



Scheduling manufacturing systems in an agile environment

David He^{a,*}, Astghik Babayan^a, Andrew Kusiak^b

^a*Department of Mechanical Engineering, The University of Illinois at Chicago, Chicago, IL 60607, USA*

^b*Intelligent Systems Laboratory, Department of Industrial Engineering, 4132 SC, The University of Iowa, Iowa, IA 52242-1527, USA*

Abstract

Producing customized products to respond to changing markets in a short time and at a low cost is one of the goals in agile manufacturing. To achieve this goal customized products can be produced using an assembly-driven product differentiation strategy. The successful implementation of this strategy lies in efficient scheduling of the system. However, little research has been done in addressing the scheduling issues related to assembly-driven product differentiation strategies in agile manufacturing. In this paper, scheduling problems associated with the assembly-driven product differentiation strategy in a general flexible manufacturing system are defined, formulated, and solved. The manufacturing system consists of two stages: machining and assembly. At the machining stage, multiple identical machines produce parts. These parts are then assembled at the assembly stage to form customized products. The products to be produced in the system are characterized by their assembly sequences that are represented by different digraphs. The scheduling problem is to determine the sequence of products to be produced in the system so that the maximum completion time (makespan) is minimized for any given number of machines at the machining stage. The scheduling problems discussed in this paper have not been solved in the literature. The originality of the paper lies in defining and formulating the problems in the context of agile manufacturing and developing optimal and near-optimal for solving them. The heuristic algorithm solves the scheduling problem in two steps. First, an optimal aggregate schedule is determined by solving a two-machine flowshop problem. Next, the optimal aggregate schedule is decomposed by solving a simple integer programming formulation model. The computational experiment shows that the heuristics provide optimal and near-optimal solutions to the scheduling problems. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: Agile manufacturing scheduling; Assembly-driven product differentiation; Manufacturing systems

1. Introduction

Producing customized products in a short time at low cost is one of the goals of agile manufacturing. To achieve this goal in a manufacturing system, products are differentiated either by a machining-driven or an assembly-driven differentiation strategy. When product differentiation takes place at the machining stage, non-standard and complex parts are machined with relatively complex machines at the machining stage and assembled at the assembly stage with relatively simple assembly equipment. This machining-driven strategy relies on machining to express model mix and achieve flexibility. When product differentiation takes place at the assembly stage, simple and common parts are machined with

simple machines at the machining stage and assembled at the assembly stage with relatively novel assembly techniques. The successful implementation of these two strategies lies in efficient scheduling of the system. However, little research has been done in addressing the scheduling issues related to assembly driven product differentiation strategy in agile manufacturing, especially when products with complicated assembly sequences are produced. In this paper, the scheduling problems related to the assembly-driven product differentiation strategy for agile manufacturing are formulated and solved.

The structure of the manufacturing system that implements the assembly-driven product differentiation strategy in agile manufacturing is shown in Fig. 1. The manufacturing process in the system consists of two stages: machining and assembly. There are m identical machines at the machining stage and there is one assembly machine at the assembly stage. The machining operations include all activities that consist of

* Corresponding author. Tel.: + 1-312-996-3410; fax: + 1-312-413-0447.

E-mail address: davidhe@uic.edu (D. He).

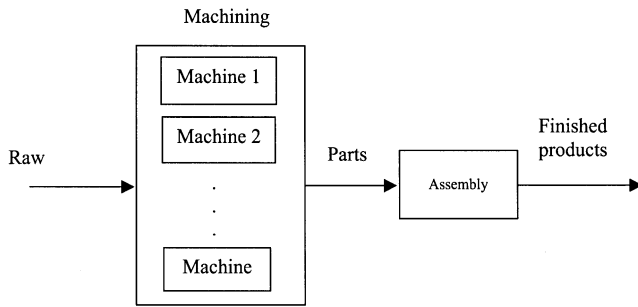


Fig. 1. The general structure of the manufacturing system.

material-removal operations. Assembly involves assembling the machined parts to form the required products.

The originality of the paper lies in formulating scheduling problems in the context of agile manufacturing and developing optimal and near-optimal methods to solve the problems. Heuristics for solving effectively the scheduling problem are developed. The heuristics solve the scheduling problem in two steps. First, an optimal aggregate schedule is determined by solving the scheduling problem as a two-machine flowshop makespan scheduling problem. The optimal aggregate schedule is then decomposed by solving a simple integer programming formulation model to obtain a feasible solution to the scheduling problem. A tight lower bound on the optimal makespan is also developed to evaluate the effectiveness of the heuristics. The computational experiment shows that the heuristic provides optimal and near-optimal solutions to the scheduling problems.

The remainder of the paper is organized as follows. In Section 2, scheduling problems to be solved in this paper are formally defined. Literature on solving related scheduling problems is reviewed in Section 3. Section 4 focuses on the methods development for solving the scheduling problems. The computational results for solving the scheduling problems with the developed solution approaches are provided in Section 5. Finally, Section 6 concludes the paper.

2. Problem definition

In this section, the scheduling problems related to implementation of the assembly-driven product differentiation strategy for agile manufacturing are defined.

2.1. The scheduling problem

There are m identical machines at the machining stage and there is single assembly equipment at the assembly stage. The objective of the scheduling is to assign parts and subassemblies/assemblies to the machines at the

machining stage and determine the processing sequences on the machines so that the makespan (C_{\max}), i.e., the maximum completion time, is minimized.

In this paper, the assembly sequence of a product refers to the order in which parts and subassemblies are assembled by the assembly equipment. The assembly sequence of a product to be produced in the system is represented by a digraph G . In a digraph G , each node represents a part or a subassembly/assembly, and an arc represents a precedence relationship between two nodes. Any node of degree 1, i.e., with the number of edges incident to the node equal to 1, denotes a part. Any node of degree greater than 1, i.e., with the number of edges incident to the node greater than 1, denotes a subassembly or an assembly. The root node of a digraph always represents an assembly.

