논문 2010-47TC-8-11

선호도 기반 최단경로 탐색을 위한 휴리스틱 융합 알고리즘

(A Combined Heuristic Algorithm for Preference-based Shortest Path Search)

옥 승 호*, 안 진 호**, 강 성 호***, 문 병 인****

(Seung-Ho Ok, Jin-Ho Ahn, Sungho Kang, and Byungin Moon)

요 약

본 논문에서는 개미 군집 최적화 (Ant Colony Optimization; ACO) 및 A* 휴리스틱 알고리즘이 융합된 선호도 기반 경로탐색 알고리즘을 제안한다. 최근 ITS (Intelligent Transportation Systems)의 개발과 함께 차량용 내비게이션의 사용이 증가하면서 경로탐색 알고리즘의 중요성이 더욱 높아지고 있다. 기존의 Dijkstra 및 A*와 같은 대부분의 최단경로 탐색 알고리즘은 최단거리 또는 최단시간 경로 탐색을 목표로 한다. 하지만 이러한 경로 탐색 결과는 더 안전하고 특정 경로를 선호하는 운전자를 위한 최적의 경로가 아니다. 따라서 본 논문에서는 선호도 기반 최단 경로 탐색 알고리즘을 제안한다. 제안된 알고리즘은 주어진 맵의 링크 속성 정보를 이용하며, 각 링크에 대한 사용자 선호도는 내비게이션 사용자에 의해 설정되어 진다. 제안된 알고리즘은 C로 구현하였으며, 64노드 및 118링크로 구성된 맵에서 다양한 파라미터를 통해 성능을 측정한 결과 본 논문에서 제안한 휴리스틱 융합 알고리즘은 선호도 기반 경로뿐만 아니라 최단 경로 탐색에도 적합함을 알 수 있었다.

Abstract

In this paper, we propose a preference-based shortest path algorithm which is combined with Ant Colony Optimization (ACO) and A* heuristic algorithm. In recent years, with the development of ITS (Intelligent Transportation Systems), there has been a resurgence of interest in a shortest path search algorithm for use in car navigation systems. Most of the shortest path search algorithms such as Dijkstra and A* aim at finding the distance or time shortest paths. However, the shortest path is not always an optimum path for the drivers who prefer choosing a less short, but more reliable or flexible path. For this reason, we propose a preference-based shortest path search algorithm which uses the properties of the links of the map. The preferences of the links are specified by the user of the car navigation system. The proposed algorithm was implemented in C and experiments were performed upon the map that includes 64 nodes with 118 links. The experimental results show that the proposed algorithm is suitable to find preference-based shortest paths as well as distance shortest paths.

Keywords : Shortest path search, navigation, ant colony optimization, A* algorithm, heuristic algorithm

* 정회원, 경북대학교 전자전기컴퓨터학부
(School of Electrical Engineering and Computer Science, Kyungpook National University)
** 정회원, 호서대학교 전자공학과
(Department of Electronic Engineering, Hoseo University)
*** 평생회원, 연세대학교 전기전자공학과
(Department of Electrical and Electronic Engineering, Yonsei University)
**** 평생회원, 경북대학교 전자공학부
(School of Electronics Engineering, Kyungpook National University)
※ 본 논문은 지식경제부 산업기술개발사업으로 지원된 연구 결과임 (과제번호: 10006927).
접수일자: 2010년5월27일, 수정완료일: 2010년8월13일

Ⅰ. 서 론

경로탐색 알고리즘은 교통시스템, 통신 네트워크, 운송시스템은 물론 이동 로봇의 경로 설정 등 다양한 분야에 사용되는 알고리즘으로써, 최근 ITS (Intelligent Transportation Systems)의 개발과 함께 차량용 내비게이션의 사용이 증가하면서 경로탐색 알고리즘의 중요성이 더욱 높아지고 있다[1~2]. 현재 차량용 내비게이션은 멀티미디어 및 정보 통신기술의 결합과 함께 다양한 기능 및 정보를 사용자에게 제공하고 있으며, 이러한 기능과 정보를 사용해서 목적지점까지의 최단경로를 탐색하는 것이 내비게이션 시스템의 핵심 기능이다.

그 동안 Dijkstra, A* 및 휴리스틱 알고리즘 등 다양한 최단경로 탐색 알고리즘이 적용된 차량용 내비게이션 시스템이 연구되었다[2~5]. Sara Nazari 등은 Dijkstra 알고리즘에 사용되는 많은 량의 메모리 및 연산 량을 줄이기 위해 경로 탐색 영역을 제한하는 수정된 Dijkstra 알고리즘을 제안하였다[2]. Hao Yue 등은 A* 알고리즘을 사용하여 동적 교통 환경에서 실시간 경로탐색이 가능한 시스템을 제안하였으며[3], M. Noto 및 H. Sato는 짧은 시간 내에 Dijkstra 알고리즘의 성능에 가까운 최단경로 탐색 결과를 얻는 방법을 제안하였다[4]. 하지만 기존의 경로탐색 알고리즘은 대부분이 최단거리 및 최소비용 탐색을 위한 알고리즘으로써, 최근 사용자의 선호에 따른 경로탐색이 중요해지면서 최단경로가 아닌 사용자 및 특정 상황에 최적화된 최적경로 탐색에 대한 연구가 필요하게 되었다. 이에 본 연구에서는 선호도에 기반을 둔 경로탐색 알고리즘을 제안한다. 본 논문의 Ⅱ장에서는 기존의 경로탐색 알고리즘에 대해 간단히 살펴보며, Ⅲ장에서는 선호도 기반 경로탐색 알고리즘을 제안한다. 이후 Ⅳ장에서 제안된 알고리즘의 구현 및 다양한 성능측정 결과에 대해 논하며, 이후 Ⅴ장에서 본 논문의 결론을 맺는다.

Ⅱ. 경로탐색 알고리즘

기존의 dijkstra 등과 같이 polynomial time의 시간복잡도를 가지는 최단경로탐색 알고리즘은 노드의 수가 증가함에 따라 경로 탐색 시간이 오래 걸리는 단점이 있다. 이에 본 연구에서는 최단경로탐색 문제를 NP-complete 문제로 보고 노드의 수가 증가하더라도 빠른 시간 내에 적절한 경로를 탐색하는 휴리스틱 알고리즘인 ACO 및 A*를 사용하여 선호도 기반 최단 경로를 탐색한다. ACO 및 A*는 NP-complete 문제를 해결하기 위한 휴리스틱 알고리즘이기 때문에 시간복잡도를 다항시간으로 나타낼 수는 없다. 본 장에서는 기존의 가중 무향 그래프(weighted undirected graph) 모델에서 각 링크의 가중치 및 휴리스틱 정보를 기반으로 최단경로를 탐색하는 알고리즘인 ACO 및 A*에 대해 간략히 설명하고, 각 알고리즘의 특징에 대해서 살펴본다.

1. Ant Colony Optimization 알고리즘

Dorigo 등에 의해 제안된 개미 군집 최적화 (ACO) 알고리즘은 개미 군집의 형태를 모방하여 최적화 문제를 해결하는 생물학적 기반의 메타휴리스틱 접근법이다[6~9]. 군집에 속한 개미들은 목표지점을 찾아가는 동안 페로몬(pheromone)이라는 물질을 분비하는데, 이러한 페로몬은 시간이 지남에 따라 일정하게 증발하는 특징이 있다. 따라서 페로몬은 목표지점까지의 최단경로에 많이 누적되는 특징이 있으며, 군집에 속한 다른 개미들에게 최단 경로를 알리는 정보로 사용된다.

