Doğuş Üniversitesi Dergisi, 6 (1) 2005, 67-78

WORKLOAD BALANCING IN PRINTED CIRCUIT BOARD ASSEMBLY SHOPS

BASKI DEVRE KARTI DİZGİ ATÖLYELERINDE HAT DENGELEME

Ekrem DUMAN
Dogus University, Department of Industrial Engineering

M. Bayram YILDIRIM
Wichita State University, Industrial and Manuf. Eng. Department

ABSTRACT : In assembling printed circuit boards (PCB), the use of numerically or computer controlled electronic component placement machines has become quite popular in the last decades. However, serious operations research problems arise through their use such as, allocation of component types to machines, board production schedule, feeder configuration and placement sequencing. In this study, the problem of allocation of component types to machines is taken up where two non-identical machines are deployed serially on a line to complete the assembly process of PCBs. For the solution of this problem three heuristic algorithms are suggested and their performances are investigated on experimental data.

Keywords: Heuristics, Printed Circuit Board Assembly, Load Balancing

ÖZET *: Son yıllarda baskı devre kartlarının (BDK) dizgisinde nümerik veya bilgisayar kontrollü elektronik dizgi makinalarının kullanımı yaygın hale gelmiştir. Ancak, bu beraberinde komponent tiplerinin makinalara atanması, kart üretim çizelgelemesi, besleyici düzeni ve dizgi sırası gibi karmaşık yöneylem araştırması problemlerini getirmiştir. Bu çalışmada, birbirinden farklı iki makinanın aynı hatta olması durumu için komponentlerin makinalara atanması problemi ele alınmıştır. Bu problemin çözümü için üç ayrı sezgisel algoritma geliştirilmiş ve performansları örnek veriler üzerinde incelenmiştir.*

Anahtar Kelimeler: *Sezgisel yöntemler, Baskılı devre kartı dizgisi, Hat dengeleme*

1. Introduction

Numerically or computer controlled electronic component placement machines have been extensively used in assembling printed circuit boards (PCB) during the last decades. As compared to manual assembly of PCBs, automated placement machines have brought major gains in productivity and efficiency through their fast, error free and reliable component placement operations. However, serious planning and scheduling problems such as, allocation of component types to machines, determination of board production sequence, allocation of component types to feeder cells (feeder configuration) and determination of component placement sequence have arisen in their use.

All of these problems are interdependent, i.e., the solution of any problem affects the solution of the others. Such interdependency is more evident between the first two and last two problems. Thus, all four problems should be considered and solved simultaneously if an overall optimal solution is desired. However, since each of these problems is quite complex by itself, trying to build and solve a monolithic model is quite difficult and intractable. Hence, in this study, they are taken as

separate problems and iterative solution methods are suggested to cope for the interaction between them (Duman, 1998).

In this study, the first one of the four major problem classes, identified with regard to the automated assembly of PCBs, is taken up. Note that, component allocation problem can be classified as machine load balancing problem in the broad sense.

The amount of research made on PCB assembly problems is quite extensive. However, most of these researches are related to the feeder configuration and placement sequencing problems and the component allocation problem has received less attention. Furthermore, the researches made on component allocation problem mostly assumed that the machines deployed serially on a line are identical. Below some of the works that are mainly related to the component allocation problem are briefly surveyed.

A general overview of PCB assembly problems is given by McGinnis et. al. (1992), and a more recent one is due to Ji and Wan (2001). Francis and Horak (1994) considered the problem of choosing the numbers of reels of each type of components to be used in populating a printed circuit board, by a surface mount technology (SMT) machine. The objective is to maximize the length of an uninterrupted machine production run, while using no more slots for reels than are available. Carmon et al. (1989) aimed at minimizing the total setup time in changing the feeder configuration and propose a different production method, called the group set-up (GSU) method (grouping similar boards), which can significantly reduce set-up times. Askin et al. (1994) addressed the problem of minimizing the makespan for assembling a batch of boards with a secondary objective of reducing the mean flow time. Ben-Arieh and Dror (1990) studied the problem of assigning component types to insertion machines with the aim of balancing the workload assigned.