The level of assembly in a digraph is assigned as follows: value of 1 is assigned to the root node (assembly) and working backward from the root node, values of increment 1 are assigned to each subassembly node [1]. Part nodes have the same assembly level as the corresponding subassembly or assembly nodes. Two products and the corresponding digraph representations of their assembly sequences are illustrated in Fig. 2. The digraphs representing the assembly sequences of products can be classified into two types: simple digraph and complex digraph. A simple digraph is a digraph in which at most one subassembly node can be found at every assembly level (see Fig. 2(a)). A simple digraph represents a linear assembly sequence of a product design. In a complex digraph, more than one subassembly node can be found on at least one assembly level (see Fig. 2(b)). Throughout the paper, an assembly sequence of a product that can be represented by a simple digraph is referred to as a simple assembly sequence and an assembly sequence of a product that can be represented by a complex digraph is referred to as a complex assembly sequence.

The scheduling problem will be solved according to the production mode of the system. Based on the representation of assembly sequence of the products, three production modes are defined: (a) production of a single product with a simple assembly sequence; (b) production of a single product with a complex assembly sequence; (c) production of N products. According to the three defined production modes, the associated scheduling problems are defined as G_s scheduling problems, G_c scheduling problems, and N -product scheduling problems, respectively.

3. Literature review

The problem of scheduling products represented by simple and complex digraphs in a two-stage flexible manufacturing system was first solved as an aggregate scheduling problem in [1]. In his paper, the aggregate

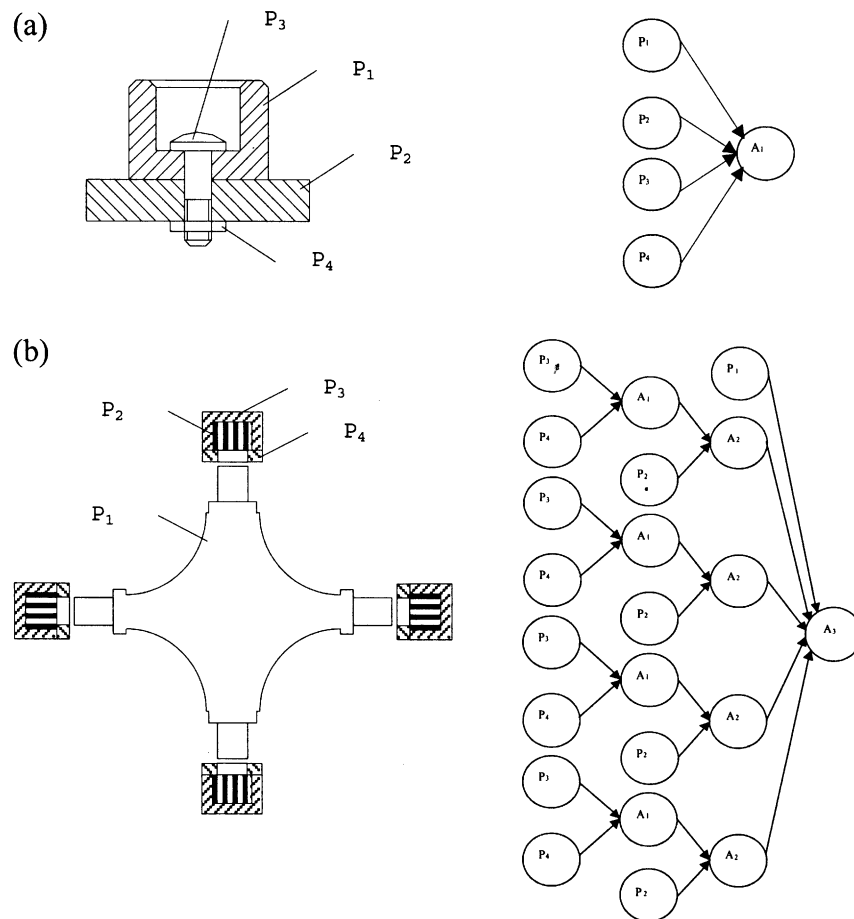


Fig. 2. Example of (a) a product and its simple digraph and (b) a product and its complex digraph.

scheduling problem is modeled as the two-machine flow shop scheduling problem. Optimal scheduling algorithms for solving both single- and N -product scheduling problems were developed. However, the aggregate scheduling problems solved in [1] assume that the only one processing unit is available at both machining and assembly stages and this assumption does not reflect the real situation in an agile manufacturing environment.

Another type of a scheduling problem solved in the literature is the flow shop problem with parallel machines (FSPM). FSPM is considered as an extension of two classical scheduling problems: flow shop scheduling problem and parallel machine scheduling problem. FSPM is also a basic model for flexible flow line scheduling problems solved in the literature. Hunsucker and Shah [2] reviewed industrial applications of scheduling in chemical engineering, computer systems, telecommunication networks, flexible manufacturing systems (FMSs), etc. Brah and Hunsucker [3] developed a branch and bound algorithm to solve the makespan FMPM scheduling problem. Mathematical models of FSPM have been discussed in [4]. Some special cases of FSPM have also been studied in the literature. Gupta [5]

developed a heuristic algorithm for a two-stage problem one machine at the second stage. Sriskandarajah and Sethi [6] developed heuristics for a two-stage case and established worst-case bounds. Two heuristic algorithms generating high-quality solutions for a two-stage case with one machine at stage one and several machines at stage two have been developed by Gupta and Tunc [7]. Chen [8] developed heuristics to solve the special cases for systems that have only two centers or systems where only one of the centers has parallel machines. Extensions of FSPM that incorporate buffers and transport between centers have been studied by Wittrock [9] and Sawik [10].

