Capacity-Driven Acceptance of Customer Orders for a Multi-Stage Batch Manufacturing System: Models and Algorithms

Robin Roundy Dietrich Chen Pan Chen Metin Çakanyildirim
Michael B. Freimer Vardges Melkonian

November 27, 2003

Abstract

An automotive parts manufacturer produces a wide variety of parts in a job-shop environment. Many of the manufacturing operations have substantial setups. When a client phones in an order, the manufacturer must decide quickly whether or not it has the capacity required to accept the order. We develop a simplified formulation of the order acceptance problem. We formulate the discrete-time version as an integer program. The problem is NP-hard, but in 51 out of 51 test problems the LP relaxation is tight. For larger problems we test several heuristics. Three of the heuristics look promising – simulated annealing, a genetic algorithm, and an LP-based heuristic.

1 Introduction

Two strong trends affecting both manufacturing companies and software vendors are *Finite-Capacity Scheduling* (FCS) systems and *Available-to-Promise* (ATP) systems. FCS systems produce and maintain detailed, capacity-feasible schedules of manufacturing facilities. ATP systems quickly decide whether incoming customer orders can be accommodated. The first ATP systems allocated finished-goods inventory and planned receipts to clients, but they did not alter production schedules. The idea of combining ATP and FCS systems by altering the production schedule in response to incoming customer orders is gaining in popularity, in spite of the computational difficulty of scheduling problems and the need for very short response times. In this paper we develop a model that combines many of the features of FCS and ATP systems, and we design and test viable algorithms for this model.

Our model is motivated by the following example. A European company manufactures a wide variety of components for the automotive industry. Often, a client will place an order for a single shipment of a single product. Usually such an order would be met by generating a single production batch. However, many of their clients call to negotiate for a single customer order that consists of a sequence of small, just-in-time shipments, often covering a time period as long as 6 months. (This is a common practice in the automotive industry.) The supplier needs to decide how to partition the just-in-time shipments into some number of production batches, to determine sizes and due dates for those production batches, and to make sure that enough capacity is available to produce the batches, before accepting the customer order. Due to the variety of parts being produced and the volumes in which they are sold, the manufacturing system is characterized by substantial setups, by unstructured multi-stage routings, and by the assembly of manufactured sub-components.

The company has a policy of storing finished goods inventory when necessary and minimizing work in process (WIP). They accomplish this as follows. The company pays close attention to capacity when loading the manufacturing facility. Lot-for-lot batch sizing is used, so batches maintain their identity as they progress through the shop. The company offsets batch due dates by relatively short manufacturing lead times to obtain due dates for each manufacturing operation. Finally, they insist that the shop floor adhere to due-date-driven sequencing, and they use overtime as needed to keep up.

We assume that a detailed, feasible schedule exists in computer memory. When a new order comes in we look for a feasible schedule that accommodates it and all of the orders we have already accepted. We may use multiple production batches to fill the new order, so lot sizing decisions are made as well.

We develop a Mixed Integer Linear Programming (MILP) formulation of the order insertion problem that bypasses detailed Gantt Chart manipulations but guarantees feasibility. Bypassing detailed Gantt Chart manipulations is important in the sense of the complexity of running time which is illustrated in references on papers of type (i) and (iii) below. The problem is proven to be NP-hard in Roundy, Çakanyildirim, Chen, Chen, Freimer, Jackson, and Melkonian (1999), but we need a fast response time. We propose heuristics based on genetic algorithms, tabu search, simulated annealing, myopic search and a shortest-path relaxation, and test them extensively.

We assume that high-priority and low-priority customers, and many of the effects of randomness, are addressed at a higher level in the decision-making hierarchy. In our experience these systems work more or less as follows. There is a function associated with the forecasting or production planning group that reserves capacity to accommodate late orders from high-priority clients, and to compensate for other forms

of randomness. This is usually accomplished by inserting phony orders into the planning and scheduling systems. When a client transmits an order to the company, an attempt is made to accommodate the order from existing inventories and planned receipts. If that fails, the request is channeled to the FCS system if the requested due date is in the near future, and to the production planning system if it is not. If the request is from a high-priority client, capacity allocated to phony orders may be released before attempting to insert the new order.

The existing ATP literature is limited. Recently-published applications of popular commercial software alter the production schedule, either with a long-time-bucket approach like MRP, or with a short-time-bucket approach by communicating with an FCS module. King (1996), Hill (1998), Gould (1998) and Taylor and Bolander (1997) discuss ATP applications at Unisys Corp and in the apparel, automotive and flow manufacturing industries, respectively. The evolution of factory management philosophies from a resource-centric point of view to a customer-centric point of view, and the consequent need to integrate ATP and FCS, are discussed in Layden (1996) and Kevin (1996).

There is a specific branch of FCS problems that involves optimization criteria based on job due dates. Such problems can be roughly divided into the following three types:

(i) For given due dates and capacities, minimize a due-date-related measure. Tardiness *is* allowed.

(ii) Evaluate feasibility. Capacities may or may not be adjusted. Tardiness is *not* allowed.

(iii) For given capacity levels, quote due dates and/or reject orders. Tardiness is usually *not* allowed.

A good portion of classical job shop scheduling theory, e.g. Conway, Maxwell, and Miller (1967) and Baker (1974), deals with problems of type (i). Most job shop scheduling approaches are inspired by branch and bound concepts. They either find bounds, as in Lawler, Lenstra, Kan, and Shmoys (1993), or propose heuristics. A survey of job shop scheduling is found in Blazewicz, Domschke, and Pesch (1996).

Among heuristics for type-(i) problems, the shifting bottleneck procedure as employed by Adams, Balas, and Zawack (1988) stands out for successfully merging analytical techniques with heuristic ideas. For a job-shop Lambrecht, Ivens, and Vandaele Lambrecht, Ivens, and Vandale (1998) first figure out lead times and lot sizes using a queueing network then schedule these lots in a multi-machine environment using ideas from shifting bottleneck procedure. van Laarhoven, Aarts, and Lenstra (1992) use simulated annealing to approximately minimize the makespan for the job shop scheduling problem. Kim and Kim (1995) use both simulated annealing and genetic algorithms to minimize the weighted sum of tardiness and earliness. Enns

(1996) evaluates blocked time and event driven algorithms for FCS in terms of their flow time, schedule stability, and delivery performance. Dauzere-Peres and Lasserre (1997) iteratively solve lot sizing and job-shop scheduling problems to reduce the makespan which includes nonzero set up times.

Problems of type (ii) often combine short term capacity planning with scheduling. Gelders and Kleindorfer (1974) combine scheduling with overtime decisions for a one machine problem, optimizing a combination of overtime costs, flow time and tardiness. Holloway and Nelson (1974) and Gelders and Kleindorfer (1975) also optimize the use of overtime in the context of detailed scheduling. For cyclic schedules, Loerch (1990) and Loerch and Muckstadt (1994) test the schedule feasibility using an activity network. In the case of infeasibility, they evaluate outsourcing and overtime alternatives.

A survey of papers of type (iii) can be found in Cheng and Gupta (1989). A due date quotation problem with stochastic order arrival times was studied by Bookbinder and Noor (1985). Duenyas and Hopp (1995) and Duenyas (1995) study the due date quotation policies with various assumptions regarding capacity, lead times, and customers with different price and lead time preferences. For finite capacity and fixed lead times, they establish that a new order should be accepted if there are fewer than a particular number of orders already in the system. For variable lead times, they show that the quoted lead times increase with the number of orders in the system. Spearman and Zhang (1999) also study due-date setting schemes to minimize average time spent in the system subject to tardiness constraints. In a deterministic setting, Luss and Rosenwein (1993) provide a due date quotation model which minimizes weighted deviations from due dates. In a similar study, Enns (1998) studies lead time selection for good delivery performance.

A body of literature closely related to ATP and the type-(iii) problems mentioned above has recently surfaced: *Order Acceptance* or *Job Insertion*. The issue is adding a new customer order to a pre-existing schedule. Duenyas and Hopp (1995), cited above, is one example. A variant of *Material Requirements Planning*, called MRP-C, schedules jobs backward in time subject to capacity constraints but does not deal with lot sizing decisions; see Hopp and Spearman (1996). Another job insertion approach, within the same realm of type (ii) problems, is given by Akkan (1996) where a minimum cost overtime plan is generated. In Slotnick and Morton (1996) late orders are allowed, and each customer order has a revenue and linear lateness penalty. Keskinocak, Ravi and Tayur (1999) considers revenues based on quoted lead times for accepted orders on a single machine without lot sizing. Wester, Wijngaard and Zijm (1992) study utilization of a single (bottleneck) machine under different order acceptance strategies.

Our paper is an *Order Acceptance* study related to category (iii). We study *a job-shop problem* rather than

the popular single machine problem in the existing literature, e.g. Wester et al., Keskinocak et al., Slotnick and Morton, Spearman and Zhang, and Duenyas. Our problem requires us to deliver all existing *orders on time without overtime*, unlike Wester et al., Akkan, Enns, and Bookbinder and Noor that either create missed due dates or allow overtime. Contrary to the existing literature where either all jobs are available initially (e.g. Slotnick and Morton) or schedules are re-generated from scratch (e.g. Enns) with the arrival of a new job, we *insert new jobs* into an existing schedule without altering previously promised due dates. We optimize *lot sizing* and scheduling simultaneously, differing from Duenyas, Spearman and Zhang, and Wester at al. which schedule jobs without lot sizing.