As the sequence of boards to be produced on a single machine and the allocation of different component reels to feeder carriage are considered together, one might adapt an iterative approach. Sadiq et. al. (1993) proposed such an iterative approach with the aim of minimizing the total production time for a group of PCB assembly jobs on a single machine. Ahmadi et. al. (1988) considered a placement machine, which features two fixtures for the delivery of components to the placement heads. They investigated the case where all components are accessible and the pick sequence is static. Crama et. al. (1990) proposed a heuristic hierarchical approach to the problem of optimizing the throughput rate of a line of several component placement machines with three placement heads, all devoted to the assembly of a single type of PCB. Given a line of placement machines and a family of boards, Klomp et.al. (2000) developed a heuristic algorithm which focused on the feeder rack assignment problem. Hillier and Brandeau (2001) developed an efficient algorithm to balance the workload among the semi-automatic placement machines and the manual assembly stations which is called the Cost Minimizing Workload Balancing Heuristic. Duman (1998) considered the distribution of workload to two placement machines deployed serially on a line. He developed and compared 28 construction algorithms and the best performing algorithms are further improved by pair-wise exchanges.

In the next section, the description of the problem environment and the problem formulation are given. The solution algorithms suggested are explained in section

three. The experimentation runs and the results obtained are discussed in section four. Finally, in section five, a short summary and the major conclusions arrived in this study are provided.

2. Problem Definition

The setting of the load balancing (component allocation) problem arising in automated PCB assembly shops shows large variability. The main reasons of this variability are due to the variances in machine architecture (type), the differences in the characteristics of the production processes and various engineering preferences. One may say without loss of generality that, the number of different load balancing problem formulations can be as large as the number of PCB assembly facilities. The implications of different machine types, production characteristics and engineering preferences on the load balancing problem are discussed in detail in Duman (1998).

2.1. Description of the Case

The machine type considered is one with component pickup device (leading to a trivial feeder configuration problem), stationary placement head and moving carrier board, which is the technology that is used by most placement machine manufacturers today (in fact, the discussions made here are valid also for many other machine types). The basic operations of such kind of a machine are described below.

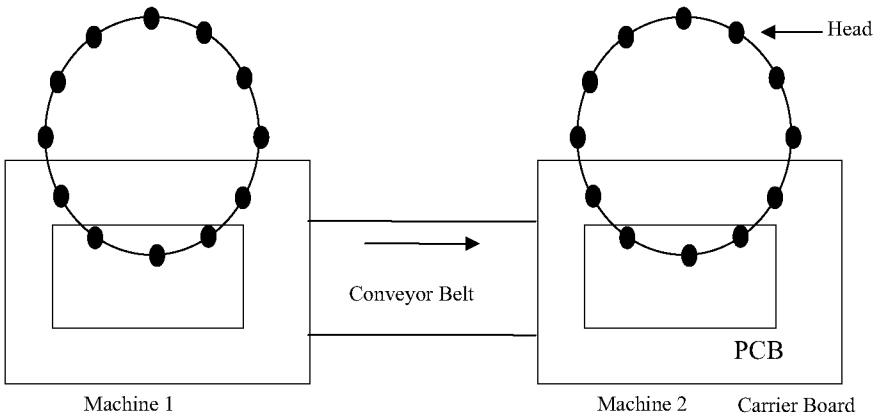

Figure 1. Two placement machines on a line.

Circular shaped rotating component pickup device takes the role of sequencer machine. The pickup device, which has 60-120 heads, picks up the components to its heads in the placement order, from the component tapes, which are placed along the perimeter of the device and performs each placement just after the desired precise placement location is aligned beneath the head currently over the carrier board. The placement sequencing problem turns out to be a Chebyshev TSP (Duman, 1998; Duman and Or, 2004) and the layout of the component tapes can be formulated as a simple allocation problem (Duman, 1998).

