2606 大韓機械學會論文集 第18卷 第10號, pp. 2606~2615, 1994

〈論 文〉

IC 테스트 핸들러의 최적분류 알고리즘 개발

김종관* · 최동훈**

(1994년 4월 15일 접수)

An Optimal Sorting Algorithm for Auto IC Test Handler

Jong-Kwan Kim and Dong-Hoon Choi

Key Words : Optimal Sorting(최적분류), Artificial Intelligence(인공지능), IC Test Handler(IC 테스트 핸들러), Heuristic Search(발견적 탐색), State Space(상태공간), Best-First Search(최적우선 탐색기법)

Abstract

Sorting time is one of the most important issues for auto IC test handling systems. In actual system, because of too much path, reducing the computing time for finding a sorting path is the key way to enhancing the system performance. The exhaustive path search technique can not be used for real systems. This paper proposes heuristic sorting algorithm to find the minimal sorting time. The suggested algorithm is basically based on the best-first search technique and multi-level search technique. The results are close to the optimal solutions and computing time is greately reduced also. Therefore the proposed algorthm can be effectively used for real-time sorting process in auto IC test handling systems.

I. 서 론

반도체 제조 공정중 마지막 공정인 IC 검사 공정은 전기적검사 및 외관검사로 구분되며 일반적으로 전수검사를 행하고 있다. 전기적검사는 IC가 가진 고유의 전기적 특성이 사양에 맞는가를 검사하는 공정이고, 외관검사는 마킹(marking) 또는 IC 리드의 변형상태 등을 검사하는 공정이다. IC를 전기적으로 검사하는 경우 일정한 검사시간이 요구되고, 검사결과에 따라 2~9등급의 부류(category)를 갖는다. 최근 IC 검사기의 발전과 공정의 개선으로 검사시간이 점점 짧아짐에 따라 IC 테스트 핸들러(IC test handler)에서의 분류시간이 이 중요한 의미를 가지게 되었다. 따라서 빠른 분류기구(sorting mechanism)와 분류시 최단거리로 이송해서 분류시간을 단축하는 방안이 요구된다. 어떤 경로에 대한 최단거리문제를 다룬 문헌[1~3]들은 많이 찾아볼 수 있으나, 자동 IC 테스트 핸들러의 분류시간 문제에로의 적용이 어렵다. 본 연구에서는 분류시간을 최소화하는 알고리즘을 개발하여 IC 자동 분류 시스템의 성능향상 및 최소 운동거리로 인한 장비의 수명연장에 기여하고자한다. IC 분류 시간을 최소로 하기 위하여 인공지능 기법을 이용하여 해를 구하고자 하며 이의 실행은 IBM RS-6000 워크스테이션에서 C언어로 하였다. 문제의 해를 구하기 위하여 우선 문제를 상태공간으로 표현하고, 시간에 대한 제한조건하에서 경로를 찾아감으로써 해를 구하고자 한다. 탐색방법에는 기본이 되는 깊이우선 탐색기법(depth-first search)과 넓이우선 탐색기법(breadth-first search) 등의 방법[4,5]이 있다. 이러한 방법 등은 본 연구에서 설

*한양대학교 대학원 기계공학과
**정회원, 한양대학교 기계설계학과

정된 분류문제를 해결하기에는, 비록 최적의 해를 찾을 수 있더라도, 너무 많은 경우의 수로 인한 계산시간이 과다 소요되어 실시간 처리가 어렵다.

분류경로 계산시간은 장비의 CPU 성능이 허용하는 제한된 시간 이내에 결과를 산출할 수 있어야 한다. 경우의 수의 조합에 따른 복잡성을 해결하기 위한 방법중의 하나인 경험적 정보를 사용하여 탐색하는 발견적 탐색기법을 사용하여 분류시간을 주어진 시간내에서 최소로 하는 해를 구하도록 하되 분류 방법의 수가 제한 계산시간 이내인 경우는 최적해를 찾고저 한다.

본 논문에서는 분류기구의 개념을 설명한 후, 분류기구의 모델링을 통하여 문제를 상태공간으로 표현하여 개발된 분류 알고리즘을 기술하고자 하며, 예제들에 적용하여 최적해와 평가함수, 분류 계산시간 등을 비교함으로써 개발된 소프트웨어의 유용성을 보이고자 한다.

