World Academy of Science, Engineering and Technology
International Journal of Industrial and Manufacturing Engineering
Vol:7, No:4, 2013

Genetic Algorithm Application in a Dynamic PCB Assembly with Carryover Sequence-Dependent Setups

M. T. Yazdani Sabouni and Rasaratnam Logendran

Open Science Index, Industrial and Manufacturing Engineering Vol:7, No:4, 2013 publications.waset.org/9222.pdf

Abstract**—**We consider a typical problem in the assembly of printed circuit boards (PCBs) in a two-machine flow shop system to simultaneously minimize the weighted sum of weighted tardiness and weighted flow time. The investigated problem is a group scheduling problem in which PCBs are assembled in groups and the interest is to find the best sequence of groups as well as the boards within each group to minimize the objective function value. The type of setup operation between any two board groups is characterized as carryover sequence-dependent setup time, which exactly matches with the real application of this problem. As a technical constraint, all of the boards must be kitted before the assembly operation starts (kitting operation) and by kitting staff. The main idea developed in this paper is to completely eliminate the role of kitting staff by assigning the task of kitting to the machine operator during the time he is idle which is referred to as integration of internal (machine) and external (kitting) setup times. Performing the kitting operation, which is a preparation process of the next set of boards while the other boards are currently being assembled, results in the boards to continuously enter the system or have dynamic arrival times. Consequently, a dynamic PCB assembly system is introduced for the first time in the assembly of PCBs, which also has characteristics similar to that of just-in-time manufacturing. The problem investigated is computationally very complex, meaning that finding the optimal solutions especially when the problem size gets larger is impossible. Thus, a heuristic based on Genetic Algorithm (GA) is employed. An example problem on the application of the GA developed is demonstrated and also numerical results of applying the GA on solving several instances are provided.

Keywords**—**Genetic algorithm, Dynamic PCB assembly, Carryover sequence-dependent setup times, Multi-objective.

I. Introduction

THE main part of every electronic device is PCBs which embodies several different types of electronic components. PCB assembly simply means to place these components which range in very different types and shapes, during a process called assembly operation. Because of the vast variety of components, the boards requiring similar types of components are usually grouped into one group of boards to lessen the number of unnecessary setups which are required whenever the assembly operation is scheduled to transfer from one group to another, and it implies the existence of group scheduling (GS) concepts for this problem. The setup time assumption in PCB studies is the most important factor that essentially affects the performance of the system. Because of the inherent complexity involved in evaluating the setup time in the assembly of PCBs, the setup time has not been exactly or correctly estimated in all of the traditional studies so far. However, the carryover sequence-dependent setup time assumed in this paper accurately captures the actual amount of setup time required to transfer the assembly operation from one board group to another. In other words, the setup time dependency traditionally considered is that it is at most dependent on the immediately preceding board group, while a careful investigation of this problem has shown that the dependency is not only on the *immediately preceding* board group, but on all of the previously scheduled board groups and their sequences. Thus, finding the best sequence of the groups which are already assigned to the machine to evaluate the minimum amount of setup time to transfer to the next board group is translated into solving a Travelling Salesman Problem which is known to be NP-hard in strong sense. The complexity gets even worse when this setup operation is integrated with another operation, namely kitting (as described below) on two-machine flow shop. The objective is to sequence all board groups and their boards where the weighted sum of total weighted flow time and total weighted tardiness is minimized.

A PCB assembly machine utilizes several feeders on which components required by boards are placed. Thus, changing the components which are currently on the feeders with the components which are required on appropriate feeders by the next board group, on a frequent basis, results in continuously updating the component-feeder requirements. These changeovers begin with the reference group which is the set of components remained on the feeders from the previous period and get carried over up to the current group, which implies that the dependency is carried over throughout all board groups and introduces the idea of developing carryover sequence-dependent setup times.

There is another kind of operation called kitting required in preparing the PCBs for the assembly operation and that must be performed by kitting staff before the assembly operation starts. The kitting operation is an essential part of PCB manufacturing whose impact has always been disregarded from the total required setup time in the assembly of PCBs by assuming that all boards have static or zero arrival times.

M.T. Yazdani Sabouni is with the School of Mechanical, Industrial and Manufacturing Engineering, Oregon State University, OR 97331-2600 USA (phone: 1-541-207-6721; e-mail: yazdanim@onid.orst.edu).

R. Logendran is with the School of Mechanical, Industrial and Manufacturing Engineering, Oregon State University, OR 97331-2600 USA (phone: 1-541-737-5239; fax: 1-541-737-2600; e-mail: Logen.Logendran@oregonstate.edu).

Open Science Index, Industrial and Manufacturing Engineering Vol:7, No:4, 2013 publications.waset.org/9222.pdf

Thus, as a new direction of research, we are inspired to investigate the possibility of integrating both types of setups, which is referred to as integrating internal setup (carryover sequence-dependent setup operation) and external setup (kitting operation), and have both of them solely be performed by the machine operator. To do that, the machine operator is assigned to perform the kitting operation required by the next board group during the time he is idle or during the time the machine is automatically performing the assembly operation of the current board group. In this research, boards are simultaneously kitted or introduced to the manufacturing line at the same time the assembly operation is being performed and this enables us to focus on a dynamic PCB assembly system for the first time. The problem investigated in this research captures all of those operational constraints that can be found in a typical PCB assembly problem. Capturing all of the operational constraints introduces much complexity to the problem which has been known to be complex even with more simplified assumptions. This necessitates the development of meta-heuristics which is a GA-based approach in this paper. The GA developed is capable of solving problems of all sizes (small to large) considered very effectively and in a very short period of time which substantiates the industrial applicability of the research as well as the methodology developed.

