동적 시간제어에 기반한 실시간 탐색 알고리즘에 관한 연구

안 종 일† · 정 태 충††

요 약

본 연구에서는 실시간 휴리스틱 탐색 알고리즘을 개발하고 이것을 기존의 mini-min lookahead 알고리즘과 비교하였다. 많은 실시간 휴리스틱 탐색의 접근 방법에서 종종 전체 문제를 몇 개의 부 분제로 문제를 분할한다. 본 연구에서는 분할된 부 문제에서 마감시간을 적용할 뿐만 아니라 전체 해를 구하는데 있어서도 마감시간을 적용하는 알고리즘을 제안한다. 실시간 휴리스틱 탐색 알고리즘으로 제안된 RTA*, SARTS, DYNORA 등의 알고리즘들은 탐색에 필요한 시간의 예측을 휴리스틱 평가 함수로부터 얻기 때문에 휴리스틱 평가의 정확도가 그 알고리즘의 성능을 보장하게 된다. 그러나 실세계의 문제에서 정확한 휴리스틱 평가 함수를 구하는 것은 매우 어려운 일이므로 부 문제 공간에서의 탐색 상황을 반영한 마감시간을 적용할 필요가 있다. 본 연구에서는 동적 마감시간 전략인 cut-off방법을 사용하는 새로운 알고리즘을 제안한다.

A Study on the Real-time Search Algorithm based on Dynamic Time Control

Jongll Ahn† · TaeChoong Chung††

ABSTRACT

We propose a new real-time search algorithm and provide experimental evaluation and comparison of the new algorithm with mini-min lookahead algorithm. Many other real-time heuristic-search approaches often divide the problem space to several sub-problems. In this paper, the proposed algorithm guarantees not only the sub-problem deadline but also total deadline. Several heuristic real-time search algorithms such as RTA*, SARTS and DYNORA have been proposed. The performance of such algorithms depend on the quality of their heuristic functions, because such algorithms estimate the search time based on the heuristic function. In real-world problem, however, we often fail to get an effective heuristic function beforehand. Therefore, We propose a new real-time algorithm that determines the sub-problem deadline based on the status of search space during sub-problem search process. That uses the cut-off method that is a dynamic stopping-criterion-strategy to search the sub-problem.

※본 연구는 한국과학재단의 '96 핵심전문연구지원사업인 '실시간 인공지능 탐색 시스템 개발 연구'의 지원에 의해 연구되었음.

† 준 회 원:경희대학교 공과대학 전자계산공학과
†† 정 회 원:경희대학교 공과대학 전자계산공학과

논문접수:1997년 5월 10일, 심사완료:1997년 9월 11일

1. 서 론

필요에 따라 실시간 시스템과 인공지능 시스템의 결합에 관한 연구가 광범위하게 진행되고 있다[7]. 그러나 두 연구분야는 성격이 매우 다른 태스크를 처리

한다는 점에서 차이를 보이고 있어 간단한 방법으로는 결합이 불가능하다[6]. 실시간 태스크의 특징은 주기적 성격을 띠며, 태스크의 처리 시간 등의 정보가 미리 알려져야 한다는 제약을 갖는 반면, 인공지능 태스크는 문제의 공간이 매우 광범위하고 애매한 성격을 갖는다. 특히 인공지능의 대표적인 연구 분야인 휴리스틱 탐색의 경우, 탐색 결과 원하는 해를 얻기까지의 시간을 사전에 알기는 매우 어렵다. 따라서 휴리스틱 탐색의 실시간 처리를 위해서는 전체 탐색을 작은 시간 단위로 분할 한후, 각 분할 시간 단위 내에서 일정한 결과를 산출하도록 하면 된다.

