can be viewed as jobs owned by other users and this notation is well-established in the existing literature [41,42,43,45].

Concerning previous results, Lee [41] and Lee et al. [43] studied identical parallel machines which may have different starting times; here, the LPT policy, where jobs are started in non-increasing order of their length, was analyzed. Lee [42] studied the case where at most one reservation per machine is permitted while one machine is continuously available and obtained suitable approximation ratios for low-complexity list scheduling algorithms. Liao et al. [47] presented an experimental study of an exact algorithm for $m = 2$ within the same scenario. Hwang et al. [26] studied the LPT policy for the case where at most one interval of unavailability per machine is permitted. They proved a tight bound of $1 + \lceil m/(m-\lambda)\rceil/2$ where at most $\lambda \in [m-1]$ machines are permitted to be unavailable simultaneously. The reader can find in [45], Chapt. 22, problem definitions and a more comprehensive survey about previous results; there, mostly NP-completeness proofs, fast greedy algorithms and dynamic programming formulations for various objectives can be found. In [58], Scharbrodt et al. present approximation schemes and inapproximability results for a setting where the reservations are seen as fixed jobs which consequently also contribute to the makespan. Note that the objective is quite different from the objective discussed in this section; the objective in [58] models makespan minimization from the perspective of the *system administrator* while the objective presented here models makespan minimization from the viewpoint of an *individual user* submitting his jobs to a system subject to preallocation of jobs.

For the model of interest, the sum of completion time objective has also been considered. In [52], a setting where at most one reservation per machine is permitted was studied; Mellouli et al. experimentally evaluated several exact algorithms. Furthermore, Kacem et al. [31] studied the corresponding single machine problem, also evaluating several exact algorithms; he also obtained an approximation algorithm with ratio 2 and running time $\mathcal{O}(n^2)$ for the same problem in [30]; a similar model was studied by Sadfi et al. in [55].

Until recently, makespan minimization for the model under consideration has not been approached with approximation schemes, not even for the special cases which have already been studied [26,42,47]. However, Kacem [29] studied the single machine problem with one reservation and tails; this is a model where jobs occupy the machine for a certain time, the tail, after their completion.

He obtained a 3/2-approximation algorithm and an FPTAS based on dynamic programming.

3.2 Inapproximability Results

In this section we present hardness and inapproximability results for the models under consideration. First it is clear that $\mathrm{P}m|nr\text{-}a|C_{\max}$ as well as $\mathrm{P}|nr\text{-}a|C_{\max}$ are NP-hard since they are generalizations of the corresponding problems $\mathrm{P}m||C_{\max}$ and $\mathrm{P}||C_{\max}$ without reservations. In addition, the problems under consideration are also hard to approximate, as shown in [15,16,17]; the respective constructions for the proofs use reductions from NP-complete partition problems [21] in combination with gap creation arguments.

Theorem 5. *Both problems* $\mathrm{P}m|nr\text{-}a|C_{\max}$ *and* $\mathrm{P}|nr\text{-}a|C_{\max}$ *do not admit a polynomial time algorithm with a constant approximation ratio unless* P = NP *holds.*

Similar to the parallel setting studied in [18], for both of these problems the inapproximability is due to the existence of intervals in which no machine is available. Consequently no work can be carried out in these intervals; since the setting is non-preemptive, this causes a large undesired delay for some jobs. In total, it is necessary to find suitable restrictions which permit constant approximation ratios. One straightforward way to do this is to require at least one machine to be permanently available. The resulting problems are denoted by $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ and $\mathrm{P}, 1up|nr\text{-}a|C_{\max}$, where $1up$ means that at least one machine is always available. These restricted problem formulations then are polynomially solvable for $m = 1$ and remain NP-hard for $m \geq 2$, which can be seen by defining a reservation for each machine except the first one and following the lines of the proof of Lemma 1 in [17]. As we discuss in Subsect. 3.3, $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ admits a PTAS [17]. On the other hand $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ does not admit an FPTAS unless P = NP; this result follows from the fact that $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ is strongly NP-hard [17] via a reduction from 3-Partition and the subsequent standard arguments for "well-behaved" objective functions as established in [20].

Theorem 6. *The problem* $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ *does not admit an FPTAS for* $m \geq 2$ *unless* P = NP *holds.*

Furthermore it is a natural question whether $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ becomes easier if the number of reservations per machine is restricted to one. Surprisingly, this is not the case, which can be shown by adaptation of a construction from [7]. The following result implies that $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ with at most one reservation per machine for $m \geq 3$ is strongly NP-hard as well.

