Int J Adv Manuf Technol (2006) 29: 980–989
DOI 10.1007/s00170-005-2584-2

ORIGINAL ARTICLE

Bülent Çatay · Asoo J. Vakharia · S. Selçuk Erengüç

Printed circuit board scheduling in an openshop manufacturing environment

Received: 15 November 2004 / Accepted: 28 January 2005 / Published online: 23 November 2005

Abstract Printed circuit boards are used in a myriad of commercial and consumer products in today's marketplace. In this paper, we address the problem of inserting/placing components on printed circuit boards using automated insertion/placement machines. The motivation to examine this problem stems from joint work with a major electronics manufacturer who not only introduced us to the manufacturing environment but also provided data for testing and validating our component allocation and scheduling methodology. The results of our analysis can be used to trade-off operational savings of reduced cycle times versus tactical costs of reorganizing the equipment on the shop floor. Further, we also are able to develop and validate a three step hierarchical scheduling methodology for use in similar manufacturing environments.

Keywords Electronics manufacturing · Openshop scheduling · PCB · Printed circuit board manufacturing · Surface mount technology

1 Introduction and motivation

Printed circuit boards are used across a variety of commercial and consumer products. The most well known of these products is the personal computer where the "motherboard" forms the basis of this ubiquitous machine. The lesser known commercial products using printed circuit boards range from control panels used in jet aircraft, automobiles, and satellites while other consumer products include hand-held personal digital assistants, printers, and of course, cellular phones.

One of the major steps in printed circuit board (PCB) manufacturing is the insertion and/or placement of various electrical and electronic components on a PCB. In general, there is substantial variation in the technology of circuit boards and components. Modern PCB assembly typically uses computer-controlled machines to automatically insert and/or place electronic components on PCBs. Production control of this process deals with several problems such as the: (1) allocation of electronic components to each machines, (2) component feeder arrangement on each machine, (3) sequencing of placement operations, and (4) sequencing of PCBs for production. A comprehensive review of the literature in the PCB assembly research area may be found in Kear [1], McGinnis et al. [2], and Askin et al. [3].

In this paper, an assembly system operated by a major electronics manufacturer using eight identical HS-180 automated insertion machines is studied. Each machine places electronic components onto PCBs using surface mount technology and these components are loaded onto one or more of the 180 distinct feeder slots. The current machine layout used by the manufacturer is a flow-line configuration with each PCB batch "visiting" each machine. Thus, an entire batch of a PCB type is introduced on the first machine and routed through all the other seven machines in the line. On completion of the entire PCB type batch, a new PCB type is introduced on the first machine and the sequence is repeated until all the PCB types are processed through the entire line.

The system described above is quite typical of most current PCB placement operations used in contemporary automated industrial settings (see, for example, the system described in Balakrishnan and Vanderbeck [4]). Our focus in this paper is on studying the component loading, and PCB sequencing and scheduling problem for this specific system. The remainder of this paper is organized as follows. In Sect. 2, prior literature on PCB assembly operations is reviewed and in Sect. 3, we provide details of the industrial system we study. Sect. 4 describes the specific methodology developed for addressing the component loading and PCB sequencing and scheduling problem. In Sect. 5,

B. Çatay (✉)
Faculty of Engineering and Natural Sciences,
Sabancı University,
Tuzla, 34956 Istanbul, Turkey
E-mail: catay@sabanciuniv.edu
Tel.: +90-216-4839531
Fax: +90-216-4839550

A.J. Vakharia · S.S. Erengüç
Department of Decision and Information Sciences,
Warrington College of Business,
University of Florida,
351 Stuzin Hall, Gainesville, FL 32611-7169, USA

we describe the results of applying this methodology to data obtained from the electronics manufacturer and also evaluate the solution quality of our methodology to changes in input data through a set of randomly generated problems. Finally, Sect. 6 highlights the implications and conclusions of our research.

2 Relevant literature

Over the last two decades, many researchers have studied the assembly process for PCBs in an effort to develop models to support the process planning, production planning, and scheduling decisions. Typical objectives in these decision models include minimizing cycle time (e.g., Drezner and Nof [5], Ball and Magazine [6], Ahmadi et al. [7], Francis et al. [8], Grotzinger [9]) and/or setup time (e.g., Lofgren and McGinnis [10], Carmon et al. [11], Maimon and Shtub [12], Jain et al. [13]).

There is also a substantive amount of research focusing on the component allocation problem in PCB manufacturing. For example, Lofgren et al. [14] group similar component types and then assign these groups to balance the workload among machines, while Ammons et al. [15] address the problem of allocating components to workstations to balance workloads and to minimize workstation visits in an environment where an assembly visits a workstation only once. Other authors who have studied the workload balancing problem are Fathi and Taheri [16] and Rajan and Segal [17]. Crama et al. [18] describe a hierarchical decomposition and mathematical programming approach for a line with several placement machines which are devoted to the assembly of a single product with restrictions on the component types that can be loaded. Brandeau and Billington [19] formulate a mixed-integer programming problem for component allocation over a long term planning horizon and present several heuristics to minimize the total sum of all setups and processing costs rather than balancing the workload. Ammons et al. [20] develop an integer programming to allocate components on multiple, nonidentical machines to minimize board assembly and machine setup time for each board type over all machines and present two heuristic solution procedures. Leipälä and Nevalainen [21] propose a model to solve component placement sequence and feeder assignment problems simultaneously. Günther et al. [22] present a recursive heuristic for assigning components to jobs and then assigning the selected components to feeders. Hillier and Brandeau [23] propose a model for assigning PCBs with the primary objective of minimizing the makespan and a secondary objective of balancing the workload.