There are numerous studies in PCB manufacturing, where each addresses different characteristics associated with the problem. However, the most challenging factor differentiating not only the problem types, but also the corresponding methods given, is the type of setup time. Thus, setup time is addressed to provide an overview of some of the studies that have been performed so far. As mentioned before, traditionally it is assumed that board groups are totally sequence-dependent or the dependency is at most on the immediately preceding board group. While the sequence-independent cases can be referred to [1] and [2], [3]–[6] studied sequence dependency. A setup strategy called the decompose and sequence (DAS) method is proposed in [7], which changes the feeders not shared by the next board group while ignoring the board groups and boards themselves. Reference [8] proposed another method called KTNS (Keep Tool Needed the Soonest) which only exchanges the required tools and neglects the positions to which the tools are assigned.

Another setup strategy called the group setup strategy ensures the fact that boards are assembled in groups making it more suitable to be applied in the assembly of PCBs than other approaches in classical studies. However, since it assumes that the dependency is only on the immediately preceding scheduled board group, it may not correctly evaluate the overall dependencies associated with the problem. References [9] and [10] proposed methods to minimize the total makespan while focusing on group setup strategy. A mathematical model and a GA to minimize makespan on non-identical parallel machines with sequence-dependent setups are developed [11]. In the work of [12] two strategies to minimize the number of feeder-changeovers with sequence-dependent board groups are proposed.

The only research in scheduling that considered the carryover sequence-dependent setup time is the work of [13], while focusing on a single objective problem and jobs with static arrival times. Like in all of the traditional works, their research also did not take into consideration of the kitting operation in the evaluation of the total setup time by assuming that all boards have static arrival times. Consequently, it can be inferred that minimizing a bi criteria objective function, integration of internal and external setups, and dynamic arrival of boards are mainly the factors of considerable significance that basically differentiate their works and ours.

In this problem, the objective function evaluation on a two-machine flow shop, when there are two opposing objectives and dynamic arrival of boards, is a challenging task unlike most of the typical scheduling problems with static arrival times of boards. However, we have developed an approach which is capable of optimally evaluating the objective function on the first machine for a given sequence of boards and groups and heuristically on the second machine, yet very close to the optimal value. The highlighted contribution of the paper is summarized as following:

1. Introducing a PCB assembly problem with dynamic arrival times of boards while taking advantage of the idea of just-in-time manufacturing in the assembly of PCBs.
2. Correctly incorporating the carryover sequence-dependent setup times in a dynamic PCB assembly.
3. Eliminating the kitting staff by integration of internal and external setups.
4. Developing an objective function evaluation approach to optimally evaluate the objective function on the first machine and heuristically on the second machine.
5. Developing an efficient genetic algorithm to arrange board groups as well as the boards within each group.

The explanation of objective function evaluation is provided in the next section. Section III outlines an overview of the GA developed in this paper. Data generation and a representative example on the application of the GA are provided in Section IV. In Section V, computational experiments after solving several problem instances are provided, followed by the last section in which an overview of the paper together with future research are given.

II. Objective Function Evaluation

Minimizing the weighted sum of weighted flow times and total weighted tardiness is the objective in this paper. Suppose there are a total of m board groups, and each has several boards. We imply $G_{[j]}$, $b_{[j,i]}$ and $n_{[j]}$ as the jth ordered group, ith ordered board within $G_{[j]}$ and number of boards in $G_{[j]}$, respectively. The flow time and tardiness of $b_{[j,i]}$ are evaluated respectively as $(O_{[j,i]k} - R_{[j,i]k})$ and $(O_{[j,i]2} - d_{[j,i]})$ where $O_{[j,i]k}$, $R_{[j,i]k}$, $w_{[j,i]}$ and $d_{[j,i]}$ are respectively the completion time, finish kitting time, weight and due date of $b_{[j,i]}$ in $G_{[j]}$ on the kth machine. Consequently the objective function is evaluated as

Open Science Index, Industrial and Manufacturing Engineering Vol:7, No:4, 2013 publications.waset.org/9222.pdf

$$\sum_{j=1}^{m}\sum_{i=1}^{n_{[j]}}\sum_{k=1}^{2}\beta_1 w_{[j,i]}(O_{[j,i]k}-R_{[j,i]k})+\beta_2 w_{[j,i]}(O_{[j,i]2}-d_{[j,i]})$$

where $\beta_1+\beta_2=1$. $n_{[j]}$ in the objective function is not defined to be a fixed value and it is possible to do so because this objective function will be algorithmically calculated.

During the time in between the assembly of the current board group starts and the machine operator performs the setup operation required of the next board group, the machine operator is idle and he can perform the kitting operation of the next board group. The difference between the finish kitting time and completion time of any board is reflected in the objective function, and as a result, finish kitting times should be correctly evaluated. On the other hand, because of dynamic arrival times of boards, start kitting times are hard to be calculated so evaluating the finish kitting times would be difficult too, and it becomes more difficult when they have to be determined on the two machines. This issue is addressed in this paper by developing an approach, including two properties, which is capable of calculating the objective function optimally on the first machine, and heuristically on the second machine. The proof of optimality on the first machine (not addressed here because of space limitations) guarantees evaluating the optimal values of finish assembly and kiting times.

Property 1: Suppose the start time and completion time of the assembly operation of the board group $G_{[j]}$ on the first machine be represented, respectively, as $T_{[j]}$ and $C_{[j]1}$. Assume the next board group containing n boards assembled after $G_{[j]}$ is $G_{[j+1]}$. If the kitting operation of all the boards of $G_{[j+1]}$ completes before $C_{[j]1}-T_{[j]}$ then the sequence of the n boards whose last board's finish kitting time is exactly $C_{[j]1}$ dominates the similar sequences of the n boards whose last board's finish kitting time is earlier than $C_{[j]1}$.

