CAD에 의한 徑路 發見 알고리즘에 있어서 發見的 探索에 관한 硏究

(A Study on Heuristic Search in the Path Finding Algorithm by CAD)

李 天 熙*, 朴 炳 哲**

(Cheon He Yi and Byung Chul Park)

要 約

본 논문에서는 여러 경우의 발견적함수가 논의되었다. 경로 찾기 알고리즘에 있어서는 목표까지 거리의 정확한 평가자를 구하는 것이 중요하다. 정확한 평가자가 존재하는 한 어느 공간이든 문제가 풀리는 영역이지만, 부정확한 평가자들로서도 가능한 영역에 흥미가 있으며 따라서 이 논문은 발견적정보의 효과적인 사용의 필요성을 느끼게 하여주는 경우들을 다루었으며, 이것은 최단경로 문제에 있어서 계산적인 접근에 사용하면 좋다.

Abstract

In this paper, many kinds of heuristic functions were discussed. It's important factor in the path finding algorithm to find an accurate estimator of distance from the goal.

Any space for which an accurate estimator exists is a domain of its problem being solved. Only domains with inaccurate estimators are interesting, so this paper deals with these cases for which the efficient use of heuristic information is necessary. It has been productive to use a computational approach to the shortest path problem.

I. 序 論

Michie, Doran[1]과 Nilsson, Hart, Raphael[2]등이 연구한 발견적 경로 알고리즘들은 만일 경로가 하나 존재하고 그라프가 한정된 것이라면 반드시 경로를 발견하게 되나 무한정한 그라프에서는 그렇치 못하다. $\omega=\infty$라고 가정한 Graph Traverser[1]에서 발견적함수 h를 사용하였는데 이는 어느 한 노드에 도달함에 있어서 낭비가 없는 가장 적당한 거리만을 사용키 위한 것이며 실제로 만일 h가 목표까지 거리의 정확한 견적이라면 그것은 가장 목표에 가까운 노드를 가리키게 되며 목표까지 남은 거리는 목표까지 찾아감에 있어서 지나쳐야 할 가장 적은 노드들을 결정하면 된다. 여기서 항상 문제가 되는 것은 발견적함수의 정확도이지만 우리가 관심을 갖는 것은 부정확한 평가자(estimator)들에 의하여 풀리는 문제영역보다는 부정확한 평가자들로서도 가능한 영역들이며 따라서 본 논문은 Hart, Nilson과 Raphael 등이 사용한 알고리즘과 비슷하나 발견적 정보의 효과적인 사용의 필요성을 더 느끼게 하여 주는 경우들(부정확한 평가자들로서도 가능한)을 다루었다.

*正會員, 淸州大學校 電子工學科

(Dept. of Electron. Eng., Chung Joo Univ.)

**正會員, 成均館大學校 工科大學 電子工學科

(Dept. of Electron. Eng., Sung Kyun Kwan Univ.)

接受日字 : 1984年 1月 31日

Ⅱ. 무한정 2진 Tree에서 병렬탐색

만일 h가 정확하다면 $\omega \geq 1$인 동안 발견적 경로 알고리즘에 의한 탐색은 효과적이며 이것은 가능한 가장 적은 노드를 탐색하게 되며 특정된 방향성의 구조에 관계없이 응용될 수 있다. 그래서 평가함수 f안에 정확한 발견적함수 h의 사용은 $1 \leq \omega < \infty$인 동안 최적이다. 만일 우리가 사용할 영역에 필요한 발견적 정보가 없다면 二進의 tree공간[8]을 통해서 h값이 0이 된다. 즉 우리가 사용하는 평가함수 f는 아래와 같이 된다.

① $f = g + \omega \cdot h = g \quad (h = 0) \quad 0 < \omega < \infty$ (1)

② $f = h = 0 \quad \omega = \infty$ (2)

g[7]의 사용은 $h = 0$일 때 $\bar{S}$ (개방노드들 : edge를 따라 S로 부터 도달될 수 있는 노드들) 안에서 모든 node들이 묶일 곳에서 탐색인 동안은 병렬탐색을 구성한다. 각 반복단계에서 발견적 경로 알고리즘의 2단계는 묶인 노드들로 부터 임의로 선택될 것이다. 그러므로 식(2)는 임의의 탐색을 만들어 낸다. 만일 tie-break 법칙 대신에 FIFO(first in/first out)이라면 병렬탐색이 되고 LIFO(last in/first out)는 depth first 탐색[3]이 될 것이다.

