Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 1, No. 3, pp. 545−564, September 2005

SCHEDULING PERSONNEL FOR PRESS MACHINES IN THE AUTOMOTIVE INDUSTRY

Erwin Pesch and Ulrich A.W. Tetzlaff

Dedicated to Professor Toshihide Ibaraki in honor of his 65^{th} *birthday.*

Abstract: For the case of a personnel planning situation for a very large car manufacturer, this paper investigates the interaction between management decisions and scheduling. In particular, we consider the manufacturing of car bodies or pieces of them within a press machine shop and propose an efficient procedure to overcome deficiencies in manpower planning and in-time production. The presented model allows to minimize the maximum number of workers required per time unit. We propose an optimal as well as a heuristic algorithm based upon a branch-and-bound procedure.

Key words: *personnel planning, branch-and-bound, resource levelling*

Mathematics Subject Classification: *90C57*

1 Introduction

Manpower planning and scheduling belong to a company's most cost sensitive areas and management can highly benefit from well made scheduling decisions on the operational level. Within the car industry in particular, the ability to reduce costs and thus to follow a low pricing strategy is of utmost importance in order to compete in today's world wide markets.

High labor productivity is usually achieved with the help of creating flexible labor schedules, reduction of personnel as well as better use of the available capacity. However, regulations and union agreements restrict the limit by which the first two aspects can be considered. Furthermore, while machine capacity adaptations can be accomplished with relative ease, demand fluctuations can usually only be accommodated to a small extend by changing the capacity of personnel. Therefore, personnel planning and scheduling procedures in the automotive industry require the development of sophisticated planning procedures in order to allow for high labor productivity while considering the inflexibility of labor capacity in the short run.

The typical personnel planning process often starts with the assessment of work force requirements using some kind of rough cut approach on the tactical level. A common procedure in order to estimate the work force requirements per day is to take the car volume produced per month, multiply it by the required production time per car and divide it by the number of working days per month as well as the work time per day and worker. To accommodate different models with different features, a weighted average value is taken for the production volume as well as for the production time per car. Additionally, corrections are made to allow for the outsourcing of some production, the production of additional service parts, lost production due to scrap, rework and machine downtime as well as startup times and productivity improvements due to learning. The result is further adjusted to include absenteeism due to vacation or illness. Once the estimate for the work force requirements is obtained, more detailed planning is performed on the operational level by scheduling and assigning the work force on a daily basis to different machines.

Inflexibility in work force adjustments in the short run necessitates careful planning and makes it desirable to coordinate labor supply and demand as accurate as possible. The above described rough cut approach, however, entails some shortcomings. In particular, the model mix calculations using weight factors to calculate some average production volume and time is quite inaccurate since it is based on past data. Furthermore, dynamic aspects like the actual fluctuations in labor demand caused by the different labor requirements of different production lots are neglected. Therefore, in what follows we present an integrated approach which simultaneously performs labor requirements planning and personnel scheduling for a set of parallel machines (see also related work in Błażewicz et al. [4]). The objective is to minimize the required workforce per time period over a given planning horizon T. For the resulting binary integer problem we propose an optimal as well as a heuristic algorithm based on a branch-and-bound type procedure.

This paper is structured as follows. The next section provides a detailed analysis of the given situation at the car manufacturer. Our proposed model formulation in order to overcome given planning deficiencies is described in section 3. Section 4 provides a detailed outline of the solution procedures suggested to solve our model. It is followed by a description of the data set used to benchmark and test our model and solution procedure. The obtained test results are provided in section 6. We conclude with some final remarks in section 7.

2 Problem Description

This section provides a detailed analysis of the planning procedure at the car manufacturer. The given hierarchical approach starts on the tactical level with a personnel requirements planning step. It will be outlined in detail in the next subsection. It is followed by another subsection which demonstrates how these results from the tactical level are then used on the operational level to perform personnel scheduling decisions.

2.1 Personnel Requirements Planning

At the time we analyzed the car plant's situation there were 1035 people employed in the press machine shop. Seventy-nine people did some administrative work, while 545 operated the press machines and 411 people were needed for manufacturing support functions like material handling, setup operations, etc. The factory produced 839 different items needed for 9 different car plants all over the world. These close ties to other plant locations documents the relative importance of the continuously ongoing production in the considered press machine shop. The shop itself is divided into 3 areas. We focus in our study on the biggest and most important one which contains 32 press machines that can be partitioned into 7 different groups. In this paper we will further limit our scope on the planning and scheduling process of only the most important group of 7 large and expensive transfer presses. However, all planning steps can be applied to the remaining groups as well, requiring only slight modifications in the input data. Of the machines under consideration (for convenience we call them $P1, P2, \ldots, P7$), a single large transfer press can operate more than 600 tons of rolled metal sheet per day.

A worker at the car plant is supposed to work 35 hours a week. Considering a lunch break, the paid number of hours per day is 7.75. In order to avoid having a 4.5-day week, workers either work 4 days (31 hours) a week or 5 days (38.75 hours) a week. Every worker has an account to keep track of his or her work hours and to ensure a credit-debit equilibrium by the end of a year. An important goal of this "corridor model" of flexible labor schedules is to buffer differences in the order volumes implied by model changes or any kind of seasonal effects. In order to calculate the number of people necessary to perform a particular task, the number of paid work hours will be reduced by the number of breaks. Thus, the work time per day of any of the workers results to 7.11 hours. Using the above given equation of work force requirements an additional 15% is added to the obtained value due to illness and vacation. The obtained calculated number of people is increased by an additional 22% of which are 17% due to production failures and machine breakdowns larger than 30 minutes and the remaining 5% due to slight production modifications that lead to a temporary performance decrease. Additional personnel is needed for a number of minor important tasks. We are not going to describe the calculations of the processing times since they heavily depend on the car model produced, vary between machines and include times for cleaning, information transfer etc.

