

# Design of Assembly Systems for Modular Products

David W. He and Andrew Kusiak, *Member, IEEE*

**Abstract**—To respond to the challenge of agile manufacturing, companies are striving to provide a large variety of products at a low cost. Product modularity allows to produce different products by combining standard components. One of the characteristics of modular products is that they share the same assembly operations for a part of their structure. The special structure of modular products provides challenges and opportunities for the design of assembly systems. Given a family of modular products, designing low cost assembly systems is an important problem. In this paper, an approach for the design of assembly systems for modular products is proposed. The assembly system is decomposed into two subsystems based on the structure of modular products. The configuration problem of the assembly system is formulated and solved by a tabu search based algorithm.

**Index Terms**—Agile assembly, engineering design, manufacturing, modular products.

## I. INTRODUCTION

TO RESPOND to the challenge of agile manufacturing, companies are striving to meet the changing market requirements at a low cost. The ability to produce a variety of products through the combination of modular components is a meaningful benefit of product modularity. Reference [1] described five different ways that modular products are developed in industry. Component-swapping modularity can be achieved when two or more alternative types of components are paired with the same basic product body to create different product variants. Examples of this type of modularity can be found in automotive industry where different audio cassette decks, windshield glass types, and wheel types for the same automobile model are produced. In the computer industry, different hard disk types, monitor types, and keyboards are matched with the same basic CPU. Component-swapping modularity is often associated with the creation of product variety as perceived by the customer [1]. Examples of combinatorial assembly of product variants are also provided in [2]. In addition to the benefits of product variety, modular designs allow to change the product appearance and functions.

An example of product variants generated through component-swapping is provided in Fig 1.

The multifunctional office desk lamp in Fig. 1 is assembled on a base. Different product variants are produced by changing the fixture.

As pointed out by [3], both the flexibility of manufacturing systems and the product architecture contribute to the ability to

Manuscript received August 15, 1994; revised July 16, 1996. This work was supported in part by the National Science Foundation Grant DDM-9215259. This paper was recommended for publication by Associate Editor J. C. Trinkle and Editor A. Desrochers upon evaluation of the reviewers' comments.

The authors are with the Intelligent System Laboratory, Department of Industrial Engineering, University of Iowa, Iowa City, IA 52242 USA.

Publisher Item Identifier S 1042-296X(97)06966-8.

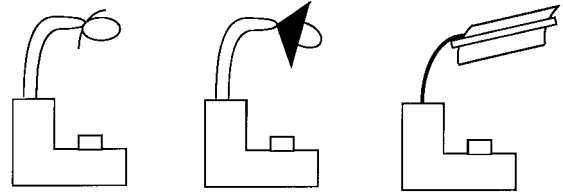


Fig. 1. Product variants accomplished with modular design.

economically create a variety of products. The special structure of modular products provides challenges and opportunities for design of assembly systems. Manufacturing systems have to be designed for configuration at minimum cost and physical movement of manufacturing equipment, i.e., to avoid frequent rearrangement of machine tools (which is expensive, especially, in metal cutting and mechanical assembly processes). Also, the configured system when brought back to its original configuration should assure repeatability of production, i.e., manufacture products and components that meet all the specifications regarding quality, standards, and so on [4]. One of the characteristics of modular products is that they share the same assembly structure for some assembly operations. An assembly system has to be designed to fit the special characteristic of modular products. Given a family of modular products, designing an assembly system at a low cost is an important problem. In this paper, an approach for design of assembly systems for modular products is proposed. The assembly system is decomposed into two subsystems based on the structure of modular products. The configuration problem of the assembly system is formulated and solved by a tabu search based algorithm.

The remainder of the paper is organized as follows. The configuration problem for modular products is defined in Section II. In Section III, a heuristic algorithm for solving the configuration problem is developed. A numerical example is provided to illustrate the heuristic algorithm. The computational experience is discussed in Section IV. Section V concludes the paper.

## II. DESIGN FOR CONFIGURABILITY

### A. Graph Representation of Product Structures

In this paper, a product structure is represented by an acyclic digraph. Nodes represent operations and arcs correspond to the precedence relations between operations. An arc from operation (node)  $f$  to operation (node)  $g$  indicates that operation  $f$  must be completed before operation  $g$  begins. Each assembly operation corresponds to a physical component in a product. For example, a component is represented by one or more corresponding operations, e.g., loading the component,

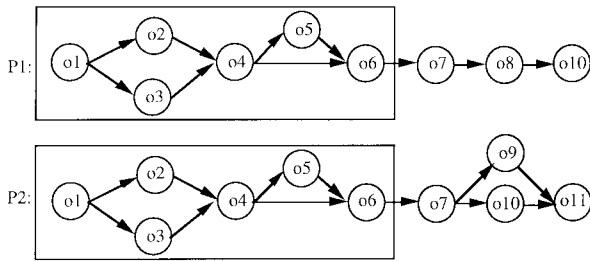


Fig. 2. Two modular products with different variant operations and variant structures.

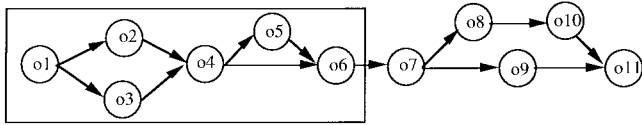


Fig. 3. The superimposed assembly graph for  $P_1$  and  $P_2$  in Fig. 2.

positioning the component on the base, etc. It is assumed that the assembly structure of a modular product includes a basic structure and a variant structure. Each product in a product family shares the same basic structure and differs from others in the variant structure. The operations in the basic structure are referred in this paper as basic operations and the operations in the variant structure are referred to as variant operations. An example of a modular product structure is provided in Fig. 2.

Note that the product variants in Fig. 2 share different variant operations and structures.

Let the assembly structure of product  $P_i$  be represented by graph  $G_i$  with the set  $N_i$  of nodes and the set  $A_i$  of arcs. The assembly structures of  $N$  products can be combined into a superimposed assembly graph  $G$  with  $N$  nodes and  $A$  arcs where

$$N = N_1 \cup N_2 \cup \dots \cup N_N$$

$$A = A_1 \cup A_2 \cup \dots \cup A_N.$$

An arc  $(f, g)$  in  $G$  is redundant if in addition to the arc itself there exists a chain from node  $f$  to node  $g$ . The redundant arcs are omitted. The superimposed assembly graph for products  $P_1$  and  $P_2$  in Fig. 2 is provided in Fig. 3.

Note that any superimposed assembly graph of modular products includes a basic structure and a variant structure. The basic structure in a superimposed assembly graph is the same as the basic structure of any modular product. However, the superimposed variant structure is the superimposition of all variant structures of the modular products.