[정리 1] 무한한 2진 tree에 있어서는 병렬탐색은 대략 $2^K + 2^{K+1} - 1/2$ 노드들을 필요로 하고 root로부터 Klevel째의 한 노드를 발견하기 위하여 확장된다.

〈증명〉 직경 K(root로 부터 최대길이)의 2진 tree에서 노드들의 수는 $B_K = 2^{K+1} - 1$이다. 노드가 K번째 level을 따르는 어느 곳에든지 있을 가능성이 동일하기 때문에 평균적으로 아래와 같은 (3)식을 사용하여 B_{K+1} 노드들 까지 $B_K + 1$을 찾아야 한다.

$$1/2(B_K + 1 + B_{K+1}) = 1/2(2^{K+1} + 2^{K+2} - 1) = 2^K + 2^{K+1} - 1/2 \quad (3)$$

각 level은 root로부터 K거리인 곳에서 2^K 노드들을 갖는다.

$$B_{K+1} = B_K + \{(K+1)\text{level에 있는 노드 수}\} = B_K + 2^{K+1} = 2^{K+1} + 2^{K+1} - 1 = 2^{K+2} - 1 \quad (4)$$

따라서 2진 tree의 병렬탐색에서 얼마나 많은 노드들이 방문되어 지는 평균적으로 (4)식 결정에 따른다. 이것은 복잡한 문제공간에서 전형적인 행동자인 root로부터 거리로서 지수적으로 변화한다. 대조적으로 root로부터 목표 노드까지, K level을 발견함에 있어서 임의의 탐색에 의하여 방문되는 노드들의 예상되는 수를 시험할 수 있는 알고리즘을 살펴보자.

Ⅲ. Searching Algorithm

최적 경로를 찾는데 있어서 탐색범위가 최소가 되도록 하는 노드들까지만 확장시키기 위해서는 탐색알고리즘[2]은 다음으로 확장되는 노드에 관하여 가능한 알려진 결정을 계속적으로 만들어야 한다. 만일 그것이 최적 경로가 아닌 노드까지 확장된다면 쓸데없는 노력을 들인 것이 되며 만일 그것이 최적 경로가 될 수 있는 노드들을 무시한다면 경로를 발견하지 못할 것이다. 효과적인 알고리즘은 분명히 다음으로 확장될 수 있는 하나를 결정하는 유용한 노드들을 평가할 수 있는 어떤 방법을 필요로 한다.

어떤 평가함수 $f^*(n)$이 어느 노드 n에서 계산될 수 있다고 가정하고 탐색알고리즘이 그러한 함수를 사용할 것인지를 설명하여야 한다. f^*의 가장 적은 값을 갖는 유용한 노드가 다음으로 확장될 노드라는 취지에서 정의된 것이 평가함수 $f^*(n)$이다. 어떤 subgraph Gs와 어느 goal set T에 대해서 f(n)을 출발 노드 s로부터 출발해서 n의 택해진 목표 노드까지 n을 통해서 가도록 되어 있는 최적 경로의 실제값이라 하자. $f(s) = h(s)$인 경우는 s로부터 s의 택해진 목표 노드까지 임의의 최적경로의 값인 경우이다. 사실 최적경로상에 모든 경로에 대하여 $f(n) = f(s)$이다. 그리고 최적경로가 아닌 모든 노드 n에 대하여 $f(n) > f(s)$이다. 그러므로 비록 f(n)가 우선권이 있고 사실 f(n)의 참값의 결정이 주된 문제일지라도 평가함수 $f^*(n)$으로서 f(n)의 평가자를 사용하는 것이 타당할 것이다. 우리는 노드 n을 통과한 최적경로의 비용 f(n)이 적절한 평가함수 $f^*(n)$에 의해서 평가될 때 탐색알고리즘의 어떤 특성을 나타내 보일 수 있다. f(n)은 다음의 두 부분으로 표시된다.

