동적계획법에 의한 멀티헤드 겐트리형 칩마운터의 피더배치 최적화

A Dymanic Programming Approach to Feeder Arrangement Optimization for Multihead-Gantry Chip Mounter

박 태 형
(Tae-Hyoung Park)

Abstract : Feeder arrangement is an important element of process planning for printed circuit board assembly systems. This paper newly proposes a feeder arrangement method for multihead-gantry chip mounters. The multihead-gantry chip mounters are very popular in printed circuit board assembly system, but the research has been mainly focused on single-head-gantry chip mounters. We present an integer programming formulation for optimization problem of multihead-gantry chip mounters, and propose a heuristic method to solve the large NP-complete problem in reasonable time. Dynamic programming method is then applied to feeder arrangement optimization to reduce the overall assembly time. Comparative simulation results are finally presented to verify the usefulness of the proposed method.

Keywords : printed circuit board assembly system, chip mounters, optimization methods, dynamic programming

I. 서론

전자조립시스템은 인쇄회로기판(PCB : printed circuit board)에 각종 부품을 조립하여 전자 제품을 생산하는 자동화 시스템이다. 저항, 컨덴서, 트랜지스터 등의 소형 칩 부품부터 각종 IC(integrated circuits) 부품 및 커넥터 등의 대형 정밀부품을 PCB에 조립한다. 전자제품의 소형 경박화 추세에 입각하여 이들 부품은 기존의 삽입형에서 표면실장형(SMD : surface mount device)으로 대체되고 있다. 특히 PCB의 홀에 부품을 삽입하여 조립하는 삽입형 부품과 비교하여, PCB 면 위에 부품을 올려놓아 조립하는 표면실장형 부품은 자동조립이 매우 수월하여 그 점유비율이 크게 증가하고 있다. 표면실장형 부품을 PCB에 조립하는 전자조립시스템을 SMT(surface mounting technology) 인라인(in-line) 시스템[1]이라 한다. 전체의 PCB 조립공정은 PCB 표면에 납 크림을 도포하는 스크린프린터(screen printer) 공정, PCB에 부품을 실장하는 칩마운터(chip mounter) 공정, PCB에 도포된 납 크림을 열경화시키는 리플로우어(reflower) 공정의 세 가지 공정으로 분류된다. 위의 PCB 조립공정 중 생산성 및 수율 측면에서 가장 중요한 공정은 칩마운터 공정으로서, 통상 1-3대의 칩마운터에 의하여 수십-수천 개의 부품을 실장한다. 본 논문은 전자조립시스템의 생산성 향상을 위한 칩마운터의 효과적 운용방법을 제시하고자 한다.

칩마운터는 기본적으로 피더에서 부품을 흡착하여 보드상의 지정된 위치에 부품을 장착하는 메커니즘을 갖으며, 부품을 흡착 및 장착하는 헤드, 헤드를 이동시키는 프레임, 부품을 공급하는 피더, PCB를 이동시키는 반송부 등으로 구성된다. 실장 메커니즘에 따라 여러 가지 형식으로 분리될 수 있으며, 본 논문에서는 그림 1의 멀티헤드 겐트리형 칩마운터를 다루고자 한다. 겐트리형 XY 프레임에 여러 개의 조립헤드가 설치되어 있으며, 각 조립헤드에는 부품의 흡착-장착을 담당하는 공압 노즐이 부착된다. 각 헤드의 노즐은 자동노즐교환기(ANC : auto nozzle changer)에 의하여 작업 중 교환이 가능하다. 칩마운터의 전, 후열에 설치된 피더열에는 여러 개의 피더슬롯이 있으며, 각 피더슬롯에는 부품을 공급하는 피더가 설치된다. 조립헤드는 피더에서 부품을 흡착하고 PCB로 이동하여 부품을 장착하는 pick-and-place의 사이클 운동을 수행한다. 이 때 피더 및 PCB 반송부

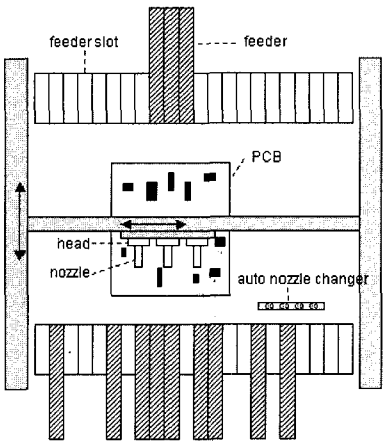


그림 1. 멀티헤드 겐트리형 칩마운터.
Fig. 1. A multihead-gentry chip mounter.

논문접수 : 2001. 11. 3., 채택확정 : 2002. 2. 26.
박태형 : 충북대학교 전기전자 및 컴퓨터 공학부(taehpark@chungbuk.ac.kr)
※ 본 논문은 한국과학재단의 해외 Post-doc. 연수 프로그램의 지원으로 수행하였습니다.

는 조립 작업 중 고정되어 있다. 멀티헤드 겐트리형은 여러 개의 부품을 동시에 흡착하고 이송할 수 있으므로, 조립헤드 수가 하나인 단일헤드 겐트리형에 비하여 조립시간을 크게 단축시킬 수 있다[2]. Yamaha사의 YV100, 180 시리즈, Juki사의 KE750, 760 시리즈, 삼성테크윈(주)의 CP40, 50 시리즈, 미래산업(주)의 MPS1020 시리즈 등이 대표적인 멀티헤드 겐트리형 칩마운터 모델들로서, 전자조립라인에서 매우 광범위하게 사용되고 있다.

칩마운터의 최적화 문제는 조립시간을 최소화하는 피더배치 및 장착순서를 결정하는 문제이다. 이 문제는 매우 복잡한 NP-complete의 최적화 문제로서, 합리적 시간 내에 최적해를 구하는 알고리즘의 구현이 매우 어렵다고 알려져 있다[3]. 따라서 발견적 접근에 의하여 근사적 최적해를 합리적 시간 내에 구하고자 하는 방법이 시도되고 있으며, 일반적으로 전체 문제를 피더배치 단계, 장착순서 단계 등 몇가지의 단계로 분리하여 해결하고 있다[2][4]. 특히 칩마운터의 운동 메커니즘에 따라 적용되는 알고리즘이 상이하다.

Sohn 등[5]은 터릿형 프레임을 갖는 로터리형 칩마운터에 대한 최적화 알고리즘을 제시하였다. 회전운동을 하는 여러 개의 조립헤드와, 직선운동을 하는 PCB 베이스 및 피더 베이스를 고려하여, 장착순서 및 피더배치에 대한 발견적 기법을 적용하였다. Ball 등[6]은 단일헤드 겐트리형 칩마운터에 적용될 수 있는 pick-and-place 기기의 장착순서 최적화 방법을 제시하였다. rural postman problem으로 모델링 하였으며, 근사적 최적해를 위한 발견적 알고리즘을 제안하였다. 박태형 등[7]은 동일한 문제를 수송문제로 모델링 하였으며, 할당 알고리즘을 적용하는 휴리스틱스를 제안하였다. Kumar 등[8]은 단일헤드 겐트리형 칩마운터의 최적화 문제를 정수계획문제로 수식화 하였으며, 피더배치 문제와 장착순서 문제를 분리하여 모델링 하였다. 피더배치 문제는 minimum weighted matching 문제로, 장착순서는 표준 TSP 문제로 변환하였다. Leu 등[9]은 로터리 형 및 단일헤드 겐트리형 등에 적용할 수 있는 유전자 알고리즘을 제시하였다. 최근에는 기존의 로터리형 및 겐트리형을 변형한 메커니즘의 칩마운터가 등장하고 있으며, 이에 적용하기 위한 최적화 방법이 제시되고 있다. Wang 등[10]은 멀티스테이션 칩마운터, Tirpak 등[11]은 회전하는 레졸버헤드를 갖는 겐트리형 칩마운터를 위한 최적화 방법을 각각 발표하였다.

