Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 188 (2008) 315–329
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Invited Review

A state-of-the-art review on scheduling with learning effects

Dirk Biskup

Department of Business Administration and Economics, University of Bielefeld, P.O. Box 10 01 31, 33501 Bielefeld, Germany

Received 30 January 2007; accepted 23 May 2007
Available online 2 June 2007

Abstract

Recently learning effects in scheduling have received considerable attention in the literature. All but one paper are based on the learning-by-doing (or autonomous learning) assumption, even though proactive investments in know how (induced learning) are very important from a practical point of view. In this review we first discuss the questions why and when learning effects in scheduling environments might occur and should be regarded from a planning perspective. Afterwards we give a concise overview on the literature on scheduling with learning effects.

Keywords: Scheduling; Learning

1. Introduction

No doubt: learning can play a role in manufacturing environments and learning effects have been proven to exist by many empirical studies (e.g. Conway and Schultz, 1959; Venezia, 1985, p. 197; Cochran, 1960, p. 323; Ghemawat, 1985; Day and Montgomery, 1983 or Webb, 1994). But what has learning to do with scheduling? Scheduling tasks are typically subject to short-term production planning and learning – on the other hand – takes time. In the present paper we discuss how to overcome this obvious antagonism and explain that learning might indeed be relevant for scheduling tasks. Therefore we review some aspects of learning theory in the next section. In Section 3, we discuss the role of learning effects in scheduling environments. Sections 4 and 5 summarize the existing literature on scheduling with position-based and sum-of-processing-time based learning effects, respectively. Some concluding remarks can be found in Section 6.

2. Learning theory

During 1930s, Wright (1936) described that in the aircraft industry the working costs per unit declined with an increasing production output. Wright (1936, p. 126) formulated the so-called 80% hypothesis, stating that with every redoubling of output the unit processing time decreases by 20%. Wright (1936) probably was the

E-mail address: dbiskup@wiwi.uni-bielefeld.de

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2007.05.040

first researcher examining learning effects on a scientific basis.[1] In general the theory of learning effects states that the time needed to produce a single unit continuously decreases with the processing of additional units and that due to the declining processing times the costs per unit also decline (see Yelle, 1979; Dutton and Thomas, 1984).

If learning effects exist in a production environment, these learning effects may have enormous consequences for the production planning on one hand and for the calculation of the variable production costs on the other hand. A prominent example stems from the semiconductor industry, where efficiency gains cause price drops of ≈10–30% per year (see Webb, 1994).

The learning theory assumes a mass production environment where identical products are processed consecutively. Hence the processing times of the single operations would be identical if not influenced by learning. Learning is frequently formalized by using the power formula

$$p_{[k]} = p_{[1]} k^a \tag{1}$$

with $p_{[k]}$ symbolising the required processing time for the kth unit of the cumulative production quantity k; $p_{[1]}$ is the processing time of the first unit and $a = \log_2 \mathrm{LR} \leqslant 0$ is the learning index depending on the learning rate LR. This context can be clarified as follows: The learning rate is defined to describe the learning effect on each redoubling of the output:

$$p_{[2k]} = p_{[k]} \mathrm{LR}. \tag{2}$$

The lower the learning rate the higher the effects from learning. Substitution of the processing times by (1) leads to

$$\mathrm{LR} = \frac{p_{[2k]}}{p_{[k]}} = \frac{p_{[1]}(2k)^a}{p_{[1]} k^a} = 2^a \tag{3}$$

or

$$a = \frac{\log \mathrm{LR}}{\log 2} = \log_2 \mathrm{LR}. \tag{4}$$

Thus for the 80% hypothesis $a = \log_2 0.8 = -0.322$ holds. Let the processing time of the first product be 100, then $p_{[k]} = 100 k^{-0.322}$ describes the learning curve depending on the cumulative output. As cumulative output is typically measured by integer numbers, the learning curve would consists of single dots which are interconnected in Fig. 1.

The existence of learning effects in manufacturing environments has widely been confirmed and accepted (see, for example, Conway and Schultz, 1959; Venezia, 1985, p. 197; Cochran, 1960, p. 323; Ghemawat, 1985; Day and Montgomery, 1983). However, despite the general acceptance of learning effects some shortcomings are inherent in its traditional functional form (see Lapré et al., 2000, p. 598 or Biskup and Simons, 2004):

- The power formula suggests that the decreasing costs per unit are externally given, i.e. are an automatism. It does not provide management with hints how to reduce costs beyond cost reductions resulting from learning-by-doing (see Zangwill and Kantor, 1998, p. 910).
- The widely varying learning rates within industries or even plants indicate that the personnel's attitude towards learning has an impact on the learning rate as well (see Sterman, 1994, p. 293). The same is true for the personnel's knowledge diversity (see Lapré and Van Wassenhove, 2001, p. 1323). These facts are not reproduced adequately by an exogenously defined learning rate.

[1] The fact that the time for the production of airplanes decreases with the output has already been mentioned by Rohrbach (1927, p. 65–66). However, Rohrbach did not analyse the learning effects. This is why Wright (1936) is considered to be the first researcher examining learning effects in production environments. In psychology learning effects have been described in the late 19th century (see, for example, Thurstone, 1919).

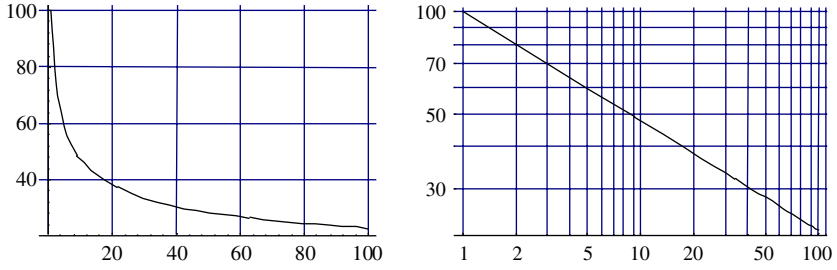

Fig. 1. The learning curve with a learning rate of 80% depicted in a normal and a double-logarithmic coordinate system.

In summary, an inherent problem of the learning curve described in (1) is its pure descriptive character: The cumulative output and the learning rate are the only parameters determining the learning. The learning given by (1) seems to be an externally given automatism, leading to lower costs per piece with higher output. The formulation of (1) does not contain any hints for the management how additional efficiency gains and thus additional cost effects, exceeding the standard "learning-by-doing", might be realized. Nevertheless it is widely accepted that a company is able to influence learning effects (by training) and that investments in training can be advantageous from an economic point of view (see Dutton and Thomas, 1984, p. 240).

To overcome this discrepancy the younger learning literature distinguishes autonomous learning (called 'learning-by-doing' or 'Horndal-Effect') and induced learning (see, for example, Adler and Clark, 1991 or Upton and Kim, 1998). Autonomous learning, as depicted in Eq. (1), cannot be influenced by the company, but results from repeating similar operations, leading to a higher familiarization and routine. Induced learning on the other hand describes the possibility of the management investing in the know how of the employees and managing the learning rate for example by additional training, changing the production environment or processes, incentive schemes, etc. For induced learning a crucial question is to determine the optimal learning rate, as a reduction of the learning rate is usually related to higher costs. In general the definitions of autonomous and induced learning are not clear-cut. While Hax and Majluf (1982) consider changes in the work-process to be independent of autonomous learning, Day and Montgomery (1983) subsume all effects causing a higher work-efficiency under this terminology. In the following we refer to autonomous learning when learning is caused by repetition, whereas induced learning implies every management activity in process design or controlled training.

3. Learning effects and scheduling

Learning effects might be important in short term production planning. Keachie and Fontana (1966) are the first to consider learning effects when calculating optimal lot-sizes: If learning effects occur during the production of the first lot, the lot is finished earlier compared to the case without learning effects. With respect to a constant demand rate this leads to higher holding costs and hence to smaller optimal lot-sizes. For a literature review, see Li and Cheng (1994). Although scheduling is one of the main tasks in short-term production planning, the effects of learning in this context were considered for the first time by Biskup (1999).

