2007−MPS−66 (20)
2007／9／4

基板生産時における部品装着機の部品吸着順序問題に対する近似解法

山田 剛史[†], 中森 眞理雄[†],
[†] 東京農工大学工学府

基板生産時において部品装着機を用いた工程におけるスケジューリングは基板生産時間に大きな影響を与える．本論文では部品装着機における問題である部品吸着順序問題に対して注目した．部品吸着順序問題を多重集合被覆問題としてモデル化を行い，貪欲法を提案する．そして，提案した貪欲法に対して解析を行った．

An Approximation Algorithm of the Pickup Sequencing of a Component Placement Machine for Printed Circuit Boards

Tsuyoshi YAMADA[†], Mario NAKAMORI[†],
[†] Faculty of Engineering, Tokyo Agriculture and Technology University

Today, almost all circuits of electronic devices are printed circuit boards. In the present paper we consider the pickup sequencing problem, which is the most time-consuming aspect of board production. Since the pickup sequencing problem can be modeled as a set multicovering problem, we propose a greedy algorithm based on the dual fitting analysis.

1 Introduction

Today, almost all circuits of electronic devices are printed circuit boards(PCBs). One of the most important processes is PCB assembly. In order to achieve high rates of throughput in PCB assembly machines, we have to generate instruction sequences of high quality. Among the many stages of PCB assembly, that of component placement is the most critical. In this process, components are placed on a PCB by component placement machines.

In the present paper, we consider the *Multi-Head Placement Machine*, which is structurally more complicated than the other component placement machines. The Multi-Head Placement Machine is configured for moving placement with an *arm* with several *heads*, *Automatic Nozzle Changer* (ANC) and *feeder slots* (see Figure 1). Some authors [1, 3, 4] focused on the the Multi-Head Placement Machine. The components to be placed on the circuit board are stored in tape reels set in the feeder slots. The head can pick up a component by using a nozzle from the feeder slot and place the component on the PCB. The heads first pick up several components, either simultaneously (in what is referred to as *gang-pick*) or by moving along the feeder array, and then above the circuit board to place the components in the appropriate locations. However, since differences in the size of components mean that the same nozzle cannot place all of the components, nozzles need to be changed. Nozzle changes are carried out automatically by the ANC throughout the process of component placement. The process of component placement therefore consists of placing the components, picking up the components, and changing nozzles.

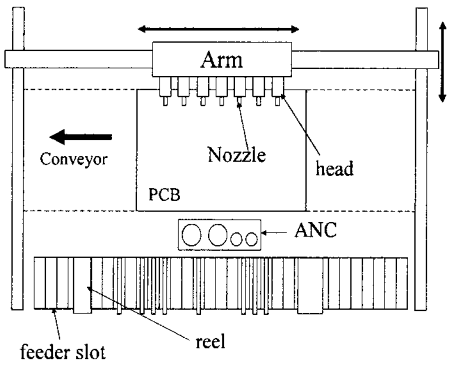

Figure 1: Multi-Head Placement Machine

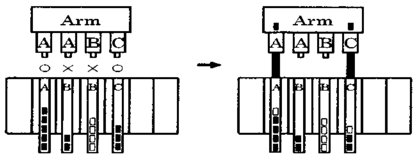

Figure 2: *Gang-pick*

These three operations are performed alternately and a set of these pickup and placement operations is referred to as a *task*. In each task, the arm carries out several pickup and placing operations. When the placement of all of the components required for the completion of a particular task is completed by the arm, the arm moves to pick up the components required for the next task from the feeder slots. A circuit board is completed when all of the tasks are finished.

2 Definition of the problem

In this paper, we consider *Pickup Sequencing Problem* (*PUSP*). PUSP determines the sequence of picking up components from the feeder slot. The each feeder slot has only parts of kinds. Also, the head can only one component. However, all heads need not pick up the component. If one head picks up one component, the arm may move on the PCB to place the component.

The pickup sequence producing the minimum picking time is given as the solution of PUSP. The main cost of PUSP is the number of occurrences required for picking up all of the components. The number of occurrences can be reduced by the gang-pick (see Figure 2). The gang-pick is that the arm picks up the several components by one occurrence. In a word, that is a multiple heads picking up one component each. If appropriate conditions are satisfied, such as if nozzle and component types are matched, then it is possible to pick more components simultaneously.

