Economic Lot Scheduling with Resources in Parallel

ÇAĞRI HAKSÖZ*
Sabancı University

MICHAEL PINEDO†
Leonard N. Stern School of Business, New York University

Current Version: December 5, 2008.

Keywords: Economic Lot Scheduling, Parallel Resources, Multiple Knapsack, Bin Packing, Heuristics

Abstract

In this paper we consider the economic lot scheduling problem with m machines (or facilities) in parallel. There are n different types of items. Item j has a demand D_j per unit time, a holding cost h_j per unit time, and a setup cost K_j. The machines have different speeds. The speed of machine i is v_i. Machine i can produce item j at rate $v_i P_j$. We consider three different models. The objective in all three models is to find an assignment of items to machines that minimizes the total cost per unit time of the entire system. In the first model each machine operates according to a rotation or cyclical schedule and the cycle lengths of the rotation schedules of the m machines have to be the same. In the second model each machine again operates according to a rotation schedule, but the rotation schedules are allowed to have different cycle lengths. In the third model the machines do not have to operate according to rotation schedules. For each model we consider a number of special cases that provide some insights into the role each parameter plays. Based on the results of the special cases we formulate for each model a heuristic that can be applied to arbitrary instances. In the concluding remarks we discuss the significance of our results for problems that occur in practice.

*Assistant Professor of Operations Management, Faculty of Management, Orhanlı, Tuzla, 34956 Istanbul, Turkey. Phone:+90-216-483-9683. Email:cagrihaksoz@sabanciuniv.edu.

†Julius Schlesinger Professor of Operations Management, IOMS Dept, 44 W 4th St., New York, NY 10012. Phone:+1-212-998-0287.Email:mpinedo@stern.nyu.edu.

1 Introduction

The Economic Lot Scheduling Problem arises often in industry. The goal is to schedule lots (or batches) of one or more items on a machine in such a way that the total setup cost and the total holding (i.e., inventory carrying) cost per unit time is minimized.

In this paper we consider the Economic Lot Scheduling Problem with multiple machines in parallel. We assume that the machines are not identical and operate at different speeds. Each item must be assigned to one and only one machine and each item must be produced in a cyclic fashion. The problem is to determine which item has to be assigned to which machine.

The Economic Lot Scheduling Problem on parallel machines occurs in many industries. Consider, for example, a paper mill. Paper mills typically have several paper machines and each machine operates at its own speed. Many different types of paper (differentiated by the basis weight, grade, and finish of the paper) have to be produced on the various machines. The custom typically is to keep on producing any given type of item on the same machine with the machine operating according to a so-called rotation or cyclic schedule. This paper aims at obtaining insights on how setup costs and inventory carrying costs affect the assignments of items to machines. The goal is to develop priority rules for assigning items to machines. The priority rules should be easily implementable and useful for the real world. Such priority rules may depend on the cycle lengths as well as on whether or not the cycle lengths of the different machines are the same.

In what follows we consider three basic models. In the first model (Model I), we assume that each machine has to follow a rotation schedule, implying that all items that are assigned to any given machine have the same cycle length. We also assume that all the machines must have the same cycle length; this implies that all items on all machines have the same cycle length. In the second model (Model II), again each machine must follow a rotation schedule. However, the cycle lengths of the rotation schedules of the various machines are allowed to be different. In the third model (Model III), each machine can follow an arbitrary schedule, not necessarily a rotation schedule. It is clear that, given a fixed set of items, the minimum total cost in Model I is higher than the minimum total cost in Model II, which is again higher than the minimum total cost in Model III.

An enormous amount of research has been done on economic lot scheduling. For an overview of this research, see Elmaghraby (1978), Potts and van Wassenhove (1992), Muckstadt and Roundy (1993), Haase (1993), Kimms (1997), Drexl and Kimms (1997), Zipkin (2000), and Pinedo (2005).

A significant amount of research has been done on lot scheduling where items have to follow given routes within a network of facilities; these routes are often referred to as "gozinto" structures, see Muckstadt and Roundy (1993). Not as much research has been done on lot sizing in parallel facilities. Carreno (1990) considered the special case of the problem described here by assuming that the m different machines are identical, i.e., operate at the same speed. Jones and Inman (1996) presented some very practical algorithms for machines in parallel, again for the case when the machines have identical speeds. Kang, Malik, and Thomas (1999) studied a problem somewhat related to ours with sequence-dependent setup costs in a multi-period setting. Finally, Bollapragada and Rao (1999) considered m machines at different speeds. They assumed that the production of an item could be split arbitrarily and spread out over multiple machines. Each machine follows a rotation schedule and the cycle lenghts of the machines are allowed to be different. In a sense, their model is a continuous version of our Model II. They develop for their model a Mixed Integer Programming formulation with a concave objective function. Bollapragada and Rao also develop two more practical heuristic rules that are not based on the mathematical programming framework. Clearly, in their optimal solutions, items of any given type may be produced on more than one machine.

A significant amount of research in the scheduling literature has also focused on parallel machines with different speeds. It is often assumed that there is a finite number of jobs (n) which have to be processed on parallel machines that operate at different speeds. Such a parallel machine environment is in the finite scheduling literature often referred to as uniform machines, see Pinedo (2002). A job in a parallel machine scheduling model is similar to the processing of an item in this paper. Such a job in a scheduling model also has to be processed on just one of the machines without any interruptions.

Some of the results that we present in this paper turn out to be intuitive. For example, an intuitive explanation can be given for the fact that when loading the machines, one should always give a preference to the machine that is the slowest and start loading that one first. However, other results are less intuitive: when loading the machines (starting with the slowest one), which items should be assigned to the slower machines and which items to the faster? Which items to assign first to the slowest machine actually depends very much on the model under consideration; the priority rules are different for Models I, II, and III. We also focus on the differences between our heuristics for Model II and the heuristics proposed by Bollapragada and Rao for their continuous version of our Model II.

This paper is organized as follows. The second section contains the notation and preliminary results. The third, fourth and fifth sections focus on Models I, II, and III, respectively. Each section considers various special cases and presents a heuristic for the general case. In the last section we present our conclusions, discuss the usefulness of the results for scheduling in practice and describe areas that deserve more attention.

2 Preliminaries

We assume in all three models that there are m machines with speeds $v_1 \leq v_2 \leq \cdots \leq v_m$ and that the speed of the slowest machine is $v_1 = 1$. Let K_j denote the setup cost of item j, $j = 1, \ldots, n$. This setup cost does not depend on the machine. Let h_j denote the holding (inventory carrying) cost of item j per unit time and D_j the demand rate for item j. Let P_j denote the production rate of item j on machine 1; the production rate of item j on machine i is $P_{ij} = v_i P_j$. Let $\rho_j = D_j/P_j$ denote the proportion of time that machine 1 must be dedicated to the production of item j if item j is assigned to machine 1; let

$$\rho_{ij} = \frac{D_j}{P_{ij}} = \frac{\rho_j}{v_i}$$

denote the proportion of time machine i has to be dedicated to the production of item j if item j is assigned to machine i.

We first focus on Models I and II. In Model I, the variable x denotes the cycle length of each one of the machines (their cycle lengths are identical). In Model II, the variable x_i denotes the cycle length of machine i. In order to simplify notation, let

$$F_{ij} = \frac{1}{2} h_j D_j (1 - \rho_{ij}).$$

If item j is assigned to machine i and it is the only item on machine i, then its cycle length can be determined independently of the other machines and

$$x_{ij} = \sqrt{\frac{2K_j}{h_j D_j (1 - \rho_{ij})}} = \sqrt{\frac{K_j}{F_{ij}}},$$

(see, for example, Zipkin (2000) or Pinedo (2005), Chapter 7).

It is a well-known fact that the optimal cycle length of a rotation schedule, when there are n different items assigned to a single machine i with speed v_i, is

$$x_i = \sqrt{\sum_{j=1}^{n} K_j \Big(\sum_{j=1}^{n} \frac{h_j D_j}{2} (1 - \rho_{ij}) \Big)^{-1}} = \sqrt{\sum_{j=1}^{n} K_j \Big(\sum_{j=1}^{n} F_{ij} \Big)^{-1}},$$

(see, for example, Pinedo (2005)). The total average cost of machine i per unit time then is

$$Z_i = x_i \sum_{j=1}^{n} F_{ij} + \frac{1}{x_i} \sum_{j=1}^{n} K_j = 2 \sqrt{\sum_{j=1}^{n} F_{ij}} \sqrt{\sum_{j=1}^{n} K_j}.$$

The heuristics for the three models considered are different, i.e., the heuristic for the case in which the cycle lengths of the m machines are the same (Model I) is different from the heuristic for the case when the cycle lengths of the various machines are allowed to be different (Model II).

