Optimal and heuristic solutions for a scheduling problem arising in a foundry

Sigrid Lise Nonås[a, 1] Kai A. Olsen[b]

[a] Department of Information Science, University of Bergen, P.O. Box 7800, 5020 Bergen, Norway and Department of Finance and Management Science, Norwegian School of Economics and Business Administration, Helleveien 30, 5045 Bergen, Norway
[b] Molde College, P.O.Box 2110, 6402 Molde, Norway and Department of Informatics, University of Bergen, Norway

Abstract

A scheduling problem for an engineer-to-order foundry that manufactures propeller blades for ships is outlined. The main objective for the foundry is to find an efficient production plan that minimizes the total number of days orders are late. This has to be done subject to a set of relatively common constraints related to the scheduling of the present jobs and resources. The constraints on the resources are quantified by a capacity matrix. A mixed integer linear programming formulation for the scheduling problem is proposed together with a set of corresponding heuristic strategies. The heuristic strategies perform very well compared to previously scheduling policies used in the company, and in comparison to optimal solutions where they can be found. Numerical results are presented to demonstrate the performance of these strategies.

Keywords: Production scheduling, Order acceptance

1. Introduction

Some years ago we were contacted by the president of a Norwegian foundry, an engineer-to-order manufacturer of propeller blades for ships. Due to an increase in orders, their manual job scheduling system was breaking down. What they needed was a planning system that could schedule all orders in some "fair" sequence, minimizing the total number of days the jobs were late, and which gave them the opportunity to promise realistic due-dates for new orders.

Modern propellers are designed to fit the hull characteristics of the ship. Based on coordinate tables with approximately 400 reference points, a model of the propeller blade is manufactured in wood by a CAD/CAM process. This wood model is placed in a two-part metal box and a sand-fixture solution is used to make a mold of the upper and lower part of the blade. To simplify, when the sand has hardened the model is removed and the upper and lower part of the box are combined, defining a space that is filled with nickel-aluminum alloy. This process is then repeated for every blade of the job, a number that varies from 1 (a replacement blade) to 30 (blades for several identical propellers or thrusters).

The firm uses five different types of boxes for the molds, in sizes (width, length, height) from $0.6 \times 0.8 \times 0.4$ to $2.2 \times 3.0 \times 1.3$ (meters). In order to save sand and space, the smallest box that can fit the blade is used. Times for hardening of the heated alloy increase as the size of the propeller blades increase. The work area is determined by the area that can be serviced by the crane holding the can of heated alloy. This, together with the available work force and the time needed

[1]Corresponding author. Email: sigrid-lise.nonas@nhh.no. Fax: +47 55 95 96 50

for the alloy to harden, limits the number of boxes that can be handled on a shift. In practice, the capacity may be expressed as a combination table, giving the number of boxes of each type that can be handled with a given combination. For a given day or shift, we are free to choose any combination out of a total set of 25.

Raw material is in the form of a special self-setting sand for making the molds and NiAl-bronze ingots. The company is also able to make its own alloy based on different sources of metal, outdated coins from EU countries being one source. A mass spectrometer is used to verify that the resulting NiAl-alloy is within tight impurity-tolerances. Sand, ingots and other raw materials are procured based on long term forecasts. The production is engineer- or made-to-order, and no inventory is maintained. Finished blades are transported by truck or ship directly to the customer. In practice this implies that planning can be performed independently of outside factors with the exception of due dates.

The problem statement for the system is *to design a planning system that can schedule all jobs, considering capacity constraints in form of a combination table, minimizing the number of days the orders are late. Each job in the system is defined by the model number (giving box size), the number of blades and a due date, and has to be scheduled on successive days until all blades have been produced.*

The system was initially developed using an approach where jobs were sequenced according to due date. Simple heuristics were then used to shuffle the jobs scheduled according to the due-date approach in order to get a new improved plan. A recursive algorithm, swapping a job finished ahead of time with a late job, was found to give the best results, reducing the total number of days the jobs were late by roughly 10% compared to the initial due-date approach. This improved due-date approach was then implemented for use at the company. What was initially far from clear, however, was how close this solution was to an optimum and if there existed better alternatives. In order to get a deeper understanding of the problem and to be able to propose a solution method which performs better than the improved due-date approach, a mathematical model for the problem will be outlined. Then, a four-stage heuristic strategy will be proposed, a heuristic that performs significantly better than the original approach.

The problem faced by this manufacturer should be common also for other engineer-to-order foundries and probably also for some other industries. Similar to the foundry in consideration many of them will have a limited number of boxes that can be used on a given day, due to constraints of space, tools and time for the alloy to harden. While the combination table for a foundry will describe what combination of boxes that can be used on a given day, "boxes" may have another representation in other industries. The basic idea is that resources are described through a combination table that measure the overall capacity. Each combination is a measure for a specific use of the different resources.

The paper will be organized as follows. First, a detailed problem description is given in Section 2, followed by a review of related literature in Section 3. In Section 4, a mathematical formulation is proposed for the problem outlined and the set of heuristic procedures are offered in Section 5. Numerical results that show the performance of the proposed solution procedures are presented in Section 6. A short discussion of alternative performance measures is given in Section 7. In section 8 we discuss the implementation of the solution procedure in our case company. A summary and a conclusion are given in Section 9.

2. Problem description

The main objective for the local foundry is to find a production plan that minimizes the total number of days jobs are late, i.e. minimizes total tardiness, taking the available resources into account. These are defined through a combination table. An example is shown in Table 1 with 10 of the 25 combinations that are currently in use. Other objectives considered by the company are discussed shortly in Section 7.

Each column in Table 1 describes a box type and each row a possible combination. The box types are numbered from small to large. The combinations are set up by the production manager and the foremen, and are based on the available space, the number of boxes available in each category and other resource considerations such as labour and tools.

Job splitting is not allowed. If the processing of a job has started, we have to process this job on successive days until it is completed. Further, at most one job can be processed for each box type on a given day. An exception occurs when a job does not need all the capacity available on its last production day. A new job of a similar box type can then use the excess capacity. These restrictions are mainly due to technical production considerations. In particular, holding and handling more than one wood model for each box type is difficult due to space considerations. Producing two jobs of the same box type in parallel will also increase tardiness. The assumptions are therefore sensible to include in a solution procedure as well.

Finally, we will assume that all jobs are available from day one. The objective of the planning system will be to meet the due dates for the present set of jobs (minimize the total number of days the jobs are late). In practice, we will make a new plan whenever a new order is received.

The problem can be viewed as a deterministic scheduling problem with N jobs and B different box types. The processing time for a job will vary in accordance with the other jobs that are processed at the same time, or to be more specific, in accordance with the box combinations used. The processing time is therefore resource dependent. Furthermore, there exist constraints on which and how many boxes that can be in use simultaneously. These possible combinations of boxes are determined by a combination table.

3. Literature Review

Sequencing and scheduling are characterized by a virtually unlimited number of problem types. We will briefly discuss how the scheduling problem defined in this paper differs from the more general scheduling problems: the pure sequencing problem (see Baker [1]); the job-shop problem (Baker [1] and French [2]) and the multiple resource constrained job shop presented by Gargeya and Deane [3]. A general survey of these common scheduling problems is provided by Lawler et al. [4] in Logistics of production and inventory edited by Graves et al.

The pure sequencing problem is a specialized scheduling problem in which the ordering of the jobs completely determines the schedule. For the problem considered here, the schedule is determined by the ordering of the jobs and the choice of box combination for each day (work shift). Therefore, this problem cannot be categorized as a pure sequencing problem. However, for the case where only one box combination is considered, with a positive capacity for each box type, the problem

decomposes into B independent sequencing problems.

The pure sequencing problem, or equivalently the single machine scheduling problem, deals with the problem of scheduling a set of jobs to a continuously available machine in order to meet some performance measures, such as meeting delivery dates. The complexity of a single machine sequencing problem, when the performance measure is to minimize total tardiness, is proven to be NP-hard in the ordinary sense by Du and Leung [5]. A state of the art survey for the single machine sequencing problem can be found in Koulamas [6]. Here, both heuristic and optimal solution methods are suggested. The most efficient optimal algorithm, considering total tardiness, is the combined dynamic and branch-and-bound algorithm developed by Potts and Van Wassenhove [7].

In the general formulation of the job-shop problem, N jobs are to be processed through B machines. Each job has between one and n operations. The B machines can work in parallel, performing one operation at a time. When the objective is to minimize total tardiness, not allowing job splitting, the problem can be shown to be NP-hard in the strong sense by utilizing the complexity result from the flow-shop problem with two machines, which Garey et al. [8] prove is strongly NP-hard. Different heuristics and priority rules have been proposed to solve this problem (see Koulamas [6] for a good survey). Recently, also random search procedures such as simulated annealing and tabu search have been proposed. Among the latest work is a paper by Armentano and Scrich [9]. They present a tabu search heuristic which initially creates a solution based on dispatching rules and then searches for new solutions in a neighborhood based on the critical paths of the jobs. Related work is performed by Wang and Wu [10,11], they present a revised simulating annealing algorithm for the job shop problem. Random search is not investigated in this paper due to the many restrictions and the large dependency present both in the job sequence and the machine combination sequence. Minor changes in the solution structure in order to get out of a local optimum will then be difficult to perform.

The scheduling problem for the foundry is a simpler variant of the general job-shop problem in the sense that each job has a set of equal operation that can be processed at one specific machine. This is in contrast to the general job shop where each job has a set of different operations which has to be processed at different machines. However, the scheduling problem for the foundry will be more complex than the general job-shop problem in the sense that the availability of auxiliary resources such as tools and labor are restricted and that these resources then have to be shared among the machines.

The common way to approach this multiple resource constrained job shop problem is to use priority rules for selecting jobs to machines and assignment rules for selecting auxiliary resources to jobs present at machines. Gargeya and Deane [3] have given a survey of this research area. Fredendall et al. [12] discuss a related problem to the one we are considering here. They present an automobile assembly plant where each job in the system requires access to both labor and tools. The plant has difficulties in meeting their due dates, and the scheduler has problems in picking the "right" jobs as well as the "right" level of work to release. In their paper Fredendall et al. presented new order review/release, job dispatching, and labor assignment rules.

The foundry approaches its scheduling problem by finding a suitable and limited set of "capacity combinations", i.e. box combinations, that each represents a specific distribution of the auxiliary resources among the machines. Instead of dealing with a combination of different assignment rules, one assignment rule is then considered that determines the "capacity combination" to use at the

set of machines. Here, we will think of the different set of foundry boxes as the machines. Due to the limited space on the floor, different times for "hardening" for the different propeller sizes, and the capacity related to the can of heated alloy, only specific combination of boxes can be used on one day, and for these combinations only a limited set of employees is needed. Solving each of the restrictions independently will for each day result in solving a two-dimensional cutting stock problem for calculating how to use the floor space, in addition we have the dependency on the different times of hardening and the processing times related to the can of heated alloy. Finally, there is the problem of scheduling jobs on successive days according to these restrictions, and having enough space for the different forms needed for these jobs. To find a solution to this complex problem the company has invested time in finding different set of box combinations that define their overall capacity restrictions for a given day.

