2002년도 대한전기학회 하계학술대회 논문집 2002.7.10-12

동적계획법에 의한 멀티헤드 겐트리형 칩마운터의 장착순서 최적화

김 동 만*, 이 재 영*, 박 태 형**
*충북대학교 대학원 제어계측공학과,
**충북대학교 전기전자 및 컴퓨터 공학부

A Dynamic Programming Approach to Mount Sequence Optimization for Multihead-Gantry Chip Mounter

Dong-Man Kim*, Jae-Young Lee*, Tae-Hyoung Park**
*Dept. of Control & Instrumentation Eng. Graduate School, Chungbuk National Univ.
**School of Electrical & Computer Eng. Chungbuk National Univ.

Abstract - 표면실장형 인쇄회로기판 조립용 칩마운터의 효율적인 운용을 위한 부품 장착 순서를 최적화하는 방법을 모색한다. 연구 과정은 멀티헤드 칩마운터 부품 장착 순서를 동적계획법으로 수학식으로 모델링하고, 동적계획법의 방법으로 대상 칩마운터의 장착 순서를 생성한다. 생성된 결과는 다른 알고리즘을 적용하여 생성된 결과와 비교 분석을 한다.

1. 서 론

현재의 전자제품은 조립의 신뢰성 및 효율성을 위하여 표면실장형(SMD:surface mounted device)형 인쇄회로기판(PCB: printed circuit board)에 조립되고 있다. 표면실장형 부품을 PCB에 조립하는 전체 시스템을 SMT(surface mounting technology) 인라인(in-line) 시스템이라 한며, PCB 표면에 납 크림을 도포하는 스크린프린터(screen printer) 공정, PCB에 부품을 실장하는 칩마운터(chip mounter) 공정, PCB에 도포된 납을 경화하는 리플로어(reflower) 공정으로 이루어져 있다.이 중에서 가장 중요한 부분은 PCB 조립공정 중에서 시간이 가장 많이 걸리는 칩마운터 공정이라 할 수 있다. 칩마운터 공정은 수십개에서 수천개의 이르는 많은 부품이 장착되므로 방법에 따라서 장착시간의 차가 생기게 되므로 칩마운터 장착 시간을 줄이기 위한 최적화에 관련된 많은 연구가 있었다.[1][2][3][4]

작업에 따라서 다수의 칩마운터로 SMT 인라인 시스템이 구성할 수 있으며, 칩마운터를 운용을 최적화하는 방법은 3가지 부분으로 나누어 생각할 수 있다. 각 칩마운터에 골고루 분배하는 방법, 각 칩마운터에서 부품을 공급하는 피더를 배치하는 방법, 각 칩마운터의 장착 순서를 최적화하는 방법 등이 있다.

본 연구에서는 각 칩마운터에 골고루 작업이 분배되어 있고 피더의 배치 또한 배치되었다는 가정에서 멀티헤드 칩마운터의 부품 장착 순서를 동적계획법(dynamic programming)문제로 모델링하고 동적계획법으로 최적해를를 구하고 결과를 greedy 알고리즘을 적용하여 구한 결과값과 비교 분석한다.

다음은 본 연구에서 대상은 일본 Eicho사의 칩마운터 모델로 멀티헤드 싱글 겐트리형인 S4(그림 1)과 멀티헤드 듀얼 겐트리형인 S4D이다.(그림 2)

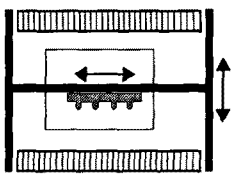


그림 1. S4 모델

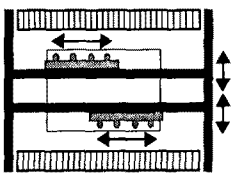
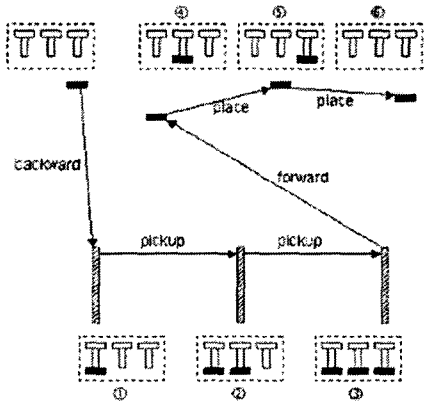