2.2 The Scheduling Problem

A job produced on a press machine is a large collection of items of the same type. Thus, a job is a batch, the size of which satisfies the demand of 3 to 5 days. Additionally, some exceptional items are produced in very small amounts. The number of workers that service a press machine depends on the machine as well as on the item's weight, size and quality requirements. Moreover, some of the machines allow a setup while processing. A comparison of the calculated personnel requirements and the real number of people needed for production revealed an increasing gap over the last 20 months. For instance, in June (a month without any unusual effects) the number of people needed exceeded the calculated number by 32 (with respect to the whole press machine shop and not restricted to the large transfer press machines $P1, \ldots, P7$) altogether accumulating 5200 overtime hours. The overtime was not compensated by taking time off and thus was not balanced in the corridor model: within 20 months the work time balance (credit minus debit) increased from -2721 to 14188 hours. As seasonal effects can be excluded, the question arises whether the workers are fully occupied. Surprisingly, the main reason for the increasing number of overtime hours was an insufficient number of workers at some of the machines where the available workers were unable to perform job assignments that require more personnel. Consequently, they were waiting for additional personnel to be set free when jobs are finished at other machines. Thus, it is desirable to have a schedule that balances the number of workers over the given planning horizon. This is even more important as all workers perform an 8 hour shift irrespectively whether they are needed for only a short time or the complete shift. In order to achieve a resource balance we therefore suggest to minimize the maximum number of workers required to produce all jobs in time. In the remainder of this paper we will present such an approach restricting our attention without much loss of generality on the 7 transfer press machines $P1, \ldots, P7$.

3 Model Formulation

This section presents the model formulation to address our personnel planning problem for a set of parallel working press machines. Next, however, we provide the notation used as well as a list of assumptions made by our model.

Notation .

indices:

t : index for time periods, $t = 1, \ldots, T$;

j : index for machines, $j = 1, \ldots, M$;

i : index for the jobs, $i = 1, \ldots, N_j$;

parameters:

d_{ij} : required number of workers per period for job i on machine j;

D : available number of workers per period;

M : number of machines

N_j : total number of jobs for machine j to be processed during the given planning horizon;

p_{ij} : processing time given by the number of periods required for job i on machine j;

T : planning horizon;

variables:

$$x_{ijt} = \begin{cases} 1 & \text{if the } i\text{-th job is started at the beginning of period } t \text{ on machine } j \\ 0 & \text{otherwise} \end{cases}$$

Our formulation is based on the following assumptions:

1. The number of time periods for the given planning horizon is known.
2. There is a given set of jobs to be processed and for each job it is known on which machine it has to be processed. Thus, we could simplify our model if the machine index j is dropped.
3. Processing times are known.
4. The required number of personnel for each job is known.
5. The number of available machines is known and constant.
6. Once a job is started at a machine it has to be finished without interruption, i.e., service is non-preemptive.
7. Each machine can only perform one job at a time.
8. The machines are available for the complete planning horizon.
9. Jobs can only start at the beginning of a time period.
10. Workers can do any kind of task on any machine.

The objective function of our formulation minimizes the number of workers by minimizing the maximal requirements over all time periods within the given planning horizon. This is achieved by minimizing the maximum value over t subject to several constraints. Hence, the number of required workers for all jobs i in process at period t (indicated by an assignment of value one to the decision variables $x_{ijt'}, t - p_{ij} + 1 \leq t' \leq t$) is calculated. A survey on closely related resource constraint scheduling problems is provided in Brucker et al. [5].

The first set of constraints (2) requires that all jobs are finished within the given time horizon. Thus the time period when the job starts plus the required processing time has to be less than or equal to the number of time periods T available, i.e. the job's ending period given by $p_{ij} - 1 + t$ is less than or at most T if job i starts at the beginning of period t. Because the time period when we start the job is subject to our decision variable x_{ijt} we sum over all periods t and multiply by the ending times. It ensures that the appropriate ending time is selected since each job is only processed once and x is a binary decision variable. The following set of constraints (3) requires that at most one job is processed at machine j at each time period t. This is incorporated by summing up all decision variables x over all jobs N_j at machine j and set this sum less than or equal to one. Since a job which is processed at time period t might have started prior to t, to be precise, sometime between $t - p_{ij} + 1$ and t, we also have to consider the sum of the decision variables over these time periods. Additionally, we have to require that all jobs to be processed at machine j are finished within the given planning horizon. Thus, in (4) the sum of x_{ijt} over all time periods T and jobs N_j for machine j has to be equal to N_j. Next in (5), we have to ensure that the number of available workers at each planning period is not exceeded. Similar as within the objective function, we take our decision variables x_{ijt} and multiply them with the number of workers d_{ij} required for processing the job i at machine j. By summing over all machines, jobs at the machines, and number of processing periods required for each job, the worker requirements are obtained for each t. The result is then set less than or equal to D, the number of workers available at each period. Of course, constraints (5) may be dropped since either the objective function ensures its satisfaction or, if (5) cannot be satisfied the acceptance of the solution is left to the user. Finally we define our decision variable x_{ijt} as binary variables.

$$\min \max_{1 \leq t \leq T} \sum_{j=1}^{M} \sum_{i=1}^{N_j} \sum_{k=0}^{p_{ij}-1} x_{ij(t-k)} \, d_{ij} \tag{1}$$

s.t.

$$\sum_{t=1}^{T} x_{ijt}(p_{ij} - 1 + t) \leq T \quad \forall i, \forall j \tag{2}$$

$$\sum_{i=1}^{N_j} \sum_{k=0}^{p_{ij}-1} x_{ij(t-k)} \leq 1 \quad \forall j, \forall t \tag{3}$$

$$\sum_{t=1}^{T} \sum_{i=1}^{N_j} x_{ijt} = N_j \quad \forall j \tag{4}$$

$$\sum_{j=1}^{M} \sum_{i=1}^{N_j} \sum_{k=0}^{p_{ij}-1} x_{ij(t-k)}\, d_{ij} \leq D \quad \forall t \tag{5}$$

$$x_{ijt} \in \{0,1\} \quad \forall i, \forall j, \forall t \tag{6}$$

Similar models on resource balancing and minimization of the peak of resource usage are described in [6]. Our problem is closely related to task scheduling of multiprogramming computer systems where each task is characterized by its processing time and memory requirements. For details we refer to Krause et al. [9].