In general scheduling problems, minimizing make span (total time to complete the N jobs) is also a common performance measure. Now, assume this objective will be considered for the foundry instead of minimizing total tardiness. Further, let us relax the restriction that job splitting is not allowed. It is then easy to see that the resulting scheduling problem can be written as a cutting stock problem that given the possible box combinations minimizes the total number of processing days necessary to meet the demands for propeller blades. The general cutting stock problem addresses the practical issue of how to cut required stock material into specified parts (how to schedule the propeller blades) with minimum use of raw material (with minimum use of work shifts), in order to satisfy the demand for the parts. Stock material is generally cut according to some cutting patterns. Here, the box combination can be viewed as a pattern for organizing the work area or, to be more specific, as a pattern for the combination of parts that can be produced at a work shift. Kantorovich [13] solved this type of problem by formulating it as a LP-problem as early as in 1939. Since then a lot of work has been done on the cutting stock problem (a good review can be found in Haessler and Sweeney [14]). While the cutting stock formulation does not capture our originally formulated problem, it may be used to give a lower bound for the number of days needed to schedule the jobs. It could also give us an idea for which types of combinations to schedule. Minimizing the number of production days may in turn have an effect in reduced total tardiness. However, for a problem where both due dates and job sizes are unlucky "distributed", the cutting–stock combinations may not be of any use.

The model has also some similarities with a multi-product periodic lot sizing problem with capacity restrictions. Instead of penalizing backordered demand we can think of a period as a single day and penalize jobs that do not meet their due date. Note that we still have a difference between penalizing the number of late orders and penalizing the quantity of each late order. In addition, we have to allow for each product that more than one demand batch are due at the same period, further that there are different sets of capacity restrictions. Finally, we have to take into consideration that it could pay off to produce a demand batch for a late period before a demand batch for an earlier period, since it is not the quantity of backordered demand that is penalized but the number of batches of backordered demand for each period. The lot sizing sequence will always be scheduled according to due date. See Van Mieghem and Rudi [15] for literature related to multi-product multi-period models with capacity restriction and Nahmias [16] for more general lot-sizing problems.

4. Mathematical formulation

Here we will propose a mixed integer linear programming (MILP) formulation for the scheduling problem discussed in the previous sections. After defining the notation we will present the objective

function for the problem, followed by the corresponding problem restrictions. The parameters involved in the formulation will be presented with uppercase letters and the variables with lowercase letters.

For the rest of this paper we will use the notation and formulations from Baker [1]. That is, we let

- p_i -*processing time*- be the time required for processing job i
- D_i -*due date*- be the day when the processing of job i is due to be completed
- c_i -*completion day*- be the day when the processing of job i is finished
- l_i -*lateness*- be the number of days the completion day of job i exceeds its due date: $l_i = c_i - D_i$
- t_i -*tardiness*- be the lateness of job i if it fails to meet its due date, zero otherwise: $t_i = \max\{0, l_i\}$

In addition, we let

- N be the number of jobs
- B be the number of box types
- K be the number of combinations
- M be an upper bound for the number of days needed to schedule the N jobs
- O^i be the number of propeller blades requested for job i
- $COMB_b^k$ be the number of b boxes present in box combination k
- $x_{k,j}$ be an indicator variable that flags if box combination k is used at day j
- $\beta_{i,j}$ be an indicator variable that flags if job i is scheduled at day j
- $\delta_{i,j}$ be the number of propeller blades scheduled of job i at day j
- t be the total tardiness in the schedule, $t = \sum_{i=1}^{N} t_i$

Let job i be a request for a given number of propeller blades O^i with a due date D_i. As noted earlier each job can only be processed using a specific box type b. Let there be B different box types. Then, let there be K different vectors $COMB^k$, each defining a unique box combination k. Further, let $COMB_b^k$ denote the number of boxes of box-type b in combination k, $b = 1, ..., B$.

To indicate which of the K box combinations that will be used for a specific day, we let $x_{k,j}$ be a zero/one variable. The variable $x_{k,j}$ is defined as follows,

$$x_{k,j} = \begin{cases} 1 \text{ if box combination } k \text{ is used for day } j \\ 0 \text{ otherwise} \end{cases}$$

A zero/one variable $\beta_{i,j}$ is used to indicate on which day(s) job i is scheduled. Let

$$\beta_{i,j} = \begin{cases} 1 \text{ if job } i \text{ is scheduled on day } j \\ 0 \text{ otherwise} \end{cases}$$

Initially, let the N jobs be sorted into B different job sequences, one for each box type. For simplicity, let them be numbered from 1 to N, where the jobs sequenced for the smallest box types are numbered first. Let J^b be the list of jobs to be produced in box-type b, initially sequenced according to due date. And let J be the set of all jobs.

The main objective is to minimize the total number of days the jobs are late. If t_i is the tardiness of job i, the objective function can be expressed by the following mathematical term

$$\text{minimize} \sum_{i=1}^{N} t_i \tag{1}$$

4.1. Constraints

As previously noted, job splitting is not allowed. Thus, if a job i is processed but not completed at day $j-1$ we have to assure that it is also processed on day j. To keep track of when a job is completed we use an integer variable $\delta_{i,j}$, where

$$\delta_{i,j} = \text{ the number of propeller blades produced for job } i \text{ at day } j.$$

Then, $\sum_{p=1}^{j-1} \delta_{i,p}$ is equal to the number of propeller blades produced for job i up to day $j-1$. If job i is scheduled at day $j-1$ ($\beta_{i,j-1} = 1$) but still not completed ($\sum_{p=1}^{j-1} \delta_{i,p} < O^i$), we have to make sure that it is scheduled also for day j ($\beta_{i,j} = 1$). To avoid job splitting we then let

$$\beta_{i,j} = \begin{cases} 1 \text{ if } \beta_{i,j-1} O^i - \sum_{p=1}^{j-1} \delta_{i,p} > 0 \\ 0 \text{ otherwise} \end{cases}$$

The corresponding restriction can be expressed as

$$\beta_{i,j} * O^i \geq \beta_{i,j-1} O^i - \sum_{p=1}^{j-1} \delta_{i,p}, \quad j = 1, ..., M, \quad i = 1, ..., N. \tag{2}$$

To schedule a job on consecutively days it is also necessary to ensure that $\delta_{i,j}$ is strictly positive if $\beta_{i,j}$ is strictly positive, i.e.

$$\delta_{i,j} \geq \beta_{i,j}, \quad j = 1, ..., M-1, \quad i = 1, ..., N. \tag{3}$$

We also need to ensure that $\beta_{i,j} = 1$ if $\delta_{i,j} > 0$ for a job i at day j. Here we let

$$\beta_{i,j} = \begin{cases} 1 \text{ if } \delta_{i,j} > 0 \\ 0 \text{ otherwise} \end{cases}$$

The final restriction to avoid job splitting can then be expressed as

$$\beta_{i,j} * O^i \geq \delta_{i,j}, \quad j = 1, ..., M, \quad i = 1, ..., N. \tag{4}$$

For the first production day j for a job i we have that $\delta_{i,j} > 0$. Relation (4) ensures that $\beta_{i,j} = 1$ for this day.

To make sure that only one box combination is assigned for each day, we let

$$\sum_{k=1}^{K} x_{k,j} = 1, \quad j = 1, ..., M. \tag{5}$$

The number of blades scheduled at day j for jobs of box-type b has to be less than or equal to the available capacity for box-type b at day j, i.e.

$$\sum_{k=1}^{K} COMB_b^k x_{k,j} \geq \sum_{i \in J^b} \delta_{i,j}, \quad j = 1, ..., M, \quad b = 1, ..., B. \tag{6}$$

The expression $\sum_{k=1}^{K} COMB_b^k x_{k,j}$ is the maximal number of blades that can be processed for box-type b at day j.

Generally, at most one job of each box type can be produced on a given day j. An exception occurs when a job does not use all the available capacity on its last production day. In this case a new job with the same box type can be scheduled for this day. To indicate that at most two jobs of equal box type can be produced on a day j we let

$$\sum_{i \in J^b} \beta_{i,j} \leq 2, \quad j = 1, ..., M, \quad b = 1, ..., B. \tag{7}$$

Here, the sum $\sum_{i \in J^b} \beta_{i,j}$ counts the number of jobs of box type b scheduled for day j. As earlier defined, $\beta_{i,j} = 1$ if a job i is scheduled for day j, otherwise $\beta_{i,j} = 0$.

To avoid that two jobs of the same box type are produced in parallel we let

$$\beta_{i,j} + \beta_{i,j+1} + \beta_{p,j} + \beta_{p,j+1} \leq 3, \quad j = 1, ..., M-1, \quad p \in J^b, \quad p > i, \quad i \in J^b, \quad b = 1, ..., B$$

and

$$\beta_{i,j-1} + \beta_{i,j} + \beta_{i,j+1} + \beta_{p,j} \leq 3, \quad j = 2, ..., M-1, \quad p \in J^b, \quad p > i, \quad i \in J^b, \quad b = 1, ..., B.$$