### B. Design of Assembly Systems for Modular Products

One of the most important operational design issues in the design of an assembly system is the assignment of assembly operations to the stations. Various performance measures are used, e.g., the cycle time or the number of stations is minimized to maximize the line balancing efficiency [5]. Through the assignment of assembly operations to the stations, the line balancing and scheduling are accomplished [6]–[10]. In order to meet the demand for a variety of products without holding an excessive level of inventory, multiproduct assembly systems are used. Reference [11] discussed the difficulty in

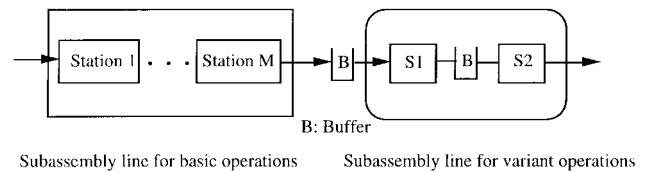


Fig. 4. The assembly line for modular products.

balancing multiproduct assembly systems and the importance of scheduling in achieving the efficiency required. Thus, in the design of an assembly system, the scheduling issue becomes crucial as multiple products are produced. An assembly system has to be reconfigured efficiently over time to accommodate changes in the product mix and designs. Reference [2] pointed out that there is a need to link design of products to an assembly system in order to rearrange the system in response to new technologies or redesign of the products. Assembly systems should be designed to fit the structure of modular products.

In this paper, the assembly system for modular products takes the form of an assembly line. To fit the special structures of modular products, the assembly line is decomposed into two parts: a subassembly line for basic operations and a subassembly line for variant operations. The basic subassembly line is designed as a paced single product assembly line with a fixed cycle time. The variant subassembly line is designed as a flowshop line and balanced by a flowshop scheduling method. Reference [12] showed that an assembly line balanced by a flowshop scheduling approach performs better than an assembly line balanced by a fixed cycle time in terms of total idle time for multiple products. To absorb the unbalanced flow of products, a work-in-process (WIP) buffer between the two subassembly lines is used. An example of the assembly line for modular products is shown in Fig. 4.

### C. The Configuration Problem

In the assembly line for modular products (see Fig. 4), the basic subassembly line can be considered as the core assembly line. Normally, the basic assembly structure forms a major part of the assembly structure shared among modular products. The basic assembly structure of modular products remains relatively stable. The changes in the product structure are often accomplished by changing the variant assembly structure to create product variants to meet the diverse customer requirements. Therefore, the configuration of an assembly line for modular products can be accomplished by redesigning the variant subassembly line. Consider the example in Fig. 5.

The assembly line  $AL_1$  in Fig. 5 is designed for modular products  $P_1$  and  $P_2$ . It consists of basic subassembly line I and variant subassembly line II. Suppose that products  $P_1$  and  $P_2$  are redesigned as products  $P_1'$  and  $P_2'$  and assembly line  $AL_1$  is reconfigured for these changes in product designs. Since the basic assembly structure of the modular products remains unchanged, the structure of basic subassembly line I remains the same. Instead of redesigning the entire assembly line, only the variant subassembly line II is redesigned as  $II'$ . The assembly line configured for new products  $P_1'$  and  $P_2'$  is  $AL_1'$  (see Fig. 5).

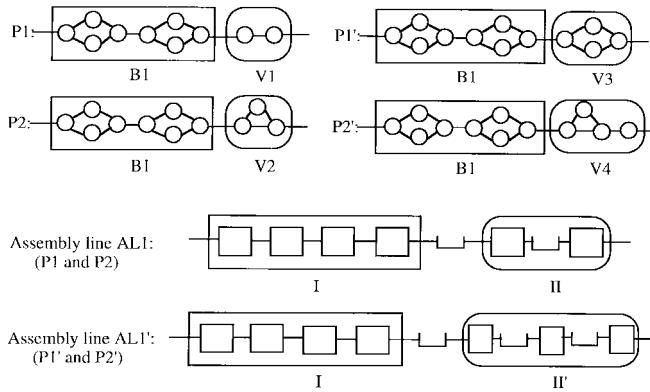


Fig. 5. Configuration of the assembly line for modular products.

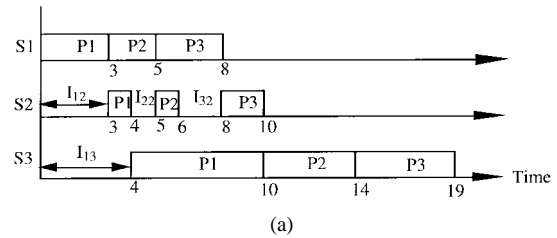
It is assumed that the number of stations in a variant sub-assembly line is constrained which is a reasonable assumption in practice. It is also assumed that this number is greater than 1. When the number of stations in a variant subassembly line is 1, the configuration is trivial and is not considered in this paper.

It is expected that an assembly line is balanced. Since buffers are used in the assembly line to absorb the unbalanced flow of products, the line balance is reflected by the idle time of the stations and the waiting time of products in the buffer. Therefore, the balancing cost of the line can be measured by the total inventory holding cost and the station idle time cost.

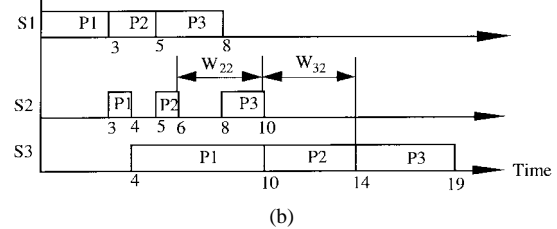
Formally, the configuration problem is defined as follows. Let  $P$  be the set of feasible partitions of operations among  $M$  stations. A feasible partition  $p = \{p(S_1), \dots, p(S_m), \dots, p(S_M)\}$ , where  $p \in P$ , and  $p(S_m)$  is the set of operations assigned to station  $S_m$ . A partition is feasible if each station contains at least one operation and the precedence relations among operations specified in the superimposed assembly graph are not violated. Let  $S(p)$  be the product sequence obtained for  $p$ . Let  $BC[S(p)]$  be the balancing cost corresponding to product sequence  $S(p)$ . The objective of the design procedure is to find partition  $p^* = \{p^*(S_1), \dots, p^*(S_m), \dots, p^*(S_M)\}$  so that  $BC[S(p^*)]$  obtains minimum, i.e.,  $BC[S(p^*)] = \min \{BC[S(p)] \text{ for all } p \in P\}$ .

In order to determine the balancing cost for an assembly line for modular products, the following notation is defined:

- $N$  number of products;
- $M_v$  number of stations in the variant sub-assembly line;
- $M$  total number of stations;
- $Q_i$  quantity of product  $i$ ;
- $Q = \sum_{i=1}^N Q_i$  total quantity of products;
- $S(p) = (1, 2, \dots, Q)$  sequence of products in partition  $p$ ;
- $\tau_{im}^{S(p)}$  processing time of  $i$ th product in  $S(p)$  at station  $m$ ,  $i = 1, \dots, Q$ ,  $m = 1, \dots, M$ ;
- $D_{im}^{S(p)}$  idle time of station  $m$  between the  $i - 1$ st product and the  $i$ th product if  $S(p)$  follows a no-delay flowshop schedule;



(a)



(b)

Fig. 6. The idle time and waiting time for the three products in a flowshop schedule. (a) Idle time. (b) Waiting time.

- $I_{im}^{S(p)}$  idle time of station  $m$  after it finishes processing  $i - 1$ st product and before it starts to process  $i$ th product if  $S(p)$  follows a flowshop schedule;
- $W_{im}^{S(p)}$  waiting time of  $i$ th product in  $S(p)$  in the buffer of station  $m$ ;
- $W^{S(p)}$  the total waiting time required to process products in  $S(p)$ ;
- $I^{S(p)}$  the total idle time required to process products in  $S(p)$ ;
- $\alpha$  unit inventory holding cost;
- $\beta$  unit idle time cost.