Scheduling problems often occur when different jobs or orders have to be processed on one or multiple machines. An inherent characteristic of these production environments is a high level of human activities. These typically are

- setting up the machines (for scheduling purposes the make-ready time is usually assumed to be included in the processing times),
- cleaning the machines after the processing of a job or after a specific period of time,
- operating and controlling the machines,
- planned maintenance of the machines,
- removal of machine failures,
- reading, understanding and interpretation of machine data,
- all kinds of handwork – either during the regular processing time or afterwards to correct errors.

As there is a significant involvement of humans in scheduling environments, the number of activities subject to learning is high, too. Hence it seems reasonable to consider learning in scheduling environments. But learning effects do not occur on every shop floor. At a site that is processing for years the same range of products with the – more or less – same workforce and an unchanged machine environment, learning effects are indeed negligible. They have been realized already. Learning effects are important if the production environment changes. Typical examples are

- new (inexperienced) employees,
- investments in new machines or replacement of equipment by technically refined machines,
- workflow changes resulting from internal optimization or externally given requirements,
- the acceptance of jobs that have never been produced before.

Furthermore all small changes to the production environment like a software update, a new design of the format of important documents (e.g. bill of material), a new organization of the spare parts depot, etc. will cause learning effects. Each time an employee needs to get accustomed to a new situation a learning experience will occur.

The interesting question now obviously is, how learning can and should be modelled to represent the reality best. The answer is: It depends on the production environment. Some of the above-mentioned tasks are independent of the overall processing times of the particular operations (like the setups or quality control), others are not. For this reason two different approaches to learning in scheduling environments have been suggested. The first one can be best described as *position-based* learning, meaning that learning is effected by the pure number of jobs being processed. Alternatively, the *sum-of-processing-time approach* takes into account the processing time of all jobs processed so far. Both approaches have their validity. The position-based approach assumes that learning takes place by processing time independent operations like setting up the machines. This seems to be a realistic assumption for the case that the actual processing of the job is mainly machine-driven and has (near to) none human interference. Practical examples are the processing of memory chips and circuit boards or running bottling plants. The sum-of-processing-time approach takes into account the experience the workers gain from producing the jobs. This might, for example, be the case for offset printing, where running the press itself is a highly complicated and error-prone process. Highly customized products, the production of high-end electric tools, maintenance of airplanes, pimping cars, etc. are further examples for this approach.

This overview follows the general distinction between position-based and sum-of-processing-time-based learning effects. The next section will present the work on position-based learning effects and existing refinements, the Section 5 will concentrate on the sum-of-completion-time learning effects discussed in the literature.

4. Position-based learning effects

Biskup (1999) uses a modification of formula (1) to formulate autonomous learning effects in scheduling environments: With $p_i, i = 1, \ldots, n$, being the normal processing time (i.e. the processing time without any learning effects) of job i, he assumes for a single-machine scheduling environment the following expression for learning:

$$p_{ir} = p_i r^a, \tag{5}$$

whereby p_{ir} is the processing time of job i if it is scheduled in position r. With respect to Eq. (5) the time needed to perform an operation decreases by the number of repetitions, meaning that learning is primarily based on the repetition of processing-time independent tasks such as setups, reading data, etc. Eq. (5) is identical to (1) if all jobs have the same processing times; otherwise there might be differences between the formulations. However, the assumptions with respect to the underlying production environments also differ. In (1) identical orders or jobs are assumed, whereas in a scheduling environment different jobs with different processing times may exist. Note that both formulas are just an approximation of how learning effects might influence processing times.

Eq. (5) and all other position-based learning effects have the shortcoming that even though it is argued that the learning primarily takes place as a result of 'processing-time independent operations' (Biskup, 1999), the

learning is applied to the processing times (and not to the setup times) of the jobs (Koulamas and Kyparisis, 2008). If, for example, the production process itself is 100% automated, it might be more appropriate to assume that learning takes place during the setup time of the machine only. A situation like this can easily be formulated: Assuming that the normal processing time p_i consists of a setup time s_i and a production time v_i, the following would hold: $p_{ir} = s_{ir} + v_i$ with $s_{ir} = s_i r^a$. The setup times can include times for special make-readies after the processing of a job or cleaning times.

Biskup (1999) was able to prove that with the learning effect described in (5), minimizing the sum of completion times (or total completion time) on a single machine, $1/p_{ir} = p_i r^a / \sum C_i$, can be solved in polynomial time by ordering the jobs according to the shortest processing time (SPT) rule. The proof makes use of the well-known pair-wise job interchange technique. For the three field notation used see Graham et al., (1979). Furthermore the single-machine common due date problem with different (but not job-individual) penalties for earliness $(E_i = \max\{d_i - C_i, 0\})$, tardiness $(T_i = \max\{C_i - d_i, 0\})$ and the completion time, $1/p_{ir} = p_i r^a, d_i = d/\sum(w_1 E_i + w_2 T_i + w_3 C_i)$, can be solved in $O(n^3)$ time by formulating the problem as an assignment problem (see Biskup, 1999). Mosheiov (2001a) showed that some well-known solutions for single-machine scheduling, namely the earliest due date (EDD) rule for minimizing maximum lateness, the weighted shortest processing time (WSPT) rule for minimizing the sum of weighted completion times, and Moore's algorithm for minimizing the number of tardy jobs are all not valid under learning assumptions. Furthermore, Mosheiov (2001a) demonstrated that the classical one-machine makespan minimization problem, $1/p_{ir} = p_i r^a / C_{\max}$, is optimized by the SPT sequence and that the single-machine common due date problem with different (but not job-individual) penalties for earliness, tardiness and the common due date, $1/p_{ir} = p_i r^a, d_i = d/\sum(w_1 E_i + w_2 T_i + w_3 d)$, can be solved in $O(n^3)$ time by formulating the problem as an assignment problem. He furthermore showed that the single-machine bi-criterial scheduling problem with the goal to simultaneously minimize total completion time and variation of completion times, $1/p_{ir} = p_i r^a / w \sum C_i + (1-w) \sum \sum |C_i - C_k|$, can be solved in polynomial time by formulating it as an assignment problems, too. Mosheiov (2001b) considered the parallel-machine total completion time minimization problem, $Pm/p_{ir} = p_i r^a / \sum C_i$. For a given number of parallel machines m, the problem can be solved by solving as many assignment problems as there are allocations of jobs to machines. Lee et al. (2004) worked on a different bi-criterial single-machine scheduling problem, namely jointly minimizing the sum of completion times and the maximum tardiness, $1/p_{ir} = p_i r^a / w \sum C_i + (1-w) T_{\max}$. Maximum tardiness $T_{\max} = \max_{i=1,\ldots,n}\{T_i\}$ takes into account only tardy jobs and is in so far different to maximum lateness $L_{\max} = \max_{i=1,\ldots,n}\{L_i\}$, $L_i = C_i - d_i$. The authors developed a branch-and-bound algorithm to solve this problem. This algorithm, based on numerous dominance rules, is able to solve problems with up to 30 jobs. Lee and Wu (2004) considered the two-machine flowshop problem with the goal to minimize total completion time, $F2/p_{ijr} = p_{ij} r^a / \sum C_i$. They assumed that learning takes place on each machine separately. This assumption is very common for multiple-machine scheduling with learning considerations. Very similar to (5), the learning effect can be formalized as $p_{ijr} = p_{ij} r^a$, with $j = 1, \ldots, m$ being the machine index and p_{ijr} being the processing time of job i if it is scheduled in position r on machine j. As $F2//\sum C_i$ is NP-hard even without learning (Gonzalez and Sahni, 1978), the authors concentrated on developing dominance properties, a strong lower bound and a branch-and-bound algorithm again. They are able to solve problems with up to 35 jobs with their branch-and-bound algorithm in a reasonable time. Chen et al. (2006) combined the work of the previous two papers. They considered a bi-criterial two-machine flowshop scheduling problem with the goal to jointly minimize the sum of completion times and the maximum tardiness, $F2/p_{ijr} = p_{ij} r^a / w \sum C_i + (1-w) T_{\max}$. This problem is of course NP-hard, too. Hence the authors developed dominance properties, a lower bound and a branch-and-bound algorithm that is capable of solving problems with up to 18 jobs. Bachman and Janiak (2004) worked with two alternative learning effects in their paper, one of them being that of Eq. (5). They were able to show that some special cases of single-machine scheduling with weighted completion times are still polynomially solvable: $1/p_{ir} = p_i r^a, p_i = p/\sum w_i C_i$ (all jobs have the same normal processing time) can be solved by scheduling the jobs in non-increasing order of w_i and $1/p_{ir} = p_i r^a / \sum w p_i C_i$ (the weight is a multiple of the normal processing time) can be solved by scheduling the jobs in the SPT sequence. Furthermore Bachman and Janiak (2004) were able to prove that the single-machine makespan minimization problem with release times (r_i), $1/r_i$, $p_{ir} = p_i r^a / C_{\max}$, is NP-hard in the strong sense. Zhao et al. (2004) worked on many polynomially solvable special cases: They were able to show that $1/p_{ir} = p_i r^a / \sum w_i C_i$ is

