Available online at www.sciencedirect.com

ScienceDirect

Applied Mathematics and Computation 185 (2007) 11–18

APPLIED MATHEMATICS AND COMPUTATION

www.elsevier.com/locate/amc

Customer order scheduling on a single machine with family setup times: Complexity and algorithms

Erdal Erel [a,*], Jay B. Ghosh [b]

[a] *Faculty of Business Administration, Bilkent University, 06800 Bilkent, Ankara, Turkey*
[b] *Apratech LLC, Los Angeles, USA*

Abstract

We consider a situation where C customers each order various quantities (possibly zero in some cases) of products from P different families, which can be produced on a continuously available machine in any sequence (requiring a setup whenever production switches from one family to another). We assume that the time needed for a setup depends only on the family to be produced immediately after it, and we follow the *item availability* model (which implies that all units are ready for dispatch as soon as they are produced). However, an order is shipped only when all units required by a customer are ready. The time from the start (time zero) to the completion of a customer order is called the *order lead time*. The problem, which restates the original description of the *customer order scheduling* problem, entails finding a production schedule that will minimize the total order lead time. While this problem has received some attention in the literature, its complexity status has remained vexingly open. In this note, we show for the first time that the problem is strongly NP-hard. We proceed to give dynamic programming based exact solution algorithms for the general problem and a special case (where C is fixed). These algorithms allow us to solve small instances of the problem and understand the problem complexity more fully. In particular, the solution of the special case shows that the problem is solvable in polynomial time when C is fixed.

Keywords: Machine scheduling; Computational complexity; Dynamic programming

1. Introduction

Suppose that there are C customers who each order from some or all of P different product families and that customer c orders t_{cp} ($t_{cp} \geqslant 0$) units from family p. Assume, without loss of generality, that a unit from each product family requires a unit processing time on a continuously available machine and that it is immediately ready for dispatch upon its completion (this is the *item availability* assumption). Let the setup time needed to start a production lot of family p on the machine be s_p. If we define σ to be a production schedule for the machine and $L_c(\sigma)$ to be the *order lead time* in σ for customer c (this equals the completion time of

* Corresponding author.
E-mail address: erel@bilkent.edu.tr (E. Erel).

0096-3003/$ - see front matter
doi:10.1016/j.amc.2006.06.086

customer c's last produced unit), then our objective is to find a schedule that will minimize the total customer order lead time given by $\sum_{1\leqslant c\leqslant C} L_c(\sigma)$.

It is easily seen that an optimal schedule, ignoring the setups, admits no machine idle time. A schedule can thus be thought of as being composed solely of setups and production lots that follow them. Similarly, it can be shown that the entire requirement of customer c from product family p is processed contiguously within a single production lot of family p in an optimal schedule. We can thus consider c's requirement for p as a single aggregate unit with a processing time of t_{cp} (which may be 0). We let N_p represent the number of customers ordering an aggregate unit from product family p (i.e., $N_p = |\{c: t_{cp} > 0, 1 \leqslant c \leqslant C\}|$) and N_c the number of product families ordered by customer c (i.e., $N_c = |\{p: t_{cp} > 0, 1 \leqslant p \leqslant P\}|$). The total number of aggregate units on order is given by $N = \sum_{1\leqslant p\leqslant P} N_p = \sum_{1\leqslant c\leqslant C} N_c$. For obvious reasons, we require that $N_p > 0$ for all p and $N_c > 0$ for all c. Based on the values of N_p and N_c, we get two versions of the above problem, which has been called the *customer order scheduling* problem (COS). If $N_p = C$ and $N_c = P$ for all p and c, we have the *full order* (F-Order) version. But if $N_p < C$ and $N_c < P$ for at least one p and c, we have the *less than full* order (LTF-Order) version. These versions often warrant different treatments.

