Combining Time and Position Dependent Effects on a Single Machine Subject to Rate-Modifying Activities

Kabir Rustogi and Vitaly A. Strusevich
School of Computing and Mathematical Sciences, University of Greenwich,
Old Royal Naval College, Park Row, Greenwich, London SE10 9LS, U.K.
e-mail: {K.Rustogi,V.Strusevich}@greenwich.ac.uk

Abstract

We introduce a general model for single machine scheduling problems, in which the actual processing times of jobs are subject to a combination of positional and time-dependent effects, that are job-independent but additionally depend on certain activities that modify the processing rate of the machine, such as, e.g., maintenance. We focus on minimizing two classical objectives: the makespan and the sum of the completion times. The traditional classification accepted in this area of scheduling is based on the distinction between the learning and deterioration effects, on one hand, and between the positional effects and the start-time dependent effects, on the other hand. Our results show that in the framework of the introduced model such a classification is not necessary, as long as the effects are job-independent. The model introduced in this paper covers most previously known models. The solution algorithms are developed within the same general framework and their running times are no worse than those available earlier for problems with less general effects.

Keywords: Scheduling; Maintenance; Deterioration; Learning; Rate Modifying Activity

1 Introduction

In this paper, we revise and generalize the area of scheduling research related to various effects on the actual processing times of the jobs and to machine maintenance. This paper can be seen as a continuation of our recent publications [1] and [2]. In this introduction, we do not attempt to give a historical account of the area, which can be found in the review [2]. Instead, we describe the scope of this paper, supplying references for illustration only. We summarize, rather informally, several messages that this paper delivers, some of them being in conflict with the way the corresponding area of research has been developed so far.

A considerable amount of publications on various scheduling models in which the processing times are not constant but are subject to various effects have been generated since the early 1990s. Traditionally, two major types of effects have been analyzed: deterioration and learning. Under *deterioration*, the later a job starts, the longer it takes to process it; typically this is attributed to the fact that during processing, machine qualities may get worse, or a human operator may get tired. In the case of the opposite effect, *learning,* the later a job starts, the shorter it takes to process it, which, for example, happens when human operators improve their skills by gaining experience. To put it simpler, deterioration has a non-decreasing effect on the actual processing times, while learning has the opposite influence. Most of the prior research, while acknowledging certain similarities between these

effects, would often address them individually; see, e.g., a survey [3] on scheduling with learning alone or a paper [4] on a particular type of deterioration. Although different meaningful interpretations of the two effects justify the tradition of their individual treatment, mathematically they can be handled by a unified approach, at least under some additional realistic assumptions, such as job-independence. Hence, our first message is that there is no need to insist on a clear distinction between learning and deterioration.

> **Message 1:** Separate learning and deterioration effects can be combined into a general enhanced effect, which does not necessarily have to influence the actual processing times monotonically. Scheduling problems with these enhanced effects can still be solved in polynomial time.

There are several classes of effects that specify how exactly the actual processing time of a job is affected. Under a *positional* effect, the actual processing time of a job depends on a factor that is derived from the position of a job in the processing sequence. Under a *time-dependent* effect, the actual processing time of a job is a function of the start time of the job. Leaving aside less popular *cumulative* effects initiated by [5] and [6], the studies on the positional and the time-dependent effects have formed the bulk of publications in the area; see, e.g., a survey [7] and the book [8], the latter providing a comprehensive exposition on scheduling with time-dependent processing times. These two effects can be further classified as being either *job-dependent* or *job-independent*. A job-dependent effect implies that each job has a different (unique) effect on the state of the machine, while a job-independent effect implies the opposite.

Notice that the main focus of prior research on scheduling with time-dependent effects has been on linear models. Although more exotic models have also been considered (see, e.g., [8] and [9] for reviews), many authors pose questions regarding their practical relevance. Thus, in this paper we concentrate on linear time-dependent effects.

Recently, researches have started to combine positional effects and time-dependent effects into a single enhanced effect; see, e.g., [10], [11] and [12]. A review and further extensions of these models can be found in surveys [2] and [9]. In particular, it is shown in [2], that if the effects are job-independent, enhanced scheduling problems that combine a linear time-dependent effect and a fairly general positional effect, can be handled by the same mathematical tools as the problems with individual effects. Our second message is that under these assumptions there is no need to insist on a clear distinction between positional and time-dependent effects.

> **Message 2:** If the effects are job-independent, separate positional and (linear) time-dependent effects can be combined into a general enhanced effect. Scheduling problems with these enhanced effects can still be solved in polynomial time.

An issue of a practical relevance in scheduling with changing processing times is *machine maintenance*. In our paper [1], we present a brief account on scheduling with maintenance, stating that the most natural range of models that use machine maintenance to restore deteriorating processing conditions, fully or at least partially, have appeared only recently, see, e.g., [13] and [14]. The duration of a maintenance period can either be a given constant or be dependent on its start time. The latter type of maintenance is introduced in [15] and [16]; see also [17] and [18] for more recent developments. With several maintenance periods in a

schedule, the jobs are split into several *groups*, one group before the first maintenance and one after each maintenance period. An important enhancement of the concept of maintenance is done in [1], in which the authors do not assume that all maintenance periods are identical and allow each one of them to leave the processing conditions of the machine in a different state, not necessarily bringing them back to an initial perfect state. In this paper, we go further and deal with a more general concept of a *rate-modifying activity* performed on a machine, which is not limited to machine maintenance but may influence the processing of the jobs in the group that follows such an activity. Thus, the effects that change the actual processing time of a job may become additionally dependent on the group a job is sequenced in. We refer to such effects as *group-dependent* effects. As shown in [1], combining machine maintenance with group-dependent positional deterioration still allows solving the corresponding scheduling problem in polynomial time. Our third message addresses the main issue of this paper: combining the general machine maintenance, or rather, rate-modifying activities, with enhanced job-independent group-dependent effects.

> **Message 3:** Single machine problems that integrate rate-modifying activities of start-time dependent duration and enhanced job-independent group-dependent effects on the actual processing times can still be solved in polynomial time.

Our last message is of a methodological nature. As demonstrated in [2], in most of the earlier publications in the area of scheduling with job-independent effects, the search for a solution algorithm was limited either to the use of simple priority rules or to reduction of the problem at hand to a full form linear assignment problem. In [2], we review and improve most of the known results and establish that the adequate tool for handling the resulting problems, with or without maintenance, would be to reduce them to a simple matching problem, i.e., to the linear assignment problem with a product cost matrix. The latter problem is solvable by a fast algorithm due to [19].

> **Message 4:** Single machine problems that integrate rate-modifying activities of start-time dependent duration and enhanced job-independent group-dependent effects on the actual processing times can still be solved in polynomial time by reduction to a linear assignment problem with a product cost matrix.

The remainder of this paper is organized as follows. Section 2 gives a formal description of our main model and discusses its relations to those studied earlier. In Section 3, we derive the formulae for computing the completion time of an arbitrary job in a schedule. Section 4 explains how the problems of minimizing the makespan and the sum of the completion times can be reduced to a linear assignment problem with a product cost matrix. Section 5 outlines the solution approach that is used to solve the resulting problem. In Section 6, we discuss extensions of our model to parallel machines. On the other hand, in Section 7 we look at various simplified models that can be written as special cases of our enhanced model. It is demonstrated that the running times of the algorithms that we develop in this paper for a fairly general model cannot be improved for simpler effects, as long as a positional effect is combined with a time-dependent effect. On the other hand, the problems with either pure positional effects or pure time-dependent effects may admit faster solution algorithms, provided that the total flow time is minimized. The summary and the concluding remarks are contained in Section 8.

2 Model Description

The jobs of set $N = \{1, 2, \ldots, n\}$ have to be processed on a single machine. The jobs are available for processing at time zero and are independent, i.e., there are no precedence constraints and any processing sequence is feasible. At time zero, the machine is assumed to be in a perfect processing state, and its conditions change with the course of processing. These changes are formally captured by various job-independent factors as described in the rest of this section.

Each job $j \in N$ is associated with its normal processing time p_j; essentially p_j is the duration of the processing of job j, provided that job j is the first to be processed on the machine under perfect conditions. Given a feasible schedule S, the completion time of a job $j \in N$ is denoted by $C_j(S)$. In this paper, we focus on the two most popular measures of the quality of a schedule: the *makespan*, denoted by $C_{\max}(S) = \max\{C_j(S)|j \in N\}$ and the *total flow time*, denoted by $\sum_{j\in N} C_j(S)$. If it is clear from the content which schedule is used, we may drop the reference to S and simply write C_j, $C_{\max}$ or $\sum_{j\in N} C_j$, etc.

We start our presentation with a fairly simple situation, assuming that all jobs are scheduled in one group, i.e., no rate-modifying activities are involved. This allows us to gently introduce all required concepts and also to illustrate Messages 1 and 2 from the introduction.

In a linear time-dependent model initiated by [20], the actual processing time of job j is given by

$$p_j(t_j) = p_j + a_j t_j,$$

where t_j denotes the start time of job $j \in N$, and a_j represents a job-dependent rate. Another general model involves piecewise linear time-dependent effects [21].

Since in this paper we concentrate on job-independent effects, we focus on an effect given by

$$p_j(t_j) = p_j + at_j, \tag{1}$$

with a job-independent rate a. If $a > 0$, we have a deterioration effect and, if $a < 0$, we have a learning effect. For this model, polynomial-time algorithms for minimizing the makespan and for minimizing the total flow time follow from [20] and [22], respectively.

On the other hand, consider a job-independent positional effect, in which the actual processing time of job j is given by

$$p_j(r) = g(r)p_j,\ 1 \le r \le n,$$

where $g(r)$ represents a factor that depends on the position of a job in the processing sequence. If the values $g(r)$, $1 \le r \le n$, form a non-decreasing sequence, we deal with a positional deterioration effect; if the sequence is non-increasing, a learning effect is observed. In earlier papers, the focus has been on particular functions that define the positional factors $g(r)$, e.g., polynomial [23, 24] or exponential [25]. In what follows, it is assumed that the values $g(r)$, $1 \le r \le n$, are either given as an array of numbers or $g(r)$ can be computed for each r in constant time. As pointed out in [2], in order to obtain a polynomial-time algorithm for a problem with job-independent positional effects, there is no need to look at a particular function $g(r)$ or assume its monotone behavior, since for an arbitrary array of numbers $g(r)$, $1 \le r \le n$, most of the traditional objective functions can still be minimized without increasing the running time of the algorithms available for various special cases.

In the recent scheduling literature, several papers have appeared that combine a linear time-dependent effect with a positional effect, so that in the job-independent case the actual

processing time of job $j \in N$ is given by

$$p_j(t_j, r) = (p_j + at_j)\, g(r),\ 1 \leq r \leq n, \tag{2}$$

where r and t_j have the same meaning as above.

Until [2], most of the earlier papers on such combined effects have emphasized on a clear distinction between the models with learning and deterioration. For example, for a special form of the model described by (2) with a polynomial positional effect $g(r) = r^b$, the problems of minimizing the makespan and the total flow time for a model with $a > 0$ (time-dependent deterioration) and $b < 0$ (positional learning) are solved in [10] and [11], while a model with $a < 0$ (time-dependent learning) and $b > 0$ (positional deterioration) is considered in [12]. In [2] however, it is shown that there is no need for any assumptions on the monotone behavior of the positional factors and on the sign of the time-dependent rate a, so that the resulting model given by (2) covers many known models with job-independent effects and allows handling simultaneous learning and deterioration effects of both types (linear time-dependent and positional) by a unified efficient approach.

For instance, a linear time-dependent deterioration effect with a rate $d > 0$ and time-dependent learning effect with a rate $l < 0$ may be combined into a single effect of the form (1) with a rate defined as $a := d + l$. Similarly, a positional deterioration effect given by an array $g_d(1) \leq g_d(2) \leq \cdots \leq g_d(n)$ and a positional learning effect given by an array $g_l(1) \geq g_l(2) \geq \cdots \geq g_l(n)$ may be combined into a non-monotone positional effect defined by $g(r) := g_d(r)\, g_l(r)$, $1 \leq r \leq n$. Moreover, the two resulting effects may be put together to produce a combined effect of the form (2).

In this paper, we extend this general model by incorporating *rate-modifying periods* (*RMPs*) in the schedule. An RMP can be defined as an activity in a schedule which is responsible for changing the processing conditions of the system. If an RMP is aimed at improving the processing conditions of a system (such as maintaining or repairing the machine or its parts, letting a human operator to have rest, etc.), that was previously undergoing deterioration, then the actual processing times of the jobs scheduled after such an RMP typically get smaller. On the other hand, an RMP can also be associated with replacing a human operator by another one, so that all learning advantages of the previous employee are lost, and the overall productivity of the system decreases. Another form of RMP which is studied by [26], can be of the form in which the learning rate of a machine is further enhanced after an RMP.

In all prior publications on scheduling with changing processing times and rate-modifying activities, a clear distinction has been made between different kinds of RMPs. As a rule, only RMPs of a similar kind are allowed to be included in a schedule. Majority of the papers deal with problems in which an RMP is treated as a maintenance period and is used in models with a deterioration effect only. In this paper, however, we do not impose any restrictions on the RMPs. Each RMP can have a different effect on the machine conditions, so that they do not necessarily restore the machine to its default "as good as new" state.

In the model that serves as the basis of this paper, we consider a general situation, in which the decision-maker is presented with a total of $K \geq 0$ possible rate-modifying activities, which can be either distinct or alike. If out of the K available RMPs, $k-1$ of them are selected and included in a schedule, then the jobs will be divided into k, $1 \leq k \leq K+1$, groups, one to be scheduled before the first RMP and one after each of the $k-1$ RMPs. Since the RMPs can be different in nature, we need a model that is capable of handling situations, in which the effects are group-dependent, so that jobs sequenced in different groups are subject

to different effects. Group-dependent effects have been first introduced in [1], with the focus on single machine models with positional deterioration effects and the RMPs to be strictly maintenance periods, i.e., aimed at improving the processing conditions, but not necessarily fully restoring them. Extensions of the multi-maintenance model to parallel machines can be found in [2].