$$f(n) = g(n) + h(n) \quad (5)$$

g(n) : s로부터 n까지 최적경로의 실제비용

h(n) : n부터 n의 택해진 목표노드까지 최적경로의 비용

만일 지금 우리가 g와 h의 평가를 갖고 있다면 우리는 f의 평가를 형성하도록 그들을 더할 수 있다. $g^*(n)$을 g(n)의 평가라 하면 $g^*(n)$을 위한 분명한 선택은 s로 부터 지금까지 알고리즘에 의해서 발견된 가장 적은 비용을 갖는 n까지 경로의 비용이다. 이때에 물론 $g^*(n) \geq g(n)$이다.

[정리 2] 무한한 2진 tree에 있어서 임의의 탐색은 root로부터 1level 떨어진 노드를 발견하기 위해서 방문하는 노드들의 수를 제한받지 않는다. 발견적 경로 알고리즘은 노드가 단지 1만 떨어져 있고 결과적으로 이 level 까지 그것의 탐색을 제한하지 않는다는 것을 전제하고 있지 않다.

〈증명〉

E = 방문한 노드의 예상되는 수

r_i = 정확히 ilevel에서 목표 노드를 발견할 가능성

p_i=i번째 level까지 목표 노드를 발견할 가능성

l_i=i번째 level전에 어느 level상에 서든 목표 노드를 발견하지 못할 가능성

$$E = 1 + \sum_{i=1}^{\infty} i \cdot r_i \tag{6}$$

1은 root 노드를 위한 것이다.

$$r_i = P_i \cdot l_i \tag{7}$$

$$P_i = \frac{1}{i+1} \tag{8}$$

2진 tree에 걸친 발견적 경로 알고리즘의 각 level로서 한 노드가 $\widetilde{S}$ (개방노드들)로부터 옮겨지고 이미 방문된 노드 S안에 놓인다. 그러나 두개의 노드는 $\widetilde{S}$ 안에 놓인다. 이것은 level i에서 $\widetilde{S}$ 안에 i+1 노드가 있다는 것을 의미한다. f= 0 라 임의의 tie-breaking 경우에 있어서는 그들 모두가 동일하게 선택되어 질 것이다. 더 나아가 목표 노드는 root로 부터 1만 떨어져 있고 그것은 발견적 경로 알고리즘에 의해서 발견될 때 까지 set $\widetilde{S}$안에 항상 있기 때문에 Pi= 1/(i+1)이며 귀납법에 의하여 $l_i = 1/i$이다.

$$l_i = 1 - \sum_{j=1}^{i-1} r_j, \quad l_1 = 1 \tag{9}$$

$$l_2 = 1 - r_1 = 1 - p_1 l_1 = 1 - \frac{1}{2} \cdot 1 = \frac{1}{2} \tag{10}$$

$l_K = 1/K$이라 가정하면 (즉 목표노드가 root로 부터 1만큼 떨어져 있을 때) $l_{K+1} = 1/(K+1)$이어야 한다.

$$\begin{aligned} l_{K+1} &= 1 - \sum_{j=1}^{K} r_j = 1 - \sum_{j=1}^{K-1} r_j - r_K \\ &= l_K - r_K = l_K(1 - p_K) \\ &= \frac{1}{K}\left(1 - \frac{1}{K+1}\right) \\ &= \frac{1}{K+1} \end{aligned} \tag{11}$$

그러므로

$$\begin{aligned} E &= 1 + \sum_{i=1}^{\infty} \left(i \cdot \frac{1}{i+1} \cdot \frac{1}{i} \right) \\ &= 1 + \sum_{i=2}^{\infty} \frac{1}{i} = \sum_{i=1}^{\infty} \frac{1}{i} \end{aligned} \tag{12}$$

이 결과는 Gambler의 ruin 문제들[4]과 비슷하다. 본질적으로 공간은 너무 빨리 커지고 목표 노드를 초기적으로 발견하는데 있어서 운이 좋지 않을 때는 $\widetilde{S}$ (개방노드)의 확장된 범위로 확산된다.