최초의 개미 알고리즘은 Dorigo 등에 의해 개발된 개미 시스템 (ant system; AS) 알고리즘이며[6], AS 알고리즘의 성능을 개선한 알고리즘이 개미 군집 시스템 (ant colony system; ACS) 알고리즘이다[7]. ACS 알고리즘은 기존의 AS 알고리즘과 다른 노드 전이 규칙 (node transition rule) 및 페로몬 업데이트 규칙이 사용되고, 지역 페로몬 업데이트 규칙 및 노드 후보 리스트 (candidate lists)의 사용이 가장 큰 차이점이다[8~9].

ACS 알고리즘에서 노드 i에 위치한, 개미 k는 수식 (1)을 사용해서 다음노드 j를 선택한다.

$$j=\begin{cases}\arg\max_{u\in N_i^k(t)}\{\tau_{iu}(t)\eta_{iu}^{\beta}(t)\} & \text{if } r\le r_0\\ \textit{choosing } j \textit{ using the probability (2)} & \text{if } r>r_0\end{cases} \quad (1)$$

여기서 N_i^k는 개미 k가 노드 i에서 선택 가능한 노드들의 집합을 나타내며, τ_{ij}는 노드 i와 j를 연결하는 링크의 페로몬을 나타낸다. 그리고 η_{ij}는 휴리스틱 함수로써 일반적으로 노드 i, j사이 거리의 역수로 정의된다. β는 휴리스틱 함수의 지수를 나타내는 파라미터로써 양의 실수로 설정된다. r은 [0, 1]사이의 랜덤 값이며, r_0는 노드 선택 방법의 확률을 결정하는 파라미터 값이다. 만약 $r\le r_0$인 경우는 노드 i에서 노드 j로 이동할 때 페로몬 및 휴리스틱 함수를 이용해서 선험적인 정보를 이용하여 노드를 선택한다. 반면 $r>r_0$인 경우 개

미는 확률 함수 (2)를 통해 확률적으로 노드를 선택하기 때문에 페로몬 및 휴리스틱 값이 큰 노드가 선택될 가능성이 높다. 하지만 선험적인 정보에만 의존해서 노드를 선택하지 않기 때문에 개미들은 다양한 경로를 탐색할 수 있다.

$$p_j^i(t) = \frac{\tau_{ij}(t)\eta_{ij}^{\beta}(t)}{\sum_{u \in N_i^k(t)} \tau_{iu}(t)\eta_{iu}^{\beta}(t)} \quad (2)$$

ACS 알고리즘은 기존의 AS 알고리즘과는 달리 최단거리를 찾은 개미의 경우에만 전역 페로몬 업데이트(global pheromone update) 규칙 (3)을 사용해서 자신의 경로에 페로몬을 증가시킨다. ρ_1은 페로몬 증발량을 결정하는 파라미터로써 (0, 1)사이의 값으로 설정되며, $\Delta\tau_{ij}(t)$는 링크 i, j 사이에 누적되는 페로몬 양을 나타낸다.

$$\tau_{ij}(t+1) = (1-\rho_1)\tau_{ij}(t) + \rho_1 \Delta\tau_{ij}(t) \quad (3)$$

ACS는 각 개미가 지나가는 모든 링크를 지역 페로몬 업데이트(local pheromone update) 수식 (4)를 사용해서 페로몬을 증가시킨다.

$$\tau_{ij}(t) = (1-\rho_2)\tau_{ij}(t) + \rho_2\tau_0 \quad (4)$$

수식 (4)에서 ρ_2은 페로몬 증발량을 결정하는 파라미터로써 (0, 1)사이 값으로 설정되며, τ_0는 양의 실수 값으로써 페로몬 갱신 양을 나타낸다.

2. A* 알고리즘

A* 알고리즘은 ACS 알고리즘과 달리 목표지점까지의 일직선 거리 값을 휴리스틱 함수로 사용하여 목적지까지의 최단 경로를 찾는 알고리즘이다[4]. A* 알고리즘은 수식 (5)를 이용해 현재노드에서 F 값이 가장 작은 노드를 연속적으로 선택함으로써 목표노드를 찾아나가는 알고리즘이다.

$$F = G + H \quad (5)$$

수식에서 F는 적합도(fitness)를 나타내며, G는 목표(goal)를 의미하는 값으로써 시작노드부터 현재노드까지의 누적된 비용을 나타낸다. 그리고 H는 휴리스틱(heuristic)을 의미하는 값으로써 현재노드에서 목표노드까지의 추정된 비용을 나타내며, 일반적으로 H는 현재노드에서 목표노드까지의 일직선 거리 값으로 설정된다.

Ⅲ. 선호도 기반 경로 탐색 알고리즘

본 장에서는 기존의 최단경로 탐색 알고리즘인 ACS 및 A* 알고리즘의 한계를 살펴보고, 두 알고리즘의 장점이 융합된 선호도 기반 경로 탐색 알고리즘을 제안한다. 선호도 기반 경로 탐색 알고리즘은 기존의 가중 무향 그래프의 각 링크마다 선호도 및 비선호도 값이 할당된 새로운 그래프 모델에서 선호도를 기반으로 최단경로를 탐색하는 알고리즘이다.

1. 기존 경로탐색 알고리즘의 한계

기존의 ACS 및 A* 알고리즘은 최단경로를 탐색하는 알고리즘으로써, 사용자의 선호도에 기반을 둔 경로탐색 알고리즘으로는 부적절하다. 이는 기존의 ACS 알고리즘은 각 링크의 거리 값을 휴리스틱 함수로 사용하며, A* 알고리즘은 현재노드부터 도착노드까지의 일직선 거리를 휴리스틱 함수로 사용하기 때문이다.

각 알고리즘의 특징을 살펴보면, ACS 알고리즘의 경우 파라미터 변화에 따라 성능 변화가 심하기 때문에 최단경로 탐색 결과가 sub-optimal 할 수 있으며, 노드의 수가 증가할수록 경로탐색을 위한 연산이 증가하는 문제가 발생한다. 그리고 A* 알고리즘은 Dijkstra 알고리즘과는 달리 모든 노드를 탐색하지 않기 때문에 최단경로를 보다 빠르게 탐색하는 특징이 있지만, 선호도 기반 경로 탐색 시 sub-optimal 한 경로를 탐색하는 문제를 가진다. 다음 2절에서는 이러한 문제를 간단히 살펴본 후 휴리스틱 융합 알고리즘을 제안한다.

2. 휴리스틱 융합 경로탐색 알고리즘 제안

A* 알고리즘은 그림 1과 같은 맵에서 최단 경로를 찾지 못한다. 그림 1에서 각 링크 옆의 수는 링크의 거리(distance)를 의미하며, 시작노드는 0, 도착노드는 15이다. 참고로 각 노드들의 위치는 실제 기하학적 지리 위치와는 다르다. A* 알고리즘은 1점 쇄선으로 표시된 (0 -> 7 -> 10 -> 15)의 경로를 출력하며, 이때 총 거리는 51이 된다. 하지만 실제 최단 경로는 (0 -> 1 -> 11 -> 15)이며, 이때 총 거리는 34가 된다.

