Wichita State University Libraries
SOAR: Shocker Open Access Repository

Mehmet Bayram Yildirim

Industrial Engineering

Dispatching Rules for Allocation of Component Types to Machines in the Automated Assembly of Printed Circuit Boards

Mehmet Bayram Yildirim

Wichita State University, bayram.yildirim@wichita.edu

Ekrem Duman

Dogus University, Acibadem, Istanbul, Turkey, eduman@dogus.edu.tr

Dilek Duman

Intertech Information Technology and Marketing Inc., Istanbul, Turkey, Dilek.Duman@denizbank.com

Recommended citation

Yildirim, Mehmet Bayram., Duman, Ekrem. and Dilek Duman. 2006. Dispatching Rules for Allocation of Component Types to Machines in the Automated Assembly of Printed Circuit Boards. *Proceedings of ISCIS'06: The 21st International Symposium on Computer and Information Sciences*, Istanbul, Turkey which is published in Lecture Notes in Computer Science, Volume 4263, pp. 55-64

This paper is posted in Shocker Open Access Repository

<http://soar.wichita.edu/dspace/handle/10057/3444>

Dispatching Rules for Allocation of Component Types to Machines in the Automated Assembly of Printed Circuit Boards

Mehmet Bayram Yildirim¹, Ekrem Duman², and Dilek Duman³

¹ Wichita State University, Industrial and Manufacturing Engineering Department,
1845 Fairmount Wichita, KS 67260-0035, USA
Bayram.Yildirim@wichita.edu

² Dogus University, Industrial Engineering Department, Zeamet
Sok. No.21,
Acibadem, Istanbul, Turkey
Eduman@dogus.edu.tr

³ Intertech Information Technology and Marketing Inc., Kasap Sok. No.15,
Esentepe, Istanbul, Turkey
Dilek.Duman@denizbank.com

Abstract. In this paper, we propose a model and algorithms to solve a load balancing problem on a printed circuit board assembly line. On this line of two identical machines, the problem of allocation of component types to machines is analyzed in detail. Twenty eight dispatching rules are developed and extensive computational experimentation is performed. It has been observed that the imbalance per board increases as a function of the number of board types. Furthermore, the greedy dispatching rules perform better than the construction dispatching rules.

Keywords: Printed Circuit Board Assembly, Load Balancing, Heuristics.

1 Introduction

In this paper, we propose simple dispatching rules which can be implemented and used fairly easily in a production environment where there are two identical resources used in producing products which require processing by both of the resources. Each resource has access to separate storage areas (i.e., bins) with fixed capacity. Each bin stores certain components that must be placed on the product. Furthermore, it is assumed that the order which these components are placed on the product is not important. It is also assumed that there is no buffer where the semi-finished products can be stored. In our setting, we assume that a conveyor belt transfers the semi-finished product from resource 1 to resource 2. Thus, resource 1 can not start to work on a new part unless the second resource is idle. Our goal is to assign components to bins in such a way that the total workload for each resource is balanced. This will ensure that both resources will be utilized efficiently and the total idle time for each resource will be minimized.

The use of numerically or computer controlled electronic component placement machines in assembling printed circuit boards (PCB) brings major gains in productivity and efficiency through their fast, error free and reliable component placement operations. To utilize these automatic machines efficiently, serious planning and scheduling decisions should be made on the production floor. Some of these decisions are 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. All these problems are interdependent, i.e., the solution of one problem affects the solution of others. Such interdependency is more evident between the first two and last two problems. When an optimal solution is sought, all four problems should be solved simultaneously. However, since each of these problems is quite complex by itself, trying to build and to solve a monolithic model is quite difficult and intractable. Hence, usually, these problems are solved separately and iterative solution methods are deployed to cope for the interaction between them.