그림 2. S4D 모델



①헤드1 흡착→②헤드2 흡착→③헤드3 흡착
→④헤드1 장착→⑤헤드2 장착→⑥헤드3 장착

그림 3. 조립 사이클 (헤드 수=3 인 경우)

그림 3은 멀티헤드 겐트리형 칩마운터의 부품 조립을 보여준다. 하나의 조립 사이클은 이전 사이클의 마지막 부품 위치에서 현 사이클의 부품 흡착을 위한 피더로 위치로 이동하는 후향이동(backward), 장착을 위한 모든 부품을 흡착하는 흡착이동(pickup), 부품 장착을 위해 처음의 장착점으로 이동하는 전향이동(forward), 각 부품을 장착하는 장착이동(place)으로 이루어져있다.[1]

2. 문제 정의[1]

칩마운터 장착순서를 수학식으로 모델링을 위하여 다음의 기호를 정의한다.

$P=\{1,\cdots,n_P\}$ 장착점의 집합

$H=\{1,\cdots,n_H\}$ 헤드 번호의 집합

$L=\{1,\cdots,n_L\}$ 피더슬롯 번호의 집합

$I=\{1,\cdots,n_I\}$ 사이클 번호의 집합

ρ_g^* 부품군 g의 사이클 수

$\lambda: I \rightarrow G$ 사이클 번호에 대하여 부품군 번호를 지정하는 함수

$l_g^h \in L$ 부품군 g 의 헤드 h 에 할당된 부품 c_g^h 의 피더 슬롯 번호,

$T_i^{place,XY}$ 사이클 i 의 장착시간 중 수평이동시간

피더배치가 이미 끝난 상태에서 장착순서에 영향을 주는 시간은 후향이동시간, 전향이동시간 및 장착시간 중 수평이동시간으로 다음과 같이 정의될 수 있다.

$$T_i^{backward} = \sum_{p \in Pc^{n_H}_{\lambda(i-1)}} XY(n_H, p, 1, l^1_{\lambda(i)})\, x_{(i-1)n_H p} \quad (1)$$

$$T_i^{forward} = \sum_{p \in Pc^1_{\lambda(i)}} XY(n_H, l^{n_H}_{\lambda(i)}, 1, p)\, x_{i1p} \quad (2)$$

$$T_i^{place,XY} = \sum_{h=1}^{n_H-1} \sum_{p_1 \in Pc^h_{\lambda(i)}} \sum_{p_2 \in Pc^{h+1}_{\lambda(i)}} XY(h, p_1, h+1, p_2)\, x_{ihp_1} x_{i(h+1)p_2} \quad (3)$$

식(1)은 후향이동 시간으로서 사이클($i-1$)의 마지막 장착점에서 사이클 i의 처음 흡착점까지의 이동시간이다. 사이클($i-1$)의 마지막 장착점은 부품군 $\lambda(i-1)$에 할당된 부품 $c^{n_H}_{\lambda(i-1)}$의 장착점 집합 $P_{c^{n_H}_{\lambda(i)}}$의 원소이어야 한다. 또한 사이클 i의 처음 흡착점은 부품군 $\lambda(i)$에 할당된 부품의 피더슬롯 $l^1_{\lambda(i)}$의 위치이다. 마찬가지로 식(2)는 전향이동시간이다. 식(3)은 장착시간 중 수평이동시간에 해당한다.