The boards are populated by two machines sequentially. There is a conveyor belt between the machines, which carries the partially completed boards from machine 1 to machine 2 (see figure 1). For this assembly environment case, the following assumptions are undertaken regarding the load balancing problem:

A1. Machines are not identical: they may have different speeds and different number of feeder slots.
A2. Component types are identical with respect to their slot requirements in the feeder area and all are handled with the same nozzle.
A3. Assembly of a new board type cannot start unless both machines are cleared by the currently assembled board type.
A4. There is no sequence dependent setup time, when switching between different board types.
A5. The total number of component types is equal to the total number of feeder locations on two machines.
A6. Total number of component types to be populated on any PCB type is larger than the feeder capacity of either machine, so that each board type requires both machines to be fully assembled.
A7. The demand for PCB types are known and fixed for the planning period under consideration.
A8. The placement of each component takes time directly proportional to the speed of the machine making the placement.
A9. The production environment is high-mix, low-volume.
A10. The setup times incurred in changing a component type in the feeder are very high.
A11. Buffer of partially completed boards is not desired due to the engineering preferences.

The justifications and/or the limitations of these assumptions are investigated in detail in Duman (1998). Thus, the reader may refer to that study for a discussion on this issue.

2.2. Problem Formulation

When the setup time to change a component in the feeder is very high, then it is not desirable to make any changes in the feeder configuration during the whole planning horizon (a strong assumption 10). In this case, the objective becomes to distribute the component types to the two machines so that, the workload among the machines has a good balance regarding each particular board type.

The notation given below will be used in the formulation of the problem:

i : component type index (i=1,..,n)
j : board type index (j=1,..,m)
a_j : number of boards of type j to be produced
P_{ij} : number of components of type i to be placed on board type j
F_i : feeder capacity of machine i (i=1, 2)
s_i : speed of machine i (i=1, 2)
X_i : {1 if component type i is assigned to machine 1, 0 otherwise}
N_j : total number of components to be placed on board type j ($= \sum_i a_j P_{ij}$)

Now, the problem can be formulated similar to Ben-Arieh and Dror (1990) and Duman (1998) as follows:

$$\text{Min} \quad \sum_{j=1}^{m} a_j \left| \sum_{i=1}^{n} X_i P_{ij} / s_1 - \sum_{i=1}^{n} (1 - X_i) P_{ij} / s_2 \right| \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{m} a_j \left(\sum_{i=1}^{n} X_i P_{ij} / s_1 - \sum_{i=1}^{n} (1 - X_i) P_{ij} / s_2 \right) \geq 0 \tag{2}$$

$$\sum_{i=1}^{n} X_i \leq F_1 \tag{3}$$

$$\sum_{i=1}^{n} (1 - X_i) \leq F_2 \tag{4}$$

$$X_i = 0 \text{ or } 1 \quad i=1,..n$$

In the above formulation, the objective function (1) shows the sum of the machine workload imbalances resulting from the assembly of each particular board type. Constraint (2) says that the workload assigned to machine 1 is greater than or equal to the workload assigned to machine 2. This is to ensure that less work-in-process inventory is accumulated between the machines. On the other hand, constraints (3) and (4) are just the feeder capacity constraints.

This problem is shown to be NP-Complete in Duman (1998). Thus, one needs to look for good performing heuristic algorithms to cope with this problem. The heuristic algorithms suggested in this study are described in the next section.

3. Solution Procedures Developed

The algorithms considered in the investigation of the non-identical case problems are the best performing algorithms for the identical machines case (CUgr, BUgr) (Duman, 1998), and the CUtd algorithm (which is specifically developed for non-identical case). It was felt that, there is no need to test the others in Duman (1998), since structurally these are quite representative for all.

The detailed description of CUgr and BUgr algorithms are available in Duman (1998), however they are shortly reviewed below for convenience.