The literature on PCB assembly problems is quite extensive. However, most of the literature is related to the feeder configuration and placement sequencing problems. A general overview of PCB assembly problems is given by McGinnis et al. [2] and Ji and Wan [3]. Francis and Horak [4] consider the problem of choosing the numbers of reels of each type of components to be used in populating a printed circuit board by an 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. [5] minimize the total setup time to change the feeder configuration and propose the group set-up method (grouping similar boards), which can significantly reduce set-up times. Askin et al. [6] address 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 [7] study the problem of assigning component types to insertion machines with the aim of balancing the workload assigned to machines. The study of Ho and Ji [8] is one of the few studies attempting to solve two PCB assembly problems (feeder configuration and placement sequencing) simultaneously.

Sadiq et al. [9] propose an iterative approach that minimizes the total production time for a group of PCB assembly jobs on a single machine when the sequencing and allocation of different component reels to feeder carriage are considered together. Ahmadi et al. [10] consider a placement machine, which features two fixtures for the delivery of components to the placement heads. They investigate the case where all components are accessible and the case of a static pick sequence. Crama et al. [11] propose a heuristic hierarchical approach to optimize 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 given a family of boards Klomp et al. [12] propose a heuristic for the feeder rack assignment problem. Hillier and Brandeau [13] develop the cost minimizing workload balancing heuristic to balance the workload among the semi-automatic placement machines and the manual assembly stations. Ben-Arieh and Dror [7] only consider the case where different board types require several different components and the demand for each board type is equal.

Although the use of electronic component placement machines has brought reliability and speed to the PCB assembly process, to get higher utilization, one needs to solve the resulting complex operations research problems efficiently. In this study, the problem of distributing the assembly workload to two machines deployed on an assembly line with two identical component placement machines to minimize the line idle time is considered. A mathematical model and several dispatching rules are proposed. We extend Ben-Arieh and Dror [7] model also to include demand information for all boards.

In the next section, the description setting considered in this study and the formulation of the problem is given. Some heuristics to solve this problem can be found in section three. In section four, the results and performances of these heuristics on randomly generated test problems are discussed. Finally, in section five, the concluding remarks are given.

2 Problem Description and Formulation

The form of the load balancing (component allocation) problems in automated PCB assembly shows a large variability. Some of the features that determine the underlying component allocation problem setting are differences in machine architecture (type), characteristics of the production processes and engineering preferences. In this study, the machine type considered is one with a component pickup device, a stationary placement head and a moving carrier board. In this machine, circular shaped turning component pickup device takes the role of the sequencer machine. The component tape is 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 pickup device, which usually has 20 to 120 heads, picks up the components to its heads in the placement order from the component tapes. The placement sequencing problem turns out to be a Chebyshev Traveling Salesman Problem and the layout of the component tapes can be formulated as a simple allocation problem [1]. Two examples of this type of machines are the Universal 6287A and the Dynapert Intellisert V12000 axial component placement machines.

In our setting, 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). It is assumed that the boards are produced in a high-mix, low-volume production environment, where the set of all

Fig. 1. Two Placement Machines on a Line

component types to be populated on all PCB types in the planning horizon, will be allocated to two identical placement machines with the same speed, the same nozzle sets and the same number of feeder locations. The demand for PCB types are known and fixed for the planning period under consideration. Furthermore, it is assumed that the insertion time is one unit for each of the components regardless of the PCB type and components' location on the boards. Component types are also identical in regard to their slot requirements in the feeder area and all use the same nozzle. Also the feeder capacity of each machine is strictly less than the minimum number of different components placed onto a single circuit board type, so that full assembly of each PCB type requires both machines. As a result, assembly of a new board cannot start unless both machines are cleared by the currently assembled one. The total number of component types is equal to the total number of feeder locations on two machines. It is assumed that there is no sequence dependent setup time when switching between different board types. However, the setup time to change a component type in a feeder is very high. Note that if this was not the case, then the load balancing problem would be considerably simplified: each time a board type is to be assembled, the feeders could be configured to balance that particular board type, by solving a single 2-partition problem (which is NP-Hard). Since the boards are reliably transferred by the conveyor belt, there is no engineering preference regarding population of the PCBs on a single machine. Buffer of partially completed boards is not desired due to the engineering preferences.