Flow shop scheduling problems and parallel machine scheduling problems represent another class of related problems. The two-machine flow shop scheduling problem can be solved by Johnson's [11] algorithm. However, in general, if the number of machines in a flow shop is more than two, then the scheduling problem is known to be NP-complete (see, for example, [12]). While the majority of research into flow shop scheduling is focused on the serial-type flow shop, a 3-machine assembly-type flow shop scheduling problem was studied by Lee et al. [13].

Parallel machine scheduling problem ($P\|C_{\max}$) has been proved by Garey and Johnson [14] as NP-hard in the strong sense when the number of machines is unlimited. However, the problem is solvable in pseudo-polynomial time when the number of machines is fixed and thus NP-hard only in the ordinary sense. A recent survey on parallel scheduling problems has been published by Cheng and Sin [15]. Blazewicz et al. [18] has proposed optimal algorithms. But most algorithms developed for solving $P\|C_{\max}$ are heuristics (e.g., [16], Goffman et al. [45], [17]).

Except for the scheduling problems solved by Kusiak [1], none of the scheduling problems solved in the literature considered product structures represented by the simple and complex digraphs even though they provide the best structural information of the products.

3.1. Solution approaches developed for solving the scheduling problems

Solution methods developed for solving the discussed NP-hard scheduling problems can be classified into four categories: (1) operations research (OR) approaches; (2) stochastic search methods such as genetic algorithm, tabu search, and simulated annealing; (3) neural network; (4) a combination of (1), (2), or (3).

Mathematical programming plays a key role in model building and testing scheduling formulations. Blazewicz et al. [18] published the first survey paper on mathematical formulations for single machine, parallel machine, and job shop scheduling problems along with the classification of problems' complexity. Even though the "solvability" of a scheduling problem cannot be easily assessed from the formulation itself, it might be used as a stepping stone to optimal solution of some hard problems. The conventional operations research methods for obtaining optimal schedules are branch-and-bound-based methods [3,13,19]. However, the branch and bound methods solve the scheduling problems in exponential time in the worst case. In order to improve the efficiency of the methods, dominance rules based on some properties derived from special cases can be used to eliminate dominated branches in the branching process (see [13]). The effectiveness of the branch and bound methods depends also on strength of the lower bound. Santos et al. [20] introduced a new procedure for developing strong global lower bounds for minimum makespan FSPM. The procedure calculates a lower bound at each stage and global lower bound is the strongest among all the stage-based lower bounds. A lower bound for a stage is the average of the lower bounds calculated for each processor at that stage. Recent developments in OR methods to overcome the complexity issues in solving scheduling problems include applying Lagrangian relaxation (LR) techniques (e.g., [21–24]). LR techniques

decompose a problem into a number of smaller subproblems that are easier to solve. Solving parallel identical machine scheduling problems with single-operation jobs is discussed in [22] and multi-operation jobs with simple fork/join-type precedence constraints in [23], LR was applied to relax the capacity constraints and decompose scheduling problems. As precedence constraints, in general, complicate the scheduling problems [25], both capacity and precedence constraints were relaxed in solving a job shop scheduling problem in [24]. Their computational experiment showed that high-quality schedules were generated efficiently with objective values within 4% of their respective lower bounds.

Recently, stochastic search methods such as genetic algorithms [26], simulated annealing [27,28], and tabu search [29,30] have been applied to solve hard scheduling problems. Van Laarhoven et al. [27] developed a simulated annealing based algorithm for job shop scheduling problems. In their approach, searching for the minimum makespan schedule is modeled as a disjunctive graph constructed based on the precedence relationship between jobs' operations. The objective of finding a schedule with the minimum makespan is equivalent to finding the minimum longest path in the digraph. They proved that the algorithm asymptotically converges to the globally minimum solution. However, in a recent paper, Kolonko [28] showed that analytical results on convergence for simulated annealing do not hold in the application to the job shop scheduling problems. He combined simulated annealing method with genetic algorithm to allow the solutions of simulated annealing to crossover in each generation using a new time-oriented crossover operation, preventing the destruction of convergence by the asymmetric neighborhood of the scheduling problem. Portmann et al. [26] combined a branch and bound method with genetic algorithm approach for solving multistage flow shop with parallel machines scheduling problems. Genetic algorithm is used to improve the upper bound during the search process. Their experimentation showed that the combination of branch and bound and genetic algorithm generated optimal solutions faster than the branch and bound method. Nowicki and Smutnicki [29] developed a tabu-search-based algorithm for solving FSPM scheduling problem. The neighborhood defined in their approach was constructed based on the concept of critical path in a digraph consisting of a block of connecting jobs. Their implementation improved the local search significantly and increased the speed of the algorithm.

Artificial neural networks (ANNs) are alternatives to conventional OR techniques for solving the scheduling problems. The types of neural networks found in scheduling applications include: (1) Hopfield networks [31–35]; (2) competitive networks [36–38]; (3) back propagation networks [39–42]. It is observed that majority of

scheduling applications of ANNs are based on Hopfield networks. Although the scheduling applications of simulated ANNs have showed promising results, excessive computational times for large size problems is a major disadvantage. One promising area of ANNs in scheduling applications is the combination of ANNs with stochastic search method such as simulated annealing since simulated annealing can help to escape from local optimum during the optimization process. The combination of Hopfield networks with Lagrangian relaxation (LR) has also showed promising improvement of scheduling solutions [43]. Luh et al. [43] showed the convergence of Lagrangian relaxation neural network for separable convex programming. Luh et al. [44] proposed a mixed integer Lagrangian relaxation neural network (MILRNN) for total weighted earliness and tardiness job shop scheduling problem. The problem is decomposed by Lagrangian relaxation into polynomial subproblems and these subproblems can be solved by dynamic programming. Then a neural dynamic programming is

4. Methods development

In this section, the methods developed for effective solving scheduling problems in agile manufacturing are presented. The method development consists of two parts. The first part involves developing effective solution methods for solving the scheduling problems. The second part is to develop methods for evaluating the effectiveness of the solution methods.