The tardiness of a job i is, as earlier noted, equal to $t_i = \max\{0, l_i\}$, and the lateness is equal to $l_i = c_i - D_i$. The last production day c_i for job i is then found by the following expressions,

$$c_i \geq j(\beta_{i,j} - \beta_{i,j+1}), \quad j = 1, ..., M-1, \quad i = 1, ..., N \tag{8}$$

$$c_i \geq M\beta_{i,M}, \quad i = 1, ..., N. \tag{9}$$

The tardiness of job i is given by

$$t_i \geq c_i - D_i, \quad i = 1, ..., N \tag{10}$$

where

$$t_i \geq 0, \quad i = 1, ..., N. \tag{11}$$

To assure that every job is assigned a set of production days, we let the sum of blades scheduled for each job be equal to the number of blades requested,

$$\sum_{j=1}^{M} \delta_{i,j} \geq O^i, \quad i = 1, ..., N. \tag{12}$$

To limit the search for an optimal solution, we add the following obvious facts. First, the completion time for a job i has to be less or equal to the value of $\min\{M,\ D_i + tt\}$, where tt is an upper bound for the total tardiness in this example. That is

$$c_i \leq \min\{M,\ D_i + tt\}, \quad i = 1, ..., N. \tag{13}$$

Further, we know that the tardiness of a job i has to be less or equal to the value of $\min\{tt,\ M - c_i\}$, that is

$$t_i \leq \min\{tt,\ M - c_i\}, \quad i = 1, ..., N. \tag{14}$$

Finally, we know that the number of propeller blades scheduled for a day j has to be less or equal to the value of $\min\{maxcap_b,\ O^i\}$, where $maxcap_b$ is the maximum allowed capacity for box-type b, i.e. $maxcap_b = \max_{k=1,...,K}\{COMB_b^k\}$. That is

$$\delta_{ij} \leq \min\{maxcap_b,\ O^i\}, \quad i = 1, ..., N. \tag{15}$$

4.2. MILP - problem

To sum up, the scheduling problem can be expressed as the following mixed integer linear programming (MILP) problem:

$$\min_{\beta_{i,j},\delta_{i,j},x_{i,j}} \quad \sum_{i=1}^{N} t_i \tag{16}$$

subject to

$$\begin{aligned}
\beta_{i,j} * O^i - \beta_{i,j-1} O^i + \sum_{p=1}^{j-1} \delta_{i,p} &\geq 0, \quad j = 1, ..., M, \quad i = 1, ..., N \\
\delta_{i,j} - \beta_{i,j} &\geq 0, \quad j = 1, ..., M, \quad i = 1, ..., N \\
\beta_{i,j} * O^i - \delta_{i,j} &\geq 0, \quad j = 1, ..., M, \quad i = 1, ..., N \\
\sum_{k=1}^{K} COMB_b^k x_{k,j} - \sum_{i \in J^b} \delta_{i,j} &\geq 0, \quad j = 1, ..., M, \quad b = 1, ..., B \\
\sum_{k=1}^{K} x_{k,j} &= 1, \quad j = 1, ..., M \\
\sum_{i \in J^b} \beta_{i,j} &\leq 2, \quad j = 1, ..., M, \quad b = 1, ..., B \\
\beta_{i,j} + \beta_{i,j+1} + \beta_{p,j} + \beta_{p,j+1} &\leq 3, \quad j = 1, ..., M-1, \quad p \in J^b, \quad p > i, \quad i \in J^b, \quad b = 1, ..., B \\
\beta_{i,j-1} + \beta_{i,j} + \beta_{i,j+1} + \beta_{p,j} &\leq 3, \quad j = 2, ..., M-1, \quad p \in J^b, \quad p > i, \quad i \in J^b, \quad b = 1, ..., B \\
c_i - (j-1)(\beta_{i,j-1} - \beta_{i,j}) &\geq 0, \quad j = 1, ..., M, \quad i = 1, ..., N \\
c_i - M\beta_{i,M} &\geq 0, \quad i = 1, ..., N \\
t_i - c_i + D_i &\geq 0, \quad i = 1, ..., N \\
\sum_{j=1}^{M} \delta_{i,j} - O^i &\geq 0, \quad i = 1, ..., N \\
c_i &\leq \min\{M,\ D_i + tt\}, \quad i = 1, ..., N \\
t_i &\leq \min\{tt,\ M - c_i\}, \quad i = 1, ..., N \\
\delta_{ij} &\leq \min\{maxcap_b,\ O^i\}, \quad i = 1, ..., N
\end{aligned}$$

$$
\begin{aligned}
t_i &\geq 0, \quad i = 1, ..., N \\
\beta_{i,j} &\in \{0,1\}, \quad j = 1, ..., M, \quad i = 1, ..., N \\
x_{k,j} &\in \{0,1\}, \quad k = 1, ..., K, \quad j = 1, ..., M \\
\delta_{i,j} &\geq 0, \quad integer, \quad j = 1, ..., M, \quad i = 1, ..., N
\end{aligned}
$$

As seen from the above, the model contains a large number of integer variables. This is the case even when the number of jobs is small. To reduce the size of the problem, M should be as tight as possible. An upper bound on M can be found from the sum $\sum_{i=1}^{N} pt_i$, where pt_i is the shortest number of processing days for job i. Given the predetermined box-combinations, $pt_i = \lceil \frac{O^i}{maxcap_b} \rceil$, where $i \in J^b$ and $maxcap_b = \max_{k=1,...,K}\{COMB_b^k\}$. Here $\lceil x \rceil$ indicates the smallest integer larger or equal to x.

An upper bound on M can also be found by analyzing a solution to (16) found by a good heuristic. If we let S denote any schedule obtained from a heuristic, and let T_S denote the total tardiness for this schedule, then M can be found from

$$
M \leq \max_{i=1,...,N} D_i + T_S \tag{17}
$$

The upper bound on M can then be refined through a back-tracking principle, exploiting gaps between due dates together with information on production capacity.

4.3. Numerical example

We shall present an example where ten jobs are to be processed using five different box types. Table 2 gives data for each box type. The objective is to schedule these ten jobs using the box combinations in Table 1 such that total tardiness is minimized. From the bounding technique indicated in Section 4.2 we find that a maximum of thirteen days is needed to schedule these jobs. Further, from the heuristics presented in the next section we find an upper bound on total tardiness equal to three.

If we let $M = 13$ in (16), we get a MILP problem with 410 variables (among them 260 binary variables). CPLEX (version 7.0) solved this problem on an Ultra-1/170E at 167 MHz using 18 seconds, given the upper bound on total tardiness. The optimal solution scheduled the jobs in 11 days, with a total tardiness of 3. The optimal sequence of box combinations was $\{comb^2, comb^2, comb^2, comb^{10}, comb^{10}, comb^8, comb^8, comb^3, comb^3, comb^4, comb^1\}$. The production plan is presented in Table 3. Here, $\delta_{i,j}$ is, as noted earlier, equal to the number of blades scheduled at day j for job i.

CPLEX uses a branch and bound method to solve MILP type formulations. Through parameter settings, branching can be guided and the number of cuts reduced. Guiding the branching did not have any significant impact on the solution time or node size. Unfortunately, due to memory allocation problems, CPLEX was only able to solve small problem instances of (16). When the problem size gets larger than 10 jobs and 13 days, CPLEX encounters memory allocation problems.

5. Heuristic procedures

Since the analytical approach offers working algorithms only for small-sized problems, we will discuss heuristic procedures to solve (16). First, the improved due-date approach currently in use in the company is presented. Then, a four-stage heuristic search procedure is proposed. The main part of this heuristic is a two-stage heuristic procedure that in the first stage finds a suitable box

combination k (and the set of jobs it will process) for each day j based on the set of jobs still to be scheduled. In the second stage, the heuristic procedure improves the production plan found in the first stage by rescheduling late jobs to an earlier point in time. Both the third and fourth stage of the four-stage heuristic is built up around the two-stage heuristic. The third stage (also called the extended two-stage heuristic) has a loop that iterates through all box combinations. For each step in the loop, it initially schedules the current box combination at day 1, then a slightly modified two-stage heuristic is used to find the resulting production plan. In the fourth and final stage of the heuristic we try to schedule a new combination with more boxes for each processing day of a tardy job i ($t_i > 0$). For each day the procedure succeeds in finding such a combination, the two-stage heuristic is used to generate a new production plan. Each of the heuristic procedures proposed will find a complete production plan pp for the N jobs, where pp states which combination to use and which jobs to process for each day in the production process.

Heuristic solution procedures defined by other authors for problems similar to the outlined problem were discussed in Section 3. However, mainly due to the complex nature of the problem (the impact the scheduling of jobs and combination at day j have on jobs and combinations that can be scheduled on day $j+1$), none of these solution procedures seems to be suitable for solving the problem outlined.

5.1 Due-date approach

Initially assume that all jobs are sorted in a job list according to due date. The due-date approach will then schedule each job in the order it appears in the job list. The i'th job in the list is scheduled at the first possible day, considering the restrictions set in pp by the $i-1$ first jobs in the list. Each job will be scheduled with the maximum capacity available for each day until it is completed. A recursive algorithm is then used that try to swap a job finished ahead of time with a late job in the initial due-date plan. The recursive algorithm reduces the total number of days the jobs are late by roughly 10%. The plan generated by this approach was the initial solution procedure in use at the foundry, and will be used as a base for comparison with other heuristic procedures. We will refer to this recursive algorithm as the improved due-date approach. The approach was implemented by one of the authors after several companies had turned down the foundries request for help, due to the complexity of the problem. The idea was to create an initial production plan that tries to minimize tardiness considering only due dates of the jobs (due-date approach). Then to improve this plan by recursively reschedule late jobs with early jobs in a random fashion that indirectly also take into account both job sizes and suitable combinations.

Example:

For the problem presented in Table 2 the initial due-date sequence is equal to: O^1, O^2, O^9, O^{10}, O^4, O^6, O^7, O^5, O^8, O^3. If more than one order is due at a given day, the order with smallest size according to the maximum capacity is sequenced first. Each order in the sequence is scheduled with as much capacity as possible, using the combinations from Table 1. The different box combinations used are given in Table 4 and the corresponding schedule is given in Table 5. The initial tardiness is equal to 11. After shuffling late jobs with early jobs the tardiness will display an average decrease of about 10% for the improved due-date approach. □

5.2 The first stage of the two-stage heuristic (The first-stage heuristic)

A new two-stage heuristic procedure will be suggested to solve (16). In the first stage we iteratively assume that for each day j a current production plan pp is found for the first $j-1$ days. The jobs to be scheduled in pp at day j then depend on the set of jobs being processed at day $j-1$, the set

of boxes that have some available capacity at day $j-1$ and the set of jobs still not scheduled up to day j. In the second stage, total tardiness is improved by rescheduling late jobs in *pp*.

The following notation is used:

$sjob_b^{j-1}$	- the job scheduled in *pp* for box b at day $j-1$
$size_b^{j-1}$	- the number of propeller blades left to schedule for job $sjob_b^{j-1}$ at day $j-1$
$hole_b^{j-1}$	- the available capacity in *pp* for box b at day $j-1$, after job $sjob_b^{j-1}$ is completed
Q_b	- the ordered set of the jobs still to be scheduled in *pp* at box b
Q_b^r	- job number r in Q_b
$O^{Q_b^r}$	- the number of propeller blades requested for job Q_b^r
$tard_b$	- the estimated total tardiness for the jobs in Q_b
$tardinc_b^k$	- the increase in $tard_b$, using combination k at day j
$tardinc^k$	- the sum of the increase in total tardiness using combination k at day j, i.e. $\sum_{i=1}^{B} tardinc_b^k$.

If two jobs of box-type b are scheduled at day j, $sjob_b^j$ includes only the last. Further, if no job of box-type b is scheduled at day j, $sjob_b^j = 0$ and $size_b^j = 0$. To avoid job splitting, combination k for day j has to be selected such that, $COMB_b^k > 0$ for at least every index b where $size_b^j > 0$. Let every combination k that satisfies this restriction for day j, be called a *feasible* combination. Note that, if a job i is to be scheduled at day j and $hole_b^{j-1} > 0$ we fill the capacity hole at day $j-1$ before we occupy capacity at day j (Step 3 (a) ii. A.).