The problem is NP-hard even for the case of dedicated machines when index j can be dropped in the model. It can be reduced to the one-dimensional bin packing problem, cf. Alvim et al. [1] and Coffman et al. [7]. Consider the one-dimensional bin packing problem that consists of finding the minimum number of bins of capacity D necessary to pack a set of items i with weights d_{ij} without violating the capacity constraints. There is a min-max "dual" formulation (see [8]) in which the bin capacity D of a fixed number T of identical bins is to be minimized. The dual bin packing problem is closely related to scheduling a set of jobs on parallel machines subject to minimizing the makespan, cf. [4].

4 Solution Procedure

A solution for the model formulation provided above can be found with a branch-and-bound type approach. Below we describe in detail such a procedure. We begin with a heuristic in order to generate a feasible solution using it as an upper bound. It is followed by a description of several lower bounds, the branching strategy and the fathoming criteria. We conclude with a pseudo-code description of the complete procedure.

4.1 Upper Bound

In order to obtain an upper bound (UB), a feasible solution is generated by a heuristic procedure. This procedure starts by generating sequences for each machine j by sequencing the machine's N_j jobs in non-increasing order based upon worker requirements d_{ij}. Jobs with the same worker requirements are ordered arbitrarily. These job sequences are then sorted according to the value of their first members such that another sequence, a machine sequence, is obtained. The first member of the machine sequence is the machine j with the highest first d_{ij} of its job sequence, the second machine is the sequence with the second highest first d_{ij} of its job sequence and so forth until all machines are considered and sorted. In case of a tie, the machine order is based on the comparison of the first period t in which a pair of jobs i and i' has different worker requirements $d_{ij} \neq d_{i'j'}$. Machine j precedes machine j' if $d_{ij} > d_{i'j'}$. In other words, we assign to every machine a T-tuple. The tuple entries are the worker requirements of the job sequence of this machine. Hence, assuming that the jobs dedicated to machine j are processed in non-increasing order of the worker requirements and without any idle time on j, the tuple entry at position t, $1 \leq t \leq T$, is d_{ij} if job i is processed in period t on machine j. The entry is 0 if all jobs have been completed on machine j. A lexicographic order of these tuples (starting with the largest one) delivers the desired machine sequence. Identical sequences are considered in arbitrary order.

Finally job assignments are performed based on this machine sequence. It is achieved by first taking the jobs of the first member of the machine sequence and assign its jobs according to the order of its job sequence. Next the second machine of the machine sequence is taken and its jobs are assigned according to the reversed order of its job sequences. Thus we continue to assign jobs to the machines by considering the machines in the order of the machine sequence and then assigning jobs in the alternating order of their job sequences, i.e., first starting from the beginning of the job sequence for the first machine, next starting from the end of the job sequence for the second machine, and after that again starting from the beginning of the job sequence for the third machine and so on. Thus the basic idea of this procedure is to separate the jobs with the highest worker requirements as much as possible. In our application this procedure always provides an easily computable upper bound. However, if the available number of workers D is small, the procedure's outcome might violate constraint set (5) and the obtained "upper bound" will be dominated by D.

Example : Consider the data of Table I. This procedure generates the job sequences $3, 2, 1, 4$ for the first machine, $1, 3, 2$ and $1, 2, 3$ for machines 2 and 3, respectively. The three $12-$tuples are (443333222222), (444433322220) and (444422222200) for the machines $1, 2$ and 3, respectively. The obtained machine sequence is $2, 3, 1$. Thus the three jobs on machine 2 are scheduled in the sequence $1, 3, 2$. The jobs on machine 3 are scheduled in the sequence $3, 2, 1$ and the job order on machine 1 is $3, 2, 1, 4$. As a machine has no idle time until all its jobs have been completed, the number of workers needed over all 12 periods is $10, 10, 9, 9, 8, 8, 9, 8, 8, 8, 4, 2$. The initial upper bound is therefore equal to 10.

Table I: Personnel requirements and processing times.

machine 1			machine 2			machine 3		
job	p_{i1}	d_{i1}	job	p_{i2}	d_{i2}	job	p_{i3}	d_{i3}
1	3	2	1	4	4	1	4	4
2	4	3	2	4	2	2	3	2
3	2	4	3	3	3	3	3	2
4	3	2						

Next it can be proved that the UB is an optimal solution for the two machine case. A proof is a consequence of a reformulation of this problem as a bottleneck assignment problem satisfying the Monge property. For details see the paper by Bein et al. [3]. However, we will provide a straightforward proof with some interchange arguments.

Theorem: The UB is an optimal solution for the two machine case.

Proof. Assume for the moment that all jobs are unit-time jobs and consider an optimal solution of (1) to (6) for the case of two machines. Moreover, because of (2) assume that the number of jobs equals T for both machines. Otherwise, if the number of jobs on a machine is smaller than T, say k, we can add $T-k$ additional unit-time jobs requiring no resources, i.e. no workers. For simplicity use the same numbering for jobs and periods. Suppose the job orders of the solution do not satisfy the condition that all jobs on the first machine are in non-increasing resource demand order d_{i1}, $i = 1, \ldots, T$, and all the jobs on the second machine are in non-decreasing order d_{i2}, $i = 1, \ldots, T$. Let i_1 and i_2 be the first job on the first and the second machine, respectively, satisfying

$$d_{11} \geq d_{21} \geq \cdots \geq d_{(i_1-1)1} < d_{i_1 1}$$

and

$$d_{12} \leq d_{22} \leq \cdots \leq d_{(i_2-1)2} > d_{i_2 2}$$

Assume that among all optimal solutions we choose that one with a job sequence which has a non-increasing resource demand of maximum length, i.e. i_1 is maximum among all optimal solutions. Among the optimal solutions with that property we select one where i_2 is maximum.

Case 1: $i_1 < i_2$. The worker requirements in periods $i_1 - 1$ and i_1 are $d_{(i_1-1)1} + d_{(i_1-1)2}$ and $d_{i_1 1} + d_{i_1 2}$, respectively. Exchange jobs i_1 and $i_1 - 1$ on the first machine. Now, the worker requirements in periods $i_1 - 1$ and i_1 are $d_{i_1 1} + d_{(i_1-1)2} \leq d_{i_1 1} + d_{i_1 2}$ and $d_{(i_1-1)1} + d_{i_1 2} \leq d_{i_1 1} + d_{i_1 2}$, respectively. Hence the exchange did not deteriorate the objective function and the obtained solution is still optimal. Repeat this exchange argument until job i_1 is positioned on machine one in non-increasing order of the resource demand. This is a contradiction that i_1 was maximum, because we obtained a solution where at least i_1 jobs are ordered in non-increasing resource demand.

