Journal of the Korean Institute
of Industrial Engineers
Vol. 19, No.3, September., 1993.

Max($\bar{x}$, $\bar{y}$) TSP를 위한 발견적 해법†

A Heuristic Method for Max($\bar{x}$, $\bar{y}$) TSP †

이화기* · 서상문**

Hwaki Lee*, Sangmoon Suh**

Abstract

In this paper, the TSP(traveling salesman problem) which its costs(distance) between nodes are defined with Max($\bar{x}$, $\bar{y}$) has been dealt. In order to find a satisfactory solution for this kind of problem, we generate weighted matrix, and then develope a new heuristic problem solving method using the weighted matrix. Also we analyze the effectiveness of the newly developed heuristic method comparing it with other heuristic algorithm already exists for Euclidean TSP.

Finally, we apply a new developed algorithm to real Max($\bar{x}$,$\bar{y}$) TSP such as PCB inserting.

1. 서 론

PCB-보드에 부품을 挿入하는 挿入機械(inserting macine)는 두대의 모터에 의해 작동된다. 가로축 방향을 調整하는 모터와 세로축 방향을 調整하는 모터가 그것이다. 부품을 PCB-보드에 찍어주는 주 장치는 고정되어 있고, 보드를 固定하고 있는 받침대가 두 대의 모터에 의해서 컨트롤을 받아 한 위치에서 다음 위치로 移動한다. 그러므로 이동거리는 가로축과 세로축중 긴 쪽만큼 움직이게 된다. 즉, PCB-보드에 필요한 모든 부품을 挿入하는 挿入機械의 生産性 문제는 마디간의 움직이는 거리가 MAX($\bar{x}$, $\bar{y}$)로 정의되는 TSP이다.

TSP는 현재까지 多項式알고리듬(polynomial algorithm)이 알려져 있지 않은, 지수알고리듬만이 알려진 전형적인 NP-hard문제들중의 하나이다. TSP의 계산량에 관한 복잡성(complexity)은 이 사실에 기인한다.

이러한 이유로 效率的인 발견적해법(heuristic method)이 모색 되어져 왔다. 발견적 해법의 效率性은 다음과 같은 基準으로 評價된다.

(a) 실행시간(running time)
(b) 구현의 용이성(ease of implementation)
(c) 유연성(flexibility)
(d) 단순성(simplicity)

본 論文에서는 위와같은 기준에 부합하는 새로운 발견적 해법을 開發하는데 그 目的이 있다.

본 論文에서 다루고자 하는 TSP는 三角不等

† 본 연구는 1993년도 인하대학교 연구비지원에 의하여 수행되었음.

* 인하대학교 산업공학과

** 한국원자력연구소 연구원

式(triangle inequality) 제약조건

$$C_{ij} \leq C_{ik}+C_{kj}(\text{모든 } k\text{에 대해 } k \neq i,j)$$

을 만족하는 MAX($\bar{x}$, $\bar{y}$) TSP와 Euclindean TSP로 限定한다. 두 마디 i, j의 座標를 각각 (x_i, y_i)라 하면, MAX($\bar{x}$, $\bar{y}$) TSP의 두 마디 상의 거리(C_{ij})는

$$C_{ij}=\max(|x_j-x_i|, |y_j-y_i|)$$

로 정의 되고, Euclidean TSP문제의 두 마디 상의 거리(C_{ij})는

$$C_{ij}=\sqrt{(x_j-x_i)^2+(y_j-y_i)^2}$$

로 정의된다.

현재 TSP를 위한 별견적해법(heuristic method)들은 크게 Tour Construction Heuristic Methods와 Tour Improvement Heuristic Methods 두 部類로 區分지을 수 있다. 전자에 속하는 알고리듬에는 最近距離隣接點(nearest-neighbor) 알고리듬, 挿入(Insertion) 알고리듬, Clarke and Wright Savings 알고리듬, Convex Hull 알고리듬과 Christofides' Heuristic 등이 있다. 후자에 속하는 알고리듬에는 r-optimal 알고리듬, Lin and Kerninghan 알고리듬과 Or-opt알고리듬 등이 있다.

2. 새로운 Tour Construction알고리즘

2.1 加重値매트릭스의 意味와 生成

다음과 같은 費用(C_{ij}) 행렬을 갖는, 총 노드 수가 10개인 Symmetric TSP를 가정해 보자.

매트릭스상의 각 가지는 그 가지의 兩端에 걸쳐있는 두 개의 마디에 따라 兩面性을 가지고 있다. 편의상 표 1의 비용행렬을 갖는 TSP의 일부분을 그림으로 나타내면 그림 1과 같다. 즉 그림 1에서 보는 바와 같이 費用(C_{65})이 8인 가지는 마디 ⑥, ⑤ 중에서 어떤 마디 상에서 意思決定을 할 것인가에 따라 이 가지를 選擇할려고 하는 意思決定者의 意志度가 각각 달라진다.

표 1. 費用 매트릭스

	1	2	3	4	5	6	7	8	9	10
1	–	8	5	9	12	14	12	16	17	22
2	8	–	9	15	17	9	11	18	14	22
3	5	9	–	7	9	11	7	12	12	17
4	9	15	7	–	3	17	10	7	15	18
5	12	17	9	3		8	10	6	15	15
6	14	9	11	17	8	–	9	14	9	16
7	12	11	7	10	10	9	–	8	6	11
8	16	18	12	7	6	14	8	–	11	11
9	17	14	12	15	15	9	6	11	–	10
10	22	22	17	18	15	16	11	11	10	–

예를들면 마디 ⑥에서 費用이 8인 가지(⑤, ⑥)를 選擇하는 것이 다른 어떤 가지를 선택하는 것 보다도 總費用을 最少化하는데 가장 크게 寄與할 것처럼 보인다. 따라서 意思決定者는 가능한한 가지(⑥,⑤)를 選擇하고자 할 것이다. 그러나 마디 ⑤에서는 費用이 8인 가지(⑤,⑥)를 선택하는 것이 그밖의 다른 가지(⑤,④)와 (⑤,⑧)중 하나를 選擇하는 것 보다도 費用面에서 상당히 의사결정자의 意志度를 떨어뜨리는 結果가 될 것이다.