Scheduling in presence of multiple customer orders (each of which may include various product families in different quantities), as above, is quite common in the *make to order* and *assemble to order* production environments. It is in this context that Julien and Magazine [17] have originally stated the COS problem. They have analyzed the structure of optimal schedules, presented a polynomial time algorithm for the case with a given order processing sequence and two product types, and identified some conditions under which the general problem is easy to solve. Subsequently, Julien [16] has shown that a particular class of schedules is dominant for problem instances where the processing times as well as the setup times are *agreeable* in a certain sense (the optimal schedule in this case can be found in polynomial time). He has successfully exploited the corresponding algorithm to generate an effective heuristic solution and a useful lower bound for the general problem. However, much of the above work relates to the F-Order version of COS and lack generality. Erel and Ghosh [9] have given an $O(P^2C^P)$ time algorithm for both the F-Order and LTF-Order versions of COS for the case where an order sequence is given. (Notice that the meaning of an order sequence becomes ambiguous in the LTF-Order version. One can talk about either an order processing sequence or an order delivery sequence. The Erel–Ghosh algorithm [9] works in both cases.)

Problems similar to COS surface when one considers scheduling multi-operation jobs on a single machine. Early examples of such problems, which are seen as special cases of COS, can be found in the works of Baker [3], Coffman et al. [7] and Vickson et al. [24], among others. More recently, Gerodimos et al. [11] have addressed a problem that is in fact equivalent to COS. They have given an essentially $O(P)$ time algorithm for the F-Order version of COS where all units from a product family are to be produced within a single setup (consistent with the *group technology* assumption). (A similar solution has also been given by Erel and Ghosh [9] earlier.) They have also given an essentially $O(P^2C^P)$ time algorithm for COS that is optimal subject to an *agreeability* assumption on the processing times. This assumption is rather restrictive and may in fact hinder the algorithm's application to the LTF-order version of COS. However, the algorithm is similar to that of Erel and Ghosh [9] and can be easily modified to solve COS if an order sequence is given.

Extending standard notation, maintaining consistency with Gerodimos et al. [11] and retaining above all our focus on the scheduling of customer orders (rather than multi-operation jobs), we use $1|s_p, \cos|\sum_c L_c$ to designate our problem. To sum up what is known about it to date, we first note that its complexity status remains vexingly open. We know trivially that if the setups are sequence-dependent, then the problem is strongly NP-Hard. We also know (Erel and Ghosh [9], Gerodimos et al. [11]) that if the setups are not needed or if the setup times are *sufficiently small*, all units required by a customer can be processed together and the customer orders themselves can be processed in non-decreasing order of $\sum_{1\leqslant p\leqslant P} t_{cp}$ (i.e., the total processing time needed by a customer); the ordering of the production lots for a given customer is immaterial if there are no setups or handled easily through an algorithm of Julien [16] if there are setups. The solution times are essentially $O(C)$ in the first case and $O(CP)$ in the second. Next, we know that if only one setup per product family is allowed (as discussed above) or if the setup times are *sufficiently large* forcing a single setup per family, the F-Order version of the problem can be solved in essentially $O(P)$ time. However, nothing is known about the exact solution of the LTF-Order version of the problem in this situation. As for the exact solution of the general problem, a polynomial time algorithm is known for

the F-Order version (Julien [16]) under very restrictive assumptions. If P is fixed, another polynomial time algorithm (Gerodimos et al. [11]) is available for the same version under somewhat less restrictive assumptions. Similarly, if an order sequence is given and P is fixed, a polynomial time algorithm (Erel and Ghosh [9], Gerodimos et al. [11]) is also possible for both the F-Order and LTF-Order versions of the general problem (without further assumptions).

In the sequel, we address some of the unresolved issues. We show for the first time that $1|s_p, \cos|\sum_c L_c$ is strongly NP-Hard. We do this by proving the NP-Hardness of the LTF-Order version of the problem when the setup times are sufficiently large (thus forcing a single setup per family). We then give an essentially $O(N2^N)$ time algorithm for the general problem (without any simplifying assumptions). For fixed C, we also give what is an essentially $O(P^{C+1})$ time algorithm. These algorithms, while not practical, can help us solve *small* problem instances. More importantly, they can help us understand the complexity of the problem more fully. For example, we now know that the problem is solvable in polynomial time if C is fixed (without further assumptions).