Note that  $I_{im}^{S(p)}$  and  $I^{S(p)}$  do not include the transportation time between two stations. Since the transportation time between two stations does not impact the balancing efficiency of an assembly line, it can be either ignored or included in the processing time of a product.

For any given partition  $p$  and product sequence  $S(p)$ , the balancing cost of an assembly line for modular products is computed as follows:

$$BC[S(p)] = \alpha I^{S(p)} + \beta W^{S(p)}. \tag{1}$$

Assume that the flowshop line is not paced. To absorb the unbalanced flow of products, buffers between stations are used. Instead of remaining on each station in a fixed time interval (usually a cycle time), a product leaves the station to the succeeding buffer once its operations are completed at this station. The product waits in the buffer until the succeeding station is available. On the other hand, once a station completes a product it retrieves another product from the buffer. If the preceding buffer is empty, the station will be idle. It can be observed that, in a flowshop assembly line, some products may have to wait in the buffer before they are processed on the next station and some stations may be idle if no products are available in the preceding buffers. For a modular product assembly line, the idle time of stations and the waiting time of the products can be computed directly from a flowshop schedule. For example, consider a three-station flowshop line where three products  $P1$ ,  $P2$ , and  $P3$  are produced.

Fig. 6(a) and (b) shows the idle time and waiting time in a flowshop schedule of the three-station flowshop assembly line. Note that  $I_{22}$  and  $I_{32}$  represent the idle time associated with products  $P2$  and  $P3$  on station  $S2$  in the flowshop schedule, respectively. The idle time in a flowshop schedule represents the time that a station remains idle after it completes the operations of a previous product and before it starts to perform the operations of a current product. For example,  $I_{22}$  represents the time that station  $S2$  remains idle after it completes the operations of product  $P1$  and before it starts to perform the operations of product  $P2$ . Fig. 6(b) shows the corresponding waiting time in the flowshop schedule.  $W_{22}$  and  $W_{32}$  represent the waiting time of products  $P2$  and  $P3$  on station  $S2$  in the flowshop schedule. The waiting time in a flowshop schedule represents the time that a product remains in a buffer between the current station and a next station after its operations are completed and before it is moved to the next station. For example,  $W_{22}$  is the time that product  $P2$  remains in the buffer between stations  $S2$  and  $S3$  after its operations are completed on  $S2$  and before it is moved to station  $S3$ . Note that the times  $I_{12}$  and  $I_{13}$  associated with the first product  $P1$  are not considered as idle times. Therefore, the value of  $I^{S(p)}$  and  $W^{S(p)}$  is computed as follows:

$$I^{S(p)} = \sum_{i=2}^Q \sum_{m=2}^M I_{im}^{S(p)} \quad (2)$$

$$W^{S(p)} = \sum_{i=2}^Q \sum_{m=1}^{M-1} W_{im}^{S(p)}. \quad (3)$$

The value of  $I_{im}^{S(p)}$  in (2) is computed from a no-delay schedule as follows [13]:

$$I_{im}^{S(p)} = \max \left\{ 0, \sum_{s=1}^i I_{s,m-1}^{S(p)} + \sum_{s=1}^i \tau_{s,m-1}^{S(p)} - \sum_{s=1}^{i-1} \tau_{sm}^{S(p)} - \sum_{s=1}^{i-1} I_{sm}^{S(p)} \right\}$$

where

$$I_{1m}^{S(p)} = \sum_{s=1}^{m-1} \tau_{1s}^{S(p)}, \quad \text{for } m = 2, \dots, M, \\ I_{i1}^{S(p)} = 0, \quad \text{for } i = 1, 2, \dots, Q. \quad (4)$$

The value of  $W_{im}^{S(p)}$  in (3) can be computed as follows:

$$W_{im}^{S(p)} = \max \{ C_{i-1,m}^{S(p)} - C_{i,m-1}^{S(p)}, 0 \}$$

where

$$C_{i,m}^{S(p)} = \sum_{s=1}^m \tau_{1s}^{S(p)} + \sum_{s=2}^i \tau_{sm}^{S(p)} + \sum_{s=2}^i I_{sm}^{S(p)} \\ \text{for } i = 2, \dots, Q, \quad m = 1, \dots, M. \quad (5)$$

Considering the basic subassembly line as a single station and equating its cycle time to the station processing time, the assembly line for modular products forms a flowshop line. Therefore,  $M = M_v + 1$  and  $\tau_{i1}^{S(p)} = c_b$  for all  $i = 1, \dots, N$ .

Note that the configuration problem is a combination of two dependent problems. The first problem is to assign variant operations to  $M_v$  stations, i.e., to find a feasible partition  $p$ . The second problem is to schedule  $N$  products in the assembly line, i.e., to find a  $S(p)$  with a minimum value of  $BC[S(p)]$  for all  $p \in P$ . The flowshop scheduling problem is an NP-hard problem [14]. Since the configuration problem involves both partitioning and scheduling at the same time, it is NP-hard. In order to solve the configuration problem in real time, efficient heuristic approaches need to be developed. Next, a heuristic algorithm for solving the configuration problem is developed.

### III. DEVELOPMENT OF A HEURISTIC ALGORITHM

In this section, the heuristic algorithm for solving the configuration problem is developed based on the tabu search approach.

Tabu search is a powerful optimization procedure that has been successfully applied for solving various combinatorial optimization problems, including the traveling salesman problem [15], graph coloring problem [16], and scheduling problem [17]. Some recent applications of tabu search for solving NP-hard problems include [18]–[20]. The tabu search approach is simple to implement and easy to incorporate problem-specific constraints. It may also act as a control mechanism to monitor and direct the progress of other optimization routines.

In general terms, tabu search is an iterative improvement procedure that begins with an initial feasible solution (i.e., a feasible assignment) and attempts to determine a better solution. Tabu search differs from the traditional iterative optimization algorithms in that it has the ability to escape local optima by using a short-term memory of recent solutions (*tabu list*). Moreover, tabu search permits backtracking to previous solutions, which may ultimately lead, via a different direction, to better solutions (*aspiration*). The features of a tabu list and aspiration make tabu search a powerful optimization tool for solving the scheduling problem. A more detailed description of the tabu search method is presented in [21].

The flowchart of the heuristic algorithm is shown in Appendix A. The basic structure of the tabu search follows that of [22]. The algorithm starts with an initial feasible solution generated by a sequential assignment procedure. A feasible solution implies a feasible partition of operations. Then the heuristic algorithm searches for the best partition in its neighborhood. The partition  $p-nb$  associated with the minimum balancing cost in each neighborhood,  $BC[S(p-nb)]$ , is accepted (provided that it is not in the tabu list). The aspiration value is set to be the current minimum cost value, i.e., a tabu partition can be released from its tabu status if its cost value is less than the current minimum cost value. The transition from one partition to another partition is defined as a move. At each move of the search, the current overall minimum solution  $p^*$  is recorded, along with its cost value  $BC[S(p^*)]$ . The stopping criterion is a fixed number of iterations. If after the specified number of iterations the problem solution is not improved, the heuristic algorithm stops and the best solution  $p^*$  and its cost  $BC[S(p^*)]$  are reported.