멀티헤드 겐트리형의 최적화 방법을 다룬 연구 결과는 매우 드물다. 다른 형식의 칩마운터와 달리 전체 문제를 부품군 문제, 피더 배치 문제 및 장착순서 문제의 3단계로 분할하여 해를 구하는 방법이 사용되고 있다. Lee 등[12]은 멀티헤드 겐트리형 칩마운터를 대상으로 부품 군, 피더배치 및 장착순서의 최적화 방법을 분리하여 제안하였다. 직관적이고 간단한 방법을 부품 군 및 피더배치 문제에 적용하였으며, TSP 알고리즘을 장착순서 문제에 적용하였다. 칩마운터의 최적화에 있어서, 피더배치 최적화의 비중이 매우 높다. Grotzinger[13]는 두 개의 겐트리를 갖는 칩마운터의 피더 배치 모델을 제시하였으며, DePuy 등[14]은 하나의 부품에 여러 개의 피더를 배치하는 것이 조립시간 단축에 유리하다는 실험적 결과를 보고하였다.

본 논문은 멀티헤드 겐트리형 칩마운터의 조립시간 단축을 위한 피더배치 방법을 새로이 제시한다. 전체 칩마운터 최적화 문제를 정수계획문제로 모델링 하며, 부품군 문제, 피더배치 문제 및 장착순서 문제를 분리하여 구성하고 그 해법을 논한다. 특히 최적성이 우수한 최적화 방법의 하나인 동적계획법을 피더배치 문제에 처음으로 적용하며, 피더배치의 개선을 통한 전체 조립시간의 단축을 도모한다. 모의 칩마운터를 대상으로 비교 시뮬레이션을 수행하여 제안된 방법의 유용성을 검증한다.

II. 칩마운터의 최적화 문제

그림 2는 멀티헤드 겐트리형 칩마운터의 조립 사이클을 보여준다. 이전 사이클의 마지막 장착 위치로부터 현 사이클의 처음 피더 위치로 후향이동 후, 헤드별로 부품을 흡착한다. 이 때 부품의 흡착은 헤드 번호순으로 수행된다. 부품 흡착 후 처음 장착 위치로 전향이동 후, 헤드별로 부품을 장착한다. 부품의 장착 또한 헤드 번호순으로 이루어진다고 가정한다. 후향이동(backward)-흡착이동(pickup)-전향이동(forward)-장착이동(place)의 사이클을 반복적으로 수행하며 PCB의 모든 장착점에 부품을 실장한다. 헤드에 부착된 노즐의 교환이 필요한 경우, 후향이동 시 자동노즐교환기로 이동하여 노즐을 교환한다. 피더레인에는 각 부품 별로 하나의 피더만 배치된다고 가정한다.

멀티헤드 겐트리형 칩마운터의 최적화 문제는 전체 조립 사이클에 소요되는 시간을 최소화시키는 부품 별 피더의 배치와 장착점 방문 순서를 결정하는 문제이다. 이 문제를 수학적으로 정의하기 위하여 다음의 기호들을 정의한다.

$P=\{1,\cdots,n_P\}$ 장착점의 집합

$C=\{1,\cdots,n_C\}$ 부품의 집합 ($n_C \le n_P$)

$H=\{1,\cdots,n_H\}$ 헤드 번호의 집합

$L=\{1,\cdots,n_L\}$ 피더슬롯 번호의 집합

$I=\{1,\cdots,n_I\}$ 사이클 번호의 집합

$\tau: P \to C$ 장착점이 소속된 부품을 지정하는 함수

$P_c=\{p \mid \tau(p)=c, \forall p \in P\}$ 부품 $c \in C$에 소속된 장착점의 집합

z_{lc} 피더배치 변수

$$=\begin{cases}1, & \text{슬롯 } l \in L \text{에 부품 } c \in C \text{의 피더 배치}\\ 0, & \text{나머지 경우}\end{cases}$$

x_{ihp} 장착순서 변수

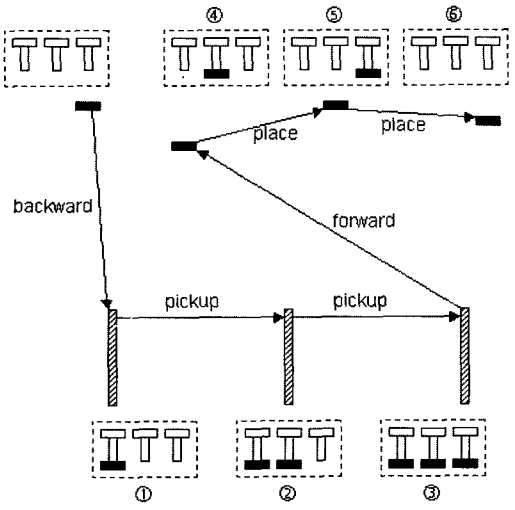
$$=\begin{cases}1, & \text{사이클 } i \in I \text{에 헤드 } h \in H \text{가 장착점 } p \in P \text{에 부품 장착}\\ 0, & \text{나머지 경우}\end{cases}$$

T_i^{anc} 사이클 i의 노즐교환시간

T_i^{backward} 사이클 i의 후향이동시간

사이클 $(i-1)$의 마지막 장착 위치에서 사이클 i의 처음 흡착 위치로의 수평이동시간

T_i^{pickup} 사이클 i의 흡착시간



① 헤드 1 흡착 → ② 헤드 2 흡착 → ③ 헤드 3 흡착 →
④ 헤드 1 장착 → ⑤ 헤드 2 장착 → ⑥ 헤드 3 장착

그림 2. 조립 사이클 (헤드 수 = 3인 경우).

Fig. 2. Assembly sequence (total heads = 3).

T_i^{forward} 사이클 i의 전향이동시간

사이클 i의 마지막 흡착 위치에서 사이클 i의 처음 장착 위치로의 수평이동시간

T_i^{place} 사이클 i의 장착시간

$XY(h_1, loc_1, h_2, loc_2)$ 헤드 h_1의 위치 loc_1에서 헤드 h_2의 위치 loc_2로의 수평이동시간

$ANC(h)$ 헤드 h에 부착된 노즐의 노즐교환시간

$\alpha(c_1, c_2)$ 부품 c_1과 부품 c_2의 노즐이 서로 다른지의 여부를 판별하는 함수

$$= \begin{cases} 1, & \text{부품 } c_1\text{과 부품 } c_2\text{의 노즐이 다를 때} \\ 0, & \text{부품 } c_1\text{과 부품 } c_2\text{의 노즐이 같을 때} \end{cases}$$

각 사이클에서 소요되는 시간들은 다음과 같이 표현될 수 있다.