Theorem 7. *The problem* $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ *does not admit an FPTAS, even if there is at most one reservation per machine, for* $m \geq 3$ *unless* P = NP *holds.*

However, $\mathrm{P}2, 1up|nr\text{-}a|C_{\max}$, if there is at most one reservation per machine, is easier to approximate in the sense that it admits an FPTAS, as we discuss in

Subsect. 3.3. Finally, the one-machine problem $1|nr\text{-}a|C_{\max}$ does not admit a polynomial time approximation algorithm if more than one reservation is permitted, but admits an FPTAS for only one reservation [17].

For the problem $\mathrm{P}, 1up|nr\text{-}a|C_{\max}$ we obtain a much stronger inapproximability result than the strong NP-hardness of $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$. Based on a construction from [58] it can be shown that for any $\epsilon \in (0, 1/2]$ the problem $\mathrm{P}, 1up|nr\text{-}a|C_{\max}$ does not admit a polynomial-time approximation algorithm with approximation ratio $3/2 - \epsilon$ unless $\mathsf{P} = \mathsf{NP}$ holds; the proof is based on a reduction from Numerical Matching with Target Sums [21]. Surprisingly, this lower bound is also valid if a constant percentage of the machines is assumed to be permanently available [15], which complements the sophisticated algorithm mentioned in Subsect. 3.3.

Theorem 8. *The problem* $\mathrm{P}, 1up|nr\text{-}a|C_{\max}$ *does not admit an approximation algorithm with ratio* $3/2 - \epsilon$*, for any* $\epsilon \in (0, 1/2]$*, not even if the percentage of permanently available machines is constant, unless* $\mathsf{P} = \mathsf{NP}$ *holds.*

From a broader perspective, the hardness is due to the approximation of $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ with a constant ratio, even if there is at most one reservation per machine, being at least as hard as approximation of Bin Packing with an additive error; however whether the latter is possible is an open problem, discussed in [23], Chapt. 2, page 67.

Theorem 9. *If there is a polynomial time algorithm for* $\mathrm{P}, 1up|nr\text{-}a|C_{\max}$ *with approximation ratio* $c \in \mathbb{N} \setminus \{1\}$*, then there is a polynomial time algorithm for Bin Packing with additive error* $2(c-1)$*.*

3.3 Approximation Algorithms

To obtain approximation algorithms for the problems under consideration, different techniques from algorithm design are put into effect. The most basic approach used in [17] is based on dual approximation [24] and multiple subset sum problems. The latter ones are special cases of knapsack problems, which belong to the oldest problems in combinatorial optimization and theoretical computer science. Hence, we benefit from the fact that they are relatively well understood. For the classical problem (KP) with one knapsack, besides the result by Ibarra & Kim [27], Lawler presented a sophisticated FPTAS [40] which was later improved by Kellerer & Pferschy [37]; see also the textbooks by Martello & Toth [49] and Kellerer et al. [38] for surveys. The case where the item profits equal their weights is called the subset sum problem and denoted as SSP. The problem with *multiple* knapsacks (MKP) is a natural generalization of KP; the case with multiple knapsacks where the item profits equal their weights is called the *multiple* subset sum problem (MSSP). Various special cases and extensions of these problems have been studied [7,8,9,10,13,28,35,36], finally yielding PTASes and FPTASes for the cases upon which our approach is partially based [8,10,28,36].

As discussed in detail in [17], we obtain a PTAS for $\mathrm{P}, 1up|nr\text{-}a|C_{\max}$ by using binary search over the target makespan and using a PTAS for MSSP

where the capacities of the knapsacks are permitted to be different [8]; this complements the inapproximability result in Subsect. 3.2 by which the PTAS is, in a certain sense, a best possible algorithm. Furthermore, instead of a PTAS for MSSP we can use a fast greedy algorithm. This results in an algorithm for $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ with an approximation ratio of $1 + m/2$. In [26], the authors studied the case where at most one reservation per machine is permitted while λ machines are permanently available. They proved that for this setting LPT yields a tight bound of $1 + \lceil 1/(1 - \lambda/m) \rceil /2$. For $\lambda = m - 1$, this also yields $1 + m/2$. In total, we obtain the same approximation ratio for a much more general problem formulation, which at comes the cost of a larger runtime bound however.

Theorem 10. *The problem* $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ *admits a PTAS and a fast greedy algorithm with approximation ratio* $1 + m/2$.