The objective is to partition the set of component types into two to maximize the average production rate. In other words, our objective is 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 problem stated above can be formulated as an integer program. First, we will introduce the notation. Let i be the component type index ($i=1, \dots, n$), j be the board type index ($j=1, \dots, m$), a_j be the number of boards of type j to be produced, P_{ij} be the number of components of type i to be placed on board type j and F be the feeder capacities of machine 1 and machine 2

In this problem, the decision is to assign the components to different machines in such a way that the total processing time required to process all of the demand on a machine is as close to the total amount of processing on the other machine. Note that we assume the total processing time is proportional to the total number of components placed. The decision variables are

$$X_i = \begin{cases} 1 & \text{if component type } i \text{ is assigned to machine 1} \\ 0 & \text{otherwise} \end{cases}$$

Now, the problem can be formulated similar to [7] as follows:

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

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

$$\sum_{i=1}^n X_i \leq F \quad (3)$$

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

$\sum_{i=1}^n X_i P_{ij} - \sum_{i=1}^n (1 - X_i) P_{ij}$ reflects the total waiting (idle) time or imbalance of the machines for a particular board type j . As a result, 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. Using constraint (2), it is ensured that the workload assigned to machine 1 is greater than or equal to the workload assigned to machine 2. This guarantees that a smaller amount of work-in-process inventory is accumulated between two machines. Constraint (3) is the feeder capacity constraint. Since

$$\sum_{j=1}^m a_j \left| \sum_{i=1}^n X_i P_{ij} - \sum_{i=1}^n (1 - X_i) P_{ij} \right| = \sum_{j=1}^m \left(2a_j \sum_{i=1}^n X_i P_{ij} - N_j \right) \quad (4)$$

where $N_j = \sum_{i=1}^n P_{ij}$ is the number of components required for board type j , the objective function and constraint (2) can further be simplified to the second term in equation (4).

When $a_j = 1$, $j = 1..n$, Ben-Arieh and Dror [7] prove that the resulting problem is NP-Complete. As a result, the problem formulated above is also NP-Complete. Below, we summarize the heuristic methods considered to solve the component allocation model presented above.

3 Solution Procedures Developed

A total of 28 dispatching rules are developed to find a good quality solution for the component allocation problem defined in the previous section in a reasonable amount of time. All of these algorithms have two mechanisms in their structure: component sorting and component assignment. Component sorting rules determine the order of the component assignment to the machines. On the other hand, component assignment rules decide which machine a given component type is to be assigned. Below, we present five component sorting and three component assignment rules:

3.1 Component Sorting Rules

i- Component Popularity (CP): The popularity of a component is measured by the number of board types on which it is used, i.e., $CP_i = \sum_{j=1}^m Y_{ij}$ where

$$Y_{ij} = \begin{cases} 1, & \text{if component type } i \text{ is placed on board type } j \\ 0, & \text{otherwise.} \end{cases}$$

In this sorting scheme, components are sorted in non-increasing CP values. If there are more than one component types with the same popularity, the one which is consumed more in the planning period gets higher priority.

ii- *Component Usage (CU)*: The total number of placements of each component type over all boards is calculated (i.e., $CU_j = \sum_{i=1}^m a_j P_{ij}$) and the components are sorted

in non-increasing order of CU values. This way, component types having higher usage are assigned to machines in a balanced manner in the earlier stages and it is hoped that the balance will not be much deteriorated by the least used component types.

iii- *Standard Error (SE)*: The standard error for a component type i is given by the following equation:

$$SE_i = \frac{\sum_{j=1}^m \left(\frac{P_{ij}}{m} - P_{ij} \right)^2}{\sum_{j=1}^m \frac{P_{ij}}{m}} \quad (5)$$

where m is the number of board types. Under this rule, the component types are ordered according to non-decreasing order of their SE values.