4.1. Development of solution methods

It can be noted that the G_s scheduling problem is a special case of the G_c scheduling problem and N -product scheduling problem. Therefore, our strategy is to develop a solution method for solving the G_s scheduling problem first and then extend the solution method to the G_c and N -product scheduling problems.

Before the solution methods are presented, the following notation is introduced:

i = index of parts

j = index of machines

l = assembly level index

L = the maximum assembly level

n_l = total number of parts at assembly level $l, l = 1, \dots, L$

$n_{\max} = \max_{1 \leq l \leq L} \{n_l\}$ = maximum number of parts over all assembly levels

P_i^l = i th part at assembly level $l, i = 1, \dots, n_l, l = 1, \dots, L$

$t(P_i^l)$ = machining time of the i th part at assembly level $l, i = 1, \dots, n_l, l = 1, \dots, L$

$T^l = \max_{1 \leq i \leq n_l} \{t(P_i^l)\}$ = longest machining time of parts at assembly level $l, l = 1, \dots, L$

$t(A_l)$ = assembly time of assembly/subassembly $A_l, l = 1, \dots, L$

t = maximum completion time when all parts and subassemblies before the final assembly node are scheduled

$$x_{ij}^l = \begin{cases} 1 & \text{if part } i \text{ at assembly level } l \text{ is assigned to machine } j \\ 0 & \text{otherwise.} \end{cases}$$

developed by mapping the LR approach onto a MILRNN. Their simulation experiment showed that MILRNN provides near-optimal solution for practical scheduling problems.

In summary, the scheduling problems addressed in this paper have not been solved in the literature. Since the system structure of the scheduling problems to be solved in this proposal is similar to FSPM, flow shop, and parallel machine shop, existing solution methods in the literature for solving FSPM scheduling problems, parallel machine scheduling problems, and flow shop scheduling problems provide a rich source for reference. Recent development in heuristic approaches for solving scheduling problems provides a direction for development of efficient optimal and effective heuristic algorithms to solve scheduling problems for agile manufacturing.

4.1.1. Solving G_s scheduling problem

When a single product with simple assembly sequence is produced in the system, the scheduling problem is an G_s scheduling problem. The G_s scheduling problem can be formulated as an integer programming model as following:

$$\text{Minimize } t \quad (1)$$

$$\text{Subject to } \sum_{l=1}^L \sum_{i=1}^{n_l} t(P_i^l) x_{ij}^l \leq t, \quad j = 1, \dots, m, \quad (2)$$

$$\sum_{j=1}^m x_{ij}^l = 1, \quad i = 1, \dots, n_l, \quad l = 1, \dots, L, \quad (3)$$

$$\sum_{k=1}^L \sum_{i=1}^{n_i} t(P_i^k) x_{ij}^k \leq t - \sum_{k=2}^l t(A_k), \quad (4)$$

$$j = 1, \dots, m, \quad l = 2, \dots, L.$$

$$x_{ij}^l = 0 \text{ or } 1, \text{ for } l = 1, \dots, L,$$

$$i = 1, \dots, n_l, \quad j = 1, \dots, m, \quad (5)$$

Note that the objective function (1) is to minimize the maximum completion time in the system. Constraint (2) ensures that the total machining time of the parts assigned to each machine will not be greater than the maximum completion time. Constraint (3) ensures that a part can be assigned to one machine only. Constraint (4) imposes that a subassembly or a final assembly cannot start until all the components are available. Constraint (5) ensures that the decision variable x_{ij} takes value of either 0 or 1. After solving models (1)–(5), the minimum makespan of the production schedule is computed as $t + t(A_1)$.

Note that G_s scheduling problem is a constrained version of the identical parallel machine scheduling problem with minimum makespan objective, which in the worst case can be solved in a pseudo-polynomial time [14]. Therefore, in the worst-case formulations (1)–(5) is solvable in pseudo-polynomial time.

4.1.2. Solving G_c scheduling problem

When a single product with a complex assembly sequence is produced in the system, the scheduling problem is a G_c scheduling problem. The G_c scheduling problem is considered as a more general version of three-machine assembly-type flowshop (3MAF) scheduling problem solved by Lee et al. [13]. If set $m = 2$ and let the complex digraph G_c be formed by connecting n number of simple digraphs that have only two-part nodes to a dummy final assembly node with assembly time of 0, then the G_c scheduling problem is the same as 3MAF scheduling problem. Since the 3MAF scheduling problem has been proved to be strongly NP-complete [13], G_c scheduling problem is at least strongly NP-complete. Due to the computational complexity of the scheduling problem, effective near-optimal algorithms for solving large problems along with the optimal algorithms are developed in this paper. A two-step heuristic for solving the G_c scheduling problem is described next. In the first step of the heuristic, an optimal aggregate schedule is obtained by applying Theorem 2 in [1]. In general, an optimal aggregate schedule obtained by Theorem 2 can always be represented by the following sequence:

$$S(G_c) = \{g_1, g_2, \dots, g_k, A_1\}, \quad (6)$$

where A_1 is the root node (final assembly) of digraph G_c , and g_k , represents either a part node or a simple digraph that consists of a number of part nodes and a subassembly node.

From this optimal aggregate sequence, a simple digraph with a root node of A_1 is constructed. In this simple digraph, g_1 represents part and subassembly nodes at the highest assembly level and g_k represents the part and subassembly nodes at the lowest assembly level. After constructing an equivalent simple digraph corresponding to the optimal aggregate sequence in (6), formulations (1)–(5) is solved to obtain an optimal schedule for

the simple digraph. The heuristic algorithm for solving the G_c scheduling problem is presented next.

