

# 9

## Parallel Scheduling for Early Completion

---

9.1	Introduction .....	9-175
9.2	Some Basic Algorithms and Relationships .....	9-176
	List Scheduling • LPT and SPT • HLF and Critical Path • Relationships	
9.3	Preemptive Scheduling .....	9-177
9.4	Nonpreemptive Scheduling .....	9-178
	Enumerative Algorithms • Approximation Algorithms	
9.5	Scheduling with Precedence Constraints .....	9-180
	For UET Jobs • For Jobs of General Lengths	

Bo Chen

*University of Warwick*

### 9.1 Introduction

---

This chapter assumes knowledge of some basic scheduling terminology provided in Part I of this book. We are concerned in this chapter with classical deterministic parallel scheduling with the objective of minimizing the makespan. Both preemptive and nonpreemptive schedules are considered. There can be precedence relationships between jobs. Excluded from consideration in this chapter are parallel scheduling models of other typical job characteristics, on-line and randomized algorithms, all of which are covered in separate chapters of this book. We will address the following aspects of the scheduling problems:

- Polynomial solvability and approximability
- Enumerative algorithms
- Polynomial-time approximations

Although we also consider scheduling jobs of different release dates, we will mainly concentrate on models of equal job release dates. The reason is as follows. If all jobs are not released at the same time, then the scheduling problems of minimizing the makespan can be viewed as special cases of the corresponding problems of minimizing maximum lateness, which is dealt with in a separate chapter. To see this, note that due to schedule symmetry on the time axis, there is an equivalence relationship (with respect to the objective value) between scheduling of the former objective and that of the latter: To a schedule  $S$  of instance  $I$  with makespan  $C_{\max}$  corresponds a schedule  $S'$  of instance  $I'$  with maximum lateness  $C_{\max}$ , where in instance  $I'$  all jobs are released at time 0, the due date and start time of job  $j$  are respectively  $d'_j = -r_j$  and  $S'_j = C_{\max} - C_j$ , where  $r_j$  and  $C_j$  are the release date and completion time of job  $j$  in schedule  $S$ , respectively.

Also note that minimization of makespan is a special case of minimization of the maximum lateness. The reader may find some further results in the separate chapter dealing with the latter objective function.

We will mainly use worst-case guarantee as algorithm performance indicator. As a result, a rich set of literature on local search algorithms, including meta-heuristics, are not considered here. This chapter is a more focused and updated study on the relevant scheduling models than those treated by Chen, Potts, and Woeginger [1].

The rest of the chapter is organized as follows. First, we present in the next section some important algorithms and basic relationships, which not only have significant influences on parallel scheduling, but on shop scheduling as well. Then in Sections 9.3, 9.4, and 9.5 we consider respectively preemptive, nonpreemptive scheduling of independent jobs and scheduling with precedence constraints.

## 9.2 Some Basic Algorithms and Relationships

---

### 9.2.1 List Scheduling

As a simplest combinatorial algorithm, *list scheduling* (LS) has been well known for almost half a century. In this algorithm, jobs are fed from a pre-specified list and, whenever a machine becomes idle, the first *available* job on the list is scheduled and removed from the list, where the availability of a job means that the job has been released and, if there are precedence constraints, all its predecessors have already been processed.

In its simplicity and the fact that any optimal schedule can be constructed by LS with an appropriately chosen list, LS is by far the most popular scheduling approach. Moreover, since LS requires no knowledge of unscheduled jobs as well as of all jobs currently being processed, it is very powerful in on-line scheduling and especially so in on-line nonclairvoyance scheduling, in which it remains a dominant heuristic. (See the chapter on on-line scheduling for more details.)

### 9.2.2 LPT and SPT

If the job list in LS can be sorted in order of nonincreasing processing times before they are scheduled, then the resulting algorithm is known as *largest processing time first* (LPT). The minor extra computing in sorting the jobs significantly improves the worst-case performance of the algorithm. Since LPT was first proposed by Graham [2], it has become the touchstone for the design of efficient algorithms, both off-line and on-line. Moreover, LPT has also found applications in combinatorial optimization.

If preemption is allowed, then LPT becomes LRPT—*largest remaining processing time first*: At all times, a job with largest remaining processing time among all available jobs is processed, preempt a job whenever necessary.

On the other hand, if the prespecified job list in LS is sorted in order of nondecreasing processing times, then the resulting algorithm is known as *shortest processing time first* (SPT). In contrast to LPT, applications of SPT in scheduling are primarily for minimizing the average job completion time. For example, problems  $1 || \sum w_j C_j$  and  $1 |r_j, pmtn| \sum C_j$  are solved to optimality with two SPT-based LS algorithms [3, 4].

