INT. J. PROD. RES., 2001, VOL. 39, NO. 14, 3085–3107

Sequence-dependent batch chemical scheduling with earliness and tardiness penalties

KENNETH E. McGRAW† and MAGED M. DESSOUKY†*

Production volume in the specialized agricultural chemical industry is typically too small to justify the capital expenditure required for continuous processing. As such, there is a trend towards building chemical processing plants for this market segment that are batch plants. Scheduling of this type of chemical plant under just-in-time operations, where both earliness and lateness penalties are included, is critical to the efficient operation of these plants. In this paper, a heuristic scheduling procedure is developed for the problem of minimizing the total weighted earliness and tardiness costs as well as the total set-up cost for the single-stage batch chemical manufacturing environment.

1. Introduction

Continuous processing plants are typically the preferred method of processing for producing large volumes of chemicals. When a large product variety exists and small amounts of a chemical are demanded, the batch mode of production is usually the preferred method because of the large capital costs associated with a continuous plant. One such example where the production volume does not justify the continuous mode of operations is agricultural chemicals, such as customized fertilizers, specialized herbicides and pesticides.

The trend in the chemical process industry to operate in batches parallels the shift to small lot-sizes in discrete-parts manufacturing. However, there are several distinguishing features between batch chemical processing and discrete-parts manufacturing. Since the production entity is continuous in nature and the batch size may be larger than the customer order quantities in order to make efficient use of the resources, a batch may be divisible in fractions. This leads to potentially numerous ways of allocating a batch to different customer orders. For example, a fraction of a chemical batch may be used to satisfy a given demand, with the remainder used to meet—or partially meet—the needs of one or more other demands. For this reason, a particular batch may satisfy the demand of more than one customer order (Musier and Evans 1991). As such, any given batch may be considered as having multiple due dates, one for each order that the batch satisfies. Another advantage of batch chemical plants is the ability to process multiple products by sharing the same process resources. Both the multiple demands aspect and the ability to share resources to produce multiple products complicate planning and scheduling of batch plants operations.

Revision received March 2001.

† Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, CA 90089, USA.

* To whom correspondence should be addressed. maged@rcf.usc.edu

International Journal of Production Research ISSN 0020–7543 print/ISSN 1366–588X online © 2001 Taylor & Francis Ltd
http://www.tandf.co.uk/journals
DOI: 10.1080/00207540110056180

The batch chemical scheduling problem discussed in this paper is similar to the single-machine weighted earliness and tardiness scheduling problem where all batches (jobs) have identical processing times and each batch may have multiple due dates (demand time points) with sequence-dependent set-up times. The focus of this work is to consider the trade-off between the earliness and tardiness penalties in a just-in-time manufacturing environment, which dictates that completing jobs earlier than their due dates should be discouraged as should be completing jobs later than their due dates. The main distinction between problems with tardiness penalties only and those with both earliness and tardiness penalties, is that in the latter problem it may be necessary to insert idle time before the start of processing of a job to avoid earliness costs, while in the former, inserted idle time will only expose the job to tardiness penalties without any savings in earliness penalties.

Although there has been a significant body of work on batch chemical scheduling with varying objectives, such as makespan and tardiness (e.g. Pekny *et al.* 1990, Tandon *et al.* 1991, Ku and Karimi 1990, 1991, Birewar and Grossmann 1990, Patsidou and Kantor 1991, Kondili *et al.* 1993, Kudva *et al.* 1994, Dessouky *et al.* 1996, Dessouky and Kijowski 1997), there has been very little work in batch chemical scheduling where both earliness and lateness penalties are taken into account. It should be pointed out that, in the discrete-parts literature, there is some work in scheduling with both earliness and tardiness penalties. Baker and Scudder (1990) provide an excellent review of the single-machine scheduling problem with earliness and tardiness penalties. For non-identical processing times, the single-machine weighted tardiness scheduling problem is known to be NP-complete (Lenstra *et al.* 1977).

For identical processing times, the single-machine weighted tardiness scheduling problem may be formulated as an assignment problem with no inserted idle time (Lawler 1964). With earliness penalties it may be optimal to insert idle time. Garey *et al.* (1988) developed a polynomial-time algorithm that determines the optimal schedule for identical processing times with the weights of the tardiness and earliness of all jobs being equal. Hall and Posner (1991) developed another polynomial-time algorithm for the identical processing times case with each job having a different weight with symmetric tardiness and earliness penalties. They also assume all jobs have a common due date. Verma and Dessouky (1998) show that the problem with each job having a distinct due date is also polynomial solvable.

Although our problem considers the case of fixed batch sizes, there has also been some work in the area of variable batch sizes. This work sets the size of the batch to a maximum size; within this maximum batch size limit, the actual batch size is adjusted to achieve an improved schedule. Brucker (1995) develops two dynamic programming solution procedures for the variable batch size problem considering only the job lateness. Crauwels *et al.* (1997) describe several procedures that partition jobs into batches in order to minimize the total weighted completion time where set-up times are sequence independent. Four solution methods are described, including neighbourhood search, simulated annealing, threshold accepting, and tabu search. Their problem formulation does not address earliness penalties. Hariri and Potts (1997) present an algorithm that reduces the problem size by employing a branching rule that starts with a single batch solution and, at each node, performs a test to determine if pairs of jobs can be combined into a single batch. The considered objective function minimizes the maximum lateness. Uzsoy (1994) develops an algorithm for scheduling non-identical size jobs on a single batch machine by grouping

jobs with similar processing times together. A branch-and-bound procedure is then used for solving the problem. The algorithm minimizes the makespan and the total completion time; earliness penalties and sequence-dependent set-up costs are not considered. Uzsoy (1995) extends his earlier work by showing that by decoupling batch formation and batch sequencing decisions, the static batch processing machine scheduling problems can be reduced to equivalent scheduling problems with unit capacity machines. Chen (1996) develops an algorithm that addresses both earliness and tardiness penalties, but it is assumed that all earliness penalties are identical, all tardiness penalties are identical, and that there is a common due date. Finally, Monma and Potts (1989) present an overview of single and parallel machine scheduling problems with batch set-ups. The objectives that were considered are maximum lateness, total weighted completion time, and number of late jobs. Earliness penalties are not considered. This paper differs from the earlier work by considering fixed batches with general earliness and tardiness penalties.

Dessouky *et al.* (1999) developed an iterative heuristic solution procedure for the joint earliness and tardiness problem in a multiple due-date environment, but assumed sequence-independent set-ups. Again, there is little work that considers both earliness and tardiness costs with sequence-dependent set-up costs (Allahverdi *et al.* 1999).

In this paper, we consider the single-machine weighted earliness and tardiness scheduling problem where all batches (jobs) have identical processing times and each batch can have multiple due dates with sequence dependent set-ups. No assumption is made on the structure of the earliness or tardiness weights. The problem is formulated as a mixed integer program that can only be solved optimally for small problem sizes. In order to solve large instances of the problem, a heuristic is developed. The heuristic is first developed for the single-product case in order to highlight the problem structure and then we later expand it for the general multi-product case.

2. Problem statement

In this section, the problem is formulated in its broader context by including multiple products and sequence dependent set-up costs. Once completed, simplifying assumptions are described. This simplified form of the problem will then serve as the foundation upon which the heuristic for solving the problem is based.