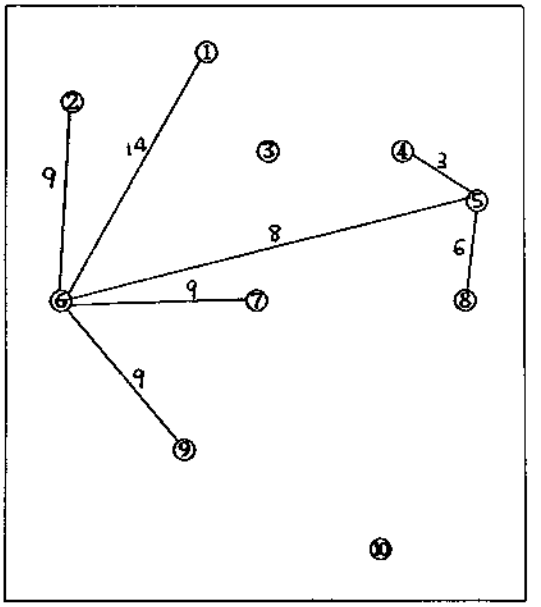

그림 1

이와같은 각 가지의 兩面性을 반영한 새로운 費用 매트릭스를 生成하면 어떤 가지를 선택할 것인가에 관한 의사결정(총비용을 최소화 하는데)에 큰 도움이 될 것 같다.

한가지 方案으로서 加重値 매트릭스(weighted matrix)를 다음과 같은 節次로 生成한다.

(a) 각 행(row)의 綜合을 구한다.

$S_1 = \sum_{j=1}^{n} C_{1j}$, (단, $1 \le j \le n$이고 $j \neq 1$인 정수)

$S_2 = \sum_{j=1}^{n} C_{2j}$, (단, $1 \le j \le n$이고 $j \neq 2$인 정수)

$\vdots$

$S_n = \sum_{j=1}^{n} C_{nj}$, (단, $1 \le j \le n$이고 $j \neq n$인 정수)

(b) 각 가지들의 相對比率을 구한다.

$$\begin{bmatrix} 0 & A_{12}=C_{12}/S_1 & A_{13}=C_{13}/S_1 & \cdots & A_{1n}=C_{1n}/S_1 \\ A_{21}=C_{21}S/_2 & 0 & A_{23}=C_{23}/S_2 & \cdots & A_{2n}=C_{2n}/S_2 \\ A_{31}=C_{31}/S_3 & A_{32}=C_{32}/S_3 & 0 & \cdots & A_{32}=C_{32}/S_3 \\ \vdots & & & \ddots & \\ A_{n1}=C_{n1}/S_n & A_{n2}=C_{n2}/S_n & A_{n3}=C_{n3}/S_n & \cdots & 0 \end{bmatrix}$$

(b) 를 적용하면 비대칭(asymmetric) 매트릭스가 생선된다. 다음의 (c)과정을 적용하여 대칭 매트릭스로 만들어 준다.

(c) 마지막으로 최종 생성된 가중치 매트릭스는 다음과 같은 對稱行列이 된다.

$$\begin{bmatrix} 0 & C_{12}(A_{12}+A_{21}) & C_{13}(A_{13}+A_{31}) & \cdots & C_{1n}(A_{1n}+A_{n1}) \\ & 0 & C_{23}(A_{23}+A_{32}) & \cdots & C_{2n}(A_{2n}+A_{n2}) \\ & & 0 & \cdots & C_{3n}(A_{3n}+A_{n3}) \\ & & & \ddots & \vdots \\ & & & & 0 \end{bmatrix}$$

생성된 加重値 매트릭스의 각 費用은 변화된 것이기 때문에 이 매트릭스를 이용한 最適解가 原問題의 最適解와 반드시 일치하지는 않는다. 단지 開發하고자 하는 발견적 해법에 하나의 도구로서만 利用될 뿐이다.

2.2 整列 알고리즘

整列 알고리즘에는 이미 開發된 것들이 많이 있다. n개의 資料를 정렬하고자 할때 가장 效率性이 낮은 버블 정렬(Bubble Sort) 알고리즘의 경우 연산회수는

$$O(n) = \frac{n(n-1)}{2}$$

이다. 기존의 整列 알고리즘중 가장 效率性이 좋은 Quick Sort알고리즘의 경우 5000개의 자료를 가지고 同種의 컴퓨터에서 버블 정렬 알고리즘과 비교 실험하였을 前者는 11초를 소요하였다.

본 論文에서 이용한 정렬 알고리즘은 Quick Sort알고리즘 이다.

2.3 加重値 정렬과 그 意味

n개의 노드를 가지는 TSP에서 加重値 매트릭스를 生成한 다음 새롭게 생성된 매트릭스에서 각 행별로 加重値값이 가장 작은 가지를 2개씩을 選定한다. 이렇게 선정된 2n개의 가지들을 새로운 費用에 대한 오름차순으로 정렬한 다음과 같다.

$$(W_1, W_2, W_3, \cdots, W_{2n}) \qquad (1)$$

앞에 나오는 가지 일수록 意思決定者가 그 가지의 兩端에 걸쳐있는 노드에서 의사결정을 할 경우 그 가지를 選擇하려는 意志度는 높아진다. 위와같이 정렬된 가지들을 利用하여 Hamiltonian Cycle을 構成하기 위한 政策은 앞에서 부터 차례로 가지들을 選擇하는 것이다. 이러한 선정과정에서 部分循環路(subtour)를 형성하는 가지와 양단의 두 노드의 degree가 2이상인 것이 하나라도 존재하는 가지는 제외 한다. 다시말하면 의사결정자의 意志度가 가장 큰 것이 비용을 최소화 하는데 가장 크게 寄與할 것이라는 意味가 담겨있는 것이다.

n개의 노드를 가지는 TSP는 費用을 최소화 하도록 n개의 가지들을 선택하여 Hamiltonian Cycle을 구성하는 것이다. 加重値 매트릭스를 이용하여 정렬된 2n개의 가지중에서 되도록 n개의 가지를 다 選定할 수 있다면, 이렇게 構成된 tour가 最適解임을 증명할 수는 없지만 보다 좋은 解임은 틀림 없을 것이다.

2.4 加重値 정렬을 利用한 발견적해법

加重値 行列을 이용하여 정렬된 각 가지들을 選擇할 때는 다음과 같은 두가지 條件들을 만족하여야 한다.

(a) 각 가지의 兩端에 걸쳐있는 두 노드의 degree는 0 또는 1이어야 한다.

(b) 그 가지를 選擇했을 때 subtour가 발생하지 않아야 한다.