If we study $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ but permit at most one reservation per machine, the situation is different; this change of complexity is illustrated in Tab. 2. The algorithm for $\mathrm{P}2, 1up|nr\text{-}a|C_{\max}$ mentioned in the next theorem is based on a dynamic programming formulation which combinatorializes the loads of the two machines in suitable intervals; in combination with a 2-approximation algorithm and scaling and rounding the values similar to [40] this dynamic programming formulation can be transformed into an FPTAS with a standard approach. Again this approximation scheme is to be compared with the complementing hardness result from Subsect. 3.2.

Theorem 11. *The problem* $\mathrm{P}2, 1up|nr\text{-}a|C_{\max}$ *with one reservation admits an FPTAS.*

For $\mathrm{P}, 1up|nr\text{-}a|C_{\max}$ we have obtained an inapproximability result in Subsect. 3.2. However, in [15], we have obtained an approximation algorithm with ratio $3/2 + \epsilon$ for the case where the percentage of available machines is constant.

Theorem 12. *The problem* $\mathrm{P}, 1up|nr\text{-}a|C_{\max}$*, provided that the percentage of permanently available machines is constant, admits a polynomial time algorithm with approximation ratio* $3/2 + \epsilon$ *for any* $\epsilon \in (0, 1/2]$.

The running time of the respective algorithm is PTAS-like, i.e. exponential in $1/\epsilon$; however, various interesting algorithmic techniques are used. These include dual approximation by guessing the makespan [24], linear grouping and rounding known from Bin Packing [19], enumeration of configurations, grouping and embedding known from Strip Packing [39] in combination with a novel method of assignment of jobs via network flow models and again a PTAS for MSSP as in [17].

4 Conclusion

We have given a survey on recent advances in scheduling with machine unavailability constraints; this is an intriguing field where online formulations are

as relevant as classical offline formulations. One interesting problem in the online setting is the rather large gap for the makespan problem: while only 2-inapproximability is known, we do not know of any algorithm with constant competitiveness. Permission of reservations in the classical offline problems changes the complexity in a surprising way. More precisely, $Pm||C_{\max}$ is NP-hard but permits an FPTAS [56], while $P, 1up|nr\text{-}a|C_{\max}$ is strongly NP-hard for $m \geq 2$; furthermore, $P||C_{\max}$ is strongly NP-hard but permits a PTAS [25], while $P, 1up|nr\text{-}a|C_{\max}$ is much harder to approximate. Concerning offline problems, it would be interesting to perform a sensitivity analysis with respect to the reservations for which a slight shift in time or increase and decrease in duration might be permitted. In total, we like to point out that here especially the algorithms for online problems tend to be suitable for implementation. This is to be compared with those for the offline problems which are rather suitable for long-term planning, where calculation of the schedule is not a time-critical issue.

Acknowledgements. The authors thank Érik Saule and Derrick Kondo for fruitful discussions and the referees for many helpful comments.

References

1. Adiri, I., Bruno, J.L., Frostig, E., Kan, A.H.G.R.: Single machine flow-time scheduling with a single breakdown. Acta Inf. 26(7), 679–696 (1989)
2. Afrati, F.N., Bampis, E., Chekuri, C., Karger, D.R., Kenyon, C., Khanna, S., Milis, I., Queyranne, M., Skutella, M., Stein, C., Sviridenko, M.: Approximation schemes for minimizing average weighted completion time with release dates. In: FOCS, pp. 32–44 (1999)
3. Albers, S., Schmidt, G.: Scheduling with unexpected machine breakdowns. Disc. App. Math. 110(2-3), 85–99 (2001)
4. Anderson, D.P., et al.: Seti@home, `http://setiathome.berkeley.edu/`
5. Baker, D., et al.: Rosetta@home protein folding, design and docking, `http://boinc.bakerlab.org/rosetta/`
6. Berman, F., et al.: World community grid, `http://www.worldcommunitygrid.org/`
7. Caprara, A., Kellerer, H., Pferschy, U.: The multiple subset sum problem. Technical report, Technische Universität Graz (1998)
8. Caprara, A., Kellerer, H., Pferschy, U.: A PTAS for the multiple subset sum problem with different knapsack capacities. Inf. Process. Lett. 73(3-4), 111–118 (2000)
9. Caprara, A., Kellerer, H., Pferschy, U.: A 3/4-approximation algorithm for multiple subset sum. J. Heuristics 9(2), 99–111 (2003)
10. Chekuri, C., Khanna, S.: A polynomial time approximation scheme for the multiple knapsack problem. SIAM J. Comput. 35(3), 713–728 (2005)
11. Cochand, M., de Werra, D., Slowinski, R.: Preemptive scheduling with staircase and piecewise linear resource availability. Methods and Models of Op. Res. 33, 297–313 (1989)
12. Crutchfield, C.Y., Dzunic, Z., Fineman, J.T., Karger, D.R., Scott, J.H.: Improved approximations for multiprecessor scheduling under uncertainty. In: Proceedings of SPAA (2008) (to appear)
13. Dawande, M., Kalagnanam, J., Keskinocak, P., Salman, F.S., Ravi, R.: Approximation algorithms for the multiple knapsack problem with assignment restrictions. J. Comb. Optim. 4(2), 171–186 (2000)