If the system is working at full capacity, i.e., if

$$\rho_1 + \rho_2 + \cdots + \rho_n = v_1 + v_2 + \cdots + v_m,$$

then there is a "fitting" or "packing" problem when searching for a combination of items that exactly covers the capacity of each machine (assuming an item cannot be produced on more than one machine). This packing problem is mathematically equivalent to the PARTITION, or SUBSET SUM problem, or the Multiple Knapsack problem, and is known to be NP-hard. So, if the system is working at full capacity, then there may not even be a feasible solution with every item produced on just one machine.

However, there are a number of scenarios in which the fitting or packing problem does not play a role. Consider the following three scenarios: (i) all the machines are operating well below their full capacity; (ii) there are many items that require very little machine time, i.e., items that have a ρ_j very close to zero; (iii) all ρ_j are identical to ρ. (In the last case, when all $\rho_j = \rho$ and the system is working at full capacity, then a feasible solution only exists provided each v_i is a multiple of ρ).

It can be shown that in any one of such scenarios the fitting or packing problem is not a real problem. We will focus in this paper on such cases in order to obtain some insights into the priority rules for scheduling the various items on the different machines. The rules obtained this way can

then be used for the development of heuristics for arbitrary instances.

In the real world, machines are often working at full capacity and there is indeed a preference not to produce the same item on two different machines. However, in practice the packing problem is typically not an issue because the quantities to be produced are somewhat adjustable and there are usually many items with very low production quantities.

Model III needs some additional discussion. In Model III, when an assignment of items is made to a machine, another problem can occur: An assignment of items to a machine may be such that there does not exist a feasible schedule of the items with each one of them having its own optimal cycle length. For example, suppose an item with a high ρ is put together on a machine with many small ρ items. Suppose the high ρ item requires a long cycle length (because of a high setup cost) and the small ρ items require very short cycle lengths (because of high inventory carrying costs). So the high ρ item requires long production runs during which it is not possible to do the short runs of the low ρ items. Given an assignment with such a combination of items on one machine, it may not be possible to find a schedule in which each item has its own optimal cycle length. However, this particular problem is not a major issue when the number of items is large and all the ρ's are relatively small. In our analysis of Model III, we assume therefore that once an assignment of items to machines has been made, it will be possible to find for each machine an optimal (or close to optimal) schedule with each item being produced according to its own optimal cycle length. We are mainly interested in the assignment problem that is part of Model III. There are two reasons why we are interested in the assignment problem assuming that for each machine an optimal schedule can be found once the assignment has been made: the first reason is that the solution of this assignment problem provides a useful lower bound for all three models. The second reason is that solving the assignment problem first and then scheduling each machine is an approach that is often used in industry. In what follows we refer to the assignment problem in Model III also as the Lower Bound Problem.

3 Rotation Schedules with Identical Cycle Lengths (Model I)

In this section we present heuristics for the case where the cycle lengths of the m machines are the same. This problem is somewhat similar to a well-known problem in combinatorial optimization called the multiple knapsack problem.

In order to obtain some insights, consider two machines and n items. The two machines operate at speeds v_a and v_b, where $v_a \leq v_b$. Let N_a (N_b) denote the set of items that are assigned to machine a (b) and let n_a (n_b) denote the number of different items in set N_a (N_b).

If the cycle length of both machines is x, then the total average cost per unit time is

$$\sum_{i \in N_a} \Big(\frac{1}{2} h_i (D_i x - \frac{D_i^2 x}{v_a P_i})\Big) + \sum_{i \in N_b} \Big(\frac{1}{2} h_i (D_i x - \frac{D_i^2 x}{v_b P_i})\Big) + \Big(\sum_{i \in N_a \cup N_b} K_i\Big)/x.$$

In order to determine the optimal cycle length x, we take the derivative of the total costs on both machines with respect to x, set that equal to zero, and obtain the equation

$$\frac{1}{x^2} \sum_{i \in N_a \cup N_b} K_i = \frac{1}{2} \sum_{i \in N_a \cup N_b} h_i D_i - \frac{1}{2v_a} \sum_{i \in N_a} h_i D_i \rho_i - \frac{1}{2v_b} \sum_{i \in N_b} h_i D_i \rho_i,$$

or

$$x = \Big(\frac{2 v_a v_b}{G} \sum_{i \in\ N_a \cup N_b} K_i\Big)^{\frac{1}{2}},$$

where

$$G = v_a v_b \sum_{i \in N_a \cup N_b} h_i D_i \ - \ v_b \sum_{i \in N_a} h_i D_i \rho_i \ - \ v_a \sum_{i \in N_b} h_i D_i \rho_i.$$

We now perform an interchange between a set of items that is originally assigned to machine a, say set R, with a set of items that is originally assigned to machine b, say set S. We assume that such an interchange is possible, i.e., that either

$$\sum_{j \in S} \rho_j = \sum_{j \in R} \rho_j,$$

or that the difference between the two sums is smaller than the current idle time on the machine

that receives the larger set. We take set $S \subseteq N_b$ from machine b and put it on machine a and take set $R \subseteq N_a$ from machine a and put it on machine b. The optimal cycle length may change and in order to determine the new optimal cycle length we must consider again the derivative of the total cost and set it equal to zero:

$$\frac{1}{x^2}\sum_{i\in N_a\cup N_b} K_i = \frac{1}{2}\Big(\sum_{i\in N_a\cup N_b} h_i D_i\Big) - \frac{1}{2v_a}\Big(\sum_{i\in (N_a\setminus R)\cup S} h_i D_i \rho_i\Big) - \frac{1}{2v_b}\Big(\sum_{i\in (N_b\setminus S)\cup R} h_i D_i \rho_i\Big).$$

So

$$x = \Big(\frac{2v_a v_b}{G^*}\sum_{i\in N_a\cup N_b} K_i\Big)^{\frac{1}{2}},$$

where

$$G^* = v_a v_b \sum_{i\in N_a\cup N_b} h_i D_i - v_b \sum_{i\in (N_a\setminus R)\cup S} h_i D_i \rho_i - v_a \sum_{i\in (N_b\setminus S)\cup R} h_i D_i \rho_i.$$

The average total cost before the interchange is

$$Z = \Big(\frac{2v_a v_b}{G}\sum_{i\in N_a\cup N_b} K_i\Big)^{\frac{1}{2}}\frac{1}{2}\Big(\sum_{i\in N_a\cup N_b} h_i D_i - \sum_{i\in N_a}\frac{h_i D_i \rho_i}{v_a} - \sum_{i\in N_b}\frac{h_i D_i \rho_i}{v_b}\Big)$$

$$= \Big(\frac{G}{2v_a v_b}\sum_{i\in N_a\cup N_b} K_i\Big)^{\frac{1}{2}},$$

whereas the average total cost after the pairwise interchange is

$$Z^* = \Big(\frac{G^*}{2v_a v_b}\sum_{i\in N_a\cup N_b} K_i\Big)^{\frac{1}{2}}.$$

The only difference in the total cost is due to the difference between G and G^*. If $G^* \leq G$, then $Z^* \leq Z$. So if

$$-v_b\sum_{j\in R} h_j D_j \rho_j - v_a \sum_{k\in S} h_k D_k \rho_k \geq -v_b \sum_{k\in S} h_k D_k \rho_k - v_a \sum_{j\in R} h_j D_j \rho_j,$$

or

$$v_a\Big(\sum_{k\in S} h_k D_k \rho_k - \sum_{j\in R} h_j D_j \rho_j\Big) \le v_b\Big(\sum_{k\in S} h_k D_k \rho_k - \sum_{j\in R} h_j D_j \rho_j\Big),$$

then $Z^* \le Z$. Since $v_a \le v_b$, it follows that the total cost is less when the set of items with the higher $\sum h_j D_j \rho_j$, i.e., set S, is assigned to the slower machine, i.e., machine a. If set R is empty and there is enough slack on the slower machine to accommodate set S, then it is better to move set S to the slower machine.