Heuristic algorithm 1 (HA1)

Step 1: Obtain an optimal aggregate schedule $S(G_c)$ for complex digraph G_c using Theorem 2 from Kusiak [1].

Step 2: Construct a simple digraph G_s from $S(G_c)$.

Step 3. Solve models (1)–(5) for G_s obtained in Step 2.

Note that G_s and G_c scheduling problems do not reflect actual scheduling scenario where normally scheduling decision for N multiple products has to be made. However, solving G_s and G_c scheduling problems provides useful insights to the development of methods for solving N -product scheduling problems.

4.1.3. Solving N -product scheduling problem

The N -product problem involves scheduling N multiple products. In solving the N -product scheduling problem, the assembly sequence of a product could be either a simple digraph or a complex digraph. The approach for solving the N -product scheduling problem is to construct a complex digraph by connecting the assembly nodes of N products to a dummy final assembly node, A_d . An assembly time of 0 is assigned to A_d , i.e., $t(A_d) = 0$. Then solving the N -product scheduling problem is equivalent to solving a G_c scheduling problem. Fig. 3 shows the transformation of N digraphs into a single complex digraph.

The heuristic algorithm for solving the N -product scheduling problem is presented next.

Heuristic algorithm 2 (HA2)

Step 1: Construct a complex digraph by connecting the assembly nodes of N products to a dummy final assembly node, A_d . Let $t(A_d) = 0$.

Step 2: Apply HA1 to solve the G_c scheduling problem for the complex digraph constructed in Step 1.

Next, an example is used to illustrate the application of the heuristic approaches for solving the scheduling problems.

Illustrative Example. Consider two products C_1 and C_2 to be produced in a manufacturing system. There are two identical machines ($m = 2$) at the machining stage and one assembly equipment at the assembly stage. The assembly sequences of the two products are shown in Fig. 4. The machining and assembly times for the two products are provided in Table 1. The objective of the scheduling problem is to assign parts to the machines at the machining stage and the processing sequence on each

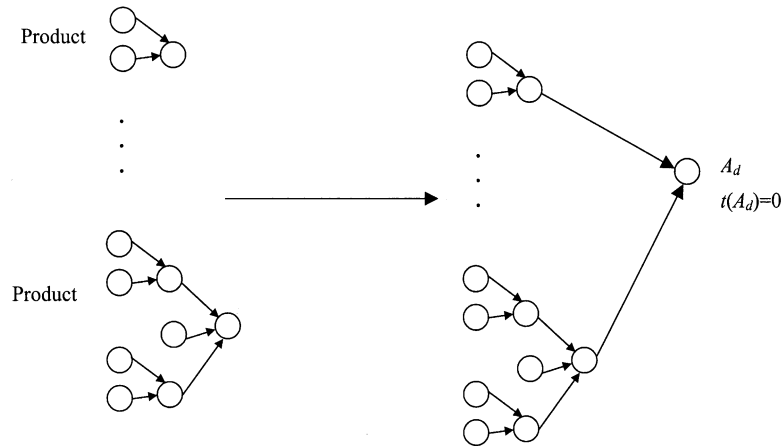


Fig. 3. Transforming an N -product scheduling problem to a G_c scheduling problem.

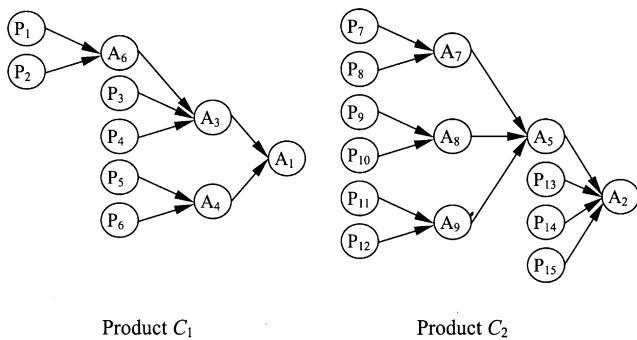


Fig. 4. Digraph representation of assembly sequences of products C_1 and C_2 .

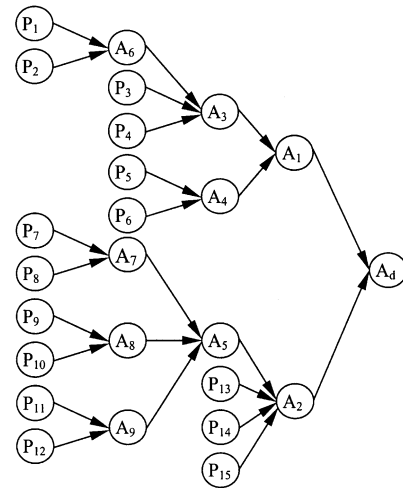


Fig. 5. The combined digraph.

machine. Since the scheduling problem is a two-product scheduling problem, heuristic algorithm HA2 is applied.

Application of Heuristic algorithm 2 (HA2)

HA2-Step 1: Connect assembly nodes A_1 and A_2 to a dummy node A_d , and let $t(A_d) = 0$. The resulting digraph is shown in Fig. 5.

HA2-Step 2: Apply heuristic algorithm HA1 for the complex digraph in Fig. 5 to solve the scheduling problem.

Application of Heuristic algorithm 1 (HA1)

HA1-Step 1: Applying Theorem 2 of Kusiak [1] to the complex digraph obtained in HA2-Step 1, the optimal aggregate schedule $S(G_c)$ is obtained as following: $S(G_c) = \{[(P11, P12, A9), (P7, P8, A7), (P9, P10, A8), A5], P13, P14, P15, A2, (P1, P2, A6), P3, P4, A3, (P5, P6, A4), A1, A_d\}$.