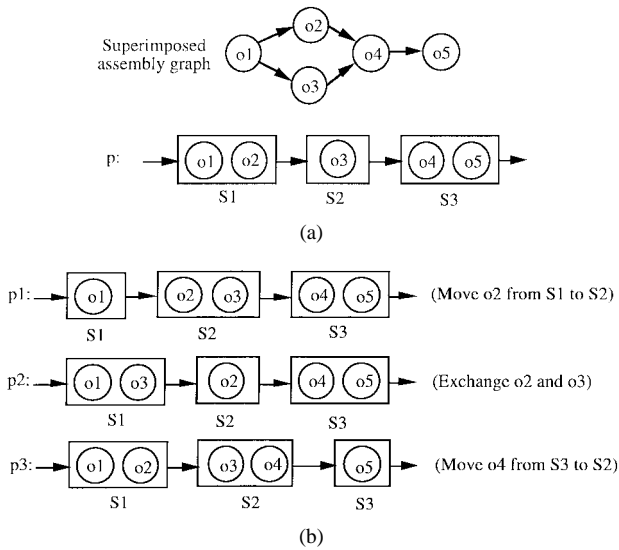


Fig. 7. The example of a neighborhood.

### A. The Neighborhood

Although tabu search is a general approach which can be applied to different types of combinatorial problems, its successful application depends on the definition of neighborhood which is problem specific. In this paper, the neighborhood  $N(p)$  of a feasible partition  $p$  is defined as the set of possible partitions generated from  $p$  by performing one of the following two steps:

- 1) move an operation from a station to one of its neighbor stations;
- 2) exchange operations between two consecutive stations.

Fig. 7 shows the example of a neighborhood. The superimposed assembly graph and the associated feasible partition  $p$  are shown in Fig. 7(a). Fig. 7(b) shows the three possible feasible partitions obtained from  $p$ .

In Fig. 7, the neighborhood  $N(p)$  of  $p$  is  $\{p1, p2, p3\}$ . Note that any partition in a neighborhood must be feasible.

### B. The Initial Solution

The initial solution plays an important role in determining the quality of the solution generated by tabu search. To find a good initial solution, a sequential assignment procedure is developed.

The sequential assignment procedure aims at maximizing the equality of average processing time among all stations.

Define:

- $Q_v$  set of variant operations;
- $J = |Q_v|$  total number of variant operations;
- $D$  set of unassigned operations without any preceding operations;
- $G$  set of operations assigned;
- $V$  set of operations unassigned;
- $G_h$  set of operations assigned to station  $h$ ;
- $t_j$  processing time of operation  $j$
- $t_{ij}$   $t_{ij} = \begin{cases} t_j, & \text{if operation } j \text{ is required by product } i \\ 0 & \text{otherwise;} \end{cases}$

and

$$T(G_h) = \sum_{j \in G_h} T_j \quad \text{processing time at station } h.$$

Compute the aggregated processing time  $T_j$  for each operation:

$$T_j = \sum_{i=1}^N Q_i t_{ij}.$$

Compute the average processing time  $T_{ave}$  at each station:

$$T_{ave} = \frac{1}{M_v} \sum_{j=1}^J T_j.$$

The sequential assignment procedure and an illustrative example are presented in Appendix B.

Having all variant assembly operations assigned to the stations, the assembly line needs to be balanced. Each time the partition of operations is determined, a scheduling procedure is called to find a sequence of products. Once the product sequence is obtained, the balancing cost is computed from (1) to (5). Next, the development of the scheduling procedure is discussed.

### C. The Scheduling Procedure

There are two reasons for using a heuristic to schedule the flowshop assembly line. First, the flowshop scheduling problem is NP-hard. Second, finding an optimal schedule may not lead to the overall optimum.

In the search for a suitable heuristic, effective heuristics minimizing the mean flowtime of the flowshop problem are of a particular interest. The criterion of minimizing the total flowtime has been found to be a practical objective since it equalizes the utilization of resources and reduces the in-process inventory [23], [24].

Two scheduling rules are used in the proposed scheduling procedure. The optimal sequence for minimizing the total flowtime of a single machine is obtained by the shortest processing time (SPT) rule [23]. By extending this principle to flowshop scheduling, it is natural to order the products in a nondecreasing order of their total processing time over  $M$  stations. Hence, the shortest total processing time (STPT) rule is used as the first scheduling heuristic.

The second scheduling heuristic is based on the work of [25]. Rajendran argued that the STPT rule may not always be effective in minimizing the total flowtime. Instead, a product sequence is generated by ordering products in a nondecreasing order of a surrogate processing time, referred to as the adjusted processing time. The adjusted processing time  $R_i$  for product  $i$  can be computed as follows [25]:

$$R_i = \sum_{j=1}^M (M - j + 1)t_{ij}. \quad (6)$$

Hence, the shortest adjusted processing time (SAPT) rule is used as the second scheduling heuristic.

To determine which scheduling rule is more effective in terms of minimizing the balancing cost, variants of the heuristic algorithm with different scheduling rules was tested.

Next, an example is provided to illustrate the solution approach for configuration of the assembly line for modular products.

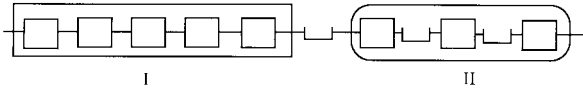


Fig. 8. The assembly line to be reconfigured.

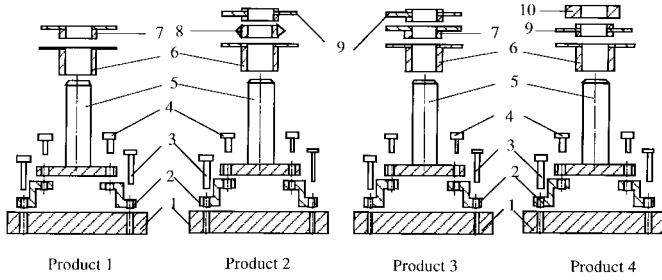


Fig. 9. The designs of modular products.

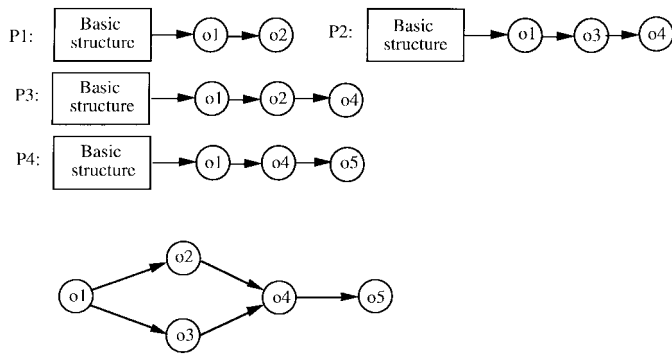


Fig. 10. Assembly operations of modular products. (top) Assembly structure of products. (bottom) The superimposed variant assembly graph.

TABLE I  
LIST OF VARIANT ASSEMBLY OPERATIONS

Operation No.	Description of variant assembly operations
1	Position and secure part 6
2	Position and secure part 7
3	Position and secure part 8
4	Position and secure part 9
5	Position and secure part 10

*Example 1:* The assembly line to be reconfigured is shown in Fig. 8. The designs of modular products are shown in Fig. 9.

The structure of modular products in Fig. 9 is similar to the stack product structure discussed in [26]. The four modular products share the same subassembly that contains components 1–5. Therefore, the assembly operations required to assemble the basic subassembly are the basic operations. Since the basic subassembly component of each product is not subject to changes in the product design, the basic operations are performed in the basic subassembly line. The assembly operations of the modular products are shown in Fig. 10. For the sake of simplicity, the structure of the basic operations is not shown.