대표적인 실시간 인공지능 탐색 시스템은 체스나 장기 바둑 등의 대국자 게임에서 계산시간의 제약조건이 있는 상황의 문제 해결이다. 대국자 게임의 문제는 전체 문제공간이 탐색을 통해 승리할 수 있는 모든 수를 계산하는 것이 최적이겠지만, 문제의 공간을 좁혀 시간 제한을 두고 현재의 상황에서 한 수를 놓는 과정으로 단순화한다. 이러한 문제를 해결하기 위해 제안된 알고리즘은 mini-max탐색 등이 있고, 이것을 일반 그래프 탐색에 적용한 것이 mini-min lookahead 전략을 사용한 RTA*알고리즘[1]이다.

그러나 대국자 게임 등의 문제는 단순히 분할된 문제에 한해 마감시간이 적용되는 것이고 문제 전체에도 마감시간이 적용된다는 제약조건이 추가되면 적용하기 곤란하다. 반면 DYNORA[2][3][4]등의 동적 시간 분할을 사용하는 알고리즘은 현재 시간으로부터 남아있는 마감시간과 앞으로 탐색해야 할 공간의 예측을 이용하여 동적으로 시간분할을 수행하는 것으로, 마감시간을 전체 문제의 해를 구하는 것에 적용하고 분할 문제 내에서는 마감시간을 적용하지 않는다. 따라서 분할된 문제의 영역 시간 제약 조건은 수행될 수 없다.

본 연구에서 제안하는 실시간 인공지능 탐색 알고리즘은 전체 마감시간을 준수할 뿐 아니라 분할된 부문제에서도 마감시간을 만족하는 알고리즘이다.

제안하는 알고리즘의 조건은 다음과 같다. 첫째, 태스크를 적절한 단위로 분할해야 한다(분할 전략). 뿐만 아니라 주어진 분할에 대한 효율적 탐색 전략(부분해 탐색 전략)이 요구된다. 둘째, 분할된 태스크는 주어진 시간 내에 반드시 결과를 내야 한다(부분해 마감시간 준수 전략). 세째, 주어진 결과는 시간 내에 풀 수 있는 최선이어야 한다(부분해 최적화 전략). 넷째, 전체 태스크의 마감시간을 준수 해야 한다(전체 문제의 마감시간 준수). 본 연구의 목적은 위와 같은 조건을 만족하는 실시간 탐색 알고리즘의 개발이다.

2장에서는 탐색 알고리즘의 부 문제 탐색과 cut-off의 전략을 밝히고, 3장에서 mini-min lookahead와 비교 실험을 수행하며, 마지막으로 4장은 결론이다.

2. 실시간 탐색 알고리즘

본 연구에서는 대표적인 휴리스틱 탐색 알고리즘인 A*알고리즘 상에서 시간적 제약 조건이 주어져 있는 문제를 해결하기 위해 변형된 A*알고리즘을 제안한다. A*알고리즘에서 마감시간 내에 목적 노드에 도달하기 위해서는, 완전한 최적해를 찾기 위해 필요한 시간에 비해 모자란 만큼을 최적성에서 비례적으로 포기하고 목적노드에 도달하는 경로를 찾아야 한다.

탐색 알고리즘에서 사용하게 될 변수를 정의하면 다음과 같다:

D : 총 마감 시간. 탐색 트리에서 하나의 노드를 확장하는데 드는 시간을 단위로 사용한다.
D_p : 부 문제의 마감 시간
P : 부 문제의 개수. 총 탐색의 과정을 몇 개로 분할할 것인지를 결정한다.
P_n : 1-n개의 부 문제 중 현재 탐색하고 있는 부 문제.
d : 탐색 트리의 깊이
b : 탐색 트리의 평균 분기계수
He : 휴리스틱 함수의 효율. 0-1사이의 값으로 1에 가까울수록 좋은 효율을 표시
h(s, g) : s(시작 노드)에서 g(목적 노드)까지의 탐색 비용 예측 값.

