The Batch Loading and Scheduling Problem

by

Gregory Dobson

and

Ramakrishnan S. Nambimadom

Working Paper QM 92-03

September 1992

Subject Classification:

1) Production/Scheduling: Approximations/heuristic: Batch Processors

2) Manufacturing: Automated systems: Batch Processors in semiconductor, steel and ceramic industries.

The Batch Loading and Scheduling Problem

by

Gregory Dobson

and

Ramakrishnan S. Nambimadom

Abstract

This paper discusses a problem that commonly occurs in the batching and scheduling of certain kinds of batch processors. Examples of these processors include heat treatment facilities, particularly in the steel and ceramic industries as well as a variety of operations in the manufacture of integrated circuits. There are a set of jobs waiting to be processed. The processing time for a batch depends only on the family and not on the number or the volume of jobs in the batch. Each job is associated with a given family. The schedule must organize jobs into batches where each batch consists of jobs from a single family. The batches must be sequenced on the facility. Each job has a weight or delay cost and a volume. The facility can handle a set of jobs from the same family as a batch only if, the total volume of the jobs is less than the capacity of the facility and all the jobs belong to the same family. The objective is minimizing the mean weighted flow time.

The paper presents an integer programming formulation for this problem. A partial LP relaxation provides a lower bound. In addition, we present several heuristics based on various subproblems. These include a greedy heuristic, a successive knapsack heuristic and a generalized assignment heuristic. The conclusions of the computational study show that the successive knapsack and generalized assignment heuristic perform better than the greedy. The generalized assignment heuristic does slightly better than the successive knapsack heuristic but is substantially slower. The study also shows that the size of the job relative to the capacity of the facility affects the performance of the heuristics. Finally a worst-case analysis of the greedy heuristic is presented.

1 Introduction

This paper investigates a single machine batching and scheduling problem that we call the batch loading and scheduling problem. There is a set of jobs available for a single machine to process. Each job, *i*, has a volume v_i. Each job also belongs to a family, g_i. The machine can process several jobs from the same family, *j*, simultaneously, as a batch, provided that the total volume of these jobs does not exceed the capacity of the machine, *V*. However, jobs belonging to different families have to be processed separately. The processing time for a batch of jobs from family *j*, t_j, depends only on the family and not on the number or volume of jobs in the batch. An example is depicted in Figure 1. It shows the families of jobs organized into batches and those batches scheduled on the processor.

Examples of these processors include heat treatment facilities, particularly in the steel and ceramic industries as well as a variety of operations in the manufacture of integrated circuits, such as diffusion, oxidation, and certain chemical vapor deposition processes. In these cases, the processor is capable of performing a variety of tasks. A family is the set of jobs that the facility must process in the same manner. In reality, the volume occupied by the jobs would not be just the sum of the individual volumes, due to the shape of the various jobs. However, we assume that the single parameter represents the volume accurately and thus obviates the necessity of considering the highly complicated two and three dimension packing problems. Such an approach is likely to be a good first cut. In addition, the volume of a job may not represent the size of a job, but rather the consumption of some other resource that may be in limited quantity, such as the energy that a job may absorb. Glassey and Weng [1991] and Fowler et al. [1991] describe processes like this for a variety of operations, particularly, the manufacture of integrated circuits.

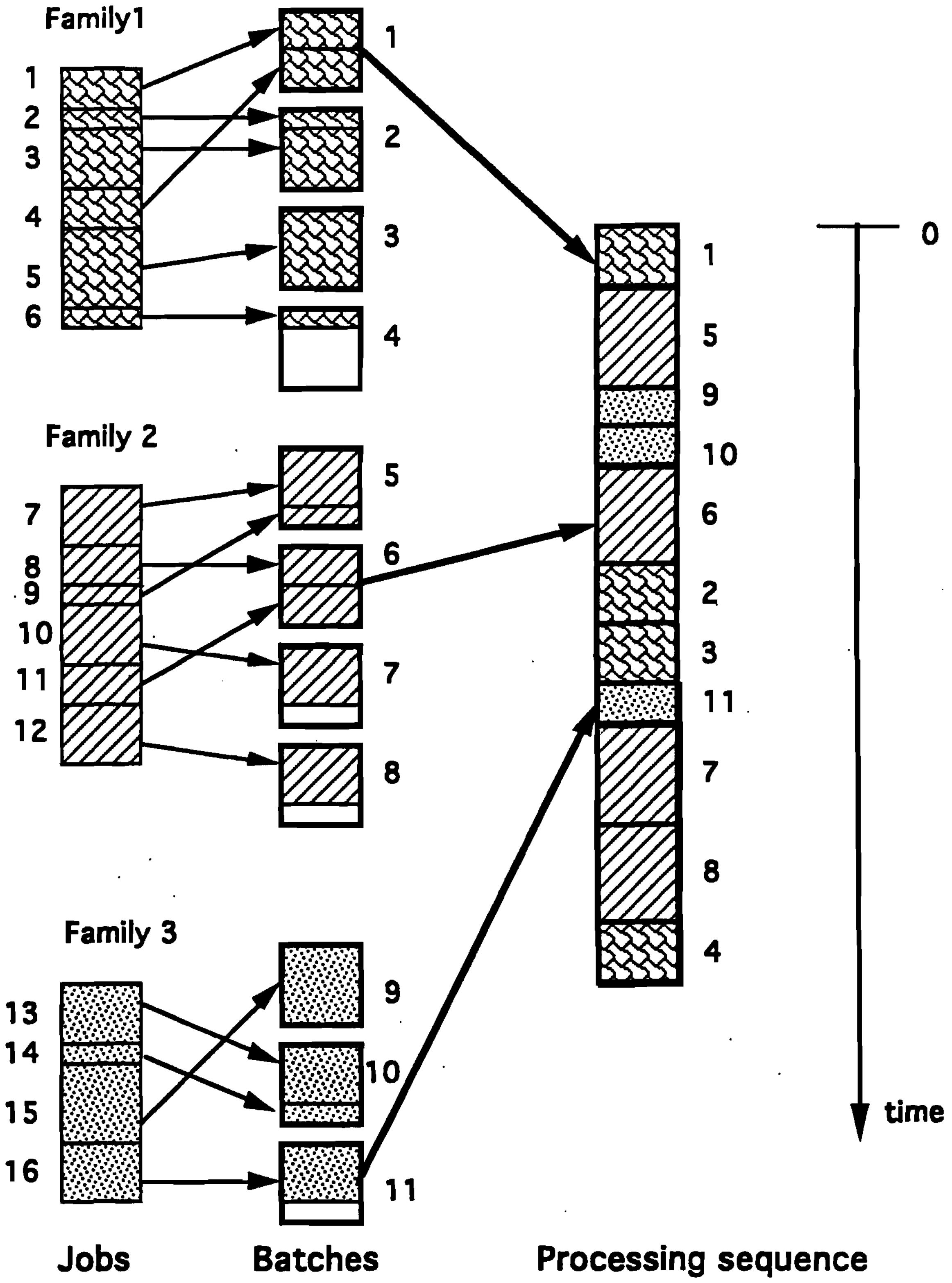

Figure 1. The batch loading problem

The objective function considered here is to minimize the mean flow time. This is a common objective function in both scheduling theory and practice. Flow time or lead time is a important measure of performance for several reasons. First, it is directly related to work-in-process inventory (WIP) via Little's Law. Second, longer lead times forces the firm to carry additional safety stock when faced with uncertain demand. Finally, forecast uncertainty is likely to rise super-linearly in lead time forcing the firm to hold additional safety stock.