Case 2: $i_1 > i_2$. This can be achieved applying a similar argument to the second machine and i_2. Exchange jobs i_2 and $i_2 - 1$ on the second machine. Now, the worker requirements in periods $i_2 - 1$ and i_2 are $d_{i_2 2} + d_{(i_2-1)1} \leq d_{(i_2-1)2} + d_{(i_2-1)1}$ and $d_{(i_2-1)2} + d_{i_2 1} \leq d_{(i_2-1)2} + d_{(i_2-1)1}$, respectively. Hence, the obtained solution is still optimal. Repeat this exchange argument until job i_2 is positioned on machine two in non-decreasing order of the resource demand. This is a contradiction that i_2 was maximum.

Case 3: $i_1 = i_2$. Exchange jobs i_1 and $i_1 - 1$ on both machines. Obviously the maximum resource demand did not change. For the obtained solution one of the three cases applies until job i_1 is positioned on machine one in non-increasing order of resource demand and job i_2 is positioned on machine two in non-decreasing resource demand order.

Hence, there is an optimal solution where all jobs on machine 1 are ordered with non-increasing resource demands while those of the second machine are ordered with non-decreasing resource demands. A job i with processing time $p_{i1} > 1$ (or $p_{i2} > 1$) will be split into p_{i1} (or p_{i2}) unit-time jobs, all of which the upper bound procedure positioned next to each other. As the resource demand is the same for the whole job duration, the procedure generates an optimal solution. □

4.2 Lower Bound

Lower bounds are generated within two steps. The first step consists of generating a candidate for the lower bound. This is achieved by taking the maximum of two possible bounds. A first bound consists of the average worker requirements rounded to its next higher integer value:

$$LB1 = \left\lceil \frac{\sum_{j=1}^{M} \sum_{i=1}^{N_j} p_{ij} d_{ij}}{T} \right\rceil \tag{7}$$

A second bound is given by taking the largest worker requirement d_{ij}, i.e.

$$LB2 = \max_{i,j} \{d_{ij}\} \tag{8}$$

Thus the candidate for the lower bound from the first step consists of

$$LB = \max\{LB1, LB2\} \tag{9}$$

Within a second step we eventually improve this candidate by taking a relaxation of the original problem formulation in order to test if the lower bound candidate allows to obtain a feasible solution for this relaxed formulation. If this is the case, the candidate is our lower bound. Otherwise, the value of the candidate is increased incrementally by one until a feasible solution for the relaxation is obtained. Hereby, the relaxation consists of the following simplifications:

1. Jobs may be interrupted, i.e., preemption is allowed.
2. Jobs may be processed in a parallel fashion on several machines, even on machines on which it is from a technical point of view not possible.
3. The number of machines is not restricted, thus the number of jobs processed in a parallel fashion might surpass the value of M.

The unlimited number of machines assures that only the resource capacity, i.e. the number of workers, but not the unavailability of a machine is a limiting factor and a reason to schedule a job in the next period. A job requiring a small number of workers will never be shifted to a later time period as long as the minimum number of still available workers (not yet occupied workers needed for processing of other jobs at the same time) during the job's processing periods is not exceeded, even if the number of machines is not sufficient.

To obtain a solution for our relaxed formulation we proceed as follows. First, each job i on machine j is split into p_{ij} unit-time jobs each of which requires d_{ij} workers during its execution. Then all jobs, independent on which machine they have to be processed, are sorted according to their worker requirements d_{ij} in descending order. From this list we take the first member and assign it to a machine for one time period. Next we test if we still have further worker capacity available, by subtracting the d_{ij} value of our assigned job from our LB value obtained during the previous step. If we still have further worker capacity available ($LB > d_{ij}$ of the assigned job) we try to assign the next unit-time job in sequence to one of the other machines. We take the next one from the list and assign it to one of the other machines, again under the condition that enough workers are available. Otherwise, if the worker requirement d_{ij} of all non-assigned jobs exceeds LB, we assign the first job in the sequence to be processed in the next time period. We continue in this fashion always assigning jobs to machines taken from our list in sequential order and trying to fit them in as early as possible to a single time period, given that enough workers are still available in that period. Once we have assigned all jobs, we calculate the obtained makespan C_{max} of our result. If C_{max} is smaller or equal to T we are finished, and we maintain our LB from the first step. If, however, C_{max} is larger than T, we increase our LB value by one and thus our available worker capacity per time period and repeat the above procedure. We continue in this way until we have $C_{max} \leq T$ and call the obtained new lower bound of this second step "artificial lower bound" ALB. Note that the preemptive version of the problem in the second step of the lower bound calculation is still NP-hard because it is identical with the one-dimensional bin packing problem which has been shown to be NP-hard, cf. [7].

Examples : The first example consists of a set of three parallel machines ($M = 3$) and a planning horizon of $T = 14$ periods. There are a total of ten jobs to be assigned. Each job has a duration of p_{ij} time units with given personnel requirements of d_{ij}. The precise data is given in Table I.

A lower bound calculation yields $LB = LB1 = \lceil 6.6 \rceil = 7$. However, there is no feasible solution for our model (1) to (6) with $D = 7$. The second step of the lower bound calculations provides a bound of 8 which is optimal. An optimal solution e.g. is

$$x_{113} = x_{416} = x_{219} = x_{31(13)} = x_{224} = x_{328} = x_{12(11)} = x_{131} = x_{235} = x_{338} = 1$$

and $x_{ijt} = 0$ for any other combination of indices i, j and t. The number of workers required in all 14 periods is described through the vector $(4, 4, 6, 8, 6, 6, 6, 7, 8, 8, 7, 7, 8, 8)$.