We proceed with a formal description of our main model. We assume that the actual processing time of a job is seen as affected by the following factors:

- the group a job is assigned to;
- the position of the job within its group;
- the number of jobs scheduled in each of the previous groups;
- the time elapsed before processing the job within its group;
- the time elapsed in each of the previous groups.

Additionally, the duration D_{RMP} of an RMP with an index y, $1 \leq y \leq K$, is given by

$$D_{RMP}(y) = \left(\alpha^{[y]}\tau + \psi^{[y]}\right) f^{[y]}(\eta), \tag{3}$$

where

- τ is the time since the last RMP was performed;
- $\alpha^{[y]}$ and $\psi^{[y]} > 0$ are given constants known for each available RMP, $1 \leq y \leq K$;
- the factor $f^{[y]}(\eta)$ represents a positional factor which makes the duration of the RMP dependent on the number of jobs η scheduled since the last RMP was performed.

The conceived model allows us to handle a wide range of practical problems, which have not been studied in the scheduling literature so far. Notice that the duration of an RMP defined by (3) is a further generalization of the start-time dependent model introduced in [15, 16], since it additionally includes a positional factor, so that the duration is dependent on the start-time of the RMP and also on the position of the RMP in the processing sequence. The latter is in line with positionally dependent setup times for models that arise in group technology scheduling; see [27]. As a result, according to the revised model the later an RMP is scheduled, both with respect to time and position, the longer/shorter it takes to complete it, depending on the sign of $\alpha^{[y]}$ and the nature of the function $f^{[y]}$.

Formally, consider a schedule $S(k)$ with $k-1$, $1 \leq k \leq K+1$, RMPs and a permutation of all jobs given by $\pi = \left(\pi^{[1]}, \pi^{[2]}, \ldots, \pi^{[k]}\right)$. Assume that each group contains a total of $n^{[x]}$ jobs, so that the permutation of jobs in the $x-$th group is given by $\pi^{[x]} = \left(\pi^{[x]}(1), \pi^{[x]}(2), \ldots, \pi^{[x]}\left(n^{[x]}\right)\right)$, $1 \leq x \leq k$, where $\sum_{x=1}^{k} n^{[x]} = n$. Depending on which RMPs are chosen and the order in which they are performed, the actual processing time of a job $j = \pi^{[x]}(r)$, scheduled in position r, $1 \leq r \leq n^{[x]}$, of the $x-$th group, $1 \leq x \leq k$, is given by

$$\begin{aligned} p_j^{[x]}(r) = \quad & \left(p_{\pi^{[x]}(r)} + a_1^{[x]}F_1 + a_2^{[x]}F_2 + \ldots + a_{x-1}^{[x]}F_{x-1} + a_x^{[x]}F_{(x,r-1)}\right) g^{[x]}(r), \\ & 1 \leq r \leq n^{[x]},\ 1 \leq x \leq k, \end{aligned} \tag{4}$$

where F_v denotes the total processing time of all jobs scheduled in a group v, $1 \leq v \leq x-1$, while $F_{(x,r-1)}$ represents the total duration of the first $r-1$ jobs in a group x, and for completeness we have the condition $F_0 = F_{(x,0)} = 0$. The terms $g^{[x]}(r)$ represent positive group-dependent job-independent positional factors, which do not have to be monotone within a particular group. Such a model for positional effects is introduced in [1] for a deterioration environment, in which the positional factors $g^{[x]}(r)$ are in a non-decreasing order in each group x. The terms $a_1^{[x]}, a_2^{[x]}, \ldots, a_x^{[x]}$ are real numbers and represent the group-dependent rates associated with time-dependent effects. Notice that some of the rates $a_1^{[x]}, a_2^{[x]}, \ldots, a_x^{[x]}$ may be negative, which corresponds to a learning effect. Without going into technical details, in what follows we assume that the rates $a_1^{[x]}, a_2^{[x]}, \ldots, a_x^{[x]}$ are such that the processing times remain positive. See, e.g., [12] where the required conditions are explicitly written out for a less general combined effect.

For a job scheduled in position r of group x, the rates $a_1^{[x]}, a_2^{[x]}, \ldots, a_{x-1}^{[x]}$, determine how the length of previous groups affects the job's processing time, whereas $a_x^{[x]}$ determines how the processing time of the job is affected by the time elapsed between the opening of the $x-$th group and the start time of the job in position r, $1 \leq r \leq n^{[x]}$. The superscript $[x]$ associated with these rates stresses that the rates are group-dependent, and can assume different values and signs depending on the group x, $1 \leq x \leq k$, a job is scheduled in. For example, to determine the actual processing time of a job scheduled in the third group, the required rates are $a_1^{[3]}, a_2^{[3]}$ and $a_3^{[3]}$, whereas if a job is placed in the fourth group, the required rates are $a_1^{[4]}, a_2^{[4]}, a_3^{[4]}$ and $a_4^{[4]}$.

Notice that it is always assumed that during an RMP the system undergoes neither learning nor deterioration. Thus, the actual processing times of the jobs given by (4) is independent of the duration of the RMPs.

Further, since the number of jobs, $n^{[x]}$, in each group and the total duration of each group, F_x, is known, it follows from (3) that the duration of an RMP after the $x-$th, $1 \leq x \leq k-1$, group can be given by

$$T_x = \alpha_1^{[x]} F_1 + \alpha_2^{[x]} F_2 + \cdots + \alpha_x^{[x]} F_x + \beta^{[x]},\ 1 \leq x \leq k-1, \tag{5}$$

where the values $\alpha_1^{[x]}, \alpha_2^{[x]}, \ldots, \alpha_x^{[x]}$, determine how the length of previous groups affect the duration of the RMP scheduled after the $x-$th group, and $\beta^{[x]} > 0$ is a constant, $1 \leq x \leq k-1$. The values $\alpha_1^{[x]}, \alpha_2^{[x]}, \ldots, \alpha_x^{[x]}$, can be seen as being analogous to the rates $a_1^{[x]}, a_2^{[x]}, \ldots, a_{x-1}^{[x]}$, defined in (4). They allow us to incorporate RMPs of a different nature in a schedule. As an example, consider an instance in which a machine undergoes deterioration and learning simultaneously. Two RMPs are to be included in the processing sequence, the first being used as a training period for the operator, while the second being a maintenance period which repairs the machine. As a result three groups are created, with the number of jobs in each being known as $n^{[1]}$, $n^{[2]}$, and $n^{[3]}$, and the duration of each group being known as F_1, F_2, and F_3. Thus, according to (3) the duration of the first RMP will be given by $D_{RMP}(1) = \left[\alpha^{[1]} F_1 + \psi^{[1]}\right] f^{[1]}\left(n^{[1]}\right)$, while the duration of the second RMP will be given by $D_{RMP}(2) = \left[\alpha^{[2]}(F_1 + F_2) + \psi^{[2]}\right] f^{[2]}\left(n^{[1]} + n^{[2]}\right)$. The latter relation holds as the machine is undergoing continuous deterioration during the processing of the first two groups, and thus, the time till the first MP is performed is equal to $F_1 + F_1$. As a result, in terms of the formula (5), for the first RMP we have $\alpha_1^{[1]} = \alpha^{[1]} f^{[1]}\left(n^{[1]}\right)$, and $\beta^{[1]} = \psi^{[1]} f^{[1]}\left(n^{[1]}\right)$, while for the second RMP we have $\alpha_1^{[2]} = \alpha_2^{[2]} = \alpha^{[2]} f^{[2]}\left(n^{[1]} + n^{[2]}\right)$, and $\beta^{[2]} = \psi^{[2]} f^{[2]}\left(n^{[1]} + n^{[2]}\right)$. Notice that the values $\alpha_1^{[x]}, \alpha_2^{[x]}, \ldots, \alpha_x^{[x]}$, can assume any sign as long as it is ensured that

the duration of the RMPs is positive.

Complicated as it looks, the model essentially incorporates and generalizes almost every job-independent effect known in scheduling with changing processing times. It is flexible enough to serve as a model of many plausible scenarios that may be found in practice, e.g., in the area manufacturing or shop floor production planning. Below we give an illustration of a possible application.

Example 1. A human operator uses a machine (or a tool) to process n jobs. The processing conditions of the machine and the efficiency of an operator are subject to time-dependent and positional effects. In particular, the machine is subject not only to time-dependent deterioration that is typical for most mechanical devices, but also to positional deterioration, which, e.g., happens to a pneumatic punching device that loses air pressure with each punch, and this increases the time needed for subsequent operations; see [2] where such a situation is described in more detail.

During the processing of the jobs, two RMPs will be included in the schedule. It is known that the first RMP is actually a maintenance period which restores the machine to its original condition. However, the deterioration rate of the machine becomes greater after the maintenance period, since the original spare parts are not used. This RMP also provides the operator with sufficient rest, so that after the first RMP the operator is as fresh as he/she was at the beginning of the schedule. The duration of this RMP is dependent on its start-time, i.e., it follows (5) for $x = 1$, with $\alpha_1^{[1]} = \alpha'$ and $\beta^{[1]} = \beta'$. The second RMP takes a constant time β'', i.e., $\alpha_1^{[2]} = \alpha_2^{[2]} = 0$ and $\beta^{[2]} = \beta''$, but does not repair the machine at all. Instead, a new operator is brought in. Below, we distinguish between the learning and deterioration parameters for the machine and those for the operator by using the subscript "m" for the machine and "w" for the operator (worker), respectively.

In a feasible schedule the jobs will be split into $k = 3$ groups. The machine suffers from a linear time-dependent deterioration effect, the deterioration rate being equal to $d'_m > 0$ before the first RMP (the first group), and equal to $d''_m > d'_m > 0$ after the first RMP (for the second and the third groups). Additionally, the machine is also affected by a positional exponential deterioration effect of the form $(d'_m + 1)^{r-1}$ before the first RMP and of the form $(d''_m + 1)^{r-1}$ after that RMP. The operators are also subject to time-dependent effects; the deterioration and learning rates for Operator 1 are $d'_w > 0$ and $l'_w < 0$, respectively, and those for Operator 2 are $d''_w > 0$ and $l''_w < 0$, respectively. It is known that in addition to the skills gained while processing the jobs, Operator 2 also passively learns with a rate $l'''_w < 0$, while he observes Operator 1 process the jobs in groups 1 and 2. Lastly, the performance of the workers is also affected by a polynomial positional effect and is quantified by r^δ, where the appropriate value of δ is one of d'_w, l'_w, d''_w and l''_w, depending on the scenario. There is no positional effect associated with the passive learning effect of Operator 2.

For the described example, the parameters of our model (4) can be set as shown in Table 1.

Notice that our model allows us to assume (unlike, e.g., [12]), that during an RMP, if the operator is not replaced, he/she does not lose his/her skills which were improved due to learning in the earlier groups of the schedule. Similarly, if during an RMP a machine is not fully repaired, our model is capable of handling the resulting situation in which the deterioration effect from the group before the RMP must be carried forward to the next group. The same phenomenon is also observed in the passive learning effect of Operator 2, in which the skills gained during groups 1 and 2 must be carried forward to group 3. These time-

Group	Parameter	Value
1	$a_1^{[1]}$	$d'_m + d'_w + l'_w$
	$g^{[1]}(r)$	$(d'_m+1)^{r-1} r^{d'_w+l'_w}, 1 \le r \le n^{[1]}$
2	$a_1^{[2]}$	l'_w
	$a_2^{[2]}$	$d''_m + d'_w + l'_w$
	$g^{[2]}(r)$	$(d''_m+1)^{r-1} \left(n^{[1]}+r\right)^{l'_w} r^{d'_w}, 1 \le r \le n^{[2]}$
3	$a_1^{[3]}$	l'''_w
	$a_2^{[3]}$	$d''_m + l'''_w$
	$a_3^{[3]}$	$d''_m + d''_w + l''_w$
	$g^{[3]}(r)$	$(d''_m+1)^{n^{[2]}+r-1} r^{d''_w+l''_w}, 1 \le r \le n^{[3]}$

Table 1: Parameters for Example 1

dependent effects can be captured by the group-dependent parameters $a_v^{[x]}$, $1 \le v \le x-1$, $1 \le x \le k$. Thus, to model the situation in the given example, we define $a_1^{[2]} := l'_w$ (implying that Operator 1 has gained an experience of F_1 time units before starting group 2), $a_1^{[3]} := l'''_w$ (implying that Operator 2 has gained a passive learning experience of F_1 time units before starting group 3) and $a_2^{[3]} := d''_m + l'''_w$ (implying that the machine has deteriorated for F_2 time units and Operator 2 has gained a passive learning experience of F_2 time units before starting group 3). These positional effects are simply captured by adjusting the relative position of a job in the relevant group. For example, the learning factor involved in the computation of $g^{[2]}(r)$ is given by $\left(n^{[1]}+r\right)^{l'_w}$ (implying that Operator 1 has completed $n^{[1]}$ jobs before starting group 2), and the deterioration factor involved in the computation of $g^{[3]}(r)$ is given by $(d''_m+1)^{n^{[2]}+r-1}$ (implying that since its last maintenance, the machine has completed $n^{[2]}$ jobs before starting group 3).

As demonstrated in the above example, with appropriate use of the parameters, $a_v^{[x]}$, $\alpha_v^{[x]}$, $1 \le v \le x$, $g^{[x]}(r)$, $1 \le r \le n^{[x]}$, and $\beta^{[x]}$, for each x, $1 \le x \le k$, our model as defined in (4) together with (5), can be used to represent a wide variety of practical situations.