The CUgr and BUgr algorithms have two mechanisms in their structure: component sort and component assignment. Component sort is the rule, which determines the order by which component types are assigned to the machines. On the other hand, component assignment is the rule of deciding to which machine a given component type is to be assigned. The component sort rules of CUgr and BUgr algorithms are *component usage* (CU) and *board-component usage* (BU) respectively and the component assignment rule is *greedy optimization* (gr) for both. The definitions of these rules are given below:

CU (component usage): $CU_i = \sum_{j=1}^{m} a_j P_{ij}$. Under this rule, total number of placements of each component type over all boards to be produced, are calculated and component types are sorted in non-increasing order of these values.

BU (board-component usage): This is a two step component sorting rule; first, board types are ordered according to the number of components to be populated on them. Board types requiring more components (higher Nj) are more important and come earlier in the list. First board type in the list is picked up and the component types that exist on that board are ordered in non-increasing order of their usage on that board. Then, the second board type is picked and the component types that exist on that board but not yet included in the ordered list before, are added at the end of the list with the same logic. This procedure continues until the ordered list contains all component types.

GR (greedy optimization): This rule seeks a partial load balance at each step. Each time a new component type is picked up from the component sort list, the value of the objective function for the partial feeder configurations attained so far is calculated for both possible machine assignments, and the machine corresponding to a lower objective value (imbalance) is chosen for assignment.

During the application of the assignment rule, in case of a tie, machine 1 is chosen for the assignment (this helps the satisfaction of constraint (2) given in the problem formulation). This rule is valid as long as there are empty feeder locations on both machines. If the feeder of one machine is filled up, then the remaining component types are necessarily assigned to the other machine.

For the CUtd another component assignment rule specifically developed for non-identical machines case is used. This is the *TD (top down)* component assignment rule and is explained below (machine one is assumed to be faster and has more feeder slots than machine two):

TD rule requires the introduction of the "filling speed" concept. Analytically, filling speeds of the machines (fs_1, fs_2, etc.) are equal to the quotient of the division of feeder sizes (F_1, F_2, etc.) by the highest common factor of them. For example, if F_1=20 and F_2=10, then the highest common factor = 10, and fs_1 = quotient (20/10) = 2 and fs_2 = quotient (10/10) = 1. For identical machines, both filling speeds are equal to unity. Another new concept introduced is the so called dlpf (desired load per feeder). To calculate dlpf, the total number of component placements is expressed as a pair, where each term represents the number of components desired to be placed on each machine respectively. These terms are calculated proportional to the speed of machines. Then the desired component placement number of a machine is divided by the feeder size and the dlpf for that machine is obtained. As an example, if F_1=20, F_2=10, s_1 (speed of machine 1) = 4, s_2 (speed of machine 2) = 1 and the total number of component placements = 1500, then (1500*4/5, 1500*1/5) are the number of components desired to be placed on the two machines. Accordingly, 1200/20=60 and 300/10=30 are the $dlpf_1$ and $dlpf_2$ values respectively. Note that, in this example, fs_1=2 and fs_2=1. In the case of identical two machines, $dlpf_1$ equals $dlpf_2$.

After the filling speeds and the dlpf's are calculated, the *td* sort rule proceeds as follows: Given a component sort list (CU is preferred), fs_1 component types from the top and fs_2 component types from the bottom of the list are taken and assigned to machines 1 and 2 respectively. Then, first the fs_1 and then the fs_2 component types already assigned to machines are considered and the imbalance resulting from these assignments is tried to be balanced by adding another fs_1 and fs_2 component types to two machines gradually (one after the other). Prior to the addition of a new component type to either one of the machines, the desired partial load (DPL) for the slots already filled up is calculated according to;

$DPL_i = dlpf_i*[(\text{number of slots already filled})_i + 1]$

where the addition of 1 is to take into account the new slot to be filled. If we define APL_i (actual partial load for machine i) as the sum of the component usages for the component types already assigned to machine i, then the aim is to minimize the difference between the actual and desired partial loads (DPL_i-APL_i). To accomplish this task, a component type among the unassigned ones is chosen and assigned to machine i. After this, another set of fs_1 and fs_2 component types is assigned to the machines and any resulting imbalance is minimized as explained above. This procedure continues until all component types are assigned to a machine.