위의 두 條件을 만족하면서도 Hamiltonian Cycle을 構成하기 위해서는 다음과 같은 새로운 表記法이 필요하다.

d_i : 각 가지의 兩端에 걸치는 두 마디는 현재 degree.

g_i : 각 노드가 속하는 group을 나타내는 정수

예를들면 차례로 가지를 선택해 나가는 과정에서 가지 양단에 걸쳐있는 두마디의 degree가 각각 1이고 속해있는 group이 같다면 subtour가 발생한다. 따라서 이 가지는 선정과정에서 제외 된다.

위와같은 主要 두 노테이션을 利用한 tour construction알고리즘은 다음과 같다.

Phase-Ⅰ :

Step 1 : 加重値 매트릭스상에서 각 행별로 C_{ij} 값이 가장작은 가지 2개씩을 선택한다. 이렇게 선택된 2n개의 가지들을 오름차순으로 整列한다. 모든 마디에 대한 d_i를 0으로 놓는다. 모든 마디에 대한 g_i를 i로 놓는다. k=1로 놓는다. $S_{sub}=\{0\}$.

Step 2 : 정렬된 가지중에서 k번째 가지 兩端의 노드 i, j에 대하여 $d_i \neq 2$과 $d_j \neq 2$이고, $g_i \neq g_j$이면 이 가지를 集合 S_{sub}에 넣고 Step 3으로 간다. 그렇지않으면 k를 1만큼 增加시키고 Step 2를 다시 반복한다.

Step 3 : k를 1만큼 증가시킨다.

i) 만약에 선택된 가지 兩端에 있는 두 노드의 $d_i=d_j=1$ 이면 g_i인 모든 노드를 g_j로 놓는다. $d_i=d_i+1$, $d_j=d_j+1$로 놓는다.

ii) 만약에 선택된 가지 兩端에 있는 두 노드의 d_i, d_j중에서 $d_i=0$이면 $g_i=g_j$로 놓고, 그렇지 않으면 $g_j=g_i$로 놓는다.

k=2n+1이면 Step 4로 가고, 그렇지 않으면 Step2로 돌아간다.

Step 4 : Phase Ⅱ를 連續的으로 適用한다.

Phase-Ⅱ

Step 1 : d_i, g_i 는 Phase-Ⅰ으로 부터 連續이다. m를 1로 놓는다.

Step 2 : $d_m=1$이면 Step 3으로 가고, 그렇지 않으면 m을 1만큼 증가시킨다음 Step2를 反復한다.

Step 3 : $i \neq m$, $d_i=1$, $g_i \neq g_m$인 모든 노드 i에 대하여 C_{im} 중 가장 작은 값을 가지는 arc를 選擇하여 집합 S_{sub}에 넣는다. $d_i=d_i+1$, $d_m=d_m+1$로 놓는다. g_m인 모든 노드의 g값을 g_i로 놓는다. m=n이면 Step 4로 가고, 그렇지않으면 m을 1만큼 增加시킨다음 Step 2를 反復한다.

Step 4 : 이 단계에서 모든 노드의 degree를 검색하면 $d_i=1$인 두개의 노드가 존재한다. 이 두 노드를 잇는 가지를 선택하여 집합 S_{sub}에 넣는다. 아직까지 d_i가 0인 모든 노드를 총비용이 최소가되는 방향으로 차례로 삽입시킨다.

주의 할 점은 Phase-Ⅱ적용 과정에서는 原매트릭스를 사용한다. 실험적으로 Phase-Ⅰ適用 過程에서는 약 0.85n~0.9n개의 가지가 選定 되어졌다. Phase-Ⅱ에서 나머지 가지들을 選定한다.

위 알고리듬을 순서도(flow chart)로 나타내면 그림 2와 같다.

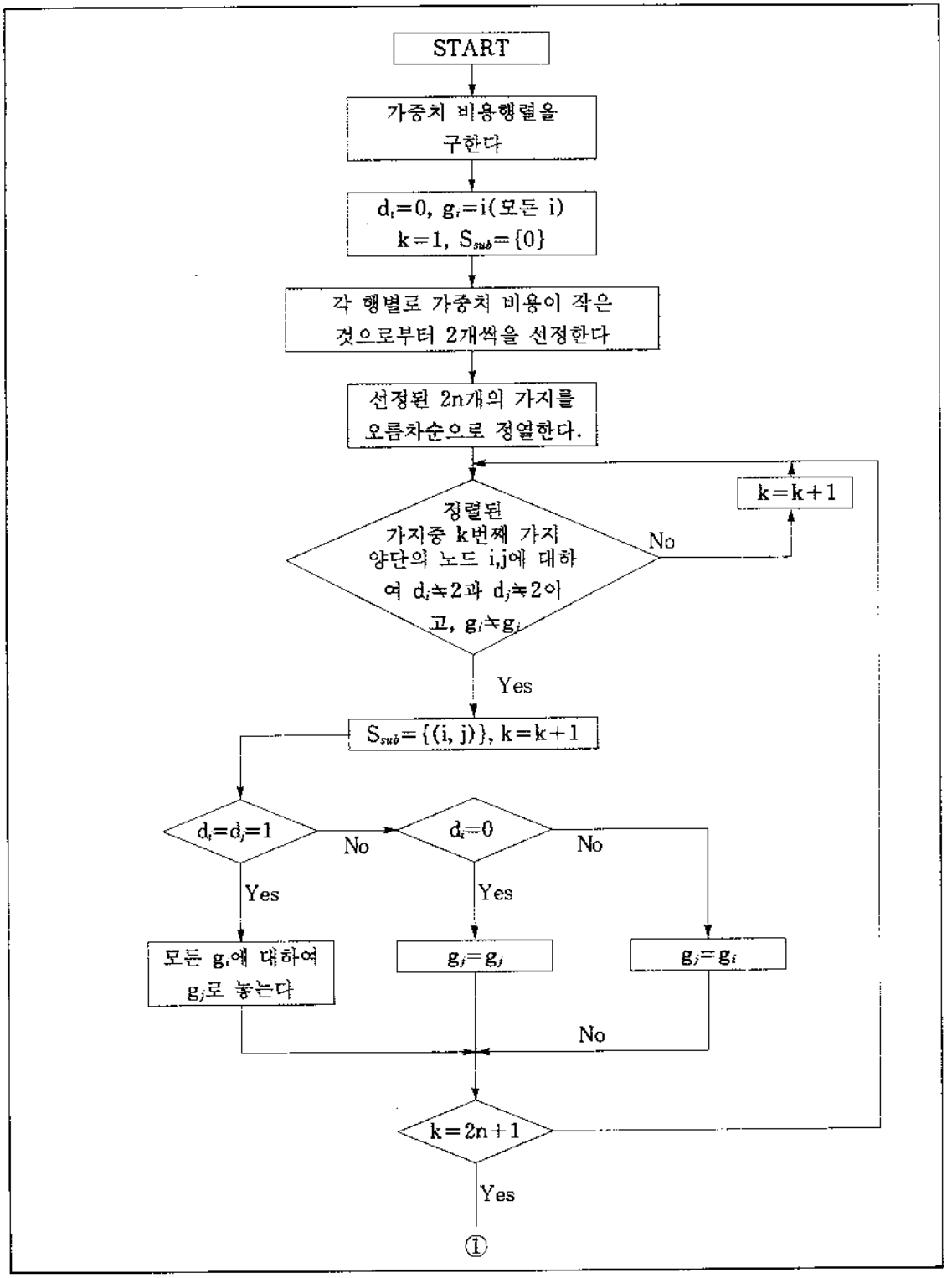
START
가중치 비용행렬을 구한다
d_i=0, g_i=i(모든 i)
k=1, S_{sub}={0}
각 행별로 가중치 비용이 작은 것으로부터 2개씩을 선정한다
선정된 2n개의 가지를 오름차순으로 정렬한다.
정렬된 가지중 k번째 가지 양단의 노드 i,j에 대하여 d_i≠2과 d_j≠2이고, g_i≠g_j
No
k=k+1
Yes
S_{sub}={(i, j)}, k=k+1
d_i=d_j=1
No
d_i=0
No
Yes
Yes
모든 g_i에 대하여 g_j로 놓는다
g_j=g_j
g_j=g_i
No
k=2n+1
Yes
①