We structure our paper as follows: in section 2 we state our key assumptions, characterize feasible schedules, and model the flexibility that exists for inserting a new customer order into a feasible production plan. Our MILP formulation is given in section 3. Various heuristics are discussed in section 4. In section 5, we examine the computational results.

2 Characterizations of Feasible Schedules

We consider the following problem. A detailed, feasible schedule exists for the current production load. When a new order comes in we look for a feasible schedule that accomodates it and all of the orders we have already accepted. If there is such a feasible schedule we look for the one which minimizes the cost.

The following assumptions underlie our model of the problem.

(1) The planning horizon spans the time interval $[0, Z]$. Machines operate continuously. All processes are determinisic.

(2) Each product type j has a multi-stage routing which may involve assembly operations. Each machine m appears in the routing of a given product at most once and has a nominal lead time $L(m)$.

(3) Each order involves only one type of product, but it may require several shipments indexed by i with specified quantities $Q(i)$ and shipment dates $D(i)$.

(4) We are already committed to a set of production batches indexed by b, with quantities $q(b)$ and due dates $\tau(b)$. If product j has B batches and an M-stage routing, then j requires a total of BM operations.

(5) An operation is performed on a batch b of product j at machine m where m is on the routing of j. Operation k is done on machine $m(k)$ with (sequence independent) setup time $s(k)$ and processing time

per part $tpp(k)$. The processing time of k is $p(k) = s(k) + tpp(k) \cdot q(b)$ if k corresponds to batch b. At any given time, at most one operation can be scheduled on a machine.

(6) Due dates for batches are offset by nominal lead times to obtain a due date $d(k)$ for operation k. We do not schedule an operation to start more than one nominal lead time before its due date.

(7) The current production schedule is feasible. An incoming order is accepted only if a feasible schedule that includes it can be found. Among all feasible schedules, the one that minimizes the sum of all setup costs and holding costs is preferred.

(8) For simplicity only, we assume that raw material costs dominate the holding cost computations. Thus added values are negligible.

A *production plan* is the set of operations associated with client orders that have already been accepted, plus the relevant data, namely, $p(k), d(k), m(k), L(m)$ for each operation k and each machine m. Overtime and other means of changing capacity are not modeled.

The assumptions above are the key attributes of our model. Assumptions 2 and 6 about lead times are especially important because they allow us to consider schedules on each machine in isolation, and to decompose key aspects of our job shop re-scheduling problem into a series of simpler single-machine problems. These assumptions result in a loss of flexibility because scheduling on each machine must now be done within a time window. However the assumptions greatly simplify the problem while maintaining a great degree of flexibility, as we discuss later in this section.

Because of assumption 8, the holding cost of work in process inventories are not affected by the sequence of intermediate operations. These costs are not relevant for our job insertion problem so they are ignored. Thus, the raw material receipts do not need to be studied explicitly provided that they do not interfere with feasibility. One can presume that raw materials are received sufficiently early that scheduled operations can be performed. Consequently, the rest of this section focuses on feasibility.

We focus on the question of whether or not a production plan is feasible, and on what flexibility exists for inserting a new customer order into a feasible production plan. Feasibility requires that each operation k must finish by its due date $d(k)$, and start at most $L(m(k))$ before that. Consequently, we can answer these questions by considering each machine in isolation. *For the remainder of this section we consider a single machine m.* We assume that the operations on m are indexed from 1 to J and that $d(k) \leq d(k+1)$. The *current workload on machine m* is the set of operations $\{[d(k), p(k)] : 1 \leq k \leq J\}$.

Let $L := L(m)$. In view of the fact that for each operation, the due date and the earliest start date differ by L, we have the following *Earliest-Due-Date* (EDD) result. It easily proven using adjacent pairwise interchange arguments.

Lemma 1 *A feasible schedule exists for machine m if and only if there is a feasible schedule that sequences the operations in EDD order, i.e., in the order* $1, 2, 3, \ldots$

Lemma 1 tells us how to construct a feasible schedule. But more importantly, it establishes the EDD schedule as a certificate for checking the existence of a feasible schedule. We define a *schedule* for machine m to be a function $\sigma(t)$, defined for $t \in [0, Z]$, with the following properties:

$$\sigma(0) = 0\ , \quad \sigma(\cdot) \text{ is non-decreasing}\ , \quad \text{and} \quad \sigma(t+\gamma) \leq \sigma(t) + \gamma \quad \forall\ t \geq 0,\ \gamma > 0. \tag{1}$$

$\sigma(t)$ is the cumulative production time in time interval $[0, t]$. This approach of defining a schedule allows us to introduce the concept of flexibility of a production schedule (by appropriate notation introduced later in this section).

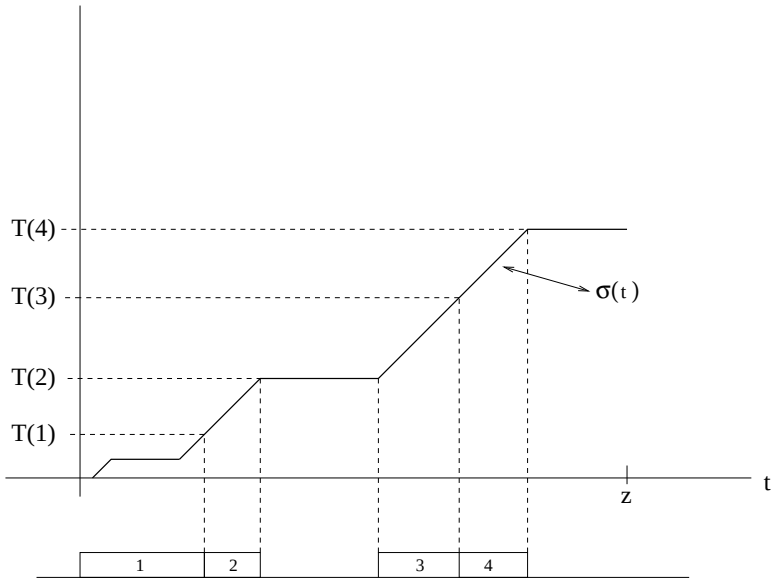

Figure 1: A Schedule

A schedule is converted into a Gantt Chart as follows (see Figure 1). The machine is working on operation k at time t if $T(k-1) < \sigma(t) < T(k)$, where $T(k) := \sum_{j=1}^{k} p(j)$. If $\sigma'(t)$ exists then it is the rate at which the machine is working at time t. In particular, suppose that $\sigma(t) = c \quad \forall\ t \in [a, b]$. If $T(k-1) < \sigma(t) < T(k)$

then $(b-a)$ units of idle time are inserted part of the way through operation k (e.g., see operation 1 in Figure 1). If $\sigma(t) = T(k)$ then $(b-a)$ units of idle time are inserted after operation k and before operation $k+1$. Our assumptions on $\sigma(\cdot)$ imply that the machine is allowed to work at any speed between 0 and 1, and that the speed can be changed at any time. More restrictive assumptions could be made without affecting any of our main results.

A schedule $\sigma(\cdot)$ is *feasible* if and only if

$$\sigma(d(k)) \geq T(k) \quad \text{and} \quad \sigma(d(k) - L) \leq T(k-1) \quad \forall\, k \geq 0, \tag{2}$$

i.e., if operation k finishes by time $d(k)$ and starts after time $d(k) - L$. To simplify beginning and end of horizon conditions let $d(0) = 0,\ d(J+1) = Z + L,\ T(0) = 0,\ T(-1) = -L$ and $T(J+1) = T(J)$.

Many different feasible schedules might correspond to the current production load on the machine. We can push the current operations back and forth in the Gantt Chart to create a space for the new orders to be inserted. Next we introduce the concepts of the earliest and latest feasible schedules which specify what are the limits of this flexibility: how far left and how far right we can push the current operations. First we need some notation.

Let $x \vee y := \max\{x, y\},\ x \wedge y := \min\{x, y\},\ x^+ := x \vee 0$. Let

$$\begin{aligned} \mathcal{L}'(k,t) &:= [T(k) - (d(k) - t)^+] && \forall\, 0 \leq t \leq Z + L,\ \forall\, 0 \leq k \leq J+1, \\ \mathcal{E}'(k,t) &:= [T(k-1) + (t + L - d(k))^+] && \forall\ -L \leq t \leq Z,\ 0 \leq k \leq J+1. \end{aligned}$$

We claim that if σ is feasible then $\mathcal{L}'(k,t) \leq \sigma(t) \leq \mathcal{E}'(k,t)$ for all t, k. Since $\sigma(d(k))$ is the cumulative production **time** by the due date $d(k)$ of operation k, if $t \geq d(k)$ then $\sigma(t) \geq \sigma(d(k)) \geq T(k) = \mathcal{L}'(k,t)$. If $t < d(k)$ then $\sigma(t)$ must be at least $T(k) - (d(k) - t) = \mathcal{L}'(k,t)$. Considering $\mathcal{E}'(k,t)$, operation k cannot be started before time $d(k) - L$, so $\sigma(t) \leq T(k-1) + [t - (d(k) - L)]^+ = \mathcal{E}'(k,t)$. Thus our claim holds.