$$T_i^{\text{anc}} = \sum_{h=1}^{n_H} \sum_{p_1 \in P} \sum_{p_2 \in P} \alpha(\tau(p_1), \tau(p_2)) ANC(h) \; x_{(i-1)hp_1} x_{ihp_2} \tag{1}$$

$$T_i^{\text{backward}} = \sum_{p_1 \in P} \sum_{p_2 \in P} \sum_{l \in L} XY(n_H, p_1, 1, l) \; x_{(i-1)n_H p_1} x_{i1p_2} z_{l\tau(p_2)} \tag{2}$$

$$T_i^{\text{pickup}} = \sum_{h=1}^{n_H-1} \sum_{p_1 \in P} \sum_{p_2 \in P} \sum_{l_1 \in L} \sum_{l_2 \in L} XY(h, l_1, h+1, l_2) \; x_{ihp_1} z_{l_1\tau(p_1)} x_{i(h+1)p_2} z_{l_2\tau(p_2)} \tag{3}$$

$$T_i^{\text{forward}} = \sum_{p_1 \in P} \sum_{p_2 \in P} \sum_{l \in L} XY(n_H, l, 1, p_1) \; x_{i1p_1} x_{in_H p_2} z_{l\tau(p_2)} \tag{4}$$

$$T_i^{\text{place}} = \sum_{h=1}^{n_H-1} \sum_{p_1 \in P} \sum_{p_2 \in P} XY(h, p_1, h+1, p_2) \; x_{ihp_1} x_{i(h+1)p_2} \tag{5}$$

간결한 식 표현을 위하여, (2) 및 (4)와 같이 모든 사이클에서 n_H개의 헤드가 모두 사용된다고 가정한다. (3)의 흡착시간은 (5)의 장착시간은 수평이동시간만을 표시한다. 실제로는 수직이동시간, 지연시간 등이 추가되어야 하나 이들 시간은 피더배치 및 장착순서와 무관하게 일정하다고 가정한다. 한편 시간 계산에 사용되는 XY, ANC, α, τ 등의 함수값은 미리 계산되어 저장될 수 있으며, 상수로서 취급될 수 있다.

멀티헤드 겐트리형 칩마운터의 최적화 문제는 전체 사이클 시간을 최소화시키는 피더배치 변수 $[z_{lc}]^{n_L \times n_C}$ 및 장착순서 변수 $[x_{ihp}]^{n_I \times n_H \times n_P}$를 구하는 문제로서, 다음과 같은 정수계획문제로 정의된다.

$$\min \sum_{i \in I} \left(T_i^{\text{anc}} + T_i^{\text{backward}} + T_i^{\text{pickup}} + T_i^{\text{forward}} + T_i^{\text{place}} \right) \tag{6}$$

s.t.

$$\sum_{l \in L} z_{lc} = 1 \;, \quad \forall c \in C \tag{7}$$

$$\sum_{c \in C} z_{lc} \le 1 \;, \quad \forall l \in L \tag{8}$$

$$\sum_{i \in I} \sum_{h \in H} x_{ihp} = 1 \;, \quad \forall p \in P \tag{9}$$

$$\sum_{h \in H} \sum_{p \in P} x_{ihp} \ge 1 \;, \quad \forall i \in I \tag{10}$$

$$\sum_{p \in P} x_{ihp} \le 1 \;, \quad \forall (i,h) \in I \times H \tag{11}$$

(7)은 각 부품 당 하나의 피더가 배치되어야 함을 의미하며, (8)은 하나의 슬롯에는 피더가 중복되어 배치될 수 없음을 의미한다. (9)는 각 장착점은 단 한번 방문되어야 함을 의미하며, (10)은 각 사이클에서는 한 개 이상의 장착점을 방문하여야 함을 의미한다. 또한 (11)은 각 사이클의 각 헤드는 최대 하나의 장착점을 방문할 수 있음을 의미한다.

III. 최적화 알고리즘의 구성

앞장에서 정의된 멀티헤드 겐트리형 칩마운터의 최적화 문제는 매우 복잡한 NP-complete의 문제에 해당하며, 그 최적해를 구하는 것이 매우 어렵다. 따라서 몇 개의 부분 문제로 분할하고, 각 문제의 해로부터 적절한 해를 구하는 발견적 접근이 시도될 수 있다. 단일헤드 겐트리형 칩마운터의 경우 전체 문제를 피더배치 최적화 문제와 장착순서 최적화 문제의 두 가지 문제로 구성하고, 최적화 알고리즘을 계층

적으로 구성하는 방법이 사용되었다[8].

그림 3은 본 논문에서 제시하는 멀티헤드 겐트리형 칩마운터의 최적화 알고리즘 구성도이다. 단일헤드 겐트리형 칩마운터의 경우와 달리 부품ㅊ군 최적화, 피더배치 최적화 및 장착순서 최적화의 3단계를 계층적으로 구성한다. 최상위의 부품군 최적화는 부품군 변수를 결정하는 단계이며, 피더배치 최적화는 이 부품군 변수로부터 피더배치 변수를 결정하는 단계이다. 마지막 장착순서 최적화는 부품군 변수와 피더배치 변수로부터 장착순서 변수를 결정하는 단계이다. 별도의 부품군 최적화 단계가 필요한 이유는, 멀티헤드 겐트리형의 피더배치 및 장착순서 결정을 보다 수월하게 하기 위함이다.

각 최적화 단계에서 설정되는 문제와 해를 구하는 방법은 다음과 같다.

1. 부품군 최적화

멀티헤드 겐트리형 칩마운터는 여러 개의 헤드가 함께 이동하며 흡착 및 장착을 수행한다. 이 때 함께 이동하는 부품들의 헤드 별 순서 리스트를 부품군이라 정의한다. 즉 어느 사이클에서 헤드 i ($i=1,\cdots,n_H$)가 부품 c_i에 대한 흡착 및 장착을 수행하는 경우, 그 사이클에서 함께 이동하는 부품의 리스트$\langle c_1,\cdots,c_{n_H}\rangle$이 하나의 부품군이 된다. 이 때 동일한 부품군이 여러 개의 사이클에 걸쳐서 존재할 수 있다.

부품군 최적화 문제는 전체 조립시간이 최소화되는 부품군들을 결정하는 문제로서, 수학적 정의를 위하여 다음의 기호를 정의한다.

$G=\{1,\cdots,n_G\}$	부품군 번호의 집합
y_{ghc}	부품군 변수, $=\begin{cases}1, & \text{부품군 } g\text{에서 헤드 } h \text{ 가 부품 } c\text{를 흡착 및 장착}\\ 0, & \text{나머지 경우}\end{cases}$
ρ_g	부품군 g의 사이클 수, $\sum_{g\in G}\rho_g=n_I$
$\hat{T}_g^{\text{anc}}$	부품군 g의 노즐교환 시간 추정치
$\hat{T}_g^{\text{backward}}$	부품군 g의 후향이동시간 추정치
$\hat{T}_g^{\text{pickup}}$	부품군 g의 흡착시간 추정치
$\hat{T}_g^{\text{forward}}$	부품군 g의 전향이동시간 추정치
$\hat{T}_g^{\text{place}}$	부품군 g의 장착시간 추정치
$\bar{c}$	부품 c의 부품 중점 ($p\in P_c$인 모든 장착점들의 중심 위치)
$\bar{l}$	피더열의 중심 위치

부품군의 사이클에서 소요되는 시간은 피더배치와 장착순서가 결정되지 않은 상태에서 추정되어야 한다. 부품군 g의 사이클에서 소요되는 시간들을 다음과 같이 정의한다.