그림 2.(계속)

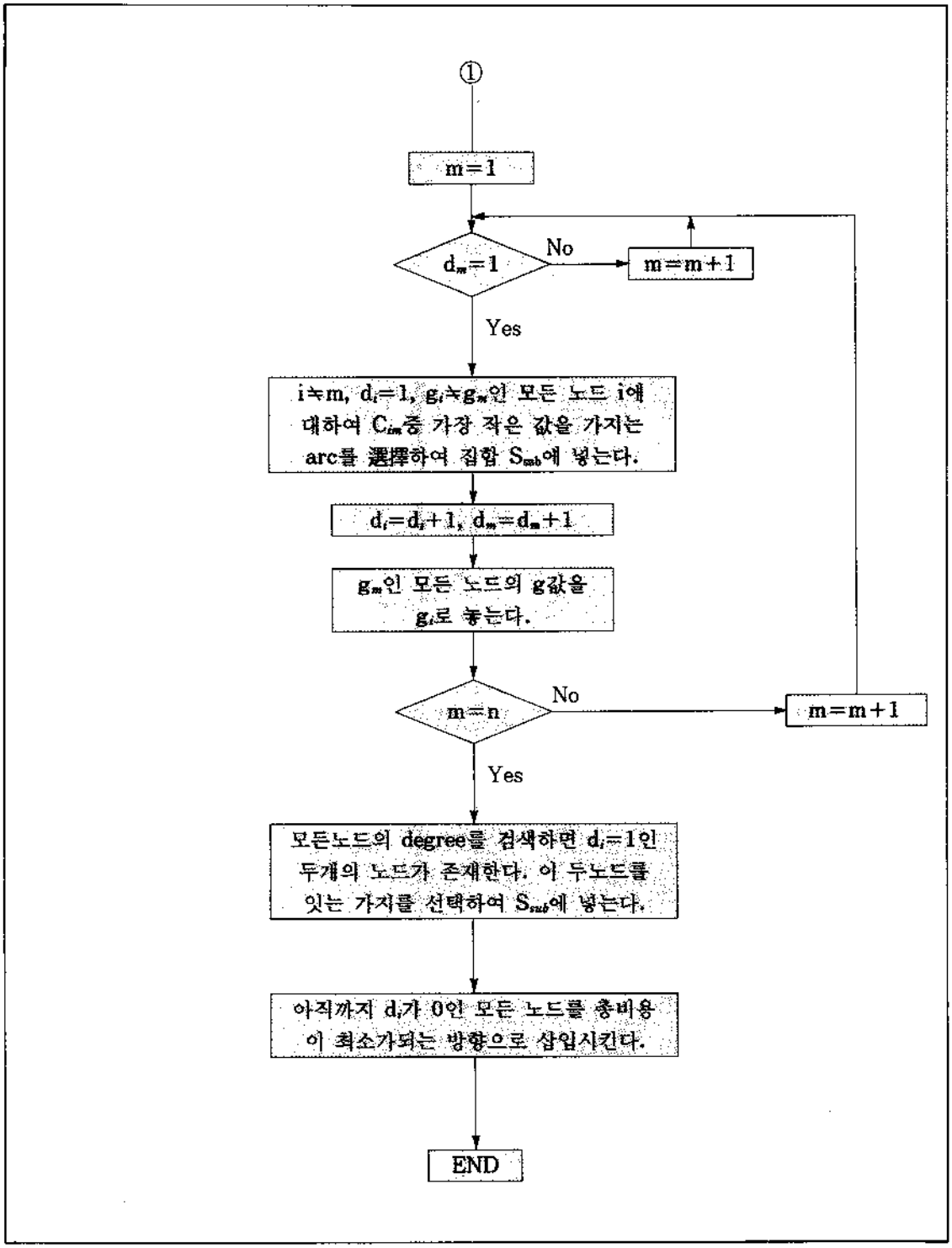
①
m=1
d_m=1
No
m=m+1
Yes
i≒m, d_i=1, g_i≒g_m인 모든 노드 i에 대하여 C_{im}중 가장 작은 값을 가지는 arc를 選擇하여 집합 S_{sub}에 넣는다.
d_i=d_i+1, d_m=d_m+1
g_m인 모든 노드의 g값을 g_i로 놓는다.
m=n
No
m=m+1
Yes
모든노드의 degree를 검색하면 d_i=1인 두개의 노드가 존재한다. 이 두노드를 잇는 가지를 선택하여 S_{sub}에 넣는다.
아직까지 d_i가 0인 모든 노드를 총비용이 최소가되는 방향으로 삽입시킨다.
END

그림 2

2.5 開發된 알고리라듬의 適用 例

표 1의 비용행렬을 갖는 10노드 문제를 개발한 Tour Construction알고리듬에 적용한 계산과정은 다음과 같다. 먼저 가중치 비용행렬을 구한다.