The logic behind this algorithm is the dlpf concept. Although in the optimal solution the resulting load per feeder values may not be equal to the dlpf values used in this algorithm (60 and 30 for the above example), intuitively, a component allocation plan which attains the dlpf values should not be far from the optimum. The strength of this algorithm is the equal filling rate of machines if their feeder sizes are not the same (i.e., at all times, the machines are being filled up with equal ratios, so there is little risk of being obliged to assign the last component types to an undesirable machine since the desirable one is already filled up).

The description of the CUtd algorithm given above implicitly assumes that the highest common factor of the feeder capacities of the machines is a large number (at least five). In other words, the ratio of the filling speeds is assumed to be a round decimal number (e.g. 1.25, 1.50, 2.00). If this is not the case (e.g. feeder capacities are prime numbers), it may be required that the whole or a great portion of the feeder mechanisms be filled up at the very first iteration of the algorithm. This obviously is not a desired situation according to the philosophy of CUtd. In such cases, one may prefer to round off the ratio of filling speeds to a round decimal number at the cost of being obliged to assign several last components to the undesired machine (like in the case of CUgr and BUgr algorithms).

In the next section, test problems are generated and solved using these algorithms.

4. Experimental Runs and Results Obtained

Test problems are randomly generated with various number of component types (n=30, 60, 90, 120) and two different number of board types (m=10, 20) are generated and solved. The speed of machine 1 (s_1) is taken as four times the speed of machine 2 (s_2), while feeder capacity of machine 1 (F_1) is taken as twice the feeder capacity of machine 2 (F_2). The results for m=10 and m=20 are displayed in tables 1 and 2 respectively.

In these tables, the following convention is used to represent the randomly generated test problems:

PnmAi

where;
P : denotes the word "problem" and has no other special meaning
n : number of component types (the problem size)
m : number of board types
A : stands for the homogeneous structure of the placement matrix (explained below)
i : the problem index (e.g. 1,2,..) if there is more than one problem with the same parameters

Another parameter not shown in the above representation is the number of boards to be produced of each particular board type. These are generated from a uniform distribution between 1 and 10.

For each (n,m) combination the number of problems generated is six. Since it is difficult to judge the performances of the algorithms just by looking at the objective function values, the percent deviations of the algorithms from the best solution are calculated and are also tabulated in tables 1 and 2. As an example, if for a particular problem, a result of 97 is obtained by algorithm X and the best result among all algorithms is 78, then the percent deviation of algorithm X from the best solution is calculated as (97-78)/78 = 0.24. Average deviation values (Aver. all) for all problems are listed at the bottom of these tables.

The placement matrix (P) of test problems has a homogeneous structure. To determine the value of each particular p_{ij} element, a uniform random number between 1 and 100 is generated and a value is assigned to p_{ij} according to the following rule:

Range	p_{ij}	Assumed Percentage
1-40	0	40
41-55	1	15
56-70	2	15
71-78	3	8
79-85	4	7
86-91	5	6
92-96	6	5
97-100	7	4

The idea here is to reflect a common real life problem characteristic, where it is usual that most component types are placed on boards in small quantities (1 or 2), while few are placed in larger numbers.