The first stage of the heuristic procedure is given by the following steps.

Initial step: Set $sjob_b^0 = 0$, $size_b^0 = 0$ and $hole_b^0 = 0$ for $b = 1, ..., B$, set $j = 1$ and initialize Q_b, $b = 1, ..., B$.

1. For every feasible combination k at day j calculate $tardinc^k$. Then, find the combination min that satisfies $tardinc_{min} \leq tardinc^k$, $k = 1, ..., K$.

2. Use combination min at day j in the current plan *pp*.

3. For every box-type b that is available for use at day j, i.e. $comb_b^{min} > 0$, find a job i to schedule at day j in accordance with the following:

   (a) If $size_b^{j-1} = 0$

      i. if Q_b is empty, no job is found for box-type b.

      ii. if Q_b is not empty

         A. if $hole_b^{j-1} > 0$, schedule job Q_b^1 at day $j-1$ with $hole_b^{j-1}$ propeller blades

         B. schedule job Q_b^1 at day j with $min(O^{Q_b^1} - hole_b^{j-1}, comb_b^{min})$ blades

         C. remove the first job from Q_b.

   (b) If $size_b^{j-1} > 0$, schedule job $sjob_b^{j-1}$ at day j with $min(size_b^{j-1}, comb_b^{min})$ blades.

4. If job i is completed at day j and the first job in Q_b can be completed by the spare capacity at day j ($O^{Q_b^1} < hole_b^j$), then

   (a) schedule job number Q_b^1 at day j, update $hole_b^j$.

5. $j = j + 1$. Update $sjob^j$, $size^j$, Q_b, $hole_b^j$, $b = 1, ..., B$ and pp. If Q_b is empty for every b, goto Step 6, else goto Step 1.

6. Quit.

Further, to avoid large holes of unused capacity in the plan pp, we can at Step 4 also schedule a new job Q_b^1 at day j if $hole_b^j$ is above a constant C_{hole}, where C_{hole} is an upper bound for the capacity we allow to be unused for a given day. This will reduce the total tardiness in $tardinc_b^k$, but may generate late jobs in other box types. This can be a problem if the size of job Q_b^1 is large and the jobs in the other box types waiting to be processed have a critical due date. Whether it is best to have some unused capacity in the plan pp or not depends on the size and due dates of the sets Q_b.

Calculation of $tardinc^k$
During each step of Stage 1 the jobs in Q_b is sequenced such that

- for every neighbour pair (i,p), p is in front of i if

$$D_i < d_p, \quad pt_i > pt_p \quad and \quad c_i > d_p,$$

  where processing time and completion time are calculated according to use of maximum capacity.

The estimated processing time used for job i is the same as the minimum processing time presented earlier, that is $pt_i = O^i/maxcap_b$. Here, $maxcap_b$ is the maximum allowed capacity for box-type b, i.e. $maxcap_b = \max_{k=1,...,K}\{COMB_b^k\}$. If we assume that the jobs in Q_b can be processed in pp from day $j + size_b^j/maxcap_b$ successively, an estimate for the total tardiness in Q_b can be calculated equal to

$$tard^b = \sum_{i \in Q_b} (c_i - D_i)^+ \tag{18}$$

Here, $(x)^+ = \max(0, x)$. Further, $c_i = j + size_b^j/maxcap_b + PT_i + pt_i$, where PT_i is the sum of the processing times for the jobs in Q_b sequenced before job i and $size_b^j/maxcap_b$ is an estimate of the remaining processing time for job $sjob_b^j$ at day j.

Since the objective is to minimize total tardiness, the first job in the ordered Q_b will always be the next job to schedule at box b from the jobs in Q_b. Now, if a combination k is chosen for day j with zero capacity or a capacity below $maxcap_b$ for box b, the total tardiness in Q_b will in general increase. If the combination has zero capacity for box b, $tard^b$ will increase with one unit for each late job in Q_b. Otherwise, if combination k has capacity below $maxcap_b$ for box b, we estimate that $tard^b$ will increase with $(1 - COMB_b^k/maxcap_b)$ units for each late job in Q_b.

Let σ_i be a zero, one variable that indicates if job i is tardy. Assume $\sigma_i = 1$ if job i is tardy and $\sigma_i = 0$ otherwise. We know that job i is tardy if $c_i > D_i$, where c_i is calculated as previously described. Then,

$$tardinc_b^k = \Big(\sum_{i \in Q_b \cup sjob_b^j} \sigma_i \Big) \times (1 - COMB_b^k/maxcap_b). \tag{19}$$

$tardinc_b^k$ is as earlier noted the increase in total tardiness combination k generates for box b. Here we have that $tardinc_b^k = 0$ if either $COMB_b^k = maxcap_b$ or $\sum_{i \in Q_b \cup sjob_b^j} \sigma_i = 0$. Further, $tardinc_b^k$

equals the total number of late jobs in Q_b if $COMB_b^k = 0$.

The estimated increase in total tardiness generated over all job streams Q_b using combination k at day j, is equal to $tardinc^k = \sum_{b=1}^{B} tardinc_b^k$.

To minimize the total number of late jobs in the production plan pp it is important to utilize the capacity that is available for each day in the plan. Therefore, at a given day j we would like to give a bonus to those combinations that can fill available capacity the day before. More specific for day j a bonus equal to $hole_b^{j-1}/maxcap_b$ will be subtracted from $tardinc^k$ if $COMB_b^k > 0$. Further, if only one job remains in Q_b and this job is tardy, we subtract $2 \times hole_b^{j-1}/maxcap_b$ from $tardinc^k$, because this job may in addition to available capacity at day $j-1$ generate available capacity on its last processing day.

The combination with the smallest increase in total tardiness over all box-types is then given by

$$tardinc_{min} = \min_{k=1,\ldots,K}\{tardinc^k\}$$

If more than one combination k satisfy this expression, and $tardinc_{min} > 0$, the combination with the largest total capacity calculated over all box types with orders left to produce is chosen. If $tardinc_{min} = 0$ for more than one combination k the combination with largest total capacity according to the jobs left to produce is chosen. For each combination k this number is calculated as the sum of $COMB_b^k \times PropellersLeftToProduce_b/maxcap_b$ for all box-types b, where $PropellersLeftToProduce_b$ is the total number of propellers left to produce of box-type b at day j.

A simple summary of the previously defined procedure follows:

1. For each box-type b, find the number of late jobs in Q_b and $sjob_b^j$, i.e. find $\sum_{i \in Q_b \cup sjob_b^j} \sigma_i$.
2. For each box-combination k:
   - find $tardinc_b^k$, the increase in tardiness for each box type.

   Calculate $tardinc^k$.
3. If $hole_b^{j-1} > 0$ and $COMB_b^k > 0$, subtract $hole_b^{j-1}/maxcap_b$ from $tardinc^k$.
4. If there is only one tardy job in Q_b, $hole_b^{j-1} > 0$ and $COMB_b^k > 0$, subtract $hole_b^{j-1}/maxcap_b$ from $tardinc^k$.
5. Calculate $tardinc_{min}$.

Example:
We will use the problem presented in Table 2 to illustrate the first stage of the two-stage heuristic. Initialization: $j = 1$, $Q_1 = \{O^1, O^2, O^3\}$, $Q_2 = \{O^4, O^5\}$, $Q_3 = \{O^6, O^7, O^8\}$, $Q_4 = \{O^9\}$, $Q_5 = \{O^{10}\}$, $hole^0 = (0,0,0,0,0)$, $sjob^0 = (0,0,0,0,0)$, $size^0 = (0,0,0,0,0)$.

The combination to use at day i can be calculated as follows:

- Day 1: For all box-types b, if no order of this box type is scheduled at day 1, increase in total tardiness will be zero, i.e. $\sum_{i \in Q_b \cup sjob_b^j} \sigma_i = 0, \quad b = 1, ..., B$. This implies that $tardinc_{min} = 0$ and the combination with largest total capacity according to the jobs left to produce should be scheduled at day 1. Combination 3 satisfy this condition, therefore combination (1,3,2,0,0) and the orders O^1, O^4 and O^6 should be scheduled at day 1.

  Update: $sjob^1 = (O^1, O^4, O^6, 0, 0)$, $size^1 = (6, 2, 1, 0, 0)$, $hole^1 = (0, 0, 0, 0, 0)$, $Q_{70} = \{O^1, O^2, O^3\}$, $Q_{100} = \{O^4, O^5\}$, $Q_{115} = \{O^6, O^7, O^8\}$, $Q_{170} = \{O^9\}$, $Q_{215} = \{O^{10}\}$.

- Day 2: Only combination 3 is possible for day 2, $tardinc_{min} = tardinc^2$. Combination (1,3,2,0,0) and the orders O^1, O^4 and O^6 should then be scheduled at day 2.

  Update: $sjob^2 = (O^1, 0, 0, 0, 0)$, $size^2 = (5, 0, 0, 0, 0)$, $hole^2 = (0, 1, 1, 0, 0)$, $Q_{70} = \{O^1, O^2, O^3\}$, $Q_{100} = \{O^5\}$, $Q_{115} = \{O^7, O^8\}$, $Q_{170} = \{O^9\}$, $Q_{215} = \{O^{10}\}$.

- Day 3: Combination 1, 2, 3, 4, 5, 6 and 9 is possible for day 3. If no order of box-type 170 or 215 is scheduled at day 3, the increase in total tardiness is 1 for both box types, i.e. $\sum_{i \in Q_b \cup sjob_b^j} \sigma_i = 1, \quad b = 170, 215$. For the other box types $\sum_{i \in Q_b \cup sjob_b^j} \sigma_i = 0$. From this we can find that $tardinc_{min} = tardinc^6$, see Table 6. Combination (2,0,2,0,1) should then be scheduled at day 3, together with the orders O^1, O^7 and O^{10}.

  Update: $sjob^3 = (O^1, 0, O^7, 0, O^{10})$, $size^3 = (3, 0, 0, 0, 3)$, $hole^3 = (0, 0, 1, 0, 0)$, $Q_{70} = \{O^1, O^2, O^3\}$, $Q_{100} = \{O^5\}$, $Q_{115} = \{O^8\}$, $Q_{170} = \{O^9\}$, $Q_{215} = \{O^{10}\}$.

- Day 4: Combination 6 and 9 is possible for day 4. If no order of a box type is scheduled at day 4, the increase in total tardiness is 1 for box-types 70, 170 and 215, and zero otherwise. From this we can find that $tardinc_{min} = tardinc^9$. Combination (1,0,0,1,1) should then be scheduled at day 4, together with the orders O^1, O^9 and O^{10}.