The list of variant operations is shown in Table I.

Assume that the cycle time of the basic subassembly line is  $c_b = 4$ . The processing time  $t_{ij}$ , production quantity for each product  $Q_i$ , and the aggregated processing time  $T_j$  of each variant operation are provided in Appendix B.

For the convenience of presentation, only the STPT scheduling rule is used in this example. Assume that the heuristic

algorithm stops if the solution is not improved in one iteration. Set  $\alpha = \beta = 1$ . Let the size of Tabu\_List be 3. Consider an arbitrary initial solution  $p_0 = \{(o1, o2), (o3, o4), (o5)\}$  with  $BC[S(p_0)] = 120$ . Initially, the following conditions are set:

$$\text{Tabu\_List} = \{\emptyset, \emptyset, \emptyset\};$$

$$p^* = p_0, BC[S(p^*)] = BC[S(p_0)].$$

**Iteration 1**

Place  $p_0$  in Tabu\_List:  $\text{Tabu\_List} = \{p_0, \emptyset, \emptyset\}$ . The neighborhood  $N(p_0)$  of  $p_0$  is obtained as follows:

$$p1 = \{(o1, o3), (o2, o4), (o5)\}, BC[S(p1)] = 123;$$

$$p2 = \{(o1), (o2, o3, o4), (o5)\}, BC[S(p2)] = 85;$$

$$p3 = \{(o1, o2), (o3), (o4, o5)\}, BC[S(p3)] = 110;$$

$$p4 = \{(o1, o2, o3), (o4), (o5)\}, BC[S(p4)] = 114.$$

Since  $p_2$  is not in Tabu\_List and  $BC[S(p_2)] = \min\{BC[S(p)]: p \in N(p_0)\}$ , the move from  $p_0$  to  $p_2$  is accepted.

Since  $BC[S(p_2)] < BC[S(p^*)]$ , set  $p^* = p_2$  and  $BC[S(p^*)] = BC[S(p_2)] = 85$ .

**Iterations 2**

Place  $p_2$  in Tabu\_List:  $\text{Tabu\_List} = \{p_2, p_0, \emptyset\}$ . The neighborhood  $N(p_2)$  of  $p_2$  is obtained as follows:

$$p5 = \{(o1), (o2, o3), (o4, o5)\}, BC[S(p5)] = 56;$$

$$p6 = \{(o1, o2), (o3, o4), (o5)\}, BC[S(p6)] = 120;$$

$$p7 = \{(o1, o3), (o2, o4), (o5)\}, BC[S(p7)] = 123;$$

Since  $p_5$  is not in Tabu\_List and  $BC[S(p_5)] = \min\{BC[S(p)]: p \in N(p_2)\}$ , the move from  $p_2$  to  $p_5$  is accepted.

Since  $BC[S(p_5)] < BC[S(p^*)]$ , set  $p^* = p_5$  and  $BC[S(p^*)] = BC[S(p_5)] = 56$ .

**Iterations 3**

Place  $p_5$  in Tabu\_List:  $\text{Tabu\_List} = \{p_5, p_2, p_0\}$ . The neighborhood  $N(p_5)$  of  $p_5$  is obtained as follows:

$$p8 = \{(o1), (o2), (o3, o4, o5)\}, BC[S(p8)] = 62;$$

$$p9 = \{(o1), (o2, o3, o4), (o5)\}, BC[S(p9)] = 85;$$

$$p10 = \{(o1), (o3), (o2, o4, o5)\}, BC[S(p10)] = 113;$$

$$p11 = \{(o1, o2), (o3), (o4, o5)\}, BC[S(p11)] = 110;$$

$$p12 = \{(o1, o3), (o2), (o4, o5)\}, BC[S(p12)] = 53.$$

Since  $p_{12}$  is not in Tabu\_List and  $BC[S(p_{12})] = \min\{BC[S(p)]: p \in N(p_5)\}$ , the move from  $p_5$  to  $p_{12}$  is accepted.

Since  $BC[S(p_{12})] < BC[S(p^*)]$ , set  $p^* = p_{12}$  and  $BC[S(p^*)] = BC[S(p_{12})] = 53$ .

**Iterations 4**

Release  $p_0$  from and place  $p_{12}$  in Tabu\_List:  $\text{Tabu\_List} = \{p_{12}, p_5, p_2\}$ .

The neighborhood of  $p_{12}$  is obtained as follows:

$$p13 = \{(o1), (o3, o2), (o4, o5)\}, BC[S(p13)] = 56;$$

$$p14 = \{(o1, o2), (o3), (o4, o5)\}, BC[S(p14)] = 110;$$

$$p15 = \{(o1, o3), (o2, o4), (o5)\}, BC[S(p15)] = 123.$$

Note that  $p_{13} = p_5$ . Although  $p_{13}$  has the smallest balancing cost, the move to  $p_{13}$  is rejected as  $p_{13}$  is in Tabu\_List and its balancing cost

$BC[S(p13)] < BC[S(p^*)]$ . The move from  $p12$  to  $p14$  with the balancing cost of 110 is accepted.

As in Iteration 4, the solution is not improved, the heuristic algorithm stops with  $p12$  being the best solution found. The product sequence obtained by the STPT rule is  $\{P1(3), P2(2), P4(1), P3(4)\}$  with the balancing cost of 53. For the small problem in Example 1, the solution space contains 9 possible assignments. The heuristic algorithm has evaluated 15 assignments. The disadvantage of using the heuristic algorithm is obvious for this small problem. However, as the problem size increases, the size of the solution space grows exponentially and the use of the heuristic is beneficial.

If the SAPT rule is used for partition  $p12$ , the balancing cost is 49, i.e., lower than for the solution obtained by the STPT rule.

Next, the heuristic algorithm with different scheduling rules are tested and evaluated for large size problems.

#### IV. COMPUTATIONAL EXPERIENCE

To select the most effective heuristic, two variants of the heuristic algorithm were tested. The heuristic algorithm was coded in C and executed on an HP-UX Apollo workstation. During the computational experiment, the size of the Tabu\_List was set to 7. The unit inventory holding cost  $\alpha$  and unit idle time cost  $\beta$  were both set to 1.

The test data generated are characterized by the following parameters:

- $N$  number of products, [20, 200];
- $Q_i$  production quantity of each product  $i = 1, \dots, N, U[40, 100]$ ;
- $M_v$  number of stations in the variant subassembly line, [5, 15];
- $J$  total number of variant operations, [10, 50];
- $t_j$  processing time of variant operation  $j, U[4, 10]$ ;
- $c_b$  cycle time of the basic subassembly line.

The variant structure of each product is assumed. The number of products was in the range [20, 200]. The production quantity of each product was generated from the uniform distribution  $U[40, 100]$ . The number of stations in the variant subassembly line was in the range [5, 15]. The variant structure of a product is characterized by the operations it requires and the precedence relations among them. For each product, the variant structure was randomly generated. The total number of variant operations  $J$  was in the range [10, 50]. The processing time of each variant operation was generated from the uniform distribution  $U[4, 10]$ .

