

유전알고리즘을 이용한 로봇 매니플레이터의 최적 시간 경로 계획

김 용호, 강 훈, 전 흥태
중앙대학교 공과대학 전자공학과

Planning a minimum time path for robot manipulator using Genetic Algorithm

* Yong-Hoo Kim, Hoon Kang and Hong-Tae Jeon

Dept. of Electronic Engineering Chung-Ang University, Seoul, Korea

ABSTRACT

In this paper, Micro-Genetic algorithms(μ -GAs) is proposed on a minimum-time path planning for robot manipulator, which is a kind of optimization algorithm. The minimum-time path planning, which can allow the robot system to perform the demanded tasks with a minimum execution time, may be of consequence to improve the productivity. But most of the methods proposed till now suffers from a significant computation burden and can't often find the optimal values. One way to overcome such difficulties is to apply the Micro-Genetic Algorithms, which can allow to find the optimal values, to the minimum-time problem. This paper propose an approach for solving the minimum-time path planning by using Micro-Genetic Algorithms. The effectiveness of the proposed method is demonstrated using the 2 d.o.f plannar Robot manipulator.

I 서 론

산업용 로봇은 다수의 링크들과 조인트들로 구성되어 있으며 공장자동화에서 중추적인 역할을 담당하는 컴퓨터로 제어되는 기계적인 매니플레이터 시스템이다. 이러한 매니플레이터는 프로그램의 변환을 통하여 다양한 작업을 할수있기 때문에 FMS의 중요한 work cell로 인식되고 있다.

일반적으로 산업용 로봇이 수행하는 작업(drilling, welding, painting, assembling, 그리고 운반작업 등)들은 여러개의 subtask(이는 로봇의 소작업을 의미함)들과 그 소작업 사이에서의 로봇의 주행경로(traveling path)들로 구성되어 있으며, 다음 두 종류로 대분될 수 있다.

첫째, 각 소작업들 사이에 수행 순위 (ordor of execution)가 미리 규정되어 있는 작업 (그림 1).

둘째, 수행 순위가 규정되어 있지 않은 소작업들로 구성된 작업.

산업용 로봇이 상기 2부류 작업들을 수행하는데 소요되는 작업 시간은 공히 각 소작업의 수행시간들과 소작업 사이에서의 주행시간들의 합이다. 따라서 이 작업시간들의 최소화(즉, 소작업 수행시간과 주행 시간의 최소화)는 로봇 생산성 극대화에 크게 기여 할 것이다. 그러나 많은 예에서 각 소작업의 수행시간은 일정하다고 간주 할 수 있기 때문에 작업

시간의 최소화는 주행시간의 최소화와 일치 한다고 생각할 수 있다.

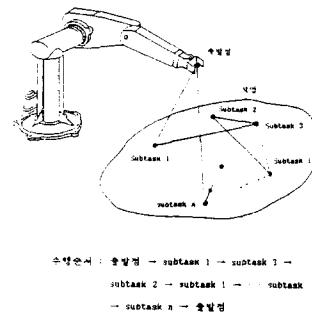


그림 1. 수행 순위가 규정된 n개의 소작업 들로 구성된 작업

작업시간의 최소화(즉, 주행시간의 최소화)는 앞서 분류한 두 종류의 작업에 따라 각기 다르게 진행되어야 한다. 전자의 경우, 즉 작업 순서가 미리 규정된 경우에는 그 주행경로들이 정해져 있기 때문에 기존의 연구 결과([1],[2])들에 의해 해결될 수 있다.

그러나 후자의 경우에는 각 소작업들 사이에서의 주행시간의 최소화 뿐만 아니라 주행경로의 선정(혹은, 각 소작업의 수행순위 결정)이 전체 작업 시간의 최소화에 큰 영향을 미친다. 따라서 후자의 작업인 경우에는 각 소작업 사이에서의 주행시간의 최소화 뿐만 아니라 최소시간 측면에서의 경로선정(selection of path)에 관한 연구가 동시에 이루어 져야한다.