14. de Werra, D.: On the two-phase method for preemptive scheduling. Eur. J. Operational Res. 37, 227–235 (1988)
15. Diedrich, F., Jansen, K.: Improved approximation algorithms for scheduling with fixed jobs. In: Proc. 20th ACM-SIAM Symposium on Discrete Algorithms (2009) (to appear)
16. Diedrich, F., Jansen, K., Pascual, F., Trystram, D.: Approximation algorithms for scheduling with reservations. (unpublished Manuscript)
17. Diedrich, F., Jansen, K., Pascual, F., Trystram, D.: Approximation algorithms for scheduling with reservations. In: Aluru, S., Parashar, M., Badrinath, R., Prasanna, V.K. (eds.) HiPC 2007. LNCS, vol. 4873, pp. 297–307. Springer, Heidelberg (2007)
18. Eyraud-Dubois, L., Mounié, G., Trystram, D.: Analysis of scheduling algorithms with reservations. In: IPDPS, pp. 1–8. IEEE, Los Alamitos (2007)
19. Fernandez de la Vega, W., Lueker, G.S.: Bin packing can be solved within $1 + \epsilon$ in linear time. Combinatorica 1(4), 349–355 (1981)
20. Garey, M.R., Johnson, D.S.: "strong" NP-completeness results: Motivation, examples, and implications. J. ACM 25(3), 499–508 (1978)
21. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, New York (1979)
22. Hall, L.A., Shmoys, D.B., Wein, J.: Scheduling to minimize average completion time: Off-line and on-line algorithms. In: SODA, pp. 142–151 (1996)
23. Hochbaum, D. (ed.): Approximation Algorithms for NP-hard Problems. PWS Publishing Company (1996)
24. Hochbaum, D.S., Shmoys, D.B.: Using dual approximation algorithms for scheduling problems: theoretical and practical results. J. ACM 34(1), 144–162 (1987)
25. Hochbaum, D.S., Shmoys, D.B.: A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. SIAM J. Comput. 17(3), 539–551 (1988)
26. Hwang, H.-C., Lee, K., Chang, S.Y.: The effect of machine availability on the worst-case performance of LPT. Disc. App. Math. 148(1), 49–61 (2005)
27. Ibarra, O.H., Kim, C.E.: Fast approximation algorithms for the knapsack and sum of subset problems. J. ACM 22(4), 463–468 (1975)
28. Jansen, K.: Parameterized approximation scheme for the multiple knapsack problem. In: Proc. 20th ACM-SIAM Symposium on Discrete Algorithms (2009) (to appear)
29. Kacem, I.: Approximation algorithms for the makespan minimization with positive tails on a single machine with a fixed non-availability interval. J. Comb. Optim. (2007)
30. Kacem, I.: Approximation algorithm for the weighted flow-time minimization on a single machine with a fixed non-availability interval. Computers & Industrial Engineering 54(3), 401–410 (2008)
31. Kacem, I., Chu, C., Souissi, A.: Single-machine scheduling with an availability constraint to minimize the weighted sum of the completion times. Computers & OR 35(3), 827–844 (2008)
32. Kalyanasundaram, B., Pruhs, K.: Fault-tolerant scheduling. SIAM Journal on Computation 34(3), 697–719 (2005)
33. Kasap, N., Aytug, H., Paul, A.: Minimizing makespan on a single machine subject to random breakdowns. Oper. Res. Lett. 34(1), 29–36 (2006)
34. Kawaguchi, T., Kyan, S.: Worst case bound of an LRF schedule for the mean weighted flow-time problem. SIAM Journal on Computation 15(4), 1119–1129 (1986)