The literature discussing batch production of this type, is relatively scarce. Ahmadi et al. [1992] give a number of situations involving batch processors whose processing time does not depend on the number of jobs being processed. They consider a 2-machine problem where one of the two machines is a batch processor. In their formulation, however; the jobs are all of the same size and hence the single machine version of their problem is easy. The presentation here generalizes their results and considers problems where the size of the jobs is arbitrary. Lee et al [1992] consider a different kind of batch processor, where the processing time of each batch is the maximum of of the processing times of the jobs in it, and obtain algorithms for a number of different performance measures. Ikura and Gimple [1986] considered the problem of determining whether there exists a schedule where all jobs are complete by their due dates, given release times and due dates. Glassey and Weng [1991] and Fowler et al. [1991] consider a dynamic version of this problem, where the use of knowledge about future arrivals is used to devise a control strategy. The first paper considers the case where all jobs belong to the same family, while the latter considers multiple families. In both cases, the volume of jobs is assumed to be the same for all jobs. Lefrancois et al. [1991] describe a study based on annealing furnaces in a Canadian rolling mill and look at a multi-objective function version of this model which integrates a simulation model with search-based subroutines to optimize batching and sequencing of jobs to minimize flow times and lateness and maximize the efficiency of the annealing furnaces. An average job size is used for the purpose of the analysis. This paper differs from these papers from at least two perspectives. First we examine a static version of this problem in which all jobs are available for scheduling at time 0. Secondly, in our model the volume of each job can be different. In some cases, the volume of individual jobs is quite small compared with the capacity of the processor. In such cases, this may not be a crucial factor. However, in many cases this may not be so. In particular in the steel industry, the volume of individual metal pieces being annealed may vary considerably from piece to piece. Another example is in the semi-conductor industry where each job may itself be batches of individual chips that cannot be split for quality control reasons. Each of these batches could be quite large compared to the size of the processor.

Some related work on batching is relevant here as the mathematical structures are fairly similar. Dobson and Karmarkar [1986] survey various problems in the area of job shop scheduling and discusses various formulations and relaxations. Dobson et al. [1987] and Santos and Magazine [1985] discuss batching to minimize flow time for the single-machine case. In these papers however, the size of the batch determines the processing time. The batch processing time is equal to the setup time plus a per-unit processing time multiplied by the size of the batch. Such models accurately represent metal cutting and bending.

The remainder of this paper is organized as follows. Section 2 presents a formulation of the problem as an integer program. Section 3 discusses a relaxation that provides a lower bound. Section 4 describes several heuristics and Section 5 describes the computational study we did to evaluate the heuristics. Section 6 gives a worst-case analysis for the simplest of these heuristics. Section 7 concludes with a discussion of some avenues for future research.

2 Formulations of the Batch Loading Problem

This section formulates the batch loading problem as an integer program. The formulation presented must make two types of decisions. The first involves assigning jobs to batches and the second involves sequencing those batches. Batches are arbitrarily numbered and the set of batches associated with a given family is known.

The data for the problem are:

i	index for jobs;

j	index for families;
k	index for batches;
l	index for positions in the processing sequence;
m	the number of jobs;
c_i	the holding cost or the cost per unit of time delay for job i;
v_i	the volume of job i;
g_i	the family of job i;
t_j	the processing time for a batch of family j;
t^k	the processing time for batch k,
	$= t_j$ where batch k is associated with family j;
I_j	the set of items in family j;
I^k	the set of items that can be included in batch k
	$= I_j$ where batch k is associated with family j;
B_j	the set of batches for family j;
B^i	the set of batches that can include i
	$= B_j$ where item i is a member of family j;
V	capacity of the processor.

For example in Figure 1,

$$B^1 = \dots = B^6 = B_1 = \{1, 2, 3, 4\}$$

$$B^7 = \dots = B^{12} = B_2 = \{5, 6, 7, 8\}$$

$$B^{13} = \dots = B^{16} = B_3 = \{9, 10, 11\}$$

$$I^1 = \dots = I^4 = I_1 = \{1, \dots, 6\}$$

$$I^5 = \dots = I^8 = I_2 = \{7, \dots, 12\}$$

$I^9 = \ldots = I^{11} = \mathrm{I}_3 = \{13,\ldots,16\}$

and

$t^1 = \ldots = t^4 = t_1$

$t^5 = \ldots = t^8 = t_2$

$t^9 = \ldots = t^{11} = t_3$

In this paper, we shall from time to time, assume that batch k will precede a batch k' if they are associated with the same family and $k < k'$. We can do this without loss of generality since we can always renumber the batches to accommodate this condition.

The integer programming formulation that assigns jobs to batches and batches to positions uses the variables:

x_{ik}	$= 1$	if job i is assigned to batch k, 0 otherwise;
y_{kl}	$= 1$	if batch k is assigned to position l in the sequence, 0 otherwise;
z_{il}	$= 1$	if job i is assigned to a batch that is assigned to position l in the sequence, 0 otherwise;
f^l	$=$	flow time of the batch at position l of the sequence.

The formulation to minimize weighted flow time is:

$$z^* \equiv \min_{x,y,z} \quad \sum_{i} \sum_{l} c_i z_{il} f^l$$

subject to

$$\sum_{k \in B^i} x_{ik} = 1 \qquad \forall i \qquad \text{(1a)}$$

$$\sum_{i \in I^k} v_i x_{ik} \le V \qquad \forall k \qquad \text{(1b)}$$

$$\sum_{l} y_{kl} = 1 \qquad \forall k \qquad \text{(1c)}$$

$$\sum_{k} y_{kl} \leq 1 \qquad \forall l \qquad \text{(1d)}$$

$$f^{0} = 0 \qquad \text{(1e)}$$

$$f^{l} \geq 0 \qquad \forall l \qquad \text{(1f)}$$

$$f^{l} = \sum_{k} t^{k} y_{kl} + f^{l-1} \qquad \forall l \qquad \text{(1g)}$$

$$z_{il} = \sum_{k \in B^{i}} x_{ik} y_{kl} \qquad \forall\, i,l \qquad \text{(1h)}$$

$$x_{ik},\, y_{kl},\, z_{il} = 0,1 \qquad \forall\, i,k,l \qquad \text{(1i)}$$

Constraints (1a) force each job i to be assigned to exactly one batch k. Constraints (1b) ensure that the volume of jobs assigned to batch k will not exceed the capacity of the processor. Constraints (1c) and (1d) guarantee that each batch is assigned exactly one position in the sequence and each position in the sequence is assigned at most one batch. Constraints (1e), (1f) and (1g) compute the flow time for each position l. Finally constraints (1h) define the variables, z, that connect jobs to sequencing positions. Note, that we define a sufficiently large number of batches for each family so that each item could, if necessary, be placed in a separate batch. The number of processing positions must be large enough to accommodate all the batches. The solution to the formulation will usually have a substantial number of empty batches assigned to positions at the end of the sequence. These will not affect the objective function since they contain no jobs and thus x_{ik} is 0 which forces z_{il} to be 0.

