동적계획법에 의한 멀티헤드 케트리형 침아우터의 장착순서 최적화

김동안**, 이재영**, 박태형**
**충북대학교 대학원 제조계측공학과
**충북대학교 전기전자 및 컴퓨터공학부

Abstract - 표면монтаж 인쇄회로기판 조립용 침아우터의 효율적인 운용을 위한 부품 장착 순서를 최적화하는 방법을 도출한다. 연구 과정은 멀티헤드 침아우터 부품 장착 순서를 동적계획법으로 수학적으로 모델링하고, 동적계획법의 방법으로 대상 침아우터의 장착 순서를 생성한다. 생성된 결과는 다른 알고리즘을 적용하여 생성된 결과와 비교 분석한다.

1. 서론

현재의 자산공정은 조립의 신뢰성 및 효율성을 위하여 표면монтаж(SMD:surface mounted device) 형태 인쇄회로기판(CB:printed circuit board)에 조립되고 있다. 표면монтаж 부품을 PCB에 조립하는 전자 시스템은 SMT(surface mounting technology) 인린인(in-line) 시스템이다 한다. SMT의 부품을 부착하기 위한 영역은 스크린프린터(screen printer)로 부착. PCB에 부품을 장착하는 침아우터(chip mounter) 공정, PCB에 도달한 납을 정착하는 리플로어(reflower) 공정으로 이루어져 있다. 이 중에서 가장 중요한 부분은 PCB 조립공정 시간이 가장 많이 걸리는 침아우터 공정이라 할 수 있다. 침아우터 공정은 수십개에서 수천개의 이중은 많은 부분이 장착된 응바로 장착공정에 따라 장착시간의 차이가 생기기 위하여 침아우터에 관련한 많은 연구가 있었다.[1][2][3][4]

작업을 위해서 도수 침아우터로 SMT 인린인 시스템이 구성할 수 있으며, 침아우터 운용을 최적화하는 방법은 3가지로 나누어 생각할 수 있다. 각 침아우터에 고르게 분배하는 방법, 각 침아우터에서 부품을 공급하는 침아우터 배치하는 방법, 각 침아우터의 장착 순서를 최적화하는 방법 등이 있다.

본 연구에서는 각 침아우터에서 고르게 작업이 분배되어 있고 정확한 배치 또는 배치되었는 가정에서 멀티헤드 침아우터의 부품 장착 속도 동적계획법(dynamic programming) 문제로 모델링하고 동적계획법으로 최적 해를 구하고 결과를 greedy 알고리즘을 적용하여 근사 결과값과 비교 분석한다.

다음은 본 연구에서 대상은 일본 Eichos의 침아우터 모델로 멀티헤드 싱글 케트리형인 S4가 그림 1과 멀티헤드 듀얼 케트리형인 S4D이다. (그림 2)

그림 1. S4 모델

그림 2. S4D 모델

2. 문제 정의(1)

침아우터 장착순서를 수학적으로 모델링을 위하여 다음과의 기호를 정의한다.

\[ P = \{1, \ldots, n_P\} \] \(\) 장착직의 집합

\[ H = \{1, \ldots, n_H\} \] \(\) 헤드 번호의 집합

\[ L = \{1, \ldots, n_L\} \] \(\) 파리은 패턴 번호의 집합

\[ I = \{1, \ldots, n_I\} \] \(\) 사이클 번호의 집합

\[ \rho^k_i \] 부품군 \(\mathbf{g}^k\)의 사이클 수

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

\[ I^k = L \] 부품군 \(\mathbf{g}^k\)의 헤드 \(h\)에 할당된 부품 \(\mathbf{c}^k_h\)의 파리은 번호.
\[ T_{\text{place,}XY} \quad \text{사자}l \text{의 장착시간 중 수품이동시간} \]

파리베치가 이며 끝난 상태에서 장착소에서 영향을 주는 시간은 후향이동시간, 전방이동시간 및 장착시간 중 수품 이동시간으로 다음과 같이 정의될 수 있다.

\[ T_{\text{backward}} = \sum_{\text{p} \in \text{P}} XY(n_{\text{h}}, \text{p}, 1, \vee_{k \in \text{k}}) \times (i-1)_{\text{w},k} \quad (1) \]

\[ T_{\text{forward}} = \sum_{\text{p} \in \text{P}} XY(n_{\text{h}}, \text{p}, 1, \vee_{k \in \text{k}}) \times i_{\text{p},k} \quad (2) \]

\[ T_{\text{place,}XY} = \sum_{\text{k} \in \text{k}} \sum_{\text{p} \in \text{P}} \sum_{\text{p} \in \text{P}} XY(k, \text{p}, 1 \pm 1, \text{p}) \times \vee_{k \in \text{k}} \quad (3) \]

식 (1)은 후향이동시간으로서 사자\(l\)의 처음 흐름 hann지의 이동시간이다. 사자\(l\)의 마지막 장착은 부품\(k\)\(\text{h}1\)에 앞바

\[ \text{원 부품} c_{2} \text{의 장착 점 검침} P_{12} \text{의 원소이다.} \]

또한 사자\(l\)의 처음 흐름은 부품\(k\)\(\text{h}1\)에 앞바턴 부품

\[ \text{의 피어스}\} \text{의 위치는} \text{으나} \text{전방으로} \text{시간이다. 식 (3)은 장착시간 중 수품이동시간으로 해당한다.} \]