Extending the standard three-field notation, we denote the problems of minimizing the makespan and the total flow time for scheduling models that satisfy the above conditions by $1\,|Combi, RMP|\,C_{\max}$ and $1\,|Combi, RMP|\,\sum C_j$, respectively. Here, in the middle field we write "*Combi*" to stress that the processing times are subject to a combined effect (4), and we write "*RMP*" to indicate that rate modifying activities are being applied in accordance with (5). This is the first study in which combined models of such a general structure are being studied.

An optimal solution to problem $1\,|Combi, RMP|\,F$, where $F \in \{C_{\max}, \sum C_j\}$ must deliver optimal choices for each of the following decisions:

Decision 1. *The number of RMPs:* If $k-1$ RMPs are included in the schedule, the jobs are divided into k groups. Determine the optimal value of k, $1 \le k \le K+1$.

Decision 2. *The choice of RMPs:* From a given list of K RMPs, choose $k-1$, $1 \le k \le K+1$, RMPs that are included in the schedule.

Decision 3. *The sequence of RMPs:* Determine the optimal order in which the selected

$k-1$ RMPs are scheduled on a machine.

Decision 4. *An optimal permutation of jobs:* For each job $j \in N$, determine the group and the position within that group, that it must be scheduled in.

In the next section, we demonstrate how to compute the completion times of any job in terms of the input parameters.

3 Computing Completion Times

To solve problem $1\left|Combi, RMP\right| F$, we first assume that Decisions 1-3 are taken in advance, so that it is known how the selected $k-1$ RMPs are included into the schedule. As a result the jobs are split into k, $1 \leq k \leq K+1$, groups. Define the parameters $a_1^{[x]}, a_2^{[x]}, \ldots, a_x^{[x]}$, $1 \leq x \leq k$, and $\alpha_1^{[x]}, \alpha_2^{[x]}, \ldots, \alpha_x^{[x]}, \beta^{[x]}$, $1 \leq x \leq k-1$, and denote the resulting problem by $1\left|Combi, RMP\left(k-1\right)\right| F$.

To solve problem $1\left|Combi, RMP\left(k-1\right)\right| F$ consider a schedule $S\left(k\right)$ with a permutation of jobs $\pi = \left(\pi^{[1]}, \pi^{[2]}, \ldots, \pi^{[k]}\right)$. Assume that each group contains a total of $n^{[x]}$ jobs, so that $\pi^{[x]} = \left(\pi^{[x]}\left(1\right), \pi^{[x]}\left(2\right), \ldots, \pi^{[x]}\left(n^{[x]}\right)\right)$, $1 \leq x \leq k$, where $\sum_{x=1}^{k} n^{[x]} = n$. We now derive an expression for the total time it takes to process all jobs in a group x, $1 \leq x \leq k$.

Notice that throughout this chapter we assume that an empty product is equal to one, while an empty sum is equal to zero, i.e., $\prod_{i=j}^{r}\left(\cdot\right) = 1$ and $\sum_{i=j}^{r}\left(\cdot\right) = 0$, for $j > r$.

Lemma 1 *Given a group x, $1 \leq x \leq k$, and job $j = \pi^{[x]}\left(r\right)$ sequenced in the $r-$th position, $1 \leq r \leq n^{[x]}$, of the group, the completion time $F_{(x,r)}$ of the job with respect to the start time of the group is given by*

$$F_{(x,r)} = \sum_{u=1}^{r}\left(A^{[x]} + p_{\pi^{[x]}(u)}\right) B^{[x]}\left(u, r\right), \tag{6}$$

where $A^{[x]} := \sum_{v=1}^{x-1} a_v^{[x]} F_v$, and

$$B^{[x]}\left(u, r\right) := g^{[x]}\left(u\right) \prod_{i=u+1}^{r}\left(1 + a_x^{[x]} g^{[x]}\left(i\right)\right), 1 \leq u \leq r. \tag{7}$$

Proof: We prove this lemma by induction. For job $j = \pi^{[x]}\left(r\right)$, the value $F_{(x,r)}$ for $r = 1$ is given in accordance with (4) by

$$\begin{aligned} p_j^{[x]}\left(1\right) &= \left(p_{\pi^{[x]}(1)} + a_1^{[x]} F_1 + a_2^{[x]} F_2 + ... + a_{x-1}^{[x]} F_{x-1}\right) g^{[x]}\left(1\right) \\ &= \left(p_{\pi^{[x]}(1)} + A^{[x]}\right) g^{[x]}\left(1\right) = \left(p_{\pi^{[x]}(1)} + A^{[x]}\right) B^{[x]}\left(1, 1\right), \end{aligned}$$

which corresponds to the right-hand side of (6).

Assume that for r, $1 < r \leq n^{[x]}$, the equality

$$F_{(x,r-1)} = \sum_{u=1}^{r-1}\left(A^{[x]} + p_{\pi^{[x]}(u)}\right) B^{[x]}\left(u, r-1\right), \tag{8}$$

holds. We know that

$$F_{(x,r)} = F_{(x,r-1)} + p_j^{[x]}(r).$$

Substituting (4) we get

$$\begin{aligned} F_{(x,r)} &= F_{(x,r-1)} + \left(p_{\pi^{[x]}(r)} + A^{[x]} + a_x^{[x]} F_{(x,r-1)}\right) g^{[x]}(r) \\ &= \left(1 + a_x^{[x]} g^{[x]}(r)\right) F_{(x,r-1)} + \left(p_{\pi^{[x]}(r)} + A^{[x]}\right) g^{[x]}(r). \end{aligned}$$

Substituting the value of $F_{(x,r-1)}$ from (8), we obtain

$$F_{(x,r)} = \left(1 + a_x^{[x]} g^{[x]}(r)\right) \sum_{u=1}^{r-1} \left(A^{[x]} + p_{\pi^{[x]}(u)}\right) B^{[x]}(u, r-1) + \left(A^{[x]} + p_{\pi^{[x]}(r)}\right) g^{[x]}(r).$$

It can be easily verified that $\left(1 + a_x^{[x]} g^{[x]}(r)\right) B^{[x]}(u, r-1) = B^{[x]}(u, r)$ and $g^{[x]}(r) = B^{[x]}(r, r)$, so that we obtain

$$\begin{aligned} F_{(x,r)} &= \sum_{u=1}^{r-1} \left(A^{[x]} + p_{\pi^{[x]}(u)}\right) B^{[x]}(u, r) + \left(A^{[x]} + p_{\pi^{[x]}(r)}\right) B^{[x]}(r, r) \\ &= \sum_{u=1}^{r} \left(A^{[x]} + p_{\pi^{[x]}(u)}\right) B^{[x]}(u, r), \end{aligned}$$

which proves the lemma. ∎

Due to Lemma 1, the total processing time of a group x, $1 \le x \le k$, can be written as

$$\begin{aligned} F_x &= F_{\left(x, n^{[x]}\right)} = \sum_{r=1}^{n^{[x]}} \left(A^{[x]} + p_{\pi^{[x]}(r)}\right) B^{[x]}\left(r, n^{[x]}\right) \\ &= \sum_{r=1}^{n^{[x]}} \left(\sum_{v=1}^{x-1} a_v^{[x]} F_v\right) B^{[x]}\left(r, n^{[x]}\right) + \sum_{r=1}^{n^{[x]}} p_{\pi^{[x]}(r)} B^{[x]}\left(r, n^{[x]}\right) \\ &= \sum_{v=1}^{x-1} \left(a_v^{[x]} \sum_{r=1}^{n^{[x]}} B^{[x]}\left(r, n^{[x]}\right)\right) F_v + \sum_{r=1}^{n^{[x]}} p_{\pi^{[x]}(r)} B^{[x]}\left(r, n^{[x]}\right). \end{aligned}$$

For convenience, denote

$$D^{[x]} := \sum_{r=1}^{n^{[x]}} p_{\pi^{[x]}(r)} B^{[x]}\left(r, n^{[x]}\right), \ 1 \le x \le k, \tag{9}$$

and

$$b_v^{[x]} := a_v^{[x]} \sum_{r=1}^{n^{[x]}} B^{[x]}\left(r, n^{[x]}\right), \ 1 \le v \le x-1, \ 1 \le x \le k. \tag{10}$$

Then F_x can be rewritten as

$$F_x = b_1^{[x]} F_1 + b_2^{[x]} F_2 + \cdots + b_{x-1}^{[x]} F_{x-1} + D^{[x]}. \tag{11}$$

Lemma 2 *The total processing time of a group x, $1 \le x \le k$, is given by*

$$F_x = \sum_{v=1}^{x} E^{[v,x]} D^{[v]}, \tag{12}$$

where $E^{[x,x]} := 1$ and for $v \le x-1$

$$E^{[v,x]} := \sum_{w=1}^{x-v} \sum_{v=v_0<v_1<\cdots<v_w=x} b_{v_0}^{[v_1]} b_{v_1}^{[v_2]} b_{v_2}^{[v_3]} \cdots b_{v_{w-1}}^{[v_w]}, \tag{13}$$

where in (13) for each w, $1 \le w \le x-v$, the second summation is taken over all increasing sequences of $w+1$ distinct integers $(v_0, v_1, \ldots, v_w)$ with $v_0 = v$ and $v_w = x$.

Proof: Please notice the zigzag pattern of the subscripts and superscripts in the right-hand side of (13), as outlined below.

$$\begin{array}{ccccccccccccc} & & [v_1] & & & & [v_2] & & \cdots & & [v_{w-1}] & & [v_w] \\ & \nearrow & & \searrow & & \nearrow & & \searrow & \nearrow\ \cdots\ \nearrow & & & \searrow & & \nearrow \\ v_0 & & & & v_1 & & & & v_2 \quad \cdots & & & & v_{w-1} & & \end{array}$$

The proof of the lemma is by induction. For $x = 1$, it follows from (11) that $F_1 = D^{[1]}$. On the other hand, for $x = 1$, (12) reduces to $F_1 = E^{[1,1]} D^{[1]}$, and since $E^{[1,1]} = 1$ by definition, we also obtain $F_1 = D^{[1]}$.

Assume now that the lemma holds for all groups $1, 2, \ldots, x-1$, and prove that it holds for group $x \le k$. Starting with (11), we write

$$F_x = \sum_{v=1}^{x-1} b_v^{[x]} F_v + D^{[x]},$$

and use the induction assumption to substitute (12) into the above expression to obtain

$$\begin{aligned} F_x &= \sum_{v=1}^{x-1} b_v^{[x]} \sum_{y=1}^{v} E^{[y,v]} D^{[y]} + D^{[x]} \\ &= \sum_{v=1}^{x-1} \sum_{y=v}^{x-1} b_y^{[x]} E^{[v,y]} D^{[v]} + D^{[x]} \\ &= \sum_{v=1}^{x-1} \left(b_v^{[x]} E^{[v,v]} + \sum_{y=v+1}^{x-1} b_y^{[x]} E^{[v,y]} \right) D^{[v]} + D^{[x]}. \end{aligned}$$

Further, using the induction assumption, replace $E^{[v,v]}$ by 1 and substitute $E^{[v,y]}, v < y$, in accordance with (13). We deduce

$$\begin{aligned} F_x &= \sum_{v=1}^{x-1} \left(b_v^{[x]} + \sum_{y=v+1}^{x-1} b_y^{[x]} \left(\sum_{w=1}^{y-v} \sum_{v=v_0<v_1<\cdots<v_w=y} b_{v_0}^{[v_1]} b_{v_1}^{[v_2]} b_{v_2}^{[v_3]} \cdots b_{v_{w-1}}^{[v_w]} \right) \right) D^{[v]} + D^{[x]} \\ &= \sum_{v=1}^{x-1} \left(b_v^{[x]} + \sum_{y=v+1}^{x-1} \sum_{w=1}^{y-v} \sum_{v=v_0<v_1<\cdots<v_w=y} b_{v_0}^{[v_1]} b_{v_1}^{[v_2]} b_{v_2}^{[v_3]} \cdots b_{v_{w-1}}^{[v_w]} b_y^{[x]} \right) D^{[v]} + D^{[x]} \\ &= \sum_{v=1}^{x-1} \left(b_v^{[x]} + \sum_{y=v+1}^{x-1} \sum_{w=1}^{y-v} \sum_{v=v_0<v_1<\cdots<v_w=y} b_{v_0}^{[v_1]} b_{v_1}^{[v_2]} b_{v_2}^{[v_3]} \cdots b_{v_{w-1}}^{[v_w]} b_{v_w}^{[x]} \right) D^{[v]} + D^{[x]}. \end{aligned}$$

Observe that for a fixed w the equality

$$\sum_{y=v+1}^{x-1} \sum_{v=v_0<v_1<\cdots<v_w=y} b_{v_0}^{[v_1]} b_{v_1}^{[v_2]} b_{v_2}^{[v_3]} \cdots b_{v_{w-1}}^{[v_w]} b_{v_w}^{[x]} = \sum_{v=v_0<v_1<\cdots<v_w\le x-1} b_{v_0}^{[v_1]} b_{v_1}^{[v_2]} b_{v_2}^{[v_3]} \cdots b_{v_{w-1}}^{[v_w]} b_{v_w}^{[x]}$$

holds, where the summation in the right-hand side is taken over all increasing sequences of $w+1$ distinct integers $(v_0, v_1, \ldots, v_w)$ with $v_0 = v$ and $v_w \le x-1$.