minimized by the WSPT sequence if the jobs have agreeable weights, i.e. if $p_i \leqslant p_k$ implies $w_i \geqslant w_k$ and that $1/p_{ir} = p_i r^a / T_{\max}$ is minimized by the EDD rule if the due date are agreeable, i.e. if $p_i \leqslant p_k$ implies $d_i \leqslant d_k$. Both proofs, again, use the pair-wise job interchange technique. Let Qm denote uniform machines, i.e. machines that differ in speed. Uniform machines is a generalization of identical parallel machines, as all parallel machines are assumed to have the same speed in the identical parallel machine case. Zhao et al. (2004) were able to develop efficient algorithms for $Qm/p_{ir} = p_i r^a, p_i = 1/\sum w_i C_i$ and $Qm/p_{ir} = p_i r^a, p_i = 1/L_{\max}$ that are based on WSPT and EDD orderings of the jobs, respectively. Furthermore they showed that $F2/p_{ijr} = p_{ij} r^a$, $p_{2j} = p_2/\sum C_i$ and $F2/p_{ijr} = p_{ij} r^a, p_{2j} = p_2/C_{\max}$ (identical processing time for all jobs on machine 2) can be solved optimally by scheduling the jobs according to the SPT sequence on machine 1. If the machines form an increasing series of dominating machines (idm), i.e. $\max_{i=1,\ldots,n}\{p_{ij}\} \leqslant \min_{i=1,\ldots,n}\{p_{ij+1}\}, j = 1,\ldots,m-1$, then a permutation schedule is optimal for $Fm/p_{ijr} = p_{ij} r^a, \text{idm}/\sum C_i$ and $Fm/p_{ijr} = p_{ij} r^a, \text{idm}/C_{\max}$ with the jobs on the positions $2,\ldots,n$ on machine m ordered by the SPT sequence. Cheng et al. (2007) confirmed these results for the case that idle times between jobs are not allowed. Koulamas and Kyparisis (2007) presented a new learning effect for scheduling environments (see below) and derived some results on special cases for the two-machine flowshop problem with position based learning (5). Let ord (for ordered) denote that for all jobs holds: $p_{i1} \leqslant p_{i2}$ and $p_{i2} \leqslant p_{k2}$ whenever $p_{i1} \leqslant p_{k1}$ for any two jobs $i,k \in \{1,\ldots,n\}$. Let prp (for proportional) denote that for all jobs holds: $p_{i2} = w p_{i1}$ with $w \geqslant 1$ is a constant factor. Koulamas and Kyparisis (2007) were able to show that $F2/p_{ijr} = p_{ij} r^a, \text{ord}/\sum C_i, F2/p_{ijr} = p_{ij} r^a, \text{prp}/\sum C_i, F2/p_{ijr} = p_{ij} r^a, \text{ord}/C_{\max}$, and $F2/p_{ijr} = p_{ij} r^a, \text{prp}/C_{\max}$ can be solved by scheduling the jobs according to the SPT sequence. An interesting result stems from Wang and Xia (2005) who demonstrated that Johnson's famous rule for $F2//C_{\max}$ does not necessarily lead to an optimal schedule if learning effects are taken into account, i.e. for $F2/p_{ijr} = p_{ij} r^a / C_{\max}$. Eren and Guner (2007) worked on the well-known single-machine total tardiness problem with learning effects: $1/p_{ir} = p_i r^a / \sum T_i$. This problem is NP-hard even without learning effects (Du and Leung, 1990), thus the authors developed an integer programming formulation and tackled the problem by means of meta-heuristical approaches, namely tabu search and simulated annealing. Kuo and Yang (2007) combined position-based learning effects and setup times that depend on the jobs already scheduled. The idea of past-sequence-dependent (psd) setup times, recently introduced by Koulamas and Kyparisis (2008), can be explained as follows. The setup time of the rth job, $s_{[r]}$, depends on the sum of processing times of the jobs scheduled on position 1 to $r-1$: $s_{[1]} = 0$ and $s_{[r]} = w\sum_{i=1}^{r-1} p_{[r]}, r = 2,\ldots,n$. $[i]$ denotes the ith position in the sequence. Kuo and Yang (2007) were able to prove that $1/p_{ir} = p_i r^a$, $\text{psd}/\sum C_i$ and $1/p_{ir} = p_i r^a$, $\text{psd}/C_{\max}$ can be optimally solved by ordering the jobs according to the SPT sequence. The authors were able to find $O(n \log n)$ algorithms to solve $1/p_{ir} = p_i r^a$, $\text{psd}/\sum\sum |C_i - C_k|$ and $1/p_{ir} = p_i r^a$, $\text{psd}/\sum(w_1 E_i + w_2 T_i + w_3 d)$ by calculating positional weights. Even when job-dependent learning rates are considered (Eq. (6) – see below), all four problem settings remain polynomially solvable by formulating them as assignment problems. Recently Wu et al. (2007a,b) tackled the single machine maximum lateness problem by means of simulated annealing. They were able to show in their paper that $1/p_{ir} = p_i r^a / L_{\max}$ with agreeable due dates is optimally solved by the EDD sequence.

Meanwhile several extensions have been suggested to the basic position-based learning model (5).

4.1. Job-dependent position-based learning effects

Mosheiov and Sidney (2003) introduced a more general form of position-based learning effects. They assumed that learning takes place by

$$p_{ir} = p_i r^{a_i}, \tag{6}$$

with $a_i \leqslant 0, i = 1,\ldots,n$ being the job-dependent learning indices. Mosheiov and Sidney (2003) argue that the learning process of the worker may be significantly affected by the job itself. A worker, for example, will hardly realize any learning gains if he is to produce a job that he has done many times before. However, on new or somehow different jobs the worker will realize (higher) gains from learning in the future.

Mosheiov and Sidney (2003) showed that $1/p_{ir} = p_i r^{a_i}/C_{\max}, 1/p_{ir} = p_i r^{a_i}/\sum C_i$, and $1/p_{ir} = p_i r^{a_i}, d_i = d/\sum(w_1 E_i + w_2 T_i + w_3 d)$ can all be solved in $O(n^3)$ time by formulating them as assignment problems. They furthermore considered an unrelated machine flow-time minimization problem, $Qm/p_{ir} = p_i r^{a_i}/\sum C_i$: For a

given number of machines m, the problem can be solved by solving as many assignment problems as allocations of jobs to machines exist, see also Mosheiov (2001b). Mosheiov and Sidney (2005) considered the single-machine problem of minimizing the number of tardy jobs (U_i) when a common due date is given: $1/p_{ir} = p_i r^{a_i}, d_i = d/\sum U_i$: They were able to formulate this problem as a different version of the classical assignment problem with a total running time of $O(n^3 \log n)$. Lin (in press) recently confirmed that $1/p_{ir} = p_i r^a/\sum U_i$ and thus $1/p_{ir} = p_i r^{a_i}/\sum U_i$ are NP-hard in the strong sense. Mosheiov and Sidney (2003, 2005) did not restrict their research to exponential learning functions. They assumed general monotone (job-dependent) learning with the only requirement that p_{ir} is non-increasing in r.