Although the problem, as developed in this section, is rather broad, there are still certain assumptions that are made. These assumptions include scheduling a single stage process and identical processing time for all batches. In chemical processing, there is typically a stage that dictates the maximum processing rate of the line and limits the overall plant's rate of production. For scheduling purposes, it is common practice to assume that the model can be reduced to one of scheduling a single processing unit, which is the bottleneck machine. This single processing unit scheduling method has been used for batch plant scheduling that considered multiple products being produced in the manufacture of latex (Heuts *et al.* 1992). An argument for the identical processing time assumption is that, for a given set of parameters such as heat and pressure, chemical reaction rates are relatively constant, and any set-up or intermediate human operations are repeatable and thus take place in a quite predictable period of time. In our pesticide application, the identical processing time was eight hours. Hence, each shift produced a single batch. Without loss of generality it is assumed that the identical processing time is unity. The problem is

formulated as a mixed integer program where the following terminology is used throughout this paper:

Parameters

$k_{\max}$	maximum number of product types,
m_k	number of orders of product type k $(k = 1, \ldots, k_{\max})$,
α_{jk}	earliness penalty of order $j = 1, \ldots, m_k$ for product k $(k = 1, \ldots, k_{\max})$,
β_{jk}	tardiness penalty of order $j = 1, \ldots, m_k$ for product k $(k = 1, \ldots, k_{\max})$,
u_{jk}	demand of order $j = 1, \ldots, m_k$ for product k $(k = 1, \ldots, k_{\max})$,
d_{jk}	due date for order $j = 1, \ldots, m_k$ of product k $(k = 1, \ldots, k_{\max})$,
S_{k_1,k_2}	set-up cost of product type k_2 when preceded by product k_1 $(k_1 = 1, \ldots, k_{\max})$, $(k_2 = 1, \ldots, k_{\max})$. Note: $S_{k_1,k_2} = 0$ if $k_1 = k_2$.

Without loss of generality, the production batch size is normalized to unity. Demands, u_{jk} are expressed in units of batch size, thus a demand equal to a batch size would be 1.0. Demands that are smaller than, or larger than, a batch would then be less than unity or greater than unity, respectively.

Assuming that all processing occurs in one shift means that any set-up or maintenance activities are assumed to take place during overtime. Set-up costs are those costs associated with preparing the chemical reactor for production of the desired product and any additional overtime cost due to set-up. This cost is primarily labour costs in that the cost of the reactants and any cleansing of the reactor once production is complete are assumed to be independent of the actual period in which the product is produced.

Penalty costs are those associated with early and late production of the product. Penalties associated with late production can take many forms, including such things as contractual penalties, which compensate the customer for losses due to missing markets, decreased production from late application of fertilizers or actual crop loss due to late application of pesticides. Although losses resulting from late fertilizer or pesticide delivery are actually borne by the farmer, they do represent a real penalty cost for the chemical plant in the form of loss of customer base. Earliness penalties can take the form of added costs incurred from spoilage or product shelf life due to such causes as decomposition of the chemicals, and increased inventory and warehouse costs caused by the need to store the product until it is actually needed.

The decision variables can be summarized as follows.

$$x_{ijk} = \text{fraction of product } k \ (k = 1, \ldots, k_{\max} \text{ batch produced in period } i \ (i = 1, \ldots, n) \text{ used to satisfy order } j \ (j = 1, \ldots, m_k),$$

$$y_{ik} = \begin{cases} 1 & \text{if a batch of product } k \text{ is produced in period } i \\ 0 & \text{otherwise,} \end{cases}$$

$$Z_{k_1,k_2,i} = \begin{cases} 1 & \text{if product type } k_1 \text{ precedes type } k_2 \text{ in period } i \\ 0 & \text{otherwise.} \end{cases}$$

Let c_{ijk} be the penalty incurred when allocating the jth order of product k to a batch produced in period i. Then,

$$c_{ijk} = \max(\alpha_{jk}(i - d_{jk}), \beta_{jk}(d_{jk} - i)).$$

The range of n is easily determined; first define N as the total number of batches needed to meet demand.

$$N = \sum_{k=1}^{k_{\max}} \left\lceil \sum_{j=1}^{m_k} u_{jk} \right\rceil .$$

Then it is easy to prove that there exists an optimal schedule that has all batches completed by time n:

$$n = \max_{\substack{j=1,\ldots,m_k \\ k=1,\ldots,k_{\max}}} (d_{jk}) + N.$$

The studied problem (referred to as Problem P) may be formally stated as follows.

Problem P

$$\min \sum_{k=1}^{k_{\max}} \sum_{i=1}^{n} \sum_{j=1}^{m_k} c_{ijk} x_{ijk} + \sum_{i=1}^{n} \sum_{k_i=1}^{k_{\max}} \sum_{k_2=1}^{k_{\max}} s_{k_1,k_2} z_{k_1,k_2,i}$$

subject to:

$$\sum_{j=1}^{m_k} x_{ijk} = y_{ik} \ (i = 1, \ldots, n), (k = 1, \ldots, k_{\max}) \tag{1}$$

$$\sum_{i=1}^{n} x_{ijk} = u_{jk} \ (j = 1, \ldots, m), (k = 1, \ldots, k_{\max}) \tag{2}$$

$$\sum_{k=1}^{k_{\max}} y_{ik} \leq 1, (i = 1, \ldots, n) \tag{3}$$

$$z_{k_1,k_2,i} \geq y_{i,k_2} - (1 - y_{i-1,k_1}) * M \ (i = 1, \ldots, n) \tag{4}$$

$$k_1, \ldots, k_{\max}, k_2 = 1, \ldots, k_{\max}$$

$$M = \text{large value}$$

$$x_{ijk} \geq 0; \ y_{ik} \in (0, 1); \ z_{k_1,k_2,i} \in \{0, 1\}.$$

Constraint (1) ensures that either an entire batch or no batch is produced in period i. Constraint (2) ensures that demand is met. Constraint (3) ensures that, at most, one batch is produced in each period. Constraint (4) defines the precedence variable.

3. Single product case

Problem P addresses the general multi-product scheduling case with sequence-dependent set-up costs. Now, consider only the single product case. This will allow us to gain insight into the problem and will lead to a heuristic to solve more easily not only this simplified problem but also the more general problem. Refer to the single product case as Problem P1. For problem P there are two decisions that must be made simultaneously to ensure minimal operational cost. They are (1) the allocation of orders to batches and (2) the schedule of batches. Note that for the single product formulation the subscript k can be dropped and constraints (3) and (4) are no longer necessary.

With the single-product formulation, it is easy to see: (1) if the batch schedule is given (i.e. the busy production periods) the problem reduces to a transportation problem and (2) if the batch allocation is given (i.e. the allocation of production batches to customer orders), the problem reduces to an assignment problem. These observations suggest a solution method for problem P1 based on being able to predetermine the busy production periods and solving a transportation problem. That is, we determine the production periods in which a batch will be produced based on a heuristic rule. Then, given this set of busy production periods, the allocation of the production batches to customer orders is determined by solving a transportation problem. We note that the above observations also hold for the multiple product case when there is zero set-up cost.

When trying to find the minimal transportation problem that will solve problem P1, it is useful to look at the dual form of the problem. Taking the derivative of the dual objective with respect to the dual variabkles, we get

$$\sum_{i=1}^{n} y_i - \sum_{j=1}^{m} u_j = 0,$$

where $y_i = 1$ if a batch is produced in period i, else 0, and u_j is the demand size in period j.

From this, it is possible to examine how to assign a production batch for any given customer order demand. The ideal case would be one in which the size of each customer's order is equal to the size of a batch. Assume that there is a single demand equal to a batch size that has a due date at period i. Now recall that $c_{ij} = 0$ when $i = j$, and this value of zero would be the best possible value for the objective function. From this it can be seen that the best time to produce a batch is at period i. In other words, for the ideal case of a single demand of the size of a batch, a batch must be produced on the demand's due date in order to achieve the minimum penalty cost.