It makes intuitive sense that the setup cost K_j does not have an effect on which machine the item is assigned. The contribution of the setup cost to the total average cost per unit time does not depend on the machine, since all machines are operating with the same cycle length.

That items should be assigned preferably to the slowest machine is also intuitive. The inventory carrying cost on a slower machine will be lower than on a faster machine that operates with the same cycle length. The reason is clear: on a slower machine the buildup of inventory takes more time and the peak inventory is lower. One measure that is proportional to the inventory carrying cost for a particular item is the product of the inventory holding cost h_j and the average amount in inventory (which is half the peak inventory), i.e.,

$$\frac{1}{2} h_j (1 - \frac{\rho_j}{v_i}) D_j x.$$

The only term in this expression that depends on the machine to which the item is assigned is $-h_j D_j \rho_j x / v_i$. If the total costs have to be minimized, then this term should be maximized (because of the minus sign). In order to maximize the sum of these terms over all the items, the highest $h_j D_j \rho_j$ should be assigned to the slowest machine.

Before describing a general heuristic for this problem, we first consider three special cases, which lead to three different heuristics. After analyzing the three special cases, we discuss heuristics that are applicable to arbitrary instances of Model I.

The first case is the easiest, since it allows for a polynomial time solution. In this first case, we assume that $\rho_j = \rho$ for $j = 1, \ldots, n$.

Theorem 1. *If all $\rho_j = \rho$, then the total cost is minimized by loading the machines starting with the slowest one, followed by the second slowest one, etc., and by assigning the jobs to the machines in decreasing order of $h_j D_j$.*

The theorem can be shown easily through pairwise interchange arguments. The result implies that this special case can be solved in polynomial time.

In the second special case, we assume that all $h_j D_j$ are equal to a constant A_{hD}, and that the ρ_j may all be different. That is, for each item the product $h_j D_j$ takes the same value A_{hD}, but the h_j and the D_j may be different from item to item. Consider now, for example, three items with parameters $h_1 D_1 = h_2 D_2 = h_3 D_3 = A_{hD}$ and

$$\rho_2 = \rho_3 = \frac{1}{2}\rho_1.$$

It is clear that if $R = \{1\}$ and $S = \{2, 3\}$, then an interchange between sets R and S is always possible, since the two sets take exactly the same amount of machine time (on any machine). From the interchange argument it follows that as far as total costs are concerned the interchange does not make any difference. It is clear that in this second special case the objective is to first maximize the *utilization* of the slowest machine, then maximize the utilization of the second slowest machine, and so on. This problem is very similar to the well-known bin-packing problem in combinatorial optimization which is known to be strongly NP-hard. It can be shown easily (through a reduction from the 3-PARTITION problem) that our problem is strongly NP-Hard even when $h_j D_j = A_{hD}$ for $j = 1, \ldots, n$ (and arbitrary ρ_j).

This second special case may be regarded as a more general version of the bin-packing problem. The machines are the bins and the items have to be stored in the bins while minimizing the number of bins used. This problem is more general than the classical bin-packing problem, because the machines have different speeds. The problem is therefore equivalent to a bin-packing problem with bins of different sizes.

Many different heuristics have been developed for the bin-packing problem; these bin-packing heuristics can be applied to our second special case with only minor modifications. A well known bin packing heuristic is the First Fit Decreasing (FFD) heuristic, see Garey and Johnson (1979). According to this heuristic, the items are first ordered in decreasing order of ρ_j, i.e., according to Longest Processing Time first (LPT), and the machines are ordered in increasing order of their speeds. Then the items are taken one after another from the top of the list and put on the slowest machine that still has enough remaining capacity. However, a well-known worst case analysis of the FFD rule (see Gary and Johnson (1979)) is not applicable here, since that worst case analysis assumes bins of the same size, i.e., machines with identical speeds. It is easy to design for this

special case a more elaborate two-phase heuristic.

In the third special case, we assume that the product $h_j D_j \rho_j$ is equal to the same constant $A_{hD\rho}$ for every j . From the interchange argument described in the beginning of this section, it follows that the objective now is to maximize the *number* of different items on the slowest machine, followed by the number of items on the second slowest machine, and so on. (This is in contrast to the maximization of the utilization in the second special case.) Again, it can be shown easily (through a reduction from 3-PARTITION) that the problem is strongly NP-Hard when $h_j D_j \rho_j = A_{hD\rho}$ for $j = 1, \ldots, n$, and ρ_j arbitrary.

Maximizing the number of jobs on the slowest machine, followed by the number of jobs on the second slowest machine, etc., can also be done via a two-phase heuristic.

Before performing Phase 1 of the two-phase heuristic, the items have to be ranked in *increasing* order of ρ_j (i.e., according to Shortest Processing Time first (SPT)). In Phase 1, load the machines beginning with the slowest machine, followed by the second slowest, etc. Assign the items to the machines according to the SPT rule. Each machine then has a number of jobs assigned to it. (The slower machines may or may not end up with more jobs than the faster machines. They may have more because they are assigned shorter jobs; on the other hand, they may have less because they have less capacity).

In Phase 2, the heuristic tries (again beginning with the slowest machine) to maximize the utilization of the machines with lower rank without reducing the number of items processed. This is done as follows: First, the remaining idle capacity on the slowest machine (i.e, machine 1) is computed. Next, perform a pairwise interchange between a short job on machine one and longer job on machine 2 in order to better utilize the remaining capacity on machine 1. After maximizing the utilization of machine 1 (using the same number of items as under the original assignment), the heuristic proceeds with machine 2. It tries to maximize the utilization of machine 2 by swapping items between machines 2 and 3, and so on.

The heuristics for the second and third special cases are specifically designed for those particular assumptions. A heuristic for an arbitrary instance of this problem should be more general. Because of the similarity of this problem to the (multiple) knapsack problem, it makes sense to adapt a heuristic for the knapsack problem to the version of the problem in this section with arbitrary h_j, D_j and K_j, assuming the machines are fully utilized. A well-known heuristic for the knapsack problem is based on a ranking of the items according to the benefit they provide (or according

to the cost incurred) divided by the space they occupy. In our model the cost is proportional to $h_j D_j \rho_j$ and the space occupied is ρ_j. So the ratio is

$$\frac{h_j D_j \rho_j}{\rho_j} = h_j D_j.$$

This suggests the following general heuristic for the problem in this section: Put the jobs in decreasing order of $h_j D_j$. Start filling the slowest machine with the item that has the highest $h_j D_j$. Without loss of generality, we can assume that

$$h_1 D_1 \geq h_2 D_2 \geq \cdots \geq h_n D_n.$$

Starting at the top of the list, every time an item is taken and put on the slowest machine that still has enough remaining capacity. (This heuristic is indeed in agreement with the three rules described for the three special cases above).

However, such a simple one-phase heuristic for the knapsack problem may not always yield good results. While loading a machine, going down the list of items, it may not be possible to use the remaining capacity of that machine fully. It may be the case that one item with a relatively high ρ_j does not fit in the remaining capacity of the machine currently being loaded and therefore has to be assigned to the next machine. However, the next item on the list with a lower ρ_j may still fit on the current machine. In order to see how badly a one-phase heuristic can perform, consider the following example with 2 machines and 2 jobs. Machine 1 has speed 1 and machine 2 has speed ∞. The jobs are ranked in decreasing order of $h_j D_j$: $h_1 D_1 = 2 + \epsilon$, $h_2 D_2 = 2$; $\rho_1 = \rho + \epsilon$, $\rho_2 = 1 - \rho$. Assume all $K_j = 0$. If we use the heuristic, then item 1 goes on the slow machine and item 2 goes on the fast machine. The total cost per unit time under this assignment (ignoring values of order ϵ) is

$$Z_1(1) + Z_2(2) = x(1 - \rho) + x = x(2 - \rho),$$

where x is the cycle time. The total cost under the assignment that puts item 2 on the slow machine and item 1 on the fast machine is

$$Z_1(2) + Z_2(1) = x(1 - (1 - \rho)) + x = x(1 + \rho).$$

So the heuristic may yield a solution that is two times more expensive than the optimal solution

(when one of the items has a ρ very close to zero and the other item has a ρ very close to 1).