4.2. Autonomous position-based and induced learning

Biskup and Simons (2004) took into consideration the possibility to invest in the know how of the employees and thus to manage learning. Training employees, distributing manuals, hanging up "cheat sheets", etc. are investments in know how that are very common in practice. For this so-called induced learning a crucial problem is to determine the optimal learning rate, as investments in know how simply cost time and money. Biskup and Simons (2004) assume that autonomous and induced learning takes place as follows:

$$p_{ir} = p_i r^{\log_2(1-x)\mathrm{LR}}. \tag{7}$$

The (standard or autonomous) learning rate LR can be reduced by x ($0 \leqslant x \leqslant x_{\max} < 1$) while this investment is depicted by a convex non-decreasing cost function $k(x)$. Biskup and Simons (2004) considered the goal to jointly minimize earliness and tardiness penalties and the costs of investing in induced learning, i.e. $1/p_{ir} = p_i r^{\log_2(1-x)\mathrm{LR}}, d_i = d/\sum(E_i + T_i) + k(x)$. After deriving certain convexity results for the objective function, the authors were able to present an $O(n^3)$ algorithm to solve this problem. However, including the effects of induced learning in scheduling problems is rather hard, as most scheduling problems have time-based not cost-based objective functions.

4.3. Position-based learning and deteriorating jobs

Wang (2006) considered scheduling problems where all jobs are (negatively) effected by the start time t and (positively) effected by learning:

$$p_{ir} = (p_i + wt)r^a, \tag{8}$$

w is the amount of increase of processing time per unit delay in its starting time. Wang (2006) was able to show that $1/p_{ir} = (p_i + wt)r^a/C_{\max}, 1/p_{ir} = (p_i + wt)r^a/\sum C_i$, and $1/p_{ir} = (p_i + wt)r^a$, $p_i = cw_i/\sum w_i C_i$ can be solved by the SPT sequence. The problems $1/p_{ir} = (p_i + wt)r^a/\sum w_i C_i$ with agreeable weights and $1/p_{ir} = (p_i + wt)r^a$, $p_i = p/\sum w_i C_i$ are optimally solved by the WSPT sequence and the problems $1/p_{ir} = (p_i + wt)r^a/L_{\max}$ with agreeable due dates and $1/p_{ir} = (p_i + wt)r^a$, $p_i = p/L_{\max}$ can be solved to optimality by the EDD sequence. The results on several flow-shop problems with learning and deteriorating jobs can be found in Appendix A. Wang and Cheng (2007) worked on a slightly different model with a common normal job processing time but with job individual weights:

$$p_{ir} = (p + w_i t)r^a. \tag{9}$$

They used $p = 1$ for all jobs throughout their paper. By concentrating on the problem $1/p_{ir} = (1 + w_i t)r^a/C_{\max}$ Wang and Cheng (2007) were able to show that the largest growth rate (LGR) schedule, which sequences the jobs in non-increasing order of w_i, is unbounded for this problem, see also Lee (2004). The authors were able to derive polynomially solvable cases for some very special settings of the parameters w_i. A similar approach to scheduling with learning and deteriorating jobs stems from Wang (2007). He assumes that each job is negatively effected by $w_1(t)$ in its start time t and positively effected by a weighted learning effect:

$$p_{ir} = p_i(w_1(t) + w_2 r^a). \tag{10}$$

He concentrated on single-machine scheduling problems and was able to prove that $1/p_{ir} = p_i(w_1(t) + w_2 r^a)/C_{\max}$, $1/p_{ir} = p_i(w_1(t) + w_2 r^a)/\sum C_i$, and $1/p_{ir} = p_i(w_1(t) + w_2 r^a)/\sum C_i^2$ can be solved by the SPT sequence.

$1/p_{ir} = p_i(w_1(t) + w_2 r^a)/\sum w_i C_i$ with agreeable weights and $1/p_{ir} = p_i(w_1(t) + w_2 r^a), p_i = p/\sum w_i C_i$ can be solved by the WSPT sequence and $1/p_{ir} = p_i(w_1(t) + w_2 r^a)/T_{\max}$ with agreeable due dates and $1/p_{ir} = p_i(w_1(t) + w_2 r^a), p_i = p/T_{\max}$ can be solved by the EDD sequence.

The models on learning and deteriorating jobs (8)–(10) have in common that two offsetting effects are considered simultaneously. Even if one is position-based and the other one is total-processing-time-based, a more practical approach might be to offset them and to use the net effect in the models.

4.4. Position-based linear learning functions

Still position-based but structurally different are the learning functions that have been suggested by Cheng and Wang (2000) for the first time. The processing time of a job scheduled on position r depends on its normal processing time p_i and a learning coefficient v_i that is multiplied by the number of jobs already processed:

$$p_{ir} = p_i - v_i \cdot \min\{r-1, n_{0i}\}. \tag{11}$$

$n_{0i} \leqslant n-1$ indicates a threshold level at which the learning of job i plateaus and obviously $v_i < p_i/n_{0i}$ is required to guarantee positive processing times. Cheng and Wang (2000) were able to prove that the single-machine maximum lateness problem, $1/p_{ir} = p_i - v_i \cdot \min\{r-1, n_{0i}\}/L_{\max}$ is NP-hard in the strong sense. If all jobs have the same learning coefficient and the same plateauing threshold, $1/p_{ir} = p_i - v \cdot \min\{r-1, n_0\}/L_{\max}$ can be solved by ordering the jobs according to the EDD sequence. And if all jobs have a common due date, the problem $1/p_{ir} = p_i - v_i \cdot \min\{r-1, n_{0i}\}, d_i = d/L_{\max}$ can be solved via an assignment problem formulation.

A structural similar but 'simpler' learning effect formulation has been suggested by Bachman and Janiak (2004):

$$p_{ir} = p_i - v_i \cdot r. \tag{12}$$

In this case $v_i < p_i/n$ is required to guarantee positive processing times. The authors showed that $1/p_{ir} = p_i - v_i \cdot r/C_{\max}$ can be solved by ordering the jobs in non-decreasing order of the learning coefficients v_i. The very same problem with ready times, $1/r_i, p_{ir} = p_i - v_i \cdot r/C_{\max}$ is NP-hard in the strong sense, but the special case with identical learning coefficients, $1/r_i, p_{ir} = p_i - v \cdot r/C_{\max}$, can be solved by sequencing the jobs in non-decreasing order of job ready-times. And $1/p_{ir} = p_i - v_i \cdot r/\sum C_i$ can be solved by formulating it as an assignment problem.

Again slightly different but structurally similar is the learning effect introduced by Wang and Xia (2005) for multiple-machine scheduling:

$$p_{ijr} = p_{ij}(w - v \cdot r). \tag{13}$$

With $v < w/n$ to guarantee positive processing times, Wang and Xia (2005) first concentrated on the single-machine case: The SPT sequence generates an optimal solution for $1/p_{ir} = p_i(w - v \cdot r)/\sum C_i$ and $1/p_{ir} = p_i(w - v \cdot r)/C_{\max}$. Johnson's rule for two-machine flowshop scheduling does not guarantee optimal schedules for $F2/p_{ijr} = p_{ij}(w - v \cdot r)/C_{\max}$. For the general flowshop problems $Fm/p_{ijr} = p_{ij}(w - v \cdot r)/\sum C_i$ and $Fm/p_{ijr} = p_{ij}(w - v \cdot r)/C_{\max}$ the SPT schedule has a worst-case performance ratio of m, and this bound is tight. For the special case that the machines form an increasing series of dominating machines, Wang and Xia (2005) were able to prove that the optimal solution for $Fm/p_{ijr} = p_{ij}(w - v \cdot r)$, idm$/\sum C_i$ and $Fm/p_{ijr} = p_{ij}(w - v \cdot r)$, idm$/C_{\max}$ is a permutation flowshop with the jobs on the positions $2, \ldots, n$ on machine m ordered according to the SPT sequence.