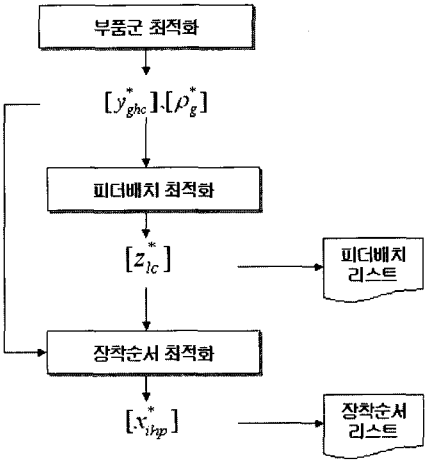


그림 3. 최적화 알고리즘의 구성.

Fig. 3. Structure of optimization algorithm.

$$\hat{T}_g^{\text{anc}}=\sum_{h=1}^{n_H}\sum_{c_1\in C}\sum_{c_2\in C}\alpha(c_1,c_2)\,ANC(h)\,y_{(g-1)hc_1}y_{ghc_2} \tag{12}$$

$$\hat{T}_g^{\text{backward}}=\sum_{c\in C}XY(n_H,\bar{c},1,\bar{l})\,y_{gn_Hc} \tag{13}$$

$$\hat{T}_g^{\text{pickup}}=0 \tag{14}$$

$$\hat{T}_g^{\text{forward}}=\sum_{c\in C}XY(n_H,\bar{l},1,\bar{c})\,y_{g1c} \tag{15}$$

$$\hat{T}_g^{\text{place}}=\sum_{h=1}^{n_H-1}\sum_{c_1\in C}\sum_{c_2\in C}XY(h,\bar{c_1},h+1,\bar{c_2})\,y_{ghc_1}y_{g(h+1)c_2} \tag{16}$$

(12)의 노즐교환 시간 추정치는 피더배치 및 장착순서와 관계없이 결정될 수 있으며, (1)의 실제 노즐교환시간과 같다. 후향이동시간은 마지막 장착점에서 다음 사이클의 처음 흡착점까지의 이동시간으로서, 장착순서와 피더배치가 결정된 이후에 정확한 계산이 가능하다. 그러나 부품군 데이터로만 그 값을 추정하기 위하여, (13)과와 같이 마지막 헤드의 부품 중점에서 피더열 중점으로의 수평이동시간으로 추정한다. 전향이동시간의 경우도 마찬가지로 (15)와 같이, 피더열 중점에서 처음 헤드 부품의 중점으로의 수평이동시간으로 추정한다. 흡착시간은 (3)과 같이 피더열에서의 수평이동시간이다. 그러나 피더배치를 알지 못하므로, (15)와 같이 0으로 추정한다. 장착시간은 (5)와 같이 장착점 사이의 수평이동시간이다. 장착순서가 결정되지 않은 상태이므로, 장착점간의 수평이동시간을 (16)과 같이 부품 중점 사이의 수평이동시간으로 대체하여 추정한다.

부품군 최적화 문제는 전체 사이클 시간의 추정치를 최소화하는 부품군 변수 $[y^*_{ghc}]^{n_G\times n_H\times n_C}$ 및 부품군의 사이클 수 $[\rho^*_g]^{n_G}$를 구하는 문제로서, 다음과 같이 정의된다.

$$\min \sum_{g\in G} (\hat{T}_g^{\text{anc}} + \rho_g \cdot (\hat{T}_g^{\text{backward}} + \hat{T}_g^{\text{pickup}} + \hat{T}_g^{\text{forward}} + \hat{T}_g^{\text{place}})) \quad (17)$$

s.t.

$$\sum_{g\in G}\sum_{h\in H}(y_{ghc}\cdot\rho_g) = |P_c| \quad , \forall c \in C \quad (18)$$

$$\sum_{h\in H}\sum_{c\in C} y_{ghc} \geq 1 \quad , \forall g \in G \quad (19)$$

$$\sum_{c\in C} y_{ghc} \leq 1 \quad , \forall (g,h) \in G\times H \quad (20)$$

(18)은 부품군에 속한 어떤 부품에 대하여, 사이클 수의 합은 그 부품에 속한 장착점의 수와 동일하여야 함을 의미한다. (19)는 모든 부품군에는 하나 이상의 부품이 존재하여야 함을 의미하며, (20)은 각 부품군의 각 헤드는 최대 하나의 부품을 흡착-장착 할 수 있음을 의미한다. 부품군 최적화 단계에서는 노즐교환 횟수와 전체 사이클 수를 직접 결정한다. 또한 후향이동시간, 전향이동시간 및 장착시간 등을 간접적으로 결정하며, 부품군 변수는 피더배치 최적화 및 장착순서 최적화에서 사용된다.

부품군 문제를 위하여, 복잡한 조합 최적화 문제에 일반적으로 많이 적용되는 국지탐색법(local search)이 적용될 수 있다. 즉 hill-climbing, simulated annealing 및 genetic search 등 근사적 최적해를 구하기 위한 방법들[15]을 적용할 수 있다. 예를 들어 hill-climbing 방법은, 적절한 초기해를 구하고, 주변 탐색에 의한 개선을 통하여 근사적 최적해에 도달시키는 방법이며, 이를 부품군 최적화 문제에 적용하는 경우 다음과 같이 할 수 있다.

우선 초기 부품군을 생성하기 위하여 우선 헤드 별 노즐의 배치를 구하고, 이로부터 부품을 각 헤드에 할당한다. 이때 전체 사이클 수가 최소화 되도록 하여야 하며, 이를 위하여 가능한 많은 수의 부품이 하나의 부품군에 속하도록 한다. 초기 부품군 생성이 완료되면, (17)의 목적함수를 계산하고, 이 값을 줄이기 위하여 서로 다른 부품군 및 헤드에 할당된 두 부품을 상호 교환한다. 이 때 새로운 부품군이 (18)-(20)의 제한조건을 만족하는 지를 검사한다. 부품의 교환으로 이루어진 새 부품군의 목적함수 값이 감소한 경우, 교환을 확정하고, 그렇지 않은 경우 다른 부품과 교환을 진행한다. 목적함수 값이 더 이상 감소하지 않을 때까지 부품의 교환을 진행하여, 근사적 최적해에 도달한다.

2. 피더배치 최적화

피더배치 최적화 문제는 전체 사이클 시간을 최소화 하는 피더배치 변수를 구하는 문제로서, 각 부품의 피더를 배치할 슬롯 번호를 결정하는 것이다. 이 때 부품군 최적화 단계에서 결정된 부품군 변수 $[y^*_{ghc}]^{n_G\times n_H\times n_C}$ 및 사이클 수 ρ^*_g를 사용한다. 피더배치 최적화 문제의 수학적 구성을 위하여 다음의 기호를 정의한다.