The logic for CP and CU rules is quite intuitive. SE sorting rule ensures that the component types which have a homogenous usage over all board types (which may imply component types used by most of the board types) are more important than the others and they should be handled first and distributed to machines as equally as possible. The numerator of the standard error expression (sum of squared errors) reflects this. On the other hand, among the component types, which have similar usage homogeneity, the ones with larger average usage are more important which is reflected by the expression in the denominator.

iv- *Board-component Usage (BU)*: This is a two step component sorting rule; first, board types are ordered according to the number of components to be populated on them. Let N_j be the total number of components to be placed on board type j ($N_j = \sum_i a_j P_{ij}$). Board types requiring more components (i.e., the higher N_j) are

prioritized in the BU order. Initially, the first board type in the order is selected. The components on that particular type of 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 using the same logic. This procedure continues until the BU list contains all component types.

v- *RN (random)*: Component types are ordered in a random manner. This rule will be used as a benchmark for comparisons with other sorting rules.

3.2 Component Assignment Rules

i- *Less Work Rule A (LWA)*: This rule selects a component type from a component order list described above and assigns the component to the machine with the least assigned workload. While processing the BU list, the assigned work of a machine is the total number of component placements for the board type being handled at that time.

- ii- *Less Work Rule B (LWB)*: This rule is same as LWA but differs only in heuristics containing the BU component sort rule. During the processing of the BU list, the assigned work of machines is considered as the total work related to all board types.
- iii- *Greedy Optimization (GR)*: This rule seeks a partial load balance at each step. Each time a new component type is picked up from a sorting 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 with the lower objective function value (imbalance) is chosen for assignment.

3.3 Proposed Dispatching Rules

In this research, 28 load balancing dispatching rules are experimented for the two identical machines case. A naming convention is used to identify the individual dispatching rules with respect to the sort and assignment rules deployed: A two letter code for component sort rule is first written in upper case and it is followed by the component assignment rule code in lower case (e.g. CUgr). The letters "a" and "b" are used to denote component assignment rules LWA and LWB, respectively. With RAN, component types in the RN sort list are assigned to machines in an alternating manner.

There are 4 sorting rules (excluding RAN) and 3 assignment rules, thus a total of 12 combinations. If the dispatching rule uses one component sort list partially and then another one, both sort rules are written one after the other and this is followed by a number (1, 2 or 3) which denotes the filling ratio of total feeder locations on both machines upon which the dispatching rule switches from the first sort list to the second: "1" denotes 25 percent filling ratio, whereas "2" and "3" denote 50 percent and 75 percent, respectively. If the filling ratio does not lead to an integer value for the number of feeder locations, it is rounded off to the nearest integer.

For example, if we have two identical machines, each having 15 feeder locations and if we are trying to assign 30 component types to these machines using CUBU1a dispatching rule, it will work as follows: A total of eight component types will be picked from the CU ordered list and will be assigned to either of the machines according to the LWA assignment rule. After this, the BU ordered list will be considered and starting from the first component type in the list that has not already been assigned to a machine, all component types of BU that are not yet assigned to a machine will be assigned to the machines according to the LWA rule.

Note that assignment for BU sort rule can be done according to both Less Work A and B rules. Considering all of these combinations, the total number of heuristics that can be considered is 28. Below is the list of the dispatching rules we experimented with for two identical machines case:

Cpa	CPgr	CPBU1a	CUBU1a	SEBU1a
CUa	CUgr	CPBU2a	CUBU2a	SEBU2a
SEa	SEgr	CPBU3a	CUBU3a	SEBU3a
BUa	BUgr	CPBU1b	CUBU1b	SEBU1b
BUb	RAN	CPBU2b	CUBU2b	SEBU2b
		CPBU3b	CUBU3b	SEBU3b

4 Computational Experimentation

In this section, we will first present the experimentation setup. Next, we will briefly compare the effectiveness of the proposed dispatching rules and discuss the results obtained to gain a better insight to the component allocation problem.