The position-based linear learning functions (11)–(13) have in common that some mathematical limitations on the learning coefficient are necessary. These limitations usually take into consideration the number of jobs n. From a learning perspective it is questionable why the number of jobs might impact the learning coefficient and why a learning coefficient that fits for, say, 10 jobs might not be appropriate for 20 jobs.

The results on position-based learning effects are summarized in Appendix A. The similarity of some of the results stems from the fact that all position-based learning effects have in common that the processing time of a job is decreasing in r, i.e. $p_{i1} \geqslant p_{i2} \geqslant \cdots \geqslant p_{in}$ for $i = 1, \ldots, n$.

5. Sum-of-processing-time based learning effects

Position-based learning effects neglect the processing times of the jobs already produced. If human interactions have a significant impact during the processing of the job, the processing time will add to the workers experience and cause learning effects. For situations like this it might be more appropriate to consider a time-dependent learning effect. Kuo and Yang (2006a) suggested to model learning as follows:

$$p_{ir} = p_i(1 + p_{[1]} + p_{[2]} + \cdots + p_{[r-1]})^a = p_i\left(1 + \sum_{k=1}^{r-1} p_{[k]}\right)^a, \tag{14}$$

$a \leqslant 0$ is the learning index. The authors state that "the number 1 ... is the modifying term to guarantee it is a learning effect". At least the number 1 guarantees that the processing time of the first job is equal to its normal processing time. Kuo and Yang (2006a) were able to show that the SPT sequence still leads to an optimal schedule if the goal is to minimize total completion time on a single-machine, i.e. for $1/p_{ir} = p_i\left(1 + \sum_{k=1}^{r-1} p_{[k]}\right)^a / \sum C_i$. Even though more complicated, the proof again is based on the pairwise job interchange technique. Kuo and Yang (2006b) concentrated on single-machine group scheduling problems. Between two consecutive jobs of the same group no setup is necessary, however, a sequence-independent group setup, s_g, is necessary before the first job of a group g can be started. Each group has its own learning index a_g. Similar to (14) the learning effect is $p_{igr} = p_{ig}(1 + p_{[1]g} + p_{[2]g} + \cdots + p_{[r-1]g})^{a_g}$ with p_{igr} being the processing time of job i of group g if scheduled in position r. p_{ig} and $p_{[i]g}$ are the normal processing time of job i and of the ith job of group g, respectively. As in Kuo and Yang (2006b) let G denote that the problem is a group scheduling problem and let S denote the existence of sequence-independent group setup times. The authors could show that for the makespan minimization problem $1/G, S, p_{igr} = p_{ig}\left(1 + \sum_{i=1}^{r-1} p_{[i]g}\right)^{a_g} / C_{\max}$ in an optimal schedule the jobs within a group are ordered according to the SPT rule and the groups can be ordered arbitrarily. As the setup times are sequence independent and learning starts for each group all over again, the order of the group cannot have an impact on the optimal solution. Hence $1/p_{ir} = p_i\left(1 + \sum_{k=1}^{r-1} p_{[k]}\right)^a / C_{\max}$ is optimized by SPT, too. Kuo and Yang (2006b) furthermore worked on the total completion time problem $1/G, S, p_{igr} = p_{ig}\left(1 + \sum_{i=1}^{r-1} p_{[i]g}\right)^{a_g} / \sum C_i$. The optimal schedule consists of two requirements: The jobs within each group are ordered according to the SPT sequence (see Kuo and Yang, 2006a) and the groups are ordered in the non-decreasing order of $\left(s_g + \sum_{i=1}^{n_g} p_{ig}^A\right)/n_g$. The superscript A indicates that the processing time is the actual and not the normal processing time, i.e. $p_{ig}^A = p_{igr}$ if job i of group g is scheduled in position r. Kuo and Yang (2006c) slightly altered the learning effect (14) by dropping the number one. They now assumed that

$$p_{i1} = p_i \quad \text{and} \quad p_{ir} = p_i(p_{[1]} + p_{[2]} + \ldots + p_{[r-1]})^a = p_i\left(\sum_{k=1}^{r-1} p_{[k]}\right)^a \quad \text{for } r \geqslant 2. \tag{15}$$

They worked on the single-machine makespan minimization problem and were able to show that in an optimal sequence for $1/p_{i1} = p_i$ and $p_{ir} = p_i\left(\sum_{k=1}^{r-1} p_{[k]}\right)^a$ for $r \geqslant 2/C_{\max}$ the jobs on the positions $2, \ldots, n$ are ordered according to the SPT sequence. The learning effects used by Kuo and Yang (2006a,b,c) have one common shortcoming: Their assumption is that the actual processing time of, say, the rth job depends on the sum of the *normal* processing times of the preceding $r - 1$ jobs. However, with the exception of the first job, all other jobs need less than their normal processing time. The base for calculating the learning experience is thus higher than the actual processing time that is available for learning. Yang and Kuo (in press) overcame this shortcoming by introducing a learning effect that is based on the sum of actual processing times:

$$p_{ir} = p_i(1 + p^A_{[1]} + p^A_{[2]} + \cdots + p^A_{[r-1]})^a = p_i\left(1 + \sum_{k=1}^{r-1} p^A_{[k]}\right)^a. \tag{16}$$

The authors decided to tackle completion-time-based-objectives: $1/p_{ir} = p_i\left(1 + \sum_{k=1}^{r-1} p^A_{[k]}\right)^a / C_{\max}$ and $1/p_{ir} = p_i\left(1 + \sum_{k=1}^{r-1} p^A_{[k]}\right)^a / \sum C_i^c$ for any $c > 0$ are again optimized by the SPT sequence. For $1/p_{ir} = p_i\left(1 + \sum_{k=1}^{r-1} p^A_{[k]}\right)^a / \sum w_i C_i$ if the jobs have agreeable weights, an optimal sequence is given by the WSPT rule (i.e. in non-decreasing order of p_i/w_i).

A different sum-of-processing time based learning effect has been proposed by Koulamas and Kyparisis (2007) recently:

$$p_{ir} = p_i\left(1 - \frac{\sum_{i=1}^{r-1} p_{[i]}}{\sum_{i=1}^{n} p_{[i]}}\right)^b = p_i\left(\frac{\sum_{i=r}^{n} p_{[i]}}{\sum_{i=1}^{n} p_{[i]}}\right)^b. \tag{17}$$

The authors state that $b \geqslant 1$ should hold. They were able to show that for the two single-machine problems $1/p_{ir} = p_i\left(\sum_{i=r}^{n} p_{[i]} / \sum_{i=1}^{n} p_{[i]}\right)^b / \sum C_i$ and $1/p_{ir} = p_i\left(\sum_{i=r}^{n} p_{[i]} / \sum_{i=1}^{n} p_{[i]}\right)^b / C_{\max}$ and for the two-machine flow-shop problems $F2/p_{ijr} = p_{ij}\left(\sum_{i=r}^{n} p_{[i]j} / \sum_{i=1}^{n} p_{[i]j}\right)^b$, prp/$\sum C_i$ and $F2/p_{ijr} = p_{ij}\left(\sum_{i=r}^{n} p_{[i]j} / \sum_{i=1}^{n} p_{[i]j}\right)^b$, prp/$C_{\max}$ the SPT sequence delivers an optimal schedule. From a learning perspective, Eq. (17) does not seem to be without problems. Let us assume $n = 10$ jobs each with $p_i = 1$ are to be produced and $b = 1$. The processing time of the second job would be 0.9. However, if there were $n = 5$ or $n = 20$ jobs, the processing time of the second job would now be 0.8 or 0.95, respectively, even though the learning experience is in all three cases exactly the same. It seems to be questionable that the learning effect strongly depends on the processing time of future jobs and only partly on the experience from already finished jobs. Furthermore $b \geqslant 1$ leads in our example to $p_{[10]} \leqslant 0.1$, i.e. to a 10th of its normal processing time. The gains from learning are forced to be extremely high by (17), maybe $b \geqslant 0$ would be more appropriate.