Although the constraints in this formulation are not linear due to the non-linear definition of z_{il} in constraints (1h), we can modify them so that they are linear at the expense of adding additional variables and constraints. Define the 0–1 variable $z_{ikl} = 1$ if job i is assigned to batch k and batch k is assigned to position l in the sequence, 0 otherwise. This variable is only defined for jobs i and batches k that are from the same family. Now constraints (1h) can be replaced by

$$z_{il} = \sum_{k \in B^i} z_{ikl} \cdot$$

The minimization objective and the additional constraints

$$z_{ikl} \geq x_{ik} + y_{kl} - 1 \qquad \forall k \in B^i \ \forall i,l$$

force the variables $\{z_{ikl}\}$ to have correct values.

The objective function can also be linearized by the following approach. First note that the flow time of the batch in position l, fl can be written as

$$f^l = \sum_{p=1}^{l} \sum_{k} t^k y_{kp}$$

We can therefore write the objective function as

$$\sum_{i} \sum_{l} c_i z_{il} \sum_{p=1}^{l} \sum_{k} t^k y_{kp} = \sum_{i} \sum_{k} c_i t^k \sum_{l} \sum_{p=1}^{l} z_{il} y_{kp}$$

We can now linearize the objective function by defining a set of variables $w_{il\,kp}$ as 1, if z_{il} and y_{kp} are both 1, 0 otherwise. The objective function can now be written as

$$\sum_{i} \sum_{k} c_i t^k \sum_{l} \sum_{p=1}^{l} w_{il\,kp}$$

Note that the following constraint and the minimization objective force $w_{il\,kp}$ to have the correct values.

$$w_{il\,kp} \geq z_{il} + y_{kp} - 1$$

Unfortunately these modifications increase the size of the formulation rather substantially. Despite the non-linear nature of the original formulation there is a relatively easy way to get a good lower bound as we show in the next section.

3. A Lower Bound

In the formulation of the previous section if we relax the integer constraint $x_{ik} = 0,1$ to $0 \leq x_{ik} \leq 1$, then the solution to the resulting problem has a rather simple form which is easy to compute. We call this relaxation the partial LP relaxation or (pLP) since we do not relax the integrality constraint on the remaining variables. The procedure to compute the solution is as follows.

Procedure R:

1. For each family, place the jobs in descending order by the index $\frac{c_i}{v_i}$ and place the batches associated with that family in ascending order of batch numbers.
2. For each family, assign the jobs (or fraction of jobs) to batches in this order. Fill each batch to the capacity V. This is possible since the relaxation allows us to split jobs across batches.
3. Sequence the batches in decreasing order by the index $\frac{c^k}{t^k}$, where $c^k \equiv \sum_{i \in I^k} c_i x_{ik}$, and x_{ik} is the fraction of item i placed in batch k. This delay cost for a batch, c^k, is simply the delay cost of the jobs (or fractions of jobs) assigned to it.

The procedure is depicted in Figure 2.

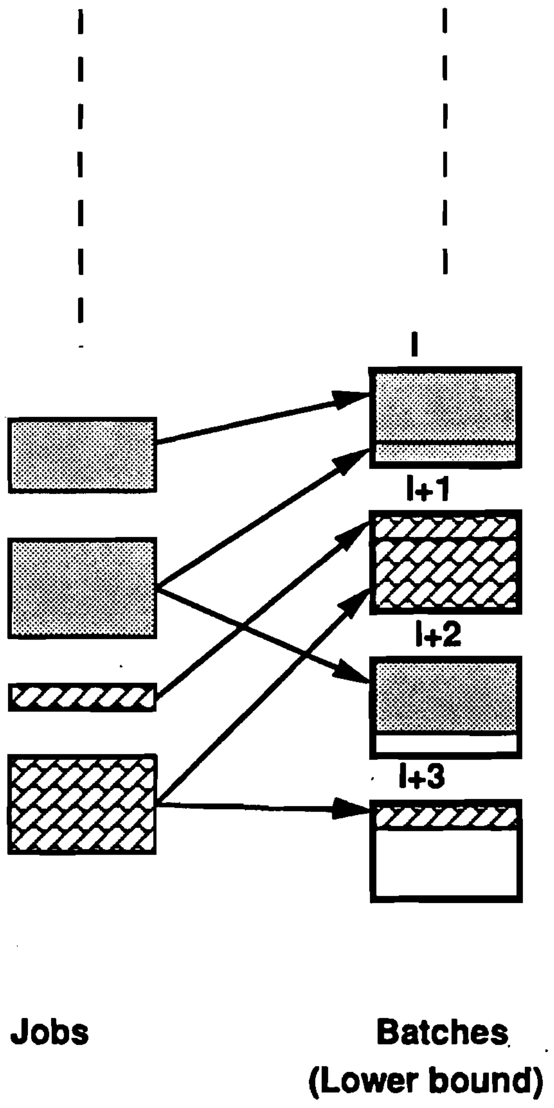

Figure 2 Formation of batches for the lower bound

We now proceed to show that procedure **R** solves the relaxed problem. First note that since the decisions made in our problem essentially involve fixing the x and y variables (assigning jobs to batches and batches to processing positions), our problem can be written as

$$z = \underset{y}{\text{Min}}\left\{ \underset{x\,:\,(x,y)\,\in\,S}{\text{Min}} \quad f(x,y) \right\}$$

where $f(x,y)$ is the objective function as a function of the x, y variables and S is the set of feasible values of (x,y).

We shall show that the x that solves the inner minimization is the same for all feasible values of y, that satisfy the condition that batch k will precede batch k', if they are associated with the same family and $k < k'$. As noted earlier the last condition does not in any way constrain our choices. Furthermore, this value of x, say x^*, is given by procedure **R**. We shall also show that the y that solves the problem for any fixed x, (in particular for $x = x^*$) is given by procedure **R**. Thus the relaxed problem can be solved by making the two sets of decisions separately.

For the first half of the proof, suppose we are given a sequence. Our problem is to assign jobs to batches which shall then be processed in this sequence. We obtain this subproblem from (1) by fixing $\{y_{kl}\}$, and substituting (1h) to eliminate the $\{z_{il}\}$. We obtain as the objective.

$$\underset{a}{\text{Min}} \sum_{i} \sum_{l} c_i \left(\sum_{k \in B_i} x_{ik} y_{kl} \right) f^l$$

or after rearranging and separating the sum over jobs i to one over families and then one over jobs for that family we obtain

$$\underset{x}{\text{Min}} \sum_{j} \sum_{i \in I_j} \sum_{k \in B_i} c_i x_{ik} \left(\sum_{l} y_{kl} f^l \right)$$

Note that with y_{kl} fixed, f^l is also fixed; so we can replace $\sum_l y_{kl} f^l$ with $\bar{f}^k$. The problem then separates by family. For a given family j we have

$$\max_{x} \sum_{i \in I_j} \sum_{k \in B^j} c_i x_{ik} \bar{f}^k$$

$$\text{s.t.} \quad \sum_{k \in B^j} x_{ik} = 1 \qquad \forall\, i \in I_j$$

$$\sum_{i \in I^k} v_i x_{ik} \leq V \qquad \forall\, k \in B_j$$

$$x_{ik} \geq 0. \qquad \forall i \in I_j,\, k \in B_j$$

To complete the proof of the first half of our assertion, we need to show that the solution to the problem above does not depend on the absolute values of $\bar{f}^k$ but rather on their order.