In the real world, the size of the customer's orders is almost never exactly equal to the size of a batch. In addition, multiple orders can occur for any particular demand period.

Our solution approach for approximating the optimal set of busy periods is to assign the demand in order of increasing size to as close to the ideal production period as possible. We now formally present this concept. The first part of the algorithm determines the set of busy production periods. Step 7 then solves the resulting transportation problem that determines the best demand to busy period allocation.

Single product heuristic

Step 1. Determine the required number of batches to produce:

$$N = \sum_{j=1}^{m} \lceil u_i \rceil.$$

Step 2. Sort the demands in order of decreasing size.
Step 3. Select an unassigned order with the largest demand along with its due date.
Step 4. Assign a batch to a production period that results in the smallest penalty cost.

- If the production period coinciding with the demand's due date is available, set the period to be busy.
- If the production period coinciding with the demand's due date is not available:

(a) determine the latest available production period that is earlier than the demand's due date. Calculate its earliness penalty cost;
(b) determine the earliest available production period that is later than the demand's due date. Calculate its lateness penalty;
(c) if earliness penalty $<$ lateness penalty, assign a batch to be produced at the production period determined in step (a). Else assign a batch to be produced at the production period determined in step (b). Ties are arbitrarily broken.

Step 5. Decrease the size of the current order by the amount of the batch size (unity in this case).
Step 6. If the number of batches assigned is $< N$, go to step 3. Else go to step 7.
Step 7. Given the set of busy periods, solve the transportation problem.

The heuristic is polynomial with computational complexity equal to solving a transportation problem with $2N$ nodes.

3.1. *Computational experiments*

The Single Product Heuristic was tested to determine its performance. This testing consisted of running trials with a total demand of 10, 20, 40 and 80 batches. To ensure that the testing was generic, but still representative of the real problem, random numbers were used for the problem's parameters. Penalties were generated from a continuous uniform random number from (1,10] where the lower limit of 1 ensures that all orders will have a penalty. Demands, in terms of batch size, were generated from a continuous uniform random number from $(0, U_{\max}]$, with $U_{\max}$ being either 1 or 5, while their associated due dates were generated from a discrete uniform random number from $(0, D_{\max}]$. Both tight due dates and scattered due dates cases were tested by setting $D_{\max}$ equal to N and $2N$ respectively. A small $U_{\max}$ indicates a situation where there are many orders of small size relative to the batch size. In this case, a production batch satisfies the demand of many orders. Due to the run time for the larger problems being several hours, the number of runs for each scenario was set to 12 in order to allow for the testing of more scenarios. Testing was performed using a commercially available mixed integer program (MIP) optimization software package. We used CPLEX as the MIP solver. Performance of the algorithm was measured by comparing the results found using the commercial MIP solver to those results found using the algorithm. The stopping criterion for the commercial MIP solver was when the branching tree became too large for the available computer memory or when the MIP solver found an optimal solution. When the MIP solver is able to find an optimal solution, it is an exact optimum, which is of use in benchmarking the Single Product Heuristic's accuracy. The performance parameters that were measured are R, N^*, R^* and T. R is calculated as R = (Best MIP Value)/(Algorithm Value). N^* is the number of times, out of each of the 12 trial runs, that the MIP solver was able to find an exact optimum. R^* is calculated the same as R, but it is for only those cases where the MIP solver was able to find an exact optimum. R^* thus measures the accuracy of the algorithm.

Using a commercial solver to find a solution to the problem can be viewed as another heuristic approach when the solver is not able to find the optimal solution. Therefore, the ratio R can be viewed as the comparison of our heuristic to an alternative approach to solving the problem. T is calculated as $T = (\text{MIP Solution Time})/(\text{Algorithm Solution Time})$.

Table 1 shows the initial set of results for the R and T performance measures. Note that the R values in some cases are greater than one since, in many cases, the stopping criterion of the MIP solver was met before an optimal solution was found. As the table shows, it can be seen that the Single Product Heuristic did produce a solution more quickly, but the quality of the solution was, in many cases, not as good as that produced by the MIP solver. In attempting to determine the cause of this poor performance, notice was made of the fact that the Single Product Heuristic performed better for the case of $U_{\max} = 5$. For this case, there are more instances when a given batch supplies an entire demand as opposed to the $U_{\max} = 1$ case where almost all the time a batch will supply multiple demands. Hence, for $U_{\max} = 1$, a production period may be set to busy where there is little demand resulting in a small fraction of the batch used to satisfy a zero penalty demand. The large remainder of the batch is forced to supply demands in other productions periods and will thus incur larger penalty costs. With this observation, it appears that the smaller size demands should be excluded from being used as part of the process of determining the assignment of busy production periods. To test this concept, steps 1–6 are only performed on orders with demand greater than a predefined level $U_{\min}$. Note that step 7 of the algorithm is no longer a transportation problem since a complete set of busy periods is not found in the previous steps. Hence, step 7 solves a mixed integer program problem. In addition, the heuristic's complexity now becomes exponential whereas with complete assignment it was polynomial. A smaller value of $U_{\min}$ results in more of a preassignment of busy periods and less integer variables in the mixed integer program. Conversely, a larger value of $U_{\min}$ results in fewer busy periods being preassigned and the mixed integer program becomes more difficult to solve.

N	$D_{\max}$	$U_{\max}$	R	T
10	N	1	0.584	13.132
10	N	5	0.926	11.036
10	2N	1	0.591	90.565
10	2N	5	0.897	91.799
20	N	1	0.600	848.300
20	N	5	0.940	34.593
20	2N	1	0.635	903.230
20	2N	5	0.926	1281.691
40	N	1	0.744	340.235
40	N	5	0.927	974.363
40	2N	1	0.805	257.059
40	2N	5	1.004	1105.226
80	N	1	0.769	72.039
80	N	5	1.002	306.760
80	2N	1	0.788	52.050
80	2N	5	1.082	226.781

Table 1. Initial single product case results.

Table 2 shows the R values as a function of $U_{\min}$. Note that the results for $U_{\min} = 0$ are shown in table 1. As table 2 shows, the best value of $U_{\min}$ is 0.70. Table 3 shows the complete set of results for $U_{\min} = 0.7$. The results show that the mean overall improvement of the Single Product Heuristic over the commercial MIP solver solution is 12.9% ($R = 1.129$). The most improvement is for problems that are large in size and with $U_{\max} = 1$, as this is the situation where there are many orders of small size relative to the batch size. Additionally, the Single Product Heuristic performance was better for $D_{\max} = 2N$. This improvement of $D_{\max} = 2N$ cases over $D_{\max} = N$ cases was greatest for large problems. Examining the CPU time measure, T, shows that the Single Product Heuristic typically solves the problem much faster than the MIP solver, and for those two cases where the algorithm did take slightly longer than the MIP solver, the improvement in the solution of more than 35% is very significant. Note that the reason the T ratio is small for the case of N being large is because, under this scenario, we ran out of memory faster, which is the stopping criteria. Measuring R^* shows that, for those problems where the MIP solver does find an exact optimum, the algorithm yields results that are within a few percent of exact optimum. To assess the relative quality of the algorithm, the lower bound was calculated by relaxing variables Y_{ik}. Using the lower bound, the ratios LB and LB* were calculated where LB = (LP value)/(Algorithm Value), and LB* = (LP value)/(optimal MIP value), when the MIPL finds the optimal solution. The ratio LB gives an evaluation of the quality of the algorithm while LB* provides a measure of the quality of LB. The variables LB and LB* are also included in table 3. As can be seen from the table, the quality of the lower bound is a function of $U_{\max}$. That is, the lower bound is close to the optimal solution when $U_{\max} = 5$. Under these cases, the heuristic gives results near the lower bound. Overall, the Single Product Heuristic gives improved solutions while reducing the required computation time.