이와 같이 A* 알고리즘이 선호도 기반 최단 경로를 탐색하지 못하는 이유는 1번 노드를 다음 현재노드로 선택하지 못하기 때문이다. 즉, A*는 이웃노드까지의 비용(G) 및 휴리스틱 (F)의 합을 기준으로 다음 현재노드를 설정하는데, A*의 경우 1번 노드까지의 G 값이

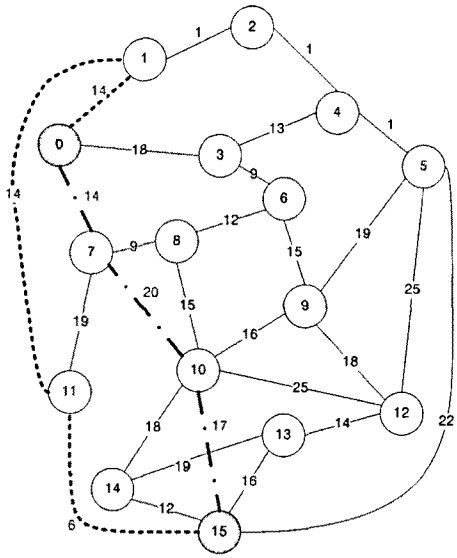


그림 1. 링크의 선호도가 포함된 맵
Fig. 1. Map that contains the preference of the links.

크기 때문에 F 값이 커지고, 따라서 1번 노드를 현재 노드로 설정하지 못한다. 이외에도 A* 알고리즘이 1번 노드를 현재노드로 설정하지 못하는 이유는 선호도가 반영된 맵임에도 불구하고, 휴리스틱 값을 현재노드와 목표노드의 일직선 거리로 설정하기 때문이다. 결론적으로 A* 는 현재노드를 중심으로 최단경로 탐색이 이루어지기 때문에 1번 노드를 현재노드로 설정하지 못할 경우 선호도가 반영된 맵에서 최적 경로 탐색이 불가능하다. 하지만 A* 알고리즘에서 (0 -> 1) 링크의 거리가 일정 값 이하로 감소되면, 1번 노드가 현재노드로 설정될 수 있고, 따라서 상기 실제 최단 경로를 찾을 수 있다.

본 논문에서는 이러한 문제를 변형된 ACS 알고리즘을 적용하여 해결한다. ACS 알고리즘에서는 확률에 기초하여 개미들이 여러 경로를 탐색하기 때문에, (0 -> 1) 링크를 포함하는 경로를 찾는 개미가 존재할 수 있다. 개미들이 찾은 경로 중에서 최소 비용 경로에 속하는 링크들에 대해서는 거리를 감소시키고 (ACS 알고리즘에서 페르몬을 증가시키듯이), 다시 A* 알고리즘에 의한 경로 탐색을 수행하도록 함으로써, (0 -> 1) 링크를 포함하는 최단 경로를 찾을 수 있게 한다. 추가적으로, 제안되는 알고리즘은 링크의 선호도 개념을 사용함으로써, 선호도에 기반한 경로를 찾을 수 있도록 한다. 변형된 ACS 알고리즘은 개미들이 선호도가 높은 링크를 선택할 가능성을 높이고, 경로의 비용 계산 시에도 선호도가 높은 링크에 대해서는 비용을 감소시킨다. 이

d_{ij} : distance of link (i, j)
$x^k(t)$: path of the kth ant
$\overline{x}(t)$: path of best ant in each iteration
$\hat{x}(t)$: current best path of *ACO Loop*
$f(x)$: cost of path x
initialize parameters;
place all ants to the source node;
// Try Loop
repeat (0 ~ *MaxTry*);
  $\tau_{ij}(t) = d_{ij}$; $(d_{ij} > 0)$
  $\hat{x}(t)$ = A* algorithm($\tau_{ij}(t)$);
  if $\beta = 0$;
    return $\hat{x}(t)$;
  // ACO Loop
  repeat (0 ~ *MaxACO*);
    for each ant k = 1,…, n_k do
      $x^k(t) = \varnothing$;
      repeat
        if $r \le r_0$ then
          choose $j \in N_i^k(t)$ from candidate list;
          if($\exists j \in candidate\,list$)
            choosing j using the rule (6);
          else
            choosing j using the rule (7);
        end
        else
          choose $j \in N_i^k(t)$ using the rule (7);
        end
        $x^k(t) = x^k(t) \cup \{(i,j)\}$;
      until full path has been constructed
      $f(x^k(t)) = \Sigma \tau_{ij}(t)\eta_{ij}^{\beta}(t)$;
    end
    select $\overline{x}(t) : f(\overline{x}(t)) = \min_{k=1,\ldots,n_k}\{f(x^k(t))\}$;
    if $f(\overline{x}(t)) < f(\hat{x}(t))$ then
      $\hat{x}(t) = \overline{x}(t)$;
      $f(\hat{x}(t)) = f(\overline{x}(t))$;
      for each link $(i,j) \in \hat{x}(t)$ do
        $\Delta\tau_{ij}(t) = 1/(\tau_{ij}(t)\eta_{ij}^{\beta}(t))$
        $\tau_{ij}(t+1) = (\rho_1 - \Delta\tau_{ij}(t))\tau_{ij}(t)$;
      end
    end
    $\hat{x}(t+1) = \hat{x}(t)$;

```
    f(x̂(t+1)) = f(x̂(t));
    t = t+1;
  until number of MaxACO;
  // End of the ACO Loop
  x̂(t) = A* algorithm(τ_ij(t));
  return x̂(t) as the solution
  β = β+α;
until number of MaxTry;
// End of the Try Loop
```

그림 2. 선호도 기반 최단경로 탐색 알고리즘 의사코드

Fig. 2. Pseudo code of preference-based shortest path search algorithm.

렇게 함으로써, 선호도가 높은 링크들로 이루어진 경로는 최소 비용 경로 선택되고, 여기에 속하는 링크들의 거리는 감소한다. 앞에서도 설명하였듯이, 다시 A* 알고리즘에 의한 경로 탐색 시에는 선호도가 높은 링크들을 포함하는 경로를 탐색하게 된다. 이 경우 변형된 ACS 알고리즘은 최단경로로 수렴할 필요가 없기 때문에 많은 연산이 필요하지 않으며, 뛰어난 적응성을 가지기 때문에 동적 특성을 가지는 환경에 매우 적합한 알고리즘이다.

그림 2는 제안하는 알고리즘의 의사코드(pseudo code)를 나타낸다. 알고리즘 동작 시 초기화 파라미터로는 개미의 수(n_k), 노드 선택 기준 값(r_0), 휴리스틱 함수의 지수(β=0), 최대 *MaxTry* 및 *MaxACO*가 있다. 그리고 모든 개미의 출발지는 시작노드로 설정된다. 알고리즘의 초기 파라미터가 설정된 후 각 링크의 페로몬은 주어진 맵의 각 링크 거리 값으로 초기화 되며, A* 알고리즘은 초기화된 페로몬을 사용하여 최단경로를 탐색 후 $\hat{x}(t)$를 설정한다. *Try Loop*의 첫 번째 iteration에서는 선호도가 반영되지 않은 최단경로 탐색 결과를 출력하고 *ACO Loop* 수행 없이 다음 iteration으로 넘어 간다.