We define the *latest feasible schedule* $\mathcal{L}(\cdot)$ and the *earliest feasible schedule* $\mathcal{E}(\cdot)$ as follows:

$$\mathcal{L}(t) := \max_{k:0 \leq k \leq J+1} \{\mathcal{L}'(k,t)\},\ 0 \leq t \leq Z + L, \quad \text{and} \quad \mathcal{E}(t) := \min_{k:1 \leq k \leq J+1} \{\mathcal{E}'(k,t)\},\ -L \leq t \leq Z. \tag{3}$$

The following lemma gives a direct characterization of the data sets $(d(\cdot), T(\cdot), L)$ which admit a feasible schedule, and establishes that $\mathcal{L}$ and $\mathcal{E}$ are the latest and earliest schedules, respectively.

Lemma 2 *A feasible schedule exists if and only if, for all $i = 0, ..., J$ and $k = 1, ..., J$, $d(i) - L \leq d(k)$ implies*

$$T(k) + (d(i) - L - d(k)) \leq T(i-1). \tag{4}$$

If $\sigma(\cdot)$ is a feasible schedule then $\mathcal{L}$ and $\mathcal{E}$ are both feasible, and $\mathcal{L}(t) \leq \sigma(t) \leq \mathcal{E}(t) \quad \forall\, 0 \leq t \leq Z$.

All proofs are found in the Appendix. Note that setting $i = 0$ in (4) we obtain

$$T(k) \leq d(k) \quad \forall\, k, \tag{5}$$

i.e., there must be adequate capacity to complete the first k operations before the deadline of the k-th operation. If $d(i) \leq d(k)$ then in the time interval $[d(i), d(k)]$, operations requiring a total of $T(k) - T(i-1)$ units of processing time are due. Processing must be completed by time $d(k)$ and cannot begin before time $d(i) - L$, so $d(k) - d(i) + L$ units of time are available to do the work. This leads to (4). Also note that by setting $k = i$ in (4) we obtain

$$p(k) \leq L. \tag{6}$$

This concludes our study of feasible schedules and our characterization of the current workloads $\{[d(k), p(k)] : 1 \leq k \leq J\}$ that admit feasible schedules.

Before addressing order insertion we make a few observations about idle time and unused capacity in feasible schedules. When inserting a new job into a feasible schedule, it is not enough to know the total amount of unused capacity that exists on a given machine. Existing commitments limit the amount of unused capacity that can be made available in a given time period. The extreme cases of the earliest and the latest feasible schedules are fundamental in understanding how the unused capacity can be utilized. To this end, we introduce the concepts of minimum and maximum idle times.

Let

$$l(t) := t - \mathcal{L}(t) \text{ for } 0 \leq t \leq Z, \quad \text{and} \quad e(t) := t - L - \mathcal{E}(t - L) \text{ for } 0 \leq t \leq Z + L. \tag{7}$$

Note that $e(\cdot)$ and $l(\cdot)$ satisfy (1), so they are continuous. By Lemma 2 , for all feasible schedules $\sigma(\cdot)$,

$$l(t) \geq t - \sigma(t) \geq e(t + L) \quad \forall\, 0 \leq t \leq Z. \tag{8}$$

Thus $l(t)$ is the maximum idle time in $[0, t]$ in any feasible schedule, and $e(t + L)$ is the minimum.

The total unused capacity in $[0, Z]$, which is available to accommodate incoming orders, is $l(Z)$. Capacity is measured in hours (the time unit). However we must meet the commitments embodied in the current workload on machine m, and these commitments constrain the manner in which this capacity is used. Conceptually, we define a quantity called the c^{th} *atom of unused capacity*, defined for each c, $0 \leq c \leq l(Z)$.

Suppose that it is feasible to add a new operation to the current workload, and that at time t the operation will consume the c^{th} atom of unused capacity for each $c \in [h', h' + p']$. Clearly, (8) implies

$$e(t' + L) \leq h' \quad \text{and} \quad h' + p' \leq l(t'). \tag{9}$$

Let d be the due date of the new operation. Processing occurs at time t', so we must have $d - L \leq t' \leq d$. However $l(\cdot)$ and $e(\cdot)$ are non-decreasing, so

$$e(d) \leq h' \quad \text{and} \quad h' + p' \leq l(d). \tag{10}$$

If a new customer order will require J' production batches, it will presumably result in J' new operations on machine m. We define a *quotation* to consist of the operations $k, 1 \leq k \leq J'$, and the associated due dates $d'(k)$ and processing times $p'(k)$. We want to determine whether or not machine m has the capacity to add these operations to its current workload. We allocate unused capacity to the operations by specifying *capacity-assignment* numbers $\{h'(k), 1 \leq k \leq J'\}$. The c^{th} atom of unused capacity is allocated to operation k for each c, $h'(k) < c \leq h'(k) + p'(k)$. Following the logic that led to (10), a *feasible quotation* consists of a quotation and a set of capacity-assignment numbers that satisfy

$$e(d'(k)) \leq h'(k), \;\; h'(k) + p'(k) \leq l(d'(k)), \text{ and } \; h'(k) + p'(k) \leq h'(k+1). \tag{11}$$

The last inequality states that the c^{th} atom of unused capacity cannot be allocated to more than one operation.

Figure 2 shows a feasible quotation: $(p'(i), d'(i), h'(i))$ for $i = 1, 2, 3$.

If there are capacity-assignment numbers that make a given quotation feasible, then the following capacity-assignment numbers work. They assign the earliest available capacity to each operation.

$$h'(1) = e(d'(1)) \quad \text{and} \quad h'(k) = e(d'(k)) \vee (h'(k-1) + p'(k-1)), \;\; k > 1. \tag{12}$$

Theorem 1 *Assume that the current workload on machine m admits a feasible schedule. Then*

(i) Given a feasible quotation, there is a feasible schedule that completes all of the operations in both the current workload on machine m and in the feasible quotation.

(ii) For a given quotation, assume a feasible schedule completes all operations in both the quotation and the current workload. Then there are capacity assignment numbers that make the quotation feasible.

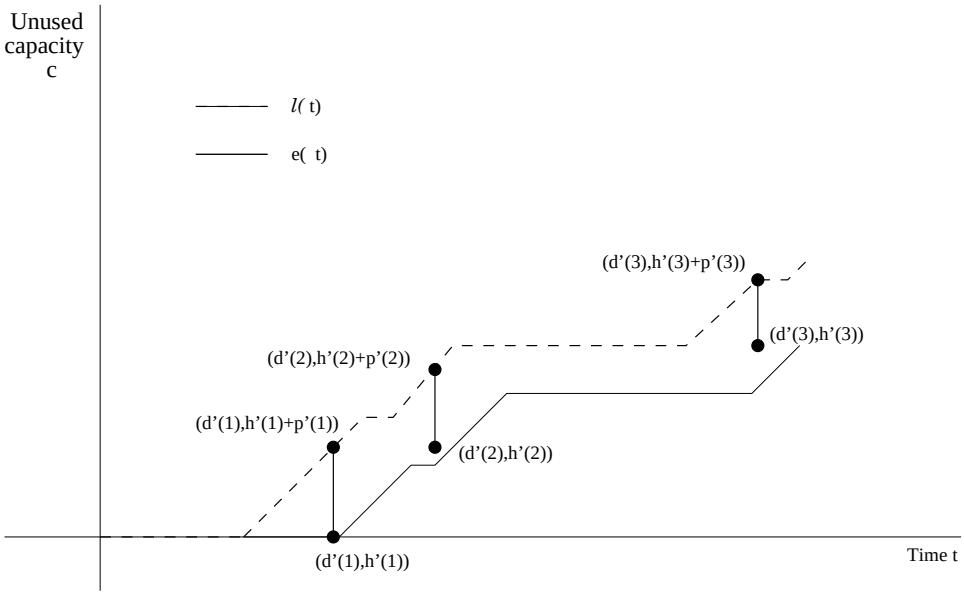

Figure 2: Feasible Quotation

3 Formulating the Order Insertion Problem

Using the results of section 2 we formulate the order insertion problem as a mixed integer programming problem. In section 2 time was treated as a continuous variable. We also focused on a single machine, and the dependence on m was often suppressed. In this section we consider all relevant machines. We assume that the current production plan is feasible, and that we want to insert a new client order for product j into the existing plan. Since we are inserting a single order, the dependence on j is often suppressed.