### 9.2.3 HLF and Critical Path

In the presence of precedence relations between jobs, Hu's level algorithm [5] for scheduling UET jobs, known as *highest level first* (HLF), is fundamental. This algorithm is based on the notion of *critical path* (CP) in network optimization and always schedules a job that heads the longest current chain of unscheduled jobs of arbitrary lengths.

### 9.2.4 Relationships

Let us consider two basic relationships in parallel scheduling to minimize the makespan, one between optimal preemptive schedules and optimal nonpreemptive schedules, while the other between schedules

of independent jobs of arbitrary processing times and schedules of UET jobs (i.e., jobs of unit processing time) of precedence constraints.

It is evident that preemption can reduce the makespan of a schedule, although such time saving might incur a cost for making the preemptions in practice. But how efficient can preemption be in reducing the makespan? Given a system of arbitrarily constrained jobs, Liu [6] conjectures that the makespan of any optimal nonpreemptive schedule is at most  $2 - 2/(m + 1)$  times that of an optimal preemptive schedule, where  $m$  is the number of parallel identical machines. This conjecture is proved by Hong and Leung [7] for two special cases: (i) of UET jobs and (ii) of *in-tree* precedence constraints, where every job has at most one immediate successor. In the case of (ii), Hong and Leung show that any CP schedule (as a nonpreemptive schedule not necessarily optimal) has already satisfied the conjectured upper bound. Note that the special case of (ii) can be extended to the case of independent jobs. As a result, any LPT schedule (as a CP schedule) satisfies the conjectured bound. This result is also obtained recently by Braun and Schmidt [8] in a different way. Furthermore, for both cases of (i) and (ii) above, the bound of  $2 - 2/(m + 1)$  holds true for any optimal (nonpreemptive) LS schedules, since Hong and Leung show that unforced idle time in any optimal nonpreemptive schedule is not beneficial in reducing the makespan.

Liu's conjecture is proved by Coffman and Garey [9] if  $m = 2$ , although it is still open in general if  $m \geq 3$ . Recently, Braun and Schmidt [8] investigate the benefit of *each* additional preemption. They show that, if the jobs are independent, the makespan of any schedule that is optimal among all schedules of at most  $k$  preemptions is at most  $2 - 2/[m/(k + 1) + 1]$  times the makespan of an optimal preemptive schedule. Note that when  $k = 0$  it reduces to a special case for Liu's conjecture.

On the other hand, complexity similarity between scheduling independent preemptable jobs of arbitrary processing times and scheduling non-preemptable UET jobs of precedence constraints has recently led an identification of a more formal relationship between the two classes of problems. In his *Preemption-Chaining Theorem*, Timkovsky [10] establishes that scheduling problems of the former class are reducible to the scheduling problems of the latter with the precedence constraints being chain-like. For more details, the reader is referred to a separate chapter dealing with reducibility of problem classes.

### 9.3 Preemptive Scheduling

Scheduling with preemption allowed greatly mitigates the difficulty of solving a scheduling problem, although each preemption may incur a cost in practice. For almost all scheduling problems, the preemptive version is no harder than its nonpreemptive counterpart, with only a few exceptions ( $J2 | n = 3 | C_{\max}$  [11, 12],  $P | p_j = p | \sum w_j U_j$  [13], and  $R || \sum C_j$  [14]). The following observation can help understand this phenomenon. If we consider a scheduling problem as a zero-one integer program with decision variables indicating the assignment of jobs (or their operations) to machines and time slots, then preemption allows a variable to take a value between zero and one, thereby indicating the proportion of the job or its operation to be assigned. From this perspective, preemption is a vital element for polynomial solvability for many scheduling problems.

Due to the reducibility mentioned at the end of the previous section to scheduling nonpreemptable UET jobs of precedence constraints, which we will consider in Section 9.5, we mention here only a few basic results of scheduling independent preemptable jobs of arbitrary processing times.

It is not surprising that the first paper in parallel machine scheduling deals with preemption. In his paper, McNaughton [15] uses a simple wrap-around procedure, which runs in  $O(n)$  time and generates at most  $m - 1$  preemptions, to solve problem  $P | pmtn | C_{\max}$  to optimality. The basic idea behind this is quite simple: First, calculate a lower bound on the value of an optimal schedule and then construct a schedule that matches the bound. Note that in this approach, the lower bound is used as a deadline for all jobs. Once a complete schedule is constructed in this way, it is automatically optimal. The lower bound used in this approach is the maximum of the average machine workload and the processing time of the longest job.