	1	2	3	4	5	6	7	8	9	10
1	–	0.80	0.49	1.50	2.76	3.53	2.96	4.70	5.16	7.60
2		–	1.56	4.04	5.38	1.40	2.42	5.77	3.38	7.33
3			–	1.03	1.76	2.48	1.13	3.00	2.93	5.27
4				–	0.17	5.56	2.18	0.95	4.28	5.48
5					–	1.26	2.24	0.71	4.42	3.94
6						–	1.71	3.73	1.49	4.19
7							–	1.38	0.75	2.28
8								–	2.28	2.02
9									–	1.61
10										–

가중치 비용행렬상에서 각 행당 작은 비용을 갖는 것으로 부터 2개씩의 가지를 선정한다. 선정된 가지와 각각의 가중치 값은 다음 표와 같다.

(1,3)	(1,2)	(2,3)	(2,6)	(3,4)	(3,7)	(4,5)	(4,8)	(5,6)	(5,8)
0.49	0.80	1.56	1.40	1.03	1.13	0.17	0.95	1.26	0.71
(6,7)	(6,9)	(7,8)	(7,9)	(8,9)	(8,10)	(9,3)	(9,10)	(10,5)	(10,7)
1.71	1.49	1.38	0.75	2.28	2.02	2.93	1.61	3.94	2.29

위와 같이 선정된 2n개의 가지들을 각각의 가중치 값에 따라 오름차순으로 정렬하면

(4,5) (1,3) (5,8) (7,9) (1,2) (4,8)
(3,4) (3,7) (5,6) (7,8) (2,6) (6,9)
(2,3) (9,10) (6,7) (8,10) (8,9) (10,7)
(9,3) (10,5)

개발한 Tour Construction알고리듬의 Phase I 을 적용하여 보자.

먼저 각노드의 현재의 degree를 나타내는 노테이션 d_i와 g_i를 다음과 같이 초기화 시켜준다.

$d_i=0$, (단, $i=1,2,3,\cdots,n$)
$g_i=0$, (단, $i=1,2,3,\cdots,n$)

(가) 첫번째 가지 (4,5)에 대하여 $d_4=0<2$, $d_5=0<2$이고 $g_4 \neq g_5$이므로 $S_{sub}=\{(4,5)\}$, $d_4=d_4+1=1$, $d_5=d_5+1=1$과 $g_4=g_5=5$로 놓는다.

(나) 두번째 가지 (1,3)에 대하여 $d_1=0<2$, $d_3=0<2$이고 $g_1 \neq g_3$이므로 $S_{sub}=\{(4,5),(1,3)\}$, $d_1=d_1+1=1$, $d_3=d_3+1=1$과 $g_1=g_3=3$로 놓는다.

(다) 세번째 가지 (5,8)에 대하여 $d_5=1<2$, $d_8=0<2$이고 $g_5 \neq g_8$이므로 $S_{sub}=\{(4,5),(1,3),(5,8)\}$, $d_5=d_5+1=2$, $d_8=d_8+1=1$과 $g_8=g_5=5$로 놓는다.

(라) 네번째 가지 (7,9)에 대하여 $d_7=0<2$, $d_9=0<2$이고 $g_7 \neq g_9$이므로 $S_{sub}=\{(4,5),(1,3),(5,8),(7,9)\}$, $d_7=d_7+1=1$, $d_9=d_9+1=1$과 $g_7=g_9=9$로 놓는다.

(마) 다섯번째 가지 (1,2)에 대하여 $d_1=1<2$, $d_2=0<2$이고 $g_1 \neq g_2$이므로 $S_{sub}=\{(4,5),(1,3),(5,8),(7,9)\}$, $d_1=d_1+1=1$, $d_2=d_2+1=1$과 $g_2=g_1=3$로 놓는다.

(바) 여섯번째 가지 (4,8)에 대하여 $d_4=2$이므로 선택 하지 않는다.

위와 같은 방식으로 20번째 가지 까지 Phase I 을 적용 시켜 나가면 그림 3과 같은 미완성 tour를 구할 수 있다.

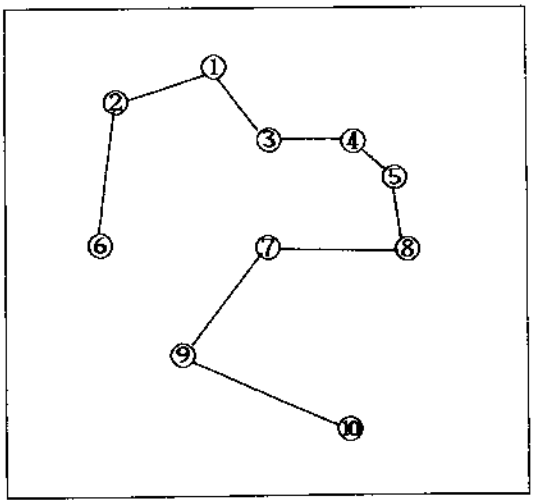

그림 3

현재까지 노드 9와 10만이 degree가 1이다. Phase Ⅱ를 연속적으로 적용하여 그림 4과 같은 완성 tour를 구한다.

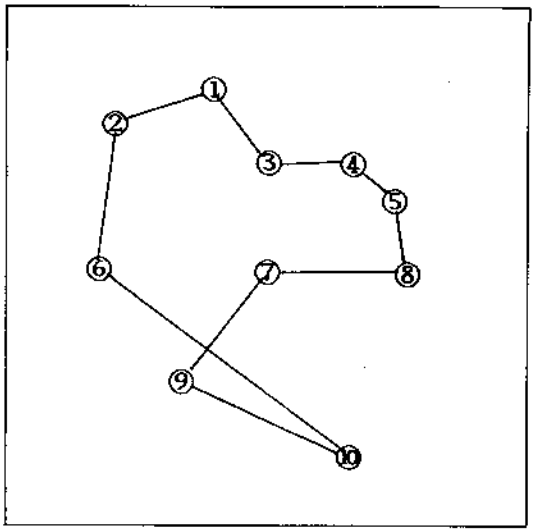

그림 4

3. 효율성 分析

새로운 알고리즘의 효율성 분석을 위해 적용되는 시험문제(test problem)는 각 문제당 마디를 座標形式으로 均一분포에서 문제당 100개씩의 노드를 抽出하여 30개를 발생시켰다. x와 y좌표의 範圍는

$0<x<640$, (단, x는 정수)

$0<y<480$, (단, y는 정수)

이다. 즉, 각각의 發生된 試驗問題는 n이 100인 對稱(symmetric) TSP이다.