본 논문에서는 각 소작업에서 로봇의 주행경로(즉, 주행 순위)가 정해져 있지 않은 경우, 그 작업 시간의 최소화를 얻기 위한 경로 선정에 관한 방법이 제시 된다. 최적의 주행 경로를 얻기 위해 제안된 방법은 자연계의 진화 이론을 바탕으로 이루어진 것으로, 최근에 최적화 알고리즘으로 각광을 받고 있는 유전알고리즘(Genetic Algorithm)이다. [4][6][7]

본 논문의 구성은 다음과 같다. 2 장에서는 유전 알고리즘을, 3 장에서는 최적의 주행경로를 얻기위한 유전알고리즘의 적용방법이 설명된다. 4장에서는 제안된 방법의 컴퓨터 모의 실험결과가 설명되고 5장에서는 결론 및 향후 연구방향이 논의 된다.

II 유전 알고리즘(Genetic Algorithm)

1970년대초 John Holland에 의해 본격적으로 연구되기 시작한 유전 알고리즘은 자연생태계의 진화과정에서 관찰된 몇 가지 처리과정에서 '적자생존(Survival of the fittest)'의 원리를 이용하여 정립된 최적화(optimization) 알고리즘이다. [4][7]

유전알고리즘의 바탕이 되는 자연 생태계의 일반적인 진화 과정은 다음과 같다. [7]

- 첫째, 진화는 염색체(Chromosome)-생물의 구조를 부호화(encoding) 하기 위한 기관소자(organic device)에서 일어난다.
- 둘째, 자연도태(Natural Selection)는 훌륭한 구조를 부호화한 염색체를 더 자주 재생산 하도록 해준다.
 - 재생산(reproduction)은 진화가 일어난 시점이다.
- 셋째, 돌연변이는 자손의 염색체를 어버이 염색체와 다르게 해준다.
- 넷째, 재조합(Reproduction) - 교배(crossover) - 과정은 어버이의 염색체를 결합하여 매우 다른 염색체를 자손에게 준다.

이 유전알고리즘의 미지수 함수 $Y=G(x)$ 의 최적해를 발견하는 Simulated evolution 형의 탐색(Search) 알고리즘의 성격을 갖추며 John Holland는 상기과정을 어려운 문제 해결의 기법으로써 컴퓨터 알고리즘과 결합시켜 유전알고리즘을 탄생시켰다. 그 과정은 다음과 같이 요약할 수 있다.

- [단계 1] 10진수 인자 1과 0으로 구성된 string형으로 재표현하기 위한 string의 길이와 초기 string을 선정한다.
- [단계 2] 단계 1의 string을 이용해 초기집단(population)을 구성함
- [단계 3] 집단의 각 string을 10진수 인자로 재이용
- [단계 4] 적합도에 따라 해당 string들을 선택함.
- [단계 5] String 간의 교배(crossover)와 돌연변이(mutation)등의 진화과정을 거쳐 새로운 집단을 구성함.
- [단계 6] 단계 3 이하의 과정을 반복함.

그림 2.(a)는 단계 1에서의 인자의 string 표현과 그 역 표현(즉 string 표현으로부터 인자로의 표현)을 보여주고 있으며 단계 3에서의 적합도는 함수에 대한 최적 정도를 의미한다. 그리고 단계 4에서의 string 선택은 적합도의 높은 정도에 의해서 이루어진다. 그림 2.(b)는 8비트인 두 string X와 Y의 교배를 보여주고 있다. 이때 교배점(crossing site)은 임의의 난수로 선택하게 된다. 돌연변이는 교배와 동시 행해지며, 임의의 비트를 선택하여 비트전환으로 이루어진다(단계5). 한 예로 $X = 1001$ 일때 임의로 선택된 비트가 3번째이면 돌연변이를 한 새로운 x' 는 1011이 된다.

		교배점	
인자	string	$X = xxxx\ xxxx \rightarrow X' = xxxxyyyy$	
0.75000	←01100000	$Y = yyyy\ yyyy \rightarrow Y' = yyyyxxxx$	

(a) 인자와 string (b) string X와 Y의 교배

그림 2. 인자의 string 재표현과 그 교배

이와 같은 유전 알고리즘의 특성은 다음과 같다.