Notice that for $y = v+1$ and for $y = x-1$ we have that $w = 1$ and $w = x-v-1$, respectively, i.e., $1 \le w \le x-v-1$. This means that

$$F_x = \sum_{v=1}^{x-1} \left(b_v^{[x]} + \sum_{w=1}^{x-v-1} \sum_{v=v_0<v_1<\cdots<v_w\le x-1} b_{v_0}^{[v_1]} b_{v_1}^{[v_2]} b_{v_2}^{[v_3]} \cdots b_{v_{w-1}}^{[v_w]} b_{v_w}^{[x]} \right) D^{[v]} + D^{[x]}.$$

Replacing w with $w-1$, rewrite

$$F_x = \sum_{v=1}^{x-1} \left(b_v^{[x]} + \sum_{w=2}^{x-v} \sum_{v=v_0<v_1<\cdots<v_{w-1}\le x-1} b_{v_0}^{[v_1]} b_{v_1}^{[v_2]} b_{v_2}^{[v_3]} \cdots b_{v_{w-2}}^{[v_{w-1}]} b_{v_{w-1}}^{[x]} \right) D^{[v]} + D^{[x]}.$$

It can be easily verified that

$$b_v^{[x]} = \sum_{w=1}^{1} \sum_{v=v_0<v_1<\cdots<v_{w-1}<v_w=x} b_{v_0}^{[v_1]} b_{v_1}^{[v_2]} \cdots b_{v_{w-2}}^{[v_{w-1}]} b_{v_{w-1}}^{[v_w]}, 1 \le v \le x-1,$$

so that $F_x, 1 \le x \le k$, can be rewritten as

$$\begin{aligned} F_x &= \sum_{v=1}^{x-1} \left(\sum_{w=1}^{x-v} \sum_{v=v_0<v_1<\cdots<v_{w-1}<v_w=x} b_{v_0}^{[v_1]} b_{v_1}^{[v_2]} b_{v_2}^{[v_3]} \cdots b_{v_{w-2}}^{[v_{w-1}]} b_{v_{w-1}}^{[v_w]} \right) D^{[v]} + D^{[x]} \\ &= \sum_{v=1}^{x-1} E^{[v,x]} D^{[v]} + E^{[x,x]} D^{[x]} = \sum_{v=1}^{x} E^{[v,x]} D^{[v]}, \end{aligned}$$

which proves the lemma. ∎

The expressions in Lemma 2 look heavy; below we illustrate them for small values of x, e.g., $x \le 4$. Recall that for $x = 1$, we have $F_1 = D^{[1]}$. For $x = 2$, the formula (11) gives $F_2 = b_1^{[2]} F_1 + D^{[2]} = b_1^{[2]} D^{[1]} + D^{[2]}$, which complies with (12) for $x = 2$, since $E^{[1,2]} = b_1^{[2]}$ and $E^{[2,2]} = 1$. Similarly, for $x = 3$, the formula (11) reduces to

$$\begin{aligned} F_3 &= b_1^{[3]} F_1 + b_2^{[3]} F_2 + D^{[3]} = b_1^{[3]} D^{[1]} + b_2^{[3]} \left(b_1^{[2]} D^{[1]} + D^{[2]} \right) + D^{[3]} \\ &= \left(b_1^{[3]} + b_1^{[2]} b_2^{[3]} \right) D^{[1]} + b_2^{[3]} D^{[2]} + D^{[3]}. \end{aligned}$$

It can be easily verified that this complies with (12) for $x = 3$. For $x = 4$, using (11) we obtain

$$\begin{aligned} F_4 &= b_1^{[4]} F_1 + b_2^{[4]} F_2 + b_3^{[4]} F_3 + D^{[4]} \\ &= b_1^{[4]} D^{[1]} + b_2^{[4]} \left(b_1^{[2]} D^{[1]} + D^{[2]} \right) + b_3^{[4]} \left(\left(b_1^{[3]} + b_1^{[2]} b_2^{[3]} \right) D^{[1]} + b_2^{[3]} D^{[2]} + D^{[3]} \right) + D^{[4]} \\ &= \left(b_1^{[4]} + b_1^{[2]} b_2^{[4]} + b_1^{[3]} b_3^{[4]} + b_1^{[2]} b_2^{[3]} b_3^{[4]} \right) D^{[1]} + \left(b_2^{[4]} + b_2^{[3]} b_3^{[4]} \right) D^{[2]} + b_3^{[4]} D^{[3]} + D^{[4]}. \end{aligned}$$

On the other hand, using (12) we obtain

$$\begin{aligned}
F_4 &= E^{[1,4]}D^{[1]} + E^{[2,4]}D^{[2]} + E^{[3,4]}D^{[3]} + E^{[4,4]}D^{[4]} \\
&= \left(\sum_{w=1}^{3} \sum_{1=v_0<v_1<\cdots<v_w=4} b_{v_0}^{[v_1]} b_{v_1}^{[v_2]} \cdots b_{v_{w-1}}^{[v_w]}\right) D^{[1]} \\
&\quad + \left(\sum_{w=1}^{2} \sum_{2=v_0<v_1<\cdots<v_w=4} b_{v_0}^{[v_1]} b_{v_1}^{[v_2]} \cdots b_{v_{w-1}}^{[v_w]}\right) D^{[2]} \\
&\quad + \left(\sum_{w=1}^{1} \sum_{3=v_0<v_1<\cdots<v_w=4} b_{v_0}^{[v_1]} b_{v_1}^{[v_2]} \cdots b_{v_{w-1}}^{[v_w]}\right) D^{[3]} + D^{[4]} \\
&= \left(\sum_{1=v_0<v_1=4} b_{v_0}^{[v_1]} + \sum_{1=v_0<v_1<v_2=4} b_{v_0}^{[v_1]} b_{v_1}^{[v_2]} + \sum_{1=v_0<v_1<v_2<v_3=4} b_{v_0}^{[v_1]} b_{v_1}^{[v_2]} b_{v_2}^{[v_3]}\right) D^{[1]} + \\
&\quad \left(\sum_{2=v_0<v_1=4} b_{v_0}^{[v_1]} + \sum_{2=v_0<v_1<v_2=4} b_{v_0}^{[v_1]} b_{v_1}^{[v_2]}\right) D^{[2]} + \left(\sum_{3=v_0<v_1=4} b_{v_0}^{[v_1]}\right) D^{[3]} + D^{[4]} \\
&= \left(b_1^{[4]} + b_1^{[2]}b_2^{[4]} + b_1^{[3]}b_3^{[4]} + b_1^{[2]}b_2^{[3]}b_3^{[4]}\right) D^{[1]} + \left(b_2^{[4]} + b_2^{[3]}b_3^{[4]}\right) D^{[2]} + b_3^{[4]}D^{[3]} + D^{[4]}.
\end{aligned}$$

Notice that the number of terms contained in the expression $\sum_{v=v_0<v_1<\cdots<v_w=x} b_{v_0}^{[v_1]} b_{v_1}^{[v_2]} b_{v_2}^{[v_3]} \cdot \cdot \cdot b_{v_{w-1}}^{[v_w]}$, involved in the right-hand side of (13) is $\binom{x-v-1}{w-1}$. Thus, to determine all values of $E^{[v,x]}$, $1 \le v \le x-1$, $2 \le x \le k$, the total number of products to be computed is $\sum_{x=2}^{k} \sum_{v=1}^{x-1} \sum_{w=1}^{x-v} \binom{x-v-1}{w-1} = 2^k - k - 1$.

Recall that the durations of the RMPs are given by (5). Including the time spent on rate-modifying activities, the completion time $C_{\pi^{[x]}(r)}$ of a job scheduled in position r, $1 \le r \le n^{[x]}$, of the $x-$th group, $1 \le x \le k$, can be written as

$$\begin{aligned}
C_{\pi^{[x]}(r)} &= F_1 + T_1 + F_2 + T_2 + \cdots + F_{x-1} + T_{x-1} + F_{(x,r)} \\
&= \sum_{v=1}^{x-1} \left[1 + \sum_{w=v}^{x-1} \alpha_v^{[w]}\right] F_v + F_{(x,r)} + \sum_{v=1}^{x-1} \beta^{[v]}.
\end{aligned}$$

For convenience, denote

$$\xi^{[v,x-1]} := 1 + \sum_{w=v}^{x-1} \alpha_v^{[w]}, 1 \le v \le x-1, 1 \le x \le k, \tag{14}$$

and rewrite

$$C_{\pi^{[x]}(r)} = \sum_{v=1}^{x-1} \xi^{[v,x-1]} F_v + F_{(x,r)} + \sum_{v=1}^{x-1} \beta^{[v]} \tag{15}$$

Applying the results of Lemmas 1 and 2, the completion time can be written in terms of the original problem parameters.

4 Computing Positional Weights

The purpose of this section is to consider problems $1\,|Combi, RMP(k-1)|\,C_{\max}$ and $1\,|Combi, RMP(k-1)|\,\sum C_j$ and to demonstrate that even without making additional assumptions regarding the sign of $a_v^{[x]}$, $1 \le v \le x$, $1 \le x \le k$, or the shape of $g^{[x]}(r)$,

$1 \leq r \leq n^{[x]}$, $1 \leq x \leq k$, each of the two problems reduces to finding a permutation of jobs that delivers the minimum to a generic objective function of the form

$$F(k) = \sum_{x=1}^{k} \sum_{r=1}^{n^{[x]}} W^{[x]}(r) p_{\pi^{[x]}(r)} + \Gamma(k), \tag{16}$$

where $W^{[x]}(r)$ is a suitably defined positional weight and $\Gamma(k)$ is a constant. The problem of minimizing function $F(k)$ can be qualified as that of minimizing a linear form over a set of all permutations. It also can be seen a special case of the linear assignment problem with a product matrix, and can be solved by a well-known matching algorithm that is due to Hardy et al. [19]. See Section 5, where the algorithmic issues are discussed in more detail.

4.1 Minimizing The Makespan

We start with problem $1\,|Combi, RMP(k-1)|\,C_{\max}$. For a schedule $S(k)$ with k groups denote the makespan by $C_{\max}(S(k))$. If schedule $S(k)$ is associated with a permutation $\pi = \left(\pi^{[1]}, \pi^{[2]}, \ldots, \pi^{[k]}\right)$, then $C_{\max}(S(k))$ is equal to $C_{\pi^{[k]}\left(n^{[k]}\right)}$, the completion time of the last job in the last $k-$th group. In Section 3, we demonstrate that if the $x-$th group contains a total of $n^{[x]}$ jobs, so that $\pi^{[x]} = \left(\pi^{[x]}(1), \pi^{[x]}(2), \ldots, \pi^{[x]}\left(n^{[x]}\right)\right)$, $1 \leq x \leq k$, then the completion time of a job $j = \pi^{[x]}(r)$ is given by (15). It follows that the makespan of schedule $S(k)$ can be written as

$$C_{\max}(S(k)) = C_{\pi^{[k]}\left(n^{[k]}\right)} = \sum_{x=1}^{k-1} \xi^{[x,k-1]} F_x + F_k + \sum_{x=1}^{k-1} \beta^{[x]}.$$

Substituting (12) in the above expression we obtain

$$C_{\max}(S(k)) = \sum_{x=1}^{k-1} \xi^{[x,k-1]} \left(\sum_{v=1}^{x} E^{[v,x]} D^{[v]} \right) + \sum_{v=1}^{k} E^{[v,k]} D^{[v]} + \sum_{x=1}^{k-1} \beta^{[x]}.$$

Rearranging the terms we get

$$C_{\max}(S(k)) = \sum_{x=1}^{k} \left(\sum_{v=x}^{k-1} \xi^{[v,k-1]} E^{[x,v]} + E^{[x,k]} \right) D^{[x]} + \sum_{x=1}^{k-1} \beta^{[x]}.$$

Substituting the value of $D^{[x]}$ from (9) we further derive

$$\begin{aligned} C_{\max}(S(k)) &= \sum_{x=1}^{k} \left(\sum_{v=x}^{k-1} \xi^{[v,k-1]} E^{[x,v]} + E^{[x,k]} \right) \sum_{r=1}^{n^{[x]}} p_{\pi^{[x]}(r)} B^{[x]}\left(r, n^{[x]}\right) + \sum_{x=1}^{k-1} \beta^{[x]} \\ &= \sum_{x=1}^{k} \sum_{r=1}^{n^{[x]}} p_{\pi^{[x]}(r)} B^{[x]}\left(r, n^{[x]}\right) \left(\sum_{v=x}^{k-1} \xi^{[v,k-1]} E^{[x,v]} + E^{[x,k]} \right) + \sum_{x=1}^{k-1} \beta^{[x]}. \end{aligned}$$

Notice, that the expression for the makespan $C_{\max}(S(k))$ has reduced to the generic objective function (16), with the constant term $\Gamma(k)$ given by

$$\Gamma(k) = \sum_{x=1}^{k-1} \beta^{[x]}, \tag{17}$$

and the positional weights given by

$$W^{[x]}(r) = B^{[x]}\left(r, n^{[x]}\right)\left(\sum_{v=x}^{k-1}\left(1+\sum_{w=v}^{k-1}\alpha_v^{[w]}\right)E^{[x,v]}+E^{[x,k]}\right),\ 1\le r\le n^{[x]},\ 1\le x\le k. \tag{18}$$

The expression (18) for the positional weights can be written in terms of the original parameters by using (7), (10) and (13). Finding all values of $B^{[x]}\left(r, n^{[x]}\right)$, $1 \le r \le n^{[x]}$, $1 \le x \le k$, by (7) requires $O\left(\sum_{x=1}^{k} n^{[x]}\right) = O(n)$ time. After that, all values of $b_v^{[x]}$, $1 \le v \le x-1$, $1 \le x \le k$, can be found by (10) in $O\left(k^2\right)$ time. As follows from Section 3, computing all values $E^{[v,x]}$, $1 \le v \le x-1$, $1 \le x \le k$, requires $O\left(2^k\right)$ time. Finally, all n positional weights $W^{[x]}(r)$, $1 \le r \le n^{[x]}$, $1 \le x \le k$, will be computed in $O(n)$ time, provided that all other quantities have been found. Thus, the overall running time needed to compute the positional weights for problem $1\,|Combi, RMP(k-1)|\,C_{\max}$ is $T(W) = O\left(2n + k^2 + 2^k\right) = O(n)$, provided that k is a given constant.