McNaughton's approach has been successfully applied to other preemptive scheduling problems. Following a generalization of McNaughton's lower bound to the case of uniform machines by Liu and Yang [16] after the machine speeds are taken into account, Horvath, Lam, and Sethi [17] suggest the *largest remaining*

*processing time on fastest machines* (LRPT-FM) rule. At each time point, the job with the most remaining processing time is scheduled on the fastest available machine; thereby they solve problem  $Q | pmtn | C_{\max}$  in  $O(mn^2)$  time. As a variant of LPT for preemptive scheduling, the LRPT-FM rule is also an application of the notion of critical path as does HLF, which we mentioned in the previous section. A less intuitive and more case-based algorithm is developed by Gonzalez and Sahni [18], which significantly reduces both the running time to  $O(n + m \log m)$  and the number of preemptions.

Further down the road when the machines are unrelated, more expensive approaches are proposed. At the cost of solving a linear program, in which a decision variable represents the total time spent by each job on each machine, Lawler and Labetoulle [19] provide a general lower bound on the optimal makespan for problem  $R | pmtn | C_{\max}$ , and then solve it to optimality as an instance of problem  $O | pmtn | C_{\max}$ .

## 9.4 Nonpreemptive Scheduling

One of the fundamental problems in scheduling theory is to schedule independent nonpreemptable jobs onto parallel machines to minimize the makespan. This general problem is difficult to solve to optimality, as two most basic models  $P2 || C_{\max}$  and  $P || C_{\max}$  are already NP-hard and strongly NP-hard, respectively (Garey and Johnson [20, 21]). In fact, it is shown by Lenstra, Shmoys, and Tardos [22] that, unless  $P = NP$ , it is not at all easier even to approximate problem  $R || C_{\max}$  within a factor better than  $3/2$ .

### 9.4.1 Enumerative Algorithms

It is immediately clear that problem  $P || C_{\max}$  allows for a very simple dynamic algorithm with all possible job assignments taken into account. A far more efficient branch and bound algorithm is proposed by Dell'Amico and Martello [23]. If the jobs are in LPT order, then a trivial lower bound is given by  $\max\{\sum_{j=1}^n p_j/m, p_1, p_m + p_{m+1}\}$ . Based on bin-packing arguments (see Section 9.4.2 for more details), they have developed a procedure to improve iteratively this lower bound, while their branching rule assigns a longest unscheduled job to a machine. Computational results show that the algorithm can solve instances with up to 10,000 jobs.

For the more general problem  $R || C_{\max}$ , branch and bound algorithms are proposed by Van de Velde [24] and by Martello, Soumis, and Toth [25]. The former is based on surrogate relaxation and duality and the latter on Lagrangian relaxations and additive techniques. Computational results demonstrate the superiority of the algorithm of Martello, Soumis, and Toth over Van de Velde's algorithm. Recently, Mokotoff and Chrétienne [26] introduce a cutting plane technique, which is based on a newly identified valid inequality of the convex hull of the problem instance. Their algorithm includes a preprocessing phase to compute an upper bound with LS heuristic and a lower bound obtained from the preemptive relaxation. Computational results show that the algorithm gives an optimal solution for almost all tested cases within the fixed time and memory limits.

### 9.4.2 Approximation Algorithms

#### 9.4.2.1 LS-Based Techniques

In the first paper on the worst-case analysis of scheduling algorithms, Graham [27] studies algorithm LS. He shows that for problem  $P || C_{\max}$ , LS is  $(2 - 1/m)$ -approximate by observing that the makespan of any LS schedule is at most  $2 - 1/m$  times the lower bound of optimum—the maximum of the length of a largest job and average machine workload. Note that this simple lower bound has been used to construct an optimal preemptive schedule (see the previous section for details).

The worst case arises in the above estimate on LS performance is when the last job has the longest length. This gives rise to an improved algorithm LPT [2] by first sorting the jobs in a better order in an attempt to avoid the worst case. LPT has improved the worst-case performance ratio to  $4/3 - 1/(3m)$ . Moreover, with a further observation that the worst case happens in LPT actually when the terminating machine

processes only three jobs, it is later proved by Coffman and Sethi [28] and Chen [29] that LPT schedule is actually asymptotically optimal with the increase of the number  $k$  of jobs on the terminating machine. In fact, the makespan of any such LPT schedule is at most  $(k + 1)/k - 1/(km)$  times the optimum.

Another variant of LS, as suggested by Graham, is to schedule the  $k$  longest jobs optimally, and then apply LS to the list of remaining jobs. This gives a ratio guarantee of  $1 + (1 - 1/m)/(1 + \lfloor k/m \rfloor)$ . Therefore, for fixed  $m$ , a family of these algorithms for different values of  $k$  constitute a polynomial-time approximation scheme (PTAS), although the running time of  $O(n^{km})$  is huge. Ibarra and Kim [30] show that LPT is no more than  $1 + 2(m - 1)/n$  away from the optimum makespan, if  $n \geq 2(m - 1)\pi$ , where  $\pi = \max_j p_j / \min_j p_j$ .

