論文 한국정밀공학회지 제9권 제3호 1992.9
Journal of the Korean Society of Precision Engineering Vol. 9, No.3 Sep. 1992.

Large-Scale TSP 근사해법에 관한 연구

유 형 선*, 김 현 승**

A Domain-Partition Algorithm for the Large-Scale TSP

Hyeong-Seon Yoo*, Hyun-Sng Kim**

ABSTRACT

In this paper an approximate solution method for the large-scale Traveling Salesman Problem (TSP) is presented. The method starts with the subdivision of the problem domain into a number of cluster by considering their geometric characteristic. Each cluster has a limited number of nodes so as to get a local solution. They are linked go give the least path which covers the whole domain and become TSPs solution with start-and end-node.

The approximate local solution in each cluster are obtained based on geometrical properties of the cluster, and combined to give an overall approximate solution for the large-scale TSP.

1. 서 론

TSP(Traveling Salesman Problem)[1,2]은 판매사원이 N개의 도시를 임의의 도시에서 시작하여 오직 한번씩 지나고, 다시 시작도시로 되돌아오는 가장 짧은 경로를 산출하는 문제이다. (Fig. 1. 참조) 기존에 발표된 TSP근사해법으로는 깊이 우선 탐색(Depth First Search), 너비 우선 탐색(Breadth First Search) 등의 Graph Search Method[3]와 MST(Minimum Spanning Tree)[4], Nearest Neighbor Algorithm[5] 등 여러가지가 있다. 이러한 대부분의 최적화 기법은 N×N의 거리행렬(Cost Matrix)[6]에 의존하는 해법으로, N의 증가시 N! 의 해의 경우의 수가 발생되어 해의 산출이 실질적으로 불가능하다.

이에 본 논문은 Large-Scale TSP[7]에서 보다 손쉽게 접근할 수 있는 근사해법에 대하여 연구하였다. 먼저 node수가 적으면 해의 산출이 용이하고, 가까운

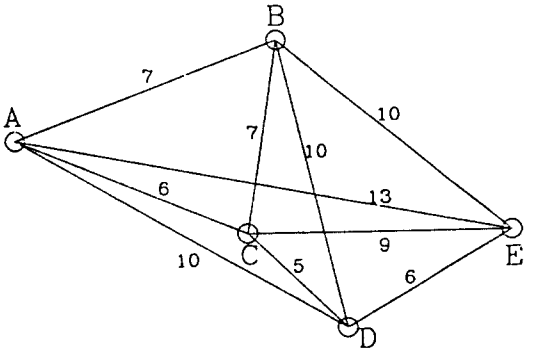


Fig. 1 Traveling Salesman Problem의 예

node들이 같은 영역(Cluster)에 속해 있을때 더욱 최적의 해가 산출된다는 가정아래, 전체 영역을 유한개의 cluster로 구분하고, 경계근처에 존재하는 node들의 cluster재지정을 통해 cluster를 좀더 뚜렷하게 구분하였다. 구분된 각각의 cluster는 자신의 시작 node,

* 인하대학교 자동화공학과 (정회원)
** 인하대학교 자동화공학과 대학원

끝 node, 중심 node가 이루는 내각과 node가 구성되는 분포 특징으로 구분되어 세가지의 고안된 새로운 해법이 적용 되었다. 또한 본 논문의 연구방향은 기존의 해법보다 빠른 시간과 더욱 짧은 경로산출에 있고, 다양한 경우를 고려하여 일반적으로 적용할 수 있도록 하였다.

본 연구에 사용된 프로그램 C 언어로 구성되었으며, 계산된 근사해는 Halo88 Graphic Function[5]을 이용하여 화면에 도시하였다.

2. 영역의 분할

2.1 분할의 목적

본 논문에서는 N개의 node를 가지는 TSP의 전체영역을 두 단계의 분할방법을 적용하여 M개의 cluster로 분할하여, 해법 적용시 예상되는 해의 경우의 수를 N ! 에서 Max_Sol 만큼의 수로 다음과 같이 줄일수 있었다.

$$N_T = N_1 + N_2 + N_3 + \sim + N_{M-1} + N_M \quad (1)$$

$$Max\_Sol = \max \{ N_i! \} \ (i = 1,2 \sim M) \quad (2)$$

여기서 N : 전체 node수

N_i : Cluster i내의 node 수

Max_Sol : Cluster 내에서 예상되는 최대 해의 경우의 수

두 가지 분할방법은 모두 정사각형 모습의 기초분할에서 시작된다. 정사각형의 기초분할은 분할후 분포의 방향성을 배제하기 위함이다. 기초분할시 방향성을 가지게 되면 이후에 설명되어지는 분할방법을 적용할 때 이상모형의 발생과 너무 많은 node들이 하나의 cluster에 포함되게 된다. N_i가 증가함에 따라 Max_Sol이 증가하므로 분할의 의미를 상실하게 된다.

2.2 분할방법 DG1

첫번째 방법으로 전체영역을 정사각형 요소의 기초분할후, (Fig. 2. 참조) 각 cluster들이 갖는 node들의 중심과 그 cluster들의 경계 근처에 존재하는 node들과의 거리판별을 통하여 node의 cluster를 재지정 하였다. 이때의 거리는 Euclidean Distance로 계산하였으며, 이 방법을 DG1 이라고 명명 하였다. 경계근처의 재배열 대상 node는 Fig. 3과 같다.