Suppose this is not the case. Assume that the batches associated with this family are numbered from 1 through p. By our assumption regarding the processing positions of batches within a family, the flow times of these batches satisfy $\bar{f}_1 < \bar{f}_2 < \cdots < \bar{f}_p$. By hypothesis there must be two jobs i in batch k at position l and i' in batch k' at position l' such that $\frac{c_i}{v_i} < \frac{c_{i'}}{v_{i'}}$ but $l < l'$ and $x_{ik} > 0$ and $x_{i'k'} > 0$.

Now adjust the x's by δ as follows

x_{ik}	decreases by δ/v_i,
$x_{ik'}$	increases by δ/v_i,
$x_{i'k}$	increases by $\delta/v_{i'}$, and
$x_{i'k'}$	decreases by $\delta/v_{i'}$.

Observe that the constraints on the x's are still satisfied provided δ is sufficiently small, less than $\min(x_{ik}, 1\text{-}x_{ik'}, 1\text{-}x_{i'k}, x_{i'k'})$ which is greater than 0. The change in the objective function is

$$c_i\left(\frac{-\delta}{v_i}\right)f_l + c_i\left(\frac{\delta}{v_i}\right)f_{l'} + c_{i'}\left(\frac{\delta}{v_{i'}}\right)f_l + c_{i'}\left(\frac{-\delta}{v_{i'}}\right)f_{l'}$$

$$= \delta\left(\frac{c_i}{v_i} - \frac{c_{i'}}{v_{i'}}\right)\left(f_{l'} - f_l\right) < 0.$$

Thus, this solution cannot be optimal. Note that the argument only depends on the fact that $f_{l'} - f_l > 0$, i.e. on the order of the batches not on the magnitude of the difference. Thus we have proved that the solution to the inner minimization over x for any given y is solved by the assignment of jobs to batches according to procedure R.

To complete our proof, observe that for any (possibly fractional) assignment of jobs to batches, the remaining problem of sequencing batches is a minimum weighted flow time problem on single machine. To see this from the formulation, fix $\{x_{ik}\}$ to any feasible value, i.e. one that satisfies (1a) (1b) and $0 \le x_{ik} \le 1$. Substituting (1h) into the objective and rearranging we obtain

$$\underset{y}{\text{Min}} \sum_l \sum_i \sum_{k \in B^i} c_i \, x_{ik} y_{kl} f^l$$

subject to (1c), (1d), (1e), (1f) and (1g).

or rearranging we have

$$\underset{y}{\text{Min}} \sum_l \sum_k \left(\sum_{i \in I^k} c_i x_{ik} \right) y_{kl} f^l$$

subject to (1c) (1d) (1e), (1f) and (1g).

This problem is that of sequencing batches for a given assignment of jobs to batches. It is clearly equivalent to the single machine minimum weighted flow time problem. Hence the optimal sequence of batches is given by procedure R.

This shows that (pLP) can be solved by solving the two subproblems separately.

4 Heuristics for the Batch Loading Problem

Recall that the Batch Loading Problem has two imbedded sub-problems; one is the assignment of jobs to batches and the other is the sequencing of batches. In the previous section we showed that the latter problem is easy to solve for any given assignment of jobs to batches; one can use the shortest weighted processing time rule, where the weights are given by the total batch cost. The job assignment problem, for a given assignment of processing positions to families splits into m subproblems - one for each family - each of which is a generalized assignment problem [Fisher et al. 1986].

This section describes three heuristics for the Batch Loading Problem. Each heuristic is based on the decomposition described above that allows us to solve the batch sequencing problem, once the job assignments have been made. Each method solves the job assignment problem approximately and then uses the batch sequencing rule for the given assignment. The heuristics are called the greedy heuristic, the successive knapsack heuristic, and the generalized assignment heuristic. The three techniques apply increasingly sophisticated ideas to the job assignment problem with, as we shall see, improvements in the solution values, but at the cost of increasing the computational effort.

4.1 A Greedy Heuristic

A procedure, that generates a reasonable feasible solution in one pass, is given below. The procedure involves the following steps:

```
Partition the jobs on the basis of their family
For each family
    Sort the jobs in non–increasing order of the cost to volume ratio, $\frac{c_i}{v_i}$.
    For each job within the family
```

```
            Add the job to the first batch of the family that has space to accommodate it.
    Sequence the batches by the weighted-processing-time rule.
```

4.2 A Successive Knapsack Heuristic

Another intuitively appealing heuristic is to maximize the cost of the first batch within each family and then repeatedly maximize the cost of the next batch by using the remaining items. Although such a procedure is obviously not optimal, this successive solving of knapsack problems leads to the following heuristic.

```
Partition the jobs on the basis of their family.
For each family
    While there remain items not assigned to a batch
        Solve the knapsack problem to obtain the contents of the next batch
Sequence the batches by weighted-processing-time rule
```

Note, like the previous heuristic, this is a one pass procedure since, the batching is done independently of the sequencing of the batches. The standard dynamic programming approach is used to solve the knapsack problem.

4.3. The Generalized Assignment Heuristic

This sub-section describes an iterative heuristic which repeatedly performs the two main tasks of sequencing and batching one at a time, keeping the other fixed. The assignment of jobs to batches is done by solving the following generalized assignment problem.

Suppose the assignment of processing positions to families is given. Then as noted before, time of the *k*th batch of, say family 1, can be calculated, and the problem separates out by family. Assume that the batches associated with family 1 are numbered from 1 through $|I_1|$ and batch k precedes $k+1$.

Let,

f_k = flow time of the kth batch of family 1, $\forall\ k = 1,\ldots,|I_1|$;

x_{ik} = 1 if job i is assigned to the kth batch of it's family, 0, otherwise, $\forall\ k \in B_1$ and $\forall\ i \in I_1$.

Then the subproblem associated with family 1 is given by,

$$\text{Min} \quad \sum_{i\in I_1} \sum_{k\in B_1} c_i x_{ik} f_k$$

$$\text{subject to} \quad \sum_i v_i x_{ik} \le V \qquad \forall k\in B_1 \qquad (4a)$$

$$\sum_k x_{ik} = 1 \qquad \forall i\in I_1 \qquad (4b)$$

$$x_{ik} = 0,1 \qquad \forall i\in I_1, k\in B_1 \qquad (4c)$$

The procedure for the generalized assignment heuristic is as follows,

1. We begin with an arbitrary assignment of processing positions to families or we obtain the optimal processing positions for an arbitrary assignment of jobs to batches. Either way we end up with an assignment of processing positions to families.

2. Keeping these assignments fixed we solve the resulting generalized assignment problem to obtain the allocation of jobs to batches. Note that we have to solve one generalized assignment problem per family.

3. The batches of the various families are pooled together and the batches are ordered based on the weighted processing time rule. This gives us a new set of allocations of processing positions to families. Thus, we can now repeat the procedure iteratively till a stopping condition, based on the number of iterations or the improvement over the last few iterations, is met.

We solve the generalized assignment problem by dualizing constraints (4b). This procedure is know to be very effective method for the general generalized assignment problem. The same approach is followed here. However, in our particular generalized assignment problem, the cost of assigning job i to batch k is a product of a factor dependent only on i (c_i), and a factor dependent only on k (f_k). The implication of this is that all jobs would prefer to be in batch k rather than batch $k + 1$. In contrast, in the general generalized assignment problem, each job has a different preferred order of batches, and hence the assignment of some jobs becomes very easy to resolve. Thus the performance of this approach, while good, does not quite match the results obtained for an arbitrary generalized assignment problem by Fisher [1981]. It also makes it unlikely that the multiplier adjustment procedure considered by Fisher et al. [1986] would be particularly successful, as it makes use of the property that not all jobs would desire the same batch.