We have introduced the problem that is of the most immediate interest to us, discussed the work done on it thus far and outlined what we purport to do in the remainder of this note. But before we move on, it behooves us to briefly touch upon the recent works done on COS related problems. Many have considered single machine variations of COS with different objectives. These include the works of Bagchi et al. [2], Liao [20], Liao and Chuang [21], Gupta et al. [13] and Gerodimos et al. [11]. Many others have considered multi-machine extensions to COS under various objectives. These include the works of Ahmadi and Bagchi [1], Roemer and Ahmadi [23], Blocher and Chhajed [4], Cheng and Wang [6], Ng et al. [22], Yang and Posner [26], Yang [25] and Leung et al. [19]. Most of these ignore setups and often assume that machines are dedicated to product families. In yet another line of research, many have considered order-oriented scheduling in a job-shop environment with a view to evaluating the effectiveness of various dispatching rules. These include the works of Hastings and Yeh [14], Kim [18] and Blocher et al. [5]. Finally, in an interesting twist, the COS perspective (both in the single and multi-machine modes) has been brought to bear on the seemingly unrelated problems of scheduling load/unload cranes at shipping terminals (Daganzo [8]) and laying out data items in wireless broadcasts (Gondhalekar et al. [12]).

2. Problem complexity

As indicated earlier, we prove the strong NP-Hardness of COS or, more precisely, that of $1|s_p, \cos|\sum_c L_c$ by proving the strong NP-Hardness of the LTF-Order version of COS when the setup times are so large that it is suboptimal to have more than one setup per product family. For this purpose, we need two known strongly NP-Complete decision problems, both of which are involved with the optimal linear arrangement (OLA) of graph vertices.

Let $G = (V, E)$ be a simple graph with the vertex set V and the edge set E. In addition, let an edge $e \in E$ connect vertices $u, v \in V$. The first decision problem, call it OLA, asks: Given $G = (V, E)$ and a positive integer K, is there a one-to-one function $f: V \to \{1, 2, \ldots, |V|\}$ such that $\sum_{e \in E} |f(u) - f(v)| \leqslant |K$? OLA is a strongly NP-Complete problem of long standing (Garey and Johnson [10]).

Next, let $G' = (V', E')$ be a pseudograph with the vertex set V' and the edge set E'. As before, let an edge e' connect vertices u', $v' \in V'$. This time around, however, the graph is allowed to have multiple edges between two vertices and self-loops at a vertex. The second decision problem, call it OLA′, asks: Given $G' = (V', E')$ and a positive integer K', is there a one-to-one function f: $V' \to \{1, 2, \ldots, |V'|\}$ such that $\sum_{e' \in E'} \max\{f(u'), f(v')\} \leqslant K'$? Gondhalekar et al. [12] have established, relatively recently, that OLA′ is strongly NP-complete, through a reduction from OLA.

While it is not strictly needed here, we believe that, an outline of a simplified version of the above reduction is instructive. Let $d(u)$ be the degree of a vertex $u \in V$ of G; $d(u)$ is thus the number of edges connected to u. Similarly, let $\Delta(G)$ be the maximum degree of the graph G; i.e., $\Delta(G) = \max_{u \in V}\{d(u)\}$. Now, from $G = (V, E)$, construct $G' = (V', E')$ as follows: Let $V' = V$; for each $e \in E$, add two copies of e to E'; for each $u \in V$, add $\Delta(G) - d(u)$ self-loops at u' in V' (recall that $u' = u$). This construction is polynomial-time. For a given f, it can be shown that: $\sum_{e' \in E'} \max\{f(u'), f(v')\} = \sum_{e \in E} |f(u) - f(v)| + 1/2|V|(|V| + 1)\Delta(G)$. (This follows from the relationship between the distance between two numbers and their average as well as the definition of the vertex

degree.) The second term in the right hand side of the equation being a constant (i.e., independent of f), if we let $K' = K + 1/2|V|(|V| + 1)\Delta(G)$, we get a polynomial-time reduction of OLA to OLA′.

We now show how to reduce OLA′ to COS in polynomial time. From an instance of OLA′, construct an instance of COS as follows. (Notice that, because of the reduction shown above, we can choose an OLA′ instance such that the minimum degree of G′, $\min_{u' \in V'}\{d(u')\}$, is at least 1.) Let each vertex u' in V' correspond to a product family p and each edge e' in E' correspond to a customer c. We assume that c orders a single unit each from the product families (say, p and q) corresponding to u' and v', the vertices connected by the edge e'. If e' is a self-loop, c orders a single unit from the product family p corresponding to the associated vertex u'. Clearly, $P = |V'|$, $C = |E'|$, $t_{cp} = \{0, 1\}$ for all c and all p, $1 \leqslant N_c \leqslant 2$, and $N_p > 0$. Finally, we assume that $s_p = s$ for all p. (Note that s is chosen to be sufficiently large, say, equal to N^3.) The COS decision problem now asks: Is there a schedule σ for the above instance of COS such that $\sum_{1 \leqslant c \leqslant C} L_c(\sigma) \leqslant s(K' + 1)$?