- 첫째, 유전 알고리즘은 해공간(solution space)에서 단일 해를 사용하는 것이 아니라 해집단(population of solution space)을 이용한다.
- 둘째, 유전이론은 'blind' 하다. 즉 최적화 함수 정보(미분가능성, 연속성 등)를 필요치 않는다.
- 셋째, 유전 알고리즘은 global 한 최적해의 발견을 가능케 한다.

상기 유전 알고리즘은 일반적으로 대규모의 유전자 집단을 사용하는데 특히 단순 유전 알고리즘(SGA: Simple Genetic Algorithms)에서는 유전자 집단의 크기가 30에서 200개 정도이다. 따라서 유전 알고리즘의 계산을 위한 시간소비는 크다고 할 수 있다. 계산시간은 적은 집단의 선택으로 줄일 수 있는데 이를 위해 제안된 방법이 '미소 유전알고리즘(μ -GA: Micro-Genetic Algorithms)이다.

유전 알고리즘에서 집단수를 작게 하면 수행능력이 감소하는 경향이 있으므로 집단수를 작게하고도 수행능력에는 영향을 미치지 않는 특성을 갖는 μ -GA는 다음과 같은 처리과정을 갖는다.

- [단계 1] 집단의 크기를 5개로 임의로 선택하거나 4개는 임의로 선택하고 1개는 이전의 탐색으로부터 나온 우수한 것으로 선택한다.
- [단계 2] 적합도를 계산한 후 제일 우수한 것을 구한 다음 이 string을 5번째 string으로 두고 다음 세대로 넘긴다. (엘리트 규칙)
- [단계 3] 남은 4개의 string을 재생산하기 위해 임의로 배열하여 짝을 지운다. (이때 같은 string끼리의 짝짓기는 피한다.)
- [단계 4] 교배를 시킨다.
- [단계 5] 수렴성을 검사한 후 수렴하면 [단계 1]로 간다.
- [단계 6] [단계 2]로 간다.

상기 과정에서 돌연변이는 시키지 않는데 이는 [단계 1]에서 4개의 string이 임의로 선택되기 때문이다.

본 논문에서는 μ -GA를 사용하여 로봇 매니플레이터의 최적의 작업경로를 구한다.

μ -GA를 이용한 최적시간경로 결정 수행순위가 규정되어 있지 않은 소작업들로 구성된 작업의 경우 소작업들은 다음과 같은 성질을 갖는다고 가정한다.

- (1) 각 소작업의 수행 우선 순위와 시간은 동일하다.
- (2) 임의의 두 소작업 사이에서의 수행경로는 직선경로이다.
- (3) 모든 소작업들을 수행한 후 매니플레이터는 처음 출발점 소작업으로 귀환한다.

상기 작업을 수행하기 위해 매니플레이터가 선택할 수 있는 주행 경로들은 많이 있지만 최소 시간 측면에서는 그 경로들은 한정되어 있다. 이 한정된 경로, 즉 최적 시간 측면에서의 경로를 결정하기 위해 본 절에서는 앞에서 제안된 μ -GA를 도입한다.

일반적으로 매니플레이터가 주어진 작업을 최소 시간내에 완료 하기 위해서는 모든 소작업들의 수행 시간과 소작업들 사이에서의 전체주행 시간의 최소화가 필요하다. 그러나 각 소작업의 수행 시간과 임의의 두 소작업들 사이에서의 최소주행시간을 규정할 수 있다면, 주어진 최적 시간 경로 설정 문제는 소작업들의 수행 순서(order of execution)를 결정하는 문제로 변환 된다. 더구나 두 소작업 사이에서의 최소 주행

시간을 거리 변수로 치환 한다면 상기 문제는 기존의 TSP 문제로 간주할 수 있으며 문제 해결에 유전 알고리즘의 도입을 가능케 한다.

유전 알고리즘에 의한 문제 해결을 위해 두 소작업 사이에서의 최소 주행 시간 결정 과정이 다음에 설명된다.