2. IC 분류기

2.1 분류기 구성

대량생산을 하고있는 IC는 보통 전자동으로 시험하고 시험결과에 따라 전자동으로 분류하고 있다. IC 테스트 핸들러의 개략적 구조는 Fig. 1와 같다. 시스템의 주요 기능은 전기적인 시험을 하기 위하여 피측정 IC를 자동으로 공급하여 시험기에 연결시키고, 측정결과에 따라 소정 등급이 정해진 곳으로 분류시켜 주는 동작을 한다. 여러 가지의 기능을 가지고 있지만 그중 분류기가 가장 중요하다. 분류기의 운동은 스텝모터에 의하고 이송은 타이밍 벨트로 하였다. Fig. 2는 분류기의 동작제어를 보여 주고 있으며 분류조건이 만족되면 분류 작업을 수행해야 하는데 이 때 어떻게 분류할 것이냐 하는 문제가 발생 하게된다. 분류순서에 따라 분류시간의 차이가 발생하므로 가능한 최단거리(또는 최단시간)을 소요하는 경로가 필요하게 된다.

2.2 분류기의 환경

검사가 끝난 IC들은 Fig. 3같이 8개의 줄로 구성된 앞 대기장소(front buffer)에 채워진다. 분류기는 좌우 왕복운동을 하며 한번에 2개의 IC를 앞 대기장소로부터 장착하여 저장소로 탈착한다. 저장소는 8개의 줄로 구성되며 각각의 줄은 특정 등급을 갖고 있다. 검사가 끝난 후 앞 대기장소에 대기

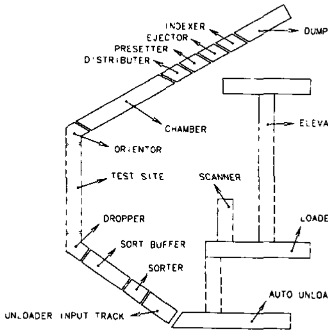

Fig. 1 Structure of IC test handling sysytem

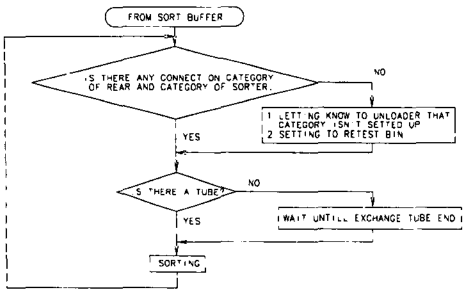

Fig. 2 Sorting control flow chart

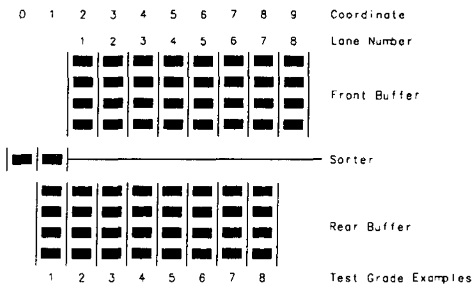

Fig. 3 Sorter coordinate

한 IC들은 검사결과에 따라 등급을 갖게되며, 분류기는 각각의 IC들의 등급과 일치하는 저장소의 해당 줄로 이동하여 탈착한다.

문제에로의 접근과 해를 구하기 위하여 분류기의 환경에 대한 규칙 또는 제한조건을 Fig. 2에 따라서 정해야 한다. 이를 정리하면 다음과 같다.

(1) 줄의 수 : 8개로 고정한다.

(2) IC의 등급 : 1에서 8까지의 값이 가능하며 불량일 경우는 0으로 한다.

(3) 분류기의 용량은 2로 한다.

(4) 분류기의 장착 동작 : 용량이 2인 분류기라면 단일 동작에 2개를 장착한다.

예) 분류기의 용량이 2인 경우 앞 대기장소로부터 장착할때 가능한 줄 번호들의 쌍은 앞 대기장소가 8줄로 구성되어 있어 4개의 조합을 만들 수 있다.

〈1, 2〉, 〈3, 4〉, 〈5, 6〉, 〈7, 8〉

(5) 분류기의 탈착 동작 : 개별적 또는 동시적인 탈착이 가능하다.

예) 분류기의 용량이 2인 경우에 가능한 동작들
좌측 IC만 탈착,
한번에 둘 다 탈착,
우측 IC만 탈착 등의 3가지 경우가 있다.

(6) 분류기의 운동 : 0부터 9까지의 X좌표로 이동하는 것이 가능하다.

(7) 분류기의 가속도 : 가변적인 입력.

(8) 분류기의 등속도 : 가변적인 입력.

(9) 줄간의 거리 : 가변적 입력.

(10) 분류기는 앞 대기장소의 어느 세로 한 줄을 처리한 후 다음 줄을 처리한다.

예) 앞 대기장소에 IC들이 있다면 세로 줄 번호가 1인 모든 IC들을 분류하고 2번을 처리해야 한다.