Property 2: Suppose the start time and completion time of the assembly operation of the board group $G_{[j]}$ on the first machine be represented respectively as $T_{[j]}$ and $C_{[j]1}$. Assume the next board group containing n boards assembled after $G_{[j]}$ is $G_{[j+1]}$. If the kitting operation of all the boards of $G_{[j+1]}$ completes after $C_{[j]1}-T_{[j]}$ then the sequence of n boards whose first board's start kitting time is exactly $T_{[j]}$ dominates similar sequences of the n boards whose first board's start kitting time is after $T_{[j]}$.

Start (finish) kitting times on the second machine are calculated very differently from the first machine. To determine the start kitting time of a board group on the second machine, it is required to identify the start kitting time of the first board of this group which is dependent on both its completion time on the first machine and also the completion time of the last board of the previous group on the second machine. To tackle this problem, two cases are considered. The first case reflects the situation where the completion time of the first board of a group on the first machine is smaller than the completion time of the previous board group plus the setup time to transfer to the current board group on the second machine. As the second case, the completion time of the first board of a group on the first machine is larger than the completion time of the previous board group plus the setup time to transfer to the current board group on the second machine.

When the first case is present, the kitting operation of the next board group on the second machine is at a time during the assembly time of the current board group (similar to the situations in Properties 1 and 2). In the second case, two possibilities ($Poss_1$ and $Poss_2$) are assumed to evaluate the start assembly time of the first board of the current group on the second machine and eventually they are developed to heuristically calculate the start times of the assembly operation of the rest of the boards of the current group on the second machine. In other words, a general approach can be developed as a form of an algorithm to heuristically evaluate the start kitting times on the second machine. As $Poss_1$ shown in Fig. 1, the assembly operation of a board on the second machine immediately starts after its completion time on the first machine. As $Poss_2$ shown in Fig. 2, the assembly operation of a board on the second machine is at the time when the next board(s) finishes its assembly operation on the first machine.

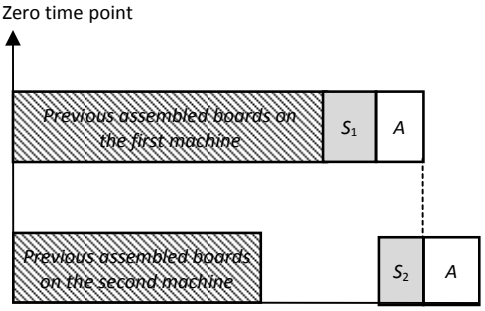

Fig. 1 Assembly operation of the first board on the second machine starts right after this board is completed on the first machine
S_1 and S_2 are respectively the setup times needed to transfer to the next board group on the first and the second machine

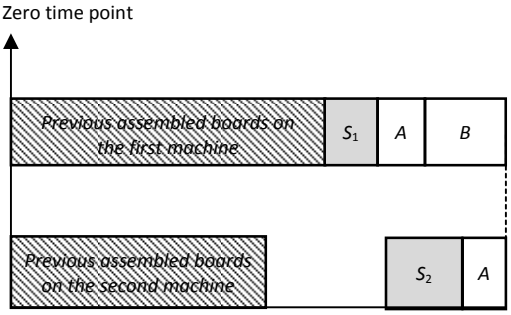

Fig. 2 Assembly operation of the first board on the second machine finishes exactly at the completion time of the next board on the first machine

Assume $G_{[c]}$ with n_c number of boards, the algorithm below finds the start kitting times of the boards of $G_{[c]}$. *MinO* and *BestKit* are respectively the best found contribution of $G_{[c]}$ in the objective function on the second machine and an array for storing the best found start kitting times on the second machine. *PossT* is a decision variable implying the selection

Open Science Index, Industrial and Manufacturing Engineering Vol:7, No:4, 2013 publications.waset.org/9222.pdf

of $Poss_1$ or $Poss_2$. i, r, l and NB are positive and integer variables.

1. Set $PossT = 1$ and $MinO$ to a very large and positive value.
2. For $i = 1$ to n_c
3. For $r = i$ to n_c
4. If $PossT = 1$, then let the start assembly time of $b_{[c,i]}$ be exactly after the completion time of $b_{[c,r]}$ on the first machine. If $i < n_c$, $i < r$ and $PossT = 2$ then start the assembly of $b_{[c,i]}$ at a time such that it finishes at the completion time of $b_{[c,r]}$ on the previous machine. Set $NB = i$. Go to step 3.
5. For $l = i - 1$ to 1
6. Assemble $b_{[c,l]}$ immediately before $b_{[c,NB]}$. Set $NB = l$ and get next l. If $l = 0$, go to step 4.
7. Put the setup operation exactly before the assembly of $b_{[c,1]}$ and let the setup follow the kitting operation of $b_{[c,nc]}$. Set $NB = n_c$. Go to step 5.
8. For $l = n_c - 1$ to 1
9. Kit $b_{[c,l]}$ exactly before $b_{[c,NB]}$. Set $NB = l$, and get next l. If $l = 0$, go to step 6.
10. For $l = i + 1$ to n_c
11. Calculate the start assembly time of $b_{[c,l]}$ according to the time progress of the assembly operation on the first and second machines (the maximum of completion time of $b_{[c,l]}$ on the first machine and completion time of $b_{[c,l-1]}$ on the second machine), get next l. If $l = n_c + 1$, go to step 7.
12. Calculate the contribution of $G_{[c]}$ in the objective function on the second machine with the current start kitting times. Update $MinO$ if a less value for the contribution of $G_{[c]}$ in the objective function is found and then update $BestKit$ to the current start kitting times. Get next r. If $r = n_c + 1$, get next i. If $i = n_c + 1$, set $PossT = 2$ and repeat step 2; otherwise, if $i = n_c + 1$ and $PossT = 2$, go to step 8.
13. Report $BestKit$ and terminate the algorithm.