If t and all processing times, due dates and nominal lead times are integers, then $e(t)$ and $l(t)$ are also integers. If machine m is used to make product j, then θ_m is the nominal lead time to complete a batch of product j that has just left m. Thus $\theta_m = 0$ if m makes the finished product. Otherwise $\theta_m = L(m') + \theta_{m'}$ where m' is the (unique) successor of m in the routing of j. Recall that a client order consists of a collection of shipments i, $1 \leq i \leq s$ with shipment quantities $Q(i)$ and shipment dates $D(i)$. We define:

- $Q = \sum_{i=1}^{s} Q(i)$: the total quantity in the client order.
- sut_m, tpp_m: setup and processing times for product j on machine m.

We capture setup and holding costs in the following way. Figure 3 illustrates the holding costs associated with one client order of product j. In this figure the solid curve is the cumulative shipment quantity required

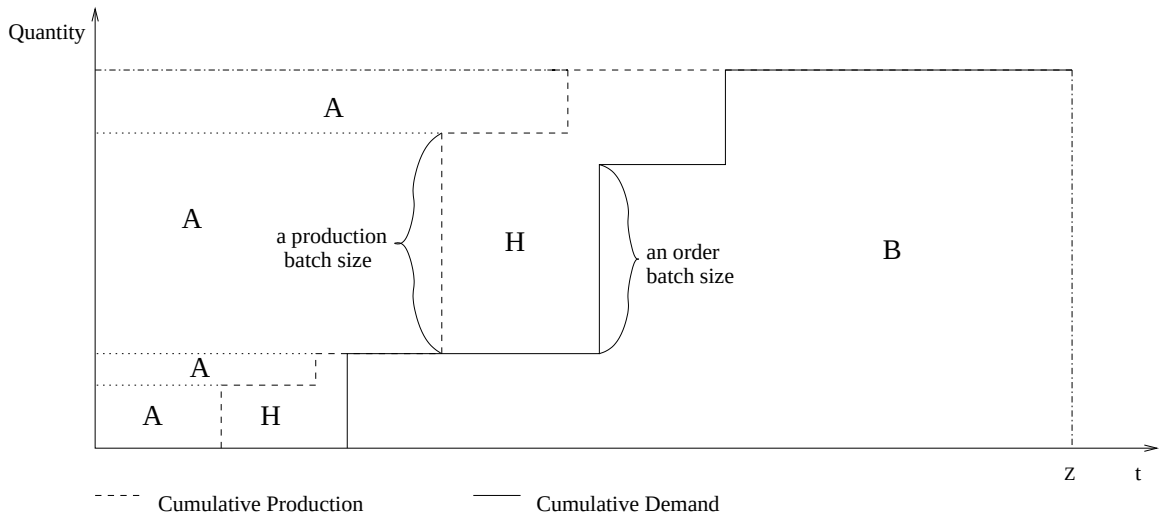

Figure 3: Cumulative Demand vs. Cumulative Production

by the client. The dashed curve is the cumulative production of j from new production batches. Since area B is constant (depends only on client order), the sum of areas H and A is also constant. Therefore, rather than minimizing the holding cost (area H), we maximize the rectangular areas indicated by A. This commonly-used technique allows us to capture the holding cost as a sum of rectangular areas. Thus the 'cost' of producing a batch of q units with completion time t is written as

$$C_{tq} = \left(\sum_{m \in M} S_m \right) - g \cdot t \cdot q \tag{13}$$

where M is the set of machines used to make product j, S_m is the setup cost on machine m, g is the holding cost per unit of time per unit, and the second term in (13) is g times the area of one of the "A" rectangles in Figure 3.

In our model, no new batch can be combined with an existing batch of the same product type to save setup costs because, we assume that batch identities are preserved throughout production steps. For example, by inserting a batch of part A on a machine before or after an existing batch of part A, setup costs cannot be avoided. In an extension of our model that captures such setup cost avoidance, the existing schedule must be represented in more detail than the current e and l based coarse representation, e.g., the detailed representation must include what batch is run at any time.

Let e_{mt} and l'_{mt} be discretizations of the functions $e(t)$ and $l(t)$ of section 2. Let $l_{mt} = l'_{m,t \wedge Z}$. Thus e_{mt} and l_{mt} are the cumulative idle time on machine m at time $t - L(m)$ in the earliest and at time t in the latest schedule, respectively. Furthermore $e_{m0} = l_{m0} = 0$, and $e_{m,Z+L(m)} = l_{mt}$ for $Z \leq t \leq Z + L(m)$. By

(8), $l_{mt} \geq e_{m,t+L(m)}$.

The order insertion problem is a collection of network flow problems, one for each machine, with side constraints. The order insertion network for machine m is $G_m = (N_m, A_m)$, where $(m,t,h) \in N_m$ if $e_{mt} \leq h \leq l_{mt}$ and $0 \leq t \leq Z + L(m)$. Node $(m,0,0)$ is the source and node $(m, Z+L(m), v_m)$ is the sink, where $v_m := l_{m,Z+L(m)}$. Three types of arcs are in A_m: production, advance-the-clock, and idle arcs.

- *production arc* (m,t,h,q). Traversing this arc means creating a new batch of q units. On machine m, this batch has due date $t + \theta_m$, and consumes the c-th atom of unused capacity for all $c \in (h, h + sut_m + tpp_m \cdot q)$. It connects nodes $(m, t-1, h)$ and $(m, t, h + sut_m + tpp_m \cdot q)$ for all $t \in \{1, \ldots, Z\}$, $h \in \{e_{mt}, \ldots, l_{m,t-1}\}$, and q such that $h + sut_m + tpp_m \cdot q \leq l_{mt}$.

- *advance-the-clock arc* (m,t,h). Traversing this arc means advancing to the next time period. It connects (m,t,h) to $(m,t+1,h)$ for all $t \in \{0,1,\ldots,Z+L(m)-1\}$ and $h \in \{e_{m,t+1}, \ldots, l_{mt}\}$.

- *idle arc* (m,t,h). Traversing this arc means deciding that for $c \in (h, h+1)$, the c^{th} atom of unused capacity will not be used for this client order. It connects (m,t,h) to $(m,t,h+1)$, $t \in \{1,\ldots,Z\}$, $h \in \{e_{mt}, \ldots, l_{m,t-1}\}$.

Figure 4 depicts an order insertion network which corresponds to a client order for one product type. All arcs are oriented from left to right, or upward. The setup and processing times are both one unit, so each operation requires at least two units of capacity.

Note that there is no production arc from node $(m,2,0)$ to either $(m,3,2)$ or $(m,3,3)$. These arcs would allocate the first atom of unused capacity to a batch with due date $3 + \theta_m$, which is not allowed because $e_{m3} = 1$. Also, nothing can be produced in period 10 because only one unit of capacity is available.

The arcs in a path specify the sizes and due dates of new production batches. For example, the path $(m,0,0)$,$(m,1,0)$,$(m,2,0)$,$(m,2,1)$,$(m,3,3)$,$(m,4,3)$,$(m,5,3)$,$(m,6,3)$,$(m,7,3)$,$(m,8,6)$,$(m,9,6)$,$(m,10,6)$, $(m,11,6)$ specifies one batch with quantity 1, processing time 2 and due date $3+\theta_m$, and another batch with quantity 2, processing time 3 and due date $8 + \theta_m$.

The binary decision variables π_{mthq}, α_{mth} and λ_{mth} denote the flows on production, advance-the-clock and idle arcs, respectively, in the order insertion network for machine m. The first set of constraints are conservation of flow constraints at the nodes.

$$\begin{aligned} \alpha_{m,t-1,h} + \lambda_{m,t,h-1} + \sum_q \pi_{m,t,h-sut-tpp\cdot q,q} &= \alpha_{m,t,h} + \lambda_{m,t,h} + \sum_q \pi_{m,t+1,h,q} \\ \forall\ (m,t,h) \in N_m,\ 0 < t &< Z + L(m) \end{aligned} \tag{14}$$