그림 2의 *ACO Loop* 에서는 모든 개미가 노드 선택 수식 (6) 및 (7)을 사용해서 도착노드까지의 경로를 탐색해 간다. 수식 (7)에서 페로몬과 휴리스틱 함수의 역수가 사용된 이유는 초기 페로몬이 링크의 거리 값으로 설정되고, 휴리스틱 함수는 수식 (8)과 같이 정의되기 때문이다. 수식 (8)에서 링크의 선호도 및 비선호도는 사용자의 설정에 따라 맵의 거리정보와 함께 알고리즘에 입력된다.

$$j = \arg\min_{u \in N_i^k(t)} \left\{ \tau_{iu}(t) \eta_{iu}^{\beta}(t) \right\} \tag{6}$$

$$p_j^i(t) = \frac{1/\tau_{ij}(t)\eta_{ij}^{\beta}(t)}{\sum_{u \in N_i^k(t)} 1/\tau_{iu}(t)\eta_{iu}^{\beta}(t)} \tag{7}$$

$$\eta = \frac{avoidace\ of\ the\ link}{preference\ of\ the\ link} \tag{8}$$

노드 선택 시 매우 작은 값의 r_0가 사용된다면, 대부분의 개미는 수식 (7)을 사용해서 확률적으로 다음노드를 선택하기 때문에 매우 다양한 경로를 탐색하는 특징을 가진다. 이와 달리 만약 큰 값의 r_0가 사용된다면 대부분의 개미들이 수식 (6)에 의해서 현재노드에서 가장 저비용의 노드만 선택하기 때문에 특정 경로를 벗어나서 탐색하기 어려운 특징을 가진다. 따라서 r_0 값은 알고리즘의 성능에 큰 영향을 미치는 파라미터이다.

그리고 모든 개미의 경로 탐색이 완료되면 각 개미들이 생성한 경로의 비용을 계산하는데, 이때 경로의 비용 계산은 페로몬 및 휴리스틱 함수의 곱으로 이루어진다. 따라서 경로의 비용계산에 링크의 거리뿐만 아니라 링크의 선호도도 포함될 수 있다.

링크의 페로몬 업데이트는 *ACO Loop*에서 현재까지 가장 낮은 비용을 가지는 개미의 경로에서만 수행된다. 각 *ACO Loop*가 완료되면 변경된 거리인 $\tau_{ij}(t)$를 적용하여 A* 알고리즘에 의한 경로 탐색을 다시 수행함으로서, 선호도가 반영된 최단 경로 탐색이 이루어진다.

*Try Loop*의 각 iteration 끝에서는 휴리스틱 함수의 지수(β) 값을 일정량 α 만큼 증가시키고, 다음 *Try Loop* iteration을 수행한다. β를 일정하게 증가시키는 이유는 선호도에 따른 다양한 경로를 탐색하기 위함이다. 즉, *Try Loop*가 진행됨에 따라 선호도가 더 많이 반영된 경로를 탐색할 수 있으며, 따라서 사용자는 선호도 반영 정도가 다른 여러 가지 최단 경로 중에서 자신에게 적절한 경로를 선택할 수 있다.

본 논문에서 제안하는 알고리즘에서는 경로 탐색 시 모든 노드를 탐색하지 않는다. 그리고 *ACO Loop* 안에서 한번의 iteration을 마친 후 iteration best ant의 경로에 대한 페로몬 값 업데이트로 인해 링크의 비용이 변한다. 만약 선호도와 비선호도를 거리(weight value)와 함께 하나의 최종 비용으로 미리 모두 계산한 후 경로탐색 알고리즘을 수행하는 방법을 사용한다면, 경로

탐색에 사용되지 않는 노드를 미리 계산하는 경우가 발생되며, 맵의 크기가 커질수록 그리고 개미의 수가 작아질수록 이러한 오버헤드는 더욱 커지게 된다. 따라서 본 논문에서는 모든 링크의 선호도와 비선호도를 거리와 함께 하나의 최종 비용으로 미리 계산하지 않고, 현재 선택된 링크에 대해서만 비용을 계산한다. 그리고 polynomial time의 최단경로 탐색 알고리즘은 노드의 수 및 링크의 수가 증가함에 따라 경로탐색 시간이 급격히 증가하는 단점이 있기 때문에, 본 논문에서는 적절한 최단 경로를 최대한 빠른 시간 내에 찾을 수 있는 휴리스틱 융합 알고리즘을 제안한다.

Ⅳ. 성능 측정 및 평가

본 장에서는 제안된 알고리즘 구현 및 성능 측정 환경에 대해 설명한 후 다양한 파라미터를 변경하여 측정된 결과를 토대로 알고리즘의 성능을 비교 평가한다.

1. 성능 측정 환경

제안된 알고리즘은 C언어를 사용하여 구현되었으며, 성능 측정에는 64개의 노드 및 118개의 링크를 가진 맵이 사용되었다. 각 링크에는 링크의 거리 값뿐만 아니라, 링크의 선호도(preference) 및 비선호도(avoidance)를 나타내는 값이 포함된다. 만약 특정 링크의 선호도 값 및 비선호도의 값이 같다면, 제안된 알고리즘은 이 링크를 일반링크(common link)로 판단한다. 하지만 특정 링크의 선호도 값이 비선호도 값보다 크다면, 이 링크를 선호링크(preferred link)로 판단하고, 이와 반대로 비선호도 값이 선호도 값보다 크다면, 이 링크를 비선호링크(avoidance link)로 판단한다. 성능 측정에 사용된 링크의 수 및 링크의 선호도 및 비선호도 값은 표 1과 같이 설정되었다. 선호도 기반 경로 탐색 결과의 적절한 비교 평가를 위해 각 링크의 선호도 및 비선호도 값은 1 또는 2로 설정되었다.

표 1. 맵을 구성하는 링크의 선호도 및 비선호도 특성
Table 1. Properties of the links of the map.

일반링크의 수		선호링크의 수		비선호링크의 수		총 링크 수
선호도	비선호도	선호도	비선호도	선호도	비선호도	
61		20		37		118
1	1	2	1	1	2	

2. 파라미터 설정 및 성능 측정 방법

제안된 알고리즘의 다양한 파라미터 중 개미의 수(n_k), 노드 선택 시 사용되는 확률 값(r_0), 휴리스틱 함수의 지수(β) 그리고 *MaxACO* 값은 알고리즘의 성능에 큰 영향을 미친다. 이러한 파라미터의 변화에 따른 알고리즘의 성능 변화를 살펴보기 위해 표 2와 같이 다양한 파라미터 셋을 설정하여 성능을 측정하였다.

표 2에서 파라미터 셋 2번은 개미의 개체 수(n_k) 증가에 따른 알고리즘의 성능 변화를 측정하기 위해 설정되었으며, 파라미터 셋 3번은 노드 선택 방법의 확률을 결정하는 값(r_0) 변화에 따른 알고리즘의 성능 변화를 측정하기 위해 사용되었다. 그리고 *MaxACO* 값의 증가에 따른 알고리즘의 성능 변화를 측정하기 위해 파라미터 셋 4번이 사용되었으며, 휴리스틱 함수의 지수 값(β) 변화에 따른 알고리즘의 성능 변화를 측정하기 위해 파라미터 셋 5번이 사용되었다.