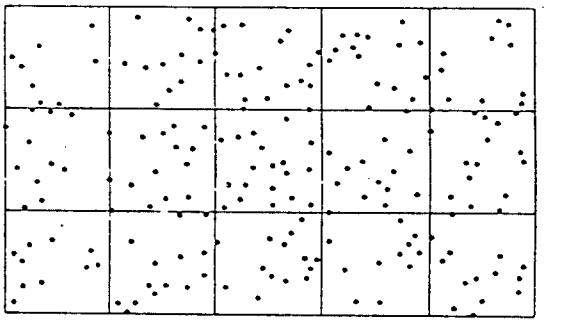


Fig. 2 영역의 기초분할시의 모습

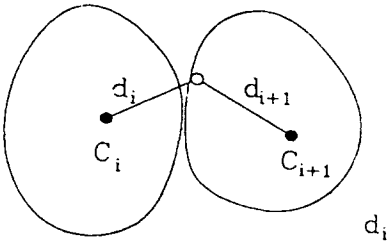


Fig. 3 경계선 부근의 재배열 대상 node

다음 procedure DG1은 분할방법 DG1의 주요과정이다.

Procedure DG1

```
LOOP i=1, N  (N : 전체 node 수)
LOOP j=1, M  (M : 전체 cluster 수)
     D_i = |Node_i (x, y) - Center_i (x, y)|
     END LOOP j
     G_Num = j (min {D_i} 일때 (j=1, ~, M))
     Group (i) = G_Num
END LOOP
```

위의 procedure DG1은 경계근처의 node를 가까운 cluster 중심으로 옮기므로 node를 잃게되는 cluster의 중심은, node를 얻게되는 cluster의 중심이 다가오는 거리보다 큰거리를 이동하므로 각 cluster를 구분할 수 있다. 위와같은 procedure DG1은 더이상 재지정 node가 없을때까지 반복된다. 이와같은 방법으로 경계선 부근의 node들을 재배열 한 뒤의 cluster는 Fig. 4와 같으며, 모든 cluster들의 local 중심점은 매 반복시 마다 다시 계산되어야 한다.

즉, $Center_i (x, y) = Modify \{Center_i (x, y)\}$

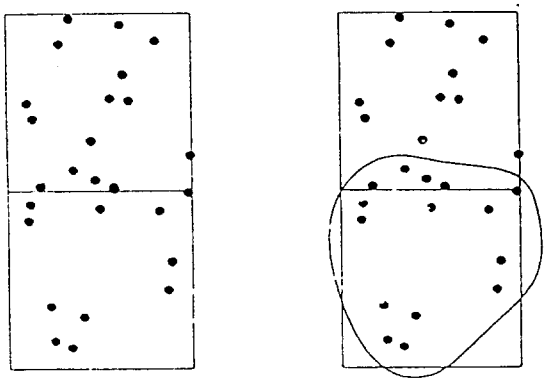


Fig. 4 기초분할시 cluster와 DG1 적용후 cluster

2.3 분할방법 DG2

두번째 방법으로 DG2라 명명한 분할방법은 DG1과는 달리 cluster의 중심에 의존하지 않고 node자신을 기준으로 정렬하는 방법이다. (Fig. 5. 참조) DG2는 먼저 cluster가 갖는 node와 그들의 중심 node와의 평균거리를 산출하고 중심node를 기준으로 반지름이 평균거리인 원을 생성시켜 경계근처의 외곽 node를 구분한다. 구분된 외곽 node들은 다음과 같은 조건에 의하여 지배열된다.

1. 임의의 node A가 cluster가 경계근처에 위치한다.
2. Node A가 일정거리(Bound_Value) 이내에 node B를 가진다.
3. Node A와 node B가 서로 다른 cluster에 속한다.

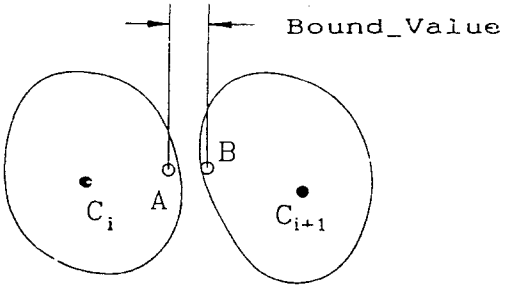


Fig. 5 분할방법 DG2 적용대상 node

```
Procedure DG2
  D = | A(x, y) - B(x, y) |
  IF (i≠j)
     IF (D≤Bound_Value)
           Cluster(A) = Cluster(B)
     Else
           Search Nearest node C
           Cluster(A) = Cluster(C)
End IF
```

여기에서 A(x, y) : Cluster의 경계근처 node
B(x, y) : Cluster의 node
C(x, y) : Node A와 가장 가까운 임의의 node

위의 procedure DG2는 외곽 node의 결정이 선행되므로 DG1에 비하여 적은 loop를 생성시킨다. DG2도 재지정후 중심 node를 modify 시킨다. 그리고 다시 DG2를 적용하여 더이상 재지정 node가 없을때까지 반복한다. 다음은 위와같은 DG1 혹은 DG2 과정을 거친 후 생성된 cluster의 모습이다. (Fig. 6. 참조)

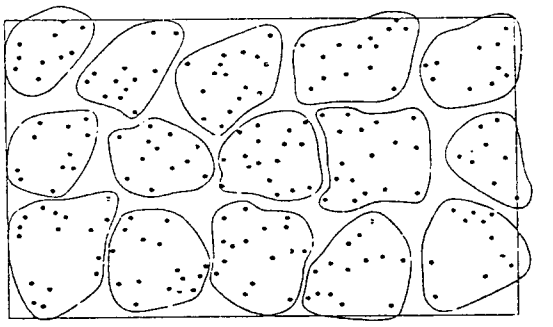


Fig. 6 DG1 혹은 DG2에 의한 영역분할의 예

이와같은 분할방법은 두가지를 조합하여 사용할 수 있다. 그러나 이 방법의 사용시, 대부분의 경우 다른 cluster를 파고드는 비정상적인 모양의 cluster가 발생되어 해법을 적용할 때, 불필요한 해를 산출하게 된다. 그러므로, 독립적인 분할방법의 적용이 바람직하다.