In a related paper, Balakrishnan and Vanderbeck [4] consider a partial setup strategy of mounting frequently used components permanently on each machine and present a model to assign product families to parallel assembly lines incorporating both workload balancing and setup time minimization objectives. Partial setup strategies have also been discussed by Leon and Peters [24] in the case of a single placement machine to minimize the makespan and by Peters and Subramanian [25] in the case of multiple placement machines operating in parallel to balance the tradeoff between processing time and changeover time.

Ahmadi et al. [7] and Tang [26] propose the use of duplicate reels of each component type. Additional feeder tapes of the same component can be placed onto different machines to balance the workload or onto the same machine to improve the cycle time. Although this practice may improve the solution in certain cases, it requires additional investment in parts, planning, and setup. Lofgren et al. [14] discuss the component allocation problem with precedence constraints and prove that the general problem of minimizing the number of workstation visits is NP-hard. They also show that standard heuristics have arbitrarily bad worst case performance.

One of the unique features of the industrial environment we study is that precedence constraints on the order of component placements on PCBs do not exist. In such a context, the machine scheduling problem can be studied in the context of a parallel machine openshop scheduling problem. In general, the openshop machine scheduling problem is defined as follows: n jobs (J_1, $\ldots$, J_n) have to be processed by m machines (M_1, $\ldots$, M_m). Job J_i, $i = 1, \ldots, n$, requires at most m operations (O_{i1}, $\ldots$, O_{im}), where O_{ir} is the operation performed on machine r for job i. Operation O_{ir} has to be processed (uninterrupted if preemption is not allowed) for t_{ir} time units on machine M_r, $r = 1, \ldots, m$. t_{ir} is a nonnegative integer. In an openshop, there is no restriction on the order in which the operations of a job are to be performed. Each job can be processed on at most one machine at a time and each machine can process at most one operation at a time. An extensive discussion and review of the literature on the openshop scheduling area may be found in Dror [27] and Vakharia and Çatay [28].

Based on this review of the literature, we note the following:

- Minimizing cycle time (or flow time) appears to be one of the more critical performance measures used in this context. Further, static scheduling studies have also considered minimizing the makespan as another performance measure of interest in this environment.
- If there is not adequate capacity to load all components on machines prior to processing multiple PCB types, then minimizing the changeover (setup) times is also critical.
- In assigning components to machines, there is remarkable consensus that load balancing is the underlying criterion of interest.
- If precedence constraints in terms of component placements do exist, then we encounter a job (or flow) shop scheduling problem while if such constraints are not relevant, then we have an openshop scheduling problem.

We now proceed to describe our industrial environment in more detail and outline the procedure developed to address the component loading and PCB scheduling problem for this specific context.

3 Industrial environment

In this paper, we study an industrial environment with the following features. There are eight automated component insertion

machines. Each machine has 180 feeder slots available for loading components. Based on an anticipated 3 month build schedule, there are 22 distinct PCB types which are processed through these eight machines. For each PCB, a bill of components specifying the component types, the number of each type of component, and the feeder slots that each component requires on each machine is available. There are 290 distinct component types placed on the PCBs. Of course, not all component types are placed on each PCB and each component type occupies three, six, or nine feeder slots on a machine.

After consulting with the company personnel, we were also able to identify several unique features of this system:

- There is adequate total feeder capacity across all machines to load all component types, and thus, there is zero time spent from switching between one PCB type to another. As a result of this, we assume that the company operates the system by loading all the distinct component types on machines prior to starting the placement operations on each PCB. This indicates that there is no focus on minimizing setup times for our application but on the other hand, the component allocation problem still needs to be addressed.
- The operator controlling the entire line of machines inputs the specified PCB sequence with required quantities (based on the master production schedule) prior to the line starting operations on the entire set of PCBs. Given the automated nature of the line, this input is simultaneously "loaded" on each machine on the line. In this context, there is of course a need to determine the sequence of PCB's which should be processed through the entire line.
- In line with current company policies, we do not consider lot splitting in our analysis. Consequently, an entire batch of a PCB type is processed through a single machine before it becomes available for placement operations on the next machine.
- There is very little variation in the placement time per component across components and PCBs. Thus, the actual placement time for a component on a PCB is strictly a function of the number of components of a particular type required to be placed on the PCB. Thus, t_{ir}'s (or the insertion time for PCB i on machine M_r) are estimated by summing the component requirements of PCB i for all components loaded on machine M_r.
- The company works off a 3 month frozen master production schedule specifying the number of PCBs of each type required by the end of the 3 month period.

For this environment, we study the component allocation and PCB sequencing problem. Note that through a component allocation scheme we determine t_{ir} and in turn, these are used to develop a sequence for scheduling PCBs. Once we have a component allocation scheme, the PCB scheduling problem differs in terms of machine layouts. As noted earlier, the eight machines are currently organized as a flow-shop. On the other hand, given that the sequence of component insertions on each PCB does not matter (i.e., there are no precedence requirements), we can view the machine layout as being that of an openshop. It is fairly obvious that an openshop layout is more flexible and would provide better performance (in terms of makespan and mean flow time) as compared to a flow shop layout. Consequently, the focus of our component allocation and PCB scheduling approach (outlined in the next section) assumes an openshop scheduling environment.

4 Component loading and PCB scheduling methodology

We address the component allocation and scheduling problem in three stages. The first stage focuses on identifying component groups (families); the second stage loads component families to the automated placement machines; and the third stage develops a sequence for scheduling the PCBs on the machines. In this final stage, the primary objective is to minimize the maximum completion time (makespan) with the secondary objective of reducing the mean flow time. Minimizing maximum completion time would either increase the production capacity or reduce work-in-process inventory, and reducing the mean flow time would also reduce work-in-process inventory without adversely affecting production capacity.