A summary on the results on sum-of-processing-time-based learning effects can be found in Appendix B. All sum-of-processing-time based learning effects have in common that the actual processing time of a job cannot be calculated without the knowledge of the normal or actual processing times of the preceding jobs. This makes the analysis harder than that for position-based learning effects, where the actual processing time of a job usually can be calculated as soon as it is know on which position it will be scheduled. However, it is rather obvious that for position-based as well as sum-of-processing-time based learning effects the well-known SPT sequence yields strong results for completion time goals.

6. Summary

This paper gives an overview on the work on the relatively young but very vivid area of scheduling with learning effects. We tried to clarify some of the economic fundamentals of scheduling and learning and afterwards sorted the literature by the different learning effects that have been suggested so far. As production environments will keep changing constantly, we are convinced that scheduling and learning is not a pure theoretical chimera but will have a practical impact in future scheduling decisions. However, to reach this goal we researchers need to model learning effects as realistic as possible – while keeping in mind that each model is only a (more or less good) approximation of the reality.

Acknowledgement

The author would like to thank the referees for their constructive criticism on an earlier version of this review.

Appendix A. Summary on scheduling with position-based learning effects

Problem	Complexity	Solution algorithm	Reference
$p_{ir} = p_i r^a$			
$1/p_{ir} = p_i r^a / \sum C_i$	$O(n \log n)$	SPT	Biskup (1999)
$1/p_{ir} = p_i r^a / C_{\max}$	$O(n \log n)$	SPT	Mosheiov (2001a)
$1/p_{ir} = p_i r^a, d_i = d / \sum (w_1 E_i + w_2 T_i + w_3 C_i)$	$O(n^3)$	Ass	Biskup (1999)
$1/p_{ir} = p_i r^a, d_i = d / \sum (w_1 E_i + w_2 T_i + w_3 d)$	$O(n^3)$	Ass	Mosheiov (2001a)
$1/p_{ir} = p_i r^a / w \sum C_i + (1-w) \sum \sum \lvert C_i - C_k \rvert$	$O(n^3)$	Ass	Mosheiov (2001a)
$1/p_{ir} = p_i r^a / w \sum C_i + (1-w) T_{\max}$		B&B	Lee et al. (2004)
$1/p_{ir} = p_i r^a, p_i = p / \sum w_i C_i$	$O(n \log n)$	Non-increasing w_i	Bachman and Janiak (2004)
$1/p_{ir} = p_i r^a / \sum w p_i C_i$	$O(n \log n)$	SPT	Bachman and Janiak (2004)
$1/p_{ir} = p_i r^a / \sum w_i C_i$ with agreeable weights	$O(n \log n)$	WSPT	Zhao et al. (2004)
$1/p_{ir} = p_i r^a / T_{\max}$ with agreeable due dates	$O(n \log n)$	EDD	Zhao et al. (2004)
$1/p_{ir} = p_i r^a / L_{\max}$ with agreeable due dates	$O(n \log n)$	EDD	Wu et al. (2007a,b)
$1/p_{ir} = p_i r^a / \sum T_i$	NP-hard	B&B	Eren and Guner (2007)
$1/p_{ir} = p_i r^a, \text{psd} / \sum C_i$	$O(n \log n)$	SPT	Kuo and Yang (2007)
$1/p_{ir} = p_i r^a, \text{psd} / C_{\max}$	$O(n \log n)$	SPT	Kuo and Yang (2007)
$1/p_{ir} = p_i r^a, \text{psd} / \sum \sum \lvert C_i - C_k \rvert$	$O(n \log n)$	Matching	Kuo and Yang (2007)
$1/p_{ir} = p_i r^a, \text{psd} / \sum (w_1 E_i + w_2 T_i + w_3 d)$	$O(n \log n)$	Matching	Kuo and Yang (2007)
$1/r_i, p_{ir} = p_i r^a / C_{\max}$	NP-hard		Bachman and Janiak (2004)
$Pm/p_{ir} = p_i r^a / \sum C_i$	$O((2n)^{m+3}/m!)$	m. Ass	Mosheiov (2001b)
$Qm/p_{ir} = p_i r^a, p_i = 1 / \sum w_i C_i$	$O(n \log n)$	WSPT	Zhao et al. (2004)
$Qm/p_{ir} = p_i r^a, p_i = 1 / L_{\max}$	$O(n \log n)$	EDD	Zhao et al. (2004)
$F2/p_{ijr} = p_{ij} r^a / \sum C_i$	NP-hard	B&B	Lee and Wu (2004)
$F2/p_{ijr} = p_{ij} r^a / w \sum C_i + (1-w) T_{\max}$	NP-hard	B&B	Chen et al. (2006)
$F2/p_{ijr} = p_{ij} r^a, p_{2j} = p_2 / \sum C_i$	$O(n \log n)$	SPT	Zhao et al. (2004)
$F2/p_{ijr} = p_{ij} r^a, p_{2j} = p_2 / C_{\max}$	$O(n \log n)$	SPT	Zhao et al. (2004)
$F2/p_{ijr} = p_{ij} r^a, \text{ord} / \sum C_i$	$O(n \log n)$	SPT	Koulamas and Kyparisis (2007)
$F2/p_{ijr} = p_{ij} r^a, \text{prp} / \sum C_i$	$O(n \log n)$	SPT	Koulamas and Kyparisis (2007)
$F2/p_{ijr} = p_{ij} r^a, \text{ord} / C_{\max}$	$O(n \log n)$	SPT	Koulamas and Kyparisis (2007)
$F2/p_{ijr} = p_{ij} r^a, \text{prp} / C_{\max}$	$O(n \log n)$	SPT	Koulamas and Kyparisis (2007)
$Fm/p_{ijr} = p_{ij} r^a, \text{idm} / \sum C_i$	$O(n^2 \log n)$	Algo	Zhao et al. (2004)
$Fm/p_{ijr} = p_{ij} r^a, \text{idm} / C_{\max}$	$O(n^2 \log n)$	Algo	Zhao et al. (2004)
$p_{ir} = p_i r^{a_i}$			
$1/p_{ir} = p_i r^{a_i} / C_{\max}$	$O(n^3)$	Ass	Mosheiov and Sidney (2003)
$1/p_{ir} = p_i r^{a_i} / \sum C_i$	$O(n^3)$	Ass	Mosheiov and Sidney (2003)

(*continued on next page*)

Appendix A. (*continued*)