It should be obvious that the COS decision problem is in NP. The construction of the COS instance is also accomplished in polynomial time.

Now, observe that the setup time s is selected such that there can be at most one setup per product family in a potentially optimal schedule. Let σ be such a schedule. σ yields a unique processing order of the product families. Let this correspond to the function f of OLA′. The position of a product family p within σ is thus given by $f(u')$, where the vertex u' of OLA′ corresponds to p. The position within σ of the last product family for customer c (associated with the edge e' connecting vertices u' and v' of OLA′) is thus given by $\max\{f(u'), f(v')\}$. The number of setups prior to the start of the last product unit of c is also $\max\{f(u'), f(v')\}$. Let the number of all product units processed on or before the completion of this unit be v_c ($v_c \leqslant N$). The order lead time for c is given by: $L_c(\sigma) = s \max\{f(u'), f(v')\} + v_c$. This leads to the total $\sum_{1 \leqslant c \leqslant C} L_c(\sigma) = s\sum_{e' \in E'} \max\{f(u'), f(v')\} + \sum_{1 \leqslant c \leqslant C} v_c$.

If there is an f that produces a "yes" answer to OLA′, we can construct σ by ordering the product families according to the linear ordering of the vertices induced by f; the product units within the family can be ordered arbitrarily. Having done that, we see from the above that $\sum_{1 \leqslant c \leqslant C} L_c(\sigma) \leqslant sK' + \sum_{1 \leqslant c \leqslant C} v_c = s[K' + (\sum_{1 \leqslant c \leqslant C} v_c)/s] < s(K' + 1)$. This implies that the COS decision problem has a "yes" answer as well.

Similarly, if there is a σ' that produces a "yes" answer to the COS decision problem, we first convert it to a schedule σ (with exactly one setup per family) by producing all family p units, for each p, immediately following the occurrence of its first setup in σ'. Because of the large value of s, it is easy to see that $\sum_{1 \leqslant c \leqslant C} L_c(\sigma) \leqslant \sum_{1 \leqslant c \leqslant C} L_c(\sigma')$. We can now easily obtain the f for OLA′ as described earlier. From the fact that $\sum_{1 \leqslant c \leqslant C} L_c(\sigma) \leqslant s(K' + 1)$, we get $\sum_{e' \in E'} \max\{f(u'), f(v')\} \leqslant K' + 1 - (\sum_{1 \leqslant c \leqslant C} v_c)/s \leqslant K'$ (this follows from the integrality of both $\sum_{e' \in E'} \max\{f(u'), f(v')\}$ and K'). The implication is that OLA′ too has a "yes" answer.

We have in effect proved that the COS decision problem is NP-Complete. The reduction has been from a strongly NP-Complete problem and the transformation has been polynomial-time. Thus, the COS decision problem is also strongly NP-Complete, and our COS problem $1|s_p, \cos|\sum_c L_c$ is strongly NP-Hard.

3. Solution algorithms

We now give two algorithms for solving COS exactly. The second algorithm exploits two solution properties first observed by Julien and Magazine [17]. We reiterate them below.

Property 1. *There is an optimal schedule where the product requirements for the customers are processed in the same sequence for all product families.*

Property 2. *There is an optimal schedule where the processing of a product family begins only when its inventory level is zero.*

3.1. The general algorithm

No algorithm has thus far been proposed for exactly solving COS in general; we now give a dynamic program for COS. Let Ω be the set of all the product lots ordered by the C customers (N is thus the cardinality of