III. Genetic Algorithm Implementation

Genetic algorithms (GA) originally developed by [14] belong to the class of optimization techniques. GA iteratively improves an objective function through several iterations of the search by allowing a set of solutions (representatives of different areas in the solution space) in each iteration to be combined with each other, resulting in the generation of other new solutions. Because of the interest in improving the objective function value, only the best of parents as well as offspring generated are qualified to be transferred to the next iteration and the rest are disregarded. A solution is typically encoded in forms of numbers or character (called genes), depending upon the type of the solution space. Accordingly, different solutions are generated by changing the positions of the genes in sequence while satisfying the feasibility conditions.

A. Algorithm Structure and Implementation

In an iteration of the GA implemented in this research, there are *pop* number of solutions or individuals which are initially randomly generated. At each iteration, (*crossover rate*) × *pop* (rounded up to the first integer and even value) numbers of individuals are randomly chosen and they are randomly assigned to the pairs of two (only two parents are allowed for each mate) to select the parents that can be mated with each other. *Crossover rate* is a number between 0 and 1 implying the ratio of the individuals in the population that can be mated. Selected parents are mated using crossover operator and generate new children. New children generated are added at the end of the current population and this population is sorted based on the increasing value of the objective function. Then, the first *pop* solutions are selected and among them (*mutation rate*) × *pop* (rounded up to the first integer value) individuals are randomly selected for the mutation, where *mutation rate* is a number between 0 and 1 implying the rate of individuals that can be mutated. The children generated by mutation operator are added to the end of the current population and, again, the population is sorted and the first *pop* individuals are chosen and are transferred as the population starting the next iteration. The best solution identified at each iteration called "elite" deserves to survive in the population of the next iteration after implementing the crossover and mutation operators in the next iteration and if its objective function value is better than that of the worst solution identified in the next iteration. The process explained happens at an iteration of the search and is repeated until a maximum allowable number of iterations is reached or number of iterations without improvement reaches to a pre-determined value. The search starts by applying the algorithm to identify the best sequence of the boards within each group by assigning a group immediately after the reference group and examining different objective function values of having different board sequences within this group and there is no group scheduled after this group. The best sequence of the boards identified for each group will be kept and it remains fixed when the search is advanced to identifying the best sequence of the groups themselves. Based upon the observation in [15] and reported by [16], *pop* is assumed to be 25, *crossover rate* is 0.6, *mutation rate* is 0.05, maximum number of iterations allowed is 500 and number of iterations without improvement is 75.

B. Crossover

Non-Wrapping Order Crossover (NWOX) operator proposed by [17] is a successful crossover that has the characteristics of position-based and relative order-based crossovers when the interest is not only in the absolute position of the elements in the sequence, but also is in their relative positions (for more information refer to [17]). The functionality of this crossover is also compared against five other different types: Uniform Crossover Operator (UXO), Partially-Mapped Crossover (PMX), Uniform Partially-Mapped Crossover (UPMX), CX and OX in [18], and it showed promising results.

Without the loss of generality, suppose the search is in group level. Thus there are chromosomes of N genes where N is the number of board groups. Using this crossover, two chromosomes are mated to generate two children. It randomly selects two points (p_1 and p_2 where $p_1 \leq p_2$) among the N

positions. Suppose that the gene in position i of parent j is represented by $g_j(i)$ where $i = 1,\ldots,N$ and $j = 1, 2$ (see Table I where $p_1 = 2$ and $p_2 = 4$). First, two children C_1 and C_2 are initialized with copies of the first and second parent (the second column in Table I). Then, the locations of $g_2(p_1)$, $g_2(p_1+1),\ldots, g_2(p_2)$ are identified in C_1 and are replaced with empty slots. Similarly, the location of $g_1(p_1)$, $g_1(p_1+1),\ldots, g_1(p_2)$ are replaced with empty slots in C_2 (the third column in Table I). Then, the empty slots are moved to the $p_1,\ldots, p_2$ part, resulting in all empty slots to be located in this part of each child (the fourth column). Finally, $p_1,\ldots, p_2$ part of C_1 is filled with $g_2(p_1)$, $g_2(p_1+1),\ldots, g_2(p_2)$ respectively and similarly this part is filled with $g_1(p_1)$, $g_1(p_1+1),\ldots, g_1(p_2)$ in C_2 (the last column).

TABLE I
APPLICATION OF CROSSOVER OPERATOR

Sol No.	Parent chromosomes	Replacing the groups with empty slots	Moving the empty slots to p_1 and p_2 part	Replacing empty slots with groups in p_1 and p_2 part
1	G_5-G_3-G_2-G_1-G_4-G_6	○-○-G_2-G_1-○-G_6	G_2-○-○-○-G_1-G_6	G_2-G_5-G_3-G_4-G_1-G_6
2	G_2-G_5-G_3-G_4-G_6-G_1	○-G_5-○-G_4-G_6-○	G_5-○-○-○-G_4-G_6	G_5-G_3-G_2-G_1-G_4-G_6

C. Mutation

This operator selects two points at random in a chromosome and exchanges their associated genes. For example, in the first parent in Table I, when the groups in the first and the fifth positions exchange the result would be G_4-G_3-G_2-G_1-G_5-G_6.