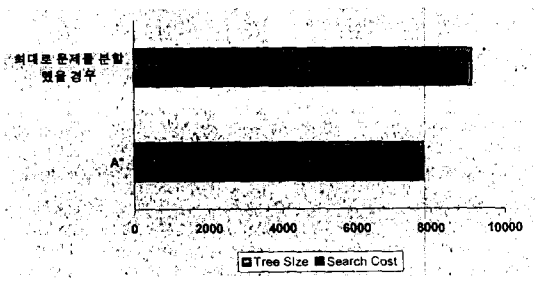
2.1 부분해 탐색 전략

이번 절에서는 부 문제(sub problem)로 전체 문제를 분할해야 하는 이유와, 몇 개의 부 문제로 나눌 것인지를 결정하는 방법을 설명한다. 탐색을 수행하는 전체의 시간은 탐색 트리의 크기와 같다고 가정한다. 문제의 분할이란 하나의 부 문제를 해결하는데 마감시간이 주어지고, 이 시간까지를 탐색한 후 얻게 되는 가장 최선의 해를 부 문제의 부분해라고 정의할

수 있다. 다음의 새로운 부 문제의 탐색 시작 점은 이전의 부분해로부터 시작하게 된다. 탐색의 과정을 몇 개의 부 문제로 분할하는 것은 탐색공간을 가지치기 하는 효과를 갖게 된다.

2.1.1 문제분할

부분해 탐색 과정에서, 만약 매회 탐색 트리를 한 번 확장하고 그 중 최선의 부분해를 찾는다면 이것은 beam search[9] 탐색 전략이 되고 해의 질은 그만큼 나빠지게 된다. 다음 (그림 1)은 분할의 개수와 해의 질의 관계를 보여주는 것으로 부 문제 내부의 탐색의 방법은 A*알고리즘을 사용했다. 실험은 570개의 노드와 평균 degree 3인 그래프에서 무작위로 시작노드와 목적노드를 50회 선정하고 A*탐색과 본 연구에서 제안한 알고리즘을 수행하여 각 시행의 평균값을 비교하였다. 이 때 설정한 부 문제는 탐색이 실패하지 않는 조건까지 최대한 문제를 분할한 것이다. 이 실험을 통해 알 수 있는 것은 부 문제의 분할 개수와 탐색결과 해의 최적성과는 서로 반비례의 관계임을 보인 것이다.

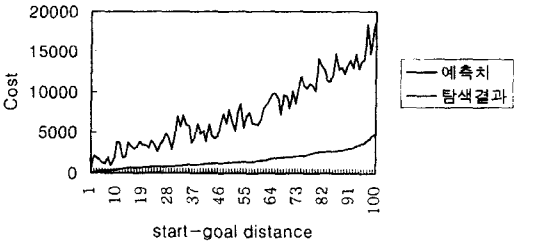


(그림 1) 탐색의 부분해와 해의 질과의 관계의 예
(Fig. 1) Number of sub-problem vs. quality of solution.

다음의 문제는, 탐색의 부분해의 개수와 해의 질과의 관계를 전제로, 어떻게 문제의 성격에 따라 동적으로 부 문제의 개수를 결정할 것인가 이다. 우선, 이 문제를 해결하기 위해서 알아야 할 것은 예측된 탐색 트리의 크기이다. 일반적인 실세계 문제에서 완전한 휴리스틱을 갖고 탐색을 진행하는 것은 매우 힘들므로 휴리스틱의 효율값(Heuristic efficiency: He)을 He라 가정한다[8]. He값은 0과 1사이의 값으로 1에 가까울 수록 효율이 좋은 것을 의미한다.