								세로 줄 번호
17	18	19	20	21	22	23	24	3
9	10	11	12	13	14	15	16	2
1	2	3	4	5	6	7	8	1

(11) 분류기에 IC가 장착되어 있을 경우는 더 이상 장착할 수 없다.

(12) 앞 대기장소 및 저장소의 용량 : 각 8개와 9개

분류기의 용량을 2라고 가정할 때, 2개의 IC를 장착하여 분류하는 과정을 설명하면 다음과 같다. 1번 줄에 3등급의 IC가 있고, 2번 줄에는 5등급의 IC가 존재한다고 가정하자. (Fig. 3 참조)

분류공정(sorting procedure)

STEP 1) 분류기를 해당위치(X좌표=2)로 이동
Move to 2

STEP 2) 이웃한 두개의 IC를 한번에 장착(1번과 2번 줄에 위치한 IC)
Load Left & Right ICs

STEP 3) 좌측 IC의 등급과 같은 저장소의 줄로 이동
Move to 3

STEP 4) 좌측 IC를 저장소로 탈착
Unload Left IC

STEP 5) 우측 IC의 등급과 같은 저장소의 줄로 이동
Move to 4

STEP 6) 우측 IC를 저장소로 탈착
Unload Right IC

위의 6개 동작 절차로 한번 장착한 IC의 분류공정이 종료된다.

3. 분류기의 모델링

3.1 상태공간 표현

상태공간을 표현하기 위하여 분류기의 좌표를 Fig. 4와 같이 표준화 하였다. 또한 한 쌍의 IC를 분류기에 넣은 후 왼쪽과 오른쪽의 순서를 결정해야 되기 때문에 쌍(pair) 표현법을 사용하였다.

쌍 표현법
· (L, R)
— L : 왼쪽 IC가 분류될 위치
— R : 오른쪽 IC가 분류될 위치

Fig. 4와 같이 IC 등급이 앞 대기장소가 좌측으로 부터 등급 1, 2, 3, 3, 2, 5, 5, 6을 순서대로 가지며, 저장소가 1, 1, 1, 2, 2, 2, 3, 5, 6을 가지고 있는 경우, 분류기가 이동 가능한 모든 쌍을 구하면 Table 1과 같고, 분류과정을 설명 하면 앞 대기장소에서 분류기가 좌측 A의 IC를 뒤버퍼에 분류할 수 있는 경우는, 우선 A의 등급이 1, 2 이므로 등급 1인 좌측 IC를 저장소 좌표 $L1$로 위치해 분류가 가능하고 우측 IC는 $R3$이 될 것이다. 따라서 이동 좌표는 $L1R3$이 된다. 이와 같은 방법으로 A의 분류 총 경우는 Table 1에서 처럼 9종류가 되나, 우측을 먼저 분류하는 경우를 고려하면, 분류기의 용량이 2이므로 이 두배인 18이 된다. 이러한 분류 과정을 상태공간으로 표현하면

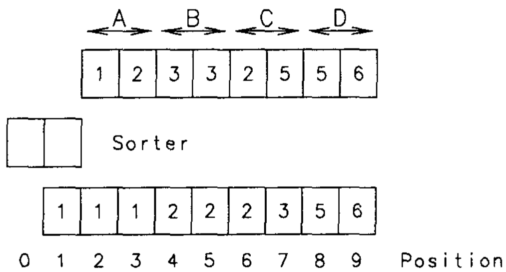

Fig. 4 Sorter coordination to represent state space

Table 1 Release pair

Nodes	Release pair	Total case
A	1) L1R3	9×2=18
	2) L2R3	
	3) L3R3	
	4) L1R4	
	5) L2R4	
	6) L3R4	
	7) L1R5	
	8) L2R5	
	9) L3R5	
B	1) L7R6	1×2=2
C	1) L4R7	3×2=6
	2) L5R7	
	3) L6R7	
D	1) L8R8	1×2=2

Fig. 5와 같이 표현할 수 있다.