  Update: $sjob^4 = (O^1, 0, 0, O^9, O^{10})$, $size^4 = (2, 0, 0, 7, 2)$, $hole^4 = (0, 0, 0, 0, 0)$, $Q_{70} = \{O^1, O^2, O^3\}$, $Q_{100} = \{O^5\}$, $Q_{115} = \{O^8\}$, $Q_{170} = \{O^9\}$, $Q_{215} = \{O^{10}\}$.

- Day 5 up to day 11 are scheduled according to Table 7. From this we find that the total tardiness for the first-stage heuristic is equal to 11.

5.3. The second stage of the two-stage heuristic (The second-stage heuristic)

To improve the plan *pp* found by the first stage, a new procedure will be outlined that iteratively creates new plans *npp* by rescheduling late jobs in *pp*. In the new plan we make use of an updated processing time $pt_i = O^i / average_b$, where $average_b$ is the average capacity used for jobs of box-type b in the initial plan *pp*.

To reduce the total tardiness in the complete production plan *pp*, we will try to reschedule every late job in *pp* to an earlier point in time. Let job r be a late job in *pp*, scheduled from day r_{start} to day r_{stop} at box-type b'. Further, let the vector $jobcomb$ include the last jobs scheduled for each box type before job r. Now we will successively try to schedule job r of box-type b' before a job s in the vector $jobcomb$ in order to improve *pp*. Let the job s be scheduled from s_{start} to s_{stop} in *pp*

first for the case where $s_{start} < r_{start}$. Let p_r and p_s be the real processing time for respectively job r and job s in *pp*. An estimate for the total tardiness by scheduling job r in front of job s is then,

$$diff_{s,r} = (t_r - (r_{start} - s_{start}))^+ + (s_{start} + p_r + p_s - d_s)^+. \tag{20}$$

The first term gives the total tardiness by scheduling job r at day s_{start}, while the second term gives the total tardiness by scheduling job s after job r is completed. $x^+ = max(0, x)$ as earlier defined. If $diff_{r,s} < t_r + t_s$, job r will be scheduled at day s_{start} using combination k. Here we assume that it is possible to find a box combination k that allows us to include job r at day s_{start}, either by removing job s, or by finding a new combination that in addition to the current jobs at day s_{start} also can include job r. If no such combination can be found, job r can not be scheduled before job s. If more than one box combination can include job r at day s_{start}, we schedule the combination k with largest number of b' boxes (considering $COMB^k_{b'} <= O^r$). If more than one combination fit this description we schedule the one that fits the set of remaining orders best. If $s_{start} = r_{start}$ we will try to schedule job r at day s_{start} with a combination k that have more boxes available of type b' then the current combination at day s_{start}.

If a new feasible combination is found at day s_{start}, a new plan *npp* is generated where *npp* is set equal to *pp* for the first s_{start} production days. Job r is here inserted at day s_{start} using combination k. The other jobs scheduled at day s_{start} are updated using the new combination k. A complete plan for all the N jobs can be found using the first-stage heuristic starting at $s_{start} + 1$, employing the updated estimated processing times to find the increase in tardiness for each job sequence Q_b. If an improved plan is found, it is saved in *tpp* where *tpp* contains the production plan with minimum total tardiness among the plans generated from rescheduling job r.

Job s of box type $b = b'$ is an exception for the procedure described above. Instead of scheduling job r before job s, we first try to schedule job r immediately after s in the new plan *npp*. That is, either at the completion day of job s (if there is spare capacity at this day) or on the following day. If $s_{start} = r_{start}$ both jobs are tried rescheduled to an earlier point in time. Secondly we try to reschedule job r before job s since this should reduce tardiness if job s is late and job r is less than job s.

When we have tried to reschedule job i before every job in *jobcomb*, we have set *pp* equal to *tpp* if the total tardiness is less in *tpp*. The improved heuristic is then repeated for every late job r sequenced after i in the current plan *pp*. If the production plan *pp*, generated after all late jobs have been considered for rescheduling, is improved, the second stage of the two-stage heuristic will be used on the improved plan. This will be repeated until no improvements are found. Six different methods to sequence the late jobs in *pp* for rescheduling have been proposed in Section 6. These methods reschedule late jobs according to the size of the box type where they belong and according to the number of late jobs in the box type.

From the discussion above we are now ready to give a more formal description of the second stage of the heuristic procedure. The procedure is given by the following steps:

Initially: Let $average_b$ be the average capacity used for jobs of box-type b in the initial plan *pp* found from the first-stage heuristic. Further, let the processing time $pt_i = O^i / average_b$ for all $i \in J^b$. Finally, arrange the jobs in J^b according to the predetermined sequencing policy.

Initial step: Let job r be the first job in J^b that is late. Further, set the temporarily plan tpp equal to pp.

1. Let job r be of box-type b' and scheduled in pp from day r_{start} to day r_{stop}. Let the vector $jobcomb$ include the last jobs scheduled of each box type before day r_{start}.

2. For each box-type $b \neq b'$:

   (a) For simplicity, let $s = jobcomb_b^k$. Further, let s be scheduled in pp from day s_{start} to day s_{stop}.

   (b) Calculate
   $$diff_{s,r} = (t_r - (r_{start} - s_{start}))^+ + (s_{start} + p_r + p_s - d_s)^+$$
   where p_r and p_s are the real processing times for respectively job r and job s in pp.

   (c) If $diff_{s,r} < t_s + t_r$:

   - For each box-combination k,
     - if it is possible to schedule job r at day s_{start} through combination k (job s can be removed if necessary), calculate $tardinc^k$
     - else $tardinc^k = \infty$.
   - Find the combination min that satisfies $tardinc_{min} \leq tardinc^k$, $k = 1, ..., K$.
   - If $min < \infty$, use combination min at day j in the current plan pp. Calculate a new plan npp from day s_{start} using the first-stage procedure.
   - If total tardiness in npp is less than the total tardiness in tpp, set the temporarily plan $tpp = npp$.

3. If $b = b'$ and $s_{start} < r_{start}$:

   (a) For each box-combination k :

   - if it is possible to schedule job r at day $s_{start} + 1$ through combination k calculate $tardinc^k$
   - else $tardinc^k = \infty$.

   (b) Find the combination min that satisfies $tardinc_{min} \leq tardinc^k$, $k = 1, ..., K$.

   (c) If $min < \infty$, use combination min at day j in the current plan pp. Calculate a new plan npp from day $s_{start} + 1$ using the first stage procedure.

   (d) If total tardiness in npp is less than the total tardiness in tpp, let the temporarily plan $tpp = npp$.

4. If $b = b'$ and $s_{start} = r_{start}$:

   (a) If $o^r < o^s$, schedule job r in front of job s in the current schedule pp.

   (b) Try to reschedule both jobs to an earlier point in time.

   - Goto Step 1 with job r and the updated pp.

5. Set the original plan pp equal the improved plan tpp, $pp = tpp$.

6. If there are more late jobs in J^b, let job r be the next late job in J^b and goto Step 1. Else goto Step 7.

7. Quit.

Note that when calculating $tardinc_b^k$ (see Equation 19) for $tardinc^k$, we use $maxcap_b$ to estimate the completion date $c_{sjob_b^j}$ when calculating if job $sjob_b^j$ is tardy, and $average_b$ to estimate the completion date c_i when calculating if job $i \in Q_b$ is tardy.

Example:
We will now apply the second-stage heuristic to refine the production plan found from using the first-stage heuristic on the problem presented in Table 2. Based on the different sequencing policies proposed in Section 6 we will use method 2, which try to reschedule the late jobs one by one in decreasing size of box-type number. From the initial plan *pp* made by the first-stage heuristic, the following sequence of late jobs is found, O^9 of box-type 170, O^5 of box-type 100, and O^1, O^2 of box-type 70, see Table 7.

From $r = O^9$, we find that $jobcomb = (O^1, O^4, O^7, 0, O^{10})$. Rescheduling O^9 before O^1 would give a new combination at day 1, where box type 170 is included and box-type 70 removed (if necessary). Only one combination fits this requirement, that is combination 7. Using combination 7 on day 1 will give a new plan *npp* with tardiness equal to 14. Rescheduling O^9 before O^4 results in scheduling combination 4 at day 1. Combination 4 is the only combination that can include box-type 170 and remove box-type 70 at day 1. The total tardiness obtained for the production plan generated when scheduling combination 4 at day 1 is 7, see Table 8. This plan is an improvement over the old plan and therefore stored in *tpp*. It is not possible to reschedule O^9 before O^7. Rescheduling O^9 before O^6 results in using box-type combination 5 on day 1. From this, a new production plan *npp* with tardiness equal to 12 is found. Rescheduling O^9 before O^{10} results in using box-type combination 5 on day 3. From this we can generate a new plan *npp* with tardiness equal to 9.

We have now tried to reschedule O^9 before each job in *jobcomb*. Rescheduling O^9 before O^4 resulted in the plan *tpp* with total tardiness equal to 7. Since $pp > tpp$ we let $pp = tpp$ and start to reschedule the next late job from the new plan.

Previously we have scanned through late jobs of box-type 170 and 215. From the current plan *pp*, we have only one late job left among the other box types, this is O^5 of box-type 100. From Table 8 we see that O^4 is scheduled right in front of O^5. In this case we try to reschedule O^4 to an earlier point of time and make sure that O^5 is scheduled right after. Then we let $r = O^4$ and find that $jobcomb = (O^2, 0, O^6, O^9, 0)$. Rescheduling O^4 before O^2 will result in rescheduling O^4 before O^1 since at day 3 both O^1 and O^2 are scheduled. This again results in scheduling combination 7 at day 1. Using combination 7 at day 1 would give a new plan *npp* with tardiness equal to 15. Rescheduling O^4 before O^6 will result in scheduling combination 5 at day 1. This will result in a total tardiness of 12 for the corresponding production plan. Rescheduling O^4 before O^9 will result in scheduling combination 3 at day 1. This will result in a total tardiness of 11 for the corresponding production plan.

The first iteration of the second stage procedure generates a production plan *pp* with a total tardiness equal to 7. Since this is an improvement over the initial plan, from which this iteration started, we repeat the second stage procedure on the new plan *pp*. No improvement in total tardiness is obtained in this iteration, therefore no further iterations are performed.□

The example above is chosen to show improvements in each step of the final four-stage heuristic.

Therefore we have accepted the fact that in this section we only have examples where late jobs are rescheduled forward to day 1. However, Section 5.5 will have an example where a tardy job is rescheduled forward to another day.

5.4 The extended two-stage heuristic (The third-stage heuristic)
In the two-stage heuristic we have tested more than one box combination only for some specific days in the plan, i.e., for days we think we did a priority mistake concerning the next job to schedule. For these days we have in the two-stage heuristic tried to schedule tardy jobs that in the previous stage were considered for scheduling but not prioritized.