$\hat{T}_g^{\text{backward}}$ 부품군 g의 후향이동시간 추정치

$\hat{T}_g^{\text{pickup}}$ 부품군 g의 흡착시간 추정치

$\hat{T}_g^{\text{forward}}$ 부품군 g의 전향이동시간 추정치

$c_g^h \in C$ 부품군 g의 헤드 h에 할당된 부품, $y^*_{ghc_g^h}=1$

부품군 변수가 결정된 상태에서, 피더배치에 영향을 받는 사이클 시간은 후향이동시간, 흡착시간 및 전향이동시간이다. 부품군 g의 사이클에서 소요되는 시간 중, 피더배치의 결정에 영향을 주는 시간들은 다음과 같이 정의된다.

$$\hat{T}_g^{\text{backward}} = \sum_{l\in L} XY(n_H, \overline{c_g^l}, 1, l) z_{lc_g^1} \quad (21)$$

$$\hat{T}_g^{\text{pickup}} = \sum_{h=1}^{n_H-1}\sum_{l_1\in L}\sum_{l_2\in L} XY(h, l_1, h+1, l_2) z_{l_1c_g^h} z_{l_2c_g^{h+1}} \quad (22)$$

$$\hat{T}_g^{\text{forward}} = \sum_{l\in L} XY(n_H, l, 1, \overline{c_g^1}) z_{lc_g^{n_H}} \quad (23)$$

(21)의 후향이동시간은 부품군 g의 마지막 헤드 부품 $c_g^{n_H}$의 장착 중점에서 처음 헤드 부품 c_g^1의 흡착점으로의 이동시간이다. (22)의 흡착시간은 (3)의 실제 흡착시간과 동일하다. (23)의 전향이동시간은 부품군 g의 마지막 헤드 부품 $c_g^{n_H}$의 흡착점에서 처음 헤드 부품 c_g^1의 부품 중점으로의 이동시간이다.

피더배치 최적화 문제는 전체 사이클 시간의 추정치를 최소화 하는 피더배치 변수 $[z^*_{lc}]^{n_L\times n_C}$를 구하는 문제로서, 다음과 같다.

$$\min \sum_{g\in G} (\hat{T}_g^{\text{backward}} + \hat{T}_g^{\text{pickup}} + \hat{T}_g^{\text{forward}}) \cdot \rho^*_g \quad (24)$$

s.t. (7), (8)

정의된 문제에 대하여 여러 가지 국지탐색법을 적용하여 그 해를 구할 수 있다. 그러나 본 논문은 동적계획법을 사용한 발견적 기법을 적용하고자 하며, 그 내용은 IV장에서 논한다.

3. 장착순서 최적화

장착순서 최적화 문제는 전체 사이클 시간을 최소화 하는 장착순서 변수를 구하는 문제로서, 각 사이클의 각 헤드에 장착점을 할당하는 것이다. 이 때 부품군 최적화 단계에서 결정된 부품군 변수 $[y^*_{ghc}]^{n_G\times n_H\times n_C}$ 및 사이클 수 ρ^*_g, 피더배치 최적화 단계에서 결정된 피더배치 변수 $[z^*_{lc}]^{n_L\times n_C}$를 사용한다. 수학적 구성을 위하여 다음 기호를 정의한다.

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

$l_g^h \in L$ 부품군 g의 헤드 h에 할당된 부품 c_g^h의 피더 슬롯 번호,
$y^*_{ghc_g^h}=1, z^*_{l_g^h c_g^h}=1$

$T_i^{\text{place},XY}$ 사이클 i의 장착시간 중 수평이동시간

부품군 및 피더배치가 이미 결정된 상태에서, 장착순서에 별도로 영향을 받는 사이클 시간은, 후향이동시간(2), 전향이동시간(4) 및 장착시간(5) 중 수평이동시간이다. 이들 시간은 다음과 같이 다시 정의될 수 있다.

$$T_i^{\text{backward}} = \sum_{p \in Pc^{n_H}_{\lambda(i-1)}} XY(n_H, p, 1, l^1_{\lambda(i)})\, x_{(i-1)n_H p} \tag{25}$$

$$T_i^{\text{forward}} = \sum_{p \in Pc^{1}_{\lambda(i)}} XY(n_H, l^{n_H}_{\lambda(i)}, 1, p)\, x_{i1p} \tag{26}$$

$$T_i^{\text{place},XY} = \sum_{h=1}^{n_H-1} \sum_{p_1 \in Pc^{h}_{\lambda(i)}} \sum_{p_2 \in Pc^{h+1}_{\lambda(i)}} XY(h, p_1, h+1, p_2)\, x_{ihp_1} x_{i(h+1)p_2} \tag{27}$$

(25)는 후향이동시간으로서 (2)의 다른 표현이며, 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 (26)은 전향이동시간이며 (4)의 다른 표현이다. (27)은 (5)의 장착시간 중 수평이동시간에 해당한다.

장착순서 최적화 문제는 장착순서 변수의 최적해 $[x^*_{ihp}]^{n_I \times n_H \times n_P}$ 를 구하는 문제로서 다음과 같다.

$$\min \sum_{i \in I} (T_i^{\text{backward}} + T_i^{\text{forward}} + T_i^{\text{place},XY}) \tag{28}$$

s.t. (10),(11),(12)

장착순서 문제는 각 사이클 번호 및 헤드 번호에 장착점을 할당시키는 문제이다. 사이클 번호와 헤드 번호를 결정하면 자동적으로 방문 순서가 결정되므로, 이 문제는 모든 장착점을 한 번씩 방문하는 방문 순서를 결정하는 TSP(traveling salesman problem)의 범주에 해당한다. 다만 (28)의 목적함수를 사용하여야 하며, 시작 위치 및 종료 위치는 지정된 헤드 대기위치로 고정된다. 따라서 TSP에 적용되는 여러 가지 알고리즘의 적용이 가능하다. TSP는 대표적인 NP-complete 의 최적화 문제로서 근사 최적해를 구하는 여러 가지 방법이 알려져 있다[17]. 예를 들어 nearest neighbor 방법에 의하여 초기 장착 순서를 구성하고, 2-opt 방법에 의하여 순서를 개선하는 방법은 매우 실용적인 방법으로서, TSP 범주에 속하는 여러 가지 문제에 용이하게 적용될 수 있다.

원 문제를 부품군 문제, 피더배치 문제 및 장착순서 문제로 분리하여 구성하였다. 새로이 구성된 각각의 문제 또한 최적해를 구하기 매우 어려운 문제이다. 그러나 결정 변수 사이의 결합성이 완화된 형태로 존재하며, 따라서 근사 최적해를 구하는 것이 상대적으로 용이하다.

Ⅳ. 피더배치 알고리즘

피더배치 최적화 문제(24)의 해를 구하기 위해서는, 모든 부품군에 대한 피더배치를 한 번에 구하여야 하나, 그 방법을 구현하는 것이 용이하지 않다. 반면에 각 부품군에 대한 피더의 배치를 구하는 것은 용이하게 구현될 수 있다. 다음의 관계식에 의하여,

$$\begin{aligned} &\min \sum_{g \in G} \rho^*_g \cdot (\hat{T}_g^{\text{backward}} + \hat{T}_g^{\text{pickup}} + \hat{T}_g^{\text{forward}}) \\ &\le \sum_{g \in G} \rho^*_g \cdot \min(\hat{T}_g^{\text{backward}} + \hat{T}_g^{\text{pickup}} + \hat{T}_g^{\text{forward}}) \end{aligned} \tag{29}$$

각 부품군 $g \in G$에 대하여 사이클 시간 추정치 $\hat{T}_g^{\text{backward}} + \hat{T}_g^{\text{pickup}} + \hat{T}_g^{\text{forward}}$를 최소화하는 피더배치를 순차적으로 구하여, 이로부터 전체 부품에 대한 피더배치를 구할 수 있다. 이 때 사이클 수 ρ^*_g가 큰 부품군에 대한 피더배치를 우선적으로 수행하면, 전향이동시간 및 후향이동시간을 전체적으로 감소시킬 수 있어서 최적해에 보다 근접시킬 수 있다. 본 논문은 사이클 수가 큰 부품군을 우선 선택하여, 선택된 부품군에 대한 피더배치를 동적계획법으로 구하는, 즉 동적계획법을 사용한 발견적 기법을 제안한다.