정상적으로 탐색은 시간 또는 공간제한들에 의해서 제한되었는데 이것은 어떤 최대 길이까지 우리의 무한한 공간을 제한하는 것과 유사한 것이다. 만일 우리가 level K의 2진 tree에서 목표 노드를 발견하기를 원한다면 $(2^{K+1}-1)$ 노드들의 최대값이 탐색되어야 한다. 만일 이 노드들 각각이 동일 가능성으로서 목표 노드라면 방문된 $1/2(B_K+B_0)$ 또는 2^K 노드들에 대하여 기대되는 같은 값을 만들어 내기 위해서는 철저한 반복탐색으로서 가능하다. 각 방법은 노드들의 다른 그룹들에 대해서 더나은 수행을 하게 될 것이다. 병렬탐색은 바로 더 가까운 노드들을 방문하고 root 근처에 목표 노드들에 대해서는 임의의 또는 DFS 혹은 LIFO 탐색보다 더 좋은 예상되는 값을 갖는다. 이 이론에서 발견적 경로 알고리즘은 목표가 level 1에 있다는 것을 알지 못하기 때문에 한정된 가능성을 가지고 해답을 넘어선 공간부분들을 탐색한 것이다. 이것의 수정으로 목표가 level K상에 있는 것을 알게 된다. 이것이 알려졌을 때 depth first[3] 또는 backtrack 방법[5][6]이 최적이다. 알고리즘은 K의 level까지 내려 갈 것이고 만일 이것이 목표 노드인지 알려고 검사할 것이다. 만일 그것이 한 level 역행이 아니었다면 level K에 다음 노드까지 더 내려간다. 그것은 역행을 계속해서 그것이 목표를 발견할 때까지 levelK상의 다음 노드까지 앞으로 나아간다. 이 탐색패턴이 가능한한 K번째 level 위에 노드들을 찾기 때문에 그것은 방문된 노드들의 예상되는 수의 관념에서 최선이 되어야 한다. 만일 목표 노드가 실제로 level 1에 있었다면 최악의 탐색패턴이 될 것이다. 왜냐하면 첫번째 시도에서 목표를 발견하거나 또는 목표 노드까지 돌아오기 전에 tree의 절반을 찾아야 되기 때문이다. 병렬탐색은 신중한 계획이며 그 level까지 모든 subtree를 조사하는데 주의를 기울이는 동안 목표를 포함하는 tree의 아래 부분으로 내려가지 않도록 해 준다.

Ⅳ. 발견적 탐색의 오차

일반적으로 우리가 갖고 있는 정보에는 약간의 오차가 있으므로 발견적 함수를 최선으로 사용함으로서 더 많은 전형적인 예들을 풀기를 희망한다. 발견적 경로 알고리즘을 사용하는 2진 tree 공간내에서 이 문제를 다루도록 하겠다. 수치문제들에 있어서 오차 분석의 관점에서 최악의 경우의 분석을 할 것이다.

h : 완전한 평가자

ε : 오차의 범위 0, 1, 2, 3……(정의 정수로 표시)

h^*: 실제의 발견적 함수

$$h - \varepsilon \le h^* \le h + \varepsilon \tag{13}$$

우리는 위의 제한에 따르는 h^*의 값을 선택할 것이다. 그러나 그러한 식으로 발견적 경로 알고리즘을 가장 큰 범위까지 잘못 적용하기도 한다. 이것을 하는데 있어서 우리는 발견적 경로 알고리즘은 해답 경로를 벗어나는 노드들의 경우에 있어서 항상 최악의 노드들을 선택할 것이라고 가정한다.

이 분석의 예가 그림 1이며 여기서 발견적 경로 알고리즘은 평가자로서 h^*함수를 바로 사용한다. 탐색의

순서는 5단계 안에서 도착되는 목표인 X로서 노드들 안쪽의 수들에 따른다. 가능한 한 나쁘게($\varepsilon = 1$) h^*를 만들기 위하여 우리는 최단경로상에 각 노드까지 ε를 더한다. 그리고 우리는 최단경로 떨어진 각 노드로부터 ε를 뺀다. 만일 h 그 자신이 사용되었다면 $\omega \geq 1$인 동안 발견적 경로 알고리즘에 의한 탐색은 최적이며 가능한 가장 적은 노드들을 방문하게 되므로 최단경로상의 3노드들만 방문할 것이다.