In this paper, a sub-gradient algorithm adjusts the multipliers until either it obtains a primal feasible solution or it reduces the step size below a certain specified number. The following proposition shows that the multiplier, connected with job i, λ_i, can be restricted to be at least, $c_i f_2$. If at any iteration, the calculated value of λ_i is less than $c_i f_2$, then the procedure resets it to the latter value

Proposition. *$\lambda_i, \geq c_i f_2$ is a valid dual constraint.*

Proof. We need to show that this constraint does not affect the value of the solution to the dual problem.

If we relax constraints (4b), then we get the following problem,

$$z_{LR}(\lambda) = \sum_i \lambda_i - \sum_k z_k(\lambda)$$

where

$$z_k(\lambda) = \max \sum_i (\lambda_i - c_i f_k) x_{ik}$$

subject to constraints (4a).

Note that if $\lambda_i \leq c_i f_k$, then $x_{ik} = 0$. Now, suppose we do not impose the above restriction on the minimum value of the multipliers. Consider any set of multipliers λ and let $x(\lambda)$ be the solution to the relaxed problem. Let,

$$\lambda_i' = \max(\lambda_i, c_i f_2)$$

and let $x'(\lambda')$ be an optimal solution to the relaxed problem corresponding to the new value of the multipliers. Let $A = \{i : \lambda_i \leq c_i f_2\}$. Thus for all $i \in A$,

$$\lambda'_i > \lambda_i \text{ and } \lambda_i, \lambda'_i \leq c_i f_k \ \forall \ k \geq 2$$

Therefore $x'_{ik} = x_{ik} = 0$ for all $k \geq 2$ and $i \in A$. and hence, $z_k(\lambda) = z_k(\lambda')$ for all $k \geq 2$

Now consider the sub-problem associated with the first batch. When we moved from λ to λ' we increased the objective function coefficients for all $i \in A$. The total increase in the optimal value of the objective function, $z_1(\lambda') - z_1(\lambda)$ is at most the sum of these increases, or

$$z_1(\lambda') - z_1(\lambda) \leq \sum_{i \in A} (\lambda'_i - \lambda_i)$$

Therefore,

$$z_{LR}(\lambda') - z_{LR}(\lambda) = \sum_{i \in A} (\lambda_i' - \lambda_i) - \left(z_1(\lambda') - z_1(\lambda)\right) \geq 0$$

Thus for any arbitrary set of multipliers, the solution of the relaxed problem is improved by ensuring that no multiplier is less than $c_i f_2$.■

Finally, if the subgradient search for multipliers ends with a primal feasible solution then clearly we have an optimal solution to the generalized assignment solution. If not, we use the following procedure to obtain a feasible solution. We choose the Lagrangian multipliers that gave the best lower bound and solve the relaxed problem batch by batch, with the added restriction that if a job has been assigned in an earlier batch then it can no longer be assigned. The procedure takes the jobs that are not assigned to any batch, and sorts them in non–increasing order of cost to volume ratio, and assigns them in this order to the first batch that has space for them.

5 Computational Experiments and Results

In order to determine how well the heuristics would perform we tested them on a wide range of problems. We generated four categories of problems. In each case the capacity of the processor was 50. The categories differed by the size of the jobs. The first category had relatively small jobs. We generated the volume of a job from a uniform distribution from 1 to 10. The second category had larger jobs. Here we generated the volume from a uniform distribution from 1 to 25. The third category had still larger jobs, with volumes generated from a uniform distribution from 1 to 50. Finally the fourth category was designed to test the heuristics when there were no small jobs. Here the volume of a job was generated from a uniform distribution from 13 to 38. For each of these categories we created problems with 3 families and 5 jobs/family, with 5 families and 20 jobs/family and with 10 families and 50 jobs/family. For each of the 12 sets, we created 20 instances and solved them using the three heuristics. We also computed the lower bound. The distribution of costs for a job was uniform from 0 and 1. Similarly the distribution of processing times of a family was uniform from 0 and 1.

The performance of the heuristics on these 12 data sets is given in Tables 1 through 3. Because optimal solutions are difficult to obtain we compared the values of the solutions we computed using the heuristics to the lower bound. In particular we report on the ratio of the greedy heuristic's solution value to the lower bound (GR/LB), the ratio of the knapsack heuristic's solution value to the lower bound (KP/LB), the ratio of the generalized assignment heuristic's solution value to the lower bound (GA/LB) and also the ratio of the generalized assignment heuristic's value to that of the knapsack heuristic's. The statistics reported are the mean of these ratios for 20 problems as well as the standard deviation, minimum and maximum.

Several trends are quite obvious. First the knapsack heuristic is superior to the greedy heuristic. Second, on average the generalized assignment heuristic did better than the knapsack heuristic although this is not the case on all problems. Since the computational effort required to use the knapsack heuristic is not substantially more than for the greedy heuristic, the knapsack heuristic is preferred. Since the generalized assignment heuristic took substantially longer to run (see Table 4) the minor improvement may not be worth the effort. Third, the performance of the heuristics did not degrade as the size of the problem increased and appears to improve slightly on the largest of the problems tested. In particular, for a given problem type, i.e. job volumes either 1–10, 1–25, 1–50 or 13–38, the average performance remained the same as the number of families and the number of jobs within a family increased.

Volumes of jobs		GR/LB	KP/LB	GA/LB	GA/KP
1–10	Mean	1.03	1.02	1.02	1
	Std. dev.	0.01	0	0	0
	Min.	1.01	1.01	1.01	1
	Max.	1.05	1.03	1.03	1
1–25	Mean	1.08	1.05	1.08	1.02
	Std. dev.	0.03	0.01	0.03	0.02
	Min.	1.03	1.02	1.01	0.98
	Max.	1.13	1.08	1.12	1.06

1–50	Mean	1.18	1.14	1.12	0.98
	Std. dev.	0.04	0.04	0.03	0.02
	Min.	1.09	1.09	1.06	0.92
	Max.	1.24	1.23	1.18	1.03
13–38	Mean	1.21	1.16	1.15	0.99
	Std. dev.	0.05	0.05	0.03	0.04
	Min.	1.13	1.08	1.1	0.91
	Max.	1.29	1.26	1.19	1.04

Table 1: Results for problems with 3 families and 5 jobs per family.

Volumes of jobs		GR/LB	KP/LB	GA/LB	GA/KP
1–10	Mean	1.02	1.01	1.03	1.01
	Std. dev.	0.01	0	0.02	0.01
	Min.	1.01	1.01	1.01	1
	Max.	1.05	1.03	1.08	1.05
1–25	Mean	1.07	1.04	1.09	1.04
	Std. dev.	0.02	0.01	0.02	0.01
	Min.	1.05	1.02	1.06	1.01
	Max.	1.1	1.06	1.11	1.06
1–50	Mean	1.18	1.14	1.12	0.99
	Std. dev.	0.03	0.02	0.02	0.02
	Min.	1.12	1.1	1.08	0.96
	Max.	1.24	1.12	1.17	1.02
13–38	Mean	1.22	1.16	1.15	0.99
	Std. dev.	0.04	0.03	0.02	0.02
	Min.	1.17	1.11	1.11	0.95
	Max.	1.29	1.21	1.22	1.02

Table 2: Results for problems with 5 families and 10 jobs per family.