IV. DATA GENERATION

The process used to generate the data and an example problem used to demonstrate the application of GA is provided in this section. The same approach as given in [13] is used to obtain the number of boards within each group, number of units of each board to produce, run times (assembly times) of boards and component-feeder assignments. Run time of a board is random number from the interval U[5, 20]. There are the total of 125 different types of components that must be located on boards, out of which 75 components are assembled on the first machine and 50 components on the second machine. Number of components in each board is selected from U[5, 12] and U[1, 5], respectively, for the first and the second machine. The weight of a board in the objective function is an integer number generated from the uniform interval [1, 3]. Since most of the producers are customer oriented, a greater importance is assigned to the tardiness factor in the objective function by assuming that β_1 and β_2 are respectively 0.4 and 0.6.

The number of boards in a group is determined by generating a random number p from U[0, 1]. The number of boards is 3, 4 or 5 when $p \leq 1/3$, $1/3 < p \leq 2/3$ and $p > 2/3$, respectively. The number of groups that can have the same number of boards is $\lceil N/3 \rceil$ where N is the number of groups. The feeder capacities on the first and second machine are assumed to be 20 and 10, respectively. Also, the number of units of each board is randomly generated from U[3, 15]. To generate the kitting times, the study by [19] is used to obtain the ratio of run time to kitting time evaluated as 45/12 = 3.75. A mechanism given in [20] is used to generate meaningful due dates. Two intervals $[\bar{d} - R\bar{d}, \bar{d}]$ and $[\bar{d}, \bar{d} + (C_{max} - \bar{d})R]$ are randomly selected using a deciding random number from [0, 1]. The first interval is selected if this random number falls in $[0, \tau]$; otherwise the second interval is chosen. Notations τ, R, $\bar{d}$ and C_{max} are respectively tardiness factor, range factor, average due dates and estimated maximum completion time of the last board on the second machine. Loose due dates and narrow range of due dates are generated by respectively selecting τ and R to be small. τ and R are deliberately assumed to be 0.8 and 0.2 to test the GA on tight and narrow range of due dates. To calculate C_{max}, approximate values of boards' completion times on the machines as well as an estimate of carryover over sequence dependent setup times for each of the groups are required. Approximate completion times of the boards on the machines are calculated using the dynamic programming algorithm below, by assuming that $\bar{S}_{gk}$ is the average carryover sequence-dependent setup time for the gth board group to be assigned in different positions on the kth machine. $O_{g,b,k}$ is the completion time of board b of group g on machine k.

$$O_{g,1,k} = \max\left\{O_{g\text{-}1,\, n_{g\text{-}1},k} + \beta_{gk} \times \bar{S}_{gk},\, O_{g,1,k\text{-}1}\right\} + rt_{g,1,k} \quad (1)$$

$\forall g;\ k = 1, 2;\ b = 1$

$$O_{g,b,k} = \max\left\{O_{g,b,k\text{-}1},\, O_{g,b\text{-}1,k}\right\} + rt_{g,b,k} \quad (2)$$

$\forall g;\ k = 1, 2;\ b > 1$

where $\bar{S}_{gk} = \frac{\sum_{\pi_j \in \Pi(j)} \sum_{j=1}^{N-1} S_{\pi_j,g,k}}{N \times q}$, $O_{g,b,0} = 0$, $O_{g,1,0} = 0$ and $O_{0,\, n_{g\text{-}1,k}} = 0$.

$\Pi(j)$ is a set of $q > 0$ random group sequences in which board group g is assigned to the jth slot and $S_{\pi_j,g,k}$ is the setup time required to transfer to the board group g after the partial sequence π_j on machine k. To obtain $\bar{S}_{gk}$, a number of random sequences of groups are generated and in each of the sequences the carryover sequence-dependent setup time for assigning the gth group in the jth slot ($S_{\pi_j,g,k}$) is evaluated. Then, all of the obtained setup time values for all slots in all random sequences are added up. This value is divided by the sum of the total number of partial sequences that can be assumed. There is a linear relationship between β_{gk} and CV_{gk} and it is used to obtain β_{gk}. $CV_{gk} = s/\bar{S}_{gk}$ where s and $\bar{S}_{gk}$ are respectively the standard deviation and average values of the carryover sequence-dependent setup times for assigning group g on machine k in different positions. $\bar{d}$ is also determined

Open Science Index, Industrial and Manufacturing Engineering Vol:7, No:4, 2013 publications.waset.org/9222.pdf

using the formula $\tau = 1 - \bar{d} / C_{max}$. Using the process outlined above to generate the data, an example problem with number of board groups of six is generated and its associated information is provided in Tables II and III. To provide a simple example for better understanding, the number of boards in a group is assumed to be at most 2. However, it is 3, 4 or 5 in the instance problems generated in the computational experiments section. Table II presents number of boards in each group, run times and kitting times of the boards on the two machines, weights and also due dates of the boards. Table III displays the feeder-component assignments required of the six board groups as well as the reference board group on the two machines. The setup time per feeder is the amount of time needed to change from a component required by the current group to a component required of the next board group on the same feeder and, based on the research in [13], it is assumed to be 180 and 200 seconds, respectively, on the first and the second machine.