Table 1. Algorithm results for 10 board types

	Total imbalance values					Deviation from best solution			
	CUgr	**BUgr**	**CUtd**	**RAN**	**MIN**	**CUgr**	**BUgr**	**CUtd**	**RAN**
P3010A1	97	78	172	142	78	0.24	0.00	1.21	0.82
P3010A2	115	149	121	224	115	0.00	0.30	0.05	0.95
P3010A3	138	155	203	270	138	0.00	0.12	0.47	0.96
P3010A4	88	100	46	175	46	0.91	1.17	0.00	2.80
P3010A5	104	127	103	115	103	0.01	0.23	0.00	0.12
P3010A6	124	91	123	311	91	0.36	0.00	0.35	2.42
Aver. (n=30)	111	117	128	206	95	0.25	0.30	0.35	1.34
P6010A1	224	205	133	511	133	0.68	0.54	0.00	2.84
P6010A2	213	250	155	414	155	0.37	0.61	0.00	1.67
P6010A3	272	324	253	705	253	0.08	0.28	0.00	1.79
P6010A4	41	38	175	172	38	0.08	0.00	3.61	3.53
P6010A5	172	178	213	297	172	0.00	0.03	0.24	0.73
P6010A6	148	141	193	419	141	0.05	0.00	0.37	1.97
Aver. (n=60)	178	189	187	420	149	0.21	0.24	0.70	2.09
P9010A1	306	324	221	481	221	0.38	0.47	0.00	1.18
P9010A2	289	305	216	614	216	0.34	0.41	0.00	1.84
P9010A3	349	390	293	727	293	0.19	0.33	0.00	1.48
P9010A4	210	250	182	484	182	0.15	0.37	0.00	1.66
P9010A5	328	352	240	723	240	0.37	0.47	0.00	2.01
P9010A6	395	406	334	667	334	0.18	0.22	0.00	1.00
Aver. (n=90)	313	338	248	616	248	0.27	0.38	0.00	1.53
P12010A1	568	659	344	1049	344	0.65	0.92	0.00	2.05
P12010A2	490	520	262	1025	262	0.87	0.98	0.00	2.91
P12010A3	529	537	343	983	343	0.54	0.57	0.00	1.87
P12010A4	377	431	346	671	346	0.09	0.25	0.00	0.94
P12010A5	547	633	331	1129	331	0.65	0.91	0.00	2.41
P12010A6	525	541	442	956	442	0.19	0.22	0.00	1.16
Aver. (n=120)	506	554	345	969	345	0.50	0.64	0.00	1.89
Aver. (all)	**277**	**299**	**227**	**553**	**209**	**0.31**	**0.39**	**0.26**	**1.71**

It can be seen in tables 1 and 2 that, the CUtd algorithm performs the best. Out of the 24 problems, CUtd found the best result 17 and 22 times for m=10 and m=20 cases respectively. The second best performing algorithm is CUgr, which was the best performing one in the identical machines case (Duman, 1998). The superiority of CUtd over the others becomes more evident as the problem gets more complicated (higher n and m).

The superiority of the CUtd algorithm against the CUgr and BUgr algorithms may be due to two reasons: First, it fills the feeders proportional to their capacities and secondly, it tries to allocate the component types with smaller usage numbers to the slower machine. On the other hand, the deficiency of the CUgr and BUgr algorithms is that, they fill up the feeders proportional to machine speeds, not to feeder capacities.