장착소에서 최적화 문제는 장착소에서 변수의 최적해를 구하

\[ \text{는 문제로서 다음과 같다.} \quad \min \sum_{\text{p} \in \text{P}} (T_{\text{backward}} + T_{\text{forward}} + T_{\text{place,}XY}) \quad (4) \]

s.t. \[ \sum_{\text{p} \in \text{P}} x_{\text{p}} = 1, \forall \text{p} \in \text{P} \quad (5) \]

\[ \sum_{\text{p} \in \text{P}} x_{\text{p}} \geq 1, \forall i \in \text{I} \quad (6) \]

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

식 (3)은 각 장착은 단 한번 방출되어야 함을 의미하

\[ \text{며, 식 (6)은 각 사자\(l\)에서 한 개 이상의 장착을 방출하여 함을 의미한다. 또한 식 (7)은 각 사자\(l\)의 각 해드는 최적 장착을 방출할 수 있음을 의미한다.} \]

3. 알고리즘

장착 순서 최적화 문제 (식 (식)의 해를 구하기 위한 방

\[ \text{법을 구현하기가 너무 복잡하기 때문에 각 사자\(l\)별로 대

\[ \text{규 기준으로} \text{방향지해법을} \text{해결한다.} \]

\[ \text{\min} \sum_{\text{p} \in \text{P}} \rho_{\text{p}} \left( T_{\text{backward}} + T_{\text{forward}} + T_{\text{place,}XY} \right) \quad (8) \]

식 (8)을 식 (1),(2),(3)의 의미로 구현한 각 사자\(l\)에 대한

\[ \text{장착소에서 최적화하는 방법.} \]

\[ T_{\text{xy}} = T_{\text{backward}} + T_{\text{place,}XY} \quad \text{이므로} \]

\[ \text{\sum_{\text{p} \in \text{P}} \sum_{\text{p} \in \text{P}} XY(n_{\text{h}}, \text{p}, 1, \vee_{k \in \text{k}}) \times (i-1)_{\text{w},k} \]
4. 시뮬레이션

제안된 알고리즘은 Microsoft 사의 Visual C++로 구현되었으며 IBM-PC 호환 기종 Pentium-III급 /MS-WindowsXP 환경에서 실행되었다. 제안된 알고리즘은 컴퓨터 시뮬레이션에 의하여 검증되었으며 작업시간 수정은 대상 컴퓨터의 사양을 기반으로 시간 수정 루틴을 작성하였으므로 실험과 거의 동일하다고 할 수 있다. 실제로 대상 컴퓨터의 CPU, RAM 및 I/O 종류는 각각 독립적인 AC 서보 모터에 의하여 구동되며, 모터의 가속도와 프로파일을 갖는다. 최 저속에서 최 고속까지 5 단계의 속도 프로파일을 갖으며, 부품 벌로 특정 단계의 프로파일을 설정하여 사용할 수 있다.

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

개선율 = 기존 조립시간 / 세 조립시간 × 100(\%)

표 1. 대상 보드

<table>
<thead>
<tr>
<th>Board No</th>
<th>Board Name</th>
<th>부품수</th>
<th>착착수</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Disable</td>
<td>8</td>
<td>204</td>
</tr>
<tr>
<td>2</td>
<td>Aee01881</td>
<td>26</td>
<td>208</td>
</tr>
<tr>
<td>3</td>
<td>Test</td>
<td>16</td>
<td>217</td>
</tr>
<tr>
<td>4</td>
<td>DemoBoard</td>
<td>21</td>
<td>310</td>
</tr>
<tr>
<td>5</td>
<td>LG</td>
<td>57</td>
<td>242</td>
</tr>
<tr>
<td>6</td>
<td>CRD4842</td>
<td>26</td>
<td>122</td>
</tr>
<tr>
<td>7</td>
<td>HANA-JO</td>
<td>30</td>
<td>234</td>
</tr>
<tr>
<td>8</td>
<td>2100top</td>
<td>43</td>
<td>111</td>
</tr>
</tbody>
</table>

표 1은 대상으로 한 PCB 보드의 부품 정보를 보여주고 있다. 대상 PCB는 비슷한 부품 착착의 수가 작은 경우에 부품이 200개 내외가 착착된다.

표 2는 대상으로 한 마사지만의 머린드 칩습관을 사용한 결과를 보여주고 있는 결과이다. 비교대상으로 알고리즘은 greedy 알고리즘으로 동적계획법의 방법으로 구한 결과보다 좋다.

표 3은 머린드 칩습관의 칩습관에 사용한 S4D모델을 동적계획법에 의해서 착착순서를 구한 결과와 greedy 알고리즘을 사용한 결과를 비교하는 표이다. 대략 5%의 개선이 있음을 알 수 있다.

5. 결론

본 논문에서는 머린드 칩습관의 부품 착착 동적계획법에 의해서 모델링하였다. 실제 전자 제품 조립라인의 대부분이 머린드 칩습관을 사용하여 부품 착착 동작에 대한 알려진 모델이 많지 않았다. 모델링된 결과로부터 동적계획법을 이용하여 하나의 시점의 최적화를 이었다. 하지만 제안된 알고리즘은 한 사이클에 대해서는 최적화를 보장하지만 여러 사이클에 대해서는 최적화를 보장하지 못하는 단점이 있다. 이를 보완하기 위해 동적계획법으로 각 사이클을 구하고 local search의 일종인 2-opt을 사용하여 각 사이클의 벤엘 순서를 바꾸어 주면 더 나은 결과를 얻을 거라 예상된다.

(참고문헌)

1) 박태형, "동적계획법에 의한 머린드 칩습관의 최적 처리", 제어자동화시스템공학회지, 2002년 6월 게재 논문
6) L.Michalewicz and David B. Fogel, "How to Solve It : Modern Heuristics", FSpringer
7) R.Sedgewick, "Algorithms in C++", Addison-Wesley