TABLE II
RUN TIMES, KITTING TIMES, WEIGHTS AND DUE DATES OF BOARDS

Group	Board	Run time		Kitting time		Weights	Due date
		M1	M2	M1	M2		
1	1	1121	100	299	27	1	2925
	2	1474	3	393	1	2	2740
2	1	457	250	122	67	1	3348
	2	604	113	161	30	2	3194
3	1	61	141	16	38	3	2920
	2	787	325	210	87	3	2755
4	1	854	9	228	2	2	3304
5	1	1257	128	335	34	2	3285
6	1	423	291	113	78	2	2736

TABLE III
COMPONENTS ASSIGNED ON FEEDERS

Feeder	*M*1							*M*2						
	R	*G*1	*G*2	*G*3	*G*4	*G*5	*G*6	*R*	*G*1	*G*2	*G*3	*G*4	*G*5	*G*6
1		C-36			C-56			C-14	C-10	C-12				C-29
2		C-37		C-19	C-42	C-12		C-13	C-14	C-11				C-06
3	C-16	C-23		C-74		C-29	C-04			C-49	C-20			C-22
4	C-42	C-19	C-31						C-33	C-16			C-50	C-13
5	C-10	C-28	C-55	C-68	C-23				C-10		C-35			C-17
6	C-67	C-62		C-44	C-15	C-29	C-15		C-48		C-39	C-16	C-20	
7		C-19	C-42					C-34	C-15	C-10				
8		C-10	C-58			C-52								
9	C-47	C-65		C-40	C-48								C-12	
10			C-48		C-11		C-73						C-31	C-30
11	C-67				C-42									
12	C-52		C-09											
13		C-72	C-15	C-07	C-21									
14	C-43		C-50			C-34								
15		C-40				C-16	C-32							
16	C-38	C-61	C-58	C-16										
17	C-17	C-56		C-65			C-61							
18	C-19	C-06		C-23	C-29									
19	C-37	C-13	C-71	C-54		C-66								
20	C-37				C-15	C-07	C-41							

V. APPLICATION OF THE GENETIC ALGORITHM

In this section, the application of the GA on an example problem generated in the previous section is presented. Since GA is an iterative-greedy search, meaning that the same operations are employed in each iteration of the search, the application of different operations used in this algorithm is demonstrated in only one iteration of the search. Also, as mentioned before, the algorithm first identifies the best sequence of the boards within the groups and using those sequences the search continues to find the best sequence of the groups themselves. Implementation of GA only makes sense when chromosomes have at least 4 genes; otherwise, there would be a very small solution space. On the other hand, since the GA structure in this paper is similar in both the group and board levels, the highest number of boards in a group in this example was assumed to be 2 to enable focusing *only* on applying GA to the group level. Out of the six considered board groups, the last three groups have only one board implying that there is no need of GA in the board level for these groups. However, the first three groups have two boards where the best board sequence can be obtained by examining only two possible sequences which again satisfy the need of employing GA in the board level. The best sequence identified for the boards inside G_1, G_2 and G_3 are respectively b_{12}-b_{11}, b_{22}-b_{21} and b_{31}-b_{32} where G_j implies the jth group and b_{ji}-b_{jk} implies the kth board follows the ith board within G_j. In Table IV, the population of 10 individuals for the 6 board groups problem is presented. This population can be initially randomly generated or be the population received from the previous iteration. Section "Current population" of this table shows 10 individuals that can be mated to generate a number of children. The second column represents different solutions with different configurations. The parents that can be mated

Open Science Index, Industrial and Manufacturing Engineering Vol:7, No:4, 2013 publications.waset.org/9222.pdf

with each other are shown in the third column with the same number of "X" (for example, P_5 with three "X" can only be mated with P_6 with three "X"). In the fourth column, two crossover points for each of the chromosomes in a mating-pair are given. The last column shows the associated objective function value for each of the solutions. The selected parents ((P_8, P_9), (P_3, P_7) and (P_5, P_6)) are mated using the crossover operator resulting in 6 (0.6*10) new solutions which are given in the "Crossover application" section of Table IV. The third column of this section displays the parents that have been selected to produce the corresponding generated child (for example, P_8 and P_9 are parents of P_{11} and P_{12}). So far, a total of 16 solutions (P1 to P16) are generated, resulting in an extended population with 16 individuals. Then, this population is sorted based on increasing OFV (objective function value) and the first 10 individuals are selected and stored in the section "Sorted population". The third column of this section shows ($0.01*10 \approx 1$) individuals which are randomly selected for mutation and their associated mutation points are provided in the fourth column. Accordingly, the chromosome P_{12} is selected for mutation generating P_{17} which is presented in section "Mutation application" of the table. Finally, a total of 11 individuals, shown in the last two sections of the table, are sorted and the first top 10 are transferred as the population for the next iteration. The best chromosome or P_4 with the OFV of 85190 is the elite chromosome identified in this iteration that can survive in the next iteration (after the crossover and mutation operators are performed) if it had an OFV better than that of the worst solution identified at the end of the next iteration.

TABLE IV
DEMONSTRATION OF GENETIC ALGORITHM'S OPERATORS

Current population

Solution	Configuration	Indvs for crossover	Crossover points	OFV
P_1	G_4-G_1-G_5-G_3-G_6-G_2			126048
P_2	G_3-G_6-G_1-G_5-G_4-G_2			95577
P_3	G_4-G_2-G_5-G_6-G_3-G_1	XX	2, 4	109296
P_4	G_3-G_4-G_6-G_2-G_5-G_1			85190
P_5	G_3-G_1-G_5-G_2-G_4-G_6	XXX	1, 3	99449
P_6	G_5-G_1-G_4-G_3-G_6-G_2	XXX	1, 3	126987
P_7	G_5-G_4-G_1-G_2-G_6-G_3	XX	2, 4	133904
P_8	G_6-G_3-G_5-G_1-G_2-G_4	X	2, 5	92657
P_9	G_1-G_5-G_4-G_3-G_6-G_2	X	2, 5	132192
P_{10}	G_3-G_1-G_6-G_5-G_2-G_4			96268