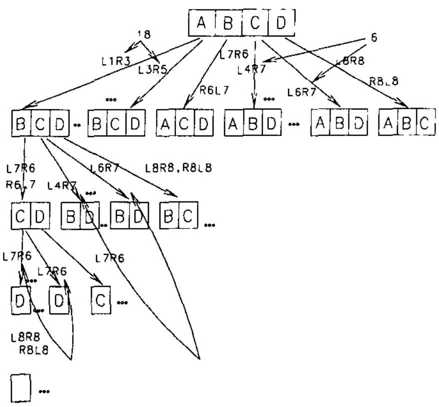

Fig. 5 State space of sorter

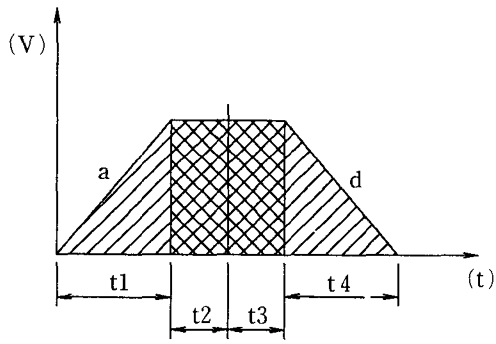

Fig. 6 Acceleration and deceleration profile

3.2 평가함수

분류기 문제에서 최소화하고자 하는 목표는 시간이므로 최소화하고자 하는 평가함수를 시간 T로하고, 속도, 가속도, 감속도 등을 입력해서 평가함수 T를 계산할 수 있다. 이동거리를 S, 이동 시간을 T, 속도를 V, 가, 감속 시간을 각각 t_1, t_2로 하고 가속도를 a, 감속도를 d로 했을때 Fig. 6과 같이 주어진 거리의 반은 가속후 등속운동을 하게되고, 나머지 반은 등속운동후 감속운동을 하게된다. 운동거리에 따라 등속운동이 가능한 경우와 그렇지 않은 경우가 있다. 처음 반의 운동시간 계산은 등속운동이 가능 하면, 즉 $1/2S > V^2/2a$이면,

$$t_1 = V/a \tag{1}$$
$$t_2 = (S - at_1^2)/2V \tag{2}$$

이고, 그렇지 않으면

$$t_1 = \sqrt{s}/\sqrt{a} \tag{3}$$
$$t_2 = 0 \tag{4}$$

나머지 반의 시간계산은 등속운동이 가능하면, 즉 $1/2S > V^2/2d$이면

$$t_4 = V/d \tag{5}$$
$$t_3 = (S - dt_4^2)/2V \tag{6}$$

이고, 그렇지 않으면

$$t_4 = \sqrt{s}/\sqrt{d} \tag{7}$$
$$t_3 = 0 \tag{8}$$

이 된다. 따라서 평가함수 T는

$$T = t_1 + t_2 + t_3 + t_4 \tag{9}$$

가 된다.

4. 탐색이론

인공지능기법을 이용한 문제풀이 방법은 초기 상태에서 시작하여 문제 공간을 효과적, 효율적으로 탐색해 나감으로써 원하는 목표를 찾아내는 연산의 순서를 규정하는 과정을 의미하고, 이러한 방법을 사용하면 탐색기법이 매우 중요하다. 왜냐하면 수많은 해들이 놓인 상태공간에서 이 중 하나의 해를

찾음으로써 문제를 풀기 때문이다. 이와 같이 상태공간에서 주어진 문제를 효과적으로 해결하기 위해서는, 주어진 문제에 대한 초기 상태와 목표 상태를 정확히 표현하여 문제에 대한 해를 얻을 수 있다. 주어진 문제를 해결하기 위하여 초기상태와 목표상태를 정확하게 표현한 다음에는 초기 상태에서 목표 상태까지 탐색하여 효과적으로 도달할 수 있는 연산자를 사용하여야 한다. 상태공간의 탐색에 의한 문제풀이 방법 중에서는 최적의 연산자를 얼마나 효과적, 효율적으로 작성하느냐가 매우 중요한 문제 중의 하나다. 그 이유는 연산자는 어느 한 상태를 다른 한 상태로 바꾸어 주는 역할을 하기 때문이다.

4.1 탐색기법

깊이우선 탐색(depth-first search)기법은 초기점(출발점)를 가지고 시작하여 목표를 찾거나 아니면 경우의 수가 너무 많아 더 이상 수행이 불가능할 때 까지 탐색을 해나간다. 탐색의 방향을 결정하기 위해 어떤 규칙(즉 '맨 좌측의 가지부터 뒤져라')이 쓰인다. 마지막점에 도달하게 되면 한 단계위로 올라와서 그 다음에 있는 우측 가지를 뒤지게 되는데 이를 역 추적(back-tracking)이라 한다. Fig. 7은 깊이우선 탐색기법을 적용한 예이다. 깊이우선 탐색기법은 목표까지의 레벨의 수가 평균적으로 많은 상태공간에서 효과적이다. 그렇지만 이의 가지 구조가 복잡하고 목표상태가 상대적으로 낮은 단계에 위치하는 경우 비효율적인 기법이 된다. 본 연구에서는 최적해를 찾는데 사용하였다.