Volumes of jobs		GR/LB	KP/LB	GA/LB	GA/KP
1–10	Mean	1.02	1.01		
	Std. dev.	0	0		
	Min.	1.01	1		
	Max.	1.02	1.01		
1–25	Mean	1.06	1.03		
	Std. dev.	0.01	0		
	Min.	1.05	1.02		
	Max.	1.08	1.03		
1–50	Mean	1.14	1.11		
	Std. dev.	0.01	0.01		
	Min.	1.13	1.09		
	Max.	1.15	1.13		
13–38	Mean	1.17	1.12		
	Std. dev.	0.01	0.01		
	Min.	1.15	1.1		
	Max.	1.2	1.14		

Table 3: Results for problems with 10 families and 50 jobs per family.

6 NP-completeness and Worst-case Analysis of a Simple Heuristic.

In this section we begin by showing that the batch loading problem is NP–hard. We then give a worst-case analysis of the greedy heuristic. We actually analyze a heuristic which differs from the greedy slightly and then show that its performance is worse than the greedy's. This heuristic gives a solution whose objective value differs from the optimal by at most a factor of 2. Finally we give an example that shows that the bound is tight. We believe that a tighter bound is possible if the problem under consideration is one where the volume of the items is such that $\frac{v_i}{V}$ is small say less than $\frac{1}{k}$. In this case, we conjecture that the bound is $\frac{k+4}{k+1}$ and in the appendix give a set of examples which have this behavior for every $k \geq 2$.

6.1 The Batch Loading Problem is NP-hard.

The decision problem associated with the Batch Loading problem is, given a set of jobs J with weights c_j, and volumes v_j that are organized into families with batch processing times, t_k, on a batch processor with capacity V, does there exist a schedule with the weighted flow time for all jobs less than K.

Proposition. *The Batch Loading Decision problem is an NP-complete problem. Thus, the associated optimization problem is NP-hard.*

Proof. The reduction is from the NP-complete problem Partition. Partition can be stated as, given a set S of objects with sizes $\{s_i\}_{i \in S}$ does there exist a subset U such that

$$\sum_{i \in U} s_i = \sum_{i \in S-U} s_i = \frac{1}{2} \sum_{i \in S} s_i$$

The transformation is to let $V = \frac{1}{2} \sum_{i \in S} s_i$, and for each object i in S, create a job with $c_i = s_i$ and $v_i = s_i$. All jobs belong to the same family and the processing time for a batch of jobs of this family is 1. The claim is that a partition exists if and only if the optimal objective value for the batch loading problem is $3V$. Suppose a partition exists. Let all jobs corresponding to elements in U be in batch 1 and the remainder in batch 2. This is a feasible solution with objective value $3V$. We shall also show that $3V$ is a lower bound to this batch loading problem. This is the value of the partial LP relaxation which was described earlier in the paper. The volume of the entire set of items is $2V$, the items all have the same c_i/v_i ratio, namely 1, and so one can place half the volume in batch 1 and the other half in batch 2. This gives a weighted flow time of $3V$. Thus the existence of a partition implies that the solution to the batch loading problem is $3V$. Now suppose there exists a feasible solution (without jobs being split across batches) to this batch loading problem with objective value $3V$. Note it cannot be lower since we just observed that $3V$ is a lower bound. Let α_i be the cost (and volume) of batch i. We have that $0 < \alpha_i \le V$ and $\sum_i \alpha_i = 2V$ and that $\sum_i i\alpha_i = 3V$. We merely need to show that all the items are in the first two batches; $\alpha_1 = \alpha_2 = V$ $\alpha_i = 0$ $i \ge 3$. Let β be the cost (or volume) of all batches past the second, $\beta \equiv \sum_{i\ge 3} \alpha_i$. The objective value tells us that $3V = \sum_i i\alpha_i \ge 1\ \alpha_1 + 2\alpha_2 + 3\beta = \alpha_1 + 2\alpha_2 + 3(2V - \alpha_1 - \alpha_2) = 6V - 2\alpha_1 - 1\alpha_2$ or rearranging we have $2\alpha_1 + \alpha_2 \ge 3V$. Since $0 < \alpha_i \le V$, the only feasible solution is $\alpha_1 = V$, $\alpha_2 = V$ and $\alpha_i = 0$ for $i \ge 3$. This provides us with a 2–batch solution which gives a solution to the partitioning problem.■

6.2 Worst–case Analysis of a Simple Heuristic

In this subsection we analyze the performance of simple heuristic, we shall denote by H, for the batch loading problem and compare that to the value of the partial LP relaxation of the problem. The heuristic works as follows. Take the solution to the partial LP relaxation and construct a feasible solution by creating two batches for each batch in the relaxation. In particular, the batch in position l in the relaxation and the batches in positions (2l-1) and 2l in the heuristic shall belong to the same family.

Consider the jobs in the batch assigned to processing position l in the solution to the relaxation. They will be assigned to the batches in position $2l - 1$ and $2l$ in the heuristic. Figure 3 shows the procedure associated with this heuristic.