To balance the flow of products between the basic subassembly line and variant subassembly, the cycle time of the basic subassembly line  $c_b$  is set as follows:

$$c_b = T_{ave} = \frac{1}{M_v} \sum_{j=1}^J Q_i t_{ij}. \quad (7)$$

The use of an initial partition generated by the sequential assignment procedure speeds up the search process. Fig. 11 shows the convergence curve for initial partitions generated by the random and the sequential assignment procedure.

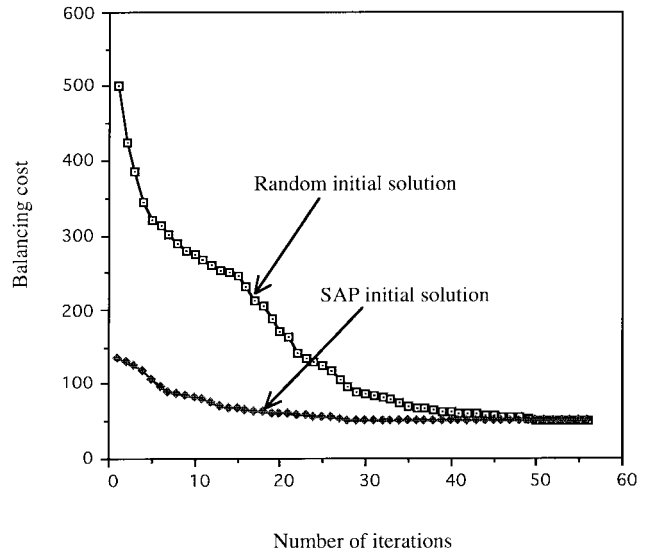


Fig. 11. The tabu search convergence curve for the random and SAP initial solutions.

TABLE II  
THE COMPUTATIONAL RESULTS

Problem data set (N, M <sub>v</sub> , J)	Average BC		Relative Difference BC	CPU time	
	a - STPT	b - SAPT	$ (a-b)/a 100\%$	a - STPT	b - SAPT
(20, 5, 20)	76.8	71.8	7.2%	3.1	5.8
(40, 5, 30)	88.6	82.8	7.0%	4.3	9.6
(50, 8, 30)	102.6	96.8	6.0%	6.2	13.5
(60, 10, 30)	117.4	111.0	5.8%	7.6	15.3
(80, 15, 40)	132.3	125.3	5.6%	9.4	18.4
(100, 15, 50)	150.5	142.9	5.3%	12.5	25.7
(150, 15, 60)	175.4	166.9	5.1%	14.2	28.6
(200, 20, 60)	194.2	185.7	4.6%	15.8	29.3

Eight data sets were considered with 20 problems being solved for each data set. For each problem, two variants of the heuristic algorithm with different scheduling rules were executed. The computational results are shown in Table II.

From the computational results in Table II, one can see that the SAPT rule generates better quality solutions than the STPT rule. However, with the increase of problem size, the difference between the value of the objective functions of the solutions produced by the two scheduling rules decreases. This result is interesting as the STPT rule offers more flexibility than the SAPT rule. Unlike a SAPT sequence, the STPT sequence does not depend on the configuration of the line but the structure of the products. Therefore, using the STPT rule in the heuristic algorithm reduces the computing time. As a STPT sequence does not depend on the structure of the assembly line, it allows for more flexibility in reconfiguring the assembly system.

To evaluate the performance of the heuristic algorithm developed in this paper, the solutions generated by the heuristic algorithm with the STPT rule were compared with the solutions generated by the mixed model line balancing method developed by [7]. Although Thompoulos' method was not developed for modular assembly lines, it is general in nature and applicable to most mixed model line balancing problems.

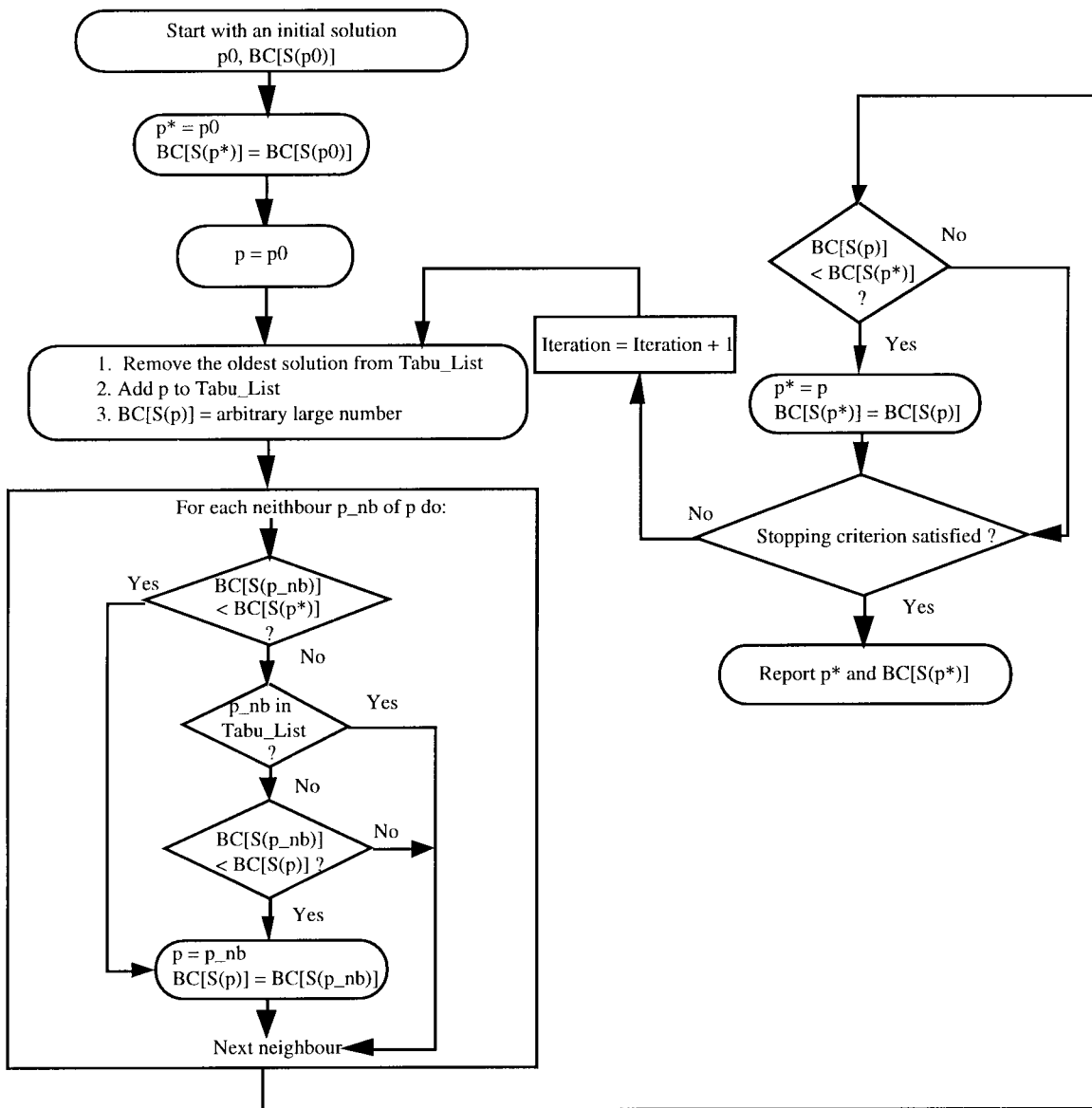


Fig. 12. Flowchart of the heuristic algorithm.