넓이우선 탐색(breadth-first search)기법[4]은 Fig. 8과 같이 초기점(출발점)을 가지고 시작하여, 초기점의 모든 후계점들을 선정하여 그래프의 왼쪽에 있는 후계점부터 첫번 단계을 확장하고, 그 다음에 두번째 왼쪽에 있는 후계점 등을 계속해서 확장한 후 다음 단계로 동일한 방법으로 탐색하는 방법이다. 넓이우선 탐색은 상대적으로 낮은 단계에 목표가 있을 경우에 유리하다.

최적우선 탐색기법[7]은 깊이우선 탐색의 장점과 넓이우선 탐색의 장점을 결합한 것이다. 이 과정은 한 점에서 새 점으로 탐색을 확장할 때 현재 평가값에 의해 가장 가능성이 높은 점를 선택하게 된다. 비용함수라 불리는 평가함수를 계산하여 현재점의 "가능성"을 평가한다.

깊이우선 및 넓이우선 탐색기법의 커다란 문제점

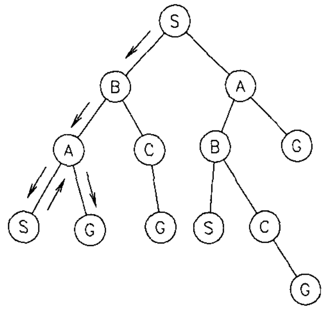

Fig. 7 Depth-first search technique

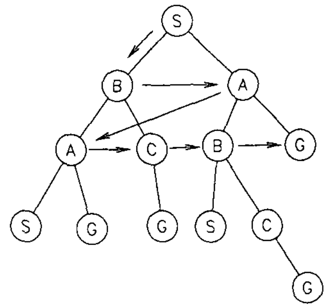

Fig. 8 Breadth-first search technique

은 문제해를 찾기 위하여 너무 많은 점이 확장되기 때문에, 탐색을 위해서 소비되는 시간 및 기억공간이 많이 소비된다. 그러므로 어떤 주어진 문제에 대한 해를 제한된 계산시간과 기억공간을 가지고 해결할 수 있는 탐색기법을 생각한 것이 발견적 기법이다. 즉 주어진 문제와 관련이 있는 정보를 이용하여 평가함수를 만들어서 이를 이용하여 효율적으로 탐색하는 기법을 발견적 탐색기법이라 말한다.

이러한 발견적 탐색기법은 해결경로가 존재할 시, 반드시 최소비용을 가지고 초기점부터 목표점까지 탐색하는 것을 보장하지 못한다는 특성을 가지고 있다. 실제의 문제를 해결시에는 목표점를 탐색하는데 사용되는 경로비용과 이러한 경로를 찾기 위하여 요구되는 탐색 노력을 합하여 전체적인 비용을 최소화하고자 하고 있는 실정이고, 본 연구에서도 이러한 개념을 도입하여 깊이 우선 탐색 기법과 최적우선의 탐색기법을 혼용하여 알고리즘을 개발하였다.

발견적 탐색기법에 사용되는 평가함수는 주어진 문제와 관련하여, 적합한 정보를 이용하여 의미있는 값을 계산하여 내는 것이다. 발견적 탐색기법에

서 평가함수를 사용하는 목적은 주어진 문제해결을 위하여 두 개 이상의 경로가 존재할 시, 어떤 경로에 우선 순위를 두어서 탐색과정을 효율적이며 효과적인 방향으로 유도할 것인가를 결정하기 위함이다.

4.2 수정된 최적우선 탐색기법

Fig. 5와 같이 탐색공간은 IC등급을 어떻게 정하느냐에 따라 달라지고, 최악의 경우에 모든 상태공간을 탐색하는데 시간이 많이 소요되므로, 상태공간의 수에 따라 적절한 알고리즘을 선택할 수 있는 발견적 탐색기법을 사용하였다. 본 연구에서는 깊이우선 탐색기법을 최적우선 탐색기법에 도입하여 수정된 최적우선의 탐색기법을 개발하여 자동 분류 시스템에 적용하였다. 모든 상태공간을 탐색하기 전에 분류방법의 수를 계산할 수 있으므로, 알고리즘의 실시간 처리를 위하여 CPU의 성능을 고려하여 제한 범위를 두어 분류방법이 제한범위 내에 있으면 최적해를 찾도록 하고, 그렇지 않으면 상태공간을 최적우선의 탐색기법으로 1단계씩 검사하고, 다시 1단계 더 검사하여 경로를 찾도록 하여, 탐색공간에 있는 점들에 대한 더 많은 정보를 제공하여 탐색 능력을 향상시켰다. 입력이 무작위로 발생하는 경우가 분류 시스템에서는 통상적이므로, 탐색경로의 수 또한 제한범위 이내일 수도 있고 아닐 수도 있다. 본 알고리즘은 이에 능동적으로 대처해 주어진 시간내에 최적 또는 최선의 해를 찾을 수 있도록 하였다. Fig. 9에서 제안된 알고리즘의 개략적 흐름도를 표시하였다.