(21),(22),(23)에 의하여, 어떤 부품군 g에 대한 피더배치의 목적함수는 다음과 같다.

$$\begin{aligned} \hat{T}_g &= \hat{T}_g^{\text{backward}} + \hat{T}_g^{\text{pickup}} + \hat{T}_g^{\text{forward}} \\ &= \sum_{l \in L} XY(n_H, \overline{c_g^{n_H}}, 1, l) z_{lc_g^1} \\ &+ \sum_{l_1 \in L} \sum_{l_2 \in L} XY(1, l_1, 2, l_2) z_{l_1 c_g^1} z_{l_2 c_g^2} + \cdots \\ &+ \sum_{l_1 \in L} \sum_{l_2 \in L} XY(n_H - 1, l_1, n_H, l_2) z_{l_1 c_g^{n_H-1}} z_{l_2 c_g^{n_H}} \\ &+ \sum_{l \in L} XY(n_H, l, 1, \overline{c_g^1}) z_{lc_g^{n_H}} \end{aligned} \tag{30}$$

이 때 다음의 기호를 정의하면,

$$\begin{aligned} J_{0,1} &\equiv \sum_{l \in L} XY(n_H, \overline{c_g^{n_H}}, 1, l) z_{lc_g^1} \\ J_{1,2} &\equiv \sum_{l_1 \in L} \sum_{l_2 \in L} XY(1, l_1, 2, l_2) z_{l_1 c_g^1} z_{l_2 c_g^2} \\ &\vdots \\ J_{n_H-1, n_H} &\equiv \sum_{l_1 \in L} \sum_{l_2 \in L} XY(n_H - 1, l_1, n_H, l_2) z_{l_1 c_g^{n_H-1}} z_{l_2 c_g^{n_H}} \\ J_{n_H, n_H+1} &\equiv \sum_{l \in L} XY(n_H, l, 1, \overline{c_g^1}) z_{lc_g^{n_H}} \end{aligned}$$

(30)의 목적함수는 다음과 같이 표시될 수 있다.

$$\begin{aligned} \hat{T}_g &= J_{0,1} + J_{1,2} + \cdots + J_{n_H-1, n_H} + J_{n_H, n_H+1} \\ &\equiv J_{0, n_H+1} \end{aligned} \tag{31}$$

위의 최적화 문제는 최적성의 원리(principle of optimality) [16]에 의하여 다음과 같이 최적해 f^*_{0,n_H+1}를 구할 수 있다.

$$\begin{aligned} f^*_{n_H,n_H+1} &= \min J_{n_H,n_H+1} \\ f^*_{n_H-1,n_H+1} &= \min\{J_{n_H-1,n_H} + f^*_{n_H,n_H+1}\} \\ &\vdots \\ f^*_{1,n_H+1} &= \min\{J_{1,2} + f^*_{2,n_H+1}\} \\ f^*_{0,n_H+1} &= \min\{J_{0,1} + f^*_{1,n_H+1}\} \end{aligned} \tag{32}$$

위의 회귀 방정식(recursive equation)에 의하여 최적해를 찾는 알고리즘이 동적계획법이다. 동적계획법은 근사적 최적해에 머무는 국지탐색법과 달리, 완전최적해를 구할 수 있는 열거탐색법(enumerative search)의 범주에 속한다. 국지탐색법은 초기해에 따라 해의 성능이 달라질 수 있으나, 동적계획법은 최적성의 원리에 의하여 최적해를 보장한다. 그림 5는 동적계획법의 적용을 위한 탐색영역을 보여준다. 가로의 헤드 축과 세로의 슬롯 축으로 구성되는 2차원 평면이며, 헤드 축은 0부터 n_H+1까지의 열과 슬롯 축은 0부터 n_L까지의 행으로 구성된다.이 때 0 열과 n_H+1열은 각각 1 개의 노드를 갖으며, 나머지 열들은 모두 n_L개의 노드를 갖는다. 따라서 전체 노드의 수는 $n_H \cdot n_L+2$개이다. 각 노드는 헤드의 번호 및 그 헤드의 위치를 나타내며, i열 j행의 노드 (i,j)에 대응하는 $\langle head(i), slot(j)\rangle$는 다음과 같이 정의된다.

$$\langle head(i), slot(j)\rangle = \begin{cases} \langle n_H, \text{장착중점 } \overline{c_g^J}\rangle & , i=0, j=0 \\ \langle i, \text{피더슬롯 } j\rangle & , i=1,\cdots,n_H, j=1,\cdots,n_L \\ \langle 1, \text{장착중점 } \overline{c_g^1}\rangle & , i=n_H+1, j=0 \end{cases} \tag{33}$$

이 때 노드(i,j)에서 노드$(i+1,k)$로 이동하는 데 소요되는 시간 $inccost(i,j,k)$은 다음과 같다.

$$inccost(i,j,k) = XY(head(i), slot(j), head(i+1), slot(k)) \tag{34}$$

알고리즘 구현을 위하여 다음의 데이터를 정의한다.

comp[*h*] 헤드 h의 부품, $h=1,\cdots,n_H$

feeder[*l*] 슬롯 l에 피더가 배치된 부품, $l=1,\cdots,n_L$

status[*l*] 슬롯 l의 상태 (available, arranged), $l=1,\cdots,n_L$

cost[*i*][*j*] 노드 (i,j)에서 마지막 노드 $(n_H+1,0)$까지 소요되는 최소 시간 $i=1,\cdots,n_H, j=1,\cdots,n_L$

*cost** 처음 노드$(0,0)$에서 마지막 노드$(n_H+1,0)$까지 소요되는 최소 시간

next[*i*][*j*] 최소시간을 위하여 노드(i,j)에 연결되는 $i+1$열의 행 번호, $i=1,\cdots,n_H, j=1,\cdots,n_L$

*next** 최소시간을 위하여 처음 노드 $(0,0)$ 결되는 1열의 행 번호

다음은 본 논문에서 제안하는 피더배치 알고리즘이다.

S1. (초기화) *feeder*[1],$\cdots$, *feeder*[n_L]을 초기화시킨다. 또한 *status*[1],$\cdots$, *status*[n_L]을 모두 available 로 초기화시킨다.

S2. (부품군 정렬) n_G개의 부품군을 부품군 사이클 수를 기준으로 하여 내림차순으로 정렬한다.

S3. 부품군 인덱스 g를 1부터 n_G까지 1씩 증가시키며 다음을 수행한다.

S3.1. 부품군 g에 속한 부품을 각각 *comp*[1],$\cdots$, *comp*[n_H]에 저장한다.

S3.2. (열 n_H) 현재 행 인덱스 j를 1부터 *next*[n_H][j]=0 n_L까지 1씩 증가하며, *cost*[n_H][j]=*incost*(n_H,j,0) 및 *next*[n_H][j]=0을 수행한다.

S3.3. (열 $n_H-1,\cdots,1$) 열 인덱스 i를 n_H-1부터 1까지 1씩 감소시키며 다음을 수행한다.