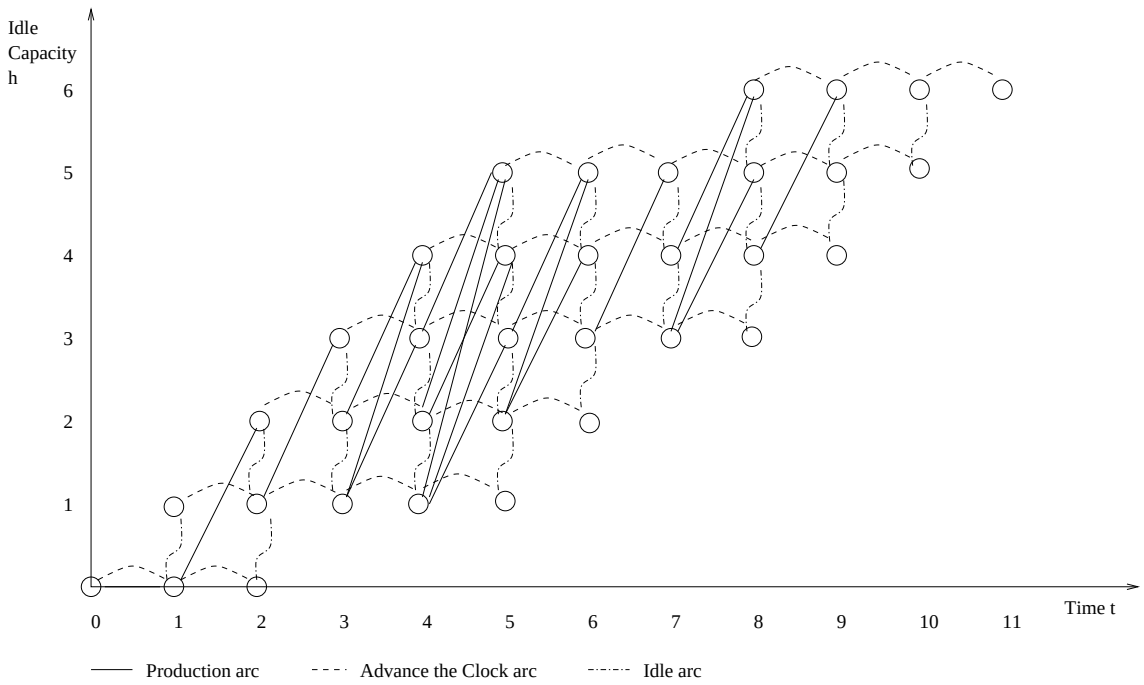

Figure 4: Order Insertion Network

The source node, $(m, 0, 0)$ has a supply of one unit. Thus

$$\forall\ m,\ \ 1 = \alpha_{m,0,0} + \sum_{q} \pi_{m,1,0,q}. \tag{15}$$

We define β_{tq} to be a binary decision variable which indicates whether a batch of q units with completion time t is produced or not. The following constraint provides that the batch is produced if and only if it is produced on all relevant machines and ensures that production batches retain their size and identity as they move from machine to machine:

$$\beta_{tq} = \sum_{h} \pi_{m,t-\theta_m,h,q} \quad \forall\ t, q, m \tag{16}$$

The earliest completion time of a batch is the maximum cumulative nominal lead time. Thus we can reduce the size of the formulation by starting the time at $t = \max_{m \in M}\{\theta_m + L(m)\}$.

The constraints introduced so far do not force production to keep pace with the client's requested shipment schedule. Hence, in the context of (14)-(16), we create machine $m = 0$, with $L(0) = 0$, $SUT_0 = 0, tpp_0 = 1$ and $\theta_0 = 0$. Producing more than the quantity Q ordered by the client is sub-optimal, so $l_{0t} = Q, \forall\ t$. We set e_{0t} equal to the cumulative quantity to be shipped by time $t-1$ (i.e., $e_{0t} = \sum_{j \le i} Q(j) \quad \forall\ i$ and $\forall\ t \in [D(i)+1, D(i+1)]$). This matches the solid line in figure 3, offset by 1 time unit.

We summarize the integer programming formulation (IP) as follows:

$$\max \sum_{t,q} \{qK - C_{tq}\}\beta_{tq}$$

$$\text{s.t.} \quad (14) - (16) \text{ hold } \forall\, m \in M \cup \{0\}, \quad \text{and} \quad \pi_{mthq}, \alpha_{mth}, \lambda_{mth}, \beta_{tq} \text{ are binary,}$$

where K is a large constant. The objective function intends to achieve two goals. The primary goal is to meet as much of the client's demand as possible. The secondary goal is to minimize the setup and holding costs while producing the maximal possible amount. The objective function is the sum of two terms corresponding to the two goals. To provide that the IP first achieves the first goal, we take a weighted sum. Namely, we give weight K to the primary goal versus weight 1 for the secondary goal.

This problem is NP-Hard (see Roundy et al (1999)). However our formulation appears to be extremely tight (see section 5). (IP) has approximately $|M|\ Z\ (T+U)$ rows and $|M|\ Z\ T\ U$ columns, where $T := \max\{L(m) : m \in M\}$ and U is the largest batch size allowed. By (33) we have $T \geq l_{mt} - e_{mt}$. By (6) we have $U \leq T$.

4 Heuristics

Directly solving (IP) is too slow for practical problems, so we propose and test five heuristics. The heuristics described in Subsections 4.2-4.5 all fit within a common architecture, and they all call the subroutine described in Subsection 4.1. At the highest level we perform a simple search for the optimal number Q of production batches. The heuristics in 4.2-4.5 all accept Q as an input, and perform a heuristic search for due dates of the batches. For each proposed set of due dates that they evaluate, they call a common subroutine that computes optimal batch sizes and the optimal cost.

In Subsection 4.6 we propose an LP-based heuristic, based on (IP), making 5 heuristics in total. In addition, we had intended to test an LP-roundoff heuristic that would start with the optimal solution to the continuous relaxation of (IP). However in all of our computational tests, either the LP relaxation for (IP) was too large to to fit into RAM, or the linear relaxation returned an integral solution.

4.1 Greedy Approach to Calculating Batch Sizes

Suppose that one of the heuristics in 4.2-4.5 has selected the number of batches $|\tau|$ and their due dates $\{t : t \in \tau\}$. Following (11) and (13), we determine batch sizes by solving

$$\text{(LP)} \qquad \min \sum_{t \in \tau} \left[\sum_{m \in M} S_m - g \cdot t \cdot q_t \right]$$

$$\text{s.t.} \qquad h_{mt} \geq e_{mt} \qquad \forall\, t \in \tau \;\; \forall\, m \in M \cup \{0\} \tag{17}$$

$$h_{mt} + tpp_m \cdot q_t + sut_m \leq l_{mt} \qquad \forall\, t \in \tau \;\; \forall\, m \in M \cup \{0\} \tag{18}$$

$$h_{mt} + tpp_m \cdot q_t + sut_m \leq h_{m\sigma(t)} \qquad \forall\, t \in \tau \;\; \forall\, m \in M \cup \{0\} \tag{19}$$

where q_t is a batch quantity, $\sigma(t)$ is the next due date in τ after t, and h_{mt} and q_t are decision variables.

A simple greedy algorithm solves (LP). We determine batch quantities one at a time, starting with the last batch and sweeping backwards in time. If t_0 is the last batch let $h_{\sigma(t_0)} = \infty$. Subsequently we set q_t equal to the maximum quantity we can produce, namely $\min\{(l_{mt} \vee h_{\sigma(t)}) - e_{mt} - sut_m)/tpp_m : m \in M \cup \{0\}\}$. We prove that this algorithm solves (LP) in Roundy et al (1999).

We now describe four heuristics for determining the timing of a fixed number of production batches.

4.2 Genetic Algorithm

Genetic algorithms are fairly standard by now; Gen and Cheng (1997) is a useful source. Our genetic algorithm uses breeding, mutation and immigration. An individual x consists of a strictly increasing sequence of due dates x_i, one for each batch. We make minor alterations to the standard breeding and mutation procedures to ensure that the resulting sequences of batch due dates are strictly increasing.

4.3 Simulated Annealing

For a general introduction to simulated annealing we refer the reader to van Laarhoven and Aarts (1987). Individuals x have the same structure they have in our genetic algorithm; they are sorted sequences of batch due dates. We use the following neighborhood topology: to get a neighbor of x we perturb one entry of x by ± 1. Formally, $\tilde{x}$ is a neighbor of x if and only if $\exists\, i_0 : |\tilde{x}_{i_0} - x_{i_0}| = 1$, and $\tilde{x}_i = x_i \;\forall i \neq i_0$.

4.4 Tabu Search

Our Tabu Search algorithm uses the same set of individuals and the same neighborhood topology as simulated annealing. The tabu set contains the solutions from the last k iterations. The next solution is the first solution we encounter that is in the neighborhood of the current solution, is not on the tabu list, and is an improvement over the current solution. To avoid getting stuck in a local optimum we diversify by randomly shifting the batch times if the entire neighborhood does not yield an improvement.

4.5 Randomized Local Search

The Randomized Local Search heuristic uses the same individuals and neighborhood topology as simulated annealing. We randomly select a pre-determined number of solutions and perturb each of them until we find a local minimum.

4.6 Single Machine Heuristic

Recall from section 3 that machine 0 models the client's requested shipment schedule. If we remove the side constraint (16) from (IP) it becomes a union of $|M|+1$ shortest path problems, one for each machine. The Single Machine Heuristic alters the order insertion network for machine 0 to capture many of the constraints imposed by other machines. It then finds a shortest path through the altered network. If this path generates a feasible solution to (IP), the heuristic terminates. Otherwise one of the arcs that enabled the infeasible solution is deleted, and the process iterates. A complete description follows.

Step 1: Eliminate the production arc $(0,t,h,q)$ from the order insertion network for machine 0 if one of the following holds:

- (C1): a batch cannot end at time t because t is less than the total cumulative lead time, i.e. $t < \max_{m\in M}\{\theta_m + L(m)\}$,
- (C2): there is inadequate capacity on one or more of the machines, i.e., for some $m \in M$ we have $sut_m + tpp_m \cdot q > l_{m,t-\theta_m} - e_{m,t-\theta_m}$.

Step 2: Find a shortest path from node $(0,0,0)$ to node $(0,z,v_0)$. The arc lengths are given by (13). If there is no path from $(0,0,0)$ to $(0,z,v_0)$ then stop; the heuristic failed to find a feasible solution.