4.2 Minimizing The Total Flow Time

Now we address problem $1\,|Combi, RMP(k-1)|\sum C_j$. For a schedule $S(k)$ with k groups associated with a permutation $\pi = \left(\pi^{[1]}, \pi^{[2]}, \ldots, \pi^{[k]}\right)$, the total flow time can be written as

$$\sum C_j = \sum_{x=1}^{k}\sum_{r=1}^{n^{[x]}} C_{\pi^{[x]}(r)}.$$

Substituting (6) into (15), we rewrite the expression for $C_{\pi^{[x]}(r)}$ as

$$\begin{aligned} C_{\pi^{[x]}(r)} &= \sum_{v=1}^{x-1}\xi^{[v,x-1]}F_v + \sum_{u=1}^{r}\left(A^{[x]} + p_{\pi^{[x]}(u)}\right)B^{[x]}(u,r) + \sum_{v=1}^{x-1}\beta^{[v]} \\ &= \sum_{v=1}^{x-1}\xi^{[v,x-1]}F_v + A^{[x]}\sum_{u=1}^{r}B^{[x]}(u,r) + \sum_{u=1}^{r}p_{\pi^{[x]}(u)}B^{[x]}(u,r) + \sum_{v=1}^{x-1}\beta^{[v]}. \end{aligned}$$

Substituting $A^{[x]} = \sum_{v=1}^{x-1} a_v^{[x]} F_v$ into the above equation, we obtain

$$C_{\pi^{[x]}(r)} = \sum_{v=1}^{x-1}\left(\xi^{[v,x-1]} + a_v^{[x]}\sum_{u=1}^{r}B^{[x]}(u,r)\right)F_v + \sum_{u=1}^{r}p_{\pi^{[x]}(u)}B^{[x]}(u,r) + \sum_{v=1}^{x-1}\beta^{[v]}.$$

Thus, the total flow time of schedule $S(k)$ can be written as

$$\begin{aligned} \sum C_j &= \sum_{x=1}^{k}\sum_{r=1}^{n^{[x]}}\left[\sum_{v=1}^{x-1}\left(\xi^{[v,x-1]} + a_v^{[x]}\sum_{u=1}^{r}B^{[x]}(u,r)\right)F_v + \sum_{u=1}^{r}p_{\pi^{[x]}(u)}B^{[x]}(u,r)\right] + \Gamma(k) \\ &= \sum_{x=1}^{k}\left[\sum_{v=1}^{x-1}\left(n^{[x]}\xi^{[v,x-1]} + a_v^{[x]}\sum_{r=1}^{n^{[x]}}\sum_{u=1}^{r}B^{[x]}(u,r)\right)F_v + \sum_{r=1}^{n^{[x]}}\sum_{u=1}^{r}p_{\pi^{[x]}(u)}B^{[x]}(u,r)\right] \\ &\quad +\Gamma(k), \end{aligned}$$

where $\Gamma(k) = \sum_{x=1}^{k}\sum_{r=1}^{n^{[x]}}\sum_{v=1}^{x-1}\beta^{[v]}$. It can be easily verified that the term $\sum_{r=1}^{n^{[x]}}\sum_{u=1}^{r}p_{\pi^{[x]}(u)}B^{[x]}(u,r)$ can be rewritten as $\sum_{r=1}^{n^{[x]}}p_{\pi^{[x]}(r)}\sum_{u=r}^{n^{[x]}}B^{[x]}(r,u)$, $1 \le x \le k$, so that we have

$$\sum C_j = \sum_{x=1}^{k}\left[\sum_{v=1}^{x-1}\left(n^{[x]}\xi^{[v,x-1]} + a_v^{[x]}\sum_{r=1}^{n^{[x]}}\sum_{u=r}^{n^{[x]}}B^{[x]}(r,u)\right)F_v + \sum_{r=1}^{n^{[x]}}p_{\pi^{[x]}(r)}\sum_{u=r}^{n^{[x]}}B^{[x]}(r,u)\right] + \Gamma(k).$$

For convenience, substitute the value $\xi^{[v,x-1]}$ from (14) and denote

$$G^{[v,x]} := n^{[x]}\left(1 + \sum_{w=v}^{x-1}\alpha_v^{[w]}\right) + a_v^{[x]}\sum_{r=1}^{n^{[x]}}\sum_{u=r}^{n^{[x]}}B^{[x]}(r,u),\ 1 \le v \le x-1,\ 1 \le x \le k, \tag{19}$$

so that we have

$$\sum C_j = \sum_{x=1}^{k}\left(\sum_{v=1}^{x-1}G^{[v,x]}F_v + \sum_{r=1}^{n^{[x]}}p_{\pi^{[x]}(r)}\sum_{u=r}^{n^{[x]}}B^{[x]}(r,u)\right) + \Gamma(k).$$

Substituting the value of F_v from (12) into the above equation, we deduce

$$\begin{aligned}
\sum C_j &= \sum_{x=1}^{k}\left(\sum_{v=1}^{x-1}G^{[v,x]}\sum_{w=1}^{v}E^{[w,v]}D^{[w]} + \sum_{r=1}^{n^{[x]}}p_{\pi^{[x]}(r)}\sum_{u=r}^{n^{[x]}}B^{[x]}(r,u)\right) + \Gamma(k)\\
&= \sum_{x=1}^{k}\left[\sum_{v=1}^{x-1}\left(\sum_{w=v}^{x-1}G^{[w,x]}E^{[v,w]}\right)D^{[v]} + \sum_{r=1}^{n^{[x]}}p_{\pi^{[x]}(r)}\sum_{u=r}^{n^{[x]}}B^{[x]}(r,u)\right] + \Gamma(k)\\
&= \sum_{x=1}^{k}\sum_{v=x+1}^{k}\left(\sum_{w=x}^{v-1}G^{[w,v]}E^{[x,w]}\right)D^{[x]} + \sum_{x=1}^{k}\sum_{r=1}^{n^{[x]}}p_{\pi^{[x]}(r)}\sum_{u=r}^{n^{[x]}}B^{[x]}(r,u) + \Gamma(k)\\
&= \sum_{x=1}^{k}D^{[x]}\sum_{v=x+1}^{k}\sum_{w=x}^{v-1}G^{[w,v]}E^{[x,w]} + \sum_{x=1}^{k}\sum_{r=1}^{n^{[x]}}p_{\pi^{[x]}(r)}\sum_{u=r}^{n^{[x]}}B^{[x]}(r,u) + \Gamma(k).
\end{aligned}$$

Further, substituting the value of $D^{[x]}$ from (9) and rearranging terms, we obtain

$$\begin{aligned}
\sum C_j &= \sum_{x=1}^{k}\left[\left(\sum_{r=1}^{n^{[x]}}p_{\pi^{[x]}(r)}B^{[x]}\left(r,n^{[x]}\right)\right)\sum_{v=x+1}^{k}\sum_{w=x}^{v-1}G^{[w,v]}E^{[x,w]}\right.\\
&\qquad \left. + \sum_{r=1}^{n^{[x]}}p_{\pi^{[x]}(r)}\sum_{u=r}^{n^{[x]}}B^{[x]}(r,u)\right] + \Gamma(k)\\
&= \sum_{x=1}^{k}\sum_{r=1}^{n^{[x]}}p_{\pi^{[x]}(r)}\left(B^{[x]}\left(r,n^{[x]}\right)\sum_{v=x+1}^{k}\sum_{w=x}^{v-1}G^{[w,v]}E^{[x,w]} + \sum_{u=r}^{n^{[x]}}B^{[x]}(r,u)\right) + \Gamma(k).
\end{aligned}$$

Thus, the total flow time can be represented as a function of the form (16) with the positional weights defined by

$$W^{[x]}(r) = B^{[x]}\left(r,n^{[x]}\right)\sum_{v=x+1}^{k}\sum_{w=x}^{v-1}G^{[w,v]}E^{[x,w]} + \sum_{u=r}^{n^{[x]}}B^{[x]}(r,u),\ 1 \le r \le n^{[x]},\ 1 \le x \le k, \tag{20}$$

and the constant term

$$\Gamma(k) = \sum_{x=1}^{k} \sum_{r=1}^{n^{[x]}} \sum_{v=1}^{x-1} \beta^{[v]}. \tag{21}$$

The expression (20) for the positional weights can be written in terms of the original parameters by using (7), (19), (13) and (10). All values $B^{[x]}(u,r)$, $1 \leq u \leq r \leq n^{[x]}, 1 \leq x \leq k$, can be computed by (7) in $O\left(\sum_{x=1}^{k} \frac{n^{[x]}\left(n^{[x]}+1\right)}{2}\right) = O(n^2)$ time. After that, all values of $G^{[v,x]}$, $1 \leq v \leq x-1$, $1 \leq x \leq k$, can be found by (19) in $O\left(k^2\right)$ time. As discussed in Section 4.1, computing all values $E^{[v,x]}, 1 \leq v \leq x-1$, $1 \leq x \leq k$, requires $O\left(2^k\right)$ time. Finally, all n positional weights $W^{[x]}(r)$, $1 \leq r \leq n^{[x]}, 1 \leq x \leq k$, will be computed in $O(n)$ time, provided that all other quantities have been found. Thus, the overall running time needed to compute the positional weights for problem $1\,|Combi, RMP(k-1)|\,C_{\max}$ is $T(W) = O\left(n + n^2 + k^2 + 2^k\right) = O\left(n^2\right)$, provided that k is a given constant.

5 The Solution Approach

The purpose of this section is to consider problem $1\,|Combi, RMP|\,F$ where $F \in \{C_{\max}, \sum C_j\}$ and design a solution approach for it. In order to solve problem $1\,|Combi, RMP|\,F$, we must find the optimal outcomes for Decisions 1-4 listed in Section 2. First, let us concentrate on the relevant sub-problem $1\,|Combi, RMP(k-1)|\,F$ in which it is assumed that Decisions 1-3 are taken in advance.

In the previous section, we demonstrate that if the number of jobs in each group $n^{[x]}$, $1 \leq x \leq k$, is known in advance both problems $1\,|Combi, RMP(k-1)|\,C_{\max}$ and $1\,|Combi, RMP(k-1)|\,\sum C_j$ reduce to minimizing a linear form (16) over a set of all permutations, or equivalently, to a special case of the linear assignment problem. Typically, an input for such a problem is determined by two arrays $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ and $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$. Provided that $\beta_1 \leq \beta_2 \leq \ldots \leq \beta_n$, the problem consists of finding a permutation $\varphi = (\varphi(1), \varphi(2), \ldots, \varphi(n))$ of the components of array α, such that for any permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$, the inequality

$$\sum_{j=1}^{n} \alpha_{\varphi(j)}\beta_j \leq \sum_{j=1}^{n} \alpha_{\pi(j)}\beta_j$$

holds. In such a problem, n items have to be assigned to n positions, the cost of assigning an item i to position j is given by $\alpha_i\beta_j$, and the total cost of the assignments is to be minimized. The classical result by Hardy et al. [19] states that a permutation φ satisfies

$$\alpha_{\varphi(1)} \geq \alpha_{\varphi(2)} \geq \ldots \geq \alpha_{\varphi(n)},$$

provided that $\beta_1 \leq \beta_2 \leq \ldots \leq \beta_n$.

Thus, the problem of minimizing a function of the form (16) can be solved by the following *matching algorithm*: permutation φ matches the larger components of one of the two given arrays with smaller components of the other array. The matching algorithm requires $O(n \log n)$ time. This algorithm is quite well-known and has become a popular technique for solving scheduling problems in this area; see, e.g., [29], [30], [31] and [32]. Also, see a survey [2], which demonstrates that many scheduling problems can be solved by the matching algorithm, but instead less suitable approaches were used in earlier papers.

Based on the matching algorithm, below we provide a formal description of an algorithm, that solves an instance of problem $1\,|Combi, RMP\,(k-1)|\,F$ by minimizing the objective function $F\,(k)$ of the form (16). This approach is similar to that used in [2] for less general models with changing processing times.

Algorithm Generate

INPUT: An instance of problem $1\,|Combi, RMP\,(k-1)|\,F$

OUTPUT: An optimal schedule $S^{*}\,(k)$

Step 1. If required, renumber the jobs in *LPT order*, i.e., in non-increasing order of the values p_j.

Step 2. Generate all possible instances in which n jobs are split into k non-empty groups, so that we have the values $n^{[x]}$, $1 \leq x \leq k$, such that $\sum_{x=1}^{k} n^{[x]} = n$.

Step 3. For each instance do

(a) Compute the constant term $\Gamma\,(k)$ and the positional weights $W^{[x]}(r)$, $1 \leq r \leq n^{[x]}$, $1 \leq x \leq k$, by (18) or (20), for the given problem.

(b) Sort the computed positional weights $W^{[x]}(r)$, $1 \leq r \leq n^{[x]}$, $1 \leq x \leq k$, in non-decreasing order of their values, and store them in a list $L := (\gamma_1, \gamma_2, \ldots, \gamma_n)$, where γ_j, $1 \leq j \leq n$, denotes the $j-$th largest element in the list L.

(c) Match job j to the $j-$th element in list L, so that the optimal value of the function $F(k)$ is given by $\sum_{j=1}^{n} \gamma_j p_j + \Gamma(k)$.

Notice that in order to compute the optimal values of $n^{[x]}$, $1 \leq x \leq k$, Algorithm Generate simply enumerates all possible options in which n jobs can be split into k non-empty groups. In a scenario where the groups are distinguishable, we need to enumerate integer *compositions* of n with k parts. According to [28], a composition of an integer n made of k summands is a sequence $(z_1, z_2, \ldots, z_k)$ of positive integers such that $n = z_1 + z_2 + \cdots + z_k$. The total number of compositions $C_n^{(k)}$ in exactly k summands is given by $\binom{n-1}{k-1}$, which can be approximated as $\frac{n^{k-1}}{(k-1)!}$. For a given composition, an optimal permutation of jobs is found by Step 3 of Algorithm Generate. The running time of Step 3 can be estimated as $O\,(T(W) + n \log n)$, where $T(W)$ denotes the running time needed to compute all positional weights $W^{[x]}(r)$, $1 \leq r \leq n^{[x]}$, $1 \leq x \leq k$. Thus, the following statement holds.