The inability of the MIP solver to find optimal solutions is related to the fact that the size of the search becomes too large, and the processor runs out of available memory. Therefore, the size of the problem must be reduced. If one of the busy periods is preassigned, then the size of the problem will decrease, and of course another busy period being assigned further reduces the size of the problem even more. However, for each busy period that is assigned, another constraint is added. Thus, there is a trade-off between prefixing some of the allocations in terms of computational speed and solution accuracy. By intelligently preassigning some of the periods, the size of the problem is decreased and thus the ability to find a better solution for larger problems is enabled. As has been shown for those cases where the MIP solver is able to find the optimal solution, the value of the optimal objective functions is not greatly impacted.

In addition to using the demand size as the order ranking method in step 2 of the Single Product Heuristic, other ordering rules were considered that included the earliness and lateness penalties. An example of one such rule tested was as follows

$$(\alpha_j + \beta_j)u_j.$$

No improvements in the algorithm results were found by using these penalties in the decreasing order of the ranking.

			R values								
N	D_{max}	U_{max}	$U_{min} = 0.35$	$U_{min} = 0.40$	$U_{min} = 0.45$	$U_{min} = 0.50$	$U_{min} = 0.55$	$U_{min} = 0.60$	$U_{min} = 0.65$	$U_{min} = 0.70$	$U_{min} = 0.75$
10	N	1	0.584	0.616	0.617	0.691	0.735	0.858	0.921	0.961	0.961
10	N	5	0.926	0.930	0.939	0.949	0.961	0.975	0.986	0.986	0.986
10	2N	1	0.591	0.620	0.624	0.687	0.732	0.880	0.921	0.962	0.972
10	2N	5	0.897	0.903	0.917	0.935	0.950	0.972	0.988	0.988	0.988
20	N	1	0.600	0.619	0.619	0.667	0.777	0.929	0.962	0.984	0.998
20	N	5	0.940	0.940	0.947	0.955	0.968	0.969	0.984	0.988	0.988
20	2N	1	0.635	0.660	0.677	0.717	0.834	1.015	1.064	1.082	1.100
20	2N	5	0.926	0.926	0.939	0.947	0.966	0.968	0.985	0.986	0.986
40	N	1	0.744	0.744	0.744	1.110	1.177	1.388	1.407	1.368	1.323
40	N	5	0.927	0.927	0.929	0.941	0.959	0.985	1.015	1.015	1.007
40	2N	1	0.805	0.805	0.805	1.051	1.332	1.565	1.568	1.514	1.452
40	2N	5	1.004	1.004	1.006	1.020	1.034	1.080	1.120	1.120	1.121
80	N	1	0.769	0.769	0.769	0.916	1.355	1.472	1.399	1.332	1.243
80	N	5	1.002	1.002	1.002	1.007	1.048	1.072	1.084	1.089	1.087
80	2N	1	0.788	0.788	0.788	0.926	1.331	1.511	1.410	1.490	1.403
80	2N	5	1.082	1.082	1.082	1.085	1.148	1.184	1.214	1.202	1.198
	Mean R		0.826	0.833	0.838	0.912	1.020	1.114	1.127	1.129	1.113

Table 2. R values as a function of U_{min}.

N	$D_{\max}$	$U_{\max}$	R	LB	R^*	LB*	N^*	T
10	N	1	0.961	0.505	0.961	0.518	12	1.87
10	N	5	0.986	0.841	0.986	0.853	12	1.98
10	$2N$	1	0.962	0.298	0.962	0.308	12	3.24
10	$2N$	5	0.988	0.742	0.988	0.749	12	7.41
20	N	1	0.984	0.392	0.964	0.401	11	13.01
20	N	5	0.988	0.901	0.988	0.912	12	10.90
20	$2N$	1	1.082	0.268	0.853	0.423	1	4.13
20	$2N$	5	0.986	0.810	0.988	0.842	11	206.00
40	N	1	1.368	0.357	X	X	0	0.738
40	N	5	1.015	0.897	0.996	0.955	5	133.07
40	$2N$	1	1.514	0.226	X	X	0	0.776
40	$2N$	5	1.120	0.806	1.000	0.922	1	43.68
80	N	1	1.332	0.271	X	X	0	0.97
80	N	5	1.089	0.882	X	X	0	1.89
80	$2N$	1	1.490	0.191	X	X	0	1.02
80	$2N$	5	1.202	0.767	X	X	0	0.76

Table 3. Algorithm performance for $U_{\min} = 0.7$.

4. Multiple product case

We now extend the algorithm to the multi-product case, first neglecting set-up costs. Recall that for the single product problem, P1, two decisions must be made simultaneously to ensure minimal operational cost. They are: (1) the allocation of orders to batches and (2) the schedule of batches. However, when multiple products are produced the schedule is no longer a problem of just deciding when to produce a batch, but instead becomes the problem of also deciding which product type to produce in which period. The Single Product Heuristic is modified to include not only the minimum demand size $U_{\min}$ but also the need to assign a given product type to a busy period. In this case, when a production period is set to busy, a product type identifier is also assigned. For example, if the current demand evaluated in step 3 of the algorithm is of chemical type 2 and in step 4 of the algorithm the selected busy period is 4 then the variable y_{42} is preset to 1 in the integer program formulation.

Multiple product heuristic

Step 1. For each product type, determine the number of batches to produce, and sum these to determine the total number of batches N.

Step 2. Sort the demands in order of decreasing size.

Step 3. Select an unassigned order with the largest demand along with its due date and product type.

Step 4. Assign a batch of the current product type to a production period that results in the smallest penalty cost.

- If the production period coinciding with the demand's due date is available, set the period to be busy.
- If the production period coinciding with the demand's due date is not available
  - (a) determine the latest available production period that is earlier than the demand's due date. Calculate its earliness penalty cost;
  - (b) determine the earliest available production period that is later than the demand's due date. Calculate its lateness penalty;

(c) if earliness penalty < lateness penalty, assign a batch to be produced at the production period determined in step a. Else assign a batch to be produced at the production period determined in step b. Ties are arbitrarily broken.

Step 5. Decrease the size of the current order by the amount of the batch size (unity in this case).

Step 6. If the number of batches assigned is $< N$, go to step 3. Else go to step 7.

Step 7. Given the set of busy periods, solve the transportation problem.

The heuristic is polynomial with computational complexity equal to solving a transportation problem with $2N$ nodes.

As was done for the Single Product Heuristic, the Multiple Product Heuristic was tested with all parameters being varied. The one new parameter needed to test the Multiple Heuristic was $K_{\max}$, the maximum number of product types. For each demand/due date combination, the associated product type was generated from a discrete uniform random number from $(0,K_{\max}]$. The ranges of $K_{\max}$ tested were 1, 2, 3, and 5, while for testing the single product case, the best value for $U_{\min}$ was determined to be 0.7, and it is this value that was used while testing the multiple product case. For each scenario, 12 runs were performed and the mean value was recorded. Results of this testing are given in table 4.