If we for one arbitrary day should test more than one box combination, it would be both wise and convenient to choose the first production day. Generally, the earlier a priority mistake considering which combination of job types to schedule for a day is made, the more severe the mistake would be. The extended two-stage heuristic is therefore implemented with a loop that iterates through all the box combinations on the first production day. For each step in the loop, we initially schedule the current box combination at day 1, then the first-stage heuristic is used to complete the plan and the second-stage heuristic is used to reschedule late jobs in an attempt to improve the plan.

Example:
By fixing the second combination at the first day and then using the two-stage heuristic we get a production plan with total tardiness equal to 5. The details of the plan are found in Table 9. □

5.5 Final stage of the four-stage heuristic (The fourth-stage heuristic)
For the fourth and final stage of the heuristic we have proposed a procedure *CombUpdate* that for each production day j, for a late order of type b, tries to replace the current combination with a new combination that has a larger number of boxes of type b. For each new replacement the procedure uses the first-stage heuristic to complete the plan, and the second-stage heuristic to try to improve the plan. If an improved production plan is obtained, the search for new replacements continues in this plan. To guide the choice of which combination to use for each day, the procedure uses average processing time to estimate the completion day when calculating the tardiness generated for each combination. This is done both in the first and in the second stage of the two-stage heuristic.

Example:
From the current best plan, *pp* in Table 9, we see that the jobs numbered as 5, 9 and 10 are late. No improvement in total tardiness is obtained by using *CombUpdate* on job number 5. However using *CombUpdate* on job number 9 resulted in a new production plan with total tardiness equal to 4. This solution is obtained by changing the combination at day 3 from number 4 to number 2. A one unit increase in propeller blades scheduled for job number 9 at day 3 is then obtained. See Table 10 for details. Now, using the second stage heuristic on this schedule, we are able to decrease the total tardiness to 3, which is the optimal solution. This was obtained by rescheduling O^{10} before O^{6}. Combination number 10 is then scheduled at day 4. See Table 3 and Table 11 for further details. □

5.6 The four-stage heuristic and the extended four-stage heuristic
The four-stage heuristic consist of four different procedures used in the following manner, first the two-stage heuristic (first and second stage), then the extended two-stage heuristic (third stage), and finally the procedure *CombUpdate* (fourth stage). An extended version of the four-stage heuristic is also implemented. The extended four-stage heuristic uses the same procedures but in a different

manner, first it uses the two-stage heuristic, then after each iteration in the extended two-stage heuristic it uses the *CombUpdate* procedure. Obviously, the extended four-stage heuristic will do a more thorough and wide search for an optimal production plan, but with the cost of an increase in the solution time.

6. Numerical Results

In this section we shall discuss the performance of the four-stage heuristic and the extended four-stage heuristic. In order to get a good quality of the final production plan, the initial plan should be loose enough to give room for the rescheduling algorithm to test several different job sequences, but tight enough to give the rescheduling some guidance on where the effort should be concentrated. The sequence in which the jobs are rescheduled, both in the second stage of the two-stage heuristic and in the use of *CombUpdate*, can make a significant difference for the final production plan. This is also the case when we estimate total tardiness for a given combination. Furthermore, a bad choice of combination on the completion day of an order may generate holes (idle times) in the production plan that will cause unnecessary tardiness for the involved box types at a later time. In order to test the performance of the two heuristics thoroughly, we have proposed six different rescheduling policies, different methods to estimate tardiness, and three different policies to fill holes in the production plan. Further, when calculating the increase in tardiness for box-type b using combination k at day j ($tardinc_b^k$), we have in the two last stages of the heuristic used both $average_b$ and $maxcap_b$ to estimate the completion date for jobs not yet completed.

6.1 Presentation of procedures used for the numerical results

In the second-stage heuristic we have used six different methods to reschedule the orders:

1. Late jobs are rescheduled according to increasing size of box types. Jobs within a box type are rescheduled according to their appearance in the schedule.

2. Late jobs are rescheduled according to decreasing size of box types. Jobs within a box type are rescheduled according to their appearance in the schedule.

3. Late jobs are rescheduled according to the box type where they belong. Jobs from the box types with largest tardiness are scheduled first. Jobs within a box type are rescheduled according to their appearance in the schedule.

4. Late jobs are rescheduled according to the box type where they belong. Jobs from the box types with smallest tardiness are scheduled first. Jobs within a box type are rescheduled according to their appearance in the schedule.

5. Late jobs, scheduled after a current day *start*, are rescheduled according to their appearance in the production plan. *Start* is the day after the first production day for the last order that is rescheduled. Initially *start* is set equal to one.

6. Late jobs, scheduled after a current day *start*, are rescheduled according to their appearance in the production plan. *Start* initially is set equal to one and then updated with one for each new order that is rescheduled.

The different rescheduling policies are chosen such that they fulfill each other.

Depending on the size of the problem and the distribution of the jobs into box types, some of the rescheduling policies may perform better on some problems than on others. But generally, due to the randomness included in the determination of a schedule, we would believe that the performance of the rescheduling methods will be quite insensitive to problem type.

For all problems tested we have used two different routines to estimate the tardiness generated by scheduling combination k at day j. One of them is presented in Section 5.2. The other differs mainly by using more realistic processing times on the jobs scheduled but not yet completed at day $j-1$.

Three procedures are also tested that rearrange jobs in order to avoid inefficient holes in the plan. These procedures try to 1) include larger jobs that fits the holes better, 2) schedule new jobs successively after a hole, and 3) increase the number of boxes scheduled on days in front of a hole.

6.2 Presentation of data used for the numerical results
Two categorizes of data are used for the numerical results. One set of small-sized data that can be solved using a MILP solver, and one larger more realistic set of data. The small set are used to test the four-stage heuristic up against solutions from a MILP solver, while the larger set are used to show the performance of the four-stage heuristic compared to previously and current implemented solution procedures for the foundry.

6.2.1 Presentation of data for the case of small-sized problems
Ten small problems are used to compare the solutions obtained from the four-stage heuristics with the optimal solutions obtained by CPLEX (version 7.0). Each problem had 10 different jobs distributed rather randomly among the 5 different box types. The number of propeller blades for each job was chosen in the range $[2, ..., 8]$.

6.2.2 Presentation of data for the case of large-sized problems
To test the four-stage procedure on a more realistic problem setting, 64 sets of data have been randomly generated using the following parameters:

- The number of box types is set to $B = 5$.
- The total number of jobs N is in the range $[0, .., 60]$.
- The number of jobs for a box type is chosen in the interval $[0, .., 12]$.
- One third of the jobs have a request for 5 propeller blades.
- One third of the jobs have a request of propeller blades in the interval $[3, .., 9]$.
- For the three smallest box types (1, 2 and 3) one third of the jobs have a request for propeller blades in the interval $[2, .., 25]$.
- For the two largest box types (Type 4 and 5) one third of the jobs have a request for propeller blades in the interval $[2, .., 17]$.
- The due date for all jobs is in the interval $[1, .., 70]$.
- The combinations used (the box combinations given in Table 1) are captured from the foundry in consideration.

We have also a set of 31 jobs collected from real data, from the time before the new scheduling policy was implemented in the company. The performance of the four-stage heuristic for this set was in general similar to the performance for the randomly generated data. No further comments on this set are therefore given. Results for the randomly generated data are summarized in Table 13 and Table 14.

6.3 Comparison of the different procedures and policies
Due to the small solution times obtained using the four-stage heuristic we removed the test (Equation 24) that estimates the pay-off by rescheduling a late job i to an earlier point in the production plan. One would then expect that the tardiness in general will decrease since more rescheduling steps could be taken. However, due to the randomness involved, the old rescheduling environment could be better for some problem cases, since more iterations of improvement could be performed. For the small-sized problems the optimal solutions were found regardless of whether the test (Equation 24) was used in the heuristic or not. For the larger-sized problems the improvement in total tardiness by removing Equation 24 is minor, but due to a marginal increase in solution time we have run the tests for the larger-sized problem without Equation 24. If we use the test only in the first iteration of the two-stage heuristic, the total tardiness from the best rescheduling policy was marginally larger, but the difference in performance for the six rescheduling policies was somewhat smaller.

The four-stage heuristic is tested with different parameter values related to the rescheduling of tardy jobs. This is done both to test the stability of the heuristic and in order to see if we can improve the total tardiness further. By forcing the procedure to calculate the tardiness differently, new combinations and new order sequences can be obtained. Generally, a job i is tardy if $c_i > d_i$. However, for the rescheduling policies proposed, we have used the following expression $c_i + x > d_i$, where the parameter $x = 1, ..., 3$.

6.3.1 Small problem sizes
CPLEX and the four-stage heuristic were used to solve ten small randomly-sized problems. The four-stage heuristic terminated with a total tardiness equal to the total tardiness from CPLEX (version 7.0) for 9 of the 10 problems. For the last problem the total tardiness differed marginally.

Generally, all rescheduling policies except method 4 perform equally well for the small-sized test problems. The performance of the policies where independent of which procedure that was used to estimate tardiness and on whether $maxcap_b$ or $average_b$ were used to estimate the completion dates in Equation 19. Further, none of the procedures that were implemented to avoid holes in the production plan had any impact on the solutions. This may be due to the thorough search for improvements done by the other heuristic procedures proposed.

Table 12 presents the average improvements in tardiness between each step of the four-stage heuristic using rescheduling method 6 with $maxcap_b$. It is worth mentioning that the procedure *CompUpdate* only improved total tardiness for one of the problems, regardless of which rescheduling policy that were used.

In Section 3 we discussed whether the optimal choice of box combinations for the LP-relaxation, minimizing number of production days, could be used as a guide for choosing box combinations for the original problem. For the set of small test problems we see that there is a small deviation between the combinations used in the optimal production schedule and the combinations found

from the LP-solution. Based on this one could suggest using a heuristic based on the set of static LP-combinations. For each day j one could first try to find a combination from the static LP set. If no combination of this set fits, choose a combination from Table 1 according to how much tardiness it generates. However, the size of the jobs and their due dates may not interact very well with the static set of LP-combinations. Thus many combinations have to be chosen outside the static LP-set. These combinations may again not interact too well with the preferred static set of LP-combinations, which may in turn result in a rather randomly generated production plan. In the four-stage heuristic the combination chosen for each day is iteratively and dynamically chosen based on both the due dates and on the work load present in each box type. As we have seen from the set of small-sized problems this heuristic will generate the optimal set (or close to optimal set) of combinations for the original problem.