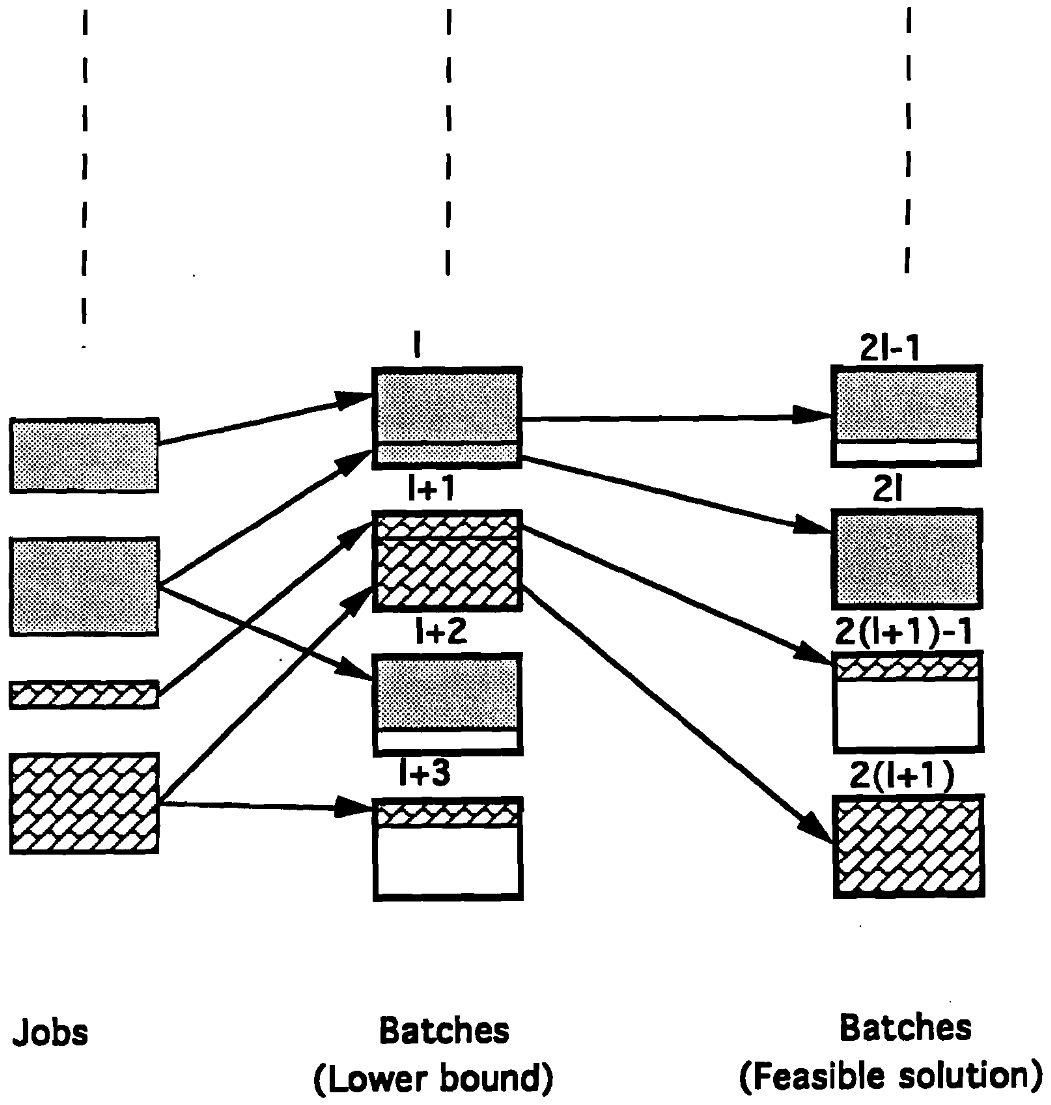

Figure 3 Construction of heuristic for worst case analysis

We will use superscripts R and H to denote values associated with the relaxation and heuristic solutions respectively. For example, f_l^R and f_l^H will denote the flow times of the batch at position in the relaxation and the heuristic, respectively. Define t^l to be the processing time of the batches in the relaxation. Thus the flow time for batch l is $f_l^R = \sum_{k=1}^{l} t^k$. In the heuristic solution the flow time for the two batches at positions $2l-1$ and $2l$ will be $f_{2l-1}^H = 2\sum_{k=1}^{l-1} t^k + t^l$ and $f_{2l}^H = 2\sum_{k=1}^{l-1} t^k + 2t^l$. Thus it is easy to see that $f_{2l-1}^H = 2f_l^R - t^l \leq 2f_l^R$ and $f_{2l}^H = 2f_l^R$. To generate a feasible solution we need to assign jobs in the batch at position l in the relaxation to batches at position $2l-1$ and $2l$. This is straight forward. If job i is entirely in the batch, place it at position $2l-1$. If the job i is only partially in the batch and partially in a later batch then assign it to position $2l$. It is easy to see that if J_l^R is the set of jobs which first appear in the batch at position l in the relaxation and this is partitioned into two sets, J_{2l-1}^H and J_{2l}^H for the jobs placed into positions $2l-1$ and $2l$ in the heuristic solution then

$$f_{2l-1}^H\left(\sum_{j\in J_{2l-1}^H} c_j\right)+f_{2l}^H\left(\sum_{j\in J_{2l}^H} c_j\right)\leq 2f_l^R\left(\sum_{j\in J_l^R} c_j\right).$$

Denote by z^H and z^R the values of the heuristic and the relaxation respectively. Summing over all processing position used in the lower bound yields $z^H \leq 2z^R$ and since $z^R \leq z^*$ we have the desired worst–case bound of 2 for this heuristic: $\frac{z^H}{z^*} \leq 2$.

We now wish to argue that the greedy heuristic will always produce a better solution than the heuristic H described above. If we let z^G denote the value of the greedy solution then we need that $z^G \leq z^H$. To see this we introduce a third value, denoted $z_{G\text{ order}}^{H\text{ positions}}$. This is the solution value one would obtain by assigning items to batches using the greedy heuristic and then using the assignment of batches to positions given by the heuristic H. Since the greedy heuristic re–sorts the batches at the end we conclude that $z^G \leq z_{G\ \text{order}}^{H\ \text{positions}}$. Furthermore since the greedy might place items into earlier batches if they had sufficient capacity we conclude $z_{G\ \text{order}}^{H\ \text{positions}} \leq z^H$. Thus $\frac{z^G}{z^*} \leq \frac{z^H}{z^*} \leq 2$. ■

6.3 Example where the bound is tight.

The example below shows that asymptotically the bound is tight for the greedy heuristic. Consider the following problem. There are n families. The capacity of the processor, V, is 1. The processing time for each family, t_j, is 1. Each family consists of three jobs. The data is given in the table below. Let $0 < \varepsilon, \delta < 1$

job	cost(c_i)	volume(v_i)	ratio($\frac{c_i}{v_i}$)
1	1/2	1/2	1
2	$\delta(1-\varepsilon)$	δ	$1-\varepsilon$
3	$1/2(1-2\varepsilon)$	1/2	$1-2\varepsilon$

It is clear that any feasible solution will require 2 batches per family since the volume of jobs for each family is $1 + \delta$. The optimal solution is to place jobs 1 and 3 in one batch and job 2 in a second batch, and then to sequence the batches by placing all the full batches first and all the nearly empty batches next. The objective value is

$$\sum_i c_i f_i = \sum_{i=1}^{n} (1-2\varepsilon)i + \sum_{i=1}^{n} \delta(1-\varepsilon)(n+i)$$

$$= (1-2\varepsilon)\frac{n(n+1)}{2} + \delta(1-\varepsilon)\left(n^2 + \frac{n(n+1)}{2}\right)$$

If we make δ and ε small this converges to $\frac{1}{2}(n)\ (n+1)$. The heuristic analyzed (as well as the greedy heuristic) would produce a solution with jobs 1 and 2 in the first batch for each family and job 3 in the second batch. This will have an objective value of

$$\sum_i c_i f_i = \sum_{i=1}^{n}\left(\frac{1}{2}+\delta(1-\varepsilon)\right)i + \sum_{i=1}^{n}(\tfrac{1}{2})(1-2\varepsilon)(n+i)$$

$$= \left(\frac{1}{2}+\delta(1-\varepsilon)\right)\frac{n(n+1)}{2} + \left(\frac{1}{2}\right)\left(1-2\varepsilon\right)\left(n^2+\frac{n(n+1)}{2}\right)$$

As we make δ and ε small this value converges to

$$\frac{n(n+1)}{4}+\frac{n(n+1)}{4}+\frac{n^2}{2}=\frac{2n^2+n}{2}$$

Taking the ratio we see that

$$\frac{z^H}{z^*}=\frac{\frac{2n^2+n}{2}}{\frac{n^2+n}{2}}=\frac{2n^2+n}{n^2+n}\rightarrow 2 \text{ as } n\rightarrow\infty.$$

Thus as the number of families grows the error increases to 2.

7. Summary

As we have seen this problem can be thought of making two types of decisions. First there is the assignment of jobs to batches and second, there is the sequencing of batches. The later is easy given the former and the former relatively easy given the later; it's a generalized assignment problem. Given this structure there are many ways to formulate the problem rather than the "assignment" style variables we have used here. We investigated several possibilities with the hope of finding a formulation where Lagrangian relaxation would yield better bounds and heuristics based on the Lagrangian multipliers. Unfortunately these formulations all resulted in relaxations with an enormous number of multipliers and thus were not solvable for practically sized problems.

The difficulty in this problem arises because of underlying generalized assignment problem, i.e., the assignment of jobs to batches. To be more precise it is the "knapsack" nature of this subproblem that makes it hard. If all the jobs had unit volume then the problem could be solved by a sorting procedure. Thus, in problems where the size of the items is relatively small compared to the capacity of the processor the heuristics do quite well.