The reader may have noticed that the lower bound obtained through the second step of our lower bound calculations need not to be a true lower bound as the following example of a schedule for 6 unit-time jobs on 3 different machines shows: Assume $T = 2$ and the worker requirements are $d_{11} = 2, d_{21} = 4, d_{12} = 2, d_{22} = 3, d_{13} = 2$ and $d_{23} = 3$. We obtain $LB = LB1 = 8$, a value that will be increased to 9 in the second step of the lower bound calculation. However, there exists an optimal solution with just 8 workers: $x_{112} = x_{232} = x_{222} = x_{131} = x_{121} = x_{211} = 1$ and $x_{ijt} = 0$ for all other combinations of i, j, and t. Hence, ”the artificial lower bound” (ALB) of the second step may cause branch truncations of optimal solutions in the branch-and-bound tree.

4.3 Branching Strategy

In order to solve our formulation by a branch-and-bound procedure, a depth-first strategy is applied. Moreover, every search tree node corresponds to a current scheduling period and a subschedule (which is empty at the beginning of the branch-and-bound. Consider a particular search tree node and the associated current scheduling period τ which is the minimum among all earliest possible starting times of all unscheduled jobs. As τ is defined by the set of non-scheduled jobs it is obviously independent of the machines and it is greater or equal to the earliest time that a machine becomes idle again. We use a branching rule based upon the calculation of a priority value which for a particular job is based upon the worker requirements multiplied with the processing time:

$$PV_{ij} = d_{ij} p_{ij} \tag{10}$$

Thus, at each node out of the set of all non-delayed and feasible jobs that job is selected for branching which has the highest priority value PV_{ij} and which has not yet been tested. A job is non-delayed if its earliest possible starting time coincides with the current scheduling period τ. By the definition of the current scheduling period we can always find a non-delayed job as long as not all jobs have been scheduled. The feasibility is hereby determined by requiring that, for the duration of the job's processing time, the required machine and a sufficient number of workers are available. In that case the scheduled job will be assigned to its machine in order to be processed for p_{ij} time units. Thus, this branch leads to a (left) search tree node, defined by assigning a not yet scheduled and non-delayed job i to its machine j. Associated with this node is again a scheduling period τ_1 and a subschedule which is obtained from the current one (in the current search tree node) through additionally scheduling job i. The start time of i in the obtained subschedule is τ which is the earliest possible start time. The completion time $\tau + p_{ij}$ of i is a lower bound on the earliest possible start time of every job that is not yet scheduled but supposed to be scheduled on machine j. The scheduling period τ_1 is greater or equal to the scheduling period τ of the preceding search tree node. Again, τ_1 is equal to the minimum among all earliest possible starting times of all unscheduled jobs, a set not containing job i any longer. An alternative branch and corresponding (right) search tree node with associated scheduling period τ_2 is generated where scheduling of the particular job i will be delayed until at least one other job in the order of the sequence of the priority values has been scheduled. In case a job is delayed its earliest possible start time is defined to be greater than the current schedule time τ. If job i can be feasibly assigned to its machine but no other job is feasible that node will not be generated. There are two reasons for delaying job i. Either i cannot be scheduled on its machine starting in period τ because of an insufficient number of available workers, or scheduling of i cannot lead to an optimal solution. In the first case, an earliest possible start time of i is the first period t where a sufficient number of workers becomes available again over at least p_{ij} time units. This can be the case if at least one currently scheduled job finishes processing. In the second case, job i will be delayed by one time unit, i.e. the machine is idle for one time unit or a number of unit time periods required for processing another job i' on the same machine. Hence, i is delayed until i' is scheduled on machine j and the earliest possible start of i is set to the completion time of i'. The only reason for keeping the machine idle for one period of time τ is that the resource demand can be kept smaller in τ if machine j is idle. There is a set of jobs that is scheduled in τ. If the same set of jobs is scheduled in $\tau + 1$ then job i must be delayed by at least two time units. Continuing this argumentation, in order not to deteriorate the current objective function value, job i must be delayed at least until another job is completed. If i is not scheduled at τ another job i' might be scheduled (not necessarily on the same machine). Again i cannot be started earlier than the completion time of another job in the partial schedule. Thus job i will be delayed until at least one other job is scheduled completely. The scheduling period τ_2 of the successor node of the current search tree node is equal to τ if there are non-delayed (earliest start time is equal to τ) and feasible jobs among the set of not yet scheduled jobs. Otherwise, τ_2 is set to the minimum earliest possible starting time from the set of all unscheduled jobs. In summary, the left search tree node always leads to a schedule where an additional job has been scheduled. The right search tree node keeps the current schedule as it is and increases the earliest starting time of a job.

4.4 Fathoming Criteria

Fathoming criteria serve to stop the depth-first strategy in case it is foreseeable that no feasible or no better solution than $\min\{UB, D\}$ can be generated. There are three different criteria:

1. At each time period t the sum of the remaining processing times for the jobs still to be processed at a given machine j is calculated for all machines. If one or more of these sums will exceed the remaining available time until T, no feasible solution is possible with the subschedule obtained so far.

2. The necessary worker capacity at any time period is greater than (or equal to) $\min\{UB, D\}$ and thus the solution cannot improve or does not exist.

3. The sum of priority values of the jobs not yet assigned at time period t is larger than the value of $((T-t)\min\{UB, D\})+(\min\{UB, D\}-$already used worker capacity at t). Thus the equation for LB1 is applied at each time period t to check for solutions which do not improve the upper bound.

4.5 The Pseudo-Code

1. Generate an upper bound UB (section 4.1).

2. Calculate a lower bound LB and the artificial lower bound ALB (section 4.2).
3. Set $UB := \min\{UB - 1, ALB, D\}$
4. Perform branch-and-bound with respect to the upper bound UB and the lower bound LB.
    (a) Generate next search tree node i
    (b) If any of the fathoming criteria in section 4.4 is met, then node i can be cut off.
    (c) Repeat step 4 until all nodes are fathomed.
5. (a) If a new feasible solution has been obtained with an objective function value $OV \leq UB$ then set $UB := OV - 1$ and perform step 4 until no feasible solution is obtained. Last feasible OV is the optimal solution.
    (b) If no feasible solution has been found so far having an objective function value $OV \leq D$ then set $UB := UB + 1$ and $LB = UB$. If $UB \leq D$ go to step 4, otherwise stop: no feasible solution exists.