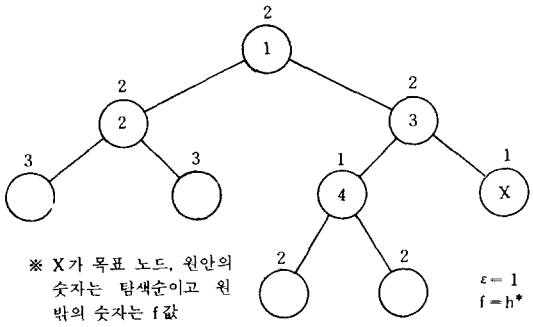

그림 1. 分析의 例示(Ⅰ)
Fig. 1. Example of analysis(Ⅰ).

기본적인 의문들의 평가자(estimator)로서 h^*와 $g+h^*$사이의 비교이다. 이것을 비교함으로서 오차 분석과 약간의 암시를 더 얻기 위하여 그림 2의 비교작업을 살펴보자. 그림 2 (b)에서 $f=h^*+g$를 사용하여 발견적 경로 알고리즘을 시험하여 보기로 한다. 목표 노드 X에서

$h(X)=0, g(X)=1$

$h^*(X)=h(X)+\varepsilon=0+2=2$

$f(X)=g+h^*3$;

노드 2에서 다음 값을 갖고 있는 동안에

$h(2)=2, g(2)=1$

$h^*(2)=h(2)-\varepsilon=2-2=0$

$f(2)=g+h^*=1$

노드 3은 $h(3)=3, g(3)=2$의 값을 갖는다. 그래서 양쪽다 그것의 전 임자 노드 2의 값들 보다 1이 증가 되었다. 그래서

$h^*(3)=3-2=1$

$f(3)=3$

f은 그것의 전임자보다 2가 증가되었다. 반대로 단지 h^*(그림2.(a))를 사용할 때에는 f가 1이 증가한다. 이것은 틀린 경로를 따라 더 빨리 잘라내도록 그림2 (b)안에서 탐색을 허용한다. 그림 2의 결과는

목표까지 거리 : 1

오차 ε의 최대값 : 2

f가 방문한 노드들 : $f=h^*$는 9개

f가 방문한 노드들 : $f=h^*+g$는 5개

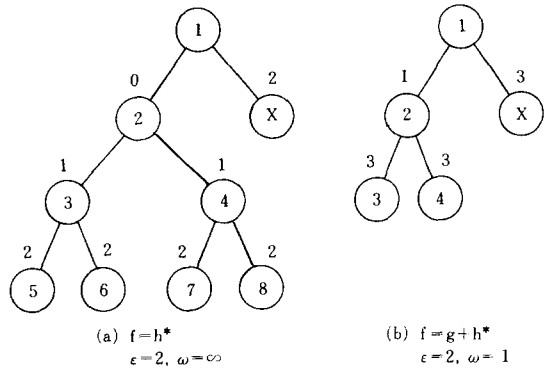

그림 2. 分析의 例示(Ⅱ)
Fig. 2. Example of analysis(Ⅱ).

우리는 위 결과를 일반화 시킬 수 있으며 2진 tree 공간에서 다른 오차들과 경로 길이들에 대해서 방문된 노드들의 수를 알아 낼 수 있는 공식을 만들 수 있다. 최악의 경우의 행동자를 분석함에 있어서 우리는 해답경로상의 $h^*=h+\varepsilon$와 가장 형편없는 탐색으로 이끄는 최단경로에서 떨어져 있는 $h^*=h-\varepsilon$를 보여야 한다.

[정리 3] 만일 $h_1=h_2$이면, $h_1(X) \geq h_2(X)$인 곳에서(X는 해답경로상에 있다) 해답경로를 제외한다. 그러면 h_1을 사용한 발견적 경로 알고리즘에 의한 탐색은 항상 h_2를 사용할 때 방문된 모든 노드들을 방문한다.