If machines have different speeds, then Morrison [31] show that LPT for problem  $Q || C_{\max}$  has a worst-case ratio of  $\max\{\sigma/2, 2\}$ , where  $\sigma = \max_i s_i / \min_i s_i$ . A modified LPT algorithm, which assigns the current job to the machine on which it will finish first, is shown by Gonzales, Ibarra, and Sahni [32] to improve the ratio guarantee to  $2 - 2/(m + 1)$ . Subsequently, this guarantee is improved (except for  $m \leq 3$ ) to  $19/12$  by Dobson [33] and Friesen [34].

#### 9.4.2.2 Bin-Packing Techniques

A second main approximation approach is to use the dual form of the problem, known as *bin-packing*. In a bin-packing problem, a number of items of various sizes are to be packed into a minimum number of bins of a common given capacity. Note that problems of scheduling to minimize the makespan and of bin-packing share the same decision version. Naturally, bin-packing techniques are considered for scheduling.

Based on this principle of duality, Coffman, Garey, and Johnson [35] proposed an algorithm for problem  $P || C_{\max}$ , called *Multifit* (MF), to find by binary search the minimum capacity of the  $m$  bins into which the  $n$  items can be packed by an efficient packing heuristic known as *first-fit decreasing* (FFD). In the FFD heuristic, each iteration packs the largest remaining item into the first bin into which it fits. By using the technique of weight functions in bin-packing, they prove that MF has a ratio guarantee of  $\rho + 2^{-k}$ , where  $\rho \leq 1.22$  and  $k$  denotes the number of binary search iterations. Subsequently, by refining the approach of weight functions, Friesen [36] improves the bound of  $\rho$  to 1.2, and Yue [37] further to  $13/11$ , which is shown tight. At the expense of a larger running time, the MF algorithm is refined by Friesen and Langston [38] to achieve a slightly better worst-case ratio of  $72/61 + 2^{-k}$ .

A similar principle of duality between scheduling and bin-packing is considered by Hochbaum and Shmoys [39]. Given the machine capacity  $d$ , a  $\rho$ -dual approximation algorithm ( $\rho > 1$ ) produces a job packing that uses at most the minimum number of machines of capacity  $d$  at the expense of possible capacity violation by no more than  $(\rho - 1)d$ . Using a family of dual approximation algorithms, Hochbaum and Shmoys provide a PTAS for problem  $P || C_{\max}$ .

The MultiFit approach and dual approximation approach are extended to scheduling of uniform machines. When MF is applied to the case of uniform machines, the corresponding bin-packing problem is of bins of different capacities. Friesen and Langston [40] show that the  $\rho$  in the corresponding worst-case ratio  $\rho + 2^{-k}$  of MF is between 1.341 and 1.40, which is later improved to 1.38 by Chen [41]. On the other hand, extension of the dual approximation approach to uniform machines by Hochbaum and Shmoys has led to a PTAS for problem  $Q || C_{\max}$  [42].

#### 9.4.2.3 Linear Programming Techniques

Extensive research has appeared in the literature on computing near-optimal solutions for scheduling models by rounding optimal solutions to linear programming relaxations. Such techniques are not only for minimization of makespan. There are two main general ways to exploit the optimal solution to a linear programming relaxation. It is used either to guide the assignment, deterministic or random, of jobs to machines or to derive job priorities to be used in constructing the schedule.

Extending the optimal solution to a linear programming relaxation by an enumerative process, Potts [43] obtains a 2-approximation algorithm for problem  $R || C_{\max}$  when  $m$  is fixed. Lenstra, Shmoys, and Tardos [22] extend Potts' approach by first establishing that the fractional solution to the linear programming relaxation can be rounded to a good integral approximation in polynomial time, thereby obviating the need

for enumeration and removing the exponential dependence on  $m$ , and then deriving a 2-approximation algorithm even when  $m$  is part of the input.

This approach is further extended to accommodate a more general objective criterion. Shmoys and Tardos [44] introduce a stronger rounding technique than that of Lenstra, Shmoys, and Tardos to develop a polynomial algorithm that can find a schedule with mean job completion time  $M$  and makespan at most  $2T$ , if a schedule with mean job completion time at most  $M$  and makespan at most  $T$  exists.