It is, of course, possible to use more elaborate multi-phase heuristics for this problem. There are also well-known fully polynomial time approximation schemes (PTAS) for the knapsack problem (see, for example, Schuurman and Woeginger (2001)). One could solve the problem using the following framework: apply a knapsack algorithm (either a heuristic or a PTAS) to the slowest machine. After having assigned items to the slowest machine, apply the same knapsack type algorithm to the remaining items on the second slowest machine, etc. Note, however, that even if every single machine knapsack problem is then solved to optimality, the solution obtained for the overall problem may not be optimal.

4 Rotation Schedules with Different Cycle Lengths (Model II)

The problem in this section is basically different from the problem in the previous section. Now the cycle length of each machine can be adapted to the items assigned to that machine independent of what happens on the other machines.

One heuristic that would immediately come to mind is the following: Compute for each item the optimal cycle length assuming it would be the only item to be produced on a machine that operates at speed $v = 1$. Rank the items according to these optimal cycle lengths, i.e., according to

$$\frac{2K_j}{h_j D_j(1-\rho_j)} = \frac{K_j}{F_j}.$$

Now, put the items in a list either in increasing order of their cycle lengths or in decreasing order. Assign the items one by one to the machines, starting with the slowest machine first. Even if the items with longer cycle lengths would be grouped together on one machine while the items with shorter cycle lengths would go to another machine, it is not clear whether the items with longer cycle lengths should be assigned to a slower machine and the items with shorter cycle lengths should be assigned to a faster machine or vice versa.

Before presenting a general heuristic for this problem, we consider first three special cases. In our first special case, we assume that we have two machines and both machines are fully utilized.

Furthermore, $\rho_j = \rho$ and $h_j D_j = A_{hD}$ for all j, and all K_j's are different. Consider two machines with speeds v_a and v_b, where $v_a \leq v_b$. The number of items on machines a and b, n_a and n_b, are, of course, independent of the schedule. Clearly, $n = n_a + n_b$. Let

$$F_a = \frac{A_{hD}}{2}\Big(1 - \frac{\rho}{v_a}\Big)$$

and

$$F_b = \frac{A_{hD}}{2}\Big(1 - \frac{\rho}{v_b}\Big).$$

If $v_a \leq v_b$, then $F_a \leq F_b$ and $n_a \leq n_b$. The average total cost on the two machines is

$$Z = Z_a + Z_b = 2\ \sqrt{n_a F_a \sum_{j \in N_a} K_j}\ +\ 2\ \sqrt{n_b F_b \sum_{j \in N_b} K_j}.$$

Without loss of generality we may assume that

$$K_1 \leq K_2 \leq \cdots \leq K_n.$$

Let

$$K_{tot} = K_1 + K_2 + \cdots + K_n.$$

It is clear that the problem of minimizing the total average cost on the two machines reduces to determining the total amount of K_j, i.e., the sum of K_j for a group of items, that has to be assigned to machine a. Let y denote this total amount. The remaining amount, $K_{tot} - y$ is then assigned to machine b. So the function to be minimized is

$$Z = 2\sqrt{n_a F_a y}\ +\ 2\sqrt{n_b F_b (K_{tot} - y)}.$$

The feasible range of y is $K' \leq y \leq K''$, where

$$K' = \sum_{j=1}^{n_a} K_j$$

and

$$K'' = \sum_{j=n-n_a+1}^{n} K_j = K_{tot} - \sum_{j=1}^{n_b} K_j.$$

Because of the fact that the total cost function Z is unimodal (see Figure 1), the function is minimized either for $y = K'$ or for $y = K''$. Either solution may occur, depending upon the data set. This implies that the optimal assignment puts either the n_a items with the lowest setup costs or the n_a items with the highest setup costs on the slower machine. This result forms a basis for the following more general result for m machines and n items. Assume again that $\rho_j = \rho$ and $h_j D_j = A_{hD}$ for all j and only the setup costs $K_1 \leq K_2 \leq \cdots \leq K_n$ are different.

Theorem 2. *If under an optimal schedule for m machines both items j and k are assigned to machine i and if $K_j < K_l < K_k$, then item l must also be assigned to machine i.*

Proof: The proof is by contradiction. Suppose item l is assigned to another machine, say machine h. Consider the subproblem that consists only of machines h and i and the items that have been assigned to these two machines. If $K_j < K_l < K_k$, then this schedule does not have the set of items with the smallest K_j assigned to one machine and the set of items with the largest K_j assigned to the other machine. This contradiction completes the proof. □

One may want to conjecture that an optimal assignment for m machines could be achieved by loading the machines starting with the slowest one and assigning the items to the machines either in decreasing order of K_j or in increasing order of K_j. Unfortunately, such a "monotone" result does not hold for three or more machines and a counterexample can be found easily.

The second case also assumes that all machines are fully utilized and that $\rho_j = \rho$. However, now $K_j = K$ and all $h_j D_j$ products are different. Consider two machines with speeds v_a and v_b, where $v_a \leq v_b$.

The average total cost on the two machines is

$$Z = Z_a + Z_b = 2\ \sqrt{n_a K (1 - \frac{\rho}{v_a}) \sum_{j \in N_a} h_j D_j}\ +\ 2\ \sqrt{n_b K (1 - \frac{\rho}{v_b}) \sum_{j \in N_b} h_j D_j}.$$

Without loss of generality we may assume that

$$h_1 D_1 \leq h_2 D_2 \leq \cdots \leq h_n D_n.$$

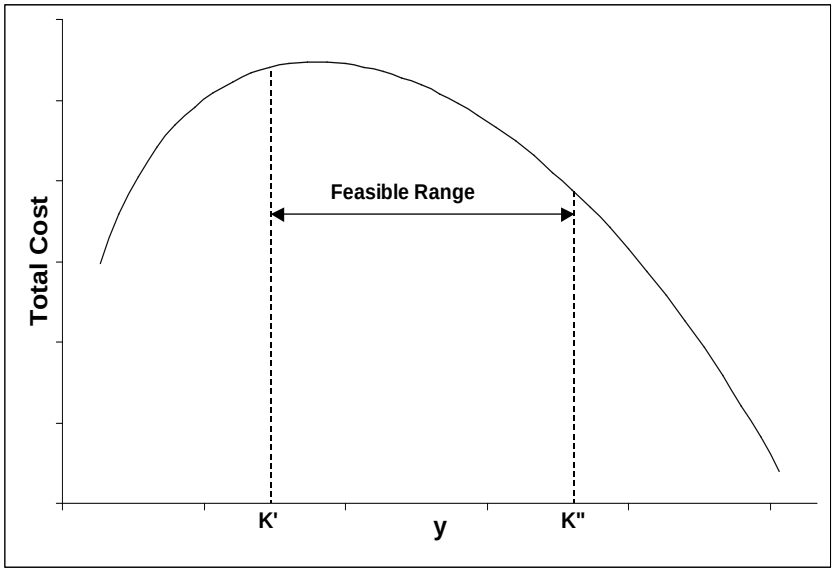

Figure 1: Total Cost Function

Let $hD_{tot} = h_1D_1 + h_2D_2 + \cdots + h_nD_n$. It is clear that the problem of minimizing the total average cost on the two machines reduces to determining the total amount of h_jD_j that has to be assigned to machine a. Let z denote this total amount. The remaining amount, $hD_{tot} - z$ is assigned to machine b. The following theorem for m machines and n items can be shown in a way that is similar to the proof of Theorem 2. Assume that $\rho_j = \rho$ and $K_j = K$ for all j and only the products $h_1D_1 \leq h_2D_2 \leq \cdots \leq h_nD_n$ are different.

Theorem 3. *If under an optimal schedule for m machines both items j and k are assigned to machine i and if $h_jD_j < h_lD_l < h_kD_k$, then item l must be assigned to machine i as well.*

Note that Theorems 2 and 3 do not give any indication whether items with long cycle lengths should be assigned to slower machines and items with short cycle lengths to faster machines or vice versa. The assignment very much depends on the particular data set.

Theorems 2 and 3 apply to the case when all machines are operating at capacity. However, if the machine utilization is below capacity, then Theorems 2 and 3 do not provide much insight. It then becomes an issue of aggregating items with similar cycle lengths on the same machine. It may be better, when the cycle lengths of two items are more or less similar, to put the two items together on a faster machine (even when there is still capacity left on the slower machine for one of the two items).