모든 성능 측정 과정에서 *MaxTry*의 값은 11, 후보노드(candidate node) 목록의 수는 3으로 설정되었으며, β 값의 증가량 α는 0.3으로 설정되었다. 그리고 시작노드 및 도착노드는 모두 동일하게 설정되었다.

표 2. 성능 측정에 사용된 파라미터 셋
Table 2. Set of parameters for the simulation.

파라미터 셋 번호	*MaxACO*	n_k	r_0	β
1	30	20	0.5	0 ~ 4
2	30	50	0.5	0 ~ 4
	30	70	0.5	0 ~ 4
3	30	70	0.1	0 ~ 4
	30	70	0.9	0 ~ 4
4	10 ~ 70	20	0.5	0 ~ 4
5	10 ~ 70	30	0.5	1.9
	10 ~ 70	30	0.5	2.5
	10 ~ 70	30	0.5	4.0

3. 성능 측정 결과 및 평가

그림 3 및 4는 표 2의 파라미터 셋 1번을 사용하여 측정된 결과를 나타낸다. 그림 3을 통해 초기 경로탐색 결과(β=0)를 살펴보면 선호링크가 포함되어져 있지 않고 비선호링크 및 일반링크로 총 19개의 링크로 구성되어 짐을 알 수 있다. 하지만 β가 1.9로 증가되면서부터 선호링크가 포함된 경로가 탐색되기 시작하고, β 가 2.5가 되는 시점 이후부터는 비선호링크가 포함되지 않은 경로를 탐색하는 것을 알 수 있다. 선호링크가 포함

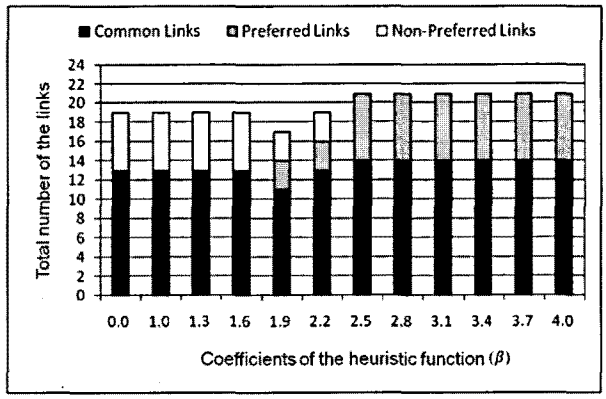


그림 3. β의 증가에 따른 링크 수의 변화 (*MaxACO* = 30, n_k = 20, r_0 = 0.5)

Fig. 3. Total number of the links as a function of β. (*MaxACO* = 30, n_k = 20, r_0 = 0.5)

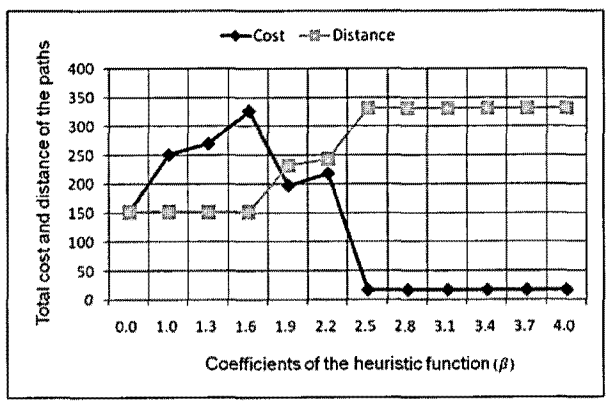


그림 4. β의 증가에 따른 경로의 총 비용 및 거리의 변화 (*MaxACO* = 30, n_k = 20, r_0 = 0.5)

Fig. 4. Total cost and distance of the paths as a function of β. (*MaxACO* = 30, n_k = 20, r_0 = 0.5)

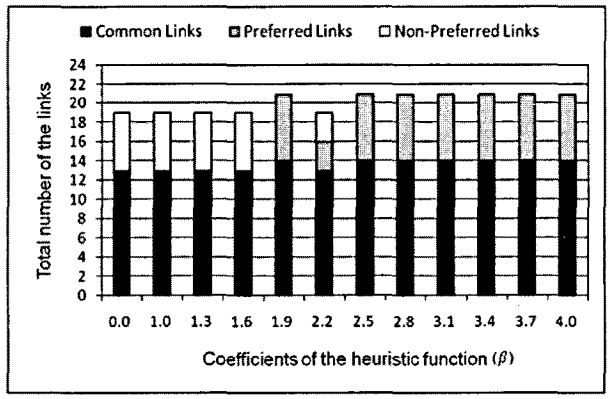


그림 5. β의 증가에 따른 링크 수의 변화 (*MaxACO* = 30, n_k = 50, r_0 = 0.5)

Fig. 5. Total number of the links as a function of β. (*MaxACO* = 30, n_k = 50, r_0 = 0.5)

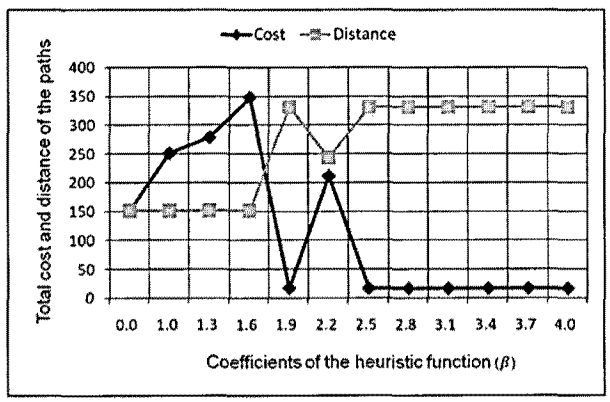


그림 6. β의 증가에 따른 경로의 총 비용 및 거리의 변화 (*MaxACO* = 30, n_k = 50, r_0 = 0.5)

Fig. 6. Total cost and distance of the paths as a function of β. (*MaxACO* = 30, n_k = 50, r_0 = 0.5)

되면서 총 링크의 수가 증가하는 것은 선호링크를 선택하기 위해 우회로가 선택되었기 때문이다.

그림 4를 살펴보면, β 가 1.9로 설정되는 시점부터 비용은 감소하는 것을 알 수 있다. 이는 비용 계산은 거리와 휴리스틱 함수의 곱으로 측정되기 때문이다. 그리고 총 거리가 증가하는 것은 선호링크가 포함되면서 우회로가 포함되기 때문이다.

파라미터 셋 1번을 사용해 성능을 측정한 결과 선호경로는 β 가 1.9일 때부터 탐색되며, β 가 2.5가 되는 시점부터 최대 선호경로가 탐색되고, 이때 거리는 최대가 되고, 비용은 최소가 되는 것을 알 수 있다.