Problem	Complexity	Solution algorithm	Reference
$1/p_{ir} = p_i r^{a_i}, d_i = d/\sum(w_1E_i + w_2T_i + w_3d)$	$O(n^3)$	Ass	Mosheiov and Sidney (2003)
$1/p_{ir} = p_i r^{a_i}, d_i = d/\sum U_i$	$O(n^3 \log n)$	s. Ass	Mosheiov and Sidney (2005)
$1/p_{ir} = p_i r^{a_i}/\sum U_i$	NP-hard		Lin (in press)
$Qm/p_{ir} = p_i r^{a_i}/\sum C_i$	$O((2n)^{m+3}/m!)$	m. Ass	Mosheiov and Sidney (2003)
$p_{ir} = p_i r^{\log_2(1-x)\mathrm{LR}}$			
$1/p_{ir} = p_i r^{\log_2(1-x)\mathrm{LR}}, d_i = d/\sum(E_i + T_i) + k(x)$	$O(n^3)$	Algo	Biskup and Simons (2004)
$p_{ir} = (p_i + wt)r^a$			
$1/p_{ir} = (p_i + wt)r^a/C_{\max}$	$O(n \log n)$	SPT	Wang (2006)
$1/p_{ir} = (p_i + wt)r^a/\sum C_i$	$O(n \log n)$	SPT	Wang (2006)
$1/p_{ir} = (p_i + wt)r^a, p_i = cw_i/\sum w_iC_i$	$O(n \log n)$	SPT	Wang (2006)
$1/p_{ir} = (p_i + wt)r^a/\sum w_iC_i$ with agreeable weights	$O(n \log n)$	WSPT	Wang (2006)
$1/p_{ir} = (p_i + wt)r^a, p_i = p/\sum w_iC_i$	$O(n \log n)$	WSPT	Wang (2006)
$1/p_{ir} = (p_i + wt)r^a/L_{\max}$ with agreeable due dates	$O(n \log n)$	EDD	Wang (2006)
$1/p_{ir} = (p_i + wt)r^a, p_i = p/L_{\max}$	$O(n \log n)$	EDD	Wang (2006)
$F2/p_{ijr} = (p_{ij} + wt)r^a, p_{2j} = p_2/\sum C_i$	$O(n \log n)$	SPT	Wang (2006)
$F2/p_{ijr} = (p_{ij} + wt)r^a, p_{2j} = p_2/C_{\max}$	$O(n \log n)$	SPT	Wang (2006)
$Fm/p_{ijr} = (p_{ij} + wt)r^a, \mathrm{idm}/\sum C_i$	$O(n^2 \log n)$	Algo	Wang (2006)
$Fm/p_{ijr} = (p_{ij} + wt)r^a, \mathrm{idm}/C_{\max}$	$O(n^2 \log n)$	Algo	Wang (2006)
$p_{ir} = p_i(w_1(t) + w_2r^a)$			
$1/p_{ir} = p_i(w_1(t) + w_2r^a)/C_{\max}$	$O(n \log n)$	SPT	Wang (2007)
$1/p_{ir} = p_i(w_1(t) + w_2r^a)/\sum C_i$	$O(n \log n)$	SPT	Wang (2007)
$1/p_{ir} = p_i(w_1(t) + w_2r^a)/\sum C_i^2$	$O(n \log n)$	SPT	Wang (2007)
$1/p_{ir} = p_i(w_1(t) + w_2r^a)/\sum w_iC_i$ with agreeable weights	$O(n \log n)$	WSPT	Wang (2007)
$1/p_{ir} = p_i(w_1(t) + w_2r^a), p_i = p/\sum w_iC_i$	$O(n \log n)$	WSPT	Wang (2007)
$1/p_{ir} = p_i(w_1(t) + w_2r^a)/T_{\max}$ with agreeable weights	$O(n \log n)$	EDD	Wang (2007)
$1/p_{ir} = p_i(w_1(t) + w_2r^a), p_i = p/T_{\max}$	$O(n \log n)$	EDD	Wang (2007)
$p_{ir} = p_i - v_i \cdot \min\{r - 1, n_{0i}\}$			
$1/p_{ir} = p_i - v_i \cdot \min\{r - 1, n_{0i}\}/L_{\max}$	NP-hard		Cheng and Wang (2000)
$1/p_{ir} = p_i - v \cdot \min\{r - 1, n_0\}/L_{\max}$	$O(n \log n)$	EDD	Cheng and Wang (2000)
$1/p_{ir} = p_i - v_i \cdot \min\{r - 1, n_{0i}\}, d_i = d/L_{\max}$	$O(n^3)$	Ass	Cheng and Wang (2000)
$p_{ir} = p_i - v_i \cdot r$			
$1/p_{ir} = p_i - v_i \cdot r/C_{\max}$	$O(n \log n)$	Non-decreasing v_i	Bachman and Janiak (2004)

Appendix A. (*continued*)

Problem	Complexity	Solution algorithm	Reference
$1/r_i, p_{ir} = p_i - v_i \cdot r/C_{\max}$	NP-hard		Bachman and Janiak (2004)
$1/r_i, p_{ir} = p_i - v \cdot r/C_{\max}$	$O(n \log n)$	Non-decreasing r_i	Bachman and Janiak (2004)
$1/p_{ir} = p_i - v_i \cdot r/\sum C_i$	$O(n^3)$	Ass	Bachman and Janiak (2004)
$p_{ijr} = p_{ij}(w - v \cdot r)$			
$1/p_{ir} = p_i(w - v \cdot r)/\sum C_i$	$O(n \log n)$	SPT	Wang and Xia (2005)
$1/p_{ir} = p_i(w - v \cdot r)/C_{\max}$	$O(n \log n)$	SPT	Wang and Xia (2005)
$Fm/p_{ijr} = p_{ij}(w - v \cdot r)$, idm/$\sum C_i$	$O(n^2 \log n)$	Algo	Wang and Xia (2005)
$Fm/p_{ijr} = p_{ij}(w - v \cdot r)$, idm/$C_{\max}$	$O(n^2 \log n)$	Algo	Wang and Xia (2005)

Appendix B. Summary on scheduling with sum-of-processing-time based learning effects

Problem	Complexity	Solution algorithm	Reference
$p_{ir} = p_i\left(1 + \sum_{k=1}^{r-1} p_{[k]}\right)^a$			
$1/p_{ir} = p_i\left(1 + \sum_{k=1}^{r-1} p_{[k]}\right)^a/\sum C_i$	$O(n \log n)$	SPT	Kuo and Yang (2006a)
$1/p_{ir} = p_i\left(1 + \sum_{k=1}^{r-1} p_{[k]}\right)^a/C_{\max}$	$O(n \log n)$	SPT	Kuo and Yang (2006a,b)
$p_{igr} = p_{ig}\left(1 + \sum_{i=1}^{r-1} p_{[i]g}\right)^{a_g}$			
$1/G, S, p_{igr} = p_{ig}\left(1 + \sum_{i=1}^{r-1} p_{[i]g}\right)^{a_g}/C_{\max}$	$O(n \log n)$	Jobs: SPT, groups: arbitrarily	Kuo and Yang (2006b)
$1/G, S, p_{igr} = p_{ig}\left(1 + \sum_{i=1}^{r-1} p_{[i]g}\right)^{a_g}/\sum C_i$	$O(n \log n)$	Jobs: SPT, groups: non-decreasing $\left(s_g + \sum_{i=1}^{n_g} p_{ig}^A\right)/n_g$	Kuo and Yang (2006b)
$p_{i1} = p_i$ and $p_{ir} = p_i\left(\sum_{k=1}^{r-1} p_{[k]}\right)^a$ for $r \geqslant 2$			
$1/p_{i1} = p_i$ and $p_{ir} = p_i\left(\sum_{k=1}^{r-1} p_{[k]}\right)^a$ for $r \geqslant 2/C_{\max}$	$O(n^2 \log n)$	Algo	Kuo and Yang (2006c)
$p_{ir} = p_i\left(1 + \sum_{k=1}^{r-1} p_{[k]}^A\right)^a$			
$1/p_{ir} = p_i\left(1 + \sum_{k=1}^{r-1} p_{[k]}^A\right)^a/C_{\max}$	$O(n \log n)$	SPT	Yang et al. (in press)
$1/p_{ir} = p_i\left(1 + \sum_{k=1}^{r-1} p_{[k]}^A\right)^a/\sum C_i^c$ for any $c > 0$	$O(n \log n)$	SPT	Yang et al. (in press)
$1/p_{ir} = p_i\left(1 + \sum_{k=1}^{r-1} p_{[k]}^A\right)^a/\sum w_i C_i$ with agreeable weights	$O(n \log n)$	WSPT	Yang et al. (in press)

(*continued on next page*)

Appendix B. (*continued*)

Problem	Complexity	Solution algorithm	Reference
$p_{ir} = p_i\left(\sum_{i=r}^{n} p_{[i]} / \sum_{i=1}^{n} p_{[i]}\right)^b$			
$1/p_{ir} = p_i\left(\sum_{i=r}^{n} p_{[i]} / \sum_{i=1}^{n} p_{[i]}\right)^b / \sum C_i$	$O(n \log n)$	SPT	Koulamas and Kyparisis (2007)
$1/p_{ir} = p_i\left(\sum_{i=r}^{n} p_{[i]} / \sum_{i=1}^{n} p_{[i]}\right)^b / C_{\max}$	$O(n \log n)$	SPT	Koulamas and Kyparisis (2007)
$F2/p_{ijr} = p_{ij}\left(\sum_{i=r}^{n} p_{[i]j} / \sum_{i=1}^{n} p_{[i]j}\right)^b, \mathrm{prp} / \sum C_i$	$O(n \log n)$	SPT	Koulamas and Kyparisis (2007)
$F2/p_{ijr} = p_{ij}\left(\sum_{i=r}^{n} p_{[i]j} / \sum_{i=1}^{n} p_{[i]j}\right)^b, \mathrm{prp} / C_{\max}$	$O(n \log n)$	SPT	Koulamas and Kyparisis (2007)

References

Adler, P.S., Clark, K.B., 1991. Behind the learning curve: A sketch of the learning process. Management Science 37, 267–281.