Example: Consider two machines with speeds $v_1 = 1$ and $v_2 = 2$. Consider four items with $\rho_j = 0.333$ for $j = 1, \ldots, 4$. This setup implies that it is possible to produce exactly three of the four items on the slow machine. Let $h_j D_j = 2$ for $j = 1, \ldots, 4$. Let $K_1 = K_2 = 9$ and $K_3 = K_4 = 1$. This implies that the cycle lengths of items 1 and 2 are three times longer than the cycle lengths of items 3 and 4. Should one put items 1, 2, and 3 on the slowest machine and item 4 on machine 2, or should one put items 1 and 2 on the slowest machine and items 3 and 4 on machine 2 (aggregating the two products with the long cycle lengths on machine 1 and the two items with the short cycle lengths on machine 2)? Straightforward computations yield

$$Z_1(1, 2, 3) + Z_2(4) = 12.33 + 1.83 = 14.16,$$

and

$$Z_1(1, 2) + Z_2(3, 4) = 9.80 + 3.65 = 13.45.$$

So in this case, it is not optimal to fully utilize the capacity of machine 1. It is better to keep the two items with a long cycle length on the slow machine while keeping the two items with a short cycle length on machine 2.

The previous cases (Theorems 2 and 3) do not give any indication of how differences in ρ_j affect the assignment of items to machines. Actually, as stated earlier, a case with different ρ_j is much harder to handle, especially because of the issues concerning fitting and packing. Nonetheless, the following somewhat stylized case is amenable to analysis and provides significant insight.

In this third case, we consider two sets of items, namely sets R and S. Set R consists of l items all with the same ρ_j, i.e.,

$$\rho_j = \rho/l = \rho^R, \qquad j \in R,$$

and set S consists of k items, each one with

$$\rho_j = \rho/k = \rho^S, \qquad j \in S.$$

All items in set R have the same value of K_j and $h_j D_j$, i.e., $K_j = K^R$ and $h_j D_j = A^R_{hD}$ for $j \in R$. All items in set S also have the same value K_j and $h_j D_j$, i.e., $K_j = K^S$ and $h_j D_j = A^S_{hD}$ for all $j \in S$. Assume that $k > l$ and $\rho^R > \rho^S$. The machines have speeds v_a and v_b, where $v_a \leq v_b$. We

assume that all items from set R (set S) would fit on either machine.

Theorem 4. *If*

$$K^R A_{hD}^R \geq K^S A_{hD}^S$$

and

$$\rho^R \geq \rho^S,$$

then the items in Set R (with the higher KhD and ρ values) have to go on the slower machine, i.e., machine a.

Proof: Assume first that the items of set R are put on machine b and the items of set S are put on machine a. The total cost per unit time of the two machines is

$$\sqrt{2l^2 K^R A_{hD}^R (1 - \frac{\rho}{l v_b})} + \sqrt{2k^2 K^S A_{hD}^S (1 - \frac{\rho}{k v_a})}.$$

After interchanging the two sets, the total cost becomes

$$\sqrt{2l^2 K^R A_{hD}^R (1 - \frac{\rho}{l v_a})} + \sqrt{2k^2 K^S A_{hD}^S (1 - \frac{\rho}{k v_b})}.$$

Let $\alpha = \rho/v_a$ and $\beta = \rho/v_b$. So $1 > \alpha > \beta$. To prove the theorem, we have to show that it is better to put the set with the larger jobs (larger ρ_j) on the slower machine and the set with the smaller jobs on the faster machine. That is, if $k > l$, then set R (S) has to go on machine a (b). So it has to be shown that

$$\left(K^S A_{hD}^S\right)^{\frac{1}{2}} (k^2 - k\beta)^{\frac{1}{2}} + \left(K^R A_{hD}^R\right)^{\frac{1}{2}} (l^2 - l\alpha)^{\frac{1}{2}} < \left(K^S A_{hD}^S\right)^{\frac{1}{2}} (k^2 - k\alpha)^{\frac{1}{2}} + \left(K^R A_{hD}^R\right)^{\frac{1}{2}} (l^2 - l\beta)^{\frac{1}{2}}.$$

Or

$$\left(K^S A_{hD}^S\right)^{\frac{1}{2}} \left((k^2 - k\beta)^{\frac{1}{2}} - (k^2 - k\alpha)^{\frac{1}{2}}\right) < \left(K^R A_{hD}^R\right)^{\frac{1}{2}} \left((l^2 - l\beta)^{\frac{1}{2}} - (l^2 - l\alpha)^{\frac{1}{2}}\right).$$

Since $K^S A^S_{hD} \leq K^R A^R_{hD}$, it suffices to show that

$$(k^2 - k\beta)^{\frac{1}{2}} - (k^2 - k\alpha)^{\frac{1}{2}} < (l^2 - l\beta)^{\frac{1}{2}} - (l^2 - l\alpha)^{\frac{1}{2}}.$$

It suffices to show that the derivative of the LHS of this expression with respect to k is negative. The derivative of the LHS of the inequality above is

$$\frac{1}{2}(k^2 - k\beta)^{-\frac{1}{2}}(2k - \beta) - \frac{1}{2}(k^2 - k\alpha)^{-\frac{1}{2}}(2k - \alpha) < 0.$$

Or

$$\frac{2k - \beta}{2k - \alpha} < \Big(\frac{k - \beta}{k - \alpha}\Big)^{\frac{1}{2}}.$$

Straightforward algebra yields

$$\beta^2(k - \alpha) < \alpha^2(k - \beta),$$

which indeed holds. □

In order to have a feel for the impact of $K^R A^R_{hD}$ and $K^S A^S_{hD}$ on the total cost under the conditions stated in Theorem 4, consider the following case: Assume $K^S A^S_{hD}$ and $K^R A^R_{hD}$ are allowed to be arbitrary while $k = l$, i.e., $\rho^R = \rho^S$. Let $v_a = 1$ and v_b be very large. If $\rho/(kv_a)$ is close to 1, then the total cost of the suboptimal assignment (i.e., set S with the lower $K^S A^S_{hD}$ assigned to the slower machine) divided by that of the optimal assignment (i.e., set R with the higher $K^R A^R_{hD}$ assigned to the slower machine) becomes approximately

$$\sqrt{\frac{K^R A^R_{hD}}{K^S A^S_{hD}}},$$

which may be arbitrary large.

To see that ρ^R and ρ^S has much less of an impact on the total cost, consider the following extreme case. Let $K^R A^R_{hD} = K^S A^S_{hD}$; $v_a = 1$ and $v_b = \infty$; and $k = \infty$ and $l = 1$. The total cost under the suboptimal assignment of sets R and S is now of the same order of magnitude as the total cost cost under the optimal assignment of sets R and S.

Theorem 4 does not provide any insight with regard to how items should be grouped with regard to their cycle lengths. All items within Set R have the same cycle length and all items

within Set S have the same cycle length; the cycle length of a Set R item can be either smaller or larger than the cycle length of a Set S item. So in some cases, items with a short cycle length would be assigned to the slowest machine, while in other cases, the items with a long cycle length would be assigned to the slowest machine.

However, Theorem 4 does give some insights into some other effects. In case $K_j = K$ and $h_j D_j = A_{hD}$ for all j, the theorem suggests the following rule: Load the machines starting with the slowest one first, and assign the items according to the Longest Processing Time first (LPT) rule. Clearly, this rule may have to deal also with packing and fitting problems, but in general it seems to be appropriate.

Theorem 4 also provides some insights into some other secondary effects when all the items in sets R and S have the same ρ and the same cycle length.