3. 정보량의 계산

3.1 초기 path의 생성

N개의 cluster로 분할된 전체영역은 하나의 근사해로 이루어져야 한다. 그러나 각 cluster 별로 각각의 근사해가 산출되므로, cluster 근사해를 연결하는 순서를

먼저 지정해 주어야 한다. 또한 이를 바탕으로 각 cluster가 갖는 정보량(시작 node & 끝 node)이 해법 적용 이전에 계산되어야 한다.

아래 Fig. 7에서 보는바와 같이 본 논문에서는 초기 path의 생성단계에서 이후에 설명되어지는 근사해법 S1을 이용하였으며, 각 cluster의 중심 node를 cluster를 대표하는 node로하여 시작 node를 첫번째 cluster 중심 node로 하고, 이와 가장 가까운 중심 node를 끝 node로 선택하였다.

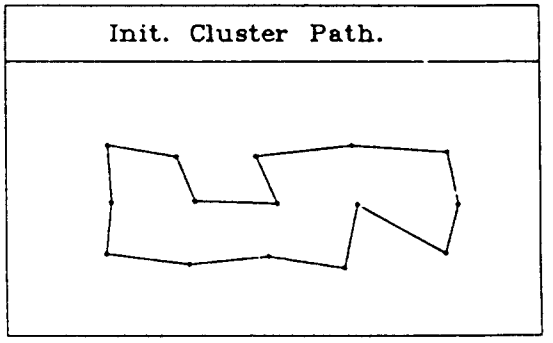


Fig. 7 근사해법 S1에 의해 생성된 초기해

3.2 연결점의 선정

위의 방법으로 구해진 초기 path 순서에 의하여 각 cluster별로 시작 node와 끝 node가 설정된다. 선행된 초기 path의 순서에 따라 전체 근사해가 구성되므로 각 cluster의 시작 node와 끝 node는 연속하는 두개의 cluster들의 중심 node를 잇는 선분의 중점 근처에 위치하여야 한다. 시작 node와 끝 node는 아래 그림과 같이 연속되는 두개의 cluster의 두 중심 node를 연결하고 중간 node CM(x, y)을 설정하여 cluster$_i$의 node 중 CM(x, y)과 가장 가까운 node가 끝 node, cluster$_{i+1}$의 node중 CM(x, y)과 가장 가까운 node가 시작 node로 결정된다. (Fig. 8. 참조)

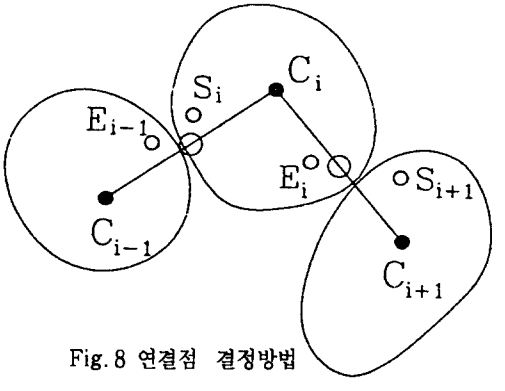


Fig. 8 연결점 결정방법

이와같은 방법으로 전체 cluster수와 같은 M개의 시작 node와 끝 node가 결정되어진다.

4. 근사해법의 연구

4.1 전체 근사해와 Cluster 근사해

TSP 근사해는 시작 node에서부터 전체 node를 지나 다시 시작 node로 돌아오는 폐곡선으로 구성된다. 그러나 본 논문에서는 유한개의 cluster로 분할되어 있으므로, 계산되어지는 각 cluster의 근사해는 시작 node로부터 끝 node로 이어지는 개곡선(Hamiltonian Circuit)[9,10]으로 구성된다. (Fig. 9. 참조)

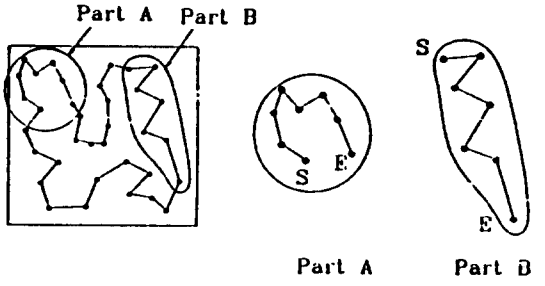


Fig. 9 전체 근사해와 cluster해와의 관계

위와같은 근거를 기준으로 시작 node, 끝 node의 위치와 cluster 중심좌표와의 관계를 통하여 세가지 근사해법을 도출해보았다. 이 세가지 방법은 각 cluster가 시작 node와 끝 node를 다른 형태로 가지고 있어 같은 해법을 적용하기에는 어렵고, 각 cluster에서 더욱 용이한 적용이 이루어지게 하기위해서 다음과 같이 구분하였다. 도출된 근사해법은 S1, S2, S3이라 명명하고 관단기준은 다음과 같다.

근사해법 S1 : 시작 node, cluster 중심 node와 끝 node가 이루는 내각의 크기가 90° 미만인 경우

근사해법 S2 & S3 : S1에서의 내각의 크기가 90°와 180° 사이일 경우

다시 근사해법 S2 & S3는 다음과 같이 분리될 수 있다.

$\Delta X = |\max\{X_i\} - \min\{X_i\}|$

$\Delta Y = |\max\{Y_i\} - \min\{Y_i\}|$

Check_Val=|△X-△Y|

Check_Val≤Bound_Val

근사해법 S2의 적용

Check_Val>Bound_Val

근사해법 S3의 적용

이와같이 도출되는 근사해법은 기존의 해법과는 달리 cluster의 node들이 갖는 특성값(node의 X-좌표값, Evaluation 값, 근접한 순서 등)에 의존하게 된다.