When extended to multiple products, the MIP solver was able to find optimal solutions (i.e. $N^* > 0$) only when the number of batches was small. When examining those cases for which $N^* > 0$, the algorithm did produce near optimal values while decreasing the amount of time needed to solve the problem. Note that the reason the T ratio is small for the case of N being large is because, under this scenario, we ran out of memory faster, which is the stopping criteria. As was the case for single product problems, the algorithm gave the most improvement in the objective function for large problems. In addition to problem size, the performance of the algorithm is influenced by the values of the parameters $U_{\max}$, $D_{\max}$ and $K_{\max}$. From table 4, it can be seen that it is the more difficult problems to solve, where $U_{\max} = 1$ and $D_{\max} = 2N$, in which the algorithm yielded the most improvement.

Upon first looking at table 4 it would appear that there is very little sensitivity to the parameter $K_{\max}$, which defines the maximum number of product types allowed to be produced. However, what is not evident is that for several cases the MIP solver was not able to find any feasible solution, and the case of no feasible solution occurred at large values of N when $K_{\max}$ was large. The values of R recorded in table 4 are the means of 12 trials of the ratio of the best feasible solution from the Multiple Product Heuristic to the best feasible solution from the MIP solver. For those trials where the MIP solver was unable to find a feasible solution the ratio would be infinite. Including this infinite value when calculating the mean for the 12 trials would cause the mean to be of no value in measuring the performance of the algorithm. However, not accounting for these cases causes the results in table 4 to not reflect truly the performance improvement for the Multiple Product Heuristic. The trials where the MIP solver was not able to find a feasible solution were thus excluded from the calculation of the mean values reported in table 4. Those cases where the MIP solver was unable to find a feasible solution were the largest problems (i.e. $N = 80$, $D_{\max} = 2N$ and $U_{\max} = 1$) with $K_{\max} = 3$ and $K_{\max} = 5$. For $K_{\max} = 3$, the MIP solver failed to find a feasible solution for four out of the 12 trials while for $K_{\max} = 5$ there were five trials in which the MIP solver did not find a feasible

			$k_{max} = 1$		$k_{max} = 2$		$k_{max} = 3$		$k_{max} = 5$	
N	D_{max}	U_{max}	R	T	R	T	R	T	R	T
10	N	1	0.961	1.87	0.863	17.03	0.937	91.64	0.959	1164.96
10	N	5	0.986	1.98	0.983	2.22	0.968	3.19	0.974	9.08
10	$2N$	1	0.962	3.24	0.917	73.30	0.942	63.29	1.139	796.36
10	$2N$	5	0.990	7.41	0.976	4.02	0.983	6.78	0.983	46.40
20	N	1	0.984	13.01	1.170	9.61	1.173	1.74	1.106	1.71
20	N	5	0.988	9.16	0.989	5.55	1.005	13.70	0.995	3.76
20	$2N$	1	1.082	4.13	1.218	0.98	1.150	0.94	1.297	0.95
20	$2N$	5	0.986	206.00	1.031	1.80	1.029	1.48	1.046	1.68
40	N	1	1.368	0.74	1.279	0.82	1.374	0.96	1.322	1.06
40	N	5	1.015	133.07	1.041	0.88	1.012	1.05	0.974	0.99
40	$2N$	1	1.514	0.78	1.414	0.80	1.388	0.90	1.321	0.96
40	$2N$	5	1.120	43.68	1.047	1.08	1.136	0.91	1.028	0.94
80	N	1	1.332	0.97	1.415	0.88	1.328	0.86	1.435	0.86
80	N	5	1.089	1.89	0.969	1.32	1.081	1.00	1.098	1.01
80	$2N$	1	1.490	1.02	1.385	0.79	1.246	0.86	1.268	0.96
80	$2N$	5	1.202	0.76	1.149	1.55	1.106	0.90	1.025	1.23
	Mean		1.129	26.86	1.115	7.66	1.16	11.89	1.123	125.93

Table 4. Multiple products results.

solution. From this it can be seen that the Multiple Product Heuristic continues to offer the most improvement for large problems.

Sequence dependent set-up costs

The developed heuristics have been shown to enable improved solutions for the single product case and the extended case of scheduling multiple products. While these heuristics performed well for these two classes of problems, we now consider extending them to the sequence-dependent set-up costs case.

The multiple product problem with earliness and tardiness penalties in itself is a difficult problem to solve optimally. When there is the added requirement to include set-up costs as part of the total costs that are to be minimized, the problem becomes even more difficult to solve. The scheduling of production batches based on customer demands may cause the set-up costs to be high, while scheduling of production batches based on set-up cost may cause the penalty costs associated with meeting customer demands to be high. These two competing costs must both be simultaneously considered when solving the problem. However, there is the complication of not knowing what the production sequence is until one has solved the problem of minimizing the customer penalty costs. Only at this time is one able to even look at the production sequence and determine if the associated set-up costs are too high. Likewise, if the set-up costs are considered to be too high and the production sequence is adjusted, one must again solve the problem to minimize customer penalty costs in order to assess the cost impacts caused by adjusting the production sequence. Due to not knowing the consequences of adjustments until after the associated problem has been solved, the exact solving of multiple products with a sequence dependent set-up cost is an iterative process and is usually done only in restricted cases. For the case of a single aggregate due date, an exact algorithm has been developed (Pekny *et al.* 1993) that makes use of a modified Travelling Salesman Problem (TSP) for determination of the required scheduling sequence. The TSP is computationally intensive. Furthermore, the TSP formulation is not easily extendable to the multiple due date case. As such, the problem is often simplified. One simplifying process (Bowers *et al.* 1995) uses cluster analysis to arrive at a production sequence that reduces the product changeover times. Similar to this is the minimizing of the number of changeovers by producing all batches of a given type, before a changeover is made to produce the next product type. This causes the number of changeovers to be equal to the number of product types. After the grouping of production to the individual product types is completed, the product groups are then sequenced in a manner to minimize the set-up cost.

We next extend the heuristics to the sequence dependent set-up cost case with the added restriction that the number of changeovers equals the number of product types. Although minimizing the number of changeovers and selecting the production sequence that yields the minimum set-up costs for the grouped production of product types simplifies the problem, it is not the complete solution to the problem. In particular, there still remains the need to determine when to produce the individual batches and how to allocate these batches to the customer demands while adhering to the selected batch grouping and sequence constraints.

There are many ways to accomplish the assignment of product groups to sets of production periods. Some of these are quite simple and are often referred to as myopic while others are more extensive in their procedure, with the main difference being the trading of increased use of computational resources against an improved

solution. To demonstrate the algorithm's ability to address sequence-dependent set-up costs, two heuristics were chosen for the assignment of product groups to sets of production periods. One heuristic is a simple myopic procedure while the other involves a more complex extensive search method. The Single Product Heuristic is applied to each of these methods, and the results are compared in order to demonstrate the capability of addressing these types of sequence-dependent set-up cost problems.

The simple myopic heuristic first finds the production sequence that yields the lowest set-up cost and then allocates a number of production periods to each product type that is proportional to the number of batches of each product type to be produced. With the production of each product type now being restricted to a given set of production periods, the Single Product Heuristic is then used to schedule each product type within its allocated set of production periods. A detailed description of the algorithm is given below.

Myopic algorithm

Step 1. Determine the set-up cost for all possible production sequences.
Step 2. Find the production sequence with minimum set-up cost.
Step 3. Determine the number of batches for each product type.
Step 4. Allocate the production periods to each product type in order of the sequence found in step 2.
Step 5. Select first product type in the sequence.
Step 6. Call 'Single Product Heuristic'.
Step 7. If all product types have been assigned, end. Else select next product type and go to *Step* 6.