An overview of our proposed methodology is shown in Fig. 1. In the first stage, component families are created in order to maximize the similarity between components without exceeding maximum slot capacity for each family. Our motivation in grouping components into families is to obtain a schedule with fewer visits to machines. We use three similarity measures based on the commonality of the components to be placed on the PCBs and the number of units placed. The definitions of similarity measures we consider are included in the Appendix 1 (see Anderberg [29] for an extensive review of matching coefficients). In the second stage, we assign component families to the insertion machines attempting to balance the workloads. When assigning these component families, we attempt to ensure that components families loaded on a machine are similar (in terms of PCB usage). In the third stage, we schedule PCBs to machines in order to minimize the makespan with the secondary objective of reduc-

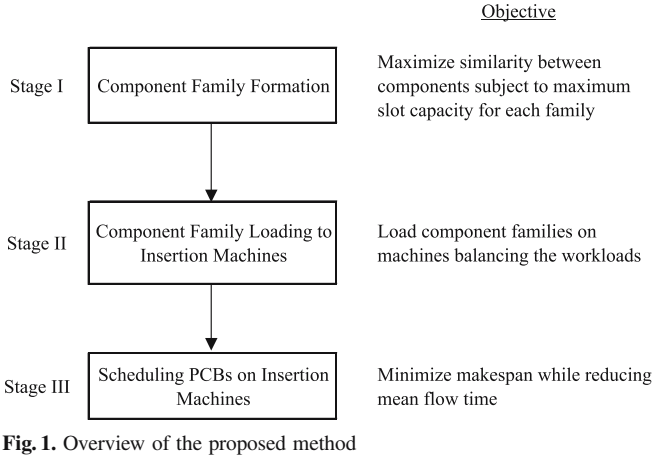

Fig. 1. Overview of the proposed method

ing the mean flow time. We now proceed to describe details of each stage.

Stage I: component family formation

The component family formation problem can be modeled as a 0-1 integer program as follows:

$$\max \sum_{i\in R}\sum_{j\in R} S_{ij}x_{ij} \tag{1}$$

$$s.t. \sum_{j\in R} x_{jj} = F \tag{2}$$

$$\sum_{j\in R} x_{ij} = 1, \quad i \in R \tag{3}$$

$$\sum_{i\in R} f_i x_{ij} \le C, \quad j \in R \tag{4}$$

$$x_{ij} \le x_{jj}, \quad i, j \in R \tag{5}$$

$$x_{ij}, x_{jj} \in \{0, 1\}, \quad i, j \in R \tag{6}$$

where i, j are indices for component types and R is the set of component types. S_{ij} is the similarity coefficient between component i and component j chosen as a median, F is the number of medians to be identified, f_i is the number of feeder slots required by component type i, and C is the feeder slot capacity of each machine. The binary decision variable x_{ij} designates whether component i is assigned to median j; $x_{jj} = 1$ indicates the selection of component j as a median.

The objective function (1) maximizes the sum of similarities between each pair of component types. Constraint (2) sets the number of component families, constraint set (3) ensures that all component types are assigned to a single median, constraint set (4) enforces the feeder slot capacity on the machines, and constraints (5) ensure assignment of a component to a median that is not selected yet. Thus, the above clustering problem's objective is to identify F components as medians and assign the remaining components to those medians as homogeneously as possible based on group technology principles, while not violating the capacity requirements of the machines.

Since this model is not practically tractable (Mulvey and Beck [30]), we propose an efficient grouping algorithm to derive good solutions to this problem. An overview of the three major steps in our procedure is as follows:

- First, a set of F components is obtained as candidate medians using an initial heuristic. Two different heuristics are used for the initialization process (namely, IA I and IA II – see Appendix 2).
- Second, component types are assigned to their most similar median without violating the feeder slot capacity. After all components are assigned, the sums of similarities between each component and all other components assigned to the same median are computed. If a component type other than the selected median achieves a greater similarity level in any family, the median is changed to that component type and the set of medians is updated.
- Finally, after the above process is complete, all possible interchanges between pairs of component types assigned to two different medians are considered. If an interchange can improve the objective function value without violating the capacity constraint, the two component types are swapped. The procedure is repeated until no further feasible exchange is possible.

Each step of the procedure is described in extensive detail in Appendix 2.

Stage II: component family loading

After all the medians are identified and each component is assigned to a particular median, the median and the assigned components form a component family. Component families are allocated to machines attempting to keep the workload on the most utilized machine as low as possible. First, we compute the processing time needed to place all the components in each family onto the PCBs that require them. Then, component families are loaded on machines such that total number of feeders required does not exceed slot capacity C and maximum of the total processing times is minimized to achieve a balanced workload. The algorithm is as follows:

Step 0: Compute the total processing time of each component family.
Step 1: Load families for which total feeder slot requirements are at capacity C. (dedicating the machine to a unique component family)
Step 2: Set the upper bound on the machine workload to the largest processing time among all families.
Step 3: Load the remaining component families in the non-increasing order of their processing times to the least loaded machine up to the upper bound on the workload. Skip families for which the machine does not have sufficient feeder slot capacity.
Step 4: If the machine is loaded up to its slot capacity, repeat step 3 for the remaining machines.
Step 5: If there exist unloaded component families, increase the upper bound on the workload by the smallest processing time among all remaining families. Go to step 3.

Stage III : scheduling PCBs on machines:

Given that the primary objective is to minimize the maximum completion time with a secondary objective of reducing the mean flow time, PCB scheduling is carried out as follows. First, we assign PCBs to machines based on their component requirements. After all of the assignments are completed, we create a PCB-machine processing time matrix where each entry represents the time based on the number of components each PCB requires. Next, the machines are sorted in the non-increasing order of their total workload:

$$\sum\nolimits_{i=1}^{n} t_{i1} \ge \sum\nolimits_{i=1}^{n} t_{i2} \ge \ldots \ge \sum\nolimits_{i=1}^{n} t_{im}.$$

The logic behind this largest total workload (LTW) type ordering is that the machine currently having the maximum workload is the bottleneck in the system and boards are scheduled to that machine first so that it will have no idle time (in the case of the first machine) or the least amount of idle time (in the case of the following machines). This LTW type ordering is employed in an effort to minimize the makespan. Then, the boards are sorted in the non-decreasing order of their machine processing times. This shortest processing time (SPT) type ordering is applied to reduce the mean flow time and PCBs are then scheduled on each machine. If a PCB is currently assigned to a previous machine, it is scheduled at an earlier time if the machine has sufficient idle time. Otherwise, it is delayed until its previous operation has been completed.