Pattern	Nozzle Used (Head Number)				
	h1	h2	h3	h4	h5
Pattern 1	A	A	B	B	C
Pattern 2	B	B	C	D	D
Pattern 3	D	A	B	C	C

Figure 3: Examples of nozzle arrangements

Slot position	Kind of Comp.	Required Nozzle	Num. of Comp.
18	C1	A	4
20	C2	B	10
24	C3	C	1
22	C4	B	4
26	C5	D	6

Figure 4: Examples of feeder arrangements

Nozzle arrangement of the arm and feeder arrangement are given and assigned before solving PUSP. In other words, in PUSP, the information that is supplied is the feeder arrangement and nozzle arrangement (see Figure 3 and 4), We call each nozzle arrangement the *nozzle pattern*. The each pattern does not decide the number of tasks used, but should minimize the total number of tasks used by each pattern. In the feeder arrangement, each component is distributed to each slot. The each component have the information of the required nozzle and the number of components.

In PUSP, gang-pick is a significant factor in reducing the number of occurrences of picking up all the components. The number of tasks is also important, and in PUSP, the sub cost is the number of tasks. The object is reducing the number of pick up components and the number of going of arm on the PCB(it is task). The number of tasks is reduced by the combination of components to be picked up in each task. The criterion of evaluating a solution of PUSP is whether the number of occurrences of picking up components is the minimum, and also whether the number of tasks is reduced. PUSP is formulated mathematically as a 0-1 integer programming problem.

3 The Algorithm

In this section, we propose an algorithm for PUSP based on a greedy algorithm.

We use the following notation to describe the algorithm.

N : upper bound of the number of tasks
J : the number of nozzles per arm
K : the last arm position
P : the number of nozzle patterns
T : the number of feeder slots

p : nozzle pattern, where $p \in \{1, 2, ..., P\}$
n : task, where $n \in \{1, 2, ..., N\}$
k : arm position, where $k \in \{1, 2, ..., K\}$
j : head number, where $j \in \{1, 2, ..., J\}$
t : slot number, where $t \in \{1, 2, ..., T\}$

The greedy algorithm produces a solution in which the objective function improves with every iteration. This algorithm roughly consists of two phases: *searching arm position* and *searching appropriate task*.

The greedy algorithm scans the feeder slot from the first arm position to the last arm position in the phase of searching arm position. At each scanning arm position k, the maximum set of components that can be picked up by the arm is calculated. The calculation is executed for each nozzle pattern. The set of head numbers, which is *simultaneous picking set*, is calculated and denoted by *SPS*(p,k).

After searching all arm positions for all nozzle patterns, the maximum *SPS*(p, k) is determined. The maximum *SPS*(p,k) may not always be unique. We denote by ω the set of the pair (p,k) corresponding to the maximum *SPS*(p,k).

Next, the algorithm selects a pair (p,k) in ω after calculating the *task occupation rate* of each *SPS*(p,k) in phase of searching the appropriate task. We define the set of head numbers as *TS*(p,n) in task n of nozzle pattern p in the constructed solution of each iteration. The *task occupation rate* is shared in *TS*(p,n) of each p,n after excluding the heads assigned to picking up the components of *SPS*(p,k). We define *task occupation rate* *TOR*(p,k,n) as follows:

$$TOR(p,k,n) = \frac{|SPS(p,k)|}{J - |TS(p,n)|}$$

However, if $TS(p,n) \cap SPS(p,k) \neq \emptyset$, the value of *TOR*($p$,$k$,$n$) is defined as 0, because it is not possible to assign *SPS*(p,k) to task n of the nozzle pattern p.

The algorithm calculates *TOR*(p,k,n) for each element of ω for all tasks in each nozzle pattern and searches the maximum value of *TOR*(p,k,n).

The pair (p,k) and task n are selected such that *TOR* becomes the maximum value. We define p,k,n of the selected pair (p,k) and task n as p',k',n'. If the value of *TOR*(p,k,n) in all elements of ω is zero (in this case, no *SPS*(p,k) will be assigned to any task), an arbitrary pair of (p,k) is selected.

The algorithm assigns the *SPS* (p',k') to an appropriate task n' of the nozzle pattern. The appropriate task is selected such that *TOR* is the maximum. If the selected *SPS* cannot be assigned to any task (that is, *TOR* cannot be calculated for any pair (p,k) in searching the appropriate task phase), the algorithm increments the number of tasks and assigns the *SPS* (p',k') to the generated new task in *TS*. When assigning *SPS* (p',k') is completed, the algorithm returns to the searching the arm position phase. The algorithm terminates when all of the components are picked up.