Table 2. Algorithm results for 20 board types

	Total imbalance values					Deviation from best solution			
	CUgr	**BUgr**	**CUtd**	**RAN**	**MIN**	**CUgr**	**BUgr**	**CUtd**	**RAN**
P3020A1	291	283	206	567	206	0.41	0.37	0.00	1.75
P3020A2	351	400	333	691	333	0.05	0.20	0.00	1.08
P3020A3	356	445	256	436	256	0.39	0.74	0.00	0.70
P3020A4	289	319	228	608	228	0.27	0.40	0.00	1.67
P3020A5	259	338	261	448	259	0.00	0.31	0.01	0.73
P3020A6	244	299	231	352	231	0.06	0.29	0.00	0.52
Aver. (n=30)	298	347	253	517	252	0.20	0.39	0.00	1.08
P6020A1	604	665	433	750	433	0.39	0.54	0.00	0.73
P6020A2	685	732	444	1019	444	0.54	0.65	0.00	1.30
P6020A3	529	540	464	981	464	0.14	0.16	0.00	1.11
P6020A4	86	142	395	216	86	0.00	0.65	3.59	1.51
P6020A5	791	896	506	1048	506	0.56	0.77	0.00	1.07
P6020A6	696	640	495	1181	495	0.41	0.29	0.00	1.39
Aver. (n=60)	565	603	456	866	405	0.34	0.51	0.60	1.19
P9020A1	744	831	648	1239	648	0.15	0.28	0.00	0.91
P9020A2	805	839	547	1389	547	0.47	0.53	0.00	1.54
P9020A3	666	702	387	949	387	0.72	0.81	0.00	1.45
P9020A4	695	678	366	958	366	0.90	0.85	0.00	1.62
P9020A5	666	779	459	923	459	0.45	0.70	0.00	1.01
P9020A6	719	755	424	1111	424	0.70	0.78	0.00	1.62
Aver. (n=90)	716	764	472	1095	472	0.56	0.66	0.00	1.36
P12020A1	966	1021	514	1438	514	0.88	0.99	0.00	1.80
P12020A2	880	1021	441	1117	441	1.00	1.32	0.00	1.53
P12020A3	954	1029	676	1481	676	0.41	0.52	0.00	1.19
P12020A4	1046	1119	620	1631	620	0.69	0.80	0.00	1.63
P12020A5	1220	1335	839	2007	839	0.45	0.59	0.00	1.39
P12020A6	1212	1190	629	2112	629	0.93	0.89	0.00	2.36
Aver. (n=120)	1046	1119	620	1631	620	0.73	0.85	0.00	1.65
Aver. (all)	**656**	**708**	**450**	**1027**	**437**	**0.46**	**0.60**	**0.15**	**1.32**

In tables 1 and 2, one may notice that, as the number of component types (n) increases, the total imbalance value seems to increase. Regarding the best results obtained by the algorithms (the MIN column in tables 1 and 2), linear regression models are built using the SPSS statistical package and the sample coefficients of determination (r^2) turn out to be 0.77 and 0.58 for m=10 and m=20 cases respectively. These values can be regarded as sufficient to accept a linear relationship between the total imbalance value and the number of component types.

To see the benefits gained by using the algorithms, the generated problems are solved by assigning the component types to machines in a random fashion (the RAN column in tables 1 and 2). It is seen that, the use of the algorithms brings about a 60 per cent reduction in the total imbalance value.

Regarding the performance of the CUtd algorithm (or, also of the other algorithms) it may be useful to look at the ratio of the imbalance values obtained to the total production time (TPT) to complete the assembly of all boards. The total production time can be calculated as;

$$TPT = n*E[P_{ij}]*m*E[a_j]/5 \quad (5)$$

where, $E[P_{ij}]$ is the average number of placements of a component type on a board type (calculated to be 1.85 for the test problems generated) and $E[a_j]$ is the average number of boards to be produced of a particular type and it is calculated to be 5.5 which is the expected value of Uniform (1,10). The division by the constant factor "5" is assembly time of any of the machines under the assumption that the total workload is perfectly distributed (recall that in the test problems machine 1 is four times faster than machine 2 and when one out of five components is allocated to machine 2, the production time will be one fifth of the total number of components). The ratios of the imbalance value obtained by CUtd algorithm to the total production time are tabulated in table 3 (the average of the imbalance values for the six problems having the same number of component types is taken into account).

Table 3. Ratio of CUtd results to total production time (%)

	m=10			m=20		
n	**CUtd**	**TPT**	**ratio**	**CUtd**	**TPT**	**ratio**
30	128	610	21.0	253	1221	20.7
60	187	1221	15.3	456	2442	18.7
90	248	1832	13.5	472	3663	12.9
120	345	2442	14.1	620	4884	12.7
average			16.0			16.2

The ratio values given in table 3 may somewhat be regarded as loose upper bounds and there may be still room for improvement. In this sense, one may be unsatisfied or unsure about the performance of the CUtd algorithm and may look for better performing algorithms. Although, this may be a proper future study area, the next step should be to solve the placement sequencing problem (which was omitted in this study) for the component allocation plan generated by the CUtd algorithm. Then the workload imbalances could be improved by exchanging some component types assigned to machines. In this approach, these two problems (component allocation and placement sequencing) should be solved iteratively (one after the other) until a satisfactory result is obtained.