〈증명〉 $\mu=(X_1, X_2, \cdots X_k)$를 해답경로라 하자. $S(X_i)$는 X_i가 set S안에 포함 될 때에 발견적 경로 알고리즘에 의해서 탐사된 tree가 될 것이다. 그래서 $S(X_k)$는 발견적 경로 알고리즘이 해답경로를 발견할 때에 찾아진 노드들의 set이다. $S_1(X_i)$를 h_1을 사용한 발견적 경로에 의해서 찾아진 tree라 하고 $S_2(X_i)$는 h_2를 사용한 tree가 될 것이다. 즉 귀납법에 의하여 $S_1(X_i) \geq S_2(X_i)$임을 알 수 있다.

a) $S_1(X_1) \geq S_2(X_1)$

$h_1(X_1) \geq h_2(X_1)$이고 해답경로에서 떨어진 노드들은 같은 값을 갖는다.

b) $S_1(X_i) \geq S_2(X_i)$라 가정하면

$S_1(X_{i+1}) \geq S_2(X_{i+1})$이다.

이것은 a)경우에서와 마찬가지로 $h_1(X_i) \geq h_2(X_i)$이고, 해답경로에서 얻어진 노드들은 같은 값을 가지므로 해답경로를 따르는 발견적 함수의 값의 증가는 단지 찾아진 노드들의 수를 증가 시킨다.

[정리 4] K를 root 노드로부터 목표 노드까지의 거리라 하고 $f=g+h^*$를 발견적 경로 알고리즘에 의

해서 사용된 함수라 하면 2진 tree 공간에서 방문된 노드들의 최대수는 $2^{\varepsilon}\cdot K+1$이다.

〈증명〉 만일 목표 노드가 root로부터 거리K이고 만일 h^*이 완전하다면 발견적 경로 알고리즘은 K+1 노드들을 방문한다(그림1). ε의 오차로서 최악의 경우에 있어서 최단경로에서 ε만큼 떨어져 있는 모든 노드들은 방문될 것이고 다음의 목표 노드들은 제외된다. (그림2)

a) $\varepsilon=0$인 경우

이것은 그림1에서 K+1에 대해서 기술한 경우와 같다.

b) $\varepsilon=1$인 경우

이것들은 최단경로상의 것에다 최단경로에서 떨어진 것들을 합한 노드들이다. K 노드들과 최단경로에서 하나 떨어진 것과 해서 $K+K+1=2K+1$이 된다.

c) $\varepsilon\geq 2$인 경우

이 시점에서 각 확장안된 노드는 2개를 갖으나 아직 탐사된 연속자들은 아니다 2진 tree 안에서 노드들의 수는 $2^{\varepsilon-1}$으로서 커진다. 그래서 오차가 $\varepsilon\geq 2$인 tree는 아래와 같이 된다.

$$2K+1+2K+2^2K+\cdots\cdots+2^{\varepsilon-1}K$$
$$=1+2K+K\sum_{i=1}^{\varepsilon-1}2^i$$
$$=1+2^{\varepsilon}\cdot K \qquad (14)$$

[정리5] K를 root 노드로부터 목표 노드까지 거리라 하고 $f=h^*$는 발견적 경로 알고리즘에 의해서 사용된 함수라 하자. 그러면 2진 tree 공간에서 방문되는 노드들의 최대수는

$$\frac{4^{\varepsilon}}{2}\cdot K+1,\ (\varepsilon\geq 1) \qquad (15)$$
$$K+1\ \ (\varepsilon=0)$$

〈증명〉

a) $\varepsilon=0$인 경우

정리4의 a)의 경우와 같다.

b) $\varepsilon=1$인 경우

앞의 정리4에서 처럼 발견적 경로 알고리즘은 해답경로에서 1만큼 떨어진 노드들을 방문한다. 그래서 $2K+1=4/2K+1$ 노드들이 방문되었다.

c) $\varepsilon\geq 2$인 경우

첫번째 level후에 오차에 있어서 각 증가는 방문되는 두개의 부가된 level들의 최대치를 허용한다. 이것은 우리가 $h+\varepsilon$를 사용하는 해답경로와 2ε만큼 여유가 있는 $h-\varepsilon$를 사용하는 떨어진 해답경로를 따르기 때문이다. 탐사된 노드들의 최대치의 수를 가진 tree들은 아래와 같다.

$$2K+1+2K+2^2K+2^3K+2^4K+\cdots\cdots+2^{2\varepsilon-3}\cdot K+2^{2\varepsilon-2}\cdot K$$
$$=1+2K+K\sum_{i=1}^{2\varepsilon-2}2^i$$
$$=1+\frac{4^{\varepsilon}}{2}\cdot K \qquad (16)$$

이 결과들은 일반적인 발견적 탐색을 위한 2개의 합리적인 결론을 제안한다.