장착순서 최적화 문제는 장착순서 변수의 최적해를 구하는 문제로서 다음과 같다.

$$\min \sum_{i \in I} (T_i^{backward} + T_i^{forward} + T_i^{place,XY}) \quad (4)$$

s.t.

$$\sum_{i \in I} \sum_{h \in H} x_{ihp} = 1, \quad \forall p \in P \quad (5)$$

$$\sum_{h \in H} \sum_{p \in P} x_{ihp} \geq 1, \quad \forall i \in I \quad (6)$$

$$\sum_{p \in P} x_{ihp} \leq 1, \quad \forall (i,h) \in I \times H \quad (7)$$

식(5)는 각 장착점은 단 한번 방문되어야 함을 의미하며, 식(6)은 각 사이클에서 한 개 이상의 장착점을 방문하여야 함을 의미한다. 또한 식(7)은 각 사이클의 각 헤드는 최대 하나의 장착점을 방문할 수 있음을 의미한다.

3. 알고리즘

장착 순서 최적화 문제 (식4)의 해를 구하기 위한 방법을 구현하기가 너무도 복잡하기 때문에 각 사이클 별로 나누어 구현을 용이하게 재정의 하였다.

$$\min \sum_{i \in I} \rho^*_g \cdot (T_i^{backward} + T_i^{forward} + T_i^{place,XY}) \leq \sum_{i \in I} \rho^*_g \cdot \min(T_i^{backward} + T_i^{forward} + T_i^{place,XY}) \quad (8)$$

식 (8)을 식(1),(2),(3)의해서 구해진 각 사이클에 대한 착순서 목적함수는 다음과 같다.

$$\begin{aligned} T_g &= T_g^{backward} + T_g^{place} + T_g^{forward} \\ &= \sum_{p \in Pc^{n_H}_{\lambda(i-1)}} XY(n_H, p, 1, l^1_{\lambda(i)})\, x_{(i-1)n_H p} \\ &+ \sum_{p_1 \in L} \sum_{p_2 \in L} XY(1, p_1, 2, p_2) x_{p_1} x_{p_2} + \cdots \\ &+ \sum_{p_1 \in L} \sum_{p_2 \in L} XY(n_H - 1, p_1, n_H, p_2) x_{p_1} x_{p_2} \\ &+ \sum_{p \in Pc^1_{\lambda(i)}} XY(n_H, l^{n_H}_{\lambda(i)}, 1, p)\, x_{i1p} \end{aligned} \quad (9)$$

이 때 다음의 기호를 정의하면,

$$\begin{aligned} J_{0,1} &\equiv \sum_{p \in Pc^1_{\lambda(i-1)}} XY(n_H, p, 1, l^1_{\lambda(i)})\, x_{(i-1)n_H p} \\ J_{1,2} &\equiv \sum_{p_1 \in L} \sum_{p_2 \in L} XY(1, p_1, 2, p_2) x_{p_1} x_{p_2} \\ &\vdots \\ J_{n_H-1,n_H} &\equiv \sum_{p_1 \in L} \sum_{p_2 \in L} XY(n_H - 1, p_1, n_H, p_2) x_{p_1} x_{p_2} \\ J_{n_H,n_H+1} &\equiv \sum_{p \in Pc^1_{\lambda(i)}} XY(n_H, l^{n_H}_{\lambda(i)}, 1, p)\, x_{i1p} \end{aligned}$$

식(30)의 목적함수는 다음과 같이 표시될 수 있다.

$$\begin{aligned} T_i &= J_{0,1} + J_{1,2} + \cdots + J_{n_H-1,n_H} + J_{n_H,n_H+1} \\ &\equiv J_{0,n_H+1} \end{aligned} \quad (10)$$

위의 최적화 문제는 최적성의 원리(principle of optimality)에 의하여 다음과 같이 최적해 f^*_{0,n_H+1} 를 구할 수 있다.[6]

$$\begin{aligned} f^*_{n_H,n_H+1} &= \min J_{n_H,n_H+1} \\ f^*_{n_H-1,n_H+1} &= \min\{J_{n_H-1,n_H} + f^*_{n_H,n_H+1}\} \\ &\vdots \\ f^*_{1,n_H+1} &= \min\{J_{1,2} + f^*_{2,n_H+1}\} \\ f^*_{0,n_H+1} &= \min\{J_{0,1} + f^*_{1,n_H+1}\} \end{aligned} \quad (11)$$