Ω), $[k]$ be the index of the lot (indicating the customer it belongs to and its family) processed in the kth last position in a schedule, $\delta_{[k]}$ be a binary variable indicating if the kth last lot is the last one for the customer concerned, $t_{[k]}$ be the processing time requirement for the kth last lot, and $s_{[k]}$ be the setup time that may be required before the processing of the kth last lot. The total customer order lead time in some schedule σ can now be written as follows:

$$\sum_{1\leqslant c\leqslant C} L_c(\sigma) = \sum_{1\leqslant k\leqslant N} (s_{[k]} + t_{[k]}) \left(\sum_{1\leqslant i\leqslant k} \delta_{[i]} \right).$$

Based on the above formula, we can develop a Held–Karp [15] like dynamic programming recursion for solving COS. To this end, we define $f_k(\theta; x)$ to be the minimum total customer order lead time realizable when the product lots in the set θ (represented by their c–p pairs) occupy the last k positions in a schedule and a lot of family x is the first among them. We let $\Gamma(\theta)$ and $\Pi(\theta)$ be respectively the sets of customers and product families represented in θ. Notice that $|\Gamma(\theta)| = \sum_{1\leqslant i\leqslant k}\delta_{[i]}$ for all schedules in which the product lots in θ occupy the last k positions. If we also let o be a c–p pair in θ such that $p = x$, the dynamic programming recursion at stage k can be expressed as follows:

$$f_k(\theta; x) = \min_{o} \min_{y\in\Pi(\theta-o)} \{f_{k-1}(\theta - o; y) + |\Gamma(\theta)|t_o + |\Gamma(\theta - o)|r_{xy}\},$$

where r_{xy} equals s_y if $x \neq y$ and zero otherwise. The recursion is carried out from $k = 2$ to $k = N$ over all k-element subsets θ of Ω and over all x in $\Pi(\theta)$; the initial conditions are: $f_1(\{(c,p)\};p) = t_{cp}$ for all $(c,p) \in \Omega$. The optimal solution is given by $\min_{1\leqslant p\leqslant P}\{f_N(\Omega;p) + Cs_p\}$. The algorithm executes in $O(N2^N)$ time, which translates to $O(CP2^{CP})$ in case of the F-Order version of COS.

3.2. The fixed C algorithm

We now present a new dynamic programming based algorithm that can solve the general COS problem in polynomial time for a fixed C. Assume that we are given the customer order sequence and that we know which product family is processed last for each customer. It is now possible to visualize a schedule as being a string of intervals demarcated by the order deliveries for the individual customers (see Fig. 1a). Assume that the customers are indexed according to their positions in the given sequence. Let H_{cp} be the time taken for the processing (including setups, if any) of product family p in the cth interval. It is easily seen that, for any schedule, we have $\sum_{1\leqslant c\leqslant C} L_c = \sum_{1\leqslant p\leqslant P}\sum_{1\leqslant c\leqslant C}(C - c + 1)H_{cp}$. As long as we maintain conformance with the customer sequence and the last product families given, we can solve this subproblem as P separate dynamic programs. Once that is accomplished, COS can be solved by enumerating over all customer order sequences and their last product families. The complexity of solving the fixed-order-sequence/last-product-families subproblem is $O(C^2P)$, and the effort required for the enumeration is $O(C!P^C)$. The overall complexity of solving COS is thus $O(C^2C!P^{C+1})$; for a fixed C, this translates to $O(P^{C+1})$ which is polynomial-time.

The dynamic programs mentioned above exploit the schedule structure induced by the given customer sequence and the last product family per customer. They also use the aforementioned optimality properties whenever possible. The following observations summarize the key features of the algorithm:

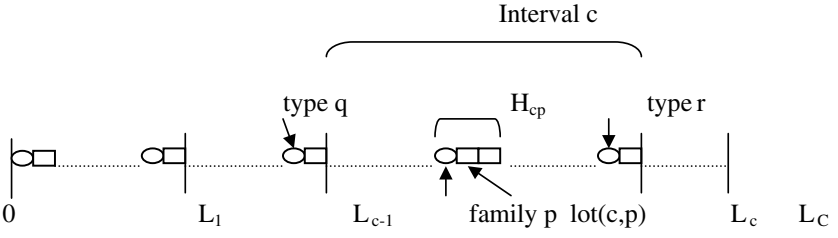

Fig. 1a. General schedule structure (also the case when $r \neq q \neq p$, $t_{cp} > 0$, and $H_{cp} > 0$).