Remark: In step 4.*b* a node based lower bound has been calculated according to 3. of the fathoming criteria given in section 4.4. If this bound is replaced by a node based artificial lower bound, nodes otherwise leading to an optimal solution in the search tree might be fathomed. In that case the run time of the branch-and-bound procedure will be decreased, however, finding an optimal solution can no longer be guaranteed.

5 The Data

The given problem appeared in the automobile production in Germany. We considered 7 huge press machines which were partitioned into two groups consisting of the machines $P1, P2, P3, P4$ and $P5, P6, P7$. There is a basic reason for machine grouping: workers can be assigned to either of the two groups allowing for group work with increased work motivation and satisfaction due to increased responsibility. In addition the algorithm's run time can be kept lower. The number of jobs to be scheduled on the machines of group 1 and 2 are 55 and 53, respectively. Each job consists of a certain lot size of several thousand pieces. A detailed description of the data is provided in Tables II-VIII.

For every machine the first column contains the job number (i.e. in practice it corresponds to a job identification and description); column 2 contains the job's lot size, while column 3 provides the number of items that can be produced per hour. Column 4 contains the number of required workers. Column 5 shows the approximate processing times for the whole lot size. These times are rounded up to full hours in order to include breaks, setup times of no more than 10 minutes (if they are larger than one hour they are, however, included as a separate job), and machine downtimes. Finally, the last column contains the result of our computation, the period for the beginning of a j ob.Aproduction cycle for the items of one specific car type consists of 5 days, for items of other car types it is only 3 days, i.e. every 3 (or 5) days the same set of items has to be produced in order to avoid production downtimes in downstream production processes. Furthermore, some items need to be processed only once within 3 weeks (in our data sets these are the items with no start time entry). In order to meet the aforementioned requirements and because different car types are produced simultaneously, the lengths of a production cycle has been set to 3 days, i.e. $T = 72$ periods with one period being equal to one hour. This requires that some of

Table II: Data for machine P1.

job	lot size	items per hr.	# workers	proc. time	start time
1	2000	780	5	3	46
2	3500	810	3	5	9
3	3000	840	3	4	55
4	3000	900	3	4	59
5	3000	900	3	4	63
6	4200	780	4	6	14
7	4200	780	4	6	25
8	4200	900	3	5	41
9	4200	900	3	5	50
10	4000	960	2	5	32
11	3000	900	2	4	37
12	3500	810	3	5	

Table III: Data for machine P2.

job	lot size	items per hr.	# workers	proc. time	start time
1	3500	960	2	4	28
2	3500	960	2	4	46
3	1800	720	3	3	60
4	2400	960	3	3	63
5	1800	960	2	2	68
6	7500	960	2	8	20
7	7500	960	4	8	12
8	7500	720	5	11	1
9	3000	960	4	4	38
10	1500	960	4	2	66
11	3000	960	4	4	42
12	1500	960	4	2	
13	1800	720	3	3	

Table IV: Data for machine P3.

job	lot size	items per hr.	# workers	proc. time	start time
1	3500	900	5	4	45
2	3500	960	2	4	39
3	3500	720	5	5	30
4	2000	720	5	3	57
5	2000	720	3	4	35
6	7000	750	3	10	10
7	7000	750	3	10	20
8	8000	960	4	9	1
9	3000	780	4	4	49
10	3000	780	4	4	53
11	2000	720	5	3	

Table V: Data for machine P4.

job	lot size	items per hr.	# workers	proc. time	start time
1	3000	720	7	5	20
2	2000	720	7	3	35
3	2000	720	5	3	49
4	2000	600	5	4	31
5	1500	600	5	3	52
6	2000	720	4	3	55
7	2000	720	4	3	58
8	3000	660	3	5	25
9	2000	660	3	3	42
10	2000	600	3	4	38
11	2000	660	3	3	61
12	6100	780	3	8	1
13	1000	660	3	2	64
14	1500	600	5	3	1
15	2500	780	3	4	38
16	2000	600	5	4	66
17	2000	660	4	3	42
18	2000	660	3	3	
19	1000	780	3	2	

Table VI: Data for machine P5.

job	lot size	items per hr.	# workers	proc. time	start time
1	11000	1140	2.5	10	1
2	3000	1080	2.5	3	57
3	11000	1140	2.5	10	11
4	9000	1200	2.5	8	30
5	4000	900	2.5	5	52
6	7000	1080	2.5	7	38
7	7000	1020	2.5	7	45
8	7000	840	2.5	9	21

Table VII: Data for machine P6.

job	lot size	items per hr.	# workers	proc. time	start time
1	5000	1080	2.5	5	44
2	7000	1080	2.5	7	19
3	1000	1200	2.5	2	67
4	2000	900	2.5	3	61
5	3000	960	2.5	4	57
6	5000	900	2.5	6	26
7	5000	900	2.5	6	32
8	2000	900	2.5	3	64
9	8000	960	2.5	9	10
10	7000	1200	2.5	6	38
11	1000	900	2.5	2	69
12	2000	1200	2	2	49
13	3000	1200	1.5	3	7
14	7000	1200	1.5	6	1
15	1000	1200	2.5	2	
16	5000	900	2.5	6	
17	2000	900	2.5	3	
18	2500	780	2.5	4	
19	2000	780	2.5	3	
20	5000	900	2.5	6	
21	1000	900	2.5	2	
22	2000	1200	2	2	
23	2000	960	1.5	3	

Table VIII: Data for machine P7.

job	lot size	items per hr.	# workers	proc. time	start time
1	3000	1020	4.5	3	54
2	7000	900	3.5	8	1
3	4000	960	3.5	5	49
4	2500	1200	2.5	3	31
5	2500	1080	2.5	3	34
6	2000	1080	2.5	2	47
7	3000	1080	2.5	3	37
8	3000	1080	2.5	3	40
9	2000	1080	2.5	2	57
10	7500	1200	2.5	7	17
11	7000	960	2.5	8	9
12	1000	1200	2.5	2	59
13	8000	1200	1.5	7	24
14	3000	1200	1.5	3	61
15	3000	1200	1.5	3	64
16	4000	1200	1.5	4	43
17	3000	1080	1.5	3	67
18	2000	1020	1.5	2	70
19	3000	1080	2.5	3	
20	2000	1200	2.5	2	
21	3000	1200	1.5	3	
22	1000	1080	1.5	2	

the items with a cycle time of 5 days to be exchanged between cycles in order to meet the 5 day cycle time requirement. An item replacing another one in a previous cycle has to have the same processing time as well as the same resource demand. Items that can replace each other are listed in Table IX. Any two jobs of a pair can replace each other on their respective machine.