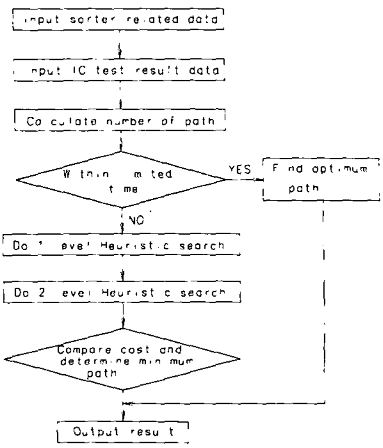

Fig. 9 Algorithm flow chart

5. 결 과

5.1 프로그램 수행과정

개발된 프로그램을 수행하려면 분류기의 운동에 관한 데이터 즉 속도, 가속도 및 운동거리 등이 우선 입력되어야 한다. 또한 분류기가 목표지점에 도달하면 IC가 분류기로 들어오는 시간과 나가는 시간도 입력되어야 한다. 프로그램을 수행시키면 Fig. 10과 같은 입력문이 출력된다. 프로그램의 수행능력을 평가하기 위해 샘플수를 사용자가 임의로 정하여 입력할 수 있도록 하였으며, 실 적용을 위하여 CPU의 성능을 고려하여 경로계산 제한 회수를 입력할 수 있도록 하였다.

데이터의 입력이 완료되면 다음과 같은 입력문이 출력된다.

Choose Category?
1) Random
2) Assigned value

이는 저장소의 등급을 무작위로 선택할 수도 있고, 지정된 값을 입력할 수도 있음을 의미한다. 즉 1)을 선택하면 난수발생에 의하여 저장소의 등급을 결정해주고, 2)를 선택하면 사용자가 임의로 등급을 정할 수 있으며, 아래와 같은 입력문이 출력된다.

Input Category
Category1=?
Category2=?
Category3=?
Category4=?
Category5=?
Category6=?

Velocity (cm/sec) = ?
Acceleration rate (cm/sec^2) = ?
Deceleration rate (cm/sec^2) = ?
Lane distance (cm) = ?
Drop time to sorter (msec) = ?
Drop time from sorter (msec.) = ?
Number of sample data = ?
Constraint of computation = ?

Fig. 10 Input data message

Category7=?
Category8=?
Category9=?

이 때 정해진 등급을 입력하면 된다. 이 입력이 종료되면 다음과 같은 입력문이 출력된다.

Choose Testing Results?
1) Random
2) Assigned Value

IC 검사결과는 무작위로 나타난다고 볼 수 있지만 실제 현장에서는 어떤 경향(비율)을 가지고 있는 경우가 많다. 따라서 무작위인 경우와 등급의 생성 비율에 따른 두가지의 경우 중에 선택할 수 있도록 하였다. 1)을 선택하면 난수발생에 의하여 검사결과 등급을 생성해주고, 2)의 선택시 등급생성 비율을 입력할 수 있다. 이때 다음과 같이 출력된다.

Input Testing Result Ratio
Grade1=?
Grade2=?
Grade3=?
Grade4=?
Grade5=?
Grade6=?
Grade7=?
Grade8=?
Grade9=?

이상의 과정을 거쳐 프로그램이 수행된다.

Table 2 Input data

Velocity	100 cm/sec
Acceleration rate	100 cm/sec^2
Deceleration rate	100 cm/sec^2
Lane distance	5 cm
Drop time to sorter	100 m/sec
Drop time from sorter	100 m/sec
Number of sample data	100

5.2 예제

본 연구에서는 프로그램의 효용성을 평가하기위해 4가지 경우의 예제에 대하여 적용하였으며, 이에 사용될 분류기의 운동에 관한 데이타 및 샘플수는 Table 2와 같이 100을 입력하였다. 결과는 깊이우선 탐색기법에 의한 최적해와 비교하였다. 탐색회수 제한은 0, 500, 1000, 2000, 4000, 8000, 16000 등 총 7개의 경우로 하여 프로그램을 수행하였다. 최적해와의 비교를 위하여 IBM RS-6000 워크스테이션에서 C언어로 수행하였다. 탐색시 발생할 수 있는 경우의 수는 작게는 수백에서 크게는 수백만 가지 이상이 발생할 수 있다. 다음의 여러 예제에서 제안된 알고리즘의 효용성을 알 수 있었다.