TABLE III  
COMPARISON OF THE SOLUTION QUALITY

Problem data set (N, M, J)	Relative Difference ( $1 - I^a/I^b$ )100%	Problem Data Set (N, M, J)	Relative Difference ( $1 - I^a/I^b$ )100%
(20, 5, 20)	5.6	(80, 15, 40)	10.8
(40, 5, 30)	6.8	(100, 15, 50)	11.2
(50, 8, 30)	7.4	(150, 15, 60)	12.6
(60, 10, 30)	8.7	(200, 20, 60)	15.3

a: I represents the balancing cost of the solution generated by the heuristic algorithm  
b: II represents the balancing cost of the solution generated by Thomopoulos' method

TABLE IV  
COMPARISON OF THE COMPUTATION TIME

Problem data set (N, M, J)	Relative Difference ( $1 - I^a/I^b$ )100%	Problem Data Set (N, M, J)	Relative Difference ( $1 - I^a/I^b$ )100%
(20, 5, 20)	12.6	(80, 15, 40)	17.2
(40, 5, 30)	13.5	(100, 15, 50)	17.8
(50, 8, 30)	15.1	(150, 15, 60)	18.3
(60, 10, 30)	16.8	(200, 20, 60)	18.8

a: I represents the computation time of the solution generated by the heuristic algorithm  
b: II represents the computation time of the solution generated by Thomopoulos' method

In applying Thomopoulos' method, the desired cycle time for each station in the line was set to  $c_b$ . The solutions generated by the heuristic and Thomopoulos' method are compared in Table III.

One can see from Table III that the solutions generated by the heuristic algorithm developed in this paper are better than

those generated by Thomopoulos' mixed model line balancing method in terms of the balancing cost. The computation time of the two methods is illustrated in Table IV.

The solution generated by the heuristic algorithm provides more flexibility for solving the configuration problem. It can be further improved by using other approaches, e.g., the insertion

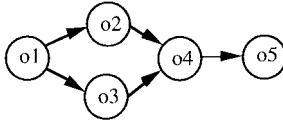


Fig. 13. The superimposed variant assembly graph.

approach presented in [25].

## V. CONCLUSION

To respond to the challenge of agile manufacturing, companies are striving to provide a large variety of products at low cost. Product modularity allows to produce different products through combination of standard components. One of the characteristics of modular products is that they share the same assembly structure for numerous assembly operations. The special structure of modular products provides challenges and opportunities for the design of assembly systems. Given a family of modular products, designing a low cost assembly system is an important issue. In this paper, an approach for design of assembly systems for modular products is proposed. The assembly system is decomposed into two subsystems based on the structure of modular products. The configuration problem of the assembly system is formulated and solved by a tabu search based algorithm. The algorithm considers concurrent partition of assembly operations and scheduling of products to minimize the total balancing cost of the assembly system.

Two variants of the heuristic algorithm with two different scheduling rules were tested. The results of the computational experiments have shown that the SAPT rule is more effective in searching for a good solution of the configuration problem than the STPT rule. However, as the problem size increases, the difference between the solutions generated by the two scheduling rules decreases. This indicates a potential use of the STPT rule as a simple and practical scheduling rule in solving the configuration problem. The solutions generated by the heuristic algorithm were compared to those generated by an existing line balancing method. The comparison results show that the heuristic algorithm developed in this paper outperforms the general balancing method.

The solution approach proposed for solving the configuration problem can also be applied to other types of manufacturing systems which can be decomposed based on the structure of the products to be produced, e.g., cellular manufacturing.

## APPENDIX A

The flowchart of the heuristic algorithm is shown in Fig. 12.

## APPENDIX B

### Sequential Assignment Procedure (SAP)

Step 0. Set the station index  $h = 1$ ; the set of operations assigned to station  $h$ ,  $G_h = \emptyset$ ; the set of operations unassigned,  $V = Q_v$ ; place all operations in  $V$  without any preceding operations in set  $D$ ; set the processing time at station  $h$ ,  $T(G_h) = 0$ , for  $h = 1, \dots, M_v$

- Step 1. While  $T(S_h) < T_{ave}$  do:  
 compute  $f = \arg \min \{T(S_h) + T_j; j \in D\}$ .  
 If  $[T(G_h) + T_f > T_{ave}]$  and  $[T(G_h) + T_f - T_{ave} > T_{ave} - T(G_h)]$ , GoTo Step 2;  
 otherwise:  
 assign operation  $f$  to station  $h$ ;  
 $G_h = G_h \cup f$ ;  
 $V = V \setminus f$ ;  
 $T(G_h) = T(G_h) + T_f$ ;  
 Update  $D$ .
- Step 2. If  $h = M_v$  then assign all unassigned operations in  $V$  to station  $h$ , STOP;  
 otherwise:  
 $h = h + 1$ , GoTo Step 1.

The example presented next illustrates the sequential assignment procedure.

*Example 2:* Find a feasible partition of variant operations among three stations ( $M_v = 3$ ). The superimposed variant assembly graph is shown in Fig. 13.

The processing time  $t_{ij}$ , production quantity for each product  $Q_i$ , and the aggregated processing time  $T_j$  of each operation are as follows.

$$t_{ij} = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} & \begin{matrix} Q_i \\ 3 \\ 2 \\ 4 \\ 1 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ T_j \end{matrix} & \left[ \begin{array}{ccccc} 4 & 5 & 0 & 0 & 0 \\ 4 & 0 & 2 & 7 & 0 \\ 4 & 5 & 0 & 7 & 0 \\ 4 & 0 & 0 & 7 & 3 \\ 40 & 35 & 4 & 49 & 3 \end{array} \right] \end{matrix}$$

The average processing time at each station is computed:

$$T_{ave} = \left\lceil \frac{3(40 + 35 + 4 + 49 + 3)}{3} \right\rceil = 44$$

- Step 0. Set the station index  $h = 1$ ; the set of operations assigned to station  $h$ ,  $G_h = \emptyset$ ; the set of operations unassigned,  $V = Q_v = \{o1, o2, o3, o4, o5\}$ ; place  $o1$  in the set of unassigned operations without any preceding operations  $D$ ,  $D = \{o1\}$ ; set the processing time at station  $h$ ,  $T(G_h) = 0$ , for  $h = 1, \dots, M_v$
- Step 1. Since  $T(G_1) = 0 < T_{ave} = 44$  and  $T(G_1) + T_1 = 40 < 44$ , do:  
 assign operation  $o1$  to station 1;  
 $G_1 = G_1 \cup 1 = \{o1\}$ ;  
 $V = \{o1, o2, o3, o4, o5\} \setminus o1 = \{o2, o3, o4, o5\}$ ;  
 $T(G_1) = T(G_1) + T_1 = 40$ ;  
 $D = \{o2, o3\}$ .  
 Since  $T(G_1) = 40 < T_{ave} = 44$ , do:  
 compute  $f = 3 = \arg \min \{40 + 35, 40 + 4\}$ .  
 Since  $T(G_1) + T_3 = 40 + 4 = 44 = T_{ave}$ , do:  
 assign operation  $o3$  to station 1;  
 $G_1 = G_1 \cup 3 = \{o1, o3\}$ ;  
 $V = \{o2, o3, o4, o5\} \setminus o3 = \{o2, o4, o5\}$ ;  
 $T(G_1) = T(G_1) + T_3 = 44$ ;  
 $D = \{o2\}$ .  
 Since  $T(G_1) = 44 = T_{ave}$ , goto Step 2.