3.1 MAX($\bar{x}$, $\bar{y}$) TSP문제의 경우

이 節에서 발생된 100개의 試驗問題에서 각 노드간의 費用(distance)을 MAX($\bar{x}$, $\bar{y}$)로 정의하여 費用 매트릭스를 생성한 다음 개발된 Construction알고리듬을 일반적인 TSP문제의 경우에 있어서 가장 效率性이 높다고 알려진 기존의 알고리듬과 比較 分析하였다. 편의상 다음과 같은 表記法을 使用하겠다.

N-N : 最近距離隣接點(nearest-neighbor)알고리듬
W-S : 개발된 Tour Construction알고리듬
L-K : Lin & Kerninghan알고리듬

표 2에서 보는바와 같이 100개의 노드를 갖는 30개의 test문제를 시험 하였을때 最近距離隣接點 알고리즘의 평균 所要時間은 0.315sec가 걸렸고, 개발한 알고리듬은 0.18sec가 소요되었다. 구해진 Hamiltonian循環路의 總費用면에서도 30개의 試驗問題중 개발된 Tour Construction알고리듬이 6개를 제외한 24개의 시험문제에서 優勢하였다. 最近距離隣接點 알고리듬을 적용 하였을 경우 출발 노드는 문제 발생시 맨 처음 발생된 노드이다.

표 2. 最近距離隣接點 알고리듬과 開發된 알고리듬의 結果 比較

Test Problem	N-N		W-S		$C_{N-N}-C_{W-S}$
	C_{N-N}	*t/sec	C_{W-S}	*t/sec	
TP 01	4850	0.33	4439	0.16	411
TP 02	4434	0.33	4219	0.16	215
TP 03	4582	0.28	4555	0.16	27
TP 04	4744	0.28	4779	0.22	−35
TP 05	4722	0.28	4493	0.22	229
TP 06	4998	0.33	4714	0.16	284
TP 07	5289	0.33	4507	0.16	782
TP 08	4620	0.28	4756	0.22	−136
TP 09	5228	0.33	4550	0.16	678
TP 10	5311	0.33	4703	0.22	608

Test Problem	N−N		W−S		$C_{N-N}-C_{W-S}$
	C_{N-N}	*t/sec	C_{W-S}	*t/sec	
TP 11	4534	0.33	4360	0.16	174
TP 12	4972	0.33	4432	0.16	540
TP 13	4771	0.33	4802	0.16	−31
TP 14	5059	0.28	4314	0.16	745
TP 15	4730	0.28	4697	0.16	33
TP 16	4630	0.33	4449	0.16	181
TP 17	4201	0.33	4818	0.16	−617
TP 18	5089	0.33	4866	0.16	223
TP 19	4911	0.33	4958	0.16	−47
TP 20	4989	0.33	4407	0.16	582
TP 21	4747	0.33	4173	0.16	574
TP 22	4291	0.28	4368	0.22	−77
TP 23	5256	0.28	4654	0.22	602
TP 24	4242	0.33	4004	0.16	238
TP 25	4668	0.33	4481	0.16	187
TP 26	5831	0.33	4465	0.22	1366
TP 27	4998	0.33	4964	0.16	34
TP 28	5017	0.28	4331	0.16	686
TP 29	4926	0.33	4592	0.22	334
TP 30	4825	0.33	4746	0.16	79

(* 현재 Super 486/25i의 CPU time)

neighbor알고리듬의 初期解를 구한다음 Lin & Kerninghan개선 알고리듬을 適用한 결과와 개발된 알고리듬으로 초기해를 구한 후 Lin & Kerninghan改善 알고리듬을 適用한 結果는 아래 표 3와 같다.

표 3. 最近距離隣接點 알고리듬+Lin & Kerninghan과 開發된 알고리듬+ Lin & Kerninghan과의 計算 結果 比較

Test Problem	N-N & L−K (C_{N-L})	W−A $ L−K (C_{W-L})	$C_{N-L}-C_{W-L}$	Test Problem	N-N & L−K (C_{N-L})	W−A $ L−K (C_{W-L})	$C_{N-L}-C_{W-L}$
TP 01	3923	3915	8	TP 16	3863	3838	25
TP 02	4059	3850	209	TP 17	4052	3988	64
TP 03	3951	3932	19	TP 18	4047	3960	87
TP 04	4066	4056	10	TP 19	3977	3942	35
TP 05	3965	3987	−22	TP 20	3890	3941	−51
TP 06	4014	3928	86	TP 21	3940	3843	97
TP 07	4030	4068	−38	TP 22	3849	3837	9
TP 08	3860	3856	4	TP 23	4071	4082	11
TP 09	4143	4053	90	TP 24	3510	3510	0
TP 10	4082	4135	−71	TP 25	4009	3975	34
TP 11	3947	3872	75	TP 26	4126	4083	43
TP 12	3806	3732	74	TP 27	3983	3887	96
TP 13	4011	3961	50	TP 28	3721	3736	−15
TP 14	4037	3880	157	TP 29	3870	3833	37
TP 15	3978	3973	5	TP 30	4075	4173	98

표 4. 最近距離前點 알고리듬+Lin & Kerninghan과 開發된 알고리듬+Lin & Kerninghan과의 소요시간 結果 比較

Test Problem	N−N & L−K *(t/sec)	W−S & L−K *(t/sec)	Test Problem	N−N & L−K *(t/sec)	W−S & L−K *(t/sec)
TP 01	1.264	1.044	TP 16	1.264	1.484
TP 02	1.429	0.989	TP 17	1.209	1.264
TP 03	1.319	1.429	TP 18	1.978	1.373
TP 04	1.319	1.703	TP 19	1.374	1.593
TP 05	1.374	1.538	TP 20	1.593	1.319
TP 06	1.978	0.824	TP 21	1.593	0.769
TP 07	1.319	1.319	TP 22	1.429	1.538
TP 08	1.154	1.648	TP 23	1.648	1.154
TP 09	1.703	0.879	TP 24	1.703	1.868
TP 10	1.923	1.319	TP 25	1.484	0.934
TP 11	1.429	1.044	TP 26	1.868	0.989
TP 12	2.308	1.209	TP 27	1.703	1.484
TP 13	1.538	1.429	TP 28	1.648	1.099
TP 14	1.374	0.934	TP 29	1.374	2.198
TP 15	1.758	1.703	TP 30	1.319	1.099