4.2 근사해법 S1

근사해법 S1에서는 시작 node와 끝 node를 잇는 직선과 수직으로 만나는 각 node의 X 좌표값이 node의 특성값으로 주어진다. 그러나 시작 node S와 끝 node S와 끝 node E가 너무 근접한 경우에는 너무나 큰 폭의 경로가 생성되어 불합리하다. (Fig. 10. 참조)

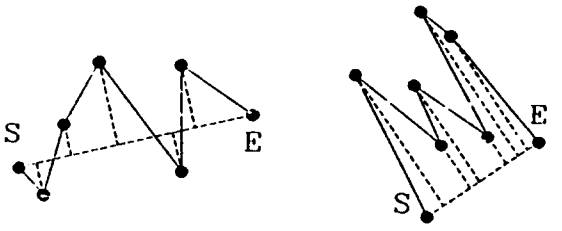


Fig. 10 근사해법 S1의 기본적 아이디어

Fig. 11. (a)에서 볼 수 있듯이 node S(x, y), E(x, y) 그리고 C(x, y)를 이용하여 이등분 node M(x, y)와 I(x, y)를 잇는 직선의 방정식으로 cluster를 다시 이등분한다. 이때 node M(x, y), I(x, y)가 이루는 직선의 방정식은 다음과 같다.

$$Y = \frac{(I_y - M_y)}{(I_x - M_x)}(X - M_x) + M_y \qquad (3)$$

식(3)의 X에 node좌표 X_{node}를 대입하여 Y_{node}값을 구한수, 계산되어진 Y와 Y_{node}을 기준으로 다음과 같이 재분할 한다.

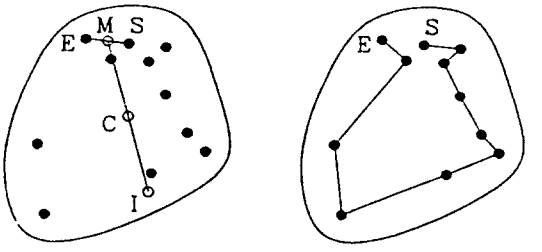
$Y \geq Y_{node}$ Sub_cluster A(node)

$Y < Y_{node}$ Sub_cluster B(node)

분리된 Sub_cluster A(node), B(node)에서 I(x, y)와 근접한 node SS(x, y), SE(x, y)를 찾아 각 Sub_cluster의 끝점 및 시작점으로 정의한다. 다음으로 시작 Sub_cluster에서 S(x, y)와 SS(x, y)가 이루는 직선의 방정식

$$Y_{sub} = \frac{(SS_y - S_y)}{(SS_x - S_x)}(X_{sub} - S_x) + S_y \qquad (4)$$

을 구한후 각 Sub_cluster의 node들과 식(4)와 수직으로 교차하는 점(X*, Y*)를 구한다. Sub_cluster A(node)의 보든 node들을 X*값의 순서대로 배열하면, X*에 의한하는 A(node)의 근사해가 구해진다. 같은 방법으로 Sub_cluster B(node)의 근사해도 구할 수 있다. 마지막으로 A(node)와 B(node)는 node SS(x, y)와 SE(x, y)를 연결하여 cluster의 근사해를 구할 수 있다. (Fig. 11. (b) 참조) 즉 이등분에 의한 node들의 위치확인과 X*값을 기준으로한 단순 순서 정렬로 해법이 이루어진다.



(a) 적용방법 (b) 적용 예

Fig. 11 근사해법 S1의 적용방법과 적용 예

4.3 근사해법 S2

근사해법 S2 적용시에는 cluster의 중심 node를 기준으로 사분면으로 분할한다. 시작 node와 끝 node가 포함된 사분면과 이외의 사분면은 Fig. 12와 같은 방법으로 사분면의 중심에 가까운 부분을 우선으로 순서를 정한 뒤, 진행순서와 각 사분면의 중심 node와의 관계로 다음에 설명되어지는 특성값 계산기준에 의하여 각 node들의 특성값이 계산된다. 이러한 값이 크기에 따라 근사특성값의 계산기준은 다음과 같다.

1. 각 순서에 따라 기본값을 가진다. (예10, 20, 30, 40)
2. 첫번째는 시작 node와의 거리를 합한다.
3. 마지막은 끝 node와의 거리를 뺀다.
4. 나머지 부분은 기본값에 이전 사분면의 중심 node와의 거리를 더하고 이후 사분면의 중심 node와의 거리를 뺀다.

이때 기본값은 지정된 사분면이 순서를 지정하기 위함이다. 이러한 해법 S2는 시작 node와 끝 node가 서로 반대편에 존재하고, 중심 node와 이루는 내각이

90°보다 크고 180°보다 작을때 적용되며, node들이 정방형으로 분포되어 있을때 적용이 용이하다. 결국 근사해법 S2도 node들의 특성값을 기준으로한 순서정렬에 의해 근사해가 계산된다. 이와같은 특성값에 의해 계산된 근사해는 다음 Fig. 13와 같다.