6.3.2 Large problem sizes

All rescheduling policies except method 4 perform very well also for the set of larger-sized problems. For each new plan generated in the third-stage heuristic, the rescheduling stage (stage two) was repeated approximately four times. For the fourth-stage heuristic the rescheduling stage was repeated approximately 90 times in total, counting all new replacements together. However, the performance of the policies was not independent of which procedure that were used to estimate tardiness and on whether $maxcap_b$ or $average_b$ were used when estimating c_i in Equation 19. The deviation in total tardiness obtained between the different policies was small. In order to obtain a production plan with a minimum of tardiness, we suggest to iteratively use the four-stage procedure, with rescheduling methods 1, 2, 3 and 6, using both $maxcap_b$ and $average_b$ to estimate the completion times. This will give a solution time of approximately 20 seconds on an Intel 1.3 MHz processor with 256 MB memory for a 36 job problem where the number of propeller blades per job differed between 5 and 21. If this approach is used the total tardiness is improved by 2.2% compared to using only rescheduling policy number 6 with $average_b$ (or 2.6% with $maxcap_b$).

To improve the total tardiness for a specific problem even further, we suggest to use the extended four-stage procedure. The reduction in total tardiness obtained from using the extended four-stage procedure reduced the deviation between the different rescheduling policies, but only marginally improved the total tardiness of the current best production plan. The difference in total tardiness between iteratively using the four-stage procedure with each rescheduling policy except method 4, and using the extended four stage heuristic with rescheduling method 6 was approximately 0.1%. The solution time for the extended four-stage procedure using method 6 was 12 seconds for a 36 job problem where the number of propeller blades per job differed between 5 and 21 (Intel 1.3 MHz processor with 256 MB memory).

The procedures that were proposed to fill large holes in the plan did not have any impact on the performance for the overall heuristics for the larger-sized problems either. However, if we use the expression $c_i + x > d_i$, $x = 1, 2, 3$, to estimate if a job is tardy as opposed to the general expression, $c_i > d_i$ used in Equation 19, the total tardiness differed, although marginally. For a few cases we even obtained a slightly better solution. But the best solutions were in general obtained when using the expression $c_i > d_i$.

The performance of the different heuristics included in the four-stage heuristic procedure is presented in Table 14. We see that the average percentage improvement in total tardiness decreases for each step in the four-stage procedure and that the average percentage improvement is largest between the first-stage heuristic (initial production plan) and the second-stage heuristic (resched-

uled production plan).

6.4 Numerical results compared to previously scheduling procedure used
The results obtained for the improved due-date approach used initially in the foundry, and the extended two-stage heuristic currently in use are presented in Table 13. The top row gives the sum of the total tardiness over the 64 different job sets using the specified methods while the bottom row gives the percentage reduction in total tardiness for each of the methods with respect to the improved due-date approach. As seen from the table, the extended two-stage procedure reduced the total tardiness with 36.1% compared to the improved due-date approach. Numbers that indicate the average improvements in total tardiness for each internal step in the four-stage and the extended four-stage heuristic are presented in Table 14. The numbers are here taken from the case where we use every rescheduling policy iteratively.

Note that some improvements have been made in the two-stage heuristic presented here compared to the version which is currently implemented in the foundry. Improvements have been made for the procedure that finds which combination to schedule at day j, in the case where the estimated increase in tardiness $tardinc^k$ is equal to zero for all available combinations k. Further, when rescheduling a late job i, improvements have been made in the procedure that finds which box combination to schedule at the new initial production day for job i. Solving the large-sized problems using the "new" extended two-stage heuristic with rescheduling method 6 reduced the average total tardiness by approximately 4%, compared to using the "old" procedure. This has to be taken into account when we compare Table 13 and Table 14.

Generally, optimal solution procedures are not available for the results presented in Table 13. In order to compare the four-stage heuristic with an optimal solution procedure (CPLEX), the set of smaller problem sizes was considered. However, optimal solutions were identified for some of the larger-sized problems. To identify these, the jobs with uncritical due date were removed from the original problem, then the relaxed problem were solved using CPLEX. Since the total tardiness for the relaxed problem was equal to the total tardiness for the original problem using the four-stage heuristic, we know that the solution obtained from the four-stage heuristic is optimal.

7. Alternative Objectives

Even if the main objective of the manufacturer is to minimize total tardiness, there are also other issues that should be reflected in the final production plan. In particular, the manufacturer would like to avoid extremely long tardiness for a specific order, something that would make the customer unhappy and might reduce future demand. Formally, this translates into keeping the maximum tardiness low, in addition to the main objective. Minimizing total tardiness and minimizing maximum tardiness are often conflicting objectives. However, several methods can be applied to find a balance between these two goals. In the heuristic procedure proposed one can for instance penalize every job i with a larger tardiness than a given constant C. The average maximum tardiness is 28 days for the extended two-stage heuristic and 20 days for the initial due-date approach which has the smallest maximum tardiness. By penalizing a maximum tardiness that is larger than 20 in the extended two-stage heuristic procedure, the average maximum tardiness decreased to 24 days. The total tardiness increased to 7088 days, which is still an improvement of 26.7% compared to the improved due-date approach. By minimizing the sum of the square of the tardiness, $\min \sum_{i=1}^{N} t_i^2$, instead of total tardiness the average maximum tardiness is at the same level as the initial due-date approach which is 20 days, the total tardiness is improved by approximately 25% compared to

the initial due-date approach and approximately 12% compared to the improved due-date approach.

8. The algorithm at work

The extended two-stage heuristic is now the core of the foundry's planning system. The heuristic is set up so that the user can choose to minimize total tardiness or minimize maximum tardiness.

The engineer-to-order foundry produces components exclusively for other manufacturers and the set of orders varies extensively. It is therefore difficult to get comparative before-after results. However, the manufacturer reports that the firm has moved from a situation with many delays to a situation where they feel that they are in complete control of tardiness.

Due to an efficient implementation of the heuristics presented above, a new production plan is created whenever there is a change in the order set, i.e. the company uses a "rolling horizon". To avoid impractical changes the plan for the current day is frozen, and already initiated orders will always be finished without interruption. An eBusiness implementation of the planning system is available where customers can offer order characteristics over the Web and get a guaranteed due date for the order in return.

Most new orders have an initial "request for a quote" phase, where the company offers a price and a due date for the order. To reserve capacity for these orders they are inserted in the production plan, but with a probability of acceptance, a value from 0 to 1 set by the user. The planning system will then book capacity according to this factor, by reducing order size (large orders) or use a probability function to determine if the order is to be included or not (small orders). If the offer is accepted this probability factor is set to 1.

"What if" functionality is implemented in the system, where the consequences of inserting a new order is found. Based on parameters set by the user this is expressed as a cost. For example, inserting a new order in the top of the plan will incur additional tardiness that increases costs, while working an extra shift may reduce costs. This is, of course, an additional benefit of having an automatic and fast planning algorithm. Note that work capacity, as additional shifts, are implemented simply by having a different set of combination tables.

There is an important discussion in the community today if "IT matters", see Carr [16]. The case presented here shows the importance for niche companies to develop their own proprietary solutions. It is a clear belief in this company that their planning algorithm is a major factor in their competitiveness, along with their ability to produce high quality castings. The algorithm allow them to utilize resources to a near optimum, at the same time as they have full control of tardiness and can offer guaranteed due dates.

We are currently working with a foundry producing car engines. Their steel department manufactures their castings in boxes of different sizes, and has similar capacity constraints as the foundry producing propeller blades, which should easily allow them to use the proposed solution procedure to solve parts of their planning problem.

9. Conclusion

We have outlined a mixed integer and linear programming (MILP) formulation for a scheduling

problem in an engineer-to-order foundry producing propeller blades. The foundry has a set of boxes of different sizes in which the blades are produced. The production of blades is done on successive days, mainly according to capacity restrictions with respect to which combination of boxes that can be used simultaneously. We believe that this production environment is representative for many engineer-to-order foundries. For the company in consideration, almost every capacity constraint involved in the production can be defined in the form of a combination table. The exceptions are capacity constraints that lead to job splitting and the fact that parallel processing is not allowed.

The MILP formulation outlined for the scheduling problem can be solved with a MILP solver (CPLEX, version 7.0), but only for small-sized problems. CPLEX was not able to handle the larger problems of the foundry since the number of integer variables involved in the mathematical formulation became too large.

For the scheduling problem at hand, we have proposed a four-stage heuristic procedure that performs a thorough search among different box combinations and different job sequences for a production plan that minimizes tardiness. The main part of the heuristic is a two-stage procedure and its recursive extension, the extended two-stage heuristic. The extended two-stage heuristic is the scheduling program currently implemented and in use in the company in consideration.

Generally, all of the production plans obtained from the four-stage heuristic generates a minimum of total tardiness. For small-sized problems the heuristic procedure will in general terminate with an optimal solution. For a set of ten small test problems, nine terminated with the optimal solution. The four-stage heuristic generated a solution that differed marginally from the optimal solution for the last problem. For larger-sized problems the four-stage heuristic improves the total tardiness significantly compared to the improved due-date approach previously used by the company. It also improves the total tardiness in comparison to the extended two-stage heuristic currently used by the company. Given problem cases similar to the company in consideration (the job sizes are not minor in comparison to the maximum number of boxes in a combination), the results obtained indicates that it is more important to concentrate on rescheduling jobs in order to minimize tardiness than on rescheduling jobs in order to fill the idle times which may appear in the plan.

Numerical results for the four-stage heuristic indicate that the proposed solution procedure has a sensible construction, which regardless of scheduling policy and parameter settings terminates with good solutions. The solution procedure could also with minor modification be applied to other engineer-to-order foundries.

Acknowledgment

We would like to express our thanks to Sverre Storøy, Jan Ubøe, Trond Steihaug and the anonymous reviewer for helpful comments and discussions.

References

1. Baker, K.R., 1974, Introduction to sequencing and scheduling, Wiley, New York.

2. French, S., 1982, Sequencing and scheduling: An introduction to the mathematics of the jobshop, Ellis Horwood Limited, England.

3. Gargeya, V.B. and Deane, R.H., 1996, Scheduling research in multiple resource constrained job shops: A review and critique, International Journal of Production Research, 34, 8, 2077-2097.

4. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., and Shmoys D.B., 1993, Sequencing and scheduling: Algorithms and complexity, in Logistics of production and inventory, Graves, S.C., Rinnooy Kan, A.H.G. and Zipkin, P.H. (eds.), Elsevier Science Publishers, The Netherlands.

5. Du, J. and Leung J.Y.T, 1990, Minimizing total tardiness on one machine is NP-hard, Mathematics of Operation Research, 15, 3, 483-495.

6. Koulamas, C., 1994, The total tardiness problem - Review and extensions, Operations Research, 42, 6, 1025-1041.

7. Potts, C.P. and Van Wassenhove, L.N., 1982, A decomposition algorithm for the single machine total tardiness problem, Operation Research Letters, 1, 177-181.

8. Garey, M.R., Johnson, D.S. and Sethi, R., 1976, The complexity of flowshop and jobshop scheduling, Mathematics of Operation Research, 1, 117-129.