5. Concluding Remarks

In this study, the problem of allocating component types to machines where two non-identical placement machines are deployed for the assembly of PCBs is considered. This study can be regarded as an extension of Duman (1998), where the focus was on two identical machines case. Two best performing algorithms of Duman (1998) are selected and applied to the non-identical case and a new algorithm CUtd is developed as a special solution of non-identical case. The performances of these algorithms are tested on randomly generated test data and the CUtd algorithm is found to be the best.

A possible future study could be to extend the ideas presented here for the existence of non-identical component types where each may require different number of slots in the feeder mechanism (relaxation of assumption two). Although it may rarely be faced in real PCB assembly shops, another possible future study area could be the consideration of more than two non-identical machines deployed on a line. Finally, a major future study could be the handling of component allocation and placement sequencing problems together.

References

AHMADI, J., GROTZINGER, S. & JOHNSON, D. (1988) Component allocation and partitioning for a dual delivery placement machine. *Operations Research*, 36/2, pp. 176-191.

ASKIN, R.G., DROR, M. & VAKHARIA, A.J. (1994) Printed circuit board family grouping and component allocation for a multimachine, Open-shop assembly cell. *Naval Research Logistics*, 41, pp. 587-608.

BEN-ARIEH, D. & DROR, M. (1990) Part assignment to electronic insertion machines: Two machine case. *International Journal of Production Research*, 28/7, pp. 1317-1327.

CARMON, T.F., MAIMON, O.Z. & DAR-EL, E.Z. (1989) Group set-up for printed circuit board assembly. *International Journal of Production Research*, 27/10, pp. 1795-1810.

CRAMA, Y. KOLEN, A.W.J. OERLEMANS, A.G. & SPIEKSMA, F.C.R. (1990) Throughput rate optimization in the automated assembly of printed circuit boards. *Annals of Operations Research*, 26, pp. 455-480.

DUMAN, E. (1998) *Optimization issues in automated assembly of printed circuit boards*. PhD Thesis, Bogazici University.

DUMAN, E. & OR, I. (2004) Precedence constrained TSP arising in printed circuit board assembly. *International Journal of Production Research*, 42/1, pp. 67-78.

FRANCIS, R.L. & HORAK, T. (1994) A note on reel allocation problem. *IIE Transactions*, 26/3, pp. 111-114.

HILLIER, M.S. & BRANDEU, N.L. (2001) Cost minimization and workload balancing in printed circuit board assembly. *IIE Transactions*, 33, pp. 547-557.

JI, P. & WAN, Y.F. (2001) Planning for printed circuit board assembly : the state-of-the-art review. *Int. J. of Computer Applications in Technology*, 14 Nos.4/5/6, pp. 136-144.

KLOMP, C., KLUNDERT, J., SPIEKSMA, F.C.R. & VOOGT, S. (2000) The feeder rack assignment problem in PCB assembly: A case study. *International Journal of Production Economics*, 6, pp. 399-407.

McGINNIS, L.F., AMMONS, J.C., CARLYLE, M., CRANMER, L., DEPUY, G.W., ELLIS, K.P., TOVEY, C.A. & XU H. (1992) Automated process planning for printed circuit card assembly. *IIE Transactions*, 24/4, pp. 18-29.

SADIQ, M., LANDERS, T.L. & TAYLOR G.D. (1993) A heuristic algorithm for minimizing total production time for a sequence of jobs on a surface mount placement machine. *International Journal of Production Research*, 11/6, pp. 1327-1341.