그림 5~8은 파라미터 셋 2번을 사용하여 측정된 결과를 나타낸다. 그림 5 및 6을 통해 개미의 수가 50으로 설정되었을 경우를 살펴보면, n_k = 30 인 파라미터

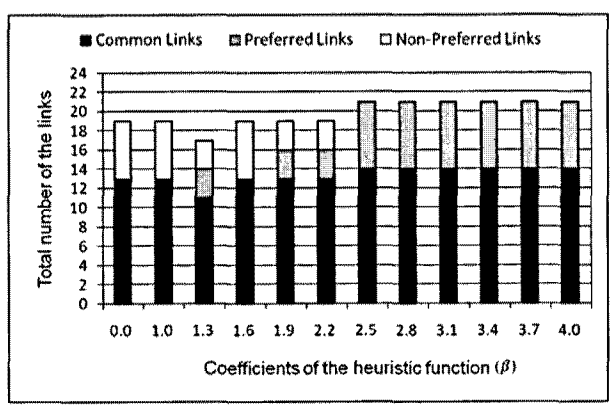


그림 7. β의 증가에 따른 링크 수의 변화 (*MaxACO* = 30, n_k = 70, r_0 = 0.5)

Fig. 7. Total number of the links as a function of β. (*MaxACO* = 30, n_k = 70, r_0 = 0.5)

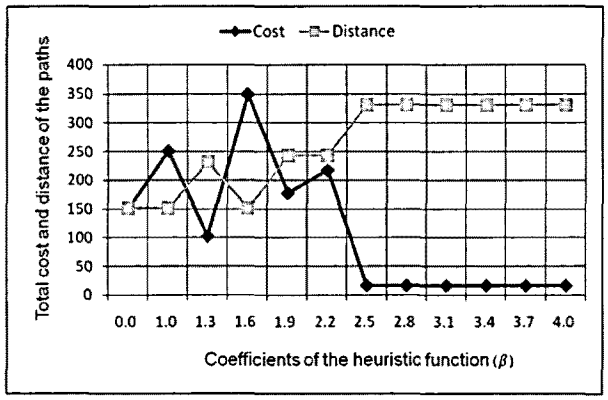


그림 8. β의 증가에 따른 경로의 총 비용 및 거리의 변화 (*MaxACO* = 30, n_k = 70, r_0 = 0.5)

Fig. 8. Total cost and distance of the paths as a function of β. (*MaxACO* = 30, n_k = 70, r_0 = 0.5)

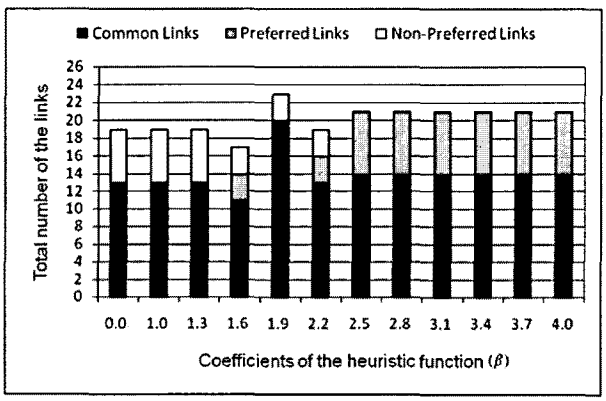


그림 9. β의 증가에 따른 링크 수의 변화 (*MaxACO* = 30, n_k = 70, r_0 = 0.1)

Fig. 9. Total number of the links as a function of β. (*MaxACO* = 30, n_k = 70, r_0 = 0.1)

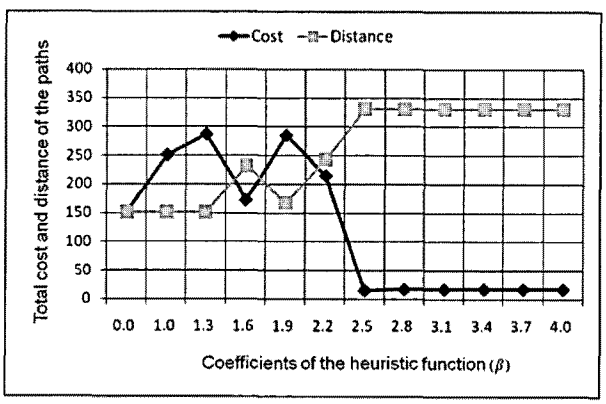


그림 10. β 값 증가에 따른 경로의 총 비용 및 거리의 변화 (*MaxACO* = 30, n_k = 70, r_0 = 0.1)

Fig. 10. Total cost and distance of the paths as a function of β. (*MaxACO* = 30, n_k = 70, r_0 = 0.1)

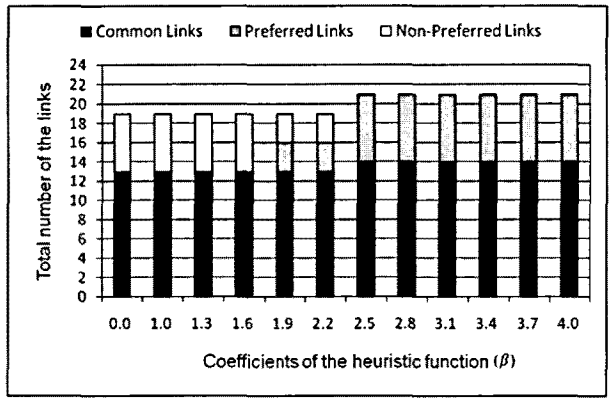


그림 11. β 값 증가에 따른 링크 수의 변화 (*MaxACO* = 30, n_k = 70, r_0 = 0.9)

Fig. 11. Total number of the links as a function of β. (*MaxACO* = 30, n_k = 70, r_0 = 0.9)

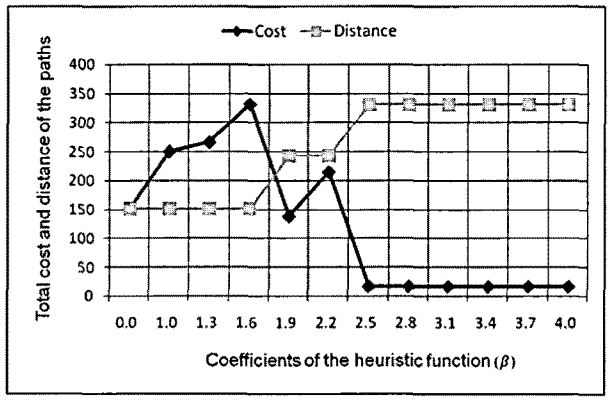


그림 12. β 값 증가에 따른 경로의 총 비용 및 거리의 변화 (*MaxACO* = 30, n_k = 70, r_0 = 0.9)

Fig. 12. Total cost and distance of the paths as a function of β. (*MaxACO* = 30, n_k = 70, r_0 = 0.9)

셋 1번과 달리 β가 1.9가 되는 시점부터 선호경로가 탐색되고 비용은 감소된다. 이는 개미의 수 증가에 따라 탐색되는 경로의 수가 다양해지기 때문이다. 특히 그림 7 및 8을 통해 개미의 수가 70으로 증가되었을 경우 성능 측정 결과를 살펴보면 선호경로는 1.3부터 탐색되어짐을 알 수 있다. 하지만 개미수가 증가하더라도 최대 선호경로는 β가 2.5일 경우부터 수렴되기 시작하는 특징이 있다.

그림 9 ~ 12는 파라미터 셋 3번을 사용하여 측정된 결과를 나타낸다. 그림 9 및 10을 통해 노드 선택 방법의 확률을 결정하는 값(r_0)이 0.1로 설정되었을 경우를 살펴보면, 선호경로는 β가 1.6이 되는 시점부터 탐색되어지며, 최대 선호경로는 β가 2.5부터 탐색되어 수렴하는 것을 알 수 있다. 그리고 총 3가지 종류의 선호경로

가 탐색되어진다.