(* 현대 Super 486.25i의 CPU time)

표 5. Euclidean비용의 비교

Test Problm	N−N & L−K (C_{N-L})	W−S & L−K (C_{W-L})	$C_{N-L}-C_{W-L}$	Test Problm	N−N & L−K (C_{N-L})	W−S & L−K (C_{W-L})	$C_{N-L}-C_{W-L}$
TP 01	4411.9	4462.1	−50.2	TP 16	4370.7	4341.8	28.9
TP 02	4425.1	4359.7	65.4	TP 17	4520.4	4407.9	112.5
TP 03	4466.8	4458.6	8.2	TP 18	4537.5	4592.6	−55.1
TP 04	4473.6	4559.2	−85.6	TP 19	4478.9	4434.6	44.3
TP 05	4509.7	4454.1	55.6	TP 20	4333.8	4436.8	−103.0
TP 06	4606.1	4491.9	114.2	TP 21	4390.0	43.77.2	12.8
TP 07	4411.8	4467.8	−56.0	TP 22	4266.9	4263.9	3.0
TP 08	4355.6	4229.3	126.3	TP 23	4620.4	4524.0	96.4
TP 09	4591.3	4661.8	−70.5	TP 24	4212.2	4184.6	27.6
TP 10	4715.3	4663.9	51.4	TP 25	4366.9	4499.3	−132.4
TP 11	4428.5	4386.0	42.5	TP 26	4629.4	4747.2	−117.8
TP 12	4200.1	4255.8	−55.7	TP 27	4457.2	4402.0	55.2
TP 13	4455.1	4512.5	−57.4	TP 28	4216.7	4267.3	−50.6
TP 14	4528.9	4331.1	197.8	TP 29	4442.3	4450.8	−8.5
TP 15	4525.2	4476.6	48.6	TP 30	4588.4	4442.1	146.3

알고리듬의 適用결과 30개의 試驗문제중 5개의 시험문제에서만 약간 解가 좋지 않았고, 나머지 25개의 시험문제에서는 優勢하였다.

표 4에서 보는바와 같이 時間 면에서도 30개의 試驗問題에 기존 알고리듬들을 適用 하였을 경우 평균 1,546sec가 所要된 반면 새로운 알고리듬을 適用한 結果는 1,342sec가 걸렸다.

3.2 Euclidean TSP의 경우

이 節에서는 3.1절에서 발생한 30개의 試驗問題를 그대로 사용하였다. 다만 각 노드간의 費用(distance)을 Euclidean distance로 정의하여 費用 매트릭스를 生成한 다음 각각의 알고리듬을 適用 하였다. 그 결과를 산출한 다음 效率性 분석을 시행 하였다. 費用을 Euclidean distance로 정의하여 시험문제를 푼 결과는 표 5와 같다.

총 30개의 시험문제중 18개의 문제에서 개발한 알고리듬으로 初期解를 구한 다음 Lin & Kerninghan개선 알고리듬을 適用한 결과가 優勢하였지만 월등하게 우세하다고는 결론을 내릴수는 없다.