4.1 Experimental Setup

In generating the test problems the following design is considered: it is assumed that the production facility can produce 10 or 20 different types of boards. The demand for each board type is uniformly distributed between 1,000 and 10,000. The number of component types on a board varies between 20 and 120. The placement matrix (P), is generated as follows: with 0.40 probability, P_{ij} is zero (i.e., component i is not used on board type j). The P_{ij} value is 1, 2, 3, 4, 5, 6 and 7 with a probability of 0.15, 0.15, 0.08, 0.07, 0.06, 0.05 and 0.04, respectively.

The following naming convention has been utilized for the randomly generated test problems. PRnm is a problem with n components (the problem size), and m boards. In our experimental setting, test problems are generated for $n=20, 30, 40, 60, 80, 100$

Table 1. Deviation from the Best Solution, $m=10$

	CPa	CUa	SEa	BUa	CUBU1a	CUBU2a	CUBU3a	CPBU1a	CPBU2a	CPBU3a	SEBU1a	SEBU2a	SEBU3a	BUB	CUBU1b
PR2010A	1.06	1.43	1.82	0.85	1.31	1.20	1.04	1.16	0.86	1.00	1.24	1.57	1.63	1.35	1.22
PR3010A	2.10	1.81	2.55	2.79	2.42	2.02	1.61	2.14	2.27	1.96	2.35	1.28	1.69	1.48	1.48
PR4010A	1.60	2.07	2.40	3.00	2.77	2.33	2.24	3.11	1.65	1.34	3.10	2.67	1.85	1.58	2.32
PR6010A	2.62	2.93	4.68	4.71	4.89	3.78	2.93	4.08	4.09	2.54	4.93	4.17	4.31	3.96	3.02
PR8010A	5.89	4.79	6.84	7.88	5.84	4.67	3.93	6.00	6.08	5.07	6.96	5.36	3.93	3.14	5.22
PR10010A	5.58	3.82	4.59	6.17	5.82	4.57	5.36	5.54	4.98	4.46	5.59	5.15	3.88	3.03	3.87
PR12010A	6.80	6.15	6.45	9.13	6.97	6.11	4.38	7.86	6.77	6.89	6.71	6.66	5.68	5.44	5.32
Average	3.67	3.28	4.19	4.93	4.29	3.53	3.07	4.27	3.81	3.32	4.41	3.84	3.28	2.85	3.21

	CUBU2b	CUBU3b	CPBU1b	CPBU2a	CPBU3b	SEBU1b	SEBU2b	SEBU3b	CPgr	CUgr	SEgr	BUgr	RAN	MIN
PR2010A	1.28	1.33	1.03	1.16	1.20	0.98	1.34	1.44	0.16	0.35	0.43	0.40	1.10	0.16
PR3010A	2.13	1.83	1.51	1.53	1.96	1.49	2.29	1.75	0.54	0.34	1.23	0.09	3.47	0.09
PR4010A	1.85	2.25	2.28	1.38	1.50	1.93	2.40	1.97	0.55	0.44	1.16	0.16	2.86	0.16
PR6010A	2.62	2.84	3.51	3.06	2.46	3.50	3.26	4.62	0.85	0.09	2.27	0.48	4.93	0.09
PR8010A	4.15	4.56	4.18	4.65	4.96	4.24	4.32	4.58	0.65	0.19	1.28	0.37	5.33	0.19
PR10010A	4.21	4.23	4.23	3.83	5.17	4.66	4.46	4.87	0.30	0.11	1.40	0.40	3.97	0.11
PR12010A	5.54	5.54	4.63	5.76	6.85	4.86	4.91	5.84	0.34	0.21	3.14	0.50	5.62	0.21
Average	3.11	3.23	3.05	3.05	3.44	3.09	3.28	3.58	0.48	0.25	1.56	0.34	3.90	0.25