그림 11 및 12를 통해 r_0가 0.9로 설정되었을 경우를 살펴보면, 선호경로는 β가 1.9가 되는 시점부터 탐색되어지며, 최대 선호경로는 β가 2.5가 되는 시점부터 탐색되어 수렴하는 것을 알 수 있다. 그리고 이 경우 r_0가 0.1인 경우보다 적은 총 2가지 종류의 선호경로가 탐색되어지는데, 이는 r_0 값이 클수록 알고리즘은 후보 노드(candidate node)를 우선적으로 사용하여 경로를 탐색하기 때문이다.

그림 13은 파라미터 셋 4번을 사용하여 측정된 결과를 나타낸다. 그림을 통해 *MaxACO* 값 증가에 따른 알고리즘의 성능 변화를 살펴보면, *MaxACO* 값이 증가함에 따라 평균 선호링크는 증가하는 반면, 평균 비선호링크는 감소하는 것을 알 수 있다. 이는 *MaxACO* 값이 증가함에 따라 경로에 누적되는 페로몬의 크기가 커지기 때문이다.

그림 14 ~ 16는 파라미터 셋 5번을 사용하여 측정된 결과를 나타낸다. 그림을 통해 휴리스틱 함수의 지수 값 변화에 따른 알고리즘의 성능 변화를 살펴보면, 그림 14에서와 같이 β가 1.9일 경우 선호경로는 *MaxACO* 값이 30일 때부터 탐색되며, 최대 선호경로는 *MaxACO* 값이 70이 되는 시점부터 탐색되어 지는 것을 알 수 있다. 반면에 그림 15에서와 같이 β가 2.5일 경우 선호경로는 *MaxACO* 값이 20 되는 시점부터 탐색되어 지며, *MaxACO* 값이 30이 되면 최대 선호경로가 탐색되는 것을 알 수 있다. 그리고 그림 16에서와 같이 β가 4.0인 경우는 *MaxACO* 값이 10이 되는 시점부터 부터 선호경로가 탐색되며, *MaxACO* 값이 20일 때부터 최대 선호경로를 탐색하는 것을 알 수 있다. 따라서 β값이 클수록 작은 값의 *MaxACO* 만으로도 선호도

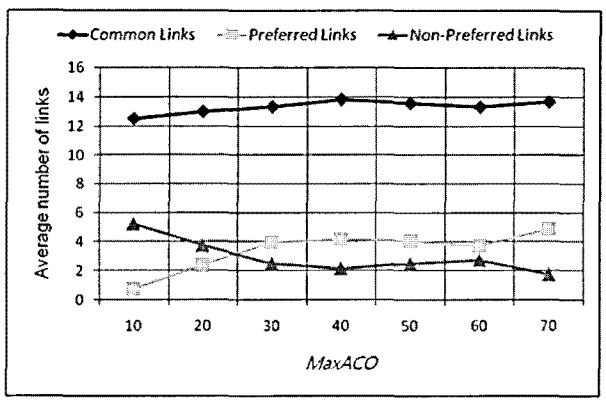


그림 13. *MaxACO* 값 증가에 따른 평균 링크 수의 변화 (n_k = 20, r_0 = 0.5, β = 0 ~ 4)

Fig. 13. Average number of links as a function of *MaxACO*. (n_k = 20, r_0 = 0.5, β = 0 ~ 4)

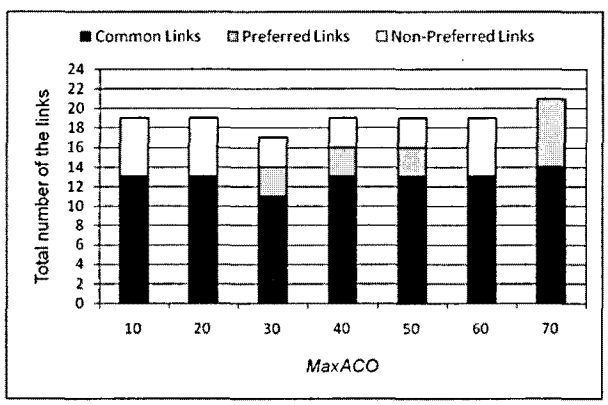


그림 14. *MaxACO* 값 증가에 따른 링크 수의 변화 (n_k = 30, r_0 = 0.5, β = 1.9)

Fig. 14. Total number of the links as a function of *MaxACO*. (n_k = 30, r_0 = 0.5, β = 1.9)

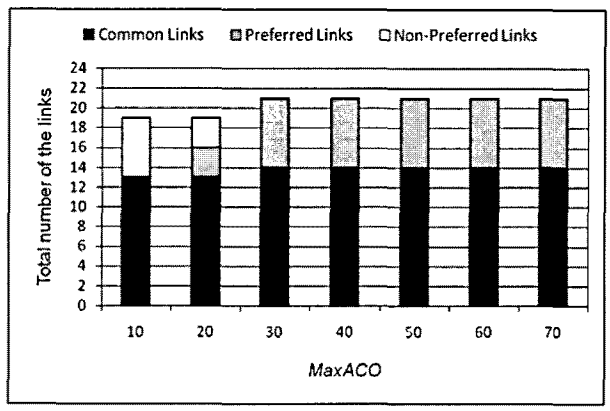


그림 15. *MaxACO* 값 증가에 따른 링크 수의 변화 (n_k = 30, r_0 = 0.5, β = 2.5)

Fig. 15. Total number of the links as a function of *MaxACO*. (n_k = 30, r_0 = 0.5, β = 2.5)

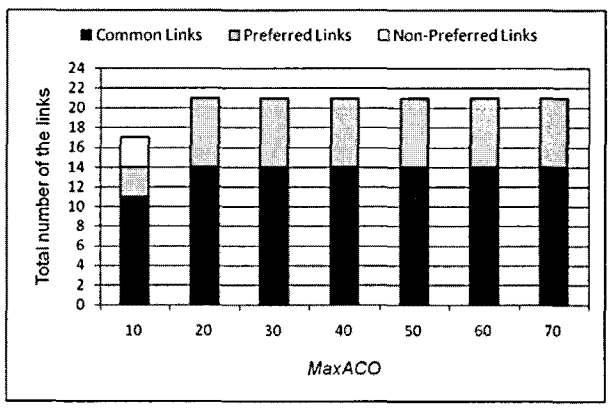
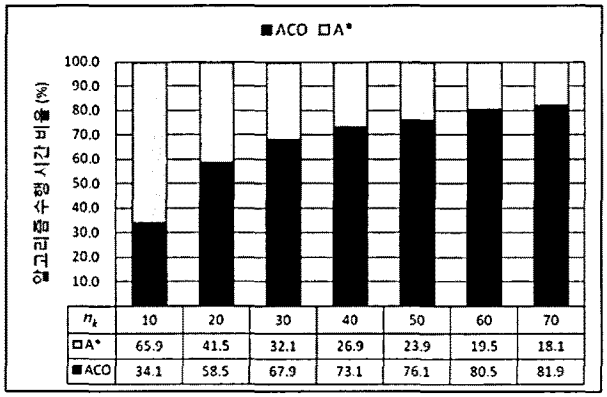


그림 16. *MaxACO* 값 증가에 따른 링크 수의 변화 (n_k = 30, r_0 = 0.5, β = 4.0)

Fig. 16. Total number of the links as a function of *MaxACO*. (n_k = 30, r_0 = 0.5, β = 4.0)



n_k	10	20	30	40	50	60	70
□A*	65.9	41.5	32.1	26.9	23.9	19.5	18.1
■ACO	34.1	58.5	67.9	73.1	76.1	80.5	81.9

그림 17. 개미의 수(n_k) 증가에 따른 A* 및 ACO 알고리즘 수행 시간 비율의 변화 (r_0 = 0.5, β = 4.0)

Fig. 17. The variation of execution time rate between the ACO and A* algorithm as a function of n_k. (r_0 = 0.5, β = 4.0)

에 기반한 경로를 빨리 탐색 할 수 있음을 알 수 있다.