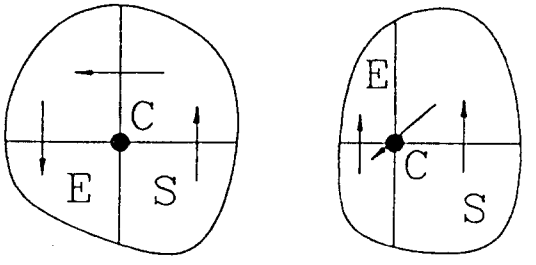
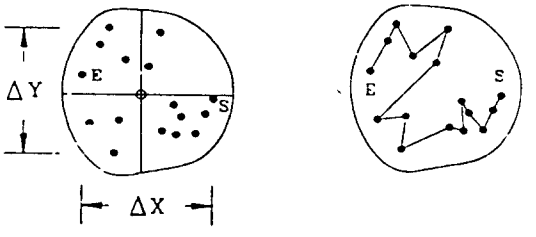


Fig. 12 사분면의 순서



$\Delta Y \cong \Delta X$

Fig. 13 근사해법 S2의 적용방법과 적용 예

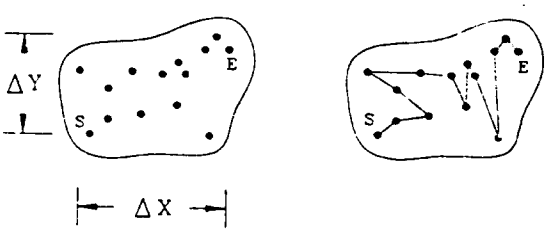
4.4 근사해법 S3

근사해법 S3는 S2와 유사한 분포를 갖는 cluster에 적용되며, node의 특성값으로는 어떤 기준으로부터의 가까운 정도가 채택된다. 즉, 시작 node와 이에 가까운 node를 다음 node로 취한후, 지나온 node들의 중심점을 구한다. 구한 중심점에서 가장 가까운 또다른 node를 선택한다. 이와같은 방법으로 마지막 node를 취할때까지 반복한다. 이와같은 방법은 node들의 분포가 타원형으로 형성되어 있을 때 적용이 용이하며, 적용예를 Fig. 14에 나타내었다.

4.5 근사해의 보정

위의 근사해법 S1, S2, S3의 과정을 거친 근사해는 보정이 필요한 부분을 부분적으로 가지고 있다. (Fig. 15. (a) 참조)

보정방법으로는 기존의 r-optimal[11] 방법 등이 있으나, 본 논문에서는 세가지 해법으로 구성된 전체 근사해를 기준으로 첫번째 node에서부터 5개의 순차적 node를 선정하여 1번과 5번을 고정하고 이루는 path의 경우중 가장 가까운 path가 갖는 node의 순서로 node의 순서를 변화시켰다. 아래 그림에서 보듯이 근사해 BCDEA는 통과하는 4개의 점CDEA 중에서 C를 시작으로 CDEA를 CEDA로 재정렬 시킬 수 있다. (Fig. 15. (b) 참조)



$\Delta Y < \Delta X$ OR $\Delta Y > \Delta X$

Fig. 14 근사해법 S3의 적용방법과 적용 예

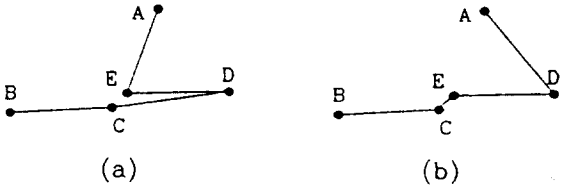


Fig. 15 보정이 필요한 근사해의 일부분

5. 프로그램의 구성

본 연구에 사용된 프로그램은 Fig. 16과 같이 구성되어 있으며, 각 subprogram은 다음과 같은 역할을 한다.

- Subprogram Input()
  프로그램에서 필요한 입력값을 읽는 부분이다. 입력값은 전체 영역의 크기, node들의 x, y값, 분할하려는 cluster의 갯수 등이 있다.
- Subprogram Near_Move()
  분할방법 DG1과정으로 경계근처에 존재하는 node와 각 cluster 중심점과의 측정하여 node의 cluster번호를 재지정 하는 과정이다.
- Subprogram N_Cent
  분할방법 DG2과정으로 node자신을 기준으로 node의

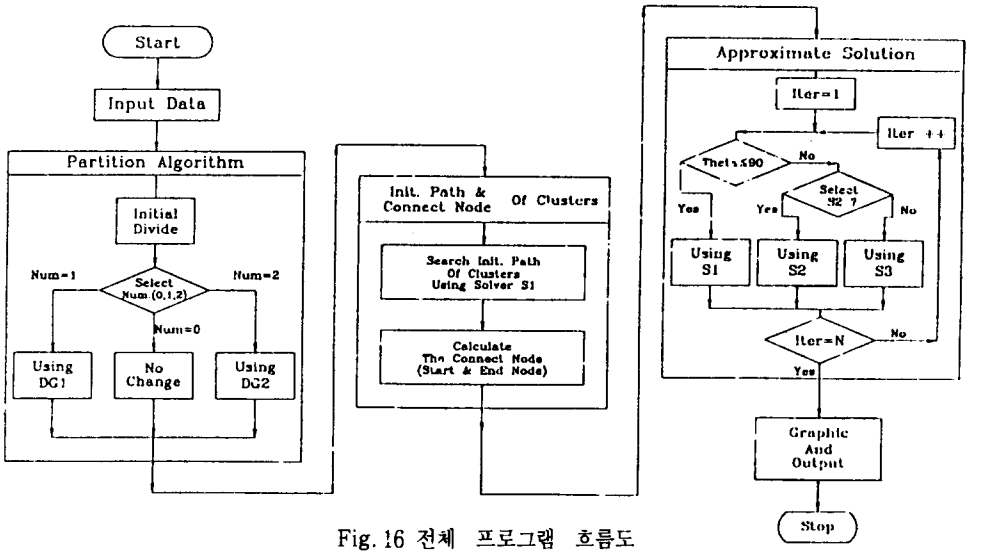


Fig. 16 전체 프로그램 흐름도

cluster번호를 재지정 하는 과정이다.