Table 1
Machining and assembly times

Part	P ₁	P ₂	P ₃	P ₄	P ₅	P ₆	P ₇	P ₈	P ₉	P ₁₀	P ₁₁	P ₁₂	P ₁₃	P ₁₄	P ₁₅
Machining time	5	7	10	8	6	9	10	5	9	8	6	7	5	6	8
Subassembly	A ₁	A ₂	A ₃	A ₄	A ₅	A ₆	A ₇	A ₈	A ₉	—	—	—	—	—	—
Assembly time	12	12	15	13	15	11	14	11	15	—	—	—	—	—	—

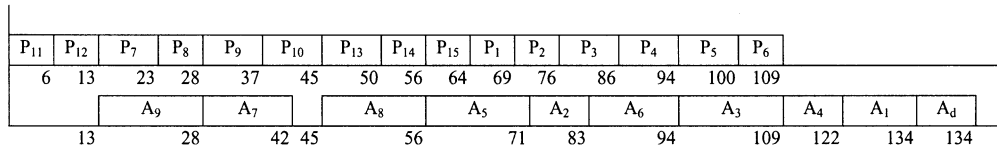


Fig. 6. The Gantt chart of the aggregate schedule.

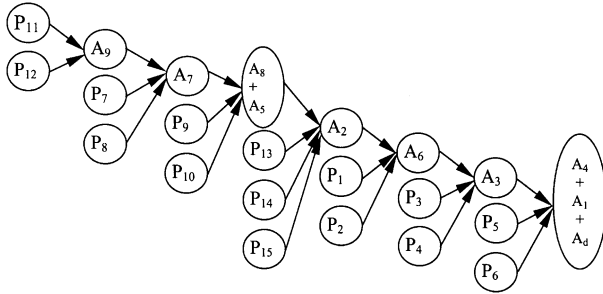


Fig. 7. The simple digraph generated from the Gantt chart in Fig. 6.

The Gantt chart of the resulting aggregate schedule is shown in Fig. 6.

HA1-Step 2: Based on the order of parts and subassemblies specified in the aggregate sequence $S(G_c)$ obtained in HA1-Step 1, a simple digraph is obtained as shown in Fig. 7.

HA1-Step 3: Solve models (1)–(5) for the simple digraph in Fig. 7, the schedule is obtained in Fig. 8.

4.2. Development of a lower bound (LB)

A standard approach for evaluating effectiveness of a heuristic is to compare the value of the heuristic solution with a lower bound on the optimal solution. For this reason, we establish a tight lower bound on the makespan. In order to develop a tight lower bound, two lower bounds are constructed and the tight bound is the largest one among the two lower bounds. The first lower bound LB_1 is developed based on the observation that the assembly work of any end subassembly node cannot begin until processing of the required parts is completed

at the machining stage. An end subassembly node is the one whose preceding nodes are all part nodes, for example, nodes $A_4, A_6, A_7, A_8,$ and A_9 in Fig. 4. Let NA be the set of end subassembly nodes and $T_i^{\max}, i \in NA$ be the maximum processing time of parts of end subassembly node A_i at machining stage. For any $A_i, i \in NA, T_i^{\max}$ can be computed as following:

$$T_i^{\max} = \begin{cases} \max_{1 \leq s \leq n_i} \{t(P_s^i)\}, & \text{if } n_i \leq m, \\ \max \left\{ \max_{1 \leq s \leq n_i} \{t(P_s^i)\}, \left\lceil \frac{\sum_{s=1}^{n_i} t(P_s^i)}{m} \right\rceil \right\}, & \text{if } n_i > m. \end{cases} \quad (7)$$

Therefore, the first lower bound LB_1 is computed as following:

$$LB_1 = \text{Min}_{i \in NA} \{T_i^{\max}\} + \sum_{i=0}^L t(A_i). \quad (8)$$

In order to develop the second lower bound LB_2 , for any digraph G , a term called “a possible path to the final assembly node”, p , is defined as a path from any subassembly node that has no succeeding part nodes to the final assembly node in a digraph G . Then define set SUA_p as the set of subassembly nodes on path p . For a given value of m and a digraph G , the lower bound on the makespan of the production schedule is computed as follows:

$$LB_2 = \left\lceil \frac{\sum_{l=1}^L \sum_{i=1}^{n_l} t(P_i^l)}{m} \right\rceil + \text{Min}_{\text{for all } p} \left\{ \sum_{i \in SUA_p} t(A_i) \right\}. \quad (9)$$

Then the tight lower bound LB is computed as follows:

$$LB = \text{Max}\{LB_1, LB_2\}. \quad (10)$$

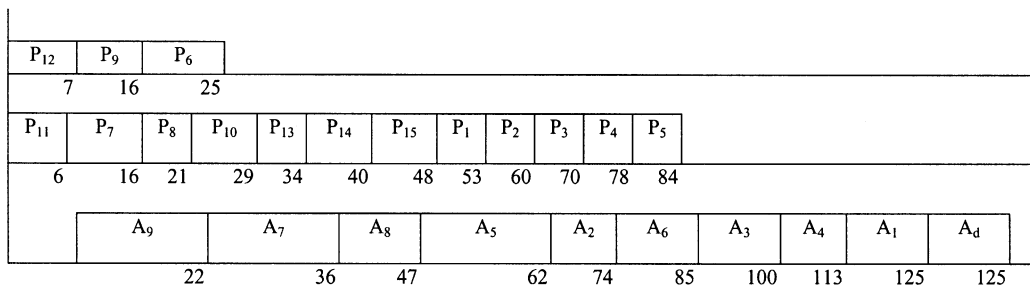


Fig. 8. The Gantt chart of the obtained schedule.