#### 9.4.2.4 Other Techniques

In approximation of NP-hard problems, a fully polynomial-time approximation scheme (FPTAS) is the strongest possible result that one can hope for unless, of course,  $P = NP$ . It seems that all FPTASs are based on *dynamic programming* (DP) formulations, which always find an optimal solution, though not necessarily in polynomial time. As a first FPTAS, Sahni [45] develops a DP-based approach for approximating problem  $Pm \parallel C_{\max}$ , in which the state variables at each stage  $i$  form a set  $S^{(i)}$  of  $(m - 1)$ -tuples, representing the total workloads of the  $m$  machines. The cardinality  $|S^{(i)}|$  of each such set is controlled by rounding the input data of the instance with use of an *interval partitioning* method. In a similar vein, Horowitz and Sahni [46] derive an FPTAS for problems  $Qm \parallel C_{\max}$  and  $Rm \parallel C_{\max}$ .

In addition to DP formulation, semidefinite programming techniques have recently also played a role in scheduling approximation. The reader is referred to a separate chapter for more details.

## 9.5 Scheduling with Precedence Constraints

---

### 9.5.1 For UET Jobs

Presence of general precedence relationships dramatically increases the difficulty of any scheduling problems. To appreciate the effect of such presence, let us first simplify other elements of scheduling and assume that all jobs are of unit length. Then all such problems are NP-hard if the number of machines is part of the input, since this is already the case for problem  $P \mid prec, p_j = 1 \mid C_{\max}$ , as shown by Lenstra and Rinnooy Kan [47]. Actually, they show that it is NP-complete even to approximate the solution of problem  $P \mid prec, p_j = 1 \mid C_{\max}$  within a factor better than  $4/3$ .

However, this stagnant situation improves immediately if either the form of the precedence relations is relaxed or if the number of machines is fixed. As one of the earliest work addressing scheduling of precedence constraints, Hu's algorithm HLF [5] (see Section 9.2) solves problem  $P \mid tree, p_j = 1 \mid C_{\max}$  to optimality in polynomial time. The HLF approach has been generalized and various algorithms have been designed for a number of special cases of problem  $Pm \mid prec, p_j = 1 \mid C_{\max}$ , which include the case where the precedence graph is an *opposing forest*, that is, the disjoint union of an in-forest where each job has at most one immediate successor and an out-forest where each job has at most one immediate predecessor (Garey, Johnson, Tarjan, and Yannakakis [48]).

Similarly, the formulation by Fujii, Kasami, and Ninomiya [49] of problem  $P2 \mid prec, p_j = 1 \mid C_{\max}$  as a maximum cardinality matching problem, which is solvable in  $O(n^3)$  time, provides the groundwork for the development of a series of improved algorithms. Moreover, polynomial solvability has been extended to *simultaneously* minimizing  $\sum C_j$  (Coffman and Graham [50] and, if preemption is allowed, Coffman, Sethuraman, and Timkovsky [51]) and to including job release dates and due dates (Garey and Johnson [52, 53]).

Some of the efficient algorithms for problem  $P2 \mid prec, p_j = 1 \mid C_{\max}$  are adapted to the more general problem  $P \mid prec, p_j = 1 \mid C_{\max}$ , and are shown to have some good worst-case guarantees. Recently, by replacing the precedence constraints with release dates and due dates, tight lower bounds are derived by Baev, Meleis, and Eichenberger [54] for  $P \mid prec, p_j = 1 \mid C_{\max}$ . These bounds are probably tighter than known lower bounds and empirical experiments demonstrate that over more than 90% of time they lead to the optimal values over a synthetic benchmark.

Research on the case where the machines have different speeds has so far been limited. Using as a guide solutions to the two relaxed problems  $Q2 \mid pmtn \mid C_{\max}$  and  $Q2 \mid p_j = 1 \mid C_{\max}$ , Brucker, Hurink, and

Kubiak [55] solve the problem  $Q2 | chains, p_j = 1 | C_{\max}$  in linear time with respect to the number of chains. However, the complexities of two slightly more difficult problems  $Q2 | tree, p_j = 1 | C_{\max}$  and  $Q3 | chains, p_j = 1 | C_{\max}$  are still open.

We remark that the aforementioned HLF-based algorithms run in time polynomial in the number of jobs, but exponential in the problem size when it comes to the chainlike precedence constraints if a succinct coding scheme is used.

## 9.5.2 For Jobs of General Lengths

It is interesting to observe that research on parallel scheduling with precedence constraints has been mainly concentrated on the preemptive version.

Due to reducibility property mentioned in Section 9.2, the polynomial solvability picture for scheduling jobs with arbitrary processing times under preemption is very similar to that for the nonpreemptive scheduling of UET jobs. This similarity suggests a close relationship between these two models, as Lawler [56] observes as a result of deriving polynomial algorithms for a series of counterparts in the former model of well-solvable problems in the latter model.