(그림 2)는 그래프상에서 선택한 두 노드간의 거리를 도로망 등에서 사용하는 대표적인 예측 함수인 유클리디언(Euclideam) 거리를 사용하여 얻은 예측치와 탐색결과를 나타낸다. 휴리스틱 함수로부터 얻은 탐색 트리의 깊이 d가 탐색의 효율값과 지수적 관계가 있음을 보여준다. 즉,

$$D_p = d^{\frac{1}{He}} \times b$$



(그림 2) 탐색 예측치와 탐색결과
(Fig. 2) Estimation and result of search cost

만약 마감시간이 D_p에 비해 충분히 크게 주어진다면, 탐색은 A*보다 더 많은 가지 치기가 필요 없게 되어 부 문제의 분할 없이 탐색을 수행하면 되지만, 마감시간이 예측된 탐색에 소요될 시간 보다 작다면 탐색 트리의 가지 치기를 그에 비례적으로 수행하여야 한다. 따라서 가지 치기의 정도를 표현하는 문제 분할 개수 P는 D_p/D로 정의 할 수 있다. 마감시간이 무한대로 주어지면, 부 문제의 개수는 0으로 수렴되고 이는 부 문제로의 분할이 필요 없는 것을 의미하므로 시작노드-목적노드 사이에서 A*알고리즘으로 탐색이 진행된다.

2.1.2 예 측

실시간 탐색 알고리즘은 시간 내에 결과를 산출하는 제약조건을 갖는다는 점이 가장 큰 특징이라고 볼 수 있다. 그러나 만약, 결과를 낼 수 없는 마감 시간일 경우에는 사용자에게 문제를 해결할 수 없음을 통보하고 그에 대한 대처 방안을 모색하게 하는 것이 사용자의 실시간 시스템에 대한 신뢰성 측면에서 중요하다.

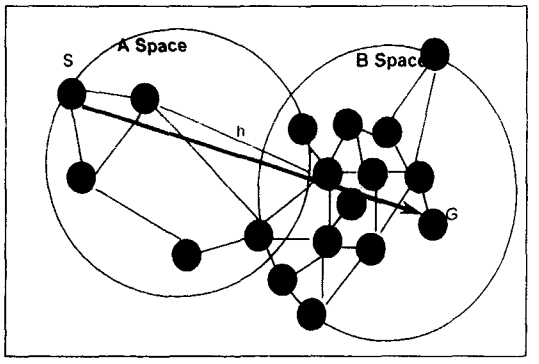
최소의 크기로 탐색을 완료할 수 있는 조건은 완전한 휴리스틱 함수를 사용해 목적노드에 도달할 때 까지 beam-search를 수행하는 것이다. 그러므로 모든 부 문제에서 1회 트리를 확장하고 선택한 최선의 해가 최적해의 경로 선상에 있는 것이어야 한다는 결론이다. 그러므로 부 문제의 개수 P는 예측된 탐색의 깊이 d보다는 최소한 커야 한다.

2.2 탐색의 cut-off전략

휴리스틱 탐색에는 평가 함수로서 식 $f=g+h$를 사용한다. 여기서 f는 탐색 트리 노드가 갖는 비용으로 현재까지 탐색비용 g와 앞으로 목표 노드까지 남아있는 비용의 예측치 h값을 의미한다.

여기서 탐색의 과정 중 현재의 상태를 판단하는데 사용할 수 있는 값은 h값이다. 그 이유는 h값이 탐색이 진행 될수록 감소하는 경향을 갖고 탐색에서 백트레킹(back-tracking)이 수행되는지를 감지할 수 있는 변수이기 때문이다. 그러므로 부 문제 내에서의 목표는 시작 노드와 목표 노드사이에 예측된 총 거리 h(s, g)를 분할된 부 문제의 개수 P로 나누어진 값 만큼 탐색을 진전하는 것이다.

그러나, 부 문제에서 동일한 크기로 탐색 트리를 확장하더라도 각각의 부분해가 h(s, g)/P 만큼을 진전하지 못하는 문제가 있다. 그것은 다음 (그림 3)의 예와 같이 에지의 길이가 비용을 의미하는 그래프에서, 탐색공간 A와 탐색 공간 B는 다른 노드의 밀집도를 보이고 있음에도 휴리스틱 함수는 이를 동일한 조건으로 판단하기 때문이다.