① 좀더 적절한 h^*와 발견적 경로 알고리즘에 의해서 방문되는 더 적은 노드들

② 평가자(evaluator)안에 g를 포함하는 것이 더 좋음.

더 나아가 이 결과들은 흥미있는 유일한 목표 노드로서 어느 조직화된 문제공간까지 확장된다.

[정리6] 만일 발견적 경로 알고리즘이 어떤 목표 노드를 위한 조직화된 공간의 어느 tree를 탐색중이라면

a) $f=h^*$는 위의 가장 나쁜 경우의 분석에서 적어도 $f=g+h^*$만큼 많은 노드들을 방문할 것이다.

b) 만일 $h-\varepsilon\leq h^*\leq h+\varepsilon_1$이고 $h-\varepsilon_2\leq h_2^*\leq h+\varepsilon_2$이면 $\varepsilon_2>\varepsilon_1$이다. 그러면 h_2^*를 사용한 발견적 경로 알고리즘에 의해서 방문된 노드들의 수는 가장 나쁜 경우와 양쪽의 평가자를 위한 것과 같이 되는 ω로서 적어도 h_1^*를 사용한 때 만큼이 될 것이다.

〈증명〉 a)는 정리2와 3으로 부터 얻을 수 있으며 b)부분은 명백하므로 생략한다.

V. 結 論

본 논문에서 논의된 발견적 함수들은 최단경로 문제에 있어서 계산의 접근에 사용하면 좋으며, 또한 이것은 적절한 ω를 선택하므로서 Hamilton 경로문제와 Traveling salesman 문제에도 유용하게 사용 될 수 있다. 그러나 본 논문에서 분석하고 정의된 결과들의 보편성을 적용하는데 있어서 중요한 단점은 문제공간들이 대개 tree가 아닌데 있다. 또한 이 분석들은 가장 나쁜 경우와 이 결과들이 성취 될 수 있는 동안의 것이고 실제에 있어서는 이러한 경우들은 적으며 그 보다는 발견적인 알고리즘의 행동자(behavior)를 감시하는 것이 더 重要하다. 따라서 앞으로의 연구과제는 적절한 ω를 선택하여 정확한 평가함수 f를 구함으로서 탐색공간을 줄이는 것을 실험하는 것이다.

參 考 文 獻

[1] Doran, J. and D. Michie, *Experiments with the Graph Traverser Program.* Pro-

ceedings of the Royal Society (A), 294, pp. 235-259, 1966.

[2] Hart, P.N. Nilsson, and B. Rapheal, *A Formal Basis for the Heuristic Determination of Minimum Cost Path.* IEEE Trans on Sys. Sci. and Cybernetics, June, 1968.

[3] Robert tarjan, "Depth-first search and linear graph algorithms," *SIAM J. Computer,* vol. 1, no. 2, June, 1972.

[4] Pohl I., *Phrase-Structure Production in PL/I,* Letter to communications of the ACM, 10, 757, Dec., 1967.

[5] Floyd. R., *Nondeterministic Algorithms.* Journal of the ACM, 14, pp. 636-644, Oct., 1967.

[6] Golomb, S., and L. Baumet, *Backtrack Programming.* Journal of the ACM, 12, pp. 516-524, Oct., 1965.

[7] A.V. Karzanor, "Determining the maximal flow in a network flow by the Methods of preflows," *Soviet Math, Dolkl,* vol. 15, no. 2, pp. 434-437, 1974.

[8] Shahid H. Bokhari, "A shortest tree algorithm for optimal assignments across space and time in distributed processor system," IEEE Transactions on software engineering, vol. SE-7, no.6, Nov., 1981.