Table 2. Deviation from the Best Solution, $m=20$

	CPa	CUa	SEa	BUa	CUBU1a	CUBU2a	CUBU3a	CPBU1a	CPBU2a	CPBU3a	SEBU1a	SEBU2a	SEBU3a	BUB	CUBU1b
PR2010A	1.05	0.88	0.77	0.98	0.80	0.93	1.01	0.77	0.83	0.99	0.98	0.83	0.77	0.84	0.62
PR3010A	1.02	1.03	1.14	0.84	0.89	0.84	1.03	1.27	0.99	0.67	0.97	1.04	0.90	0.89	1.09
PR4010A	0.70	0.70	0.98	1.33	1.44	1.02	0.90	1.51	1.08	0.95	1.52	1.29	1.06	1.13	1.23
PR6010A	1.64	1.38	1.56	2.09	2.13	1.20	1.17	1.75	1.74	1.48	2.06	1.89	1.54	1.50	1.24
PR8010A	1.52	2.09	2.61	2.34	2.19	1.71	1.87	2.30	1.91	1.81	2.51	2.15	2.63	1.90	2.11
PR10010A	2.57	3.24	2.83	3.53	3.19	2.90	2.22	3.92	2.25	2.44	3.72	3.24	2.92	2.14	1.85
PR12010A	2.60	3.50	2.61	4.44	3.49	3.13	4.00	3.81	2.74	2.34	3.58	3.21	2.54	2.66	2.83
Average	1.59	1.83	1.79	2.25	2.02	1.68	1.74	2.19	1.65	1.53	2.19	1.95	1.76	1.58	1.57

	CUBU2b	CUBU3b	CPBU1b	CPBU2a	CPBU3b	SEBU1b	SEBU2b	SEBU3b	CPgr	CUgr	SEgr	BUgr	RAN	MIN
PR2010A	0.85	0.87	0.95	0.89	1.05	0.91	0.63	0.82	0.33	0.23	0.42	0.01	0.91	0.01
PR3010A	1.00	1.16	0.89	1.19	1.04	0.59	1.18	0.87	0.15	0.06	0.55	0.26	1.33	0.06
PR4010A	0.92	0.93	1.28	1.28	1.07	1.26	1.30	0.92	0.19	0.15	0.42	0.11	0.92	0.11
PR6010A	1.48	1.42	1.52	1.37	1.48	1.30	1.44	1.29	0.07	0.18	0.47	0.19	1.54	0.07
PR8010A	2.15	1.85	2.00	2.02	1.67	2.16	2.37	2.21	0.23	0.24	0.71	0.15	2.26	0.15
PR10010A	2.57	3.23	2.60	2.51	2.82	2.62	2.67	2.59	0.47	0.18	0.82	0.25	3.12	0.18
PR12010A	3.36	3.43	2.98	2.89	2.67	2.65	2.57	3.07	0.49	0.07	1.11	0.44	3.31	0.07
Average	1.76	1.84	1.75	1.73	1.69	1.64	1.74	1.68	0.28	0.16	0.64	0.20	1.91	0.16

and 120 and $m=10$ and 20. For each (n,m) combination, heuristics are tested on six randomly generated test problems. For all dispatching rules the percent deviations of the algorithms from the best solution are calculated and displayed on tables 1 and 2 (i.e., each entry in these tables shows the average imbalance percent deviation from the best solution of six problems having the same parameters). In these tables, the overall average for all problems (average) is also displayed.