Steps 1 and 2 of the algorithm are of complexity $O(k_{\max}!)$. The remaining steps require solving $k_{\max}$ transportation problems.

As an example, if there are three product types with maximum due date in period 10 and the number of batches for each product type are: $n_A = 2$, $n_B = 5$ and $n_C = 3$. For this example, it is assumed that the sequence-dependent set-up costs are as follows: $s_{A,B} = 3$, $s_{A,C} = 1$, $s_{B,A} = 5$, $s_{B,C} = 4$, $s_{C,A} = 2$, $s_{C,B} = 8$. Since there are a total of ten batches that need to be produced, we need to consider only the first 20 production periods since previously we showed that an optimal schedule will be completed by time $2N$ when the maximum due date equals N. Product type C would be allocated production periods 1 through 6, product type A assigned production periods 7 through 10, and finally product type B assigned production period 11 through 20. Note that there is reserved $2n_k$ periods for each product type, where n_k is the number of batches to produce of type k. A step-by-step description of this example problem is given below. This grouping is shown in figure 1.

Step 1. Enumerate all possible production sequences.
Step 2. Determine the production sequence with the lowest set-up cost.
Results: C, A and the B.
Step 3. Determine the number of batches required for each product type.

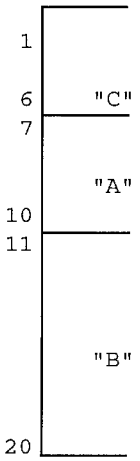

Figure 1. Myopic algorithm product grouping.

Results: $n_A = 2$ batches

$n_B = 5$ batches

$n_C = 3$ batches

Step 4. Allocate production periods to each product type.

Results: A allocated to periods 7 to 10

B allocated to periods 11 to 20

C allocated to periods 1 to 6

Step 5. Select first product type in the production sequence.
Result: Select product C.

Step 6. Invoke Single Product heuristic.
Result: $y_{1,C} = 1, y_{2,C} = 1, y_{4,C} = 1; x_{1,1,C} = 1.0, x_{2,1,C} = 1.0, x_{4,1,C} = 0.5, x_{4,2,C} = 0.5$
Repeat step 6 two more times.
Result: $y_{7,A} = 1, y_{8,A} = 1; x_{7,2,A} = 0.7, x_{7,1,A} = 0.3, x_{8,1,A} = 1.0$

and,

$y_{11,B=1}, y_{12,B=1}, y_{13,B} = 1, y_{14,B} = 1, y_{15,B} = 1; x_{11,3,B} = 0.4,$
$x_{11,1,B} = 1.0, x_{12,1,B} = 1.0, x_{13,1,B} = 0.9, x_{13,4,B} = 0.1, x_{14,4,B} = 0.5,$
$x_{14,2,B} = 0.5, x_{15,2,B} = 1.0$

Find: Penalty cost = 125.15
Set-up cost = 5.0
Total cost = 130.15

The inefficiency associated with fixing the production periods for each product type can be seen by the previous example. If production of product B is pushed to start in period 9 instead of period 11, there will be less penalty cost associated with tardiness. This is the motivation behind the exhaustive search algorithm described next.

The Exhaustive Search Algorithm is similar to the Myopic algorithm in that it uses the same production sequence and assigns batches of a given type only within the set of production periods allocated for each given product type. However, instead of rigidly fixing the size of the production periods allocated to each product type, the algorithm performs an exhaustive search to find an appropriate interval length. The exhaustive search algorithm starts with the same production sequence yielding the lowest set-up costs as did the myopic algorithm. Starting with the first two product types in the production sequence, it operates on the product types one pair at a time. This operation on the sequential pairs of product types is performed by determining the beginning and end of the set of production periods that are to be allocated to the given sequential pair of products, but the boundary between the production periods for each of the two product types currently being scheduled is not fixed. Instead, it is continuously modified through an exhaustive search to determine the value of the bound between the two product types that will produce the smallest penalty cost for this pair of products. Once the exhaustive search has determined the bound for the first and second product types in the production sequence, the bound and assignment for the first product type of the production sequence is fixed, and the algorithm then performs this same exhaustive search for the second and third product in the production sequence. This search continues until all product pairs have been searched.

When the Exhaustive Search algorithm is operating on a given pair of product types, the size of the set of production periods is made large enough to ensure that all possible values of the bound between the two product types of the product pair are included. Let the variables First and Horz be the beginning and ending periods allocated to the pair of products, and let $\mathrm{DUE}_{\max}(k)$ be the maximum due date for any order of product type k. Then, $\mathrm{Horz} = \mathrm{First} + \mathrm{DUE}_{\max}(k) + \mathrm{DUE}_{\max}(k+1)$ for this product pair. At each iteration of the search, production periods (First, Bound) are allocated to product k and periods (Bound + 1, Horz) are allocated to product $k+1$. The algorithm increments the variable Bound by one for each iteration. It performs the exhaustive search of all possible values for the variable Bound up to the value $\mathrm{Horz} - n_{k+1}$, where n_k is the number of batches required of type k.

We now illustrate the exhaustive search algorithm using the same data set as in the previous example. For this example, it will also be assumed that the maximum due date for A is 12, for B is 18, and for C is 17. First, operate on only the first two products in the minimum set-up cost sequence: product C and product A. Given that this is the initial pair of products, the value of the variable First= 1. The planning horizon can also be calculated by summing the maximum due dates for the two products and adding the sum to the variable First. $\mathrm{Horz} = 1 + 17 + 12 = 30$. The initial and final values of the variable 'Bound' that will just allow the assignment of the respective product types batches would then be 3 and 28 respectively. The initial and final values of 'Bound' and the associated production periods available for the assignment producing batches of product C and Product A are shown in figure 2.

Once this initial Bound for the above shown production periods has been made, the assignment heuristic developed for the multiple products case is used to assign individual production batches. The resulting transportation problem is solved, the solution is recorded as the BestSoln, and the value of Bound is stored. The value of Bound is next incremented to 4, and again the assignment heuristic assigns individual production batches, and the transportation problem is again solved. If the solution

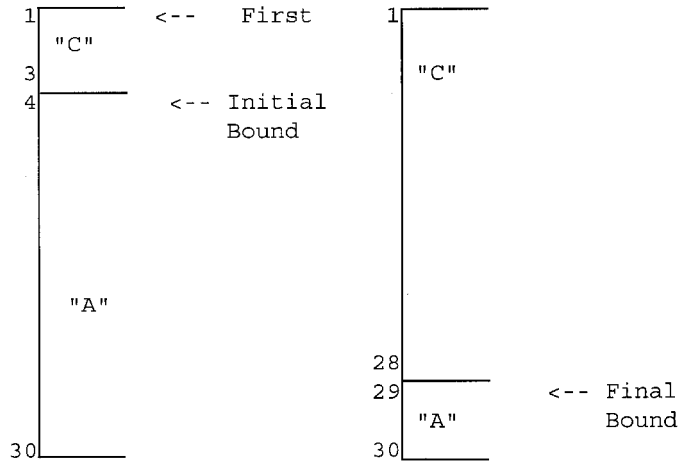

Figure 2. Initial and final product grouping for first pair of exhaustive searches.

to this iteration's transportation problem is better than the prior BestSoln then the stored values of BestSoln and its associated Bound are updated with these new values. This iterative process continues until Bound is at its limit of 28. Once complete, the assignments for product C are fixed, and the value of Bound associated with the BestSoln is used to establish the starting point for the algorithm operating on the next pair of products. For this example it is assumed that the value of this best Bound is 11.