6 The Results

The algorithm has been implemented in Delphi and all computations were performed on a PC with a 1.4 GHz processor and 512 MB RAM. Let us first consider the 3 machine subproblem. Its solution took less than a second. The starting times of the jobs are provided in Tables VI-VIII.

The maximum number of workers required per period of an optimal solution is 9. The initial solution (obtained through the procedure given in section 4.1) provides an upper bound $UB = 10$. The lower bound is equal to $LB = LB1 = 8$. The artificial lower bound is obtained as $ALB = 9$ and turns out to be a correct lower bound. Hence the first schedule generated by the branch and bound procedure was optimal. The solution is also shown by means of a Gantt-chart (see Figure 1). The branch-and-bound procedure selects variables with respect to decreasing priority values PV_{ij}, which results in a larger resource usage in the earlier periods. Indeed, for the very last period no job is scheduled. Up to period 50 all

Table IX: Possible job replacements.

machine	job replacements
P1	(2, 12)
P2	(9, 12) (3, 13)
P3	(4, 11)
P4	(5, 14) (10, 15) (4, 16) (7, 17) (9, 18) (13, 19)
P5	–
P6	(3, 15) (6, 16) (8, 17) (5, 18) (4, 19) (7, 20) (11, 21) (12, 22) (13, 23)
P7	(8, 19) (12, 20) (17, 21) (18, 22)

machines are continuously busy. P7 is busy over all 71 periods because the sum of processing times for all jobs on this machine equals 71. Periods in which P5 is idle require at most 6 workers. All jobs on P5 can be operated by only 3 workers (rounded up). Hence, any job schedule on P5 does not effect the overall solution and a machine breakdown of P5 for at most 13 hours does not lead to a new production plan and can be overcome requiring no further personnel. Moreover, bottlenecks on other press machines might be overcome easily if the workers from P5 are reassigned. Thus P5 has at least 13 hours of overcapacity for bottlenecks in other areas.

The optimal solution of the 4-machine subproblem requires up to 12 workers in some periods. The initial solution of the procedure from section 4.1 has an upper bound of $UB = 13$. The lower bound is $LB = 11$. In contrast to the 3-machine subproblem all 4 machines have certain amounts of idle time for machine repairs or inspection. Again, an overcapacity during at least 13 work hours is available to overcome bottlenecks in other production areas. Surprisingly, for the last 3 hours all machines are idle and the 12 workers are available for other tasks such as machine setup, material transportation etc. The solution of the 4-machine problem is shown in the Gantt-chart of Figure 2 and Tables II-V show the job starting times.

The original personnel requirements planning procedure for the press machine shop of the car plant was based upon the calculation of the least average number of workers necessary. This, however, corresponds to the calculation of our lower bound LB. To the group of 3 press machines were 8 workers assigned, while to the machine group consisting of 4 press machines 11 workers were assigned. As a result some of the jobs could not be finished in time and had to be worked off in overtime requiring additional work hours. For the remaining jobs either too many workers were assigned or there were not enough workers available. In the latter case the available workers could not do their jobs and had to wait for support set free at other machines. These effects led to a steadily increasing number of expensive overtime hours.

7 Conclusions

We considered personnel planning and scheduling in one particular area of automobile production, the press machine shop. A simple personnel requirements planning procedure on the tactical level combined with a planning model considering flexible labor schedules in order to include seasonal effects and non predictable demands did not provide the required labor capacity without an accumulation of tremendous overtime hours. The factory wished to produce in batches to cover the demand for 3 to 5 days. This resulted to a reduction in the number of setups and to cycle-stock inventory policies which guarantee a highly secured service. However, keeping the batch sizes flexible would reveal another potential for optimization and probably an improved work schedule. An analysis of the real situation showed that the number of people at work often was insufficient to produce all jobs. Thus, jobs had to be postponed to a later time in order to be produced in avoidable overtime hours. As a result the number of people exceeded the number required to work off the remaining jobs. In order to overcome these shortcomings, we provided a model to balance the number of workers subject to constraints ensuring that all jobs can be finished in time. An exact as well as a heuristic solution procedure, both of a branch-and-bound type, generated excellent results in the considered practical planning situation. The heuristic has been obtained from the branch-and-bound through an incomplete enumeration of the search, based on an artificial lower bound that might have prevented the exploration of promising search tree branches. All optimization runs were limited to different groups of machines of moderate size. The management's objective to consider machine groups as separate units is based on the desired outcome to obtain groups of workers as well. Hence, a group of workers remains stable for some time, it has a corresponding machine group and replacing or transferring workers to different groups should be exceptional. Our machine groups were established manually, however in a way to avoid unnecessary long distances (hundreds of meters) for workers when moving from one machine to another. Considering the workers' qualifications for their group and their machine assignment was less important. However, group formation and flexible job assignments provide another area for potential savings (cf. Askin and Standridge [2]). A welcomed side-effect of considering machine groups of moderate size is the potential to generate an optimal solution for each of them within a reasonable time frame. The combination of these solutions resulted to excellent outcomes for the factory's manpower planning.

Acknowledgement

We would like to thank two anonymous referees for their constructive comments and suggestions that improved the presentation of this paper.

References

[1] A.C.F. Alvim, M. Iori, C.C. Ribeiro, F. Glover and D.J. Aloise, A hybrid improvement heuristic for the one-dimensional bin packing problem, *Journal of Heuristics* 10 (2004) 205–229.

[2] R.G. Askin and C.R. Standridge, *Modeling and Analysis of Manufacturing Systems*, John Wiley & Sons, New York, 1993.

[3] W.W. Bein, P. Brucker, L.L. Larmore, J.K. Park, The algebraic Monge property and path problems, *Discrete Applied Mathematics* 145 (2005) 455–464.