The scheduling algorithm is performed as follows:

Step 0: Load PCBs to machines where the components they require have been loaded and create PCB-machine processing time matrix.

Step 1: Sort the machines in non-increasing order of the total processing times.
(LTW rule for machines)

Step 2: Sort PCBs on each machine in non-decreasing order of their processing time on that machine. (SPT rule for PCBs)

Step 3: Starting with the machine that has longest processing time, i.e., starting with the first machine in the sequence, schedule PCBs in the SPT order. If a PCB is being processed on another machine during a certain time window, schedule it to an earlier time window, provided that the machine is idle and PCB is not being processed. Otherwise, delay it until its operation on the other machine is completed.

This methodology was coded in Pascal and implemented on a PC. We now proceed to describe an application of this methodology in the industrial environment studied in this paper.

5 Application of the methodology

Before discussing the results of applying the methodology to industry data, two aspects need to be clarified. First, we carry out a data reduction step. The focus is on reducing the size of the component set by combining components required to be inserted on two or more PCBs. Thus two or more components which are inserted on the same PCB or PCBs are merged so as to be considered as a single component type, *provided* that the sum of the number of slots required by the new component does not exceed the machine capacity of 180 slots that are available. By doing so, we obtain a smaller component set that consists of 74 distinct component types (as compared to the original 290 components). This helped to reduce the memory requirements and run time of our methodology, and to prevent unnecessary data handling. Accordingly, we update the slot requirements of the new component types and the number of units mounted on each PCB. An example of the reduction in component types resulting from this step is shown in Table 1 where we detail the information for PCB type #19 (originally requiring ten distinct component type insertions and now requiring only six component type insertions).

Second, the grouping and improvement algorithms determine component families for the specified number of medians (stage I). A large range of families (eight to 40) was examined for experimental purposes. The reason for doing so is that the component family population and number of families may have a significant affect on the assignment of components to machines, therefore on the schedule. Since the processing times for PCB types on each machine differ significantly and component family formation algorithms do not require any intensive computations it may be useful to investigate different number of families to compare the performance results.

The results discussed next are organized as follows. First, we examine the impact of how the three similarity indices and the two initialization heuristics impact performance. Next, we discuss the impact of the scheduling heuristic on performance. Finally, we report the results of applying our methodology to 25 additional randomly generated problems with parameter settings reflecting the industry data.

Impact of similarity indices and initialization heuristics

The charts in Fig. 2 show the makespan and mean flow times obtained for different number of component families using the three similarity coefficients (SC). First, it is quite clear that we get better performance (in terms of makespan and mean flow

Table 1. Component information for PCB type #19

Prior to reduction			After reduction		
Component reference	Process time (units mounted)	Feeder slots required	Component reference	Process time (units mounted)	Feeder slots required
74	48	6	55	48	6
79	10	6	28	10	6
95	1	6	60	1	6
179	1	6	64	1	6
177	3	6	72	13	24
358	4	6			
468	5	6			
469	1	6			
465	1	3	70	3	6
466	2	3			