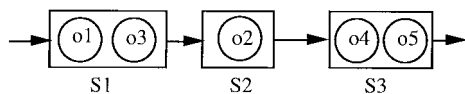


Fig. 14. The initial partition of operations.

- Step 2. Since  $h = 2$ , goto Step 1.
- Step 1. Since  $T(G_2) = 0 < T_{ave} = 44$ , do:  
 compute  $f = 2 = \arg \min \{0 + 35\}$ .  
 Since  $T(G_2) + T_2 = 35 < T_{ave}$ , do:  
 assign operation  $o_2$  to station 2;  
 $G_2 = G_2 \cup 2 = \{o_2\}$ ;  
 $V = \{o_2, o_4, o_5\} \setminus o_2 = \{o_4, o_5\}$ ;  
 $T(G_2) = T(G_2) + T_2 = 35$ ;  
 $D = \{o_4\}$ .  
 Since  $T(G_2) = 35 < T_{ave} = 44$ , do:  
 Compute  $f = 4 = \arg \min \{35 + 49\}$ .  
 Since  $[T(G_2) + T_4 = 35 + 49 = 84 > T_{ave}]$   
 and  $[T(G_2) + T_4 - T_{ave} = 84 - 44 = 40 > T_{ave} - T(G_2) = 44 - 35 = 9]$ ,  
 goto Step 2.
- Step 2. Since  $h = 3$ , assign  $o_4$  and  $o_5$  in  $V$  to station 3,  $G_3 = G_3 \cup o_4 \cup o_5 = \{o_4, o_5\}$ .  
 STOP.

The initial partition of operations among three stations is obtained (see Fig. 14).

#### REFERENCES

- [1] K. Ulrich and K. Tung, "Fundamentals of product modularity," in *Issues in Design Manufacture/Integration*, A. Sharon, R. Behun, F. Prinz, and L. Young, Eds. New York: ASME, 1991, pp. 73–79.
- [2] J. L. Nevins and D. E. Whitney, Eds., *Concurrent Design of Products and Processes*. New York: McGraw-Hill, 1989.
- [3] K. Ulrich, "The role of product architecture in the manufacturing firms," *Research Policy*, vol. 24, no. 3, pp. 419–440, 1995.
- [4] A. Kusiak, "Concurrent engineering: Issues, models, and solution approaches," in *Handbook of Design, Manufacturing and Automation*, R. Dorf and A. Kusiak, Eds. New York: Wiley, 1994, pp. 35–49, ch. 3.
- [5] S. Ghosh and R. J. Gagnon, "A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems," *Int. J. Production Res.*, vol. 27, no. 4, pp. 637–670, 1989.
- [6] N. T. Thomopoulos, "Line balancing sequencing for mixed-model assembly," *Manage. Sci.*, vol. 14, no. 2, pp. 59–75, 1967.
- [7] ———, "Mixed model line balancing with smoothed stations assignments," *Manage. Sci.*, vol. 16, no. 9, pp. 593–603, 1970.
- [8] E. M. Dar-El and R. F. Cothor, "Assembly line sequencing for model-mix," *Int. J. Production Res.*, vol. 13, no. 5, pp. 463–477, 1975.
- [9] E. M. Dar-El and S. Cucuy, "Assembly line sequencing for balanced assembly lines," *OMEGA*, vol. 5, no. 3, pp. 333–342, 1977.
- [10] C. A. Yano and R. Rachamadugu, "Sequencing to minimize work overload in assembly lines with product options," *Manage. Sci.*, vol. 37, no. 5, pp. 572–586, 1991.
- [11] W. Chow, *Assembly Line Design: Methodology and Applications*. New York: Marcel Dekker, 1990.
- [12] D. W. He and A. Kusiak, "Design of an assembly line for modular products," Rep. ISL94-02, Intell. Syst. Lab., Dep. Indust. Eng., The Univ. Iowa, 1994.

- [13] D. W. He, A. Kusiak, and A. Artiba, "A scheduling problem in glass manufacturing," *IIE Trans.*, vol. 28, pp. 129–139, 1996.
- [14] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Math. Oper. Res.*, vol. 1, no. 1, pp. 117–129, 1976.
- [15] F. Glover and H. J. Greenberg, "New approaches for heuristic search: A bilateral linkage with artificial intelligence," *Euro. J. Oper. Res.*, vol. 39, no. 1, pp. 119–130, 1989.
- [16] A. Hertz and D. de Werra, "Using tabu search techniques for graph coloring," *Comput.*, vol. 39, no. 2, pp. 345–351, 1987.
- [17] M. Widmer and A. Hertz, "A new heuristic method for the flow shop sequencing problem," *Euro. J. Oper. Res.*, vol. 41, no. 1, pp. 186–193, 1989.
- [18] P. Hansen, E. de Luna Pedrosa Filho, and C. C. Ribeiro, "Location and sizing of offshore platforms for oil exploration," *Euro. J. Oper. Res.*, vol. 58, no. 2, pp. 202–214, 1992.
- [19] A. P. Punnen and Y. P. Aneja, "Categorized assignment scheduling: A tabu search approach," *J. Oper. Res. Soc.*, vol. 44, no. 7, pp. 673–679, 1992.
- [20] J. Skorin-Kapov and A. J. Vakharia, "Scheduling a flow-line manufacturing cell: A tabu search approach," *Int. J. Prod. Res.*, vol. 31, no. 7, pp. 1721–1734, 1993.
- [21] F. Glover, "Tabu search: A tutorial," *Interfaces*, vol. 20, no. 4, pp. 74–94, 1990.
- [22] J. A. Bland and G. P. Dawson, "Tabu search and design optimization," *IEEE Computer-Aided Design*, vol. 23, pp. 195–201, 1991.
- [23] K. R. Baker, *Introduction to Sequencing and Scheduling*. New York: Wiley, 1974.
- [24] S. French, *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*. Chichester, U.K.: Ellis Horwood, 1982.
- [25] C. Rajendran, "Heuristic algorithm for scheduling in a flowshop to minimize total flowtime," *Int. J. Prod. Econom.*, vol. 29, no. 1, pp. 65–73, 1993.
- [26] M. M. Andreasen, S. Kahler, and T. Lund, *Design for Assembly*. Bredford, U.K.: IFS, 1988.



**David W. He** received the Ph.D. degree in industrial engineering from the University of Iowa, Iowa City.

He is a Visiting Assistant Professor at the American University of Armenia. He is interested in agile manufacturing, design of manufacturing systems, production management, and optimization. He has authored numerous research papers in journals sponsored by the ASME, IEEE, and IIE.



**Andrew Kusiak** (M'90) is a Professor with the Department of Industrial Engineering, University of Iowa, Iowa City. His interests include engineering design, manufacturing, medical technology, and applications of artificial intelligence and optimization. He has authored and edited numerous books, and published research papers in journals sponsored by various societies, including AAAI, ASME, IEEE, IIE, INFORMS, ESOR, IFIP, IFAC, IPE, ISPE, and SME.

Dr. Kusiak serves on the editorial boards of a number of journals, edits book series, and is the Editor-in-Chief of *The Journal of Intelligent Manufacturing*.