In parallel to the complexity results for scheduling nonpreemptable UET jobs, problem  $P | pmtn, prec | C_{\max}$  is shown to be NP-hard by Ullman [57], whereas problems  $P | pmtn, tree | C_{\max}$  and  $Q2 | pmtn, prec | C_{\max}$  are both polynomially solvable by the algorithms of Muntz and Coffman [58, 59] and Horvath, Lam, and Sethi [17], respectively.

Interestingly, the algorithm of Muntz and Coffman [58, 59] also focuses on the critical path of the precedence graph, as is the case in Hu's algorithm HLF for scheduling UET jobs. However, Gonzalez and Johnson [60] use a totally different LS approach for solving problem  $P | pmtn, tree | C_{\max}$ . Their algorithm segregates the jobs into two classes. In one class there is what can be termed the "backbone" of the problem, a superset of those jobs whose start and finish times are fixed in any optimal schedule. The other jobs can, in general, be scheduled with some freedom. Their algorithm runs in  $O(n \log m)$  time. In the same spirit as their LRPT-FM algorithm for problem  $Q | pmtn | C_{\max}$  (see Section 9.3), Horvath, Lam, and Sethi [17] solve problem  $Q2 | pmtn, prec | C_{\max}$  in  $O(mn^2)$  time.

Lam and Sethi [61] adapt the algorithm of Muntz and Coffman for problem  $P | pmtn, prec | C_{\max}$  and show that it has a worst-case ratio of  $2 - 2/m$ . Similarly, in the case of uniform machines, Horvath, Lam, and Sethi [17] prove that this algorithm has a ratio guarantee of  $\sqrt{3m/2}$ , which is tight up to a constant factor. To improve the Muntz–Coffman algorithm, Jaffe [62] suggests scheduling jobs without unforced idleness by always using fastest machine available. He proves that this improves the ratio guarantee to  $\sqrt{m} + 1/2$ .

While also consider multiprocessor jobs, Blazewicz and Liu [63] study problems  $P | chains | C_{\max}$  and  $P | chains, r_j | C_{\max}$ . As a variant of LPT and HLF, their algorithms run in  $O(n \log n)$  time and in  $O(n^2)$  time, respectively, and are still applicable if preemption is allowed. Their work represents one of a few results dealing with nonpreemptable jobs of arbitrary lengths and of precedence constraints. The same remark applies here as the one made at the end of Section 9.5.1.

As another recent study dealing with nonpreemptable jobs, Hurink and Knust [64] present complexity results that have influence on the strength of LS algorithm. They show that, with sequence-dependent setup times, dominant schedules for problem  $P | prec, s_{ij} | C_{\max}$  cannot be calculated efficiently with LS techniques.

## References

- [1] Chen, B., Potts, C.N., and Woeginger, G.J., A review of machine scheduling: complexity, algorithms and approximability, in: Du, D.-Z. and Pardalos, P.M. (eds.), *Handbook of Combinatorial Optimization*, Kluwer, Boston, 1998, 21–169.
- [2] Graham, R.L., Bounds on multiprocessing timing anomalies, *SIAM Journal on Applied Mathematics* 17 (1969), 416–429.