Given this value for the best Bound from the first pair of products, one may now calculate First and Horz for the pairing of A and B. One thus arrives at First = 12 and Horz = 41 as shown in figure 3. The algorithm then operates on this second pair of product types.

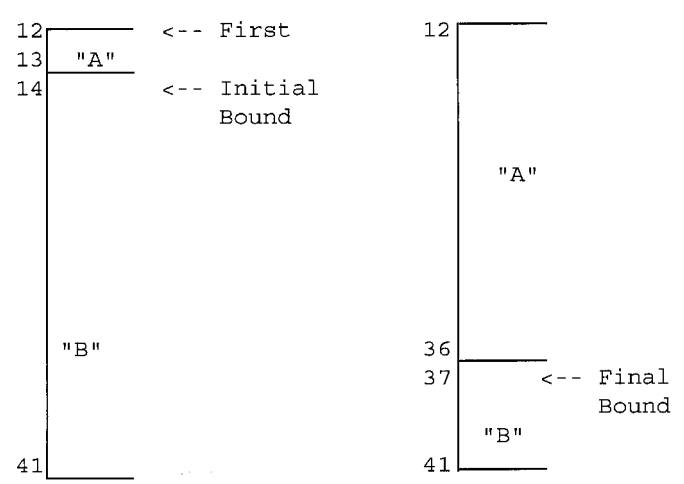

Figure 3. Initial and final product grouping for second pair of exhaustive searches.

Exhaustive search algorithm

Step 1. Determine the set-up cost for all possible production sequences.
Step 2. Find the production sequence with minimum set-up cost.
Step 3. Determine the number of batches for each product type. Set $k = 1$.
Step 4. Determine the first production period by using the best bound from the prior pair of products.
First = BestBound + 1
Note: For the first pair of products; First = 1
Step 5. Determine the horizon for this pair of products by using the maximum due dates, $\mathrm{DUE}_{\max}$, for each of the product types. Horz = First + $\mathrm{DUE}_{\max}(k) + \mathrm{DUE}_{\max}(k+1)$
Step 6. Determine the initial bound and final bound to be used for the exhaustive search over the range of the bound between the two product types by using the number of batches of the first product, n_K, and the second product, n_{K+1}.
StartBound = First + n_K, EndBound = Horz − n_{K+1}, CurrentBound = StartBound
Step 7. Invoke the single product heuristic for product k using periods (First, CurrentBound) and for $k+1$ using periods (CurrentBound + 1, Horz).
Step 8. Compare the current solution to the best solution so far. If it is better than the best solution, replace the best solution with this current solution. Also, update the best bound value (i.e. BestBound = Current Bound).
Step 9. If this is the last value for bound as determined by CurrentBound < EndBound go to step 10.
Else, increment CurrentBound by one and go to step 7.
Step 10. If this is the last product pair in the production sequence, end. Else, $k = k + 1$, go to step 4.

Steps 1 and 2 of the algorithm are of complexity $O(k_{\max}!)$. The remaining steps require solving $2N$ transportation problems.

Both the myopic and the exhaustive search algorithms were tested to determine their individual and relative performance over a range of parameters. Another variable, which is unique to this problem, is the sequence-dependent set-up cost, which is a $K_{\max}$ by $K_{\max}$ array representing all possible combinations of changeovers. Elements of this array are a continuous uniform random number from (1, $C_{\max}$],

Product type	Order number	Demand size	Due date	Early penalty	Late penalty
A	1	1.3	4	3	4
A	2	0.7	2	2	6
B	1	2.5	7	1	3
B	2	1.5	8	3	2
B	3	0.4	6	4	4
B	4	0.6	10	2	3
C	1	2.5	1	2	4
C	2	0.5	4	3	5

Table 5.

			$k_{max} = 2$		$k_{max} = 3$		$k_{max} = 5$	
N	D_{max}	U_{max}	Q	T	Q	T	Q	T
10	N	1	1.163	76.71	1.056	162.11	0.893	309.47
10	N	5	1.214	52.07	1.125	94.10	0.928	127.04
10	$2N$	1	1.173	123.53	1.121	256.66	1.007	491.57
10	$2N$	5	1.235	84.87	1.228	143.69	1.224	197.04
20	N	1	1.061	184.25	1.052	396.78	0.892	919.02
20	N	5	1.344	136.39	1.140	254.64	1.044	451.95
20	$2N$	1	1.062	309.55	1.096	634.16	1.003	1468.94
20	$2N$	5	1.415	217.65	1.231	401.43	1.117	723.74
40	N	1	1.017	417.72	1.038	902.45	1.011	2486.40
40	N	5	1.102	344.26	1.017	679.40	0.933	1387.42
40	$2N$	1	1.022	722.14	1.075	1555.47	1.084	4399.88
40	$2N$	5	1.095	545.53	1.082	1093.61	1.040	2264.71
Mean			1.159	267.92	1.105	547.88	1.015	1268.93

Table 6. Sequence dependent set-up costs results.

with C_{max} being 1000. For each scenario, 12 runs were performed and the mean value was recorded.

Results of this testing are given in table 6 where Q is the ratio of the myopic algorithm's penalty to the penalty from the exhaustive search algorithm, and T is the ratio of the solution time for the exhaustive search to the solution time for the myopic algorithm. As can be seen in table 5, the exhaustive search algorithm yielded solutions that were, overall, better than those of the myopic algorithm.

However, for those cases of there being a larger number of product types, $K_{max} = 5$, the exhaustive search algorithm typically gave solutions that were only marginally better than those of the myopic algorithm. In addition, the amount of computation time required by the exhaustive search algorithm was dramatically more than that needed by the myopic algorithm. The overall average for Q is 1.093. However, table 5 shows that for $K_{max} = 5$ the relative performance is only 1.015. When looking at the cases for $K_{max} = 2$ and $K_{max} = 3$ one finds the exhaustive search algorithm gives performance improvements of 1.159 and 1.105, respectively. The performance improvement of the exhaustive search algorithm over that of the myopic algorithm decreases as the number of batches being produced increases. However, for $K_{max} = 2$ and $K_{max} = 3$, there is an increase in the performance improvement when $U_{max} = 5$. This case is when there are fewer individual orders but each may be larger in size. However, for $K_{max} = 5$, the situation is at times reversed with the most improvement being for $U_{max} = 1$.

The exhaustive search algorithm was designed to allow for the search for the best bound between sequential sets of product types. As such, there was no limit placed on the range of production periods to which individual batches could be assigned. It should be noted that the exhaustive search procedure is guaranteed to equal or outperform the myopic algorithm when $K_{max} = 2$. For a small number of product types, the exhaustive search algorithm did yield improved results when trying to minimize earliness and tardiness penalty costs while simultaneously trying to reduce set-up costs. For $K_{max} = 5$ there are more product pairs that are sequentially operated on by the exhaustive search algorithm, and each set of product pairs allocation begins after the best bound found for the prior set of product pairs.

This means that for the last set of product pairs, the earliest possible assignment of batches to production periods can be so late that an assignment that produces reasonably small penalty costs will not be possible.

5. Conclusions

In this paper, the scheduling of chemical processing as it relates to the production of specialized chemicals was examined. This branch of the industry typically produces in lot sizes that are too small to warrant the capital investment of continuous processing plants and thus resorts to fixed batch size production plants. However, the customer orders are not restricted to be the size of a batch. When an order is larger than a batch size the excess production from other batches is used to fill the larger orders. For small orders, a single batch is used to fill several orders. This complicates the problem in that each batch can have multiple due dates.