9. Armentano, V.A. and Scrich, C.R., 2000, Tabu search for minimizing total tardiness in a job shop, International Journal of Production Economics, 63, 2, 131-140.

10. Wang, T.Y., and Wu, K.B, 2000, A revised simulated annealing algorithm for obtaining the minimum total tardiness in job shop scheduling problems, *International Journal of Systems Science*, 31, 4, 537-542.

11. Wang, T.Y., and Wu, K.B, 1999, A parameter set design procedure for the simulated annealing algorithm under the computational time constraint, *Computers and Operations Research*, 26, 7, 665-678.

12. Fredendall, L.D., Melnyk, S.A. and Ragatz, G., 1996, Information and scheduling in a dual resource constrained job shop, International Journal of Production Research, 34, 10, 2783-2802.

13. Kantorovich, L.V., 1960, Mathematical Methods of organizing and planning production, Management Science, 6, 366-422.

14. Haessler, R.W. and Sweeney P.E., 1991, Cutting stock problems and solution procedures, European Journal of Operational Research, 54, 141-150.

15. Mieghem, J. V. and Rudi, N., 2002, Newsvendor networks: dynamic inventory management and capacity investment with discretionary pooling, Manufacturing and Service Operations Management, 4, 4, 313-335.

16. Nahmias, S., 1997, Production and operations analysis, Mc Graw Hill, Signapore.

17. Carr, N. G., 2003, IT Doesn't matter, Harward Business Review, May, 41-49.

Tables

Box type	Combinations									
	1	2	3	4	5	6	7	8	9	10
70	7	5	1	2	2	2	0	0	1	0
100	0	0	3	2	0	0	2	1	0	1
115	0	0	2	0	2	2	2	2	0	0
170	0	2	0	1	1	0	1	0	1	1
215	0	0	0	0	0	1	0	1	1	1

Table 1. Allowed combinations.

	Type 70			Type100		Type115			Type 170	Type 215
Job number	1	2	3	4	5	6	7	8	9	10
Number of blades	7	5	5	5	5	3	3	4	8	4
Due date	4	5	13	7	9	7	8	9	5	6

Table 2. Jobs and due date.

	Type 70			Type100		Type115			Type 170	Type 215
Job number	1	2	3	4	5	6	7	8	9	10
Number of blades	7	5	5	5	5	3	3	4	8	4
Due date	4	5	13	7	9	7	8	9	5	6
Completion date	2	3	11	8	9	7	8	10	5	7
Tardiness	0	0	0	1	0	0	0	1	0	1

Table 3. Optimal schedule found through CPLEX and heuristics.

	Dates											
Boxes of type	1	2	3	4	5	6	7	8	9	10	11	12
70	7	5	5	0	0	0	0	1	1	1	1	1
100	0	0	0	1	1	1	1	3	3	3	3	3
115	0	0	0	0	0	0	0	2	2	2	2	2
170	0	2	2	1	1	1	1	0	0	0	0	0
215	0	0	0	1	1	1	1	0	0	0	0	0

Table 4. Jobs scheduled according to initial due-date sequence.

	Type 70			Type100		Type115			Type 170	Type 215
Job number	1	2	3	4	5	6	7	8	9	10
Number of blades	7	5	5	5	5	3	3	4	8	4
Due date	4	5	13	7	9	7	8	9	5	6
Completion date	1	2	3	8	8	9	10	12	7	7
Tardiness	0	0	0	1	0	2	2	3	2	1

Table 5. Jobs scheduled according to initial due-date sequence.

	Combination 1:	Combination 2:	Combination 3:	Combination 4:
$tardinc^k_{70}$	$0*(1-\frac{7}{7})=0$	$0*(1-\frac{5}{7})=0$	$0*(1-\frac{1}{7})=0$	$0*(1-\frac{2}{7})=0$
$tardinc^k_{100}$	$0*(1-\frac{3}{3})=0$	$0*(1-\frac{0}{3})=0$	$0*(1-\frac{3}{3})=0$	$0*(1-\frac{0}{3})=0$
$tardinc^k_{115}$	$0*(1-\frac{0}{2})=0$	$0*(1-\frac{0}{2})=0$	$0*(1-\frac{2}{2})=0$	$0*(1-\frac{2}{2})=0$
$tardinc^k_{170}$	$1*(1-\frac{0}{2})=1$	$1*(1-\frac{2}{2})=0$	$1*(1-\frac{0}{2})=1$	$1*(1-\frac{1}{2})=\frac{1}{2}$
$tardinc^k_{215}$	$1*(1-\frac{0}{1})=1$	$1*(1-\frac{0}{1})=1$	$1*(1-\frac{0}{1})=1$	$1*(1-\frac{0}{1})=1$
$\sum_{b=1}^{B}\left[\frac{hole_b^{j-1}}{maxcap_b}\right]^{++}$	0	0	$\frac{1}{3}+\frac{1}{2}$	$\frac{1}{2}$
Last order	0	0	0	0
$tardinc^k$	2	1	1,17	1

Table 6. Part I. Calculation of $tardinc^k$ for day 3. Priority=(1,0,0,0,0). Hole=$(0,1,1,0,0)^T$. Maxcap=(7,3,2,2,1).

	Combination 5	Combination 6	Combination 9
$tardinc^k_{70}$	$0*(1-\frac{2}{7})=0$	$0*(1-\frac{2}{7})=0$	$0*(1-\frac{1}{7})=0$
$tardinc^k_{100}$	$0*(1-\frac{2}{3})=0$	$0*(1-\frac{0}{3})=0$	$0*(1-\frac{0}{3})=0$
$tardinc^k_{115}$	$0*(1-\frac{0}{2})=0$	$0*(1-\frac{2}{2})=0$	$0*(1-\frac{0}{2})=0$
$tardinc^k_{170}$	$1*(1-\frac{1}{2})=\frac{1}{2}$	$1*(1-\frac{0}{2})=1$	$1*(1-\frac{1}{2})=\frac{1}{2}$
$tardinc^k_{215}$	$1*(1-\frac{0}{1})=1$	$1*(1-\frac{1}{1})=0$	$1*(1-\frac{1}{1})=0$
$\sum_{b=1}^{B}\left[\frac{hole_b^{j-1}}{maxcap_b}\right]^{++}$	$\frac{1}{3}$	$\frac{1}{2}$	0
Last order	0	0	0
$tardinc^k$	1,17	0,5	0,5

Table 6. Part II. Calculation of $tardinc^k$ for day 3. Priority=(1,0,0,0,0). Hole=$(0,1,1,0,0)^T$. Maxcap=(7,3,2,2,1).

Tardiness=11		Production days, $j=$										
Box type	Blades	1	2	3	4	5	6	7	8	9	10	11
70	δ_{1j}	1	1	2	1	1	1					
70	δ_{2j}							5				
70	δ_{3j}								2	2	1	
100	δ_{4j}	3	3									
100	δ_{5j}										2	3
115	δ_{6j}	2	1									
115	δ_{7j}		1	2								
115	δ_{8j}								2	2		
170	δ_{9j}				1	1	1	2	1	1	1	
215	δ_{10j}			1	1	1	1					

Table 7. The production plan obtained using the first-stage heuristic.

Tardiness=7		Production days, $j=$											
Box type	Blades	1	2	3	4	5	6	7	8	9	10	11	12
70	δ_{1j}	2	2	3									
70	δ_{2j}			2	3								
70	δ_{3j}											1	4
100	δ_{4j}					2	2	1					
100	δ_{5j}								1	1	1	2	
115	δ_{6j}	2	1										
115	δ_{7j}					2	1						
115	δ_{8j}						1	2	1				
170	δ_{9j}	1	1	2	2	1	1						
215	δ_{10j}							1	1	1	1		

Table 8. The production plan obtained using the second-stage heuristic.

Tardiness=5		Production days, $j=$										
Box type	Blades	1	2	3	4	5	6	7	8	9	10	11
70	δ_{1j}	5	2									
70	δ_{2j}		3	2								
70	δ_{3j}										1	4
100	δ_{4j}				2	2	1					
100	δ_{5j}							1	1	1	2	
115	δ_{6j}			2	1							
115	δ_{7j}				1	2						
115	δ_{8j}							2	2			
170	δ_{9j}	2	2	1	1	1	1					
215	δ_{10j}						1	1	1	1		

Table 9. The production plan obtained using the extended two-stage (third-stage) heuristic.

Tardiness=4		Insert larger combination, Production days, $j=$										
Box type	Blades	1	2	3	4	5	6	7	8	9	10	11
70	δ_{1j}	5	2									
70	δ_{2j}		3	2								
70	δ_{3j}										1	4
100	δ_{4j}				2	2	1					
100	δ_{5j}							1	1	1	2	
115	δ_{6j}				2	1						
115	δ_{7j}					1	2					
115	δ_{8j}							2	2			
170	δ_{9j}	2	2	2	1	1						
215	δ_{10j}						1	1	1	1		

Table 10. Intermediate production plan obtained using the fourth-stage heuristic.

Tardiness=3		Production days, j =										
Box type	Blades	1	2	3	4	5	6	7	8	9	10	11
70	δ_{1j}	5	2									
70	δ_{2j}		3	2								
70	δ_{3j}								1	1	2	1
100	δ_{4j}	0	0	0	1	1	1	1	1			
100	δ_{5j}								2	3		
115	δ_{6j}	0	0	0	0	0	2	1				
115	δ_{7j}							1	2			
115	δ_{8j}									2	2	
170	δ_{9j}	2	2	2	1	1						
215	δ_{10j}				1	1	1	1				

Table 11. The production plan obtained using the fourth-stage heuristic.

Two-stage heuristic		Extended two-stage heuristic	Four-stage heuristic	
First-stage	Second-stage	Third-stage	Fourth-stage	Extended fourth-stage
0%	35.0%	28.8%	5.4%	0%
0%	35.0%	53.8%	56.3%	60.0%

Table 12. Small-sized problems: Percentage improvements in total tardiness for each step in the proposed four-stage heuristic.

Improved	Two-stage heuristic		Extended two-stage heuristic
due date	First-stage	Second-stage	Third-stage
9669	8424	7131	6183
0%	12.9%	26.2%	36.1%
	0%	15%	26%

Table 13. Large-sized problems: Percentage improvements in total tardiness for the current heuristic implemented in the foundry.

Two-stage heuristic		Extended two-stage heuristic	Four-stage heuristic	
First-stage	Second-stage	Third-stage	Fourth-stage	Extended fourth-stage
0%	22.7%	5.0%	3.1%	1.3%
0%	22.7%	26.6%	28.8%	29.7%

Table 14. Large-sized problems: Percentage improvements in total tardiness for each step in the proposed four-stage heuristic.