위의 회귀 방정식(recursive equation)에 의하여 최적해를 찾는 알고리즘이 동적계획법이다. 동적계획법은 근사적 최적해에 머무는 국지탐색법과 달리, 완전 최적해를 구할 수 있다.[6][7]

부품의 장착순서를 정하는데는 피더배치가 선행되어야 하므로 피더가 최적으로 배치되었다는 가정에서 알고리즘을 실행하였다. 본 연구에서는 동적계획법을 이용하여 구한 피더 배치를 적용하였다.[1]

본 논문에서는 시간추정의 편의성을 위하여 동적계획법을 위한 일반적인 사이클과는 다르게 사이클을 재정의하였다. 재정의 된 사이클의 시작점은 헤드가 피더에서 부품을 마지막으로 흡착하는 피더의 위치로 설정하였다. 재정의 된 사이클의 끝점은 다음 사이클의 부품을 장착하기 위해 처음으로 부품을 흡착하는 피더의 위치로 설정하였는데 동일한 사이클을 반복한다는 가정하에 사이클의 부품을 처음으로 흡착하는 피더로 정해졌다. 또한 각 스테이지의 스테이트의 상태정보를 나타내는 구조체를 선언하였다. 각 스테이트의 상태정보를 나타내는 데이터를 이용하여 각 사이클의 최적 장착 순서를 정할 수 있다. 다음은 본 연구에서 제안한 알고리즘이다.

S1. 사용할 부품리스트, 피더 배치, 각 부품군, 생성 등 장착 순서를 구하기 위한 변수들을 초기화한다.
S2. 동적계획법을 이용하여 각 사이클의 경로를 구한다.
S2.1. 동적계획법을 위한 스테이트 상태 구조체를 초기화, 각 스테이지의 사용 부품리스트, 사이클의 사용 head수 등 동적계획법을 위한 변수를 초기화한다.
S2.2. 동적계획법의 종점인 부품을 처음으로 흡착하는 피더에서 마지막 부품 위치로 이동하는데 걸리는 시간을 계산하여 마지막 부품 스테이트 상태 구조체를 작성한다..
S2.3. 부품간의 XY이동시간을 구해 스테이트 상태 구조체을 작성한다.
S2.4. 동적계획법의 시작점인 부품흡착이 마지막으로 이루어지는 피더로부터 첫번째 조립할 부품까지의 이동시간을 계산하여 현재 사이클의 스테이트 상태 구조체을 완성한다.
S2.5. 완성된 스테이트 상태 구조체로 현재 사이클의 경로를 생성한다
S3. 마지막 사이클까지 경로를 구한다

4. 시뮬레이션

제안된 알고리즘은 Microsoft 사의 Visual C++ 로 구현되었으며 IBM-PC 호환 기종 Pentium-III 급 / MS-WindowsXP 환경에서 실행되었다. 제안된 알고리즘은 컴퓨터 시뮬레이션에 의해 검증되었으며 작업시간 추정은 대상 칩마운터의 사양을 기반으로 시간 추정 루틴을 작성하였으므로 실제와 거의 동일하다고 할 수 있다. 실재로 대상 칩마운터의 X축, Y축 및 Z축은 각 각 독립적인 AC 서보 모터에 의하여 구동되며, 모두 S자 가감속 프로파일을 갖는다. 최 저속에서 최 고속까지 5 단계의 속도 프로파일을 갖으며, 부품 별로 특정 단계의 프로파일을 설정하여 사용할 수 있다.

본 논문에서 제안한 동적계획법을 적용하여 구한 부품 장착순서와 greedy algorithm을 이용하여 구한 장착순서를 구한 결과를 컴퓨터 시뮬레이션을 통해서 비교하였다. 기존 알고리즘과 새 알고리즘을 적용하였으며, 전체 조립시간을 비교하였다. 이 때 개선율은 다음과 같이 계산되었다.

$$\text{개선율} = \frac{\text{기존 조립시간} - \text{새 조립시간}}{\text{기존 조립시간}} \times 100(\%)$$

표 1. 대상 보드

Board No	Board Name	부품수	장착점수
1	DisableHead	8	204
2	Aee01881	26	208
3	Test	16	217
4	DemoBoard	21	310
5	LG	57	242
6	CRD8482	26	122
7	HANA-IO	30	234
8	2100top	43	111

표 1은 대상으로 한 PCB 보드의 각종 정보를 보여주고 있다. 대상 PCB는 비교적 부품 창작의 수가 작은 경우로 부품이 200개 내외가 장착된다.