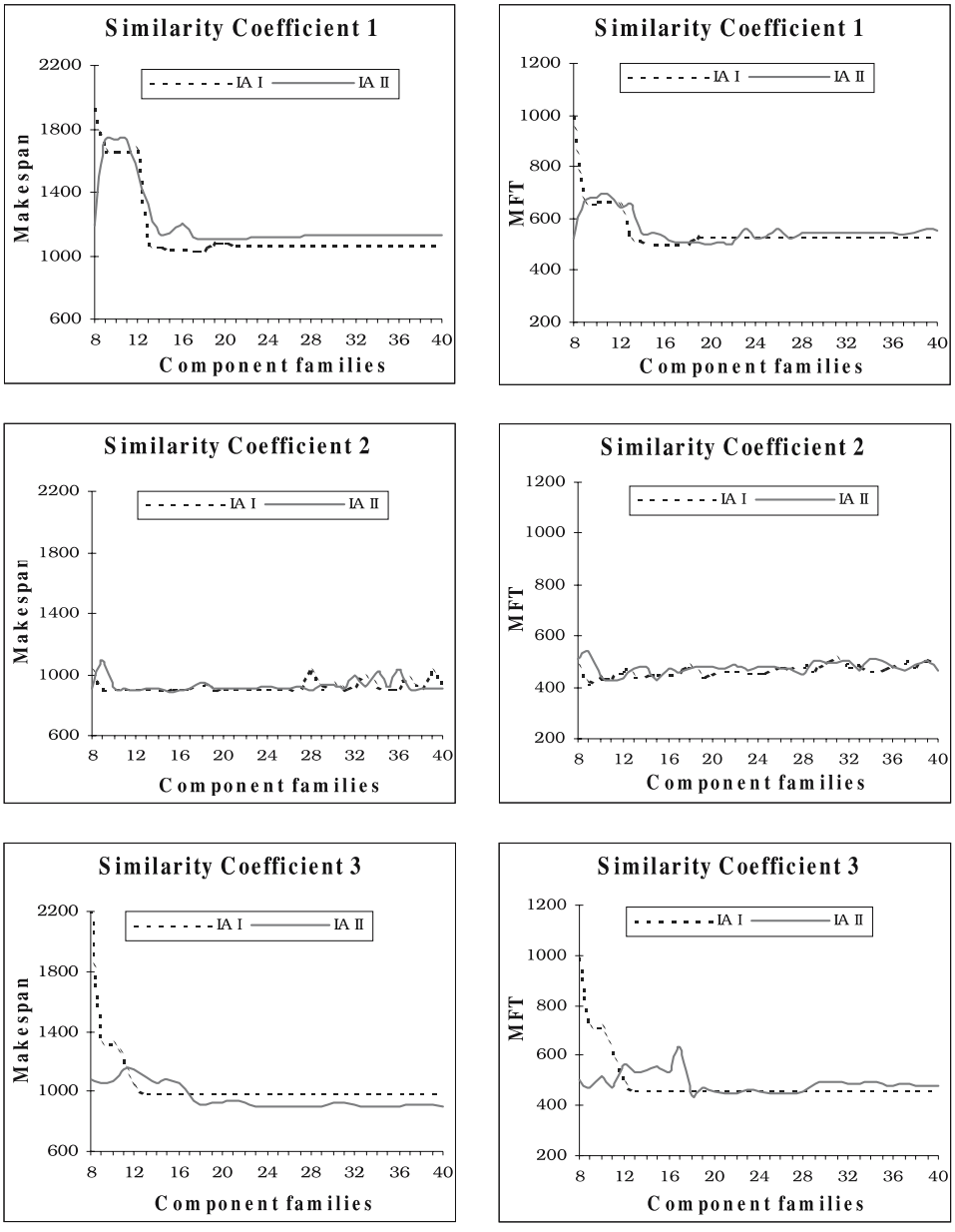

Fig. 2. Makespans and MFTs using different similarity coefficients

time) when the number of component families is 14 or more. Second, there is no question that similarity measures SC2 and SC3 dominate measure SC1 for this data (as noted in Vakharia and Wemmerlov [31], performance of similarity measures are data specific). Finally, there are also differences between the performance of the two initialization algorithms used at this stage. Although these differences are not substantive for SC1 and SC2, for SC3, IAI is quite dominant for makespan while IA II is a better performer for mean flow time.

Performance of scheduling algorithms

For evaluating the performance of the scheduling algorithm, out of 33 different numbers of families examined (eight thru 40), the makespan is achieved at its lower bound[1] in 43 instances using SC1 (in ten using IA I and 33 using IA II), in 57 instances using SC2 (in 28 using IA I and 29 using IA II), and in 64 instances using SC3 (in 32 using IA I and 32 using IA II). The deviation from the lower bound is 1.14% on the average and 12.85% in the worst case. The results obtained using two initialization algorithms are summarized in Table 2. The best makespan achieved using each similarity measure is reported along with the corresponding number of families. The percent difference between the heuristic makespan and optimal makespan is indicated in the gap column. The optimal makespan (of 868) is found by solving the

[1] The lower bound is determined as the largest total processing time across all machines after the component families are loaded.

Table 2. Summary of results for industrial data

		Number of families	Makespan	% Gap	MFT
IA I	SC 1	18	1033	15.97	499
	SC 2	14	893	2.80	443
	SC 3	13	992	12.50	465
IA II	SC 1	22	1106	21.52	498*
	SC 2	15	883	1.70	425*
	SC 3	36	894	2.91	480

* indicates smallest MFT across all numbers of families examined

following min-max generalized assignment problem[2]:

$$\min z$$

$$s.t. \sum_{i \in R} t_i x_{im} \leq z, \quad m \in M$$

$$\sum_{m \in M} x_{im} = 1, \quad i \in R$$

$$\sum_{i \in R} f_i x_{im} \leq C, \quad m \in M$$

$$x_{im} \in \{0, 1\}, i \in R, \quad m \in M$$

[2] The optimal makespan of the reduced problem is obtained using CPLEX optimization library (CPU time is 1608 s on a 300Mhz PC with Pentium II processor) for the purpose of comparison. The optimal makespan of the original (larger) problem could not be determined.

where M is the set of machines and t_i is the total number of components of type i to be mounted on all PCBs.

Although it is not of our primary focus, we observe that component family formation reduces the number of machines to be visited substantially compared to the optimal solution obtained from min-max GAP. In all of the instances, a fewer number of visits is achieved compared to 115 total machine visits (5.23 machines/PCB on the average) in the optimal solution. In the case where the shortest makespan is obtained, the average number of visits is 4.27 (94 visits/22 PCBs) whereas the average is 3.77 (83 visits/22PCBs) in the best case. It should be noted here that although the heuristic generates a better solution than the min-max GAP solution in terms of visits, we do not know whether it provides the best result.

Results for randomly generated data

We also carried out 25 more experiments for randomly generated data. In this experimental design we kept the number of machines (eight), numbers of slots/machine (180), and number of feeder slots required per component (three, six, or nine) the same as in the industry data. The number of PCB types is drawn from a uniform distribution $U[15, 30]$. The total number of component types is also generated using a uniform distribution $U[200, 400]$. The number of component types inserted on a PCB type and number of units of a component type inserted on a PCB type are generated from uniform distributions $U[6, 20]$ and $U[1, 20]$, respectively. Note here that the total capacity of all