- Subprogram Init_Tour()
  초기 cluster의 순서를 지정하는 과정으로 cluster의 중심node를 기준으로 초기 근사해를 구하는 과정이다.
- Subprogram Con_g()
  초기 Init_Tour()에서의 근사해를 바탕으로 cluster의 연결점(시작 node, 끝 node)을 구하는 과정이다.
- Subprogram Order1()
  근사해법 S1의 적용과정으로 수직으로 만나는 X의 값에 의존하는 근사해가 계산된다.
- Subprogram Eval()
  근사해법 S2의 적용과정으로 사분면 사이의 관계로 이루어진 특성값에 의하여 근사해가 구해진다.
- Subprogram Eval_()
  근사해법 S3의 적용과정으로 판단된 가까운 정도의 특성값에 의하여 근사해가 구해진다.
- Subprogram Total_Check()
  근사해의 보정과정으로 줄일 수 있는 error를 최소화 한다.
- Subprogram Dwg()
  Halo '88 Graphic function을 이용하여 화면에 결과를 출력하는 과정이다. 또한, Print를 위한 출력화일을 생성시킨다.

6. 결과 및 실행의 예

6.1 실행의 결과

실험의 모델로는 200개의 node를 선정하였으며, 그 영역을 X는 0.0에서 1.0, Y는 0.0에서 0.6으로 선정하였다. 그리고 node의 분포를 2가지 형태로 선정하였다. 또한 근사해의 비교를 위하여 기존에 발표된 Cheapest Insertion Method[5]와 r-optimal으로 이루어진 근사해법을 이용하였다. 기존해법의 흐름도는 Fig. 17과 같다. 실험결과는 다음과 같은 두가지 관점에서 분석될 수 있다. 첫번째는 실행시간의 차이이다. 표 1에서 보는바와 같이 기존해법에 비하여 적은 실행시간이 나타났다. 결과를 먼저 살펴보면 기존해법은 37sec의 실행시간이 나타났고, 본 논문의 해법은 4~6sec의 적은 시간으로 나타났다. (표1참조) 실행시간의 구성을 살펴보면, 본 논문의 프로그램은 분할방법에서 2~3sec의 실행시간이 나타나고, 근사해법에서 나머지 시간이 나타났다. 반면에 기존해법에서는 Cheapest Insertion Method에서만 20sec의 시간이 소비되었다. 이방법의 해를 는 기준을 살펴보면 다음과 같다.

식(5)에서 보는바와 같이 subtour T에 가장 가까운 node x를 잡고, T의 각 node와 비교하여 최적 COST를 갖는 새로운 subtour T를 만들어 낸다. 본인이 count해본 결과에 의하면 Insertion Method에서만

백만번 이상의 iteration이 나타난다. 그러나 본 논문에서는 iteration이 가장 많은 분할방법에서 1회 반복에 2700회씩 최대 15회 반복(node의 밀도에 영향을 받음)이므로, 실행시간에 차이가 날수밖에 없다. 참고로 386-SX에서 단순 loop의 시간 측정결과 50,000 iteration은 2sec, 1,500,000회에서는 7sec가 측정되었다. 결국 large-scale TSP의 경우에 실행시간에 있어서는 분할하는 방법이 효과적임을 증명할 수 있다.

두번째로는 계산된 해(거리)의 관점이다. 표1에서도 보듯이 DG1 혹은 DG2를 적용했을때가 기초분할시보다 적은 값을 산출해 냈고, 기존의 해법과는 뚜렷한 차이없이 근사한 값으로 나타났다. 그러나, 근사해법 S1, S2, S3와 같이 각 node의 특성값으로 해를 푸는 방법이 기존의 search method의 효과를 발휘함을 알수 있다. 또 분할방법 DG1 혹은 DG2중에서 무엇이 더 효과적이라고 말할 수 없으며, 표 1에서 나타나듯 분포된 node들의 밀도등의 분포상태에 영향을 받는다. 이와 같은 결과로 볼때 본 논문의 전체적인 해법을 기존해법이 갖는 해에 근접한 값을 산출하며 더불어 기존의 해법보다 빠른 실행시간을 갖는 TSP의 근사해법이라 할 수 있다.

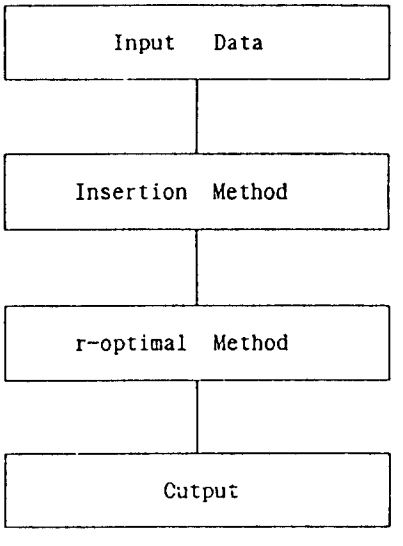


Fig. 17 이용된 기존해법의 흐름도

6.2 Bound의 설정 및 근사해의 검증

참고로 예상되는 근사해가 존재할 수 있는 최소값(Lower Bound)과 최새값(Upper Bound)을 추측하여 계산된 근사해의 타당성을 조사하였다.

■ **Lower Bound의 설정**

- 존재할 수 있는 최소값
- Lower_Bound = Cluster_Path
  Cluster_Path = Cluster 중심 node를 잇는 초기 근사해.
- 문제점 : 시작과 끝 cluster에 의하여 path가 변하므로 Lower_Bound가 유동적임.

■ **Upper Bound의 설정**

- 인정할 수 있는 최대값
- Upper_Bound = A_Dist * Node수
  A_Dist = Aver{C_{ij}} (C ->Cost Matrax)
- 문제점 : 시작과 끝 cluster에 의하여 path가 변하므로 Lower_Bound
- 문제점 : 영역의 크기가 증가함에 따라 A_Dist의 값이 변함

이와같이 두가지의 Bound조건에 의하여 사용된 모델의 Bound는 다음과 같다.