III-1 각 소작업 사이에서의 최소 주행 시간 결정

작업 공간내에 위치한 i 번째 소작업($i=1, 2, \dots, n$)의 위치(position)와 방위(orientation)는 다음과 같은 4×4 Homogeneous transformation matrix에 의해 표현될 수 있다.

$$H_i \in \mathbb{R}^{4 \times 4} = \begin{bmatrix} n_i & s_i & a_i & p_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

여기에서 $n (= [n_{x_i}, n_{y_i}, n_{z_i}]^T)$, $s_i (= [s_{x_i}, s_{y_i}, s_{z_i}]^T)$, 그리고 $a_i (= [a_{x_i}, a_{y_i}, a_{z_i}]^T)$ 는 각각 방위를 나타내는 단위 normal, slide, approach 벡터들이며 $p_i (= [p_{x_i}, p_{y_i}, p_{z_i}]^T)$ 는 위치 벡터이다.

i 번째 소작업을 수행하는 동안, end-effector의 카르테시안 주행속도(Cartesian travelling velocity) 성분은 0으로 간주한다. 따라서 i 번째 소작업 수행후 매니플레이터가 일정한 위치 및 방위 속도를 갖고서 직선 경로를 수행하기 위해서는 일정한 가속구간이 필요하며 $i+1$ 번째 소작업을 수행하기 직전에는 직선 주행속도의 감속을 위한 감속 구간이 요구된다. 이러한 가속 및 감속구간을 설정하기 위해 중간 경로점들을 다음과 같이 설정한다. (그림 3)

$$\begin{aligned} H_{ai} &= H_i + k(H_{i+1} - H_i) \\ &= [n_{ai}, s_{ai}, a_{ai}, p_{ai}] \end{aligned} \quad (6)$$

$$\begin{aligned} H_{di} &= H_i + k(H_{i-1} - H_i) \\ &= [n_{di}, s_{di}, a_{ai}, p_{di}] \end{aligned} \quad (7)$$

여기에서 k ($0 \leq k \leq 1$)은 감/가속 구간의 결정을 위한 스칼라 값이며 구간 $H_{di}-H_i$ 는 가속 구간 그리고 구간 $H_{ai}-H_i$ 는 감속 구간이 된다.

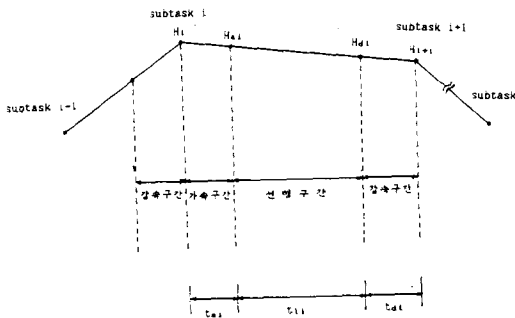


그림 3. 가속 및 감속 구간의 설정

한편 각 구간에서 매니플레이터 end-effector의 주행 시간들을 t_{ai} , t_{li} , 그리고 t_{di} 라 하면 각 구간에서의 end-effector의 카르테시안 선형 속도와 가속도 성분들을 다음과 같이 얻어진다.

$$V_{ai} = \frac{V_{li}}{t_{ai}} t, \quad V_{ai} = \frac{V_{li}}{t_{ai}} \quad (\text{가속 구간}) \quad (8)$$

$$V_{li} = \frac{P_{di} - P_{ai}}{t_{li}}, \quad V_{li} = 0 \quad (\text{선형 구간}) \quad (9)$$

$$V_{di} = -\frac{V_{li}}{t_{di}}(t - t_{ai} - t_{di}), \quad V_{di} = \frac{-V_{li}}{t_{di}} \quad (\text{감속 구간}) \quad (10)$$

여기에서 V_{ai} , V_{li} , V_{di} 는 각 구간에서의 3×1 선형 속도 벡터들이며 V_{ai} , V_{li} 그리고 V_{di} 는 3×1 선형 가속도 벡터이다. 마찬가지로 각 구간에서의 각 속도와 가속도들은 다음과 같이 유도된다.