Of particular interest to the agricultural industry is the on-time delivery of the customer orders. If the customer order is late, there could be reduced crop yields or excessive crop damage. These consequences result in there being a penalty in the form of contractual stipulations, or loss of customers for being late in filling a customer's order. In addition, there are penalties resulting from such items as increased inventory and warehouse costs. This leads to operating the batch chemical plant in a manner that is analogous to a just-in-time practice.

Most of the research in the joint earliness and tardiness problem that applies to the batch chemical manufacturing environment makes simplifying assumptions in order to more easily solve the problem. For example, the prior research assumes all jobs have a common due date, or symmetric earliness and tardiness penalties. The heuristics developed in this paper do not make these simplifying assumptions.

The decision variables of our formulation consist of both integer and continuous variables. The integer decision variables determine the set of busy production periods, and the continuous variables determine the allocation of the production batches to the customer orders. Our solution procedure first applies heuristic rules to determine the set of busy production periods (i.e. integer variables). Then, a transportation problem is solved to determine the best allocation of the production batches to customer orders (i.e. the continuous variables) given the current set of busy production periods. An augmenting solution approach would be to use a neighbourhood search technique such as simulated annealing or tabu search. However, these methods work best on pure integer formulations. In our context, these techniques could only be employed to search for the best set of busy production periods. At each step of the search, our proposed transportation problem will still need to be solved to determine the allocation of batches to customer orders. Although using a neighbourhood search technique to identify the set of busy periods may improve the quality of the solution, it may be computationally prohibitive since a new transportation problem will have to be solved at each step of the search. The issue of whether augmenting our heuristic approach with a neighbourhood search will significantly improve the results and whether it is computationally feasible can be a subject for further research.

References

Allahverdi, A., Gupta, J. N. D. and Aldowaisan, T., 1999, A review of scheduling research involving setup considerations. *Omega: The International Journal of Management Science,* **27**, 219–239.

BAKER, K. R. and SCUDDER, G. D., 1990, Sequencing with earliness and tardiness penalties: a review. *Operations Research,* **38**, 22–36.

BIREWAR, D. B. and GROSSMANN, I. E., 1990, Simultaneous production planning and scheduling in multiproduct batch plants. *Industrial Engineering and Chemical Research,* **29**, 570–580.

BRUCKER, P., 1995, *Scheduling Algorithms,* Chapter 8, Section 8.1 (Springer Verlag).

CHEN, Z.-L., 1996, Scheduling and common due date assignment with earliness-tardiness penalties and batch delivery costs. *European Journal of Operations Research,* **93**, 49–60.

CRAUWELS, H. A. J., POTTS, C. N. and VAN WASSENHOVE, L. N., 1997, Local search heuristics for single machine scheduling with batch set-up times to minimize total weighted completion. *Annals of Operations Research,* **70**, 261–279.

BOWERS, M. R., GROOM, K., NG, W. M. and ZHANG, G., 1995, Cluster-analysis to minimize sequence-dependent changeover times. *Mathematical and Computer Modeling,* **21**, 89–95.

DESSOUKY, M. M. and KIJOWSKI, B. A., 1997, Production scheduling of single-stage multiproduct batch chemical processes with fixed batch size. *IIE Transactions,* **29**, 399–408.

DESSOUKY, M. M., KIJOWSKI, B. A. and VERMA, S. K, 1999, Simultaneous batching and scheduling for chemical processing with earliness and tardiness penalties. *Production and Operations Management,* **8**, 433–444.

DESSOUKY, Y. M., ROBERTS, C. A., DESSOUKY, M. M. and WILSON, G., 1996, Scheduling multi-purpose batch plants with junction constraints. *International Journal of Production Research,* **34**, 525–541.

GAREY, M. R., TARJAN, R. E. and WILFONG, G., 1988, One-Process scheduling with symmetric earliness and tardiness penalties. *Mathematics of Operations Research,* **13**, 330–348.

HALL, N. G. and POSNER, M. E., 1991, Earliness-tardiness scheduling problems, part I: weighted deviation of completion times about a common due date. *Operations Research,* **39**, 836–846.

HARIRI, A. M. A. and POTTS, C. N., 1997, Single machine scheduling with batch set-up times to minimize maximum lateness. *Annals of Operations Research,* **70**, 75–92.

HEUTS, R. M. J., SEIDEL, H. P. and SELEN, W. J., 1992, A comparison of two lot sizing-sequencing heuristics for the process industry. *European Journal of Operational Research,* **59**, 413–424.

KONDILI, E., PANTELIDES, C. C. and SARGENT, R. W. H., 1993, A general algorithm for short-term scheduling of batch operations I. MILP formulation. *Computers and Chemical Engineering,* **17**, 211–227.

KU, H. M. and KARIMI, I. A., 1990, Scheduling in serial multiproduct batch processes with due-date penalties. *Industrial Engineering Chemical Research,* **29**, 580–590.

KU, H. M. and KARIMI, I. A., 1991, Scheduling algorithms for serial multiproduct batch processes with tardiness penalties. *Computers and Chemical Engineering,* **15**, 283–286.

KYDVA, G., ELKAMEL, A., PEKNY, J. F. and REKLAITIS, G. V., 1994, Heuristic algorithm for scheduling batch and semi-continuous plants with production deadlines, intermediate storage limitations and equipment changeover costs, *Computers and Chemical Engineering,* **18**, 859–875.

LAWLER, E. L., 1964, On scheduling problems with deferral costs. *Management Science,* **11**, 280–288.

LENSTRA, J. K., RINNOOY KAN, A. H. G. and BRUCKNER, P., 1977, *Complexity of Machine Scheduling Optimization: Annotated Bibliographies,* edited by O'hEighertaigh *et al.* (New York: Wiley), pp. 164–189.

MONMA, C. L. and POTTS, C. N., 1989, On the complexity with batch setup times, *Operations Research,* **37**, 798–804.

MUSIER, R. F. H. and EVANS, L. B., 1991, Schedule optimization with simultaneous lot sizing in chemical process plants. *American Institute of Chemical Engineering Journal,* **37**, 886–896.

NAGAR A., HADDOCK J. and HERAGU, S., 1995. Multiple and bicriteria scheduling: a literature survey. *European Journal of Operations Research,* **81**, 88–104.

PATSIDOU, E. P. and KANTOR, J. C., 1991, Scheduling of a multipurpose batch plant using a graphically derived mixed-integer linear program model. *Industrial Engineering Chemical Research,* **30**, 1548–1572.

PEKNY, J. F., MILLER, D. L. and MCRAE, G. J., 1990, An exact parallel algorithm for scheduling when production costs depend on consecutive system states. *Computers and Chemical Engineering,* **14**, 1009–1023.

PEKNY, J. F., MILLER, D. L. and KUDVA, G., 1993, An exact algorithm for resource constrained sequencing with application to production scheduling under an aggregate deadline. *Computers and Chemical Engineering,* **17**, 671–682.

TANDON, M., CUMMINGS, P. T. and LEVAN, M. D., 1991, Flowshop sequencing with non-permutation schedules. *Computers and Chemical Engineering,* **15**, 601–607.

UZSOY, R., 1994, Scheduling a single batch processing machine with non-identical job sizes. *International Journal of Production Research,* **32**, 1615–1635.

UZSOY, R., 1995, Scheduling batch processing machines with incompatible job families. *International Journal of Production Research,* **33**, 2685–2708.

VERMA, S. and DESSOUKY, M. M., 1998, Single-machine scheduling of unit-time jobs with earliness and tardiness penalties. *Mathematics of Operations Research,* **23**, 930–943.