Table 3. Summary of results for randomly generated data

	IAI						IAII					
	SC 1		SC 2		SC 3		SC 1		SC 2		SC 3	
	Mspan	% Gap	Mspan	% Gap	Mspan	% Gap	Mspan	% Gap	Mspan	% Gap	Mspan	% Gap
Data set 1	556	76.51	339	7.62	549	74.29	333	4.72	353	11.01	401	26.10
Data set 2	1214	142.32	525	4.79	983	96.21	577	14.71	544	8.15	583	15.90
Data set 3	1298	130.14	605	7.27	1100	95.04	744	39.07	546	2.06	642	20.00
Data set 4	741	32.32	565	0.89	647	15.54	638	16.21	561	2.19	599	9.11
Data set 5	408	15.91	373	5.97	523	48.58	380	9.51	389	12.10	449	29.39
Data set 6	384	12.28	342	0.00	388	13.45	353	8.62	344	5.85	368	13.23
Data set 7	330	23.60	271	1.50	471	76.40	263	6.48	280	13.36	299	21.05
Data set 8	815	62.03	518	2.98	952	89.26	567	13.86	547	9.84	529	6.22
Data set 9	565	42.32	420	5.79	1013	155.16	459	16.20	434	9.87	425	7.59
Data set 10	717	41.70	506	0.00	1002	98.02	697	37.75	506	0.00	530	4.74
Data set 11	707	48.22	522	9.43	1151	141.30	580	19.83	553	14.26	801	65.50
Data set 12	904	144.32	403	8.92	558	50.81	534	44.32	399	7.84	408	10.27
Data set 13	947	75.05	596	10.17	1165	115.34	1030	89.34	573	5.33	651	19.67
Data set 14	411	21.24	355	4.72	650	91.74	402	25.23	342	6.54	420	30.84
Data set 15	760	50.79	504	0.00	801	58.93	565	12.10	504	0.00	816	61.90
Data set 16	471	65.26	287	0.70	639	124.21	414	43.75	316	9.72	347	20.49
Data set 17	986	80.59	580	6.23	966	76.92	694	27.11	586	7.33	588	7.69
Data set 18	791	74.61	485	7.06	718	58.50	627	37.80	491	7.91	533	17.14
Data set 19	295	28.26	240	4.35	346	50.43	271	41.15	210	9.38	258	34.38
Data set 20	832	26.25	659	0.00	971	47.34	785	11.51	704	0.00	715	1.56
Data set 21	1343	75.10	775	1.04	1110	44.72	931	22.82	758	0.00	778	2.64
Data set 22	735	57.05	498	6.41	1021	118.16	552	17.95	499	6.62	494	5.56
Data set 23	881	77.98	509	2.83	865	74.75	827	74.47	536	13.08	536	13.08
Data set 24	1081	96.90	556	1.28	1306	137.89	865	59.89	549	1.48	563	4.07
Data set 25	887	37.73	652	1.24	700	8.70	702	14.71	612	0.00	661	8.01
Average		61.54		4.05		78.47		28.36		6.56		18.25

the machines is still greater than the total feeder slots required for loading all component types. The results are illustrated in Table 3. Here, under each initialization algorithm the Gap is calculated using the lower bound of the best makespan (the lower bound is determined as the largest total processing time across all machines after the component families are loaded) achieved across three similarity coefficients, considering the overall range of number of families. These results indicate that in 23 out of 25 cases the best makespan is achieved using SC2. Furthermore, the makespan achieved using SC2 is only 1.8% and 3% worse than the best makespan for the remaining two cases. The average gap is 4.1% when IA I is used and 6.6% when IA II is used if SC2 is selected. These performances are substantially better than those of SC1 and SC3.

In sum, the results indicate that component grouping using SC2 with either IA I or IA II provides near-optimal solutions for both the environment studied and randomly generated data. The grouping and scheduling algorithms developed may be applied in other openshop environments where component mounting times do not depend on the location of the component feeders on the machine, no setup is needed when switching from one PCB to another, and machines are identical in terms of placement times and number of feeder slots required for each component. Such a setting may be encountered in small flexible manufacturing cells that consist of a few machines where the machines are linked to each other with low setup costs.

6 Implications and conclusions

In this paper, we have presented a hierarchical method for grouping components into families, loading these families of components on individual insertion machines, and finally, scheduling PCBs on the machines. Our approach is shown to provide near-optimal makespan solutions in a reasonable period of time. Further, when comparing it to a procedure where we simply load individual components to machines rather than loading component families on machines, we show that the creation of families results in fewer visits of PCBs to individual machines (leading to a reduction in materials handling costs).

We also show that using our approach, an estimate of the "best" number of families that should be formed based on scheduling measures rather than the structural measures can be obtained. This, in our opinion, is a much more effective approach for determining the number of families since it is based on actual shop performance measures.

From a practical perspective, there is no question that our methodology with its assumption of the openshop layout configuration would improve operational performance (since both makespan and mean flow time are considerably lower for this type of layout). Obviously, these improvements when translated into savings have to be traded-off against the "costs" of rearranging the current shop floor layout and also setting up a line linking each pair of machines. In our opinion, this is a much better form of shop floor configuration for PCB placement/insertion machines and it is our understanding that the organizational operations planning personnel will be evaluating the costs/benefits of such a reconfiguration.

Extensions of our study could focus on several aspects. First, more effective algorithms could possibly be developed for the component grouping. Second, it may be advantageous to allow lot splitting and lot streaming for technological reasons and for improved machine utilization. Third, component duplication may be considered to achieve better workload balance. Finally, although we have only experimented with one industry setting in this paper, the approach could also be extended/adapted for other similar manufacturing applications (machine tool manufacturing) characterized by high component variety and few final products.