그림 17은 개미의 수(n_k) 증가에 따른 *ACO Loop*의 각 iteration에서 ACO의 평균 수행 시간 및 *Try Loop*의 각 iteration에서 A*의 평균 수행 시간 비율의 변화를 나타낸 것이다. 그림을 통해 *ACO Loop*의 각 iteration 평균 수행 시간 비율을 살펴보면 개미의 수가 10일 경우 34.1%이지만 개미의 수가 70일 경우 ACO의 평균 수행 시간 비율이 81.9%까지 증가하여, 전체 알고리즘 수행에서 ACO가 차지하는 비율이 높은 것을 알 수 있다. 이는 개미의 수가 증가할수록 각 iteration에서 모든 개미들이 경로를 탐색하는데 걸리는 시간이 증가하기 때문이다. 따라서 적절한 개미 수의 선택이 전체 알고리즘의 수행 시간을 결정짓는 중요한 요소인 것을 알 수 있다.

V. 결 론

본 논문에서는 기존의 최단경로 탐색을 위한 A* 및 ACS 알고리즘을 융합하여, 최단경로뿐만 아니라 사용자의 링크 선호도에 기반한 경로 탐색이 모두 가능한 알고리즘을 제안하였으며, 다양한 파라미터를 사용하여 성능을 측정 및 평가하였다.

성능 측정 및 평가 결과 제안된 알고리즘은 휴리스틱 함수의 지수(β)값이 0일 경우 A* 알고리즘에 의한 최단경로를 탐색하며, 이후 β가 증가함에 따라 최단경로 탐색 결과에서 선호링크 수는 증가하고, 비선호링크의 수는 감소하는 특징을 나타내었다. 그리고 본 알고리즘은 파라미터의 변화에 따른 성능변화가 크지 않아, 비교적 적은 수의 개미 및 작은 값의 *MaxACO*를 사용해서도 선호경로를 찾는 특징을 나타내었으며, β 값이 클수록 작은 값의 *MaxACO* 만으로도 선호도에 기반한 경로를 탐색 할 수 있음을 알 수 있었다.

제안된 알고리즘은 기존의 최단경로 탐색을 위한 시스템뿐만 아니라, 선호도에 기반한 경로 탐색이 필요한 다양한 시스템에 적용될 수 있을 것으로 기대된다.

참 고 문 헌

[1] 최병호, 정문호, 전희영, "T-DMB 상용 교통정보 서비스 시스템 소개," 대한전자공학회, 전자공학회지, 제35권, 제9호, 106-118쪽, 2008년 9월

[2] Sara Nazari, M.Reza Meybodi, M. Ali Salehigh, Sara Taghipour, "An Advanced Algorithm for Finding Shortest Path in Car Navigation System," International Workshop on Intelligent Networks and Intelligent Systems, pp. 671-674, Wuhan, China, Nov. 2008.

[3] Hao Yue, Chunfu Shao, "Study on the Application of A* Shortest Path Search Algorithm in Dynamic Urban Traffic," Third International Conference on Natural Computation, vol. 3, pp. 463-469, Haikou, Hainan, China, Aug. 2007.

[4] Noto, M.; Sato, H. "A method for the shortest path search by extended Dijkstra algorithm," Proc. of IEEE International Conference on Systems, Man and Cybernetics, Nashville, TN, vol. 3, pp. 2316 - 2320, Oct. 2000.

[5] Salehinejad, H. Talebi, S. "A new ant algorithm based vehicle navigation system: A wireless networking approach," International Symposium on Telecommunications, pp. 36-41, Tehran, Iran, Aug. 2008.

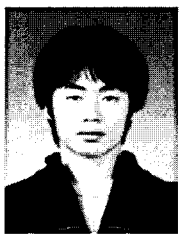
[6] M Dorigo, V Maniezzo, A Colorni, "Ant system: optimization by a colony of cooperating agents," IEEE Transactions on Systems, Man, and Cybernetics-Part B, vol. 26(1), pp. 29-41, 1996.

[7] LM Gambardella, M Dorigo, "Solving Symmetric and Asymmetric TSPs by Ant Colonies," Proceedings of the IEEE Conference on Evolutionary Computation, pp. 622-627, Nagoya, Japan, May. 1996.

[8] 안진호, 김홍식, 김현진, 박영호, 강성호, "규칙적인 NoC 구조에서의 네트워크 지연 시간 최소화를 위한 어플리케이션 코어 매핑 방법 연구," 대한전자공학회, 전자공학회논문지 SD편, 제45권, 제4호,

117-123쪽, 2008년 4월
[9] Marco Dorigo, Thomas Stützle, "Ant Colony Optimization," pp. 76-79, MIT Press, 2004.

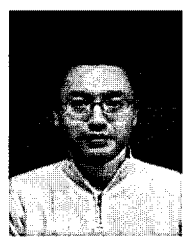
저 자 소 개



옥 승 호(정회원)
2006년 동의대학교 메카트로닉스 공학과 학사 졸업.
2008년 경북대학교 전자공학과 석사 졸업.
2010년 현재 경북대학교 전자전기 컴퓨터학부 박사과정.
<주관심분야 : SoC 설계 및 응용, 디지털 VLSI>

안 진 호(정회원)
1995년 연세대학교 전기공학과 학사 졸업.
1997년 연세대학교 전기공학과 석사 졸업.
2002년 엘지전자 DTV연구소 연구원.
2006년 연세대학교 전기공학과 박사 졸업.
2010년 현재 호서대학교 전자공학과 교수.
<주관심분야 : SoC 설계 및 응용, 테스트>

강 성 호(평생회원)
1986년 서울대학교 제어계측 공학과 학사 졸업.
1988년 The University of Texas, Austin 전기 및 컴퓨터 공학과 석사 졸업.
1992년 The University of Texas, Austin 전기 및 컴퓨터공 학과 박사 졸업.
1992년 미국 Schlumberger 연구원.
1994년 미국 Motorola 선임연구원.
2010년 현재 연세대학교 전기전자공학과 교수.
<주관심분야 : SoC 설계 및 테스트>



문 병 인(평생회원)-교신저자
1995년 연세대학교 전자공학과 학사 졸업.
1997년 연세대학교 전자공학과 석사 졸업.
2002년 연세대학교 전기전자 공학과 박사 졸업.
2002년~2004년 하이닉스반도체 선임연구원.
2004년~2005년 연세대학교 연구교수.
2005년~현재 경북대학교 전자공학부 조교수.
<주관심분야 : SoC, 디지털 VLSI, 컴퓨터 구조>