- [3] Smith, W.E., Various optimizers for single-stage production, *Naval Research Logistics Quarterly* 3 (1956), 59–66.
- [4] Schrage, L., A proof of the shortest remaining processing time processing discipline, *Operations Research* 16 (1968), 687–690.
- [5] Hu, T.C., Parallel sequencing and assembly line problems, *Operations Research* 9 (1961), 841–848.
- [6] Liu, C.L., Optimal scheduling on multi-processor computing systems, *Proceedings of the 13th Annual Symposium on Switching and Automatic Theory*, IEEE Computer Society, Los Alamitos, CA, 1972, 155–160.
- [7] Hong, K.S. and Leung, J.Y.-T., Some results on Liu's conjecture, *SIAM Journal on Discrete Mathematics* 5 (1992), 500–523.
- [8] Braun, O. and Schmidt, G., Parallel processor scheduling with limited number of preemptions, *SIAM Journal on Computing* 62 (2003), 671–680.
- [9] Coffman, Jr., E.G. and Garey, M.R., Proof of the  $4/3$  conjecture for preemptive vs. non-preemptive two-processor scheduling, *Journal of the ACM* 20 (1993), 991–1018.
- [10] Timkovsky, V.G., Identical parallel machines vs. unit-time shops and preemptions vs. chains in scheduling complexity, *European Journal of Operational Research* 149 (2003), 355–376.
- [11] Brucker, P., Kravchenko, S.A., and Sotskov, Y.N., Preemptive job-shop scheduling problems with a fixed number of jobs, *Mathematical Methods of Operations Research* 49 (1999), 41–76.
- [12] Kravchenko, S.A. and Sotskov, Y.N., Optimal makespan schedule for three jobs on two machines, *Mathematical Methods of Operations Research* 43 (1996), 233–238.
- [13] Brucker, P. and Kravchenko, S.A., Preemption can make parallel machine scheduling problems hard, *Osnabruecker Schriften zur Mathematik*, Reihe P (Preprints), Heft 211 (1999).
- [14] Sitters, R., Two NP-hardness results for preemptive minsum scheduling of unrelated parallel machines, in: Aardal, K. and Gerards, B. (eds.), *Integer Programming and Combinatorial Optimization*, LNCS 2081 (2001), 396–405.
- [15] McNaughton, R., Scheduling with deadlines and loss functions, *Management Science* 6 (1959), 1–12.
- [16] Liu, J.W.S. and Yang, A., Optimal scheduling of independent tasks on heterogeneous computing systems, *Proceedings of the ACM Annual Conference*, 1974, 38–45.
- [17] Horvath, E.C., Lam, S., and Sethi, R., A level algorithm for preemptive scheduling, *Journal of the ACM* 24 (1977), 32–43.
- [18] Gonzalez, T. and Sahni, S., Preemptive scheduling of uniform processor systems, *Journal of the ACM* 25 (1978), 92–101.
- [19] Lawler, E.L. and Labetoulle, J., On preemptive scheduling of unrelated parallel processors by linear programming, *Journal of the ACM* 25 (1978), 612–619.
- [20] Garey, M.R. and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.
- [21] Garey, M.R. and Johnson, D.S., Strong NP-completeness results: motivation, examples and implications, *Journal of the ACM* 25 (1978), 499–508.
- [22] Lenstra, J.K., Shmoys, D.B., and Tardos, É., Approximation algorithms for scheduling unrelated parallel machines, *Mathematical Programming* 46 (1990), 259–271.
- [23] Dell'Amico, M. and Martello, S., Optimal scheduling of tasks on identical parallel processors, *ORSA Journal on Computing* 7 (1995), 191–200.
- [24] Van de Velde, S.L., Duality-based algorithms for scheduling unrelated parallel machines, *ORSA Journal on Computing* 5 (1993), 192–205.
- [25] Martello, S., Soumis, F., and Toth, P., Exact and approximation algorithms for makespan minimization on unrelated parallel machines, *Discrete Applied Mathematics* 75 (1997), 169–188.
- [26] Mokotoff, E. and Chrétienne, P., A cutting plane algorithm for the unrelated parallel machine scheduling problem, *European Journal of Operational Research* 141 (2002), 515–525.
- [27] Graham, R.L., Bounds for certain multiprocessing anomalies, *Bell System Technical Journal* 45 (1966), 1563–1581.