Sorted population

Solution	Configuration	Indvs for mutation	Mutation points	OFV
P_4	G_3-G_4-G_6-G_2-G_5-G_1			85190
P_8	G_6-G_3-G_5-G_1-G_2-G_4			92657
P_2	G_3-G_6-G_1-G_5-G_4-G_2			95577
P_{10}	G_3-G_1-G_6-G_5-G_2-G_4			96268
P_5	G_3-G_1-G_5-G_2-G_4-G_6			99449
P_{15}	G_3-G_1-G_5-G_4-G_6-G_2			99742
P_{12}	G_4-G_3-G_5-G_1-G_2-G_6	*	3, 6	104449
P_3	G_4-G_2-G_5-G_6-G_3-G_1			109296
P_{13}	G_4-G_2-G_5-G_6-G_1-G_3			125044
P_1	G_4-G_1-G_5-G_3-G_6-G_2			126048

Crossover application

Solution	Configuration	Associated parents	OFV
P_{11}	G_1-G_5-G_4-G_3-G_6-G_2	(P_8,P_9)	132192
P_{12}	G_4-G_3-G_5-G_1-G_2-G_6	(P_8,P_9)	104449
P_{13}	G_4-G_2-G_5-G_6-G_1-G_3	(P_3,P_7)	125044
P_{14}	G_5-G_4-G_1-G_2-G_6-G_3	(P_3,P_7)	133904
P_{15}	G_3-G_1-G_5-G_4-G_6-G_2	(P_5,P_6)	99742
P_{16}	G_5-G_1-G_4-G_3-G_2-G_6	(P_5,P_6)	127715

Mutation application

Solution	Configuration	Associated parent	OFV
P_{17}	G_4-G_3-G_6-G_1-G_2-G_5	P_{12}	100009

VI. COMPUTATIONAL RESULTS

To evaluate the performance of the GA, it was tested on solving several problem instances with number of groups of 3, 4, 5 and 6 with the average setup time per feeder of 30 and 180. For each of these problems differentiated by the number of groups and average setup time per feeder, two instance problem instances were randomly generated. A general mathematical model for this problem is presented in [21], and by taking advantage of that model, the associated mathematical model for each of the considered problem instances was formulated and optimally solved using ILOG CPLEX [22]. CPLEX found the optimal solution for all of the considered problems except the problems with number of groups of 6. For these two problems, CPLEX failed to find the optimal solution even after spending the maximum time limit of 3 days. Consequently, for these problems only the best objective value and the best lower bound found by CPLEX were reported. The results after implementing GA on the problems and comparing against the optimal values are provided in Table V. The first and second columns of this table respectively show the average setup time per feeder and number of groups. The objective function values found by CPLEX and GA are provided in the third and fourth columns together with their percentage deviation ($\frac{\text{GA}-\text{CPLEX}}{\text{CPLEX}} \times 100\%$), which is given in the fifth column. The rest of the columns show computation times (in seconds) and number of enumerated solutions for both the approaches. As it can be observed, GA found the optimal solution in most of the cases; however, the largest and the average deviation are respectively, 1.39% and 0.35%, which are considerably small. The average computation time and number of enumerated solutions for CPLEX and GA are respectively 38595 s, 0.81 s and 1.71×10^8, 3642 which substantiates the fact that the performance of GA is both effective and very efficient in terms of solution quality and negligible computation times required to solve all of the problems.

VII. CONCLUSION

A bicriteria scheduling problem in the assembly of PCBs in a two-machine flow shop system is considered in this paper.

Open Science Index, Industrial and Manufacturing Engineering Vol:7, No:4, 2013 publications.waset.org/9222.pdf

The type of setup time assumed in this study called carryover sequence-dependent exactly complies with the real application of this problem and assumes the dependency to be carried through the entire sequence of board groups. The kitting operation, which has been traditionally ignored in PCB studies, is brought to light by simultaneously performing the kitting operation and the assembly operation, which is characterized as "integration of external and internal setups". Consequently, the boards' ready times which have always been assumed to be static become dynamic in this paper, which also enables applying the concepts of just-in-time manufacturing. Objective function evaluation is a difficult task in this research because of the requirement of evaluating the start (finish) kitting times and completion times on both machines. Thus, an approach capable of evaluating the objective function optimally on the first machine and heuristically on the second machine is developed. In attempting to embrace all of the operational constraints in the real application, this problem has become extremely complex to be solved by optimal solution finding mechanisms. As a result, a meta-heuristic based on genetic algorithm is developed and its application is demonstrated by solving an example problem. Also, to test the effectiveness of the algorithm, several problem instances were solved and the GA showed very promising results. As future research, a lower-bounding mechanism would be developed for providing a benchmark for solving large-size problems in which CPLEX fails to provide the optimal solution.