S3.3.1. 현재 행 인덱스 j를 1부터 n_L까지 1씩 증가시키며 다음을 수행한다.
단, *status*[j]=arranged이고 *feeder*[i]≠*omp*[i]인 j에 대하여는 수행을 생략한다.

S3.3.1.1. *cost*[i][j]를 ∞로 초기화하고, *next*[i][j]를 NULL로 초기화한다.

S3.3.1.2. 다음 행 인덱스 k를 1부터 n_L까지 1씩 증가시키며 다음을 수행한다. 단, *status*[k]=arranged이고 *feeder*[$i+1$]≠*omp*[$i+1$]인 k에 대하여는 수행을 생략한다. *value*=*inccost*(i,j,k)+*cost*[$i+1$][k]를 구하고, 만약 *value* < *cost*[i][j]이면, *cost*[i][j]에 *value*를대입하고 *next*[i][j]에 k를 대입한다.

S3.4. (열 0) 다음 행 인덱스 k를 1부터 n_L까지 1씩 증가하며, 다음을 수행한다. 단, *status*[k]=arranged이고 *feeder*[1]≠*omp*[1]인 k에 대하여는 수행을 생략한다. *value*=*inccost*(0,0,k)+*cost*[1][k]를 구하고, 만약 *value* < *cost**이면, *cost**에 *value*를 대입하고 *next**에 k를 대입한다.

S3.5. 슬롯 번호 l에 *next**를 대입한다.

S3.6. (피더배치) 열 인덱스 i를 1부터 n_H까지 1씩 증가시키며 다음을 수행한다.

S3.6.1. *feeder*[l]에 *comp*[i]를 대입한다. 또한 *status*[l]을 arranged 로 설정한다.

S3.6.2. 슬롯번호 l에 *next*[i][l]을 대입한다.

S4. (출력) 피더배치 리스트 $feeder[1],\cdots,feeder[n_L]$를 출력하고, 알고리즘을 종료한다.

위 알고리즘의 실행시간은 $n_G\times n_H\times n_L\times n_L$에 비례한다. 일반적으로 헤드 수 n_H는 10개 이내이므로 상수 취급을 할 수 있으며, 따라서 알고리즘의 계산 복잡도는 $O(n_G n_L^2)$으로 표시될 수 있다.

V. 시뮬레이션

새로이 제시한 알고리즘의 성능을 평가하고 그 유용성을 검증하기 위하여, 모의 칩마운터를 대상으로 하여 시뮬레이션을 수행하였다. 칩마운터의 헤드 수는 최대 6개이며, 헤드 사이의 간격은 20mm로 설정하였다. 칩마운터의 전후면 피더열에는 각각 42개씩 모두 84개의 피더 슬롯이 있으며, 슬롯 사이ㄴ의 간격은 10mm이다. X축, Y축 및 Z축은 각 각 독립적인 AC 서보 모터에 의하여 구동되며, 모두 S자 가감속 프로파일을 갖는다. 최 저속에서 최 고속까지 5단계의 속도 프로파일을 갖으며, 부품 별로 특정 단계의 프로파일을 설정하여 사용할 수 있다.

알고리즘은 C++ 언어로 프로그래밍 되었으며, Microsoft 사의 Visual C++로 제작되었다. 제작된 프로그램은 IBM-PC 호환 기종 Pentium-III급 / MS-Windows 2000 환경에서 실행되었다. 기 제작된 칩마운터용 그래픽 시뮬레이터[18]를 사용하여 시뮬레이션을 수행하였다. 칩마운터를 구동시키지 않고 오프라인에서 피더배치 및 장착순서를 그래픽으로 쉽게 확인할 수 있었으며, 또한 전체 사이클 시간을 추정할 수 있었다.

제안된 알고리즘의 성능을 평가하기 위하여, 기존에 제작된 알고리즘과 비교 시뮬레이션을 수행하였다. 비교 대상 알고리즘은 부품군 및 장착순서 결정 방식은 동일하며, 피더배치를 결정하는 방법만 차별화 된다. 피더의 초기 배치를 구하고, 피더의 배치 위치를 상호 교환하여 배치 결과를 개선시키는 국지탐색법을 적용하였다. 피더의 초기배치를 위하여 각 헤드 별로 피더슬롯의 우선 순위를 정한다. 이 때 우선 순위는 PCB의 중심으로부터 떨어진 거리와, 헤드 사이의 간격을 고려하여 결정한다. 각 부품군을 사이클 수를 기준으로 정렬하고, 사이클 수 가 많은 부품군부터 차례로 피더를 배치한다. 이때 미리 결정된 헤드 별 피더슬롯의 우선 순위를 고려하여 각 헤드 별 부품의 피더슬롯을 할당한다. 이와 같이 초기 배치된 피더들의 위치를 상호 교환하고 전체 조립시간을 비교하여, 전체 조립시간을 단축시키는 방향으로 교환을 진행한다. 더 이상의 개선이 없는 경우 교환을 중지한다. 비교대상 피더배치 알고리즘의 계산 복잡도는 초기 배치를 개선시키는 알고리즘의 계산 복잡도에 의존하며 $O(n_C^2 n_L)$로 표시할 수 있다, 제시한 알고리즘과 비교대상 알고리즘의 실행시간은 그 실측 결과 평균적으로 유사하였다.

표 1은 시뮬레이션에 사용된 보드들의 부품 및 장착점의 수를 보여준다. 이 보드들은 실제 제작되어 있는 보드이며, 각 부품 별 노즐, 최고 이동 속도, 장착 지연 시간 및 흡착 지연 시간 등을 설정하였다. 피더배치 방법의 비교가 목적이므로, 모든 부품은 동일한 노즐을 사용한다고 가정하였다. 또한 모든 부품은 한 개의 피더만 사용할 수 있다고 가정하였으며, 트레이 피더나 스틱 피더는 제외하였다.

표 2, 3, 4는 동일한 칩마운터에 대하여 각각 헤드를 2개, 4개, 6개 사용하는 경우에 대한 결과이다. 각 헤드 수에 대하여 기존 알고리즘과 새 알고리즘을 적용하였으며, 전체 조립시간을 비교하였다. 이 때 개선율은 다음과 같이 계산되었다.

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

일반적으로 헤드의 수가 증가할수록 전체 사이클 수는 감소하며, 따라서 전체 조립시간은 감소한다. 이 때 헤드 수가 증가할수록 동적계획법 적용 알고리즘의 개선율이 증가함을 볼 수 있다. 헤드 수가 증가함에 따라 경우의 수가 증가하여 문제의 복잡성이 커지나, 국지탐색법을 적용하는 경우 초기해에 크게 의존하므로 효율이 낮아질 수 있다. 헤드 수가 2개인 경우 평균 개선율은 9.9%, 4개인 경우 12.2%, 6개인 경우 13.2% 이다. 그러므로 본 논문에서 제시한 피더배치 알고리즘은 헤드 수가 많은 멀티헤드 겐트리형 칩마운터에 유용함을 알 수 있다.

부품 수가 증가함에 따라 피더의 수가 증가하며 많은 피더슬롯을 점유하게 된다. 이 경우 국지적 탐색법의 적용을 위한 초기 피더 배치의 효율이 저하될 수 있다. 그 이유는 초기 피더배치를 위하여 필요한 헤드 별 피더슬롯의 우선 순위의 결정이 어려워지기 때문이다. 반면 동적계획법을 적용하는 경우, 부품 수의 많고 적음에 관계없이 일정한 성능을 유지할 수 있음을 확인하였다.