4. PCB-보드의 例

開發된 알고리듬을 PCB-보드 挿入 문제에

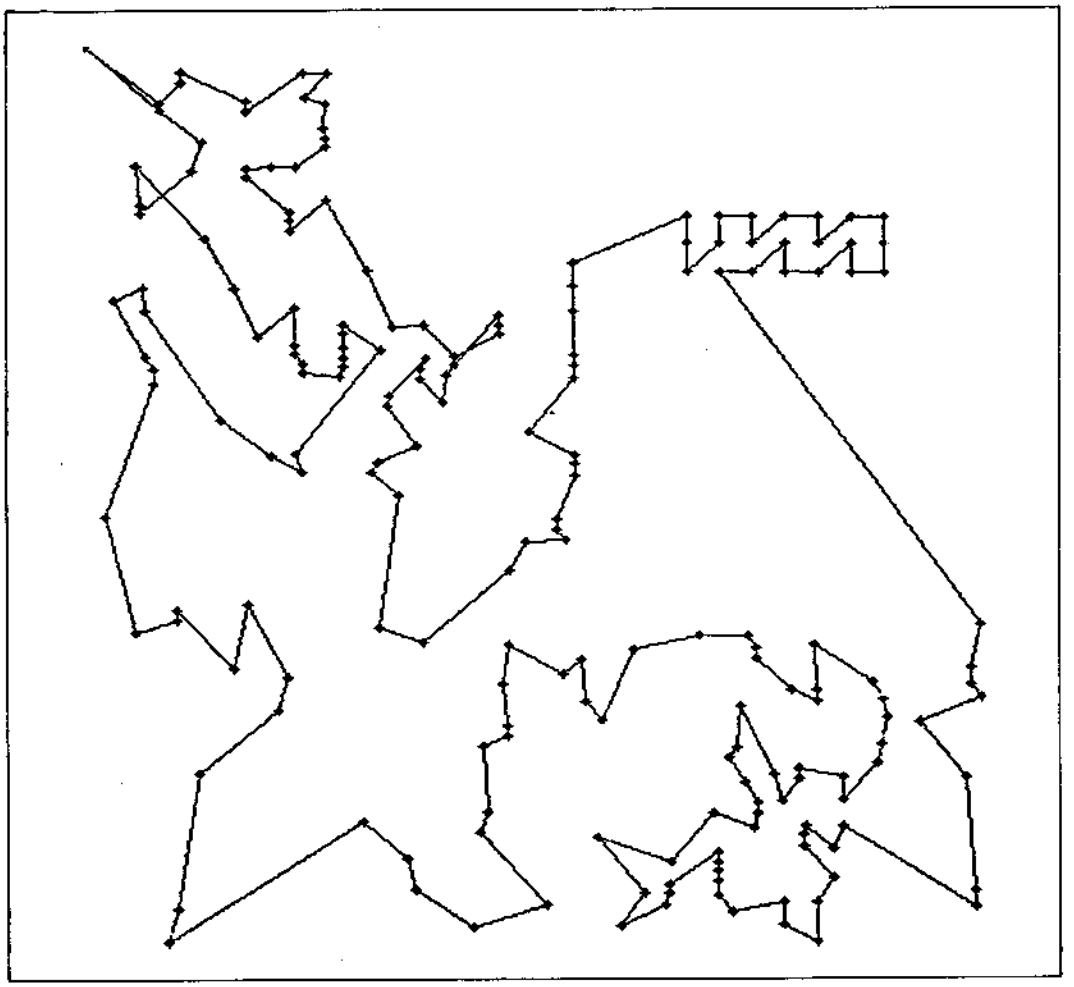

그림 5. 195경우의 경로

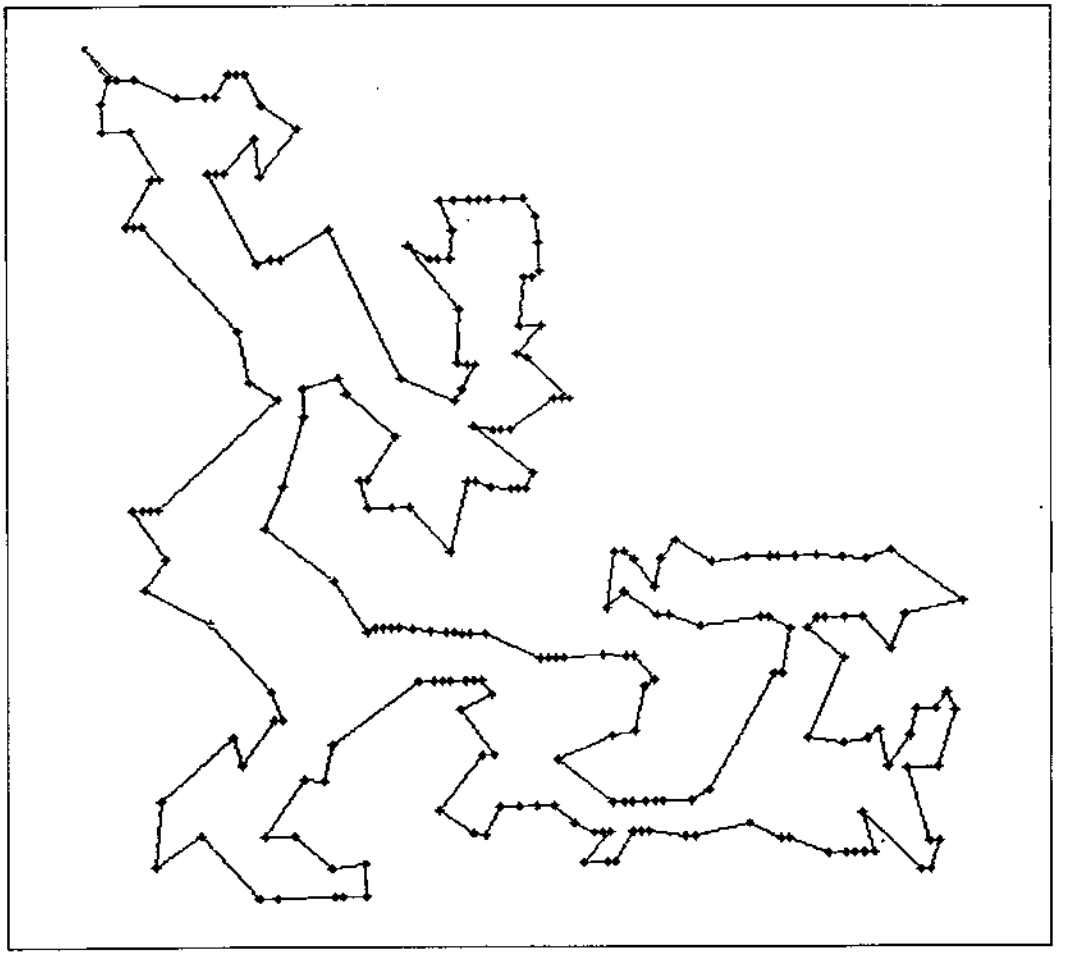

그림 6. 236경우의 경로

適用하였다. PCB-보드는 다품종 중량생산체제로 다양한 보드마다 매번 효율적인 삽입이 요구 된다. 이 중 한예를 들면, 對象 PCB-보드는 총 430개의 마디로 構成되어 있다. 그러나 실제로는 部品을 挿入해주는 방향에 따라 각각 195개와 235개의 노드로 구성되는 두개의 Max($\bar{x}$, $\bar{y}$) TSP로 分類 할 수 있다.

그림 5는 195개의 경우 개발된 알고리즘을 적용한 해이고, 그림 6은 235개의 경우의 해이다.

5. 結 論

본 論文에서는 새로운 아이디어를 바탕으로 max($\bar{x}$, $\bar{y}$) TSP에 대한 tour Construction 알고리듬을 개발 하였다. 開發된 알고리듬을 Construction알고리듬으로 하여 기존의 Tour Improvement알고리듬을 적용 하였다. Euclidean TSP에서의 初期解 구성 알고리즘으로 가장 一般的으로 사용되는 最近距離隣接點(nearest-neighbor) 알고리즘 보다도 새로 開發한 알고리즘의 效率性이 max($\bar{x}$, $\bar{y}$) TSP의 경우 실행시간 면에서나, 구해진 解 면에서 優勢하다. 또한 改善 알고리즘으로서 Lin and Kerninghan을 적용하였을 때도 좋은 結果를 구할 수 있었다.

참 고 문 헌

1. 姜孟圭(1991), "네트워크와 알고리듬", 博英社.
2. 尹錫喆(1990). "계량 경영학", 經文社.
3. LEE J. BAIN, MAX ENGELHARDT. *Introduction to Probability and Nathmatical Statistics*(1987). Duxbury Press.
4. B. GOLDEN, L. BODIN, T. DOYLE and W. STEWART.JR. *Approximate Traveling Salesman Algorithms*(1979). Oper. Res. 28, 694-711.
5. E.L.Lawler, J.K.Lenstra, A.H.G.Rinnooy Kan, D.B.Shmoys, *The traveling salesman problem*(1983). JOHN WILEY & SONS
6. DON T. PHILLIPS, ALBERTO GARCIA -DIAZ. *Fundamentals of network analysis* (1981). Prentice-Hall, Inc.