$$\omega_{ai} = \frac{\omega_{li}}{t_{ai}} t, \quad \omega_{ai} = \frac{\omega_{li}}{t_{ai}} \quad (\text{가속구간}) \quad (11)$$

$$\omega_{li} = \frac{\theta_{li}}{t_{li}} \cdot \rho_{li}, \quad \omega_{li} = 0 \quad (\text{선형구간}) \quad (12)$$

$$\omega_{di} = -\frac{\omega_{li}}{t_{di}}(t - t_{ai} - t_{li}), \quad \omega_{di} = -\frac{\omega_{li}}{t_{di}} \quad (\text{감속구간}) \quad (13)$$

여기에서 ω_{ai} , ω_{li} , ω_{di} 는 각 구간에서의 3×1 각 속도를 나타내며 ω_{ai} , ω_{li} , 그리고 ω_{di} 는 3×1 각 가속도 벡터이다. 또한 각 θ_{li} 와 회전 기준 벡터 $\rho_{li}(\in \mathbb{R}^3)$ 는 다음 식에 의해 정의된다.

$$\theta_{li} = \{[(n_{di})^T n_{ai} + (s_{di})^T s_{ai} + (a_{di})^T a_{ai} - 1]\} \quad (14)$$

$$\rho_{li} = \frac{1}{2\sin\theta_{li}} \begin{bmatrix} (a_{di})^T s_{ai} - (s_{di})^T a_{ai} \\ (n_{di})^T a_{ai} - (a_{di})^T n_{ai} \\ (s_{di})^T n_{ai} - (n_{di})^T s_{ai} \end{bmatrix} \quad (15)$$

한편 매니플레이터는 제한된 구동력을 가지므로 end-effector의 선형속도와 가속도 그리고 각속도와 각가속도는 다음과 같은 제한치를 갖는다고 가정할 수 있다.

$$\|v(t)\| \leq K_v \quad (16a)$$

$$\|a(t)\| \leq K_a \quad (16b)$$

$$\|\omega(t)\| \leq K_w \quad (16c)$$

$$\|\dot{\omega}(t)\| \leq K_{\dot{\omega}} \quad (16d)$$

상기 제한값과 식 (8)부터 (13)을 이용해 i 번째 소작업과 $(i+1)$ 번째 소작업 사이에서의 최소 주행 시간 t_i^* 는 다음과 같이 구할 수 있다.

$$t_{li}^* = \max(\|Pd_i - P_{ai}\| / k_v, \|\theta_i\| / k_w) \quad (17)$$

$$t_{ai}^* = t_{di}^* = (\|P_{li}\| / k_a, \|\omega_{li}\| / k_{\omega a}) \quad (18)$$

$$t_i^* = t_{ai}^* + t_{li}^* + t_{di}^* \quad (19)$$

여기에서 $(\cdot)^*$ 은 최소 시간을 의미하며

$$P_{li}^* = (Pd_i - P_{ai}) / t_{li}^*, \omega_{li}^* = |\theta_{li}| / t_{li}^*$$

이다.

III-2 μ -GA에 의한 최소 시간 경로 결정

앞서 언급 했듯이 식(19)에 의해 결정된 임의의 두 소작업 사이에서의 최소주행시간을 거리변수로 바꾸어 생각하면, 주어진 문제는 기존의 TSP 문제와 일치하게 된다. 따라서 본 절에서는 TSP문제 해결에 적용된 μ -GA을 이용, 주어진 작업공간에서 최적시간경로를 결정 한다.

μ -GA의 적용을 위해서 다음과 같은 선택이 필요하다. 첫째, TSP문제 해결을 위한 string 표현은 다음과 같은 '인접 표현(adjacency representation)' 과 10진수 string을 사용한다.

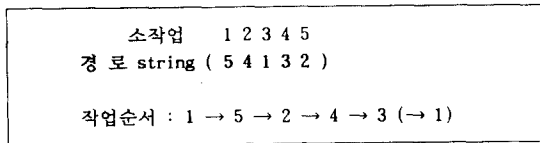


그림 4. 인접표현에의한 string 구성

둘째, 소작업에 순서가 있으므로 순서를 고려한 Greedy 교배를 사용한다.

Greedy 교배의 단계는 다음과 같다.