(그림 3) 다양한 탐색공간의 예
(Fig. 3) The example of search space

그러므로 부 문제간의 탐색에는 cut-off전략이 요구된다. 한 개의 부 문제를 풀기 위한 시간이 주어져 있다면 이 시간 동안에 탐색을 중단해야 할 것이라는 판단이 서면 부 문제의 탐색을 종료하고 바로 다음 탐색할 부 문제로 나머지 시간을 넘겨주는 것이 이 cut-off전략의 목적이다.

시작 노드 s에서 목표노드 g까지의 예측값 h(s, g) 값을 이용한 cut-off시점은 새로운 현재의 부 문제 P_n이 시작된 이후부터 현재 선택된 최선의 노드 c까지 진전한 휴리스틱 예측치가 앞으로 남아 있는 부 문제를 통해 해결해야 할 예측치 보다 클 때이다.

$$h(s, g)-h(c, g)>h(s, g)/P^{*}P_n \quad (1)$$

두 번째 문제는 위의 조건에 따라 부 문제를 중단 할 것인지를 판단하는 시점의 결정이다. 이는 부 문제의 최적성에 관련된 것이다. 이 최적성은 부 문제의 공간에서 가장 목적 노드에 근접해 있는 최적 경로 선상의 노드를 선택하는 것을 말한다. 이러한 조건을 만족하기 위해서는 부 문제 내부에서의 A*탐색의 시간을 최대한 보장해 주어야 한다. 이와 더불어 현재의 부 문제를 중단하고 다음 부 문제로 이양할 탐색 시간을 어느 정도로 할 것인지를 결정해야 한다. 만약 이 시간이 과도하게 크다면 한 부 문제에서의 최적성의 만족을 기대할 수 없게 되고 반대로 과도하게 작다면, 정적 부 문제 분할과 결과가 유사해 질 것이기 때문이다. 따라서 본 연구에서는 부 문제의 전체 시간에 50%로 그 범위를 설정했다. 이 cut-off 검사 시작 점(cut-off check point)은 마치 마감시간에 근접해 있음을 알리는 카운트 다운과 유사한 개념이라 할 수 있다. 이 시점 이전에는 식 1의 조건을 만족하는 노드가 있다면 그것을 저장하고 있고, 시작점을 지나 식 1의 조건을 만족한다면 탐색을 중단하게 된다.

부 문제 내에서 식 1에 의해 발생하는 경우는 다음과 같다:

경우 1) 부 문제의 탐색 과정 중 cut-off check point 이후에 식 1에 만족하는 노드를 한개 발견할 경우
- 정상적으로 부 문제를 종료하고 나머지 시간은 다음 부 문제로 이양한다.

• 정상적으로 부 문제를 종료하고 나머지 시간은 다음 부 문제로 이양한다.

경우 2) 부 문제의 마감시간 까지 식 1을 만족하는 노드를 찾지 못했을 경우

• 현재 까지 찾은 노드 중 A*가 찾은 가장 작은 f값을 갖는 노드를 부분해로 정한다.

경우 3) cut-off check point 전에 여러 개의 대상 노드를 찾는 경우

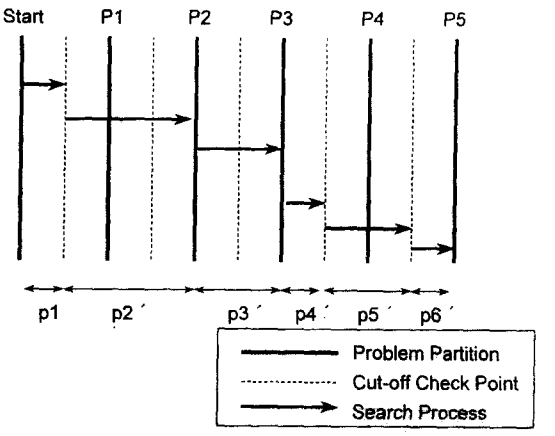
• 이 중 가장 작은 f값을 갖는 노드를 부분해로 결정한다.