VI. 결론

본 논문은 전자조립시스템의 핵심장비인 칩마운터를 대상으로 생산성 향상을 위한 하나의 방법을 제안하였다. 가격 대비 성능이 우수하여 생산 현장에서 많이 사용되고 있는 멀티헤드 겐트리형 칩마운터를 대상으로 하였으며, 조립시간의 단축에 매우 큰 영향을 주는 피더배치의 최적화를 위한 새로운 알고리즘을 제시하였다. 특히 기존의 연구가 단일헤드 겐트리형 칩마운터에 집중되어온 것에 대하여, 본 논문은 멀티헤드 겐트리형 칩마운터의 최적화 문제를 수학적으로 모델링 하였으며, 근사적 최적해를 구하기 위한 효율적 알고리즘을 제시하였다.

복잡한 최적화 문제에 적용될 수 있는 국지탐색법이 멀티헤드 겐트리형 칩마운터 문제에도 적용될 수 있다. 그러나 본 논문은 동적계획법을 피더배치 최적화 문제에 적용하였으며, 실제 칩마운터를 대상으로 한 시뮬레이션 결과, 국지탐색법에 비하여 약 9%-13% 의 성능향상을 확인 할 수 있었다. 개발된 알고리즘은 칩마운터의 자동 프로그래밍을 지원하는 소프트웨어에 최적화 엔진으로서 탑재될 수 있을 것이다. 향후 개발된 알고리즘의 상용화와 함께, 형식이 다른 칩마운터에도 적용을 확대하기 위한 연구를 지속할 예정이다.

표 1. 샘플 보드.
Table.1. Sample Board.

Board No	Board Name	No. of components	No. of points
1	Teach Box	7	48
2	GDS-main	10	55
3	Demo	21	310
4	WinTech-vga	31	155
5	2100top	43	111
6	AEE01881	53	396
7	WeTech-I33	58	107
8	HANA-IO	61	464
9	CRD8482	79	267
10	SDR2	83	723

표 2. 조립시간 비교(헤드 수 = 2).
Table.2. Comparison of assembly time(no. of heads = 2).

Board No	Assembly Time (sec)		Improvement Ratio (%)
	Local Search	Dynamic Programming	
1	36.1	33.4	7.5
2	41.0	37.7	8.1
3	270.6	245.4	9.3
4	117.9	107.5	8.8
5	94.9	85.9	9.5
6	322.0	288.5	10.4
7	90,6	81.3	10.3
8	373.2	330.3	11.5
9	212.1	189.8	10.5
10	611.7	534.0	12.7

표 3. 조립시간 비교(헤드 수=4).
Table.3. Comparison of assembly time(no. of heads=4).

Board No	Assembly Time (sec)		Improvement Ratio (%)
	Local Search	Dynamic Programming	
1	29.9	27.5	8.1
2	29.6	27.0	8.8
3	237.5	208.5	12.2
4	87.3	79.0	9.5
5	77.3	69.6	10.0
6	267.6	232.3	13.2
7	70.4	60.5	14.1
8	326.1	285.7	12.4
9	169.3	143.9	15.0
10	529.8	430.2	18.8

표 4. 조립시간 비교(헤드 수 = 6).
Table.4. Comparison of assembly time(no. of heads = 6).

Board No	Assembly Time (sec)		Improvement Ratio (%)
	Local Search	Dynamic Programming	
1	28.5	26.0	8.7
2	28.0	25.8	8.0
3	223.5	195.6	12.5
4	84.0	73.8	12.1
5	75.3	65.4	13.2
6	246.0	214.0	13.0
7	65.5	56.0	14.5
8	298.8	257.0	14.0
9	159.0	133.9	15.8
10	501.0	400.3	20.1

참고문헌

[1] T. L. Landers, W. D. Brown et. al., "Electronics manufacturing processes," *Prentice-Hall*, 1994.

[2] 박태형, "전자조립용 CAM 시스템의 기술동향", 전자공학회지, 제26권 제3호, pp.48-61, 1999.

[3] K. M. Nelson and L. T. Wille, "Comparative study of heuristics for optimal printed circuit board assembly," *Proc. of 1995 Southcon*, pp. 322-327, 1995.

[4] I. Or and E. Duman, "Optimization issues in automated production of printed circuit boards : operations sequencing, feeder configurations and line balancing problems," *Proc. 1996 IEEE Conf. on Emerging Technology and Factory Automation*, pp. 227-232, 1996.

[5] J. Sohn and S. Park, "Efficient operation of a surface mounting machine with a multihead turret," *International J. of Production Research*, vol. 34, no. 4, pp. 1131-1143, 1996.

[6] 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.

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

[8] 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.

[9] M. C. Leu, H. Wong, and Z. Li, "Planning of component placement / insertion sequence and feeder setup in PCB assembly using genetic algorithm," *Journal of Electronic Packaging in Trans. of the ASME*, vol. 115, 1993.

[10] W. Wang, P. C.Nelson, and T. M. Tirpak, "Optimization of high-speed multistation SMT placement machine using evolutionary algorithms," *IEEE Trans. on Electronics Packaging Manufacturing*, vol. 22, no. 2, pp. 137-145, 1999.

[11] T. M. Tirpak, P. C. Nelson, and A. J. Aswani, "Optimization of resolver head machine using adaptive simulated annealing (ASA)," *IEEE/CPMT Int'l Electronics Manufacturing Technology Symposium*, pp. 214-220, 2000.

[12] S. H. Lee, J. M. Hong, D. W. Kim, and B. H. Lee, "An effective algorithm for a surface mounting machine in printed circuit board assembly," *Proc. of the IROS 97*, pp. 932-937, 1997.

[13] S. Grotzinger, "Feeder assignment models for concurrent placement models," *IIE Transactions*, vol. 24, no. 4, pp. 31-46, 1992.

[14] G. W. DePuy, J. C. Ammons, and L. F. McGinnis, "Multiple assignment of component types to feeder slots on automated printed circuit card placement machines," *IEEE Trans. on Electronic Packaging Manufacturing*, vol. 23, no. 3, pp. 157-164, 2000.

[15] E.Arts and J.K.Lenstra, *Local Search in Combinatorial Optimization*, John Wiley & Sons, 1997.

[16] C. H. Papadimitriou, K. Steiglitz, *Combinatorial Optimization : Algorithms and Complexity*, Dover Publications, Inc., 1998.

[17] G.Reinelt, *The Traveling Salesman Problem : Computational Solutions for TSP Applications*, Springer-Verlag, 1994.

[18] 박기범, 박태형, "확률 페트리 네트를 이용한 객체지향 기반의 칩마운터 시뮬레이터 구현," 제어 자동화 시스템 공학 논문지, 제7권 제6호, pp. 540-549, 2001.

박 태 형

1963년 8월 29일 생. 1988년 서울대학교 제어계측공학과 학사. 1990년 동 대학원 석사 및 1994년 동 대학원 박사, 1992년~1994년 제어계측신기술 연구센터 연구원, 1994년~1997년 삼성항공산업(주) 정밀기기연구소 선임연구원.

2000년~2001년 Univ. of Toronto 방문교수, 1997년~현재 충북대학교 전기전자 및 컴퓨터공학부 조교수. 관심분야는 반도체 및 전자 조립 시스템, 로봇 동작계획 및 최적화 알고리즘.