Bachman, A., Janiak, A., 2004. Scheduling jobs with position-dependent processing times. Journal of the Operational Research Society 55, 257–264.

Biskup, D., 1999. Single-machine scheduling with learning considerations. European Journal of Operational Research 115, 173–178.

Biskup, D., Simons, D., 2004. Common due date scheduling with autonomous and induced learning. European Journal of Operational Research 159, 606–616.

Chen, P., Wu, C.-C., Lee, W.-C., 2006. A bi-criteria two-machine flowshop scheduling problem with a learning effect. Journal of the Operational Research Society 57, 1113–1125.

Cheng, T.C.E., Wang, G., 2000. Single machine scheduling with learning effect considerations. Annals of Operations Research 98, 273–290.

Cheng, M.-B., Sun, S.-J., Yu, Y., 2007. A note on flow shop scheduling problems with a learning effect on no-idle dominant machines. Applied Mathematics and Computation 184, 945–949.

Cochran, E.B., 1960. New concepts of the learning curve. The Journal of Industrial Engineering 11, 317–327.

Conway, R.W., Schultz, A., 1959. The manufacturing progress function. The Journal of Industrial Engineering 10, 39–54.

Day, G.S., Montgomery, D.B., 1983. Diagnosing the experience curve. Journal of Marketing 47, 44–58.

Du, J., Leung, J.Y.-T., 1990. Minimizing total tardiness on one machine is NP-hard. Mathematics of Operations Research 15, 483–495.

Dutton, J.M., Thomas, A., 1984. Treating progress functions as a managerial opportunity. Academy of Management Review 9, 235–247.

Eren, T., Guner, E., 2007. Minimizing total tardiness in a scheduling problem with a learning effect. Applied Mathematical Modelling 31, 1351–1361.

Ghemawat, P., 1985. Building strategy on the experience curve – a venerable management tool remains valuable – in the right circumstances. Harvard Business Review 63 II, 143–149.

Gonzalez, T., Sahni, S., 1978. Flowshop and jobshop schedules: Complexity and approximation. Operations Research 26, 36–52.

Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., 1979. Optimization and approximation in deterministic sequencing and scheduling: A survey. Annals of Discrete Mathematics 5, 287–326.

Hax, A.C., Majluf, N.S., 1982. Competitive cost dynamics: The experience curve. Interfaces 12, 50–61.

Keachie, E.C., Fontana, R.J., 1966. Effects of learning on optimal lot size. Management Science 13, B102–B108.

Koulamas, C., Kyparisis, G.J., 2007. Single-machine and two-machine flowshop scheduling with general learning function. European Journal of Operational Research 178, 402–407.

Koulamas, C., Kyparisis, G.J., 2008. Single-machine scheduling with past-sequence-dependent setup times. European Journal of Operational Research 187, 68–72.

Kuo, W.-H., Yang, D.-L., 2006a. Minimizing the total completion time in a single-machine scheduling problem with a time-dependent learning effect. European Journal of Operational Research 174, 1184–1190.

Kuo, W.-H., Yang, D.-L., 2006b. Single-machine group scheduling with a time-dependent learning effect. Computers and Operations Research 33, 2099–2112.

Kuo, W.-H., Yang, D.-L., 2006c. Minimizing the makespan in a single machine scheduling problem with a time-based learning effect. Information Processing Letters 97, 64–67.

Kuo, W.-H., Yang, D.-L., 2007. Single machine scheduling with past-sequence-dependent setup times and learning effects. Information Processing Letters 102, 22–26.

Lapré, M.A., Van Wassenhove, L.N., 2001. Creating and transferring knowledge for productivity improvement in factories. Management Science 47, 1311–1325.

Lapré, M.A., Mukkherjee, A.S., Van Wassenhove, L.N., 2000. Behind the learning curve: Linking learning activities to waste reduction. Management Science 46, 597–611.

Lee, W.-C., 2004. A note on deteriorating jobs and learning in single-machine scheduling problems. International Journal of Business and Economics 3, 83–89.
Lee, W.-C., Wu, C.-C., 2004. Minimizing total completion time in a two-machine flowshop with a learning effect. International Journal of Production Economics 88, 85–93.
Lee, W.-C., Wu, C.-C., Sung, H.-J., 2004. A bi-criterion single-machine scheduling problem with learning considerations. Acta Informatica 40, 303–315.
Li, C.-L., Cheng, T.C.E., 1994. An economic production quantity model with learning and forgetting considerations. Production and Operations Management 3, 118–132.
Lin, B.M.T. Complexity results for single-machine scheduling with positional learning effects. Journal of the Operational Research Society, in press.
Mosheiov, G., 2001a. Scheduling problems with a learning effect. European Journal of Operational Research 132, 687–693.
Mosheiov, G., 2001b. Parallel machine scheduling with a learning effect. Journal of the Operational Research Society 52, 1–5.
Mosheiov, G., Sidney, J.B., 2003. Scheduling with general job-dependent learning curves. European Journal of Operational Research 147, 665–670.
Mosheiov, G., Sidney, J.B., 2005. Note on scheduling with general learning curves to minimize number of tardy jobs. Journal of the Operational Research Society 56, 110–112.
Rohrbach, A., 1927. Economical production of all-metal airplanes and seaplanes. Journal of the Society of Automotive Engineers 20, 57–66.
Sterman, J.D., 1994. Learning in and about complex systems. System Dynamics Review 10, 291–330.
Thurstone, L.L., 1919. The learning curve equation. Psychological Monographs 26, 1–51.
Upton, D.M., Kim, B., 1998. Alternative methods of learning and process improvement in manufacturing. Journal of Operations Management 16, 1–20.
Venezia, I., 1985. On the statistical origins of the learning curve. European Journal of Operational Research 19, 191–200.
Wang, J.-B., 2006. A note on scheduling problems with learning effects and deteriorating jobs. International Journal of Systems Science 37, 827–833.
Wang, J.-B., 2007. Single-machine scheduling problems with the effects of learning and deterioration. Omega 35, 397–402.
Wang, J.-B., Xia, Z.-Q., 2005. Flow-shop scheduling with a learning effect. Journal of the Operational Research Society 56, 1325–1330.
Wang, X., Cheng, T.C.E., 2007. Single-machine scheduling with deteriorating jobs and learning effects to minimize the makespan. European Journal of Operational Research 178, 57–70.
Webb, G.K., 1994. Integrated circuit (IC) pricing. High Technology Management Research 5 (II), 247–260.
Wright, T.P., 1936. Factors affecting the cost of airplanes. Journal of Aeronautical Sciences 3, 122–128.
Wu, C.-C., Lee, W.-C., Chen, T., 2007a. Heuristic algorithms for solving the maximum lateness scheduling problem with learning considerations. Computers and Industrial Engineering 52, 124–132.
Wu, C.-C., Lee, W.-C., Wang, W.C., 2007b. A two-machine flowshop maximum tardiness scheduling problem with a learning effect. International Journal of Advanced Manufacturing Technology 31, 743–750.
Yang, D.-L., Kuo, W.-H. Single-machine scheduling with an actual time-dependent learning effect. Journal of the Operational Research Society, in press.
Yelle, L.E., 1979. The learning curve: Historical review and comprehensive survey. Decision Sciences 10, 302–328.
Zangwill, W.I., Kantor, P.B., 1998. Toward a theory of continuous improvement and the learning curve. Management Science 44, 910–920.
Zhao, C.-L., Zhang, Q.-L., Tang, H.-Y., 2004. Machine scheduling problems with learning effects. Dynamics of Continuous, Discrete and Impulsive Systems, Series A: Mathematical Analysis 11, 741–750.