TABLE V
RESULTS OBTAINED FOR CPLEX AND GA BY SOLVING PROBLEM INSTANCES

AST	*N*	Solution value			CPU time		Solution number	
		CPLEX	GA	Deviation (%)	CPLEX	GA	CPLEX	GA
180	3	77315	77315	0	0.1	0.54	615	3169
	3	95476	95476	0	1.9	0.69	9316	3457
	4	210276	210276	0	31	0.75	1.02×10^{6}	3469
	4	231581	231581	0	239	0.79	7.38×10^{6}	3697
	5	366306	366514	0.06	3841	0.97	6.14×10^{7}	3745
	5	309019	309019	0	2735	0.96	7.18×10^{7}	3835
	6	530861* 507954**	530977		259200	0.96	1.01×10^{9}	3734
30	3	39338	39338	0	0.61	0.69	847	3265
	3	42152	42152	0	1.6	0.71	815	3265
	4	154035	154581	0.35	102	0.81	2.69×10^{6}	3505
	4	141702	144267	1.81	65	0.83	1.92×10^{6}	3697
	5	189901	191090	0.63	14881	0.97	2.36×10^{8}	4032
	5	137828	139743	1.39	78	0.82	1.82×10^{6}	3704
	6	376471* 358971**	377424		259200	1.16	1.00×10^{9}	4425
Average				0.35	38598	0.81	1.71×10^{8}	3642

The numbers represented by '*' and '**'respectively imply the best objective value and the best lower bound found by CPLEX

ACKNOWLEDGMENT

This research is funded in part by the National Science Foundation (USA) Grant No. CMMI-1029471. Their support is gratefully acknowledged.

REFERENCES

[1] J.E. Schaller, "A new lower bound for the flow shop group scheduling problem," *Computers and Industrial Engineering*, vol. 41, pp. 151–161, 2001.

[2] S.W. Choi, and Y.D. Kim, "Minimizing total tardiness on a two-machine re-entrant flowshop," *European Journal of Operational Research*, vol. 199, pp. 375–384, 2009.

[3] V.A. Strusevic, "Group technology approach to the open shop scheduling problem with batch setup times," *Operations Research Letters*, vol. 26, pp. 181–192, 2000.

[4] D.H. Eom, H.J. Shin, I.H. Kwun, J.K. Shim, and S.S. Kim, "Scheduling jobs on parallel machines with sequence-dependent family setup times," *International Journal of Advanced Manufacturing Technology*, vol. 19, pp. 926–932, 2002.

[5] J.E. Schaller, J.N.D. Gupta, and A.J. Vakharia, "Scheduling a flowline manufacturing cell with sequence dependent family setup times," *European Journal of Operational Research*, vol. 125, pp. 324–339, 2000.

[6] Y.D. Kim, H.G. Lim, and M.W. Park, "Search heuristics for a flowshop scheduling problem in a printed circuit board assembly process," *European Journal of Operational Research*, vol. 91, pp. 124–143, 1996.

[7] L.F. McGinnis, J.C. Ammons, M. Carlyle, L. Cranmer, G.W. Depuy, K.P. Ellis, C.A. Tovey, and H. Xu, "Automated process planning for printed circuit card assembly," *IIE Transactions*, vol. 24, pp. 18–30, 1992.

[8] C.S. Tang, and E.V. Denardo, "Models arising from a flexible manufacturing machine, part 1: Minimization of the number of tool switches," *Operations Research*, vol. 36, pp. 767–777, 1988.

[9] I.O. Yilmaz, and H.O. Günther, "A group setup strategy for PCB assembly on a single automated placement machine," *Operations Research Proceedings*, Bremen, 2005, pp.143–148.

[10] V.J. Leon, and I.J. Jeong, "An improved group setup strategy for PCB assembly," *International Conference on Computational Science and its Applications*, Singapore, 2005, pp. 312–321.

[11] N. Van Hop, and N.N. Nagarur, "The scheduling problem of PCBs for multiple non-identical parallel machines," *European Journal of Operational Research*, vol. 158, pp. 577–594, 2004.

[12] J. Ashayeri, and W. Selen, " A planning and scheduling model for onsertion in printed circuit board assembly," *European Journal of Operational Research*, vol. 183, pp. 909–925, 2007.

[13] C.A. Gelogullari, and R. Logendran, "Group-scheduling problems in electronics manufacturing," *Journal of Scheduling*, vol. 13, pp. 177–202, 2010.

[14] J.A. Holland, *Adaptation in natural and artificial systems*, University of Michigan, Ann Arbor, 1975.

[15] K.A. De Jong, *Analysis of the behavior of a class of genetic adaptive systems*, Doctoral Dissertation, University of Michigan, USA, 1975.

[16] D.E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, Massachusetts: Wesley, 1989.

[17] V.A. Cicirello, "Non-wrapping order crossover: An order preserving crossover operator that respects absolute position," *Proceedings of Genetic and Evolutionary Computation Conference*, *GECCO*, USA, 2006, pp. 1125–1131.

Open Science Index, Industrial and Manufacturing Engineering Vol:7, No:4, 2013 publications.waset.org/9222.pdf

[18] O. Abdoun, and J. Abouchabaka, "A comparative study of adaptive crossover operators for genetic algorithms to resolve the traveling salesman problem," *International Journal of Computer Applications*, vol. 31, pp. 49–57, 2011.
[19] S.S. Joshi, Phadnis, K. Srihari, and R. Seeniraj, "Use of simulation to improve the kitting process at an EMS provider's facility," *Computers and Industrial Engineering Conference*, Singapore, 2002.
[20] V. Pandya, and R. Logendran, "Weighted tardniess minimization in flexible flow shops," *Proceedings (CD-ROM)*, *19th Annual Industrial Engineering Research Conference*, Cancun, Mexico, 2010.
[21] M.T. Yazdani Sabouni, and R. Logendran, "Bicriteria carryover sequence-dependent group scheduling in PCB manufacturing," *Proceedings (CD-ROM), 20th Annual Industrial Engineering Research Conference (IERC)*, Reno, NV, USA, 2011.
[22] ILOG CPLEX. IBM, Version 12.2, 2010.

Open Science Index, Industrial and Manufacturing Engineering Vol:7, No:4, 2013 publications.waset.org/9222.pdf