4.2 Comparison of the Proposed Heuristics

When $m=10$, CUa, which is very similar to the algorithm proposed by Ben-Arieh and Dror [7], on average, behaves 328% worse than the best solution found by 28 dispatching rules as seen in table 1. Among 28 dispatching rules, other than the greedy ones, when $m=10$, the best average solution obtained is worse than 285%. The greedy dispatching rules perform much better than the construction dispatching rules. On average SEgr finds solutions which are 156% worse than the best solution (SE's performance is 419%). The other greedy dispatching rules, on average find solution 35.6% worse than the best solution. Similar observations can be found when $m=20$ (see table 2). Among the greedy dispatching rules, BUgr provides better solutions when n is small. When n gets larger, CUgr is the most efficient dispatching rule. Another interesting point to note is that some dispatching rules perform worse than the random allocation.

To summarize, when average deviations are considered, it can be observed that the greedy dispatching rules certainly outperform the others: CUgr and BUgr are the first best and second best heuristics for both $m=10$ and $m=20$ problems. Also, note that the performance of CUgr dispatching rule gets better as the problem size increases.

5 Concluding Remarks

In this study, the problem of allocating component types to machines in PCB assembly is considered. The focus is on a two identical machines case. Twenty eight different dispatching rules are developed and their performances are tested. When the demand for each board type is different, the CUa dispatching rule proposed by Ben-Arieh and Dror [7] performs quite poor. Among the heuristics proposed in this study, the CUgr greedy dispatching rule performed the best.

For future research, one can extend the proposed dispatching rules to environments where there are restrictions on allocating certain components to specific machines, or when the slot requirements of component types are not the same, or when the machines have different nozzle sets that they can handle.

References

1. Duman E.: Optimization issues in automated assembly of printed circuit boards. Dissertation, Bogazici University, Turkey, 1998.
2. McGinnis, L.F., Ammons, J.C., Carlyle, M., Cranmer, L., Depuy, G.W, Ellis K.P., Tovey, C.A., Xu, H.: Automated process planning for printed circuit card assembly. IIE Transactions 1992; 24/4: 18-29.

3. Ji, P., Wan, Y.F.: Planning for printed circuit board assembly: The state-of-the-art review. *Int. J. of Computer Applications in Technology* 2001; 14 Nos.4/5/6: 136-144.
4. Francis, R.L., Horak, T.: A note on reel allocation problem. *IIE Transactions* 1994; 26/3: 111-114.
5. Carmon, T.F., Maimon, O.Z., Dar-El, E.Z.: Group set-up for printed circuit board assembly. *International Journal of Production Research* 1989; 27/10: 1795-1810.
6. Askin, R.G., Dror, M., Vakharia, A.J.: Printed circuit board family grouping and component allocation for a multimachine, Open-shop assembly cell. *Naval Research Logistics* 1994; 41: 587-608.
7. Ben-Arieh, D., Dror, M.: Part assignment to electronic insertion machines: Two machine case. *International Journal of Production Research* 1990; 28/7: 1317-1327.
8. Ho, W., Ji, P.: Component scheduling for chip shooter machines: a hybrid genetic algorithm approach. *Computers and Operations Research* 2003; 30/14: 2175-2189.
9. Sadiq, M., Landers, T.L., Taylor, G.D.: A heuristic algorithm for minimizing total production time for a sequence of jobs on a surface mount placement machine. *International Journal of Production Research* 1993; 11/6: 1327-1341.
10. Ahmadi, J., Grotzinger, S., Johnson, D.: Component allocation and partitioning for a dual delivery placement machine. *Operations Research* 1988; 36/2: 176-191.
11. Crama, Y., Kolen, A.W.J., Oerlemans, A.G., Spieksma, F.C.R.: Throughput rate optimization in the automated assembly of printed circuit boards. *Annals of Operations Research* 1990; 26: 455-480.
12. Klomp, C., Klundert, J., Spieksma, F.C.R., Voogt, S.: The feeder rack assignment problem in PCB assembly: A case study. *International Journal of Production Economics* 2000; 6: 399-407.
13. Hillier, M.S., Brandeu, N.L.: Cost minimization and workload balancing in printed circuit board assembly. *IIE Transactions* 2001; 33: 547-557.