모델 1 : Lower Bound : 2.961006, 2.831063, 2.972010
Upper Bound : 81.90007

모델 2 : Lower Bound : 3.126582, 3.140164, 3.071073
Upper Bound : 85.79455

위의 Bound와 표1.의 근사해와 비교해 볼때, 근사해가 설정된 Bound이내에 존재함을 알수 있다.

6.3 실행의 예

다음은 2가지 모델에 대하여 분포상태, 기초분할후 해법적용, 분할방법 DG1적용후 해법적용, 분할방법 DG2적용후 해법적용의 순서로 실행된 결과를 출력하였다. (Fig. 18~27)

6.4 결과분석 및 향후과제

Fig. 18~27의 결과와 같이 기존과 유사한 해가 산출되었다. 그러나 본 논문에서 적용된 모델의 분포형태가 일반적으로 고른 분포를 형성하여 실제 문제와는 다를 수 있다. 특히 영역의 모양이 다름으로해서 발생되는 빈 cluster의 발생과 밀도에 의한 node 갯수의 문제도 제기된다. 그러나, 여러 영역을 본 논문의 예와같은 영역으로 mapping시켜 적용하면 약간의 차이는 발생되지만 유사한 해를 산출할 수 있다.

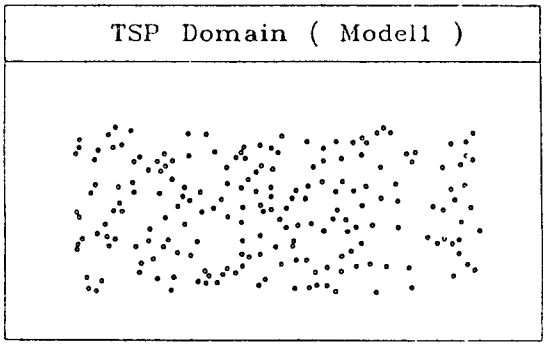


Fig. 18 node의 분포상태(모델 1)

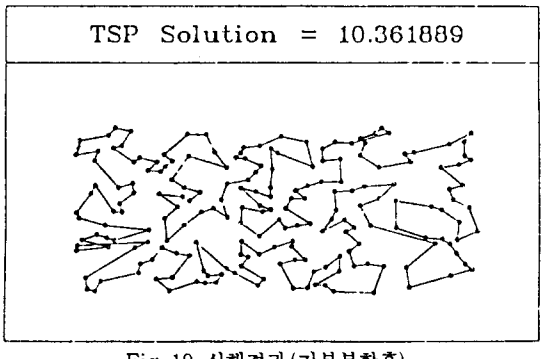


Fig. 19 실행결과(기본분할후)

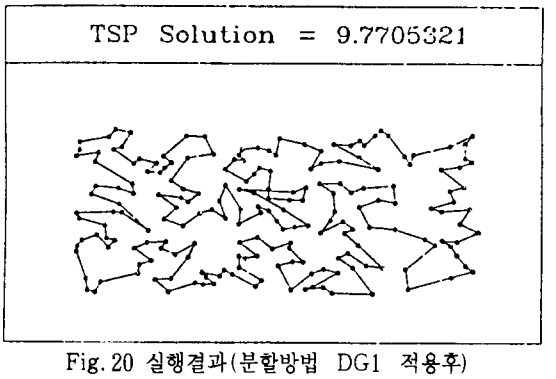


Fig. 20 실행결과(분할방법 DG1 적용후)

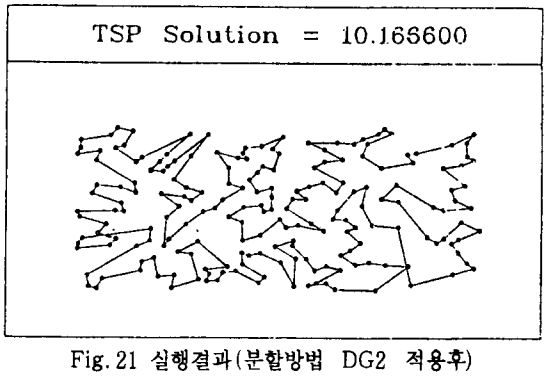


Fig. 21 실행결과(분할방법 DG2 적용후)

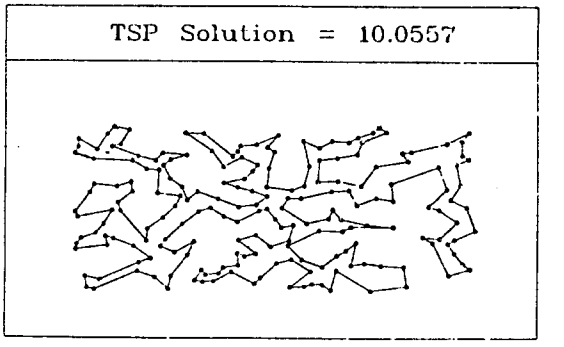


Fig. 22 실행결과(기존해법)

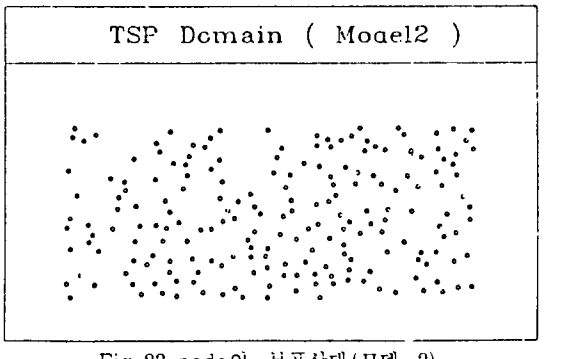


Fig. 23 node의 분포상태(모델 2)