Step 3: Determine production batches by setting $\beta_{tq} = 1$ if $(0, t, h, q)$ is on the shortest path for some h. Using the logic of (12) and (11) test these production batches for feasibility on each machine $m \in M$. If they are feasible on all machines then stop; a feasible solution has been obtained.

Step 4: Identify the production arcs that are in the shortest path and contributed to the infeasibility, delete one of them, and go to *Step 2*.

Note that if a feasible solution is found the first time *Step 3* is executed, it is necessarily optimal.

5 Computational Results

We want a robust heuristic that provides good solutions in a reasonable amount of time. The goal is to determine whether a new customer order can be accepted while the client is still on the phone, so a "reasonable" amount of time is something on the order of a minute. We create three versions of the Genetic, Simulated Annealing (SA), Tabu and Randomized Local Search heuristics by limiting each of them to 30 seconds, 1 minute, and 5 minutes. This gives us twelve heuristics to test, in addition to the Single Machine heuristic.

We create test problems having feasible solutions as follows. We first generate the data associated with a feasible schedule for a new quotation, assuming that no other work has been scheduled. Then we generate a current workload for each machine m by adding scheduled operations until m reaches a pre-specified level of utilization. Finally we remove the operations corresponding to the new quotation. A complete description is found in Roundy et al (1999). Our problem-generation code takes the following input parameters: the number of time periods, the number of machines, the size and depth of the BOM, and the number of shipments. We select a base case using realistic parameter settings, and we test a range of additional settings for each parameter. All computational tests are performed on a SPARC Ultra-1 Sun station.

We perform three sets of experiments. The Small and Large Problems are randomly generated. The Small Problems are sized so that we can solve the continuous relaxation of (IP). The last experiment uses data from the manufacturer.

For the Small Problems we generate 6 random problem instances of the base case, and 3 random instances for each of 15 variations on the base case. These problems have time horizons of approximately 70 periods. In all of the 51 problems, solving the continuous relaxation yields an integral solution. This indicates that the formulation of (IP) is very tight. Table 1 gives the average ratio of cost to optimal cost for each heuristic. The table is based on 49 instances; for the other two the optimal cost is 0, and the heuristics are optimal or

nearly so.

	Sec.	$\frac{Cost}{Opt}$	**Sec.**	$\frac{Cost}{Opt}$	**Sec.**	$\frac{Cost}{Opt}$
GA	30	1.04	60	1.04	300	1.04
Rand	30	1.07	60	1.06	300	1.06
SA	30	1.10	60	1.11	300	1.13
Tabu	30	1.39	60	1.39	300	1.39
1 Mach						1.00

Table 1: Small Problems, Deviation from Optimality

For the Large Problems we use thirty-five parameter settings to generate sample problems. The parameters are chosen to span a wide range of potential situations. Each parameter setting is replicated 20 times, for a total of 700 problems. We calculate the fraction of problems where each heuristic found a feasible solution (see Table 2). We also report the average ratio of the heuristic cost to the lowest-cost solution produced by any of the 13 algorithms. For each time limit we only average problems in which all four heuristics find a feasible solution. The running time for the Single Machine heuristic has mean 320 seconds, standard deviation 300 seconds and maximum 1242 seconds. We include it in the 300-second group.

The Randomized Local Search and Tabu heuristics are not competitive. The Single Machine heuristic is not as effective in finding feasible solutions as the Simulated Annealing and Genetic heuristics, but when it does the solutions are of slightly better quality. In 82% of the problems where the Single Machine heuristic does not find a feasible solution, AMPL runs out of memory. This could be improved by re-coding.

For the Genetic and Simulated Annealing heuristics the improvements in performance (i.e., (i) percentage of instances for which a feasible solution is found, (ii) the average ratio of the heuristic cost to the lowest-cost solution) from 30 to 60 to 300 seconds are significant. Practically speaking one would probably want a version between 60 and 300 seconds.

Finally, we take a feasible schedule from the manufacturer's planning system that includes 421 products and 43 workcenters. The length of the schedule was 1400 time periods. Since the factory runs three 8-hour shifts 5 days per week, the schedule covers about three months. Setup times and processing times per 100 pieces are both rounded to an integer number of hours. We form seven sample insertion problems by removing multi-batch quotes for seven different products from the schedule. The tests are summarized in Tables 3 and 4. All heuristics find feasible solutions with the exception of the Single Machine Heuristic for problem 2. Cost

Seconds	30	30	30	60	60	60	300	300	300
	%	$\frac{Cost}{Best}$	**StD**	%	$\frac{Cost}{Best}$	**StD**	%	$\frac{Cost}{Best}$	**StD**
GA	83.6	1.32	0.76	96.1	1.23	0.38	99.7	1.10	0.18
Rand	87.4	1.49	0.69	89.9	1.41	0.65	95.7	1.27	0.55
SA	89.0	1.29	0.55	95.3	1.18	0.32	98.9	1.07	0.13
Tabu	74.3	1.49	0.91	72.6	1.36	0.68	85.3	1.23	0.37
1 Mach							60.9	1.05	0.27
# Probs	700	485	485	700	494	494	700	388	388

Table 2: Large Problems, % Found a Feasible Solution, Avg. and St. Dev. of (Heur. Cost)/(Best Cost)

Problem #	**1**	**2**	**3**	**4**	**5**	**6**	**7**
# of Shipments	3	8	4	2	3	7	2
# of Machines	7	2	2	2	3	3	5
Single Machine Feasible?	yes	no	yes	yes	yes	yes	yes
Single Machine Provably Optimal?	yes	no	no	yes	yes	no	no
Single Machine Time	18		385	12	85	1112	796

Table 3: Problems from Industrial Data

Seconds	30	60	300
GA	1.36	1.33	1.15
Rand	1.39	1.39	1.22
SA	1.31	1.19	1.17
Tabu	1.60	1.53	1.47
1 Mach			1.00

Table 4: Industrial Data, Avg. of (Heur. Cost)/(Best Cost)

performance is consistent with earlier tests, with the exception of a stronger-than-anticipated performance by the Single-Machine Heuristic on cost. Quotations with a greater numbers of shipments seem to be more difficult.

6 Conclusions

We model a job insertion problem and prove it to be NP-hard (see Roundy et al (1999) for NP-hardness proof). The continuous relaxation of our integer programming formulation is solved for 51 randomly generated problem instances. In every case the solution is integral. The formulation seems to be remarkably tight.

We develop and test a number of heuristics, using both randomly-generated and real-world problem instances. The genetic, simulated annealing and single-machine heuristics all look promising.

The greatest limitation of our heuristics is that we treat time as a discrete variable. All operation start times and processing times are an integral number of hours. This is not adequate for a realistic industrial setting unless time periods are shortened. Continuous-time versions of our heuristics would be feasible. In discrete time, without increasing the number of time periods it would not be commercially available unless processing times could be modelled as multiples of an hour. Adapting the heuristics to handle a greater number of time periods or to work in continuous time is one of the main areas of future research. Also, we did not consider overtime or allowing for some late jobs, and we believe those extensions would be good directions to move in. Other areas for future research include developing approximation algorithms, a study of facets and cuts for (IP).

7 Acknowledgements

We are greatly indebted to Peter L. Jackson, who worked with the parts manufacturer and provided insights and assistance with the industrial data.

8 Appendix

Lemma 2 A feasible schedule exists if and only if, for all $j = 0, ..., J$ and $k = 1, ..., J$, $d(j) - L \leq d(k)$ implies

$$T(k) + (d(j) - L - d(k)) \leq T(j-1). \tag{4}$$

If $\sigma(\cdot)$ is a feasible schedule then $\mathcal{L}$ and $\mathcal{E}$ are both feasible, and $\mathcal{L}(t) \le \sigma(t) \le \mathcal{E}(t) \quad \forall\, 0 \le t \le Z$.

Proof. Assume that $\sigma(\cdot)$ is feasible and that $d(j) - L \le d(k)$. Then $T(k) \le \sigma(d(k)) + T(j-1) - \sigma(d(j) - L) \le T(j-1) + d(k) - d(j) + L$, where the first inequality follows from (2) and the second from (1). This proves necessity. To complete the proof, assume that (4) holds whenever $d(j) - L \le d(k)$. Note that $\mathcal{L}'(0,t) = 0$, $\mathcal{E}'(0,t) = t$, and $\mathcal{L}'(J+1,t) = T(J)$. Clearly, $[\mathcal{L}'(k,t) \vee \mathcal{L}'(0,t)]$ and $[\mathcal{E}'(k,t) \wedge \mathcal{E}'(0,t)]$ satisfy (1). It is easy to verify that maxima (resp., minima) of a set of functions which satisfy (1) must also satisfy (1). Thus $\mathcal{L}(\cdot)$ and $\mathcal{E}(\cdot)$ are schedules.