As for future work, the most obvious limitation of the model is the lack of due dates. Thus an important extension would be to extend, presumably by the standard Lagrangian techniques, the problem to include release dates, due dates and tardiness penalties for the jobs. Another aspect worth considering is how scheduling is affected when the batch processor is part of a flow shop and the other machines have a more standard processing time that is the sum of setup and per part run time.

Appendix

In section 6, we give a proof that the worst case performance of the greedy heuristic has a ratio $z^G/z^* \leq 2$ and a worst case example for the greedy with a ratio z^G/z^* that approaches 2. We conjecture that if the items are all small then the bound is tighter. In particular for a given integer $k \geq 2$, if all items have volumes less than or equal to $\frac{1}{k}$, then we conjecture that the worst case behavior of the greedy is $\frac{k+4}{k+1}$. For $k = 2$ this would give the bound of 2; for $k = 3$ the bound would be $\frac{7}{4}$; for $k = 4$ the bound would be $\frac{8}{5}$. The example below shows that the greedy can do at least this badly when restricted to problem instances where all volumes are at most $\frac{1}{k}$.

In this example, there are n families. Let $k \geq 2$. Each family has $k + 1$ items. Their volumes and weights are given in table A.1.

number of items	volume	cost	ratio
$k - 1$	$\frac{1}{k}$	$(1 + \delta)w_i$	$(1+\delta)w_ik$
1	ε	εw_ik	w_ik
1	$\frac{1}{k}$	$(1 - \delta)w_i$	$(1 - \delta)w_ik$

Table A.1: Data of items in family *i*.

Think of ε and δ as small. The factor w_i for family i is

$$w_i \equiv \frac{n(k-1) - 1 - i(k-2)}{n - 1}.$$

Observe that the $\{w_i\}$ are decreasing since $w_i - w_{i+1} = \frac{k-2}{n-1} \geq 0$. The processing time for the families are identical, say $t^k = 1$. In the optimal solution all the items of size $\frac{1}{k}$ would be processed together in a batch and the remaining item of volume ε would appear in a batch by itself. The greedy on the other hand would place the first k items in a batch and the remaining item of volume

$\frac{1}{k}$ would have to be processed in another batch. For both solutions the first batches (for each family) would be sequenced in decreasing order of the w_i followed by the second batches again in decreasing order of the w_i. To see this for the greedy we must check that $\frac{k-1}{k}\frac{w_n}{t_n} \geq \frac{1}{k}\frac{w_1}{t_1}$. These terms are in fact both equal to $\frac{k-1}{k}$.

The optimal value, z^*, once we let ε and δ go to 0 is $\sum_{i=1}^{n} iw_i$.

$$\begin{aligned}
z^* &= \sum_{i=1}^{n} \frac{n(k-1) - 1 - (k-2)i}{(n-1)} i \\
&= \left(\frac{n(k-1)-1}{n-1}\right)\frac{(n)(n+1)}{2} - \left(\frac{k-2}{n-1}\right)\left(\frac{n(n+1)(2n+1)}{6}\right) \\
&= \frac{n(n+1)}{2(n-1)}\left[n(k-1) - 1 - \frac{k-2}{3}(2n+1)\right] \\
&= \frac{n(n+1)}{2(n-1)}\left[\frac{nk + n - 1 - k}{3}\right] \\
&= \frac{n(n+1)(k+1)}{6}
\end{aligned}$$

On the other hand the greedy produces a solution whose value is, z^G, once we let ε and δ go to 0.

$$\begin{aligned}
z^G &= \sum_{i=1}^{n}\left(\frac{k-1}{k}\right)w_i i + \sum_{i=1}^{n}(n+i)(\frac{1}{k})w_i \\
&= \left(\frac{k-1}{k}\right)z^* + \frac{1}{k}z^* + \frac{n}{k}\sum_{i=1}^{n} w_i \\
&= z^* + \frac{n}{k}\sum_{i=1}^{n} w_i
\end{aligned}$$

Now,

$$
\begin{aligned}
\sum_{i=1}^{n} w_i &= \sum_{i=1}^{n} \frac{n(k-1) - 1 - i(k-2)}{n-1} \\
&= \frac{n(k-1)-1}{(n-1)} n - \frac{k-2}{n-1}\left(\frac{n(n+1)}{2}\right) \\
&= \frac{n}{2(n-1)}\left[2nk - 2n - 2 - kn + 2n - k + 2\right] \\
&= \frac{nk}{2}
\end{aligned}
$$

Thus,

$$
\begin{aligned}
\frac{z^G}{z^*} &= 1 + \frac{\frac{1}{2}n^2}{z^*} \\
&= 1 + \frac{n^2}{2} \frac{6}{n(n+1)(k+1)} \\
&= 1 + \frac{3n}{(n+1)(k+1)} \\
&\rightarrow \frac{k+4}{k+1} \text{ as } n \rightarrow \infty
\end{aligned}
$$

Acknowledgments

The authors wish to thank Uday S. Karmarkar for introducing us to this problem and John. W. Fowler for his helpful comments.

References

Ahmadi, J.H., Ahmadi, R.H., Dasu, Sriram and Tang, Christopher S., (1989). Batching and Scheduling Jobs on Batch and Discrete Processors, *Operations Research,* **40,** 750-763.

Dobson, G., Karmarkar, U.S., and Rummel, J.L.,(1987). Batching to Minimize Flow Times on One Machine, *Management Science,* **33,** 784-799.

Dobson, G., and Karmarkar, U.S.,(1986). Large Scale Shop Scheduling Formulations and Decompositions, Working paper QM 86–31.

Fisher, M.L., Jaikumar, R., and Wassenhove, L.N.,(1986). A Multiplier Adjustment Procedure for the Generalized Assignment Problem, *Management Science*, **32**, 1095–1103.

Fisher, M.L.,(1981). The Lagrangian Relaxation Method for Solving Integer Programming Problems, *Management Science*, **27**, 1–18.

Fowler, J. W., Phillips, D. T., and Hogg, G. L. (1991). Strategic Control of Multiproduct Bulk Service Diffusion/Oxidation Processes, Sematech Working Paper INEN/MS/WP/03/3-91

Glassey, C. R., and Weng,(1991) W. W., Dynamic Batching Heuristic for Simultaneous Processing, *IEEE Transactions on Semiconductor Manufacturing*, **4**, 77-82

Ikura, Y., and Gimple, M., (1986) Scheduling Algorithms for a Single Batch Processing Machine, *Operations Research Letters*, **5**, 61-65.

Lee, C. Y., Uzsoy, R., and Martin-Vega, L. A. (1992) Efficient Algorithms for Scheduling Semiconductor Burn-in Operations, *Operations Research*, **40**, 764-775.

Lefrancois, P., P. L'Esperance and M. Turmel (1991) "Batching Annealing Operations to Optimize Queueing Times and Furnace Efficiency: A Simulation Model," Working Paper 91-22, Department Operations et systemes de decision, Universite Laval, Quebec, Canada

Nemhauser, G.L., and Wolsey,(1988). L.A., *Integer and Combinatorial Optimization*, John Wiley.

Santos, C. and Magazine, M. (1985) Batching in Single Operation Manufacturing Systems, *Operations Research Letters*, **4**, October, 99-103.