Appendix 1: similarity measures used for component family formation

For each i and j, i, $j \in R$, and $i \neq j$, we defined three similarity coefficients (S_{ij}) as follows:

$$S_{ij}^1 = \frac{NCB_{ij}}{TNB_i + TNB_j - NCB_{ij}}$$

NCB_{ij} is the number of PCBs requiring both component types i and j, TNB_i is the number of PCBs requiring component type i and similarly TNB_j is the number of PCBs requiring component type j.

$$S_{ij}^2 = \sum_{b \in B} (UM_{ib} - UM_{jb})^2$$

UM_{ib} is the number of units of component i mounted on PCB b, UM_{jb} is the number of units of component j mounted on PCB b, and B is the set of PCBs requiring either component i or $j$or both. S_{ij}^2 is in fact a dissimilarity coefficient. In that case, the component family formation problem becomes a minimization problem.

$$S_{ij}^3 = \frac{\sum_{b \in B} \frac{\min\{UM_{ib}, UM_{jb}\}}{\max\{UM_{ib}, UM_{jb}\}}}{NCB_{ij}}$$

Appendix 2: details of stage I of the proposed methodology

Notation

P	Set of medians
F	Number of component families (may or may not be prespecified)
NF	Number of component families at each iteration
p_{ik}	kth most similar median to component type i, $k = 1, 2, 3, \ldots$
$CurrLoad(j)$	Current number of feeder slots required by all components assigned to median j

FS_i Feeder slots required by component type i

OBJ Objective function value

P_{Temp} Temporary set of medians

$SMS(P)$ Sum of the maximum similarities between all pairs of $i \in R$ and $j \in P$

$$= \begin{cases} \sum\limits_{i \in R} \max\limits_{j \in P} S_{ij} \; for P \subset R \\ i \neq j \\ -\infty \; for P = \emptyset \end{cases}$$

SS_{ik} Sum of similarities between component i assigned to median k and all other components assigned to the same median

$$= \sum_{\substack{j \in R \\ j \neq i}} S_{ij} x_{ik} x_{jk}, i \in R, k \in P, i \neq k$$

Initialization algorithms

Two heuristics are used to form the initial set of medians. The first is a greedy algorithm where, for a given set of medians, the component type that gives the greatest immediate increase in the value of the sum of the similarities is chosen as the next component to be included in the median set. If the total number of medians is prespecified, the algorithm will stop when this number is reached. Otherwise it will stop when the value of the sum of similarities fails to increase. In the second algorithm, the pair of component types that has the greatest similarity are merged so as to be considered as a single component type (it is in fact a component family), provided that the number of feeder slots required by both does not exceed the machine's capacity. We merge two components at a time and update the set of components and the similarity coefficients accordingly. We also select one of the components at the initial merger of two original components as the median and include it in the median set. The algorithm stops when the components can no longer be merged, i.e., if the similarity coefficient between two component types is zero, or when the number of medians, if prespecified, is reached.

Initial algorithm 1 (IA I):

Step 0: Let $P^0 = \emptyset, NF = 0, t = 1$.

Step 1: Let $j_t = \arg\max_{j \in R \setminus P^{t-1}} SMS\left(P^{t-1} \cup \{j\}\right)$.

Step 2: If $SMS(P^{t-1} \cup \{j\}) \leq SMS(P^{t-1})$, stop. P^{t-1} is an initial median set.
Otherwise, set $P^t = P^{t-1} \cup \{j_t\}$ and let $NF = NF + 1$.

Step 3: If $P^t = R$, or if $NF = F$ in case F is prespecified, stop. R is an initial median set. Otherwise, let $t = t + 1$, go to step 1.

Initial algorithm 2 (IA II):

Step 0: Let $P^0 = \emptyset, NF = |R|, t = 1$.

Step 1: Identify i_t and j_t such that $S_{ij} = \max_{i<j} \{S_{ij}\}$.

Step 2: If $S_{ij} = 0$, stop. P^{t-1} is an initial median set.
Otherwise, compute total number of feeders required by i_t and j_t: TF.

Step 3: If $TF > C$ let $S_{ij} = 0$, go to step 1.
Otherwise, go to step 4.

Step 4: If $NF = F$ in case F is prespecified, stop. P^{t-1} is an initial median set.
Otherwise go to step 5.

Step 5: Set $NF = NF - 1$; merge i_t and j_t to create a new temporary component type k_t. Let $P^t = P^{t-1} \cup \{j_t\}$. Let $t = t + 1$, go to step 1.

Grouping algorithm:

After an initial set of medians is determined component types are assigned to their most similar median as long as the machine slot capacity is not exceeded. The assignment procedure is as follows: for each component type, the absolute values of the differences between the similarity coefficients of the component and its first and second most similar medians are computed[3]. Then, component types are assigned to medians in non-increasing order of these absolute values, in an attempt to prevent the assignment of a component to a very dissimilar median.

When all of the assignments are completed for each component type in every family the sum of similarities between that component and other components assigned to the same median is computed. If a component type other than the selected median achieves the maximum sum of similarities then the set of medians is updated by adding that component type which improves the objective function value and removing the old median. In other words, the component which has been chosen as the median is replaced by another component which improves the total similarity measure. The method is repeated with the new set of F medians as long as the objective function value continues to improve with a new component type.

The details of the procedure to group components into families are as follows:

Step 0: Let $P^1 = P^t$ from the initial algorithm and set $t = 1$.

Step 1: For each component type i, sort median j's in non-increasing order of $S_{ij}, i \in R, j \in P$.

Step 2: Compute $DIF_i = |S_{i,p_{i1}} - S_{i,p_{i2}}|$. Sort DIF_i's in non-increasing order. Set $k = 1$, go to next step.

Step 3: If component type i is not assigned and $CurrLoad(p_{ik}) + FS_i \leq C$, assign i to median p_{ik}. Let $CurrLoad(p_{ik}) = CurrLoad(p_{ik}) + FS_i$, go to step 4. Otherwise, let $k = k + 1$, go to step 3.

Step 4: Set $P_{\text{Temp}} = \emptyset$. Compute SS_{ik}'s.

Step 5: Let $i* = \arg\max_{i \in R \setminus P} SS_{ik}$. If $i*$ is not a median, replace median k with component type $i*$: $i*$ becomes the new median. Set $P_{\text{Temp}} = P_{\text{Temp}} \cup \{i*\}$. Go to step 6.