4 Analysis

We considered converting PUSP to a problem whose objective function is the number of all components to be picked up without consideration of the number of tasks. This problem is regarded as a set multicovering problem without weight. Considering this problem, the proposed algorithm essentially equal to the natural greedy algorithm for a set multicovering problem. Let us, the number of elements is E for a set multicovering problem. The natural greedy algorithm achieves an approximation guarantee of H_E by authers[2) . Therefore, the proposed algorithm achieves an approximation guarantee of H_J in the number of picking up all components for RPUSP.

5 Experimental Results

In order to evaluate the performance of the proposed algorithm, we implemented our algorithm and made experiments with randomly generated data of PCBs. We generated two kinds of data. The details of the generated data are as follows:

- J:5 and 10
- the number of nozzle patterns P:[2,5]
- the number of component types T:[10,25]
- the number of components t C_t:[1,25]
- number of types of nozzles:[3,8]

The number of generated data was 200. The number for the five nozzle data was 100 and the number of ten nozzle data was 100. We implemented the algorithm using Visual C++ for Windows XP. The implementation of the algorithm was performed on a Intel Xeon Processor 3 GHz and 4 GB RAM. We also obtained the optimal solution with the default parameter of ILOG CPLEX10.1. The time limit was four hours and other options were the defaults in CPLEX.

Results of the five nozzle data showed the average gap of 2.55% for the number of picked-up components. The average gap for the number of tasks is 11.24%, which is relatively large given that the number of tasks is small. In the results of the ten nozzle data, the average gap of 6.09% is for the number of all picked-up all components. The average result for the number of tasks is 4.42%. The each average gap is taken for 100 data.

The example of the experiments result in ten nozzle data is shown in Table 1. Table 1 shows the nearly optimal solutions and the obtained solutions by our algorithm for the number of tasks.

The proposed algorithm achieves an approximation guarantee of H_J (in this data, H_{10}) indicating that the accuracy of the solution decreases. However, the number of tasks is better than the results obtained for five nozzle data. In the result of the number of tasks, more data than for optimal tasks existed (for example: Data No. 8 of Table 1). Even though the frequency of the number of picked-up components was poor, the number of picked-up components in one task is large. This is because there are numerous picking combinations associated with one task. Combining tasks works well since the number of picking combinations is high in phase of searching appropriate task. Consequently, the number of tasks indicates a tendency toward improvement as the frequency of the number of picked-up components decreases. The computation time of the proposed algorithm was 0.1 second or less for all data.

6 Conclusion

In this paper, we considered the pickup sequencing problem. After formulating the problem, we proposed an algorithm for solving the problem by heuristic methods.

Table 1: Example of 10 nozzle result (Task)

No	Cplex	GR	Gap(%)	CplexTime(s)
1	20	26	30.00	119.3
2	23	23	0	14400
3	19	18	-5.26	14400
4	21	21	0	4.39
5	19	20	5.26	14400
6	25	25	0	2.67
7	25	26	4.00	17.23
8	32	24	-25.00	15.08
9	21	24	14.29	7.11
10	34	37	8.82	1.2
Average	-	-	4.42	-

Our findings show that our algorithm achieved a solution that produces an approximately optimal value for the number of components to be picked up. However, within the context of the number of tasks, there exists the potential for improvement of the algorithm.

References

1) Lee, S. H., Park, T. H., Lee, B. H., Kwon, W. H. and Kwo, W.: a Dynamic Programming Approach to a Reel Assignment Problem of a Surface Mounting Machine in Printed Circuit Board Assembly, *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference*, pp. 227–232 (1998).

2) Lovasz, L.: On the Ratio of Optimal Integral and Fractional Covers, *Discrete Mathematics*, Vol. 13, pp. 383–390 (1975).

3) Magyar, G., Johnsson, M. and Nevalainen, O.: on Solving Single Machine Optimization Problems in Electronics Assembly, *Journal of Electronics Manufactruing*, Vol. 9, No. 4, pp. 249–267 (1999).

4) Wilhelm, W. E., Arambula, I. and Choudhry, N. N. D.: Optimizing Picking Operations on Dual-Head Placement Machines, *IEEE Transactions on Automation Science and Engineering*, Vol. 3, No. 1, pp. 1–15 (2006).