Theorem 1 *Algorithm Generate solves an instance of problem* $1\,|Combi, RMP\,(k-1)|\,F$ *in* $O\left((T\,(W) + n \log n)\,\frac{n^{k-1}}{(k-1)!}\right)$ *time, where* $T(W)$ *denotes the time needed to compute all positional weights* $W^{[x]}(r)$, $1 \leq r \leq n^{[x]}$, $1 \leq x \leq k$, *for a given composition of* n.

To determine the optimal solution for the general problem $1\,|Combi, RMP|\,F$, all options associated with Decisions 1-3 must be enumerated. For a known k, $1 \leq k \leq K+1$, the number of ways to select $k-1$ RMPs and fix their order is equal to the number of $(k-1)\,-$arrangements of K elements, i.e., the number of permutations of $k-1$ elements selected from K possibilities, which is equal to $\binom{K}{k-1}\,(k-1)!$. Trying all possible values of k, $1 \leq k \leq K+1$, the total number of options can be approximated by $\sum_{k=1}^{K+1} \binom{K}{k-1}\,(k-1)!$.

Since Algorithm Generate requires $O\left(\left(T\left(W\right)+n\log n\right)\frac{n^{k-1}}{(k-1)!}\right)$ time for a given k, $1\leq k\leq K+1$, the total running time required to solve problem $1\left|Combi,RMP\right|F$ can be approximated by

$$\begin{aligned}\sum_{k=1}^{K+1}\binom{K}{k-1}n^{k-1}\left(T\left(W\right)+n\log n\right) &\leq \left(\sum_{k=1}^{K+1}\binom{K}{k-1}\right)\left(\sum_{k=1}^{K+1}n^{k-1}\right)\left(T\left(W\right)+n\log n\right)\\ &= \left(\sum_{k=0}^{K}\binom{K}{k}\right)\left(\sum_{k=0}^{K}n^{k}\right)\left(T\left(W\right)+n\log n\right)\\ &= 2^{K}\left(\frac{n^{K+1}-1}{n-1}\right)\left(T\left(W\right)+n\log n\right)\\ &= O\left(n^{K}\left(T\left(W\right)+n\log n\right)\right).\end{aligned}$$

Recall from Sections 4.1 and 4.2 that for the problems of minimizing the makespan and the total flow time, the required positional weights $W^{[x]}\left(r\right)$, $1\leq r\leq n^{[x]}$, $1\leq x\leq k$, can be computed in $T\left(W\right)=O\left(n\right)$ and $T\left(W\right)=O\left(n^{2}\right)$ time, respectively. It follows that an optimal solution to problem $1\left|Combi,RMP\right|C_{\max}$ can be found in $O\left(n^{K}\left(n+n\log n\right)\right)=O\left(n^{K+1}\log n\right)$ time, and an optimal solution to problem $1\left|Combi,RMP\right|\sum C_{j}$ can be found in $O\left(n^{K}\left(n^{2}+n\log n\right)\right)=O\left(n^{K+2}\right)$ time.

Below we provide a numerical example, in which we solve the problems of minimizing the makespan and the total flow time for the situation discussed in Example 1. We present the solution for known outcomes of Decisions 1-3 and a given composition of n.

Example 2: Eight jobs must be processed on a single machine. The normal processing times of the jobs, after renumbering them in LPT order are:

$$p_{1}=8,\ p_{2}=7,\ p_{3}=6,\ p_{4}=6,\ p_{5}=4,\ p_{6}=2,\ p_{7}=1,\ p_{8}=1.$$

The number of jobs in each of the three groups is known in advance as

$$n^{[1]}=3,\ n^{[2]}=2,\ n^{[3]}=3.$$

The values of the parameters for the machine and the operators as defined in Example 1 are also known and given by

$$\begin{aligned}d'_{m} &= 0.1,\ d''_{m}=0.15,\ d'_{w}=0.2,\ l'_{w}=-0.15,\ d''_{w}=0.15,\ l''_{w}=-0.1,\ l'''_{w}=-0.02,\\ \alpha' &= 2,\ \beta'=5,\ \beta''=6.\end{aligned}$$

Using the formulae provided in Table 1, the functions for the positional factors reduce to

$$\begin{aligned}g^{[1]}\left(r\right) &= \left(d'_{m}+1\right)^{r-1}r^{d'_{w}+l'_{w}}=\left(1.1\right)^{r-1}r^{0.05},\ 1\leq r\leq 3;\\ g^{[2]}\left(r\right) &= \left(d''_{m}+1\right)^{r-1}\left(n^{[1]}+r\right)^{l'_{w}}r^{d'_{w}}=\left(1.15\right)^{r-1}\frac{r^{0.2}}{\left(r+3\right)^{0.15}},\ 1\leq r\leq 2;\\ g^{[3]}\left(r\right) &= \left(d''_{m}+1\right)^{n^{[2]}+r-1}r^{d''_{w}+l''_{w}}=\left(1.15\right)^{r+1}r^{0.05},\ 1\leq r\leq 3,\end{aligned}$$

and all required input parameters are computed as

$$\begin{aligned}&g^{[1]}\left(1\right)=1.00,\ g^{[1]}\left(2\right)=1.14,\ g^{[1]}\left(3\right)=1.28; && a_{1}^{[1]}=0.15;\\ &g^{[2]}\left(1\right)=0.81,\ g^{[2]}\left(2\right)=1.04; && a_{1}^{[2]}=-0.15,\ a_{2}^{[2]}=0.20;\\ &g^{[3]}\left(1\right)=1.32,\ g^{[3]}\left(2\right)=1.57,\ g^{[3]}\left(3\right)=1.85; && a_{1}^{[3]}=-0.02,\ a_{2}^{[3]}=0.13,\ a_{3}^{[3]}=0.20.\end{aligned}$$

Group 1	$B^{[1]}(1,1) = 1.00;$
	$B^{[1]}(1,2) = 1.17, \quad B^{[1]}(2,2) = 1.14;$
	$B^{[1]}(1,3) = 1.40, \quad B^{[1]}(2,3) = 1.36, \quad B^{[1]}(3,3) = 1.28;$
Group 2	$B^{[2]}(1,1) = 0.81;$
	$B^{[2]}(1,2) = 0.98, \quad B^{[2]}(2,2) = 1.04;$
Group 3	$B^{[3]}(1,1) = 1.32;$
	$B^{[3]}(1,2) = 1.74, \quad B^{[3]}(2,2) = 1.57;$
	$B^{[3]}(1,3) = 2.38, \quad B^{[3]}(2,3) = 2.16, \quad B^{[3]}(3,3) = 1.85.$

Table 2: Values of $B^{[x]}(u,r), 1 \leq u \leq r \leq n^{[x]}, 1 \leq x \leq 3$, for Example 2

and

$$\alpha_1^{[1]} = 2,\ \beta^{[1]} = 5,\ \alpha_1^{[2]} = \alpha_2^{[2]} = 0,\ \beta^{[2]} = 6.$$

Notice that while solving this example, we have computed all values with high precision, but here and below for the ease of presentation we report them rounded to two decimal places. Applying (7), all values of $B^{[x]}(u,r)$, $1 \leq u \leq r \leq n^{[x]}$, $1 \leq x \leq 3$, are computed and are presented in Table 2. For the problem of minimizing the makespan, we will only need to use the values given in the last row of each block in Table 2, whereas for the problem of minimizing the total flow time we will require all of them.

Applying (10), all values of $b_v^{[x]}$, $1 \leq v \leq x-1$, $1 \leq x \leq 3$, are computed as

$$\begin{aligned} b_1^{[2]} &= (-0.15)(0.98 + 1.04) = -0.30; \\ b_1^{[3]} &= (-0.02)(2.38 + 2.16 + 1.85) = -0.13, \\ b_2^{[3]} &= (0.13)(2.38 + 2.16 + 1.85) = 0.83. \end{aligned}$$

Applying (13), all values of $E^{[v,x]}, 1 \leq v \leq x, 1 \leq x \leq 3$, are computed as

$$\begin{aligned} E^{[1,1]} &= 1.00; \\ E^{[1,2]} &= b_1^{[2]} = -0.30,\ E^{[2,2]} = 1.00; \\ E^{[1,3]} &= b_1^{[3]} + b_1^{[2]} b_2^{[3]} = -0.38,\ E^{[2,3]} = b_2^{[3]} = 0.83,\ E^{[3,3]} = 1.00. \end{aligned}$$

Finally, applying (18), the positional weights $W^{[x]}(r), 1 \leq r \leq n^{[x]}, 1 \leq x \leq 3$, for the problem of minimizing the makespan can be rewritten as

$$\begin{aligned} W^{[1]}(r) &= B^{[1]}(r,3)\left(\left(1 + \alpha_1^{[1]} + \alpha_1^{[2]}\right) E^{[1,1]} + \left(1 + \alpha_2^{[2]}\right) E^{[1,2]} + E^{[1,3]}\right),\ 1 \leq r \leq 3; \\ W^{[2]}(r) &= B^{[2]}(r,2)\left(\left(1 + \alpha_2^{[2]}\right) E^{[2,2]} + E^{[2,3]}\right),\ 1 \leq r \leq 2; \\ W^{[3]}(r) &= B^{[3]}(r,3)\left(E^{[3,3]}\right),\ 1 \leq r \leq 3, \end{aligned}$$

and their values computed as

$$\begin{aligned} W^{[1]}(1) &= 3.23,\ W^{[1]}(2) = 3.15,\ W^{[1]}(3) = 2.96; \\ W^{[2]}(1) &= 1.80,\ W^{[2]}(2) = 1.90; \\ W^{[3]}(1) &= 2.38,\ W^{[3]}(2) = 2.16,\ W^{[3]}(3) = 1.85. \end{aligned}$$

To solve problem $1\,|Combi, RMP(k-1)|\,C_{\max}$, the computed positional weights are sorted in non-decreasing order and stored in list $L' := (\gamma'_1, \gamma'_2, \ldots, \gamma'_8)$. The constant term $\Gamma(3)$, can be computed as $\Gamma(3) = \beta^{[1]} + \beta^{[2]} = 11.00$. The resulting optimal schedule $S'(3)$ is associated with a permutation $\mu^* = \left(\mu^{[1]}, \mu^{[2]}, \mu^{[3]}\right)$; all relevant computation is presented in Table 3.

Next, for the problem of minimizing the total flow time, we also must additionally compute the values of $G^{[v,x]}$, $1 \le v \le x-1$, $1 \le x \le 3$. Using (19) and the values obtained in Table 2, we compute

$$G^{[1,2]} = 5.58,\ G^{[1,3]} = 8.78,\ G^{[2,3]} = 4.43.$$

Applying (20), the positional weights $W^{[x]}(r), 1 \le r \le n^{[x]}, 1 \le x \le 3$, for the problem of minimizing the total flow time can be rewritten as

$$\begin{aligned}
W^{[1]}(r) &= B^{[1]}(r,3)\left(G^{[1,2]}E^{[1,1]} + G^{[1,3]}E^{[1,1]} + G^{[2,3]}E^{[1,2]}\right) + \sum_{u=r}^{3} B^{[1]}(r,u),\ 1 \le r \le 3;\\
W^{[2]}(r) &= B^{[2]}(r,2)\left(G^{[2,3]}E^{[2,2]}\right) + \sum_{u=r}^{2} B^{[2]}(r,u),\ 1 \le r \le 2;\\
W^{[3]}(r) &= \sum_{u=r}^{3} B^{[3]}(r,u),\ 1 \le r \le 3,
\end{aligned}$$

and their values computed as

$$\begin{aligned}
W^{[1]}(1) &= 21.72,\ W^{[1]}(2) = 20.16,\ W^{[1]}(3) = 17.91;\\
W^{[2]}(1) &= 6.14,\ W^{[2]}(2) = 5.64;\\
W^{[3]}(1) &= 5.44,\ W^{[3]}(2) = 3.73,\ W^{[3]}(3) = 1.85.
\end{aligned}$$

To solve problem $1\,|Combi, RMP(k-1)|\,\sum C_j$, the computed positional weights are sorted in non-decreasing order and stored in list $L'' := (\gamma''_1, \gamma''_2, \ldots, \gamma''_n)$. The constant term $\Gamma(3)$, can be computed as $\Gamma(3) = \beta^{[1]}n^{[2]} + \left(\beta^{[1]} + \beta^{[2]}\right)n^{[3]} = 43.00$. The resulting optimal schedule $S''(3)$ is associated with a permutation $\varphi^* = \left(\varphi^{[1]}, \varphi^{[2]}, \varphi^{[3]}\right)$; all relevant computation is presented in Table 3.