1. A customer order sequence and last product family combination is deemed *feasible* if and only if the last product family p for each customer c is actually ordered by c, that is $t_{cp} > 0$. (This is obvious.)
2. All orders from customer c are processed in intervals $1, \ldots, c$. (This is dictated by the schedule structure.)
3. All products from family p are processed according to the given customer order sequence; that is, the order for p from c is processed only after (before) the orders for p from $1, \ldots, c-1$ ($c+1, \ldots, C$) have been processed. (This is done in keeping with Property 1.)
4. If two consecutive customers, say $c-1$ and c, have the same families for their last products, then these two are processed together with no other products in between (see Fig. 1b). (This is done as there is no advantage in having a setup before c's last product.)
5. If the condition of "4" is not true and customer c' last product family is p, only the order for p from c is processed in interval c initiating a setup; orders for p from $c+1, \ldots, C$ are processed in later intervals (see Fig. 1c). (This is dictated by the schedule structure.)
6. If the condition of "4" is not true and customer $(c-1)$'s last product family is p, then the order for p from c is processed in interval c sharing the setup with and immediately following customer $(c-1)$'s order for p (see Fig. 1d). (Saving a setup is clearly more advantageous.)
7. If the conditions of "4" through "6" are not true, customer c's order for p is processed either in interval c initiating a setup or in an earlier interval sharing a setup (see Figs. 1a and 1e). (This is in line with Property 2.)
8. If the conditions of "4"–"6" are not true and c has not ordered p, orders for p from $c+1, \ldots, C$ are processed either in intervals $1, \ldots, c-1$ or $c+1, \ldots, C$ (see Fig. 1f). (This is also in line with Property 2.)

Let $h_c(j;p)$ be the minimum total customer lead time realizable when j customers requiring product family p are processed on or before the delivery for the cth customer for a specific combination of customer order sequence and last product families. Note that this contributes only to $\sum_{1 \leqslant c \leqslant C} (C - c + 1)\, H_{cp}$ for product family p, which is a component of the overall objective function value, say H. We now describe the algorithm.

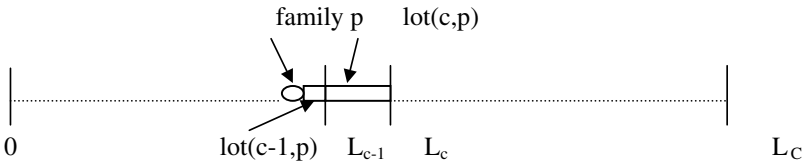

Fig. 1b. The case when $q = r = p$.

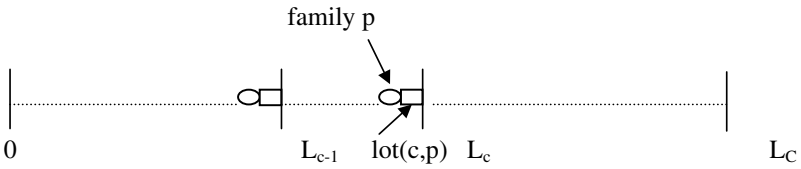

Fig. 1c. The case when $r = p$.

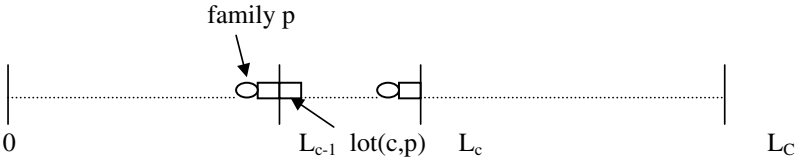

Fig. 1d. The case when $q = p$.

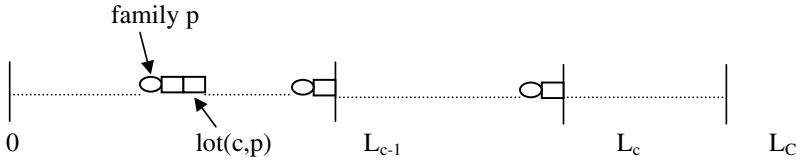

Fig. 1e. The case when $r \neq q \neq p$, $t_{cp} > 0$, and $H_{cp} = 0$.