Example: Consider two machines that operate at speeds v_a and v_b and four items with the same ρ. Each machine has to be assigned two items. The following data are associated with the four items:

$$K_1 = K_2 = \gamma K, \qquad h_1 D_1 = h_2 D_2 = \gamma A_{hD},$$

$$K_3 = K_4 = \delta K, \qquad h_3 D_3 = h_4 D_4 = \delta A_{hD},$$

where $\gamma < \delta$. So

$$\frac{K_1}{h_1 D_1} = \frac{K_2}{h_2 D_2} = \frac{\gamma K}{\gamma A_{hD}} = \frac{K}{A_{hD}}, \qquad \frac{K_3}{h_3 D_3} = \frac{K_4}{h_4 D_4} = \frac{\delta K}{\delta A_{hD}} = \frac{K}{A_{hD}}.$$

The individual cycle lengths of all four items are the same. Since

$$x_i = \sqrt{\Big(\sum_{j=1}^{n} F_j\Big)^{-1} \sum_{j=1}^{n} K_j}$$

and

$$Z_i = 2 \sqrt{\sum_{j=1}^{n} F_j} \sqrt{\sum_{j=1}^{n} K_j},$$

it turns out that if $v_a = v_b$, *any* combination of two items on one machine and the remaining two on the other machine minimizes the total cost. What happens if $v_a < v_b$ ($\alpha > \beta$)? Applying

Theorem 4 to this example indicates that if Set S consists of items 1 and 2 and if Set R consists of items 3 and 4, then putting items 3 and 4 (with the higher value of $K_j h_j D_j$) together on the slower machine results in the assignment with the lowest cost.

The results obtained in this section for the special cases (and also the computational results reported in Haksöz and Pinedo (2001)) suggest the following general heuristic for arbitrary instances: Put the items in a monotone order of their cycle times, i.e., either in increasing order or in decreasing order of

$$\sqrt{\frac{2K_j}{h_j D_j (1-\rho_j)}}$$

Load the items in that order on the machines starting with the slowest one. If there is a tie (or close to a tie) and the remaining capacity of the machine being loaded is still high, choose the item with the higher value of $K_j h_j D_j$; if the remaining capacity of the machine that is being loaded is low, then choose the items based on goodness of fit.

It turns out that our heuristics for Model II (allowing each machine to have its own cycle length) are basically different from the heuristics designed by Bollapragada and Rao (1999) for the continuous version of this problem. In our heuristics we try to keep items with similar cycle lengths on the same machine. Bollapragada and Rao, who do allow items of any given type to be produced on more than one machine, propose a heuristic based on an entirely different philosophy. (Their heuristic was actually designed for a more elaborate model; however, one can adapt their heuristic to a model with similar assumptions as ours.) They rank all the items in decreasing order of the production cost on any machine per unit of the demand rate of that item, i.e., they order the items in decreasing order of

$$\frac{1}{D_j}\sqrt{K_j h_j D_j (1-\rho ij)} = \sqrt{\frac{K_j h_j}{D_j}(1-\rho ij)}.$$

They find the best cost rate of any item on any machine and produce as much as possible of that item on that machine, i.e., either till the machine capacity is exhausted or till enough of that item has been produced. They proceed with finding among the remaining items that still need to be produced, the lowest cost rate on any machine, and so on. Bollapragada and Rao's heuristic, starting out with a completely different ranking of the items, clearly would work very well for

their problem (especially when setup costs are low). However, it could be the case that items with similar production cost rates have very different cycle lengths. So it could be the case that items with completely different cycle lengths are combined on the same machine.

5 Arbitrary Schedules on Machines with Different Speeds (Model III)

In this section we consider again m different machines with speeds

$$1 = v_1 \leq v_2 \leq \cdots \leq v_m$$

and the schedules do not have to be rotation schedules. This problem can be approached via a two-phase procedure. Phase 1 consists of an assignment problem that requires a heuristic for assigning the different items to the different machines. This phase is somewhat similar to the multiple knapsack problem. Phase 2 consists of a problem that has been studied extensively in the literature. Given a machine and a set of items to be produced, find the optimal schedule that minimizes the total cost on that machine, see Dobson (1987). Clearly, solving the problem in two separate phases may result in a suboptimal solution.

An assignment to a machine may result in a set of items that, in a sense, may be somewhat incompatible. For example, suppose an item with a high ρ is put together on a machine with many small ρ items. The high ρ item may require a long cycle length (because of a high setup cost) and the small ρ items may require a very short cycle length (because of high inventory carrying costs). This makes the second phase of the solution procedure very hard to solve. The high ρ item may require long production runs during which it is not possible to have short runs of low ρ items.

In spite of this difficulty, we are still interested in the first phase of this two-phase approach, mainly to obtain some insights in the factors that influence the assignment of a particular item to a given machine. We analyze Phase 1 of the two-phase procedure assuming that, if item j is assigned to machine i, it will be produced on that machine according to a cycle that is optimal with regard to item j. An optimal solution for this assignment problem provides a lower bound for the optimal schedules for Model III as well as a lower bound for the optimal schedules for Models I and II.

From the previous sections, it is clear that for any item, the slowest machine is the least expensive. The minimum total cost per unit time if item j is assigned to machine i is

$$Z_{ij} = \sqrt{2K_j h_j D_j (1 - D_j/(v_i P_j))} = 2\sqrt{K_j F_{ij}}.$$

Our first special case of the assignment problem is somewhat comparable to the special cases considered in Theorems 1, 2, and 3. Consider the case where $\rho_j = \rho$ for all j. As in the previous sections, this problem is relatively easy.

Theorem 5. *An optimal assignment is obtained by loading the slowest machine first and assigning the items in decreasing order of $K_j h_j D_j$.*

Proof: Under the assumptions made, the cycle lengths of all items are independent of one another, which makes this case particularly easy. A simple pairwise interchange shows that the item with the highest $K_j h_j D_j$ should go on the slowest machine. □

So in this case, the K_j, the h_j, and the D_j do not play any role on their own; only their product matters. Note that the assumption with regard to the values of K_j, h_j and D_j in Theorem 5 are less strict than the assumptions with regard to the K_j, h_j, D_j values in Theorems 2 and 3 for Model II. The result of Theorem 5 applies to any combination of values for K_j, h_j and D_j and is basically the solution to a classical assignment problem. This was not the case in Theorems 2 and 3. Also, in contrast to the result in Theorem 2, the result here applies even when the machines are operating below capacity. It is clear that under the assumptions of Theorem 5 the utilization of the slowest machine always has to be maximized.

Our second special case of the assignment problem is somewhat similar to the third special case in the previous section. We again consider two sets of items, namely Set R and Set S. Set R consists of l items all having the same ρ_j, namely

$$\rho_j = \rho^R = \rho/l, \qquad j \in R,$$

and set S consists of k items, each one with

$$\rho_j = \rho^S = \rho/k, \qquad j \in S.$$

All items in Set R have the same value of $K_j h_j D_j = A^R_{KhD}$ for $j \in R$. All items in Set S have the same value $K_j h_j D_j = A^S_{KhD}$ for all $j \in S$. Assume that $k > l$. So $\rho^R > \rho^S$. We have two machines a and b with speeds v_a and v_b, where $v_a \leq v_b$. A result similar to Theorem 4 can now be shown.

Theorem 6. *If*

$$A^R_{KhD} \geq A^S_{KhD}$$

and

$$\rho^R \geq \rho^S,$$

then the items in Set R with a higher $K_j h_j D_j$ and a larger ρ_j have to be processed on the slower machine, i.e., machine a.

Proof: Assume first that the items of Set R are put on machine b and the items of Set S are put on machine a. The total cost per unit time of the two machines is

$$l\sqrt{\big(2(1-\frac{\rho}{lv_b})\big)} \;+\; k\sqrt{\big(2(1-\frac{\rho}{kv_a})\big)}.$$

After the interchange of the two sets the total cost becomes

$$l\sqrt{\big(2(1-\frac{\rho}{lv_a})\big)} \;+\; k\sqrt{\big(2(1-\frac{\rho}{kv_b})\big)}.$$

It turns out that it is better to put the set with the larger jobs (larger ρ_j) on the slower machine and the smaller jobs on the faster machine. That is, if $k > l$, then set R (S) has to be assigned to machine a (b). In order to prove this, it has to be shown that

$$(k^2 - k\alpha)^{\frac{1}{2}} + (l^2 - l\beta)^{\frac{1}{2}} < (k^2 - k\beta)^{\frac{1}{2}} + (l^2 - l\alpha)^{\frac{1}{2}},$$

where $\alpha > \beta$. This is exactly the same inequality that had to be shown in Theorem 4 for Model II.