표 2는 대상으로 한 칩마운터 중에서 멀티헤드 싱글 겐트리 타입인 S4모델을 대상으로 장착 순서를 구하고 작업시간을 추정한 결과이다. 비교대상으로 한 알고리즘은 greedy알고리즘으로 동적계획법의 방법으로 구한 결과가 더 좋다.

표 3은 멀티헤드 듀얼 겐트리 타입의 칩마운터인 S4D모델을 동적계획법에 의해서 장착순서를 구한 결과와 greedy알고리즘을 사용한 결과를 비교하는 표이다. 대략 5%의 개선이 있음을 알 수 있다.

표 2. S4모델 장착 순서 최적화 구현 결과

Board	Assembly Time(sec)		Improvement
No	Greedy	DP	(%)
1	131.9	128.9	2.28
2	73.3	64.8	11.60
3	202.0	198.4	1.78
4	192.7	191.2	2.28
5	105.5	100.8	4.46
6	42.5	42.0	1.18
7	78.3	73.8	5.75
8	79.4	78.5	1.13

표 3. S4D모델 장착 순서 구현 결과

Board	Assembly Time(sec)		Improvement
No	Greedy	DP	(%)
1	112.3	107.9	3.92
2	66.3	63.7	3.92
3	147.8	140.2	5.14
4	103.8	102.8	0.96
5	87.4	82.3	5.84
6	40.1	39.2	2.24
7	77.1	72.3	6.23
8	44.9	44.3	1.34

5. 결 론

본 논문에서는 멀티헤드 칩마운터의 부품 장착 동작을 동적계획법에 의해서 모델링하였다. 실제 전자 제품 조립라인의 대부분이 멀티헤드 칩마운터를 사용하지만 부품 장착 동작에 대한 알려진 모델이 많지 않았다.

모델링된 결과로부터 동적계획법을 이용하여 한 사이클의 최적해를 구했다. 하지만 제안된 알고리즘은 한 사이클에 대해서는 최적해를 보장하지만 여러 사이클에 대해서는 최적해를 보장하지 못하는 단점이 있다. 이를 보안하기 위해 동적계획법으로 각 사이클을 구하고 local search의 일종인 2-opt를 사용하여 각 사이클의 배열 순서를 바꾸어 주면 더 나은 결과를 얻을 거라 예상된다.

[참 고 문 헌]

[1]박태형, "동적계획법에 의한 멀티헤드 겐트리형 칩마운터의 피더 배치 최적화", *제어자동화시스템공학회지*, 2002년 6월 게재 예정

[2] 박태형, 김철한, "수송 알고리즘에 의한 칩마운터의 조립순서계획", *제어 자동화 시스템 공학 논문지*, 제6권 제9호, pp.836-843, 2000.

[3] R.Kumar and H.Li, "Integer programming approach to printed circuit board assembly time optimization", *IEEE Trans. on Components, Packaging, and Manufacturing Technology, Part-B:Advanced Packaging*, vol. 18, no. 4, pp.720-727, 1995.

[4] M.O.Ball and M.J.Magazine, "Sequence of insertions in printed circuit board assembly", *Operations Research*, vol.36, no. 2, pp. 192-201, 1988.

[5]S.H.Lee, B.H.Lee and T.H.Park, "A hierarchical method to improve the productivity of a multi-head surface monting machine", *Proc of the 1999 IEEE Int. Conf. on Robotics & Automation*, pp.210-2115, 1999.

[6]Z.Michalewicz and David B. Forgel, "How to Solve it : Modern Heuristics", *PSpringer*

[7]R.Sedgfwick "Algorithms in C++", *Addison-Wesley*