For example, the first lower bound LB_1 for the digraph in Fig. 5 with $m = 2$ is computed as

$$LB_1 = 7 + 118 = 125.$$

The second lower bound LB_2 for the digraph in Fig. 5 with $m = 2$ is computed as follows. The set of possible paths to the final assembly node A_d contains: $(A_3 \rightarrow A_1 \rightarrow A_d)$, $(A_4 \rightarrow A_1 \rightarrow A_d)$, $(A_2 \rightarrow A_d)$. Therefore, based on Eq. (9), when $m = 2$, the lower bound is computed as

$$LB_2 = 54.5 + 12 = 67.5.$$

Then the tight lower bound is computed as

$$LB = \text{Max}\{LB_1, LB_2\} = \text{Max}\{125, 67.5\} = 125.$$

As a coincidence, the lower bound is the same as the makespan of the heuristic schedule shown in Fig. 8. This indicates that the solution in Fig. 8 is optimal.

5. Computational experience

In order to evaluate performance of the heuristic approaches developed in this paper, a computational experiment was conducted. In the computational experiment, makespans of the schedules obtained by the heuristic approaches were compared with the lower bounds. We have tested 16 types of problems with different assembly sequences. For each testing problem, 10 instances were generated and for each instance, assembly sequence, number of parts, number of subassembly nodes, level of assembly, machining times, assembly times were randomly generated with random number generator

in the PC computer. For each problem type 10 instances were generated. The results of the comparison are provided in Table 2.

As one can see from Table 2, the relative differences between the makespans of the heuristic solutions and the lower bounds are small for all the problems tested in the computational experiment. The maximum percentage relative difference is 2.5. For many problems tested in the experiment, the makespans were the same as the lower bounds, which indicates that the solutions are optimal. The results of the computational experiment indicate that the heuristic solution approaches developed provide optimal and near optimal solutions.

6. Conclusions

Producing customized products to respond to changing market in a short time at low cost is one of the goals in agile manufacturing. To achieve this goal in a manufacturing customized products can be produced by an assembly-driven product customization strategy. The successful implementation of this strategy lies in efficient scheduling of the system. However, little research has been done in addressing the scheduling issues related to assembly-driven product differentiation strategy for agile manufacturing.

In this paper, solving scheduling problems associated with the assembly-driven product differentiation strategy in a general flexible manufacturing system was discussed. The flexible manufacturing system consists of two stages: machining and assembly. At the machining stage, simple and common parts are produced on multiple identical machines. These parts are then assembled at the assembly stage to form customized products. The products to be produced in the system are characterized by their assembly sequences that are represented by different digraphs. The scheduling problem is to determine the sequence of products to be produced in the system so that the maximum completion time (makespan) is minimized for any given number of machines at the machining stage.

Depending on the production mode and the characteristics of the assembly sequences of the products, the scheduling problems associated with the assembly-driven product differentiation strategy in agile manufacturing were classified into G_s , G_c , and N -product scheduling problems. An integer programming formulation model for solving the G_s scheduling problem and effective heuristics for solving the G_c and N -product scheduling problem were developed. The heuristics solve the scheduling problems in two steps. An optimal aggregate schedule is determined by solving the scheduling problem as a two-machine flowshop scheduling problem. Based on the optimal aggregate schedule, the scheduling problems are converted into a G_s scheduling problem by constructing a simple digraph of assembly sequence based on the

Table 2
Comparison with the lower bounds

Problem no.	C_{\max}^H (Heuristic solution)	LB (Lower bound on C_{\max})	$\left(\frac{C_{\max}^H - LB}{C_{\max}^H}\right)100\%$
1	73	73	0.00
2	80	78	2.50
3	136	136	0.00
4	136	135	0.74
5	116	114	1.72
6	114	112	1.75
7	33	33	0.00
8	34	34	0.00
9	55	54	1.82
10	58	57	1.72
11	75	75	0.00
12	75	75	0.00
13	104	104	0.00
14	168	168	0.00
15	156	156	0.00
16	116	114	1.72

optimal aggregate schedule. A tight lower bound on optimal makespan was developed to evaluate the effectiveness of the heuristics. The computational experiment shows that the heuristics provide optimal and near-optimal solutions.

The scheduling problems discussed in this paper have not been solved in the literature. The originality of the paper lies in formulating the problems in the context of agile manufacturing and developing optimal and near-optimal solution methods.