35. Kellerer, H.: A polynomial time approximation scheme for the multiple knapsack problem. In: Hochbaum, D.S., Jansen, K., Rolim, J.D.P., Sinclair, A. (eds.) RANDOM 1999 and APPROX 1999. LNCS, vol. 1671, pp. 51–62. Springer, Heidelberg (1999)
36. Kellerer, H., Mansini, R., Pferschy, U., Speranza, M.G.: An efficient fully polynomial approximation scheme for the subset-sum problem. J. Comput. Syst. Sci. 66(2), 349–370 (2003)
37. Kellerer, H., Pferschy, U.: A new fully polynomial time approximation scheme for the knapsack problem. J. Comb. Optim. 3(1), 59–71 (1999)
38. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, Heidelberg (2004)
39. Kenyon, C., Rémila, E.: A near-optimal solution to a two dimensional cutting stock problem. Math. Oper. Res. 25, 645–656 (2000)
40. Lawler, E.L.: Fast approximation algorithms for knapsack problems. Math. Oper. Res. 4(4), 339–356 (1979)
41. Lee, C.-Y.: Parallel machines scheduling with non-simultaneous machine available time. Disc. App. Math. 30, 53–61 (1991)
42. Lee, C.-Y.: Machine scheduling with an availability constraint. J. Global Optimization, Special Issue on Optimization of Scheduling Applications 9, 363–384 (1996)
43. Lee, C.-Y., He, Y., Tang, G.: A note on parallel machine scheduling with non-simultaneous machine available time. Disc. App. Math. 100(1-2), 133–135 (2000)
44. Lee, C.-Y., Yu, G.: Parallel-machine scheduling under potential disruption. Opt. Lett. 2(1), 27–37 (2008)
45. Leung, J.Y.-T. (ed.): Handbook of Scheduling. Chapman & Hall, Boca Raton (2004)
46. Li, W., Cao, J.: Stochastic scheduling on a single machine subject to multiple breakdowns according to different probabilities. Oper. Res. Lett. 18(2), 81–91 (1995)
47. Liao, C.-J., Shyur, D.-L., Lin, C.-H.: Makespan minimization for two parallel machines with an availability constraint. Eur. J. Operational Res. 160, 445–456 (2003)
48. Liu, Z., Sanlaville, E.: Preemptive scheduling with variable profile, precedence constraints and due dates. Disc. App. Math. 58(3), 253–280 (1995)
49. Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. Wiley, Chichester (1990)
50. McNaughton, R.: Scheudling with deadlines and loss functions. Mgt. Science 6, 1–12 (1959)
51. Megow, N., Vredeveld, T.: Approximation in preemptive stochastic online scheduling. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 516–527. Springer, Heidelberg (2006)
52. Mellouli, R., Sadfi, C., Chu, C., Kacem, I.: Identical parallel-machine scheduling under availability constraints to minimize the sum of completion times. Eur. J. Operational Res. (2008)
53. Nurmi, D., Brevik, J., Wolski, R.: Modeling machine availability in enterprise and wide-area distributed computing environments. In: Cunha, J.C., Medeiros, P.D. (eds.) Euro-Par 2005. LNCS, vol. 3648, pp. 432–441. Springer, Heidelberg (2005)
54. Pinedo, M.: Scheduling: Theory, Algorithms and Systems. Prentice Hall, Englewood Cliffs (1995)
55. Sadfi, C., Penz, B., Rapine, C., Błażewicz, J., Formanowicz, P.: An improved approximation algorithm for the single machine total completion time scheduling problem with availability constraints. Eur. J. Operational Res. 161(1), 3–10 (2005)
56. Sahni, S.: Algorithms for scheduling independent tasks. J. ACM 23(1), 116–127 (1976)

57. Sanlaville, E.: Nearly on line scheduling of preemptive independent tasks. Disc. App. Math. 57(2-3), 229–241 (1995)
58. Scharbrodt, M., Steger, A., Weisser, H.: Approximability of scheduling with fixed jobs. J. Scheduling 2, 267–284 (1999)
59. Schwarz, U.M.: Scheduling related machines with failures (unpublished manuscript)
60. Schwarz, U.M.: Online scheduling on semi-related machines. Information Processing Letters 108(1), 38–40 (2008)
61. Schwarz, U.M., Diedrich, F.: Scheduling algorithms for random machine profiles (unpublished manuscript)
62. Schwarz, U.M., Diedrich, F.: A framework for scheduling with online availability. In: Kermarrec, A.-M., Bougé, L., Priol, T. (eds.) Euro-Par 2007. LNCS, vol. 4641, pp. 205–213. Springer, Heidelberg (2007)
63. Woltman, G., et al.: The great internet mersenne prime search, `http://www.mersenne.org/`