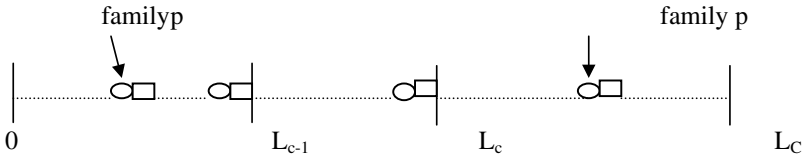

Fig. 1f. The case when $r \neq q \neq p$, and $t_{cp} = 0$.

(During the execution of the algorithm, the index c is updated so that it corresponds to the customer order sequence under consideration at the time of reference.)

1. Select a customer order sequence; for this sequence, select the last product family for each customer. Renumber the c's according to the customer order sequence.
2. Do the following if the combination generated in "1" is *feasible*:
   2.0. For $1 \leqslant p \leqslant P$:$h_0(0;p) = 0$, and $h_0(j;p) = \infty$, $1 \leqslant j \leqslant C$.
   2.1. Do over C customers ($c = 1, \ldots, C$):
      2.1.1. If last product family of c = last product family of $c - 1 = q$, then
         For $1 \leqslant p \leqslant P$ and $p \neq q$:$h_c(j;p) = h_{c-1}(j;p)$, $c \leqslant j \leqslant C$.
         For $p = q$:$h_c(c;p) = h_{c-1}(\mathrm{c} - 1;p) + (C - c + 1)t_{cp}$, and $h_c(j;p) = \infty$, $c + 1 \leqslant j \leqslant C$.
      2.1.2. Else Do over P product families ($p = 1, \ldots, P$):
         If p= last product family of c, then
         $h_c(c;p) = h_{c-1}(c - 1;p) + (C - c + 1)\ (s_p + t_{cp})$, and
         $h_c(j;p) = \infty$, $c + 1 \leqslant j \leqslant C$.
         Else If p = last product family of $c - 1$, then
         $h_c(j;p) = h_{c-1}(c - 1;p) + (C - c + 1)(\sum_{c \leqslant k \leqslant j} t_{kp})$, $c \leqslant j \leqslant C$.
         Else If $t_{cp} = 0$, then
         $h_c(c;p) = h_{c-1}(c - 1;p)$, and
         $h_c(j;p) = h_{c-1}(j;p)$, $c + 1 \leqslant j \leqslant C$.
         Else
         $h_c(j;p) = \min\{h_{c-1}(c - 1;p) + (C - c + 1)(s_p + \sum_{c \leqslant k \leqslant j} t_{kp}),\ h_{c-1}(j;p)\}$, $c \leqslant j \leqslant C$.
         End If
   2.2. Compute $H = \sum_{1 \leqslant p \leqslant P} h_C(C;p)$.
3. Compute the minimum of H over all customer order sequence and last product families combinations; this provides an optimal solution to COS.

Clearly, Step 1 generates all combinations of customer order sequence and last product families. In Step 2 (which is the heart of the algorithm), dynamic programming recursions are carried out for each feasible combination over all P product families until the last customer interval C in the generated sequence is encountered. At this point, the optimal schedule and the associated total customer order lead time for this particular combination are noted. Notice that the recursions in this step consider only those subschedules that conform to features 1 through 8 given above. Finally, Step 3 enumerates over all feasible combinations generated to arrive at the optimal solution to COS.

4. Conclusion

The customer order scheduling problem as posed here and otherwise is practically relevant and theoretically interesting. We have presented a number of new results for the basic COS problem that should be of theoretical value and of help in the evaluation of heuristic performance over small problem instances. We know now for sure that the problem is strongly NP-hard and further that it is polynomial-time solvable if the number of customers remains fixed. Based on our limited experience, we also know that if we are willing to spend adequately large CPU times, we can expect to solve problem instances with N up to 30 (for any admissible C and P) or up to 200 (for $C \leqslant 4$ and $P \leqslant 50$).

A number of issues remains open for future research. Many of these involve the development of practical solution strategies, both exact and approximate. In particular, attention should be given to finding good lower bounds and effective heuristic solution methods. The work of Julien [16] provides a nice lead in this direction, albeit with respect to the F-Order version of COS only. More work, of course, needs to be done with respect to the LTF-Order version as well.