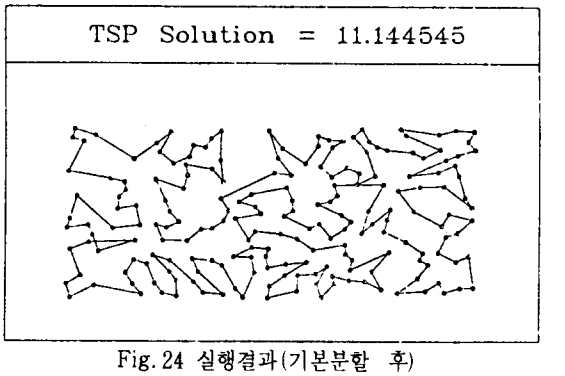


Fig. 24 실행결과(기본분할 후)

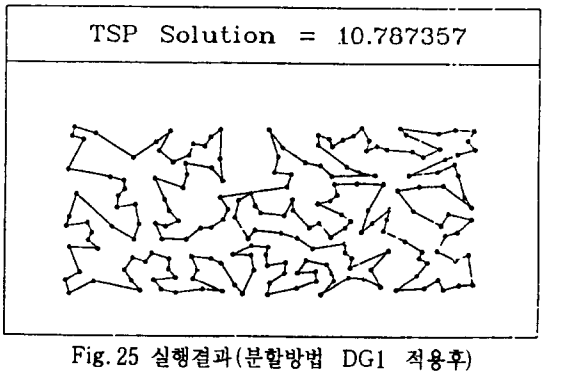


Fig. 25 실행결과(분할방법 DG1 적용후)

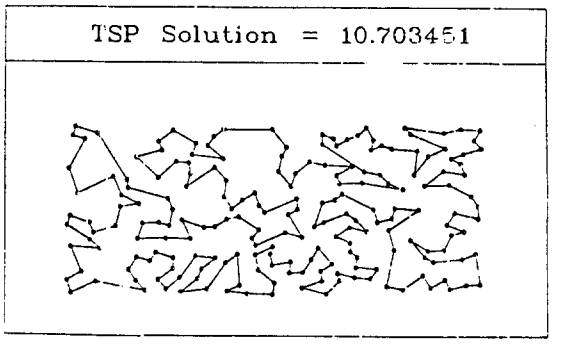


Fig. 26 실행결과(분할방법 DG2 적용후)

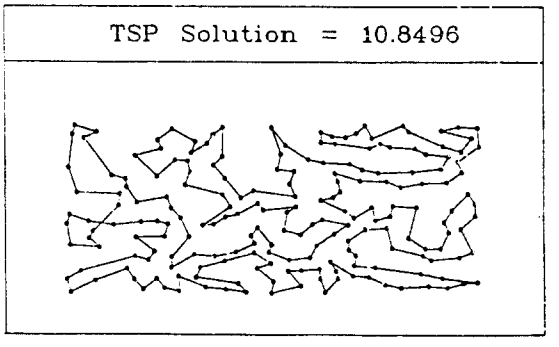


Fig. 27 실행결과(기존해법)

표 1. 실행결과

실행예 1	근사해(실행시간)	기존의 해법(실행시간)
기초분할(15)	10.361889(2 sec)	10.0557 (37 sec)
DG1 적용(15)	9.770533(6 sec)	
DG2 적용(15)	10.166605(5 sec)	

실행예 2	근사해(실행시간)	기존의 해법(실행시간)
기초분할(15)	11.144546(2 sec)	10.8496 (37 sec)
DG1 적용(15)	10.787357(4 sec)	
DG2 적용(15)	10.703451(6 sec)	

향후 연구 과제로는 문제영역의 모양과 분포특성을 고려한 영역분할방법이 있다. 또한 보정방법에 대한 연구도 검토되어야 한다.

참고문헌

1) C. H. Papadimitrion, K. Steiglitz, "Some Example of Difficult Traveling Salesman Problem", Operation Research, vol. 26, No. 3, pp. 434~443, 1978.
2) R. G. Parker, R. L. Rardin, "The Traveling Salesman Problem : An Update of Research", Naval Res. Logistic. Q. Vol. 30, pp. 69~96, 1983.
3) 유석인, "인공지능 원론", 교학사, pp. 63~73, 1988.
4) Micheal Held, Richard. M. Karp, "The Traveling Salesman Problem and Minimum Spanning Tree", Mathematical Programming, pp. 6~26, 1971.
5) A. M. Frieze, G. Galbiati, F. Maffioli, "On the Worst-Case Performance of Some Algorithms for the Asymmetric Traveling Salesman Froblem", Networks, Vol. 12, pp. 23~39, 1982.
6) Nicos Christofides, "Graph Theory-An Algorithmic Approach", Academic Press Inc, pp. 236~281, 1975.
7) H. Crowder, M. Padberg, "Solving Large-scale Symmetric Traveling Salesman Problem to Optimality", Management Science, Vol. 26, pp. 495~509, 1980.
8) Halo '88 "Graphic Kernel System", (Library Reference & Language and Device Reference).
9) J. P. Norback, R. F. Love, "Heuristic for the Hamiltonian Path Problem in Euclidian Two Space", J. of Opl. Res. Soc. Vol. 30, No. 4, pp. 363~368, 1979.
10) M. Gondran, M. Minoux, "Graphs and Algorithms", Wiley, 1984.
11) Nicos Christofides, Samuel Eilon, "Algorithms for Large-scale Traveling Salesman Problems", Opl. Res. Q, Vol. 23, No. 4, pp. 511~518, 1972.