[예제 1] 앞 대기장소와 저장소의 등급을 무작위로 했을 경우 Table 3에 이의 결과를 도시하였다.

Table 3 Results of example 1

Constraint of computation	New algorithm		Optimum value		Number of Best method selected
	Average sorting time (sec)	Average computation time (sec)	Average sorting time(sec)	Average computation time(sec)	
0	8.3807	0.0026	7.6289	1.5986	0
500	8.3631	0.0066	7.6281	1.5870	1
1000	8.3602	0.0276	7.6289	1.6196	5
2000	8.2996	0.0705	7.6353	1.6397	14
4000	8.1281	0.2129	7.6701	1.5571	39
8000	8.0145	0.4562	7.7021	1.6125	62
16000	7.8722	0.9308	7.7441	1.6089	96

경로 탐색회수 제한이 0인 경우, 즉 경로탐색을 제안된 알고리즘만으로 수행한 결과 평균 분류시간은 최적해와 비교하여 각각 8.3631 sec, 7.6281 sec로 9.8530%의 차이를 보이는 반면 계산시간은 제안된 알고리즘이 불과 0.0066 sec인데 비해 최적해를 찾는데는 1.5870 sec나 소요되었다. 알고리즘 특징상 경로탐색 회수를 증가시키면 제안된 알고리즘의 계산시간은 증가하고 분류시간은 감소한다. 이는 최적해가 선택되는 경우가 증가하기때문인데, 따라서 시스템 제어기의 계산 능력에 따라 제한회수를 적정히 정하면 보다 최적해에 가까운 결과를 낼것이다. 제한회수를 16000으로 한 경우 최적해가 선택된 경우는 86회였고 분류시간 차이는 1.6542%에 불과했다.

[예제 2] 시험결과는 무작위로 입력하고 저장소의 등급은 지정해 준 경우 저장소의 등급을 순서대로 1, 1, 2, 3, 4, 5, 6, 7, 0으로 입력하였으며, 이의 결과는 Table 4에 도시하였다. 이때의 최대 경로수는 3072로 비교적 작아 경로탐색 제한회수가 4000 이상인 경우 100%의 최적해를 선택했으며, 경로탐색 제한회수를 0으로 하였을 경우 최적치와 비교하여 4.370%를 보였지만 계산시간은 121.82배 향상시킬 수 있었다.

[예제 3] 앞 대기장소에 IC 등급을 지정해서 입력하고, 저장소의 등급을 무작위로 했을 경우 이 경우는 Table 5에서 보듯이 분류시간은 최대 8.58%의 차이를 보였고, 최대 분류수는 98034로 비교적 많아 탐색 제한회수가 16000번의 경우에도 최적해가 선택된 경우가 5번에 불과하였다. 제안된 알고리즘의 계산시간은 0.0071 sec~0.7100 sec 범위 내

Table 4 Results of example 2

Constraint of computation	New algorithm		Optimum value		Number of Best method selected
	Average sorting time (sec)	Average computation time (sec)	Average sorting time (sec)	Average computation time (sec)	
0	7.8860	0.0017	7.3636	0.2071	0
500	7.6811	0.0540	7.3596	0.2120	43
1000	7.5187	0.1205	7.3599	0.2147	77
2000	7.4593	0.1625	7.4096	0.2084	92
4000	7.4100	0.2196	7.4100	0.2142	100
8000	7.4100	0.2196	7.4100	0.2142	100
16000	7.4100	0.2196	7.4100	0.2142	100

Table 5 Results of example 3

Constraint of computation	New algorithm		Optimum value		Number of Best method selected
	Average sorting time (sec)	Average computation time (sec)	Average sorting time (sec)	Average computation time (sec)	
0	7.0054	0.0071	6.4517	6.8089	0
500	7.0054	0.0058	6.4517	6.9218	0
1000	7.0054	0.0084	6.4517	6.9004	0
2000	7.0054	0.0023	6.4517	6.9861	0
4000	7.0054	0.0718	6.4517	7.0174	0
8000	6.9955	0.0910	6.4531	7.0799	1
16000	6.9672	0.1700	6.4539	7.0459	5