References

[1] R.H. Ahmadi, U. Bagchi, Coordinated Scheduling of Customer Orders, Working Paper, John E. Anderson Graduate School of Management, University of California, Los Angeles, 1994.
[2] U. Bagchi, F.M. Julien, M.J. Magazine, Due-date assignment to multi-job customer orders, Management Science 40 (1994) 1389–1392.
[3] K.R. Baker, Scheduling the production of components at a common facility, IIE Transactions 20 (1988) 32–35.
[4] J.D. Blocher, D. Chhajed, The customer order lead-time problem on parallel machines, Naval Research Logistics 43 (1996) 629–654.
[5] J.D. Blocher, D. Chhajed, M. Leung, Customer order scheduling in a general job shop environment, Decision Sciences 29 (1998) 951–981.
[6] T.C.E. Cheng, G. Wang, Customer Order Scheduling on Multiple Facilities, Working Paper No. 11/98-9, Faculty of Business and Information Systems, The Hong Kong Polytechnic University, Hong Kong, 1999.
[7] E.G. Coffman, A. Nozari, M. Yannakakis, Optimal scheduling of products with two subassemblies on a single machine, Operations Research 37 (1989) 426–436.
[8] C.F. Daganzo, The crane scheduling problem, Transportation Research Part B 23 (1989) 159–175.
[9] E. Erel, J.B. Ghosh, The Customer Order Scheduling Problem Revisited, Discussion Paper No. 97-10, Faculty of Business Administration, Bilkent University, Ankara, Turkey, 1997.
[10] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, New York, 1979.
[11] A.E. Gerodimos, C.A. Glass, C.N. Potts, T. Tautenhahn, Scheduling multi-operation jobs on a single machine, Annals of Operations Research 92 (1999) 87–105.
[12] V. Gondhalekar, R. Jain, J. Werth, Scheduling on Airdisks: Efficient Access to Personalized Information Services via Periodic Wireless Data Broadcast, Technical report No. CS-TR-96-25, Department of Computer Sciences, University of Texas, Austin, 1996.
[13] J.N.D. Gupta, J.C. Ho, J.A.A. van der Veen, Single Machine Hierarchical Scheduling with Customer Orders and Multiple Job Classes, Working paper, The Netherlands Business School, Nijenrode University, The Netherlands, 1995.
[14] N.A.J. Hastings, C.-H. Yeh, Job oriented production scheduling, European Journal of Operational Research 47 (1990) 35–48.
[15] M. Held, R.M. Karp, A dynamic programming approach to sequencing problems, Journal of the Society for Industrial and Applied Mathematics 10 (1962) 196–210.
[16] F.M. Julien, Scheduling Multi-Product Customer Orders: Heuristics and Lower Bounds, Working Paper 96-09, University of Ottawa, 1996.
[17] F.M. Julien, M.J. Magazine, Scheduling customer orders: an alternative production scheduling approach, Journal of Manufacturing and Operations Management 3 (1990) 177–199.
[18] Y.-D. Kim, A comparison of dispatching rules for job shops with multiple identical jobs and alternate routings, International Journal of Production Research 28 (1990) 953–962.
[19] J.Y.-T. Leung, H. Li, M. Pinedo, Order scheduling in an environment with dedicated resources in parallel, Journal of Scheduling 8 (2005) 355–386.
[20] C.-J. Liao, Tradeoff between setup times and carrying costs for finished items, Computers & Operations Research 20 (1993) 697–705.
[21] C.-J. Liao, C.-H. Chuang, Sequencing with setup time and order tardiness trade-offs, Naval Research Logistics 43 (1996) 971–984.
[22] C.T. Ng, T.C.E. Cheng, J.J. Yuan, Concurrent open shop scheduling to minimize the weighted number of tardy jobs, Journal of Scheduling 6 (2003) 405–412.
[23] T.A. Roemer, R.H. Ahmadi, Complexity of Scheduling Customer Orders, Working Paper, John E. Anderson Graduate School of Management, University of California, Los Angeles, 1997.
[24] R.G. Vickson, C.A. Santos, M.J. Magazine, Batching and sequencing components at a single facility, IIE Transactions 25 (1993) 65–70.
[25] J. Yang, The complexity of customer order scheduling problems on parallel machines, Computers & Operations Research 32 (2005) 1921–1939.
[26] J. Yang, M.E. Posner, Scheduling parallel machines for the customer order problem, Journal of Scheduling 8 (2005) 49–74.