6 Extension to Parallel Machines

Problem $1\,|Combi, RMP|\,\sum C_j$ can be further extended to a setting with uniform parallel machines, without many technical difficulties. This is done similarly to [2], where a version of the problem with pure positional effects is considered. For a problem with $m \ge 1$ machines, suppose that machine M_i, $1 \le i \le m$, has a speed s_i, and the number of jobs to be scheduled on M_i is equal to $h^{[i]}$, $1 \le i \le m$, where $\sum_{i=1}^{m} h^{[i]} = n$. Given a total of K RMPs to choose from, assume that it is known which $k-1$, $1 \le k \le K+1$, RMPs are selected and how they are allocated to each machine. As a result, a total of $m+k-1$ groups are generated across all machines. In particular, if the $h^{[i]}$ jobs on a machine M_i are divided into $k^{[i]}$, $1 \le i \le m$, groups we will have $\sum_{i=1}^{m} k^{[i]} = m+k-1$. It is convenient to associate group x, $1 \le x \le k^{[i]}$, on a machine $M_i, 1 \le i \le m$, with the index $[i,x]$; for example, with this notation the number of jobs scheduled in a group $[i,x]$ is equal to $n^{[i,x]}$, where $\sum_{x=1}^{k^{[i]}} n^{[i,x]} = h^{[i]}$.

j	p_j	γ'_j	μ^*	$\gamma'_j p_j$	γ''_j	φ^*	$\gamma''_j p_j$
		Makespan			Flow time		
1	8	1.80	$\mu^{[2]}(1)$	14.36	1.85	$\varphi^{[3]}(3)$	14.78
2	7	1.85	$\mu^{[3]}(3)$	12.93	3.73	$\varphi^{[3]}(2)$	26.12
3	6	1.90	$\mu^{[2]}(2)$	11.39	5.44	$\varphi^{[3]}(1)$	32.66
4	6	2.16	$\mu^{[3]}(2)$	12.94	5.64	$\varphi^{[2]}(2)$	33.82
5	4	2.38	$\mu^{[3]}(1)$	9.53	6.14	$\varphi^{[2]}(1)$	24.56
6	2	2.96	$\mu^{[1]}(3)$	5.93	17.91	$\varphi^{[1]}(3)$	35.83
7	1	3.15	$\mu^{[1]}(2)$	3.15	20.16	$\varphi^{[1]}(2)$	20.16
8	1	3.23	$\mu^{[1]}(1)$	3.23	21.72	$\varphi^{[1]}(1)$	21.72
		Total		73.46	Total		209.65
$C_{\max}(S'(3)) = 73.46 + 11.00 = 84.46;$							
$\sum C_j(S''(3)) = 209.65 + 43.00 = 252.65;$							
$\mu^* = (8,7,6,1,3,5,4,2)$; $\varphi^* = (8,7,6,5,4,3,2,1)$							

Table 3: Calculation of the optimal value of the makespan and the optimal value of the total flow time for the problem outlined in Example 2

For a given instance, computing the total flow time on a particular machine M_i, $1 \leq i \leq m$, is essentially a single machine problem $1\,|Combi, RMP(k-1)|\sum C_j$, that can be solved as described in Section 4.2. The quantities $W^{[x]}(r)$ as defined earlier for a single machine by (20) can be redefined as $W^{[i,x]}(r)$, with the group index $[x]$ being replaced by $[i,x]$, so that an explicit reference can be made to the machine a particular group is associated with. Additionally, the factors $W^{[i,x]}(r)$, will also incorporate the speed $s_i, 1 \leq i \leq m$, of a machine, as shown in [2]. As a result, the problem of minimizing the total flow time can be reduced to minimizing a problem of the form $\sum_{i=1}^{m}\sum_{x=1}^{k^{[i]}}\sum_{r=1}^{n^{[i,x]}} W^{[i,x]}(r)\, p_{\pi^{[i,x]}(r)} + \Gamma(m,k)$. We refrain from presenting the exact expressions for $W^{[i,x]}(r)$, $1 \leq r \leq n^{[i,x]}$, $1 \leq x \leq k^{[i]}$, $1 \leq i \leq m$, and $\Gamma(m,k)$; these expressions can be derived from their single-machine analogues presented in Section 4.2.

Once the positional weights are found, the resulting problem can be solved by running a version of Algorithm Generate. In Step 2 of Algorithm Generate, generating all possible values for $n^{[i,x]}$, $1 \leq x \leq k^{[i]}$, $1 \leq i \leq m$, requires enumeration of integer compositions of n with $m+k-1$ parts; the number of these options is $\binom{n-1}{m+k-2}$ which can be approximated by $\frac{n^{m+k-2}}{(m+k-2)!}$. For a given instance of the problem, the number of positional weights $W^{[i,x]}(r)$ to be computed and sorted is no more than those in the single machine case, thus, Step 3 of the algorithm still requires $O(n^2)$ time for each instance. As a result, for the parallel machine case Algorithm Generate requires $O\left(\frac{n^{m+k}}{(m+k-2)!}\right)$ time. The number of iterations of Algorithm Generate that must be run in order to determine the optimal solution for the general problem is computed as follows.

For a known k, $1 \leq k \leq K+1$, selecting $k-1$ of K available RMPs and taking all permutations over $k-1$ RMPs can be done in $\binom{K}{k-1}(k-1)!$ ways. Distributing $m+k-1$ groups over m machines can be done by generating all compositions of $m+k-1$ into exactly m positive summands; the number of these options is $\binom{m+k-2}{m-1} = \frac{(m+k-2)!}{(m-1)!(k-1)!}$. Thus, for a given k, the total number of options to be numerated is $\binom{K}{k-1}\frac{(k+m-2)!}{(m-1)!}$. Trying all possible values of k, $1 \leq k \leq K+1$, the overall running time required for minimizing the total flow time on

$m \geq 1$ uniform parallel machines can be approximated by $\sum_{k=1}^{K+1} \binom{K}{k-1} \frac{n^{m+k}}{(m-1)!} = O\left(n^{m+K+1}\right)$. This is consistent with the previously found estimate of $O\left(n^{K+2}\right)$ valid for $m = 1$.

7 Some Reduced Models

In this section, we explore single machine models, which can be expressed as special cases of the general problems $1\,|Combi, RMP|\,C_{\max}$ and $1\,|Combi, RMP|\,\sum C_j$. We look at their simplified versions with effects less general than those given by (4) and (5), and possibly with additional simplifications. The main purpose of this section, is to verify whether the running times of the solution algorithms given in Section 5 for the general problems $1\,|Combi, RMP|\,C_{\max}$ and $1\,|Combi, RMP|\,\sum C_j$ can be reduced for their simplified counterparts.

We start with the problems studied in [12] and [35], both of which look at a simple combined effect of the form (2). The former paper considers a problem in which the system undergoes time-dependent learning, positional polynomial deterioration and a single start-time dependent maintenance period, while the latter considers a problem in which the system undergoes time-dependent deterioration, positional polynomial learning and K identical start-time dependent maintenance periods, all of which must be run. Both these models assume that the effects are group-independent and after an RMP, the learning advantages are lost and both the operator and the machine are brought to the original conditions; these assumptions can hardly be justified. Notice that for both problems, Decisions 1-3 need not be taken, since the RMPs are identical and their number to be included in a schedule is already known in advance. As a result, both these problems can be written as special cases of problem $1\,|Combi, RMP\,(k-1)|\,F$ with the parameters $g^{[x]}(r) = r^b$, and $a_1^{[x]} = a_2^{[x]} = ... = a_{x-1}^{[x]} = 0$, $a_x^{[x]} = a$, for all x, $1 \leq x \leq K+1$, and $\alpha_1^{[x]} = \alpha_2^{[x]} = ... = \alpha_{x-1}^{[x]} = 0$, $\alpha_x^{[x]} = \alpha$, and $\beta^{[x]} = \beta$, for all x, $1 \leq x \leq K$. For the former case studied by [12], we have the additional conditions $K = 1$, $b > 0$, and $a < 0$, while for the latter we have $b < 0$ and $a > 0$. For both models, the published running times are given as $O\left(n^{K+1} \log n\right)$, for both problems of minimizing the makespan and of minimizing the total flow time. However, the correct running time required to solve the problem of minimizing the total flow time must be $O\left(n^{K+2}\right)$, since the time needed for computation of the positional weights must be taken into account. Thus, solving these rather special problems takes as much time as to solve the general problems $1\,|Combi, RMP|\,C_{\max}$ and $1\,|Combi, RMP|\,\sum C_j$, respectively.

We notice, however, that a minor reduction in the running time can be achieved if a and b are of the same sign. In this case, for the problem of minimizing the makespan the resulting positional weights

$$W^{[x]}(r) = \begin{cases} r^b \prod\limits_{i=r+1}^{n^{[x]}} \left(1 + ai^b\right)(1+\alpha), & 1 \leq r \leq n^{[x]},\ 1 \leq x \leq K; \\ r^b \prod\limits_{i=r+1}^{n^{[x]}} \left(1 + ai^b\right) & 1 \leq r \leq n^{[x]},\ x = K+1, \end{cases}$$

are monotonically ordered in each group, and thus, sorting all positional weights in a non-decreasing order requires $O\left(n \min\left\{K, \log n\right\}\right)$ time. As a result, Step 3b of Algorithm Generate requires $O\left(n \min\left\{K, \log n\right\}\right)$ time instead of $O\left(n \log n\right)$ time and the overall running time needed to solve the problem of the problem can be given as $O\left(\left(n + n \min\left\{K, \log n\right\}\right) \frac{n^K}{(K)!}\right) = O\left(n^{K+1} \min\left\{K, \log n\right\}\right)$. For the problem of minimizing the total flow time, in which a and

b are of the same sign, a similar observation can also be made. However, no running time reduction is possible, since for this objective function Step 3a of Algorithm Generate still requires $O\left(n^2\right)$ time.

Hence, it can be seen that as long as a model incorporates a combined time-dependent and a positional effect, even the simplest versions of the general problems $1\left|Combi, RMP\right| C_{\max}$ and $1\left|Combi, RMP\right| \sum C_j$ require $O\left(n^{K+1}\log n\right)$ and $O\left(n^{K+2}\right)$ time, respectively. However, if either a pure positional effect or a pure time-dependent effect is considered, faster solution algorithms are possible.

7.1 Pure Positional Effects

In this section, we discuss problems with a pure positional effect, so that for known outcomes of Decisions 1-3, the actual processing time of a job j scheduled in position r of a group x, $1 \leq x \leq k$, is given by

$$p_j^{[x]}(r) = p_j g^{[x]}(r), \ 1 \leq r \leq n, \ 1 \leq x \leq k.$$

Similar to the full model introduced in Section 2, the positional factors $g^{[x]}(r)$ are non-monotone within a group and the RMPs can be of an arbitrary nature, with their durations given by (5). The actual processing time of a job is seen as affected by the following factors:

- the group a job is assigned to;
- the position of the job within its group;
- the number of jobs scheduled in each of the previous groups.

Let us denote problems of this type by $1\left|Posi, RMP\right| F$, where $F \in \{C_{\max}, \sum C_j\}$. Problem $1\left|Posi, RMP\right| F$ can be seen as a special case of problem $1\left|Combi, RMP\right| F$. If Decisions 1-3 are taken in advance, denote the resulting problem as $1\left|Posi, RMP\left(k-1\right)\right| F$. The latter problem can be written in the form of problem $1\left|Combi, RMP\left(k-1\right)\right| F$ with $a_1^{[x]} = a_2^{[x]} = ... = a_{x-1}^{[x]} = a_x^{[x]} = 0$, for all x, $1 \leq x \leq k$. To solve problem $1\left|Posi, RMP\left(k-1\right)\right| C_{\max}$, the required positional weights can be obtained by making relevant substitutions in (18), so that we have

$$W^{[x]}(r) = \begin{cases} \left(1 + \sum_{w=x}^{k-1} \alpha_x^{[w]}\right) g^{[x]}(r), & 1 \leq r \leq n^{[x]}, \ 1 \leq x \leq k-1, \\ g^{[x]}(r) & 1 \leq r \leq n^{[x]}, \ x = k. \end{cases} \tag{22}$$

Notice that all positional weights $W^{[x]}(r)$, $1 \leq r \leq n^{[x]}$, $1 \leq x \leq k$, can be computed by (22) in $T(W) = O(n)$ time. Similarly, to solve problem $1\left|Posi, RMP\left(k-1\right)\right| \sum C_j$, the required positional weights can be obtained by making relevant substitutions in (20), so that we have

$$W^{[x]}(r) = \begin{cases} \left(\sum_{v=x+1}^{k} n^{[v]} \left(1 + \sum_{w=x}^{v-1} \alpha_x^{[w]}\right) + \left(n^{[x]} - r + 1\right)\right) g^{[x]}(r), & 1 \leq r \leq n^{[x]}, \\ & 1 \leq x \leq k-1, \\ \left(n^{[x]} - r + 1\right) g^{[x]}(r), & 1 \leq r \leq n^{[x]}, \\ & x = k. \end{cases} \tag{23}$$

Again, all positional weights $W^{[x]}(r)$, $1 \leq r \leq n^{[x]}$, $1 \leq x \leq k$, can be computed by (23) in $T(W) = O(n)$ time. Thus, each of the problems $1|Posi, RMP(k-1)|C_{\max}$ and $1|Posi, RMP(k-1)|\sum C_j$ can be solved by Algorithm Generate in $O\left(\frac{n^k \log n}{(k-1)!}\right)$ time; see Theorem 1. Trying all possible options for Decisions 1-3, an optimal solution to each of the problems $1|Posi, RMP|C_{\max}$ and $1|Posi, RMP|\sum C_j$ can be found in $O\left(n^{K+1} \log n\right)$ time.

It can be noted that although there is no change in status for the problem of minimizing the makespan, the problem of minimizing the total flow time is solved faster for a model with a pure positional effect. This speed up is possible because the time taken to compute the positional weights $W^{[x]}(r)$, $1 \leq r \leq n^{[x]}$, $1 \leq x \leq k$, for problem $1|Posi, RMP(k-1)|\sum C_j$ is $T(W) = O(n)$, as opposed to $T(W) = O\left(n^2\right)$ required for problem $1|Combi, RMP(k-1)|\sum C_j$.

A considerable reduction in the running time is possible for the problem of minimizing the makespan, if the positional factors $g^{[x]}(r)$ are known to be sorted in a non-decreasing order within each group, which, e.g., happens if a deterioration effect is in place, and the RMPs are machine maintenance periods that improve the processing conditions. Moreover, the positional factors should not be dependent on the number of jobs scheduled in previous groups. Several versions of the resulting problem are studied in [1], where efficient algorithms, with running times ranging from $O(n \log n)$ time to $O\left(n^3\right)$ time, are delivered. Notice that these running times do not depend on K, which is achieved because the authors develop a method able to compute the optimal number of jobs in each group on the fly, without generating all possible options.