Table 6 Results of example 4

Constraint of computation	New algorithm		Optimum value		Number of Best method selected
	Average sorting time (sec)	Average computation time (sec)	Average sorting time (sec)	Average computation time (sec)	
0	8.9259	0.0035	8.5434	6.0678	0
500	8.9259	0.0710	8.5434	6.3459	0
1000	8.9259	0.0750	8.5434	6.2388	0
2000	8.9259	0.0276	8.5434	6.0102	0
4000	8.9259	0.0705	8.5434	6.1723	0
8000	8.9008	0.1129	8.5469	6.1187	4
16000	8.7676	0.5000	8.5658	6.0910	24

에서 수행 되었지만, 최적해를 구하는데는 6.8089 sec~7.0459 sec가 소요되어 계산속도가 빨라 제안된 알고리즘의 유용성을 알 수 있고, 실시간 처리가 가능 함을 알수있다.

[예제 4] 앞 대기장소와 저장소 모두 지정된 등급을 입력한 경우 Table 6에 도시한 바와 같이 분류 시간은 최대 4.48%의 차이를 보이는 반면 경로 탐색시간은 2022.6배에서 최소 12배 단축됨을 알 수 있다. 경로의 수가 많은 경우 탐색에 소요되는 시간은 희생한 분류시간에 비해 계산시간 절감효과가 큼을 알 수 있다.

예제 1과 예제 2에서 알 수 있듯이 경우의 수가 많은 경우와 작은 경우가 무작위로 발생함으로 알고리즘 상에 탐색 제한회수를 제어기의 처리능력에 따라 정해줌으로써 주어진 시간내에 분류시간을 줄여주는데 중요한 요소로 작용하고있어 이의 삽입은 큰 의미를 가진다. 깊이우선 탐색기법에 의한 결과와 비교해 보면 분류시간이 4.480%~9.853% 정도의 차이를 보이는 반면 계산에 소요되는 시간은 현격히 줄일 수 있었다. 본 연구의 분류기 문제와는 직접적인 비교는 곤란하지만 최단거리 문제의 일종인 TSP 문제들을 비교한 문헌[8]에서 알 수 있듯이 방법들 간에 큰 차이를 보여주는 것과는 대조적으로 제안된 알고리즘의 효용성을 알 수 있다.

6. 결 론

본 연구는 인공지능기법을 이용한 실시간 처리 IC 분류 프로그램 개발에 목적을 두었다. 이를 위해 발견적 분류 알고리즘을 제안 하였으며, 제안된 알고리즘의 성능평가를 하기위해 4가지 경우의 예제에 적용하였다. 그 결과 깊이우선 탐색기법에 의한 최적해와 비교하여 본 결과, 분류시간은 최악의 모든 경우에 10% 이내의 차이를 내는 반면 탐색시간은 현격히 줄일 수 있었음을 알 수 있다. 따라서 제안된 알고리즘을 자동 IC 분류기에 적용하면 실시간 처리와 경로의 최적화로 수행성능을 향상시킬 수 있음과 동시에 기구의 최소운동으로 인한 수명연장에도 기여할 수 있으리라 사료된다. 향후 이를 이용한 분류시간을 최소로 하는 분류기의 개수 결정에 관한 연구나 다른 응용분야에 적용하기 위한 연구가 필요할 것이다.

참고문헌

(1) Aarts, E. and Korst, J., 1989, *Simulated Annealing and Boltzman Machines : A Stochastic Approach to Combinatorial Optimization and Neural Computing*, John Wiley & Sons.

(2) Hiroshi Yokoi and Yukinori Kakazu, 1991, "Intelligent Engineering System Through Artificial Neural Net-Works," *ASME Press*, pp. 883~888.

(3) Hopfield, J. J. and Tank, D. W., 1985, "Neural Computation of Decision in Optimization Problem," *Biol. Cybern.*, Vol. 52, pp. 141~152.

(4) Luger and Stubblefield, 1989, *Artificial Intelligence and Design of Expert System*, The Ben-

jamin/Cummings Publishing Company, Inc.

(5) Ivan Bratko, 1986, *Prolog Programming for Artificial Intelligence*, Addison-Wesley publishing company.

(6) Korf, R. E., 1985, "Depth-First Iterative-Deepening : An Optimal Admissible Tree Search," *Artificial Intelligence*, Vol. 27, pp. 97 ~109.

(7) Korf, R. E., 1993, "Linear-Space Best-First Search," *Artificial Intelligence*, Vol. 62, pp. 41 ~78.

(8) Kwon, T. M., 1991, "A Comparative Study of the Traveling Sales Problem," Intelligent Engineering Systems Through Artificial Neural Networks, *ASME Press*, pp. 889~894.