[4] J. Błażewicz, K. Ecker, E. Pesch, G. Schmidt and J. Weglarz, *Scheduling Computer and Manufacturing Processes*, 2nd ed., Springer-Verlag, Berlin Heidelberg, 2001.

[5] P. Brucker, A. Drexl, R. Möhring, K. Neumann and E. Pesch, Resource-constraint project scheduling: notation, classification, models, and methods, *European Journal of Operational Research* 112 (1999) 3–41.

[6] M. Caramia and P. Dell'Olmo, Assessing the resource usage in scheduling with incompatabilities, *OR Spectrum* 25 (2003) 521–547.

[7] E.G.Jr. Coffman, M.R. Garey and D.S. Johnson, Approximation algorithms for bin packing: a survey, in *Approximation Algorithms for NP-hard Problems*, D. Hochbaum (ed.), PWS Publishing 1997, pp. 46–93.

[8] R. Hübscher and F. Glover, Applying tabu search with influential diversification to multiprocessor scheduling, *Computers and Operations Research* 21 (1994) 877–884.

[9] K.L. Krause, V.Y., Shen and H.D. Schwetman, Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems, *J. Assoc. Comput. Mach.* 22(1975) 522–550. Erratum: *J. Assoc. Comput. Mach.* 24(1977) 527.

Manuscript received 17 October 2004
revised 13 March 2005
accepted for publication 18 April 2005

Erwin Pesch
University of Siegen, FB 5, Institute of Information Systems Hoelderlinstr. 3, D-57068 Siegen, Germany
E-mail address: erwin.pesch@uni-siegen.de

Ulrich A.W. Tetzlaff
DANET GmbH, Gutenbergstr. 10, D-64331 Weiterstadt, Germany
E-mail address: uli.tetzlaff@web.de

List of Figures

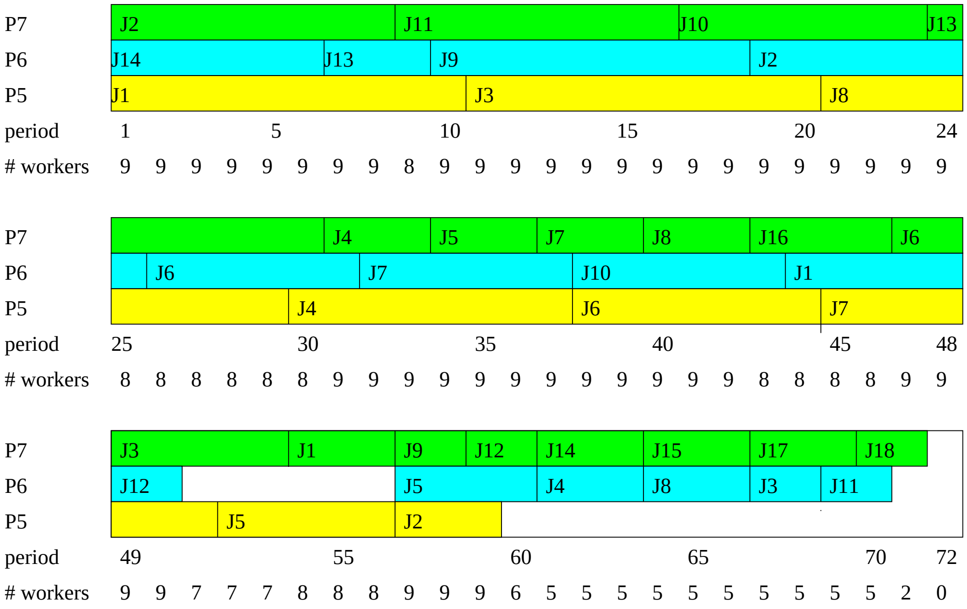

Figure 1: Gantt-chart for P5, P6 and P7

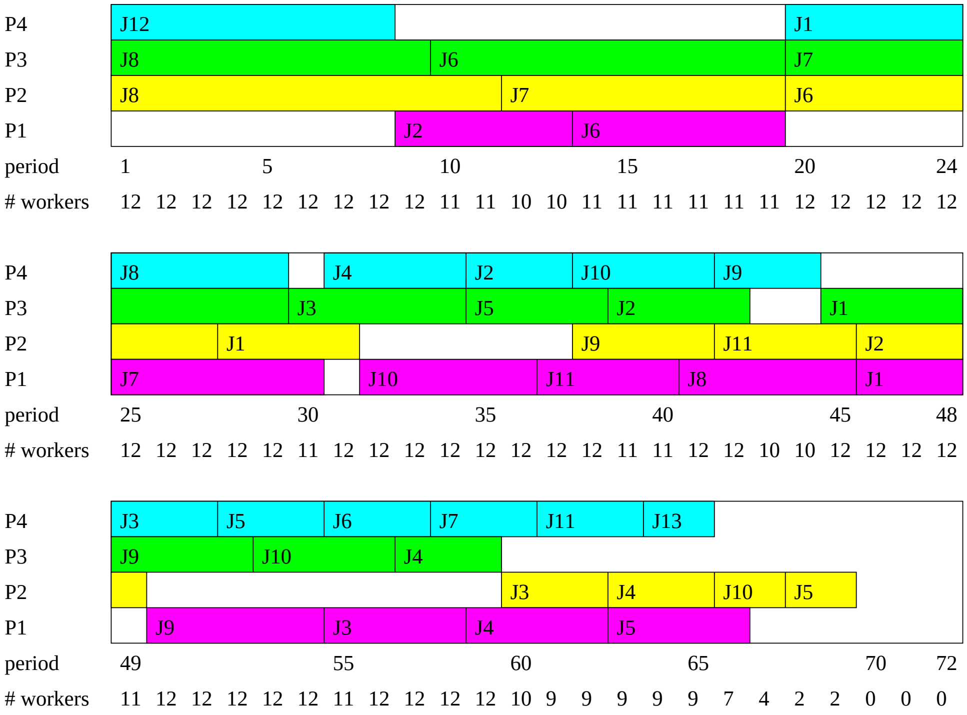

Figure 2: Gantt-chart for P1, P2, P3 and P4