- [28] Coffman, Jr., E.G. and Sethi, R., A generalized bound on LPT sequencing, *Revue Française d'Automatique Informatique, Recherche Operationnelle*, Supplément au 10 (1976), 17–25.
- [29] Chen, B., A note on LPT scheduling, *Operations Research Letters* 14 (1993), 139–142.
- [30] Ibarra, O.H. and Kim, C.E., Heuristic algorithms for scheduling independent tasks on nonidentical processors, *Journal of the ACM* 24 (1977), 280–289.
- [31] Morrison, J.F., A note on LPT scheduling, *Operations Research Letters* 7 (1988), 77–79.
- [32] Gonzalez, T., Ibarra, O.H., and Sahni, S., Bounds for LPT schedules on uniform processors, *SIAM Journal on Computing* 6 (1977), 155–166.
- [33] Dobson, G., Scheduling independent tasks on uniform processors, *SIAM Journal on Computing* 13 (1984), 705–716.
- [34] Friesen, D.K., Tighter bounds for LPT scheduling on uniform processors, *SIAM Journal on Computing* 16 (1987), 554–660.
- [35] Coffman, Jr., E.G., Garey, M.R., and Johnson, D.S., An application of bin-packing to multiprocessor scheduling, *SIAM Journal on Computing* 7 (1978), 1–17.
- [36] Friesen, D.K., Tighter bounds for the multifit processor scheduling algorithm, *SIAM Journal on Computing* 13 (1984), 170–181.
- [37] Yue, M., On the exact upper bound for the multifit processor scheduling algorithms, *Annals of Operations Research* 24 (1990), 233–259.
- [38] Friesen, D.K. and Langston, M.A., Evaluation of a MULTIFIT-based scheduling algorithm, *Journal of Algorithms* 7 (1986), 35–59.
- [39] Hochbaum, D.S. and Shmoys, D.B., Using dual approximation algorithms for scheduling problems: Theoretical and practical results, *Journal of the ACM* 34 (1987), 144–162.
- [40] Friesen, D.K. and Langston, M.A., Bounds for multifit scheduling on uniform processors, *SIAM Journal on Computing* 12 (1983), 60–70.
- [41] Chen, B., Tighter bounds for MULTIFIT scheduling on uniform processors, *Discrete Applied Mathematics* 31 (1991), 227–260.
- [42] Hochbaum, D.S. and Shmoys, D.B., A polynomial approximation scheme for machine scheduling on uniform processors: Using the dual approximating approach, *SIAM Journal on Computing* 17 (1988), 539–551.
- [43] Potts, C.N., Analysis of a linear programming heuristic for scheduling unrelated parallel machines, *Discrete Applied Mathematics* 10 (1985), 155–164.
- [44] Shmoys, D.B. and Tardos, É., An approximation algorithm for the generalized assignment problem, *Mathematical Programming* 62 (1993), 461–474.
- [45] Sahni, S., Algorithms for scheduling independent tasks, *Journal of the ACM* 23 (1976), 116–127.
- [46] Horowitz, E. and Sahni, S., Exact and approximate algorithms for scheduling nonidentical processors, *Journal of the ACM* 23 (1976), 317–327.
- [47] Lenstra, J.K. and Rinnooy Kan, A.H.G., Complexity of scheduling under precedence constraints, *Operations Research* 6 (1978), 22–35.
- [48] Garey, M.R., Johnson, D.S., Tarjan, R.E., and Yannakakis, M., Scheduling opposing forests, *SIAM Journal on Algebraic Discrete Mathematics* 4 (1983), 72–93.
- [49] Fujii, M., Kasami, T., and Ninomiya, K., Optimal sequencing of two equivalent processors, *SIAM Journal on Applied Mathematics* 17 (1969), 784–789. Erratum: *SIAM Journal on Applied Mathematics* 20 (1971), 141.
- [50] Coffman, Jr., E.G. and Graham, R.L., Optimal scheduling for two-processor systems, *Acta Informatica* 1 (1972), 200–213.
- [51] Coffman, Jr., E.G., Sethuaraman, J., and Timkovsky, V.G., Ideal preemptive schedules on two processors, *Acta Informatica* (to be published).
- [52] Garey, M.R. and Johnson, D.S., Scheduling tasks with nonuniform deadlines on two processors, *Journal of the ACM* 23 (1976), 461–467.
- [53] Garey, M.R. and Johnson, D.S., Two-processor scheduling with start-times and deadlines, *SIAM Journal on Computing* 6 (1977), 416–426.

- [54] Baev, I.D., Meleis, W.M., and Eichenberger, A., Lower bounds on precedence-constrained scheduling for parallel processors, *Information Processing Letters* 83 (2002), 27–32.
- [55] Brucker, P., Hurink, J., and Kubiak, W., Scheduling identical jobs with chain precedence constraints on two uniform machines, *Mathematical Methods of Operations Research* 49 (1999), 211–219.
- [56] Lawler, E.L., Preemptive scheduling of precedence-constrained jobs on parallel machines, in: Dempster M.A.H., Lenstra, J.K., and Rinnooy Kan, A.H.G. (eds.), *Deterministic Stochastic Scheduling*, Reidel, Dordrecht, 1982, 101–123.
- [57] Ullman, J.D., Complexity of sequencing problems, in: Coffman, Jr., E.G. (ed.), *Computer and Job-Shop Scheduling Theory*, Wiley, New York, 1976, 139–164.
- [58] Muntz, R.R. and Coffman, Jr., E.G., Optimal scheduling on two-processor systems, *IEEE Transactions on Computing* C-18 (1969), 1014–1020.
- [59] Muntz, R.R. and Coffman, Jr., E.G., Preemptive scheduling of real tasks on multiprocessor systems, *Journal of the ACM* 17 (1970), 324–338.
- [60] Gonzalez, T. and Johnson, D.B., A new algorithm for preemptive scheduling of trees, *Journal of the ACM* 27 (1980), 287–312.
- [61] Lam, S. and Sethi, R., Worst case analysis of two scheduling algorithms, *SIAM Journal on Computing* 6 (1977), 518–536.
- [62] Jaffe, J.M., An analysis of preemptive multiprocessor job scheduling, *Mathematics of Operations Research* 5 (1980), 415–521.
- [63] Blazewicz, J. and Liu, Z., Linear and quadratic algorithms for scheduling chains and opposite chains, *European Journal of Operational Research* 137 (2002), 248–264.
- [64] Hurink, J. and Knust, S., List scheduling in a parallel machine environment with precedence constraints and setup times, *Operations Research Letters* 29 (2001), 231–239.