- [단계 1] 자손의 경로의 시작점으로써 아버지의 시작경로중 임의로 1개를 선택한다.
- [단계 2] 자손의 2번째 경로로써 두 아버지의 두번째 경로중 짧은 경로를 선택한다.
- [단계 3] 자손의 경로를 선택해가는 도중에 이미 선택한 경로만 남아있는 경우에 임의의 경로를 선택한다.
- [단계 4] 자손의 경로를 다 선택할 때까지 위 과정을 반복한다. (교배후 자손의 수는 1개가 된다.)

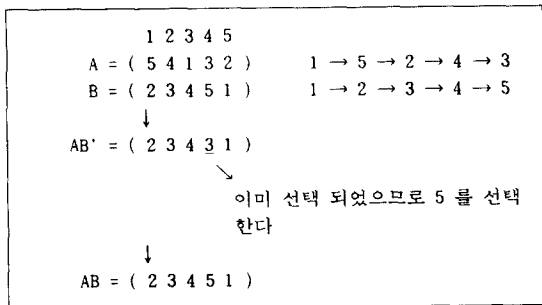


그림 5. Greedy 교배에의한 generation

- 문제해결을 위해 수정 적용된 μ -GA의 단계는 다음과 같다.
- [단계 1] 초기 집단 크기를 임의로 5개를 선택하고 이후에 4개는 임의로 선택하고 1개는 이전의 탐색으로부터 나온 우수한 것으로 선택한다. (이때 string은 10진수로 표현하며 한개의 string내에 중복된 숫자가 나오지 않도록 선택한다.)
- [단계 2] 적합도를 계산한 후 제일 우수한 것을 구한 다음 이 string을 5번째 string으로 두고 다음 세대로 넘긴다. (엘리트 규칙)
- [단계 3] 남은 4개의 string을 재생산하기 위해 임의로 배열하여 8쌍의 짝을 지운다. (이때 같은 string끼리의 짝짓기는 피한다.)
- [단계 4] 교배를 시킨다. (Greedy crossover 사용)
- [단계 5] 수렴성을 검사한 후 수렴하면 [단계 1]로 간다.
- [단계 6] [단계 2]로 간다.

IV 실험 결과

본 논문에서는 2자유도의 평면 로봇 매니퓰레이터를 사용했으며 μ -GA가 최적의 경로를 찾는가를 시험하기 위해 20개의 정방형으로 위치시킨 소작업을 선택하였고 실제적인 작업을 고려하여 임의로 위치시킨 30개의 소작업을 선택하였다. 다음 그림 6은 실험에 사용된 2자유도의 평면 로봇 매니퓰레이터이다.

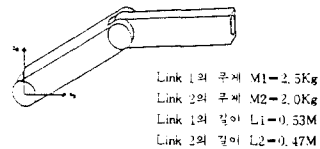


그림 6. 2축 평면 로봇 매니퓰레이터

그림 7은 20개의 정방형 소작업의 경로를 찾기 시작할때의 경로이고 그림 8과 그림 9는 μ -GA가 찾은 최적 경로를 나타내고 있다. 이때 최적의 경로가 1개 이상임을 알 수 있다. 그림 10은 적합도값과 Generation과의 관계를 보여주고 있는데 약 500 Generation에서 최적의 경로를 찾음을 알 수 있다. 500 Generation이라 할지라도 집단수가 5개에 불과하므로 소요된 계산시간은 매우 작다. 그림 11은 임의의 30개 소작업의 경로를 찾기 시작할때의 경로이며 그림 12는 μ -GA가 찾은 최적의 경로를 보여주고 있다. 직관적으로도 최적임을 알 수 있다. 그림 13은 적합도값과 Generation과의 관계를 나타내고 있으며 약 800번 정도에서 최적의 경로를 찾았음을 알 수 있다. 상기 과정에서 적합도값이 최적 주행시간이다.

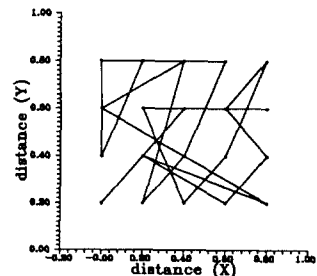


그림 7. 정방형 소작업의 초기 경로