We now prove that

$$\mathcal{L}(t) \le \mathcal{E}(t) \quad \forall\, t. \tag{20}$$

By (3) it suffices to show that $\mathcal{L}'(k,t) \le \mathcal{E}'(j,t) \quad \forall\, j,k,t$. The shapes of $\mathcal{L}'(k,\cdot)$ and $\mathcal{E}'(j,\cdot)$ imply that it suffices to show that $T(k) = \mathcal{L}'(k, d(k))$ is less than or equal to $\mathcal{E}'(j, d(k)) = A := [T(j-1) + (d(k) + L - d(j))^+]$. If $d(k) + L - d(j) \le 0$ then $j > k$, so $T(k) \le T(j-1) = A$. If $d(k) + L - d(j) \ge 0$ then by (4), $T(k) \le T(j-1) + d(k) + L - d(j) = A$. Thus $\mathcal{L}(t) \le \mathcal{E}(t)\ \forall\, t$.

By (4) we have $T(k) = \mathcal{L}'(k, d(k)) \le \mathcal{L}(d(k)) \le \mathcal{E}(d(k))$ and $T(k-1) = \mathcal{E}'(k, d(k) - L) \ge \mathcal{E}(d(k) - L) \ge \mathcal{L}(d(k) - L)$. Thus both $\mathcal{L}(\cdot)$ and $\mathcal{E}(\cdot)$ satisfy (2), and are feasible. All that remains is to show that $\mathcal{L}'(k,t) \le \sigma(t) \le \mathcal{E}'(k,t) \quad \forall\, k,t$. Considering (1) and the shape of $\mathcal{L}'(k,t)$ and $\mathcal{E}'(k,t)$, it suffices to show that $\sigma(d(k)) \ge \mathcal{L}'(k, d(k)) = T(k)$ and $\sigma(d(k) - L) \le \mathcal{E}'(k, d(k) - L) = T(k-1)$. This follows from (2). ■

We can write $\mathcal{L}(\cdot)$ and $\mathcal{E}(\cdot)$ as

$$\mathcal{L}(t) := \max_{k:0 \le k \le J+1} \{\mathcal{L}'(k,t)\} = \max_{d(k) \le t} \{T(k)\} \vee \max_{d(k) \ge t} \{T(k) - (d(k) - t)\},\ 0 \le t \le Z + L, \tag{21}$$

$$\mathcal{E}(t) := \min_{k:1 \le k \le J+1} \{\mathcal{E}'(k,t)\} = \min_{d(k) - L \le t} \{t - d(k) + L + T(k-1)\} \wedge \min_{d(k) - L \ge t} \{T(k-1)\},\ -L \le t \le Z. \tag{22}$$

Thus $\mathcal{L}(t) \ge 0, t \ge \mathcal{E}(t)$, and $T(J) \ge \mathcal{E}(t)$. Also, note that by (2), (3), and (20), $T(J) \ge \mathcal{E}(Z) \ge \mathcal{L}(Z) \ge T(J)$, so $\mathcal{E}(Z) = \mathcal{L}(Z) = T(J)$. If $d(k) \le t < d(k+1)$ then

$$\mathcal{L}(t) = T(k) \vee \max_{k_* > k} \{T(k_*) - (d(k_*) - t)\} = \max_{k_* \ge k} \{T(k_*) - [d(k_*) \vee t] + t\}. \tag{23}$$

Similarly, if $d(j-1) < t \le d(j)$ then

$$\mathcal{E}(t - L) = \min_{j_* < j} \{t - d(j_*) + T(j_* - 1)\} \wedge T(j-1) = \min_{j_* \le j} \{t - [d(j_*) \wedge t] + T(j_* - 1)\}. \tag{24}$$

Furthermore

$$d(k) \le t < d(k+1) \quad \stackrel{(23)}{\Longrightarrow} \quad \mathcal{L}(t) \ge T(k) \quad \Longrightarrow \quad l(t) \le t - T(k) \text{ and} \tag{25}$$

$$d(j-1) < t \le d(j) \quad \stackrel{(24)}{\Longrightarrow} \quad \mathcal{E}(t-L) \le T(j-1) \quad \Longrightarrow \quad e(t) \ge t - L - T(j-1). \tag{26}$$

Lemma 3 *Suppose that* $\{d'(k), p'(k), h'(k) : 1 \le k \le J'\}$ *is a feasible quotation, and let* $T'(k) := \sum_{k_* \le k} p'(k_*)$. *Then* $T'(k) \le p'(k) + h'(k)$. *If* $j \le k$ *then* $T'(k) - T'(j-1) \le p'(k) + h'(k) - h'(j)$.

Proof. The result is obtained by summing the last inequality in (11) with successive values of k. ■

Theorem 1 Assume that the current workload on machine m admits a feasible schedule. Then

(i) Given a feasible quotation, there is a feasible schedule that completes all of the operations in both the current workload on machine m, and the feasible quotation.

(ii) For a given quotation, assume a feasible schedule completes all operations in both the quotation and the current workload. Then there are capacity assignment numbers that make the quotation feasible.

Proof. To prove (i) we combine the operations $\{d(k), p(k) : 1 \le k \le J\}$ in the current workload and the operations $\{d'(k'), p'(k') : 1 \le k' \le J'\}$ in the feasible quotation into a single set, and re-index them as $\{d''(k''), p''(k'') : 1 \le k'' \le J''\}$ where $J'' = J + J'$ and $d''(k'') \le d''(k''+1)$. For convenience only, we assume that $d''(k'') < d''(k''+1)$. We define $k(k'')$, $k'(k'')$, $j(j'')$, and $j'(j'')$ as follows:

$$\begin{array}{rcccl}
d(k(k'')) & \le & d''(k'') & < & d(k(k'')+1) \\
d'(k'(k'')) & \le & d''(k'') & < & d'(k'(k'')-1) \\
d(j(j'')) & \ge & d''(j'') & > & d(j(j'')-1) \\
d'(j'(j'')) & \ge & d''(j'') & > & d'(j'(j'')-1)
\end{array}$$

We must show that (4) holds if the workload on machine m is

$$\{d''(k''), p''(k'') : 1 \le k'' \le J''\}, \tag{27}$$

and if j'', k'' satisfy $d''(j'') - L \le d''(k'')$. Let $k := k(k'')$, $k' := k'(k'')$, $j := j(j'')$, and $j' := j'(j'')$. Then

$$T(k) + T'(k') = T''(k) \text{ and } T(j-1) + T'(j'-1) = T''(j''-1). \tag{28}$$

Let $B := T''(k'') - T''(j''-1) + d''(j'') - L - d''(k'')$. Then (4) is equivalent to $0 \ge B$. If $k'' < j''$ then the sum of the first two terms of B is less than or equal to zero, and our assumption implies (4). If $k'' = j''$ and

$d''(k'') = d(k)$ then (6) implies (4). Thus we assume that $k'' \geq j''$, and that if $k'' = j''$ then $d''(k'') = d'(k')$. Note that $k' \geq j'$.

By Lemma 3, (11), the monotonicity of $l(\cdot)$ and $e(\cdot)$, and (25)-(26),

$$\begin{aligned} T'(k') - T'(j'-1) \;&\leq\; p'(k') + h'(k') - h'(j') \;\leq\; l(d'(k')) - e(d'(j')) \\ \leq\; l(d''(k'')) - e(d''(j'')) \;&\leq\; [d''(k'') - T(k)] - [d''(j'') - L - T(j-1)]. \end{aligned}$$

Consequently by (28),

$$B \;=\; [T(k) - T(j-1)] + [T'(k') - T'(j'-1)] + [d''(j'') - L - d''(k'')] \;\leq\; 0.$$

This proves the first assertion.

To prove the second assertion we assume that the current workload $\{d(k), p(k) : 1 \leq k \leq J\}$ admits a feasible schedule, that $\{d'(k'), p'(k') : 1 \leq k' \leq J'\}$ is a quotation, and that the workload given by (27) admits a feasible schedule. We need to show that the quotation is feasible. Let $h'(\cdot)$ be given by (12). Clearly the first and third assertions in (11) hold. It suffices to show that $h'(k') + p'(k') \leq l(d'(k')) \quad \forall\, k'$.

A simple induction proof based on (12) implies that there is a $j' \leq k'$ such that

$$h'(k') + p'(k') = e(d'(j')) + \sum_{k'_* = j'}^{k'} p'(k'_*) = e(d'(j')) + T'(k') - T'(j'-1).$$

Thus it suffices to show that

$$\begin{aligned} 0 \;&\geq\; e(d'(j')) - l(d'(k')) + T'(k') - T'(j'-1) \\ &=\; d'(j') - L - \mathcal{E}(d'(j') - L) - d'(k') + \mathcal{L}(d'(k')) + T'(k') - T'(j'-1). \end{aligned}$$

We define k, k'', j and j'' by

$$d(k) < d'(k') = d''(k'') < d(k+1) \quad \text{and} \quad d(j-1) < d'(j') = d''(j'') < d(j). \tag{29}$$

Recall that $j' \leq k'$, so $j'' \leq k''$. By (23) and (24), it suffices to show that

$$\begin{aligned} 0 \;\geq\; & d'(j') - L - d'(k') + T'(k') - T'(j'-1) - \min_{j_* \leq j} \{d'(j') - [d(j_*) \wedge d'(j')] + T(j_*-1)\} \\ & + \max_{k_* \geq k} \{T(k_*) - [d(k_*) \vee d'(k')] + d'(k')\} \end{aligned} \tag{30}$$

Consequently it suffices to show that for all $k_* \geq k$ and $j_* \leq j$,

$$0 \;\geq\; [d(j_*) \wedge d'(j')] - L - [d(k_*) \vee d'(k')] + T'(k') - T'(j'-1) + T(k_*) - T(j_*-1). \tag{31}$$

We define k'_*, k''_*, j'_* and j''_* by

$$d'(k'_*) < d(k_*) = d''(k''_*) < d'(k'_* + 1) \text{ and } d'(j'_* - 1) < d(j_*) = d''(j''_*) < d'(j'_*) \tag{32}$$

We claim that

(i) If $k_* > k$ then $k''_* > k''$ and $k'_* \geq k'$. If $k_* = k$ then $k''_* < k''$ and $k'_* < k'$

(ii) If $j_* < j$ then $j''_* < j''$ and $j'_* \leq j'$. If $j_* = j$ then $j''_* > j''$ and $j'_* > j'$.