2.3 탐색 프로시저

부 문제에서 탐색을 중단하게 되는 2개의 조건은 부 문제에 주어진 마감시간을 완전히 소모하거나, 또는 cut-off전략에 의한 것이다. 최악의 경우 모든 부 문제에서 첫번째 조건에 의해 탐색이 진행된다면 mini-min lookahead와 같은 정적 부 문제 분할과 결과가 동일해 진다.

다음은 탐색의 전과정이 동작하는 한 예를 도식으로 표현한 것이다.

(그림 4)의 p1′와 p4′에서는 탐색의 cut-off가 수행



(그림 4) 제안된 실시간 탐색 알고리즘의 동작 예
(Fig. 4) Execution example of proposed real-time search algorithm

되어 다음 부 문제로 탐색 시간을 이양한 것을 보여주고 p2′와 p5′에서는 앞서 넘겨받은 탐색시간을 해당 부 문제에 적용하여 문제를 풀고 있는 것을 보여준다.

위의 전과정을 가상코드로 작성면 다음과 같다.

(그림 5)의 탐색 알고리즘 프로시저는 전통적인 A*

1. *부 분할 문제의 갯수P 를 결정* $P = D_f / Deadline$, $P_n = 1$
2. Start Node를 Open_Q에 삽입
3. Open_Q의 첫 번째 노드를 Current Node로 하고 이것이 목표노드가 아니거나 마감 시간 내에 있는 동안 다음을 시행
   - 3.1 목표노드에 도달했다면
     - 3.1.1 목표노드 도달을 통고, 탐색을 중단
   - 3.2 만약 현재가 *cut-off check point이전 이라면*
     - *3.2.1 만약* $h(s,g) - h(current,g) > h(s,g)/P*P_n$ *이라면 Candidte Node로 현재의 노드를 저장*
   - 3.3 *만약 현재가 cut-off check point이라면*
     - *3.3.1 만약* $h(s,g) - h(current,g) > h(s,g)/P*P_n$ *이라면 Min(f(Candidate Node),f(Current Node))를 root로한 새로운 부 문제 시작*
   - 3.4 *만약 현재가 부 문제 마감시간 이라면*
     - *3.4.1 Current Node를 root로한 새로운 부 문제 시작* P_{n++}
     - *3.4.2 cut-off check point = next(cut-off check point)*
   - 3.5 Current Node의 자식 노드를 생성하고 Open_Q에 삽입 하고 Step 3으로

(그림 5) 탐색 알고리즘 프로시저
(Fig. 5) overall procedure

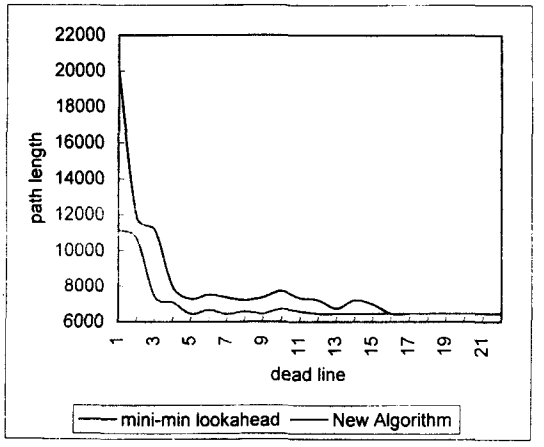
알고리즘에서 1, 3.2, 3.3, 3.4 부분을 추가함으로서 구현될 수 있다. 알고리즘의 순환 루틴인 3.2, 3.3, 3.4는 단지 cut-off인지를 검사하고 새로운 부 문제를 시작할지를 결정하는 것으로 A*알고리즘의 복잡도를 증가 시키지 않는다.