Just like with Model II, the optimal schedule here is to assign the longest job to the slowest machine. □

The assumptions in Theorem 6 are more general in two respects than the assumptions underlying Theorem 4. First, in Theorem 4, the K_j as well as the $h_j D_j$ of all the items belonging to the same set have to be identical; in Theorem 6, the product $K_j h_j D_j$ of all the items belonging to

the same set have to be identical. The assumptions underlying Theorem 6 are also more general in another aspect: in Theorem 4, it was assumed that Sets R and S were the only items on the two machines. The interchange argument for Model II could only work under this assumption; if there had been additional items scheduled for production on the two machines, then the total cost per unit time would have depended on the interactions between these additional items and the items in Sets R and S. This assumption is not necessary in Theorem 6. Additional items are allowed on the two machines, since the total cost per unit time does not depend in this case on the interplay between these additional items and the items in Sets R and S.

From Theorem 6, it also follows that the LPT rule is an appropriate rule when $K_j h_j D_j = A_{KhD}$ for all j. However, similar to Model II, the $K_j h_j D_j$ appears to be also a more important factor than the ρ_j.

The following heuristic can be used for the general case. Load the machines starting with the slowest; when loading machine i, assign jobs in decreasing order of their $K_j h_j D_j$ values among the remaining items.

If the machines are operating below capacity, then the optimal solution attempts to fill the slowest machine as much as possible. This is in contrast to Model II, in which the capacity of the slowest machine may in certain instances not be fully utilized.

The conclusions drawn in this section are less firm than those drawn in the previous sections, mainly because they do depend on the assumption that items with different cycle lengths can be combined with one another on one machine.

6 Computational Results

All the experiments reported in this section involve two machines with $v_1 = 1$ and $v_2 = 3$. The same data sets are used for all three models. This is done in order to be able to compare the effects of the different cycle length assumptions on the values of the objective functions.

For each model we compare a number of different scheduling rules with one another. We randomly select 20 different samples of job sets and on each sample we apply each one of the various rules we are interested in. The samples are selected randomly as follows: The K_j and the h_j are both generated from a Uniform $[0, 5]$ distribution. The D_j and P_j are generated randomly from an exponential distribution with mean 5. The number of jobs are determined by generating

jobs till at least 95% of the system capacity is utilized. This implies that the number of jobs is usually approximately 40.

In our experiments for model I we compare two rules with one another, namely the rule that loads the machines in decreasing order of $h_j D_j$ (starting with the slowest machine) and the rule that loads the machine in increasing order of $h_j D_j$ (starting with the slowest machine first). From Table 1 it follows that in these instances the first rule performs approximately consistently 0.3% better than the second rule. All tables are located in the appendix.

(Insert Table 1 about here.)

In our experiments for Model II we compared three rules with one another, namely the rule that loads the machines in decreasing order of $K_j/h_j D_j$ (starting with the slowest machine first), the rule that loads the machines in increasing order of $K_j/h_j D_j$ and the rule that loads the machines in decreasing order of $K_j h_j D_j$. It is clear why the first two rules are of interest: they both combine items that favor long cycle lengths on one machine and items that favor short cycle lengths on the other machine. In our experiments the rule that assigns the items with the shorter cycle lengths to the slower machine performs slightly better than the rule that assigns the items with the longer cycle lengths to the slower machines. The third rule is only of interest because we know that this rule works very well for Model III (see Theorem 6). From Table 2 it is clear that the performance of the first two rules is more or less similar; one rule beats the other by 0.3%, which is statistically not significant. However, the third rule does worse. Yet, the difference is still lower than 5%. The value of the objective function under the third rule is 4.75% higher than under the first two.

(Insert Table 2 about here.)

In our experiments for Model III we compared four rules with one another. The first rule we tried is the rule that is optimal when all ρ_j are equal to one another, i.e., loading the machines in decreasing order of $K_j h_j D_j$. We compare this rule with the rule that loads the machines in decreasing order of $K_j/h_j D_j$. In Model II, this rule performs much better than the rule that loads the machines in decreasing order of $K_j h_j D_j$. In Model III, the first rule beats the second in most instances, but only by 0.6%.

With the third and fourth rules we try to measure the effect of the ρ_j. The third rule orders the items in decreasing order of $K_j h_j D_j \rho_j$ and and the fourth rule in decreasing order of $K_j h_j D_j/\rho_j$.

It appears from the results in Table 3 that all four rules perform more or less similar. Not one of the four rules performs consistently better.

When the cost performance of the three models are compared with one another, it appears that Model II performs approximately 8% better than Model I and Model III performs approximately 9% better than Model II. These results are in agreement with our expectations.

(Insert Table 3 about here.)

7 Conclusions, Extensions, and Real World Applications

Comparing the results obtained for Models I, II and III, it is clear that each model requires its own priority rules.

A comparison of the results for the three models when all $\rho_j = \rho$ can be summarized as follows: In Model I, K_j does not play any role at all and the items should be loaded on the m machines in decreasing order of $h_j D_j$ starting with the slowest machine first. This rule holds whether the machines are fully loaded or not. It appears that because of the fact that all the machines have to adhere to the same cycle length x, the value of the objective function does not depend very much on the assignment of the different items to the various machines.

In Model II, when all $\rho_j = \rho$, K_j plays a role that is very similar to the role of the $h_j D_j$. A monotone rule such as the one obtained for Model I does not hold here. However, it appears that a good heuristic would be to load the items in either increasing or decreasing order of $K_j/h_j D_j$, starting with the slowest machine first. This is approximately equivalent to putting items that favor longer cycle lengths together on one machine and items that favor shorter cycle lengths together on another machine. If the machines are not fully loaded, it may not be optimal to keep the slowest machine fully loaded. Computational experiments show that the value of the total cost in Model II is more sensitive to the schedule than the value of the total cost in Model I (see Haksöz and Pinedo (2001)). However, the performance of the $K_j/h_j D_j$ rule is somewhat comparable to the performance of the $h_j D_j/K_j$ rule (this rule puts items that favor shorter cycle lengths together on the slower machines and items that favor longer cycle lengths on the faster machines). The very bad rules are, of course, those rules that combine items which favor a long cycle length with items

which favor a short cycle length on the same machine; an empirical study (see Haksöz and Pinedo (2001)) shows that the total cost then can be easily 10 percent higher than the optimal cost.

In Model III, when all $\rho_j = \rho$, it is optimal to load the items in decreasing order of $K_j h_j D_j$ starting with the slowest machine first (assuming that there exist feasible schedules under which each item can be produced according to its own optimal cycle length). Note that this rule does not perform very well for Model II. In Model III, it is always better to utilize the slowest machine as much as possible.

When the ρ_js are different, all three models become significantly more complicated. However, the ρ_j tends to have less of an impact on the values of the objectives than K_j, h_j and D_j. The ρ_j also affects the solution in a different way: there is now a packing or fitting problem. Theorems 4 and 6 (and the discussion following Theorem 4) indicate that, in an optimal schedule, an item with a high ρ_j is more likely to be assigned to a slow machine.

When all the ρ_j's are different, the problems are typically variations of the classical knapsack problem or of the multiple knapsack problem. The two machine case with $v_1 = 1$ and $v_2 = \infty$ becomes a special case of the knapsack problem and the m machine case with $v_1 = v_2 = \cdots = v_{m-1} = 1$ and $v_m = \infty$ becomes a special case of the multiple knapsack problem. We expect that all the heuristics discussed should work reasonably well when all the ρ_j's are small (i.e., $\ll 1$).

Several other observations apply to all three models discussed. It is clear from the results that there is most of the time (but not always) a preference to utilize the slower machines. The two parameters h_j and D_j of item j never play a separate role in any one of the heuristics or algorithms for any one of the models discussed. They only play a role as part of the product $h_j D_j$. This is in a clear contrast to the greedy priority rule based on production costs per unit of demand that was suggested by Bollapragada and Rao (1999) (their rule would not be appropriate for our Model II, since it could yield solutions that combine items with completely different cycle lengths on the same machine).

The results presented in this paper may be very useful for scheduling in practice, even though there are still some differences between the modelling assumptions and certain customs in practice. For example, we have assumed in this paper that a machine has only a single parameter, namely a speed v_i. In practice, a machine may have various parameters that are important. For example, a paper machine has its own speed, as well as its own "width". The width is also a measure that affects the paper production per unit time. So, it may be more appropriate to view the speed of machine i as the amount of paper it can produce per unit time.