If $k_* > k$ then $k_* \geq k + 1$. By (29) and (32), $d'(k'_* + 1) > d''(k''_*) = d(k_*) \geq d(k + 1) > d'(k') = d''(k'')$, so the first assertion holds. If $k_* = k$ then by (29) and (32), $d''(k'') = d'(k') > d(k) = d(k_*) = d''(k''_*) > d'(k'_*)$. Hence, $k''_* < k''$, $d(k_*) < d'(k')$. But (32) implies that $d(k_*) > d'(k'_*)$, so $k'_* < k'$. This establishes (ii). The proof of (ii) is similar.

A consequence of (29), (32) and this claim is the following:

$$T''(k''_* \vee k'') = T'(k'_* \vee k') + T(k_*) \quad \text{and} \quad T''((j''_* \wedge j'') - 1) = T'((j'_* \wedge j') - 1) + T(j_* - 1).$$

Note that Lemma 2 holds for (27). Since $j'' \leq k''$, we can apply Lemma 2 for $k''_* \vee k''$ and $j''_* \wedge j''$. Thus,

$$\begin{aligned} 0 &\geq d''(j''_* \wedge j'') - L - d''(k''_* \vee k'') + T''(k''_* \vee k'') - T''(j''_* \wedge j'' - 1) \\ &= [d(j_*) \wedge d'(j')] - L - [d(k_*) \vee d'(k')] + T'(k'_* \vee k') - T'((j'_* \wedge j') - 1) + T(k_*) - T(j_* - 1). \end{aligned}$$

By the monotonicity of $T'(\cdot)$, (31) holds. ■

Finally, if $d(k-1) \leq t \leq d(k)$ then $\mathcal{L}(t) - \mathcal{E}(t-L) \geq \mathcal{L}(k-1,t) - \mathcal{E}(k,t-L) = T(k-1) - T(k-1) = 0$, i.e.,

$$\mathcal{E}(t) \leq \mathcal{L}(t+L). \tag{33}$$

References

J. Adams, E. Balas, and D. Zawack. 1988. The Shifting Bottleneck Procedure for Job Shop Scheduling. *Management Science*, **34**, 391–401.

C. Akkan. 1996. Overtime Scheduling: An Application in Finite Capacity Real Time Scheduling. *Journal of the Operational Research Society*, **47**, 1137–1149.

K.R. Baker. 1974, *Elements of Sequencing and Scheduling.* John Wiley & Sons, New York.

J. Blazewicz, W. Domschke, and E. Pesch. 1996. The Job Shop Scheduling Problem and New Solution Techniques. *European Journal of Operational Research*, **93**, 1–33.

J.H. Bookbinder and A.I. Noor. 1985. Setting Job-shop Due Dates with Service Level Constraints. *Journal of the Operational Research Society*, **36**, 1017–1026.

T.C.E. Cheng and M.C. Gupta. 1989. Survey of Scheduling Research Involving Due Date Determination Decisions. *European Journal of Operations Research*, **38**, 156–166.

R.W. Conway, W.L. Maxwell, and L.W. Miller. 1967, *Theory of Scheduling*. Addison Wesley.

S. Dauzere-Peres and J.B. Lasserre. 1997. Lot Streaming in Job-Shop Scheduling. *Operations Research*, **45**, 584–595.

I. Duenyas. 1995. Single Facility Due Date Setting with Multiple Customer Classes. *Management Science*, **41**, 608–619.

I. Duenyas and W.J. Hopp. 1995. Quoting Order Lead Times. *Management Science*, **41**, 43–57.

S. T. Enns. 1996. Finite Capacity Scheduling Systems: Performance Issues and Comparisons. *Computers & Industrial Engineering*, **30**, 727–739.

S. T. Enns. 1998. Lead Time Selection and the Behaviour of Work Flow in Job Shops. *European Journal of Operational Research*, **109**, 122–136.

L. Gelders and P.R. Kleindorfer. 1974. Co-ordinating Aggregate and Detailed Scheduling Decisions in the One Machine Job Shop: Part I Theory. *Operations Research*, **22**, 46–60.

L. Gelders and P.R. Kleindorfer. 1975. Co-ordinating Aggregate and Detailed Scheduling Decisions in the One Machine Job Shop: Part II Computation and Structure. *Operations Research*, **23**, 312–324.

M. Gen and R. Cheng. 1997, *Genetic Algorithms and Engineering Design*. John Wiley & Sons, Inc.

L.S. Gould. 1998. Introducing APS: Getting Production in Lock Step with Customer Demand. *Automotive Manufacturing & Production*, **10**, 54–58.

S. Hill. 1998. SAP's Apparel/Footwear Solution: Does It Have All the Answers? *Apparel Industry Magazine*, **59**, 62–63.

C.A. Holloway and R.T. Nelson. 1974. Job Shop Scheduling with Due Dates and Overtime Capability. *Management Science*, **21**, 68–78.

W.J. Hopp and M.L. Spearman. 1996, *Factory Physics*. Irwin, Chicago.

P. Keskinocak, R. Ravi, and S. Tayur. Scheduling and Reliable Lead Time Quotation for Orders with Availability Intervals and Lead Time Sensitive Revenues. Working paper, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1999.

P. Kevin. 1996. Right From the Get-go. *Manufacturing Systems*, **14**, 34.

J. Kim and Y. Kim. 1995. Simulated Annealing and Genetic Algorithms for Scheduling Products with Multi-Level Product Structure. *Computers and Operations Research*, **23**, 857–868.

J. King. 1996. Software Delivers Customized PC Orders, Tracking. *Computerworld*, **30**, 45.

M.R. Lambrecht, P.L. Ivens, and N.J. Vandale. 1998. ACLIPS: A Capacity and Lead Time Integrated Procedure for Scheduling. *Management Science*, **44**, 1549–1561.

E.L. Lawler, J.K. Lenstra, A.H.G. Rinnoy Kan, and D.B. Shmoys. 1993, *Sequencing and Scheduling: Algorithms and Complexity*, volume 4. Elsevier Science Publishers.

J.E. Layden. 1996. A Rapidly Changing Landscape. *Manufacturing Systems*, **March Issue A10**.

A.G. Loerch. 1990. A New Approach to Production Planning Scheduling and Due-Date Quotation in Manufacturing Systems. Ph.D. Dissertation, Cornell University, Ithaca, New York.

A.G. Loerch and J.A. Muckstadt. 1994. An Approach to Production Planning and Scheduling in Cyclically Scheduled Manufacturing Systems. *International Journal of Production Research*, **32**, 851–871.

H. Luss and M.B. Rosenwein. 1993. A Due Date Assignment Algorithm for Multiproduct Manufacturing Facilities. *European Journal of Operational Research*, **65**, 187–198.

R. Roundy, M. Çakanyildirim, D. Chen, P. Chen, M. Freimer, P.L. Jackson, and V. Melkonian. Capacity-Driven Acceptance of Customer Orders for a Multi-State Batch Manufacturing System: Models and Algorithms. Technical Report TR1233, Cornell University, Ithaca, New York, 1999.

S.A. Slotnick and T.E. Morton. 1996. Selecting Jobs for a Heavily Loaded Shop with Lateness Penalties. *Computers and Operations Research*, **23**, 131–140.

M.L. Spearman and R.Q. Zhang. 1999. Optimal Lead Time Policies. *Management Sci.*, **45**, 290 – 295.

S.G. Taylor and S.F. Bolander. 1997. Process Flow Scheduling: Past, Present and Future. *Production and Inventory Management Journal*, **38**, 21.

P.J.M. van Laarhoven and E.H.L. Aarts. 1987, *Simulated Annealing: Theory and Applications.* D. Reidel Publishing Company.

P.J.M. van Laarhoven, E.H.L Aarts, and J.K. Lenstra. 1992. Job Shop Scheduling by Simulated Annealing. *Operations Research*, **40**, 113–125.

F.A.W. Wester, J. Wijngaard, and W.H.M. Zijm. 1992. Order Acceptance Strategies in a Production-to-order Environment with Setup Times and Due-dates. *International Journal of Production Research*, **30**, 1313–1326.