3. 실험 및 결과

본 연구에서 제안한 알고리즘과 실시간 휴리스틱 탐색 알고리즘에서의 가장 원론적인 mini-min lookahead를 비교한다. 첫번째 두 알고리즘의 마감시간의 변화에 따른 탐색 비용의 변화량을 관찰한다. 이 실험은 알고리즘의 부문제 내에서의 탐색전략에 따른 비교로서 얼마나 최소의 마감시간을 갖고 최적해에 가까운 해를 얻을 수 있는가를 분석하는데 중요한 관점이 된다고 하겠다.

실험은 570개의 노드와 degree가 3인 그래프에서 무작위로 두개의 노드를 시작노드와 목적노드로 50회 선택하여 탐색을 수행한 후, 각 탐색된 경로들의 길이를 평균하였다. 휴리스틱 함수는 일반적인 road map에서 사용하는 유클리디언 거리를 사용하였다.

실험 결과, (그림 6)에서와 같이 본 연구에서 제안한 알고리즘은 마감시간 500정도에서 안정 상태에 돌입한 반면 mini-min lookahead의 경우 1600에서 안정

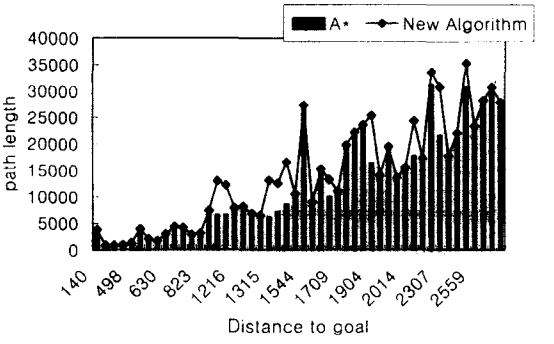


(그림 6) mini-min lookahead와의 탐색 시간 비교 실험
(Fig. 6) Comparison of search time between mini-min lookahead and proposed algorithm

상태에 도달하는 것을 알 수 있다. 또한 마감시간이 최소화 되었을 경우에도 탐색 비용이 11000과 22000으로 많은 차이를 보인다. 따라서 제안된 알고리즘이 마감시간을 잘 지키면서도 해의 질도 좋음을 알 수 있다.

마감시간을 고려하지 않고 최적해를 찾는 A*와, 전체 및 부분 마감 시간을 지키며 최선의 해를 제안된 알고리즘이 얼마나 최적해에 가까운 해를 찾는지 비교할 필요가 있다.

(그림 7)에서는 A*가 최적해를 찾는데 걸리는 시간과 동일한 시간을 본 연구에서 제안한 알고리즘에 마감시간으로 주고 탐색하였을 경우에 어느 정도 탐색된 해의 질이 최적에 가까운지 보이는지 비교하여 보았다. 실험은 그래프상에서 임의로 선택한 50개의 시작노드-목적노드 상을 선택해, A*와 본 연구에서 제안한 알고리즘으로 경로를 찾고, 그 경로 길이 값을 비교하여 보았다. 즉 이것은 본 연구에서 제안한 알고리즘이 마감시간을 지키면서도 얼마나 최적해에 가까운 해를 구하는지 실험한 것이다. 실험 결과 최적 해와 평균 11.74%의 차이를 보이고 있으므로 많은 차이를 보이지 않음을 알 수 있다.



(그림 7) A*와 동일한 시간으로 탐색 했을 때의 탐색비용 비교
(Fig. 7) Comparison of search costs between A* and proposed algorithm

4. 결론 및 향후 연구 방향

지금까지 실시간 인공지능 탐색을 위한 알고리즘을 제안하고 실험을 통해 알고리즘의 성능을 분석하였

다. 본 연구에서 제안한 알고리즘은 기존의 mini-min lookahead와 마감시간의 변화량에 따른 해의 질을 비교하여 본 결과, 탐색의 효율이나 마감시간의 준수에 있어서 우수함을 보였다.