On the other hand, a similar speed up is not possible for the problem of minimizing the total flow time, even if a simplified model as above is considered. As an illustration, see an algorithm in [32], which achieves a running time of $O\left(n^{K+1} \log n\right)$ for the problem of minimizing the total flow time for a model with a group-independent polynomial deterioration effect and K identical maintenance periods that have start-time dependent durations. In fact, the problem in [32] can be written as a special case of our problem $1|Posi, RMP|\sum C_j$, with $g^{[x]}(r) = r^a$, $a > 0$, and $\alpha_1^{[x]} = \alpha_2^{[x]} = ... = \alpha_{x-1}^{[x]} = 0$, $\alpha_x^{[x]} = \alpha$, $\beta^{[x]} = \beta$, for all x, $1 \leq x \leq K$.

7.2 Pure Time-Dependent Effects

In this section, we discuss problems with a pure time-dependent effect, so that for known outcomes of Decisions 1-3, the actual processing time of a job j scheduled in position r of a group x, $1 \leq x \leq k$, is given by

$$p_j^{[x]}(r) = p_{\pi^{[x]}(r)} + a_1^{[x]} F_1 + a_2^{[x]} F_2 + ... + a_{x-1}^{[x]} F_{x-1} + a_x^{[x]} F_{(x,r-1)},\ 1 \leq r \leq n,\ 1 \leq x \leq k. \quad (24)$$

Similar to the full model introduced in Section 2, the rates $a_1^{[x]}, a_2^{[x]}, \ldots, a_x^{[x]}$ can be of an arbitrary sign and the RMPs can be of an arbitrary nature with their durations given by (5). The actual processing time of a job is seen as affected by the following factors:

- the group a job is assigned to;
- the number of jobs scheduled in each of the previous groups;
- the time elapsed before processing the job within its group;
- the time elapsed in each of the previous groups.

Let us denote problems of this type by $1\,|Time, RMP|\,F$, where $F \in \{C_{\max}, \sum C_j\}$. Problem $1\,|Time, RMP|\,F$ can be seen as a special case of problem $1\,|Combi, RMP|\,F$. If Decisions 1-3 are taken in advance, denote the resulting problem as $1\,|Time, RMP\,(k-1)|\,F$. The latter problem can be written in the form of problem $1\,|Combi, RMP\,(k-1)|\,F$ with $g^{[x]}(r) = 1,\ 1 \leq r \leq n,\ 1 \leq x \leq k$. To solve problem $1\,|Time, RMP\,(k-1)|\,C_{\max}$, the required positional weights can be obtained by making relevant substitutions in (18), so that we have

$$W^{[x]}(r) = \left(1 + a_x^{[x]}\right)^{n^{[x]}-r} \left(\sum_{v=x}^{k-1} \left(1 + \alpha^{[v]}\right) E^{[x,v]} + E^{[x,k]}\right),\ 1 \leq r \leq n^{[x]},\ 1 \leq x \leq k, \quad (25)$$

where the quantities $E^{[v,x]}$, $1 \leq v \leq x-1$, $1 \leq x \leq k$, are given by (13), while the quantities $b_v^{[x]}$ defined by (10) reduce to

$$b_v^{[x]} = \frac{a_v^{[x]}}{a_x^{[x]}} \left(\left(1 + a_x^{[x]}\right)^{n^{[x]}} - 1\right),\ 1 \leq v \leq x-1,\ 1 \leq x \leq k.$$

Notice that all positional weights $W^{[x]}(r)$, $1 \leq r \leq n^{[x]}$, $1 \leq x \leq k$, can be computed by (25) in $T(W) = O(n)$ time. Similarly, to solve problem $1\,|Time, RMP\,(k-1)|\,\sum C_j$, the required positional weights can be obtained by making relevant substitutions in (20), so that we have

$$W^{[x]}(r) = \left(1 + a_x^{[x]}\right)^{n^{[x]}-r} \sum_{v=x+1}^{k} \sum_{w=x}^{v-1} G^{[w,v]} E^{[x,w]} + \sum_{u=r}^{n^{[x]}} B^{[x]}(r,u),\ 1 \leq r \leq n^{[x]},\ 1 \leq x \leq k, \quad (26)$$

where the quantities $G^{[v,x]}$, $1 \leq v \leq x-1$, $1 \leq x \leq k$, are given by (19), while the quantities $B^{[x]}(r,u)$ defined by (7) reduce to

$$B^{[x]}(u,r) = \left(1 + a_x^{[x]}\right)^{r-u},\ 1 \leq u \leq r \leq n^{[x]},\ 1 \leq x \leq k.$$

For a fixed x, $1 \leq x \leq k$, the difference $r-u$ takes at most $n^{[x]}-1$ values, so that at most $n^{[x]}-1$ distinct values of $B^{[x]}(u,r)$ need to be computed. Summing up for all x, we deduce that the number of all distinct values $B^{[x]}(u,r)$ to be found in order to compute the positional weights $W^{[x]}(r)$ is $O(n)$. As a result, all positional weights $W^{[x]}(r)$, $1 \leq r \leq n^{[x]}$, $1 \leq x \leq k$, can be computed by (26) in $T(W) = O(n)$ time.

Thus, each of the problems $1\,|Time, RMP\,(k-1)|\,C_{\max}$ and $1\,|Time, RMP\,(k-1)|\,\sum C_j$ can be solved by Algorithm Generate in $O\left(\frac{n^k \log n}{(k-1)!}\right)$ time; see Theorem 1. Trying all possible options for Decisions 1-3, an optimal solution to each of the problems $1\,|Time, RMP|\,C_{\max}$ and $1\,|Time, RMP|\,\sum C_j$ can be found in $O\left(n^{K+1} \log n\right)$ time.

It can be noted that although there is no change in status for the problem of minimizing the makespan, the problem of minimizing the total flow time is solved faster for a model with a pure time-dependent effect. This speed up is possible because the time taken to compute the positional weights $W^{[x]}(r)$, $1 \leq r \leq n^{[x]}$, $1 \leq x \leq k$, for problem $1\,|Time, RMP\,(k-1)|\,\sum C_j$ is $T(W) = O(n)$, as opposed to $T(W) = O\left(n^2\right)$ required for problem $1\,|Combi, RMP\,(k-1)|\,\sum C_j$.

Notice that the same running time of $O\left(n^{K+1} \log n\right)$ for the problem of minimizing the total flow time for a much simpler model is achieved in [32]. The authors consider a reduced

form of the effect defined by (24), in which the duration of previous groups do not affect the actual processing time of the current job, so that $a_1^{[x]} = a_2^{[x]} = ... = a_{x-1}^{[x]} = 0$, for all x, $1 \leq x \leq k$. Moreover, the authors study a group-independent deterioration effect, so that $a_x^{[x]} = a > 0$, $1 \leq x \leq k$, and the RMPs are strictly maintenance periods which have start-time dependent durations, i.e., $\alpha_1^{[x]} = \alpha_2^{[x]} = ... = \alpha_{x-1}^{[x]} = 0$, $\alpha_x^{[x]} = \alpha$, $\beta^{[x]} = \beta$, for all x, $1 \leq x \leq K$.

8 Conclusion

The paper addresses single machine scheduling problems to minimize the makespan (the maximum completion time) and the total flow time (the sum of the completion times). A very general model for changing processing times is introduced, in which the actual processing times of the jobs depend on the group a job is scheduled into and on the location of the jobs in those groups. The location reflects both the position of a job in the group and its start time relative to the beginning of the group. The durations of the rate-modifying activities in general depend on their start times. Unlike most other papers in which changing processing times are considered, we do not insist on monotone effects. The model introduced in this paper covers most previously known models, provided that the introduced effects are job-independent and the start-time effects are linear.

We reduce these generalized problems to linear assignment problems with product matrices, solvable by a matching algorithm, and present close form relations for computing the necessary input parameters, such as positional weights. The resulting running times are no worse than those earlier known for less general problems, however faster algorithms are available for the problems with pure positional or time-dependent effects to minimize the total flow time.

It should be noticed that many results discussed in this paper can be transferred to other objective functions with no major technical difficulties, since we have provided the formula for the completion time of an arbitrary job in the schedule.

A further extension of the main model of this paper would involve problems with changing times subject to job-dependent combined effects. For these models, the main algorithmic tool is expected to be a full form assignment problem.

Acknowledgement

The authors are grateful to the Associate Editor and to three anonymous referees for their comments aimed at clarifying the scope of this paper and at improving the presentation.

References

[1] Rustogi K, Strusevich VA. Single machine scheduling with general positional deterioration and rate-modifying maintenance. Omega 2012; 40:791–804.

[2] Rustogi K, Strusevich VA. Simple matching vs linear assignment in scheduling models with positional effects: A critical review. European Journal of Operational Research 2012; 222:393–407.

[3] Biskup D. A state-of-the-art review on scheduling with learning effects. European Journal of Operational Research 2008; 188:315–29.

[4] Janiak A, Kovalyov MY. Scheduling in a contaminated area: a model and polynomial algorithms. European Journal of Operational Research 2006; 173:125–32.

[5] Kuo W-H, Yang D-L. Minimizing the makespan in a single machine scheduling problem with a time-based learning effect. Information Processing Letters 2006; 97:64–7.

[6] Kuo W-H, Yang D-L. Minimising the total completion time in a single-machine scheduling problem with a time-dependent learning effect. European Journal of Operational Research, 2006; 174: 1184–90.

[7] Cheng TCE, Ding Q, Lin BMT. A concise survey of scheduling with time-dependent processing times, European Journal of Operational Research 2004; 152:1–13.

[8] Gawiejnowicz S. Time-Dependent Scheduling. Berlin: Springer; 2008.

[9] Janiak A, Krysiak T, Trela R. Scheduling problems with learning and ageing effects: A survey. Decision Making in Manufacturing Services 2011; 5:19–36.

[10] Wang J-B. A note on scheduling problems with learning effects and deteriorating jobs. International Journal of Systems Science 2006; 37:827–32.

[11] Yang D-L, Kuo W-H. Single-machine scheduling with both deterioration and learning effects. Annals of Operational Research 2009; 172:315–27.

[12] Yang S-J. Single-machine scheduling problems with both start-time dependent learning and position dependent aging effects under deteriorating maintenance consideration. Applied Mathematics and Computation 2010; 217:3321–9.

[13] Zhao C-L, Tang H-Y. Single machine scheduling with general job-dependent aging effect and maintenance activities to minimize makespan. Applied Mathematical Modelling 2010; 34:837–41.

[14] Yang S-J, Yang D-L. Minimizing the makespan on single-machine scheduling with aging effects and variable maintenance activities. Omega 2010; 38:528–33.

[15] Kubzin MA, Strusevich VA. Two-machine flow shop no-wait scheduling with machine maintenance. 4OR 2005; 3:303–13.

[16] Kubzin MA, Strusevich VA. Planning machine maintenance in two-machine shop scheduling. Operations Research 2006; 54:789–800.

[17] Mosheiov G, Sidney JB. Scheduling a deteriorating maintenance activity on a single machine. Journal of the Operational Research Society 2010; 61:882–7.

[18] Wang J-J, Wang J-B, Liu F. Parallel machines scheduling with a deteriorating maintenance activity. Journal of the Operational Research Society 2011; 62:1898–902.

[19] Hardy GH, Littlewood JE, Polya G. Inequalities. London: Cambridge University Press; 1934.

[20] Browne S, Yechiali U. Scheduling deteriorating jobs on a single processor. Operations Research, 1990; 38: 495–8.

[21] Cheng TCE, Ding Q, Kovalyov MY, Bachman A, Janiak A. Scheduling jobs with piecewise linear decreasing processing times. Naval Research Logistics 2003; 50:531–2.

[22] Ng CT, Cheng TCE, Bachman A, Janiak A, Three scheduling problems with deteriorating jobs to minimize the total completion time. Information Processing Letters 2002; 81:327–33.

[23] Mosheiov G. Scheduling problems with a learning effect. European Journal of Operational Research 2001; 132:687–93.

[24] Mosheiov G. A note on scheduling deteriorating jobs. Mathematical and Computer Modelling 2005; 41:883–6.

[25] Gordon VS, Potts CN, Strusevich VA, Whitehead JD. Single machine scheduling models with deterioration and learning: Handling precedence constraints via priority generation. Journal of Scheduling 2008; 11:357–70.

[26] Ji M, Cheng TCE. Scheduling with job-dependent learning effects and multiple rate-modifying activities. Information Processing Letters 2010; 110:460–3.

[27] Lee W-C, Wu C-C. A note on single-machine group scheduling problems with position-based learning effect. Applied Mathematical Modelling 2009; 33:2159–63.

[28] Flajolet P, Sedgewick R. Analytic Combinatorics. Cambridge: Cambridge University Press; 2009 (pp 39–46).

[29] Gawiejnowicz S. A note on scheduling on a single processor with speed dependent on a number of executed jobs. Information Processing Letters 1996; 57:297–300.

[30] Kuo W-H, Yang D-L. Single machine scheduling with past-sequence-dependent setup times and learning effects. Information Processing Letters 2007; 102:22–6.

[31] Okołowski D, Gawiejnowicz S. Exact and heuristic algorithms for parallel-machine scheduling with DeJong's learning effect. Computers and Industrial Engineering 2010; 59:272–9.

[32] Yang S-J, Yang D-L. Minimizing the total completion time in single-machine scheduling with ageing/deteriorating effects and deteriorating maintenance activities. Computers and Mathematics with Applications 2010; 60:2161–9.

[33] Conway RW, Maxwell WL, Miller LW. Theory of Scheduling. Reading (MA): Addison Wesley; 1967.

[34] Brucker P. Scheduling Algorithms. Guildford, Surrey: Springer; 2007.

[35] Yang S-J. Single-machine scheduling problems simultaneously with deterioration and learning effects under deteriorating multi-maintenance activities consideration. Computers & Industrial Engineering 2012; 62:271–5.