[3] If the first choice of median for a particular component is not feasible due to the violation of slot capacity, the absolute value of the difference between the similarity coefficients of the component and its second and third most similar medians are computed. If the second, third, etc. choice of median is not feasible either, third, forth, etc. choice medians are considered.

Step 6: Compute OBJ. If $P_{\text{Temp}} = P^t$, stop. Otherwise, let $P^t = P_{\text{Temp}}$, $t = t + 1$, go to step 1.

Improvement algorithm:

Once the grouping procedure terminated, we consider all possible pair-wise component exchanges between the two sets of components assigned to two different medians to further improve the similarity between components. The algorithm is as follows:

Step 0: Let $i, j \in R$ and $k, l \in P$.

Step 1: If $(S_{ik} + S_{jl}) > (S_{il} + S_{jk})$ and $CurrLoad(k) + FS_i - FS_j \leq C$ and $CurrLoad(l) + FS_j - FS_i \leq C$, switch component types i and j between median k and median l.

Step 2: Update $CurrLoad(k)$ and $CurrLoad(l)$. If all components and medians are considered, stop. Otherwise, go to step 0.

References

1. Kear FW (1987) Printed circuit assembly manufacturing. Dekker, New York
2. McGinnis LF, Ammons JC, Carlyle M, Cranmer L, DePuy GW, Ellis KP, Tovey CA, Xu H (1992) Automated process planning for printed circuit card assembly. IIE Trans 24:8–30
3. Askin RG, Dror M, Vakharia AJ (1994) Printed circuit board family grouping and component allocation for a multimachine, open-shop assembly cell. Naval Res Logist 41:587–608
4. Balakrishnan A, Vanderbeck F (1999) A tactical planning model for mixed-model electronics assembly operations. Oper Res 47:395–409
5. Drezner Z, Nof S (1984) On optimizing bin packing and insertion plans for assembly robots. IIE Trans 16:262–270
6. Ball MO, Magazine MJ (1988) Sequencing of insertions in printed circuit board assemblies. Oper Res 36:192–201
7. Ahmadi J, Grotzinger S, Johnson D (1988) Component allocation and partitioning for a dual delivery placement machine. Oper Res 36:176–191
8. RL, Hamacher HW, Lee C-Y, Yeralan S (1989) On automating robotic assembly workplace planning. Research Report, Industrial and Systems Engineering Department, University of Florida, Gainesville, FL
9. Grotzinger S (1992) Feeder assignment models for concurrent placement machines. IIE Trans 24:31–46
10. Lofgren CB, McGinnis LF (1986) Soft configuration in automated insertion. Proc 1986 IEEE Conference on Robotics and Automation, San Francisco, CA, pp 138–142
11. Carmon TF, Maimon OZ, Dar-El EM (1989) Group set-up for printed circuit board assembly. Int J Prod Res 27:1795–1810
12. Maimon O, Shtub A (1991) Grouping methods for printed circuit board assembly. Int J Prod Res 29:1379–1390
13. Jain S, Johnson ME, Safai F (1996) Implementing setup optimization on the shop floor. Oper Res 43:843–851
14. Lofgren CB, McGinnis LF Tovey CA (1991) Routing printed circuit cards through an assembly cell. Oper Res 39:992–1104
15. Ammons JC, Lofgren CB, McGinnis LF (1985) A large scale machine loading problem in flexible assembly. Ann Oper Res 3:319–332
16. Fathi Y, Taheri J (1989) A mathematical model for loading the sequencers in a printed circuit pack manufacturing environment. Int J Prod Res 27:1305–1316
17. Rajan A, Segal M (1989) Assigning components to robotic workcells for electronic assembly. AT&T Tech J May–June:93-102
18. Crama Y, Kolen AWJ, Oerlemans AG, Spieksma FCR (1990) Throughput rate optimization in the automated assembly of printed circuit boards. Ann Oper Res 26:455–480
19. Brandeau ML, Billington A (1991) Design of manufacturing cells: operation assignment in printed circuit board manufacturing. J Intell Manuf 2:95–106
20. Ammons JC, Carlyle M, Cranmer L, DePuy G, Ellis K, McGinnis LF, Tovey CA, Xu H (1997) Component allocation to balance workloads in printed circuit card assembly systems. IIE Trans 29:265–275
21. Leipälä T, Nevalainen O (1989) Optimizition of the movements of a component placement machine. Eur J Oper Res 38:167–177
22. Günther HO, Gronalt M, Zeller R (1998) Job sequencing and component setup on a surface mount placement machine. Prod Plan Control 9:201–211
23. Hillier MS, Brandeau ML (2001) Cost minimization and workload balancing in printed circuit board assembly. IIE Trans 33:547–557
24. Leon VJ, Peters BA (1996) Replanning and analysis of partial setup strategies in printed circuit board assembly systems. Int J Flex Manuf Syst 8:389–412
25. Peters BA, Subramanian GS (1996) Analysis of partial setup strategies for solving the operational planning problem in parallel machine electronic assembly systems. Int J Prod Res 34:999–1021
26. Tang CS (1988) A max-min allocation problem: its solutions and applications. Oper Res 36:359–367
27. Dror M (1992) Openshop scheduling with machine dependent processing times. Discret Appl Math 3:197–205
28. Vakharia AJ, Çatay B (1997) Two machine openshop scheduling with machine-dependent processing times. Discret Appl Math 73:283–288
29. Anderberg MR (1973) Cluster analysis for applications. Academic, New York
30. Mulvey JM, Beck MP (1984) Solving capacitated clustering problems. Eur J Oper Res 18:339–348
31. Vakharia AJ, Wemmerlöv U (1995) A comparative investigation of hierarchical clustering techniques and dissimilarity measures applied to the cell formation problem. J Oper Manage 13:117–138