또한 이 알고리즘의 특징은 하나의 분할된 부 문제 내에서 최소한 하나 이상의 부분해를 찾는 mini-min lookahead와 같이 정적인 시간 단위를 사용하면서도 동적인 탐색 전략을 갖고 있다는 점이다. 이것은 다른 특성을 갖는 태스크들과도 동시에 스케줄링이 가능하게 되었다는 것을 의미한다. 또한 다른 알고리즘이 갖고 있지 못한 특징으로 하나의 부문제에서만 마감시간을 적용하고 있는 것이 아닌 전체 완전해를 얻는데 까지를 마감시간으로 동시에 고려하고 있다는 점이 특징이라 할 수 있다.

휴리스틱 탐색은 최적성을 보장하는 해를 얻는데 기본적인 방법이라 알려져 있지만, 실세계의 문제에 적용하려면 대부분의 문제에서 시간제약 조건을 만족할 수 없기 때문에 적용에 어려움이 있다. 본 연구에서 제안한 알고리즘은 자율적으로 경로를 탐색하고 작업을 처리하는 로보틱스 분야, 각종 자동 제어 시스템 등에 적용이 가능하며, 인터넷 환경에서는 원하는 정보를 임의의 시간 내에 찾아 주는 인터넷 에이전트의 개발에도 핵심적인 기술이라 할 수 있다. 앞으로의 연구는 이러한 실제 문제에 효과적 적용하여 보는 것이라 할 수 있다.

참 고 문 헌

[1] Richard E. Korf, "Real-Time Heuristic Search: New Result", Proc. of AAAI-88, pp. 133-144, 1988.

[2] Shashi Shekhar and Babak Hamidzadeh, "Self-Adjusting Real-Time Search: A Summary of Results", Proc. of the 1993 IEEE Int'l Conf. on Tools for AI, pp. 224-321, 1993.

[3] Babak Hamidzadeh and Shashi Shekhar, "DYNORA: A Real-Time Planning Algorithm to Meet Response-Time Constraints in Dynamic Environments", Proc. of the 1991 IEEE Int'l Conf. on Tools for AI, pp. 228-235, 1991.

[4] Shashi Shelkar and Soumitra Dutta, "Minimizing Response Time in Real-time Planning and Search", Proc of the 11th Int'l Joint Conf. on Artificial Intelligence, pp. 238-242, 1986.

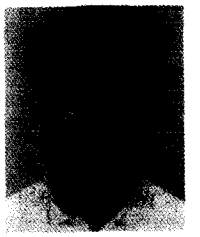
[5] C.J Paul. "A Structured Approach to Real-Time Problem Solving", Doctor of Philosophy in Electrical and Computer Engineering, CMU. 1993.

[6] Pemberton and Kork, "Incremental search algorithms for real-time decision making", Proc. of the Second International Conf. on Artificial Intelligence Planning System, AIPS-94, pp. 140-145, 1994.

[7] Thomas J. Laffey, Preston A. Cox, James L. Schmidt, Simon M. Kao & Jackson Y. Read, "Real-Time Knowledge-Based Systems", AI Magazine, Vol. 9, No. 1, pp. 27-45, 1988.

[8] C.J Pearl, "Heuristics", Addison Wesley, Reading, MA, 1984.

[9] Patrick Henry Winston, "Artificial Intelligence 3rd ed.", 1992, Addison Wesley, 1992.



안 종 일

1992년 국민대학교 기계설계학과 졸업(학사)

1994년 경희대학교 대학원 전자계산공학과(공학석사)

1994년~현재 경희대학교 대학원 전자계산공학과 박사과정 재학중

관심분야: 인공 지능, 신경망 이론 등.

정 태 충

1980년 서울대학교 전자공학과 졸업(학사)

1982년 한국과학기술원 전자계산학과(공학석사)

1987년 한국과학기술원 전자계산학과(공학박사)

1987년 9월~1988년 3월 KIST 시스템 공학센터 선임연구원

1988년 3월~현재 경희대학교 전자계산공학과 교수

관심분야: 인공 지능, 자연어 처리, 멀티미디어 등