References

- [1] Kusiak A. Aggregate scheduling of a flexible machining and assembly system. *IEEE Trans Robotics Automat* 1989;5(4):451–9.
- [2] Hunsucker JL, Shah JR. Comparative performance analysis of priority rules in a constrained flow shop with multiple processors environment. *Eur J Oper Res* 1994;72:102–14.
- [3] Brah S, Hunsucker JL. Branch and bound algorithm for the flow shop with multiple processors. *Eur J Oper Res* 1991;51:88–99.
- [4] Brah SA, Hunsucker JL, Shah JR. Mathematical modeling of scheduling problems. *J Inform Optim Sci* 1991;12:113–37.
- [5] Gupta JND. Two-stage hybrid flowshop scheduling problem. *J Oper Res Soc* 1988;34(4):359–64.
- [6] Sriskandarajah C, Sethi SP. Scheduling algorithms for flexible flowshop: worst and average case performance. *Eur J Oper Res* 1989;43(2):143–60.
- [7] Gupta JND, Tunc EA. Schedules for a two-stage hybrid flowshop with parallel machines at the second stage. *Int J Prod Res* 1991;29(7):1489–502.
- [8] Chen B. Analysis of classes of heuristics for scheduling a two-stage flow shop with parallel machines at one stage. *J Oper Res Soc* 1995;46:234–44.
- [9] Wittrock RJ. An adaptable scheduling algorithm for flexible flow lines. *Oper Res* 1988;36:445–53.
- [10] Sawik T. A scheduling algorithm for flexible flow lines with limited intermediated buffers. *Appl Stochastic Models Data Anal* 1993;9:127–38.
- [11] Johnson SM. Optimal two and three-stage production schedule with setup times included. *Nav Res Logistics Quart* 1954;1(1): 61–8.
- [12] Gonzalez T, Sahni S. Flowshop and jobshop schedules: complexity and approximation. *Oper Res* 1978;26(1):36–52.
- [13] Lee C-Y, Cheng TCE, Lin BMT. Minimizing the makespan in the 3-machine assembly-type flowshop scheduling problem. *Manage Sci* 1993;39(5):616–25.
- [14] Garey MR, Johnson DS. Strong NP-completeness results: motivations examples implications. *J Assoc Comput Math* 1978; 25:499–508.
- [15] Cheng TCE, Sin CCS. A stage-of-the-art review of parallel-machine scheduling research. *Eur J Oper Res* 1990;47:271–92.
- [16] Graham RL. Bounds on multiprocessing timing anomalies. *SIAM J Appl Math* 1969;17:416–29.
- [17] Friesen DK, Langston MA. Evaluation of a multifit-based scheduling algorithm. *J Algorithms* 1986;7:35–59.
- [18] Blazewicz J, Dror M, Weglarz J. Mathematical programming formulations for machine scheduling: a survey. *Eur J Oper Res* 1991;51:283–300.
- [19] Shaukat AB, Hunsucker JL. Branch and bound algorithm for the flowshop with multiple processors. *Eur J Oper Res* 1991;51: 88–9.
- [20] Santos DL, Hunsucker JL, Deal DE. Global lower bounds for flow shops with multiple processors. *Eur J Oper Res* 1995; 80:112–20.
- [21] Fisher ML, Lageweg BJ, Lenstra JK, Rinnooy Kan AHG. Surrogate duality relaxation for job shop scheduling. *Discrete Appl Math* 1983;5:65–75.
- [22] Luh PB, Hoitomt DJ, Max E, Pattipati KR. Schedule generation and reconfiguration for parallel machines. *IEEE Trans Robotics Automat* 1990;6(6):687–96.
- [23] Hoitomt DJ, Luh PB, Pattipati KR. A practical approach to job-shop scheduling problems. *IEEE Trans Robotics Automat* 1990;9(1):1–13.
- [24] Hoitomt DJ, Luh PB, Max E, Pattipati KR. Scheduling jobs with simple precedence constraints on parallel machines. *IEEE Control System Manage* 1990;10(2):34–40.
- [25] Lenstra JK, Rinnooy Kan AHG. Complexity of scheduling under precedence constraints. *Oper Res* 1978;26(1):22–35.
- [26] Portmann MC, Vignier A, Dardilhac D, Dezalay D. Branch and bound crossed with GA to solve hybrid flow shops. *Eur J Oper Res* 1998;107:389–400.
- [27] Van Laarhoven PJM, Aarts EHL, Lenstra JK. Job shop scheduling by simulated annealing. *Oper Res* 1992;40(1):113–25.
- [28] Kolonko M. Some new results on simulated annealing applied to the job shop scheduling problem. *Eur J Oper Res* 1999; 113(1):123–36.
- [29] Nowicki E, Smutnicki C. The flow shop with parallel machines: a tabu search approach. *Eur J Oper Res* 1998; 106:226–53.
- [30] Valls V, Perez AM, Quintanilla SM. A tabu search approach to machine scheduling. *Eur J Oper Res* 1998;106:2300–777.
- [31] Hopfield JJ, Tank DW. Neural computation of decisions in optimization problems. *Biol Cybernet* 1985;52:141–52.
- [32] Gulati S, Iyengar SS. Nonlinear networks for deterministic scheduling. *Proceedings of the International Conference on Neural Networks*, vol. 4, 1987. p. 745–52.
- [33] Arizono I, Yamamoto A, Ohto H. Scheduling for minimizing total actual flow time by neural networks. *Int J Prod Res* 1992; 30:503–11.
- [34] Vaithyanathan S, Ignizio JP. Stochastic neural network for resource constrained scheduling. *Comput Oper Res* 1992;19: 241–54.
- [35] Johnston MD, Adorf HM. Scheduling with neural networks — the case of Hubble space telescope. *Comput Oper Res* 1992; 19:179–89.
- [36] Fang L, Li T. Design of competition based neural networks for combinatorial optimization. *Int J Neural Systems* 1990;1:221–35.
- [37] Pellerin D, Hérault J. Scheduling with neural networks: application to timetable construction. *Neurocomputing* 1994;6(4): 419–42.
- [38] Min H-S, Yih Y, Kim C-O. Competitive neural network approach to multi-objective FMS scheduling. *Int J Prod Res* 1998;36(7): 1749–65.
- [39] Chrystosouris G, Lee M, Domroese M. The use of neural networks in determining operational policies for manufacturing systems. *J Manuf Systems* 1991;10:166–75.
- [40] Hayes PV, Sayeh SI. Supervised neural network approach to optimization as applied to the N-job, M-machine job sequencing problem. *Intell Eng Systems Artif Neural Networks* 1992;2:943–8.
- [41] Rabelo L, Jones A, Tsai J. Using hybrid systems for FMS scheduling. *The 2nd Industrial Engineering Research Conference Proceedings*, 1993, p. 471–5.
- [42] Yih Y, Liang T-P, Moskowitz H. Robot scheduling in a circuit board production line: a hybrid OR/ANN approach. *IIE Trans* 1993;25(2):26–33.
- [43] Luh PB, Zhao X, Wang Y, Thakur TS. Lagrangian relaxation neural networks for job shop scheduling. *Proceedings of International Conference on Robotics and Automation*, Leuven, Belgium, 1998, p. 1799–804.

[44] Luh PB, Zhao X, Wang Y, Thakur TS. Lagrangian relaxation neural networks for job shop scheduling. Proceedings of NSF/DMII Grantees Conference, Long Beach, California, 1999.

[45] Goffman EG, Garey MR, Johnson DS. An application of bin-packing to multiprocessor scheduling. *SIAM Journal of Computing* 1978;7:1–17.