In practice, the cycle length of a machine may not be just an arbitrary number. Often, it is a round number such as 10 days, half a month, or one month. That means that in the real world the problem of finding the optimal cycle length is not really a continuous problem but rather a discrete problem. Nevertheless, we believe that the mathematical insights obtained for the continuous case are also useful for the discrete case.

In practice, the cycle lengths of the various items on a machine are often different. However, one cycle length has to be a multiple of a base cycle length. That means, the cycle length of one item may be x, while the cycle length of another item is $2x$ or $4x$. Nonetheless, even the insights obtained with Model I may prove useful in practice. Often different machines (with different capacities) operate with the same cycle length.

Many related problems appear to be of interest. We mention here three: First, what happens if setup costs have the structure $K_{ij} = \alpha_i K_j$, where $\alpha_1 \leq \alpha_2 \leq ... \leq \alpha_m$? (In practice, a setup cost is often a simple function of a machine parameter and an item parameter.) Second, consider the problem with preemptions allowed and items may be produced on two machines (thereby incurring two setup costs). Packing and fitting become then less important. In what other respects do the models change? Third, can we prove some limiting result stating that with a large number of items that all take a very small amount of the machine capacity, the heuristics will yield solutions that are very close to optimal (within ϵ)?

Acknowledgements

We greatly acknowledge the beneficial discussions with Sridhar Seshadri.

References

R. Bollapragada and U. Rao (1999) "Single-Stage Resource Allocation and Economic Lot Scheduling on Multiple, Nonidentical Production Lines," *Management Science,* Vol. 45, pp. 889-904.

W. Brüggemann (1995) *Ausgewählte Probleme der Produktionsplanung,* Physica-Verlag (Springer

Verlag), Heidelberg.

J.J. Carreno (1990) "Economic Lot Scheduling for Multiple Products on Parallel Identical Machines," *Management Science,* Vol. 36, pp. 348-358.

Z.-L. Chen and M. Powell (1999) "Solving Parallel Machine Scheduling Problems by Column Generation," *INFORMS Journal of Computing,* Vol. 11, pp. 78-94.

G. Dobson (1987) "The Economic Lot-Scheduling Problem: Achieving Feasibility using Time-Varying Lot Sizes," *Operations Research,* Vol. 35, pp. 764-771.

G. Dobson (1992) "The Cyclic Lot Scheduling Problem with Sequence-Dependent Setups," *Operations Research,* Vol. 40, pp. 736-749.

A. Drexl and A. Kimms (1997) "Lot Sizing and Scheduling - Survey and Extensions," *European Journal of Operational Research*, Vol. 99, pp. 221-235.

R.J. Duffin, E.L. Peterson, and C.M. Zener (1967) *Geometric Programming*, J. Wiley and Sons, New York.

S.E. Elmaghraby (1978) "The Economic Lot Scheduling Problem (ELSP): Review and Extensions," *Management Science,* Vol. 24, pp. 587-598.

M.R. Garey and D.S. Johnson (1979) *Computers and Intractability - A Guide to the Theory of NP-Completeness,* W.H. Freeman and Company, San Francisco.

K. Haase (1993) *Lot Sizing and Scheduling for Production Planning,* Lecture Notes in Economics and Mathematical Systems, Vol. 408, Springer Verlag, Berlin.

Ç. Haksöz and M. Pinedo (2001) "Economic Lot Scheduling with Resources in Parallel," Working Paper, Department of Information, Operations, and Management Sciences, New York University, New York, New York.

P.C. Jones and R.R. Inman (1996) "Product Grouping for Batch Processes," *International Journal of Production Research,* Vol. 34, pp. 3095-3105.

S. Kang, K. Malik, and L.J. Thomas (1999) "Lot sizing and scheduling on parallel machines with sequence dependent setup costs," *Management Science,* Vol. 45, pp. 273-289.

A. Kimms (1997) *Multi-Level Lot Sizing and Scheduling,* Physica Verlag, Heidelberg.

S. Martello and P. Toth (1990) *Knapsack Problems: Algorithms and Computer Implementations,* J. Wiley, Chichester, England.

J.A. Muckstadt and R. Roundy (1993) "Analysis of Multi-Stage Production Systems," Chapter 2 in Handbooks in OR and MS, Vol. 4, pp. 59-132, S.C. Graves et al. Eds., Elsevier Science Publishers.

M. Pinedo (2002) *Scheduling – Theory, Algorithms, and Systems*, Prentice-Hall, Upper Saddle River, New Jersey.

M. Pinedo (2005) *Planning and Scheduling in Manufacturing and Services,* Springer, New York, NY.

D. Pisinger (1999) "An Exact Algorithm for Large Multiple Knapsack Problems," *European Journal of Operational Research,* Vol. 114, pp. 528-541.

C.N. Potts and L.N. van Wassenhove (1992) "Integrating Scheduling with Batching and Lot Sizing: A Review of Algorithms and Complexity," *Journal of the Operational Research Society,* Vol. 43, pp. 395-406.

M. Salomon, L.G. Kroon, R. Kuik, and L.N. van Wassenhove (1991) "Some Extensions of the Discrete Lotsizing and Scheduling Problem," *Management Science,* Vol. 37, pp. 801-812.

M. Van den Akker, H. Hoogeveen and S. Van de Velde (1999) "Parallel Machine Scheduling by Column Generation," *Operations Research,* Vol. 47, pp. 862-872.

P. Schuurman and G.J. Woeginger (2001) "Approximation Schemes - A Tutorial," START Project Y43-MAT Combinatorial Approximation Algorithms, TU Graz, Austria.

P.H. Zipkin (2000) *Foundations of Inventory Management,* McGraw-Hill, Boston, MA.

Appendix

Table 1: Computational Results for Model I

Test Sample	h_jD_j	$1/h_jD_j$
1	339.59	346.93
2	297.11	299.38
3	357.63	363.44
4	394.52	399.85
5	336.50	319.33
6	350.92	356.75
7	483.62	434.74
8	332.86	336.86
9	444.93	450.83
10	302.80	305.23
11	362.61	365.99
12	414.53	421.14
13	305.72	309.52
14	378.91	382.45
15	339.20	344.37
16	437.18	441.48
17	347.27	351.77
18	364.65	371.79
19	345.14	349.97
20	366.58	371.46
Average Cost	**365.11**	**366.16**

Table 2: Computational Results for Model II

Test Sample	K_j/h_jD_j	h_jD_j/K_j	$K_jh_jD_j$
1	301.26	298.35	308.20
2	261.53	267.31	290.73
3	333.63	330.96	347.34
4	363.84	362.91	377.20
5	280.02	300.05	317.97
6	330.65	324.31	343.81
7	438.56	433.46	458.49
8	296.25	302.28	317.63
9	392.97	391.06	417.34
10	278.04	279.16	290.97
11	324.07	336.70	352.80
12	368.05	366.47	387.19
13	277.19	280.40	295.43
14	352.36	354.32	367.61
15	310.45	305.76	315.56
16	406.83	410.58	426.33
17	323.05	321.04	339.16
18	321.40	319.54	337.15
19	315.01	313.42	323.56
20	343.17	338.54	353.11
Average Cost	**330.92**	**331.83**	**348.38**

Table 3: Computational Results for Model III

Test Sample	$K_j h_j D_j$	$K_j/h_j D_j$	$K_j h_j D_j \rho_j$	$K_j h_j D_j/\rho_j$
1	279.24	283.35	279.62	279.60
2	236.77	237.47	236.86	237.04
3	309.92	314.61	307.37	310.52
4	337.74	341.00	338.04	337.94
5	282.44	263.78	275.29	283.59
6	311.79	314.87	312.49	312.75
7	403.82	409.49	404.51	403.75
8	278.40	280.40	278.48	278.60
9	360.18	363.70	361.05	359.77
10	259.66	261.11	259.97	260.40
11	290.07	291.23	290.22	290.29
12	340.24	343.17	340.81	340.63
13	253.48	255.35	253.93	253.73
14	327.06	329.01	327.38	327.60
15	282.07	284.58	282.46	282.44
16	376.27	378.91	376.80	376.53
17	304.83	307.35	305.28	305.30
18	301.17	305.52	301.63	301.53
19	286.48	288.70	286.79	286.65
20	319.21	322.89	320.13	319.59
Average Cost	**307.04**	**308.82**	**306.96**	**307.41**