유전자알고리즘을 이용한 유비쿼터스 컴퓨팅 환경의 프린터 스케쥴링 기법

A Genetic Algorithm-based Method for Printer Scheduling in Ubiquitous Computing Environment

문영학, 이지형
성균관대학교 컴퓨터공학과
Yong-He Wen, Jee-Hyong Lee
Dept of Computer Engineering, SungKyunKwan University
E-mail: heeowen@skku.edu, jhlee@ece.skku.ac.kr

요 약

유비쿼터스 컴퓨팅 환경에서는 각 클라이언트는 자신이 처한 환경에 따라서 여러 가지 장치를 사용할 수 있고, 각 장치는 각기 다른 클라이언트의 요구를 동시에 받을 수 있다. 본 논문에서는 클라이언트들이 보내는 프린트 요청을 받아 사용자에게 적절한 프린터로 출력하는 유비쿼터스 컴퓨팅 환경에서의 프린터 서버를 가정한다. 그러나 프린터 서버는 많은 클라이언트로부터 동시에 프린트 요청을 받을 수 있으므로, 가능한 모든 클라이언트의 요청을 만족시킬 수 있는 스케쥴링을 해야 한다. 이러한 목적을 위한 유전자알고리즘을 이용한 유비쿼터스 환경에서의 실시간 프린터 스케쥴링 방법을 제안한다.

키워드: 유비쿼터스 컴퓨팅(Ubiquitous Computing), 유전자 알고리즘(Genetic Algorithm), 프린터 스케쥴링(Printer Scheduling).

1. 서론

유비쿼터스 컴퓨팅(Ubiquitous Computing) 이란 언제 어디서나 네트워크에 접속할 수 있는, 즉 우리의 일상이 네트워크로 연결되어 있는 상태를 의미한다[1, 2]. 이러한 환경 속에서 각 클라이언트는 자신이 처한 환경에 따라서 여러 가지 장치를 사용할 수 있고, 각 장치는 각기 다른 클라이언트의 요구를 동시에 받을 수 있다. 현재의 컴퓨터 성능이 기가플롭스(giga-flops) 단위로 1초당 1조 번의 부동점 연산을 수행하는 슈퍼컴퓨터의 개발로 시간이 많이 줄일 수 있게 되었다. 그러나 다양한 사용자 요구가 점점 쌓이게 됨에 따라 스케줄링 문제가 중요한 논의되고 있다. 본 논문에서는 유비쿼터스 환경에서 사용자들이 보내는 프린트 요청을 받아 사용자에게 적절한 프린터로 출력하는 유비쿼터스 컴퓨팅 환경에서의 프린터 서버를 가정한다. 이러한 프린터 서버는 많은 클라이언트로부터 동시에 프린트 요청을 받을 수 있으므로, 가능한 모든 클라이언트의 요청을 만족시킬 수 있는 스케쥴링을 해야 한다. 이러한 목적을 위한 유전자알고리즘을 이용한 유비쿼터스 환경에서의 실시간 프린터 스케쥴링 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 스케쥴링 문제를 분석하고, 3장에서 스케쥴링 문제에 대해서 제안한 유전자 알고리즘을 문제에 적용하는 방법을 기술하고, 4장에서는 이 문제에 유전자 알고리즘을 어떻게 적용하는지 설명한다. 5장에서는 실험 과정을 설명하고, 6장에서는 결론을 맺는다.

2. 프린터 스케쥴링 문제

유비쿼터스 환경에서 사용자는 이동하며 자신이 속한 환경에서 존재하는 임의의 여러 프린터를 사용할 수 있다. 따라서 사용자는 사용하고자하는 프린터를 지정하는 것이 아니라 자신이 원하는 프린터의 속성을 프린터 서버에게 보내면 프린터 서버가 각 사용자 요구에 따라 적적한 프린터로 선택한다고 가정 한다. 이러한 환경에서 m명의 사용자는 프린트 요청을 하고, 프린터 서버는 n개의 프린터 $P_1, P_2, P_3 \sim P_n$을 관리한다고 가정하자. 각 사용자는 자신이 원하는 프린터를, 자신과 프리터와의 거리(Distance), 대기 시간(Time), 프린터 출력의 질(Quality)에 대한 선호정도를 기술하면 프린터서버는 이를 보고 적절한 프린터를 선택한다. 프린터 질(Quality)에 대한 선호정도는 1.0 이고 거리(Distance)와 시간(Time)에 대한 선호정도는 0.5 이 라고하자. 그러면 이 사용자에는 프린터의 질(Quality)의 대해서 많이 고려해 주고, 사용자와 프린터의 거리(Distance), 사용자의 대기 시간(Time)에는 비중을 적게 둔다.

그러나 유비쿼터스 환경에서는 동시 인쇄를 요청하는 사람이 여러 명 있을 수 있으므로 각 사용자가 원하는 것을 최적한 만족시키기 위한 스케쥴링 방법이 필요하다. 이런 스케쥴링을 하기 위해서 본 논문에서는 유전자 알고리즘을 이용해서 프린터 스케쥴링 기법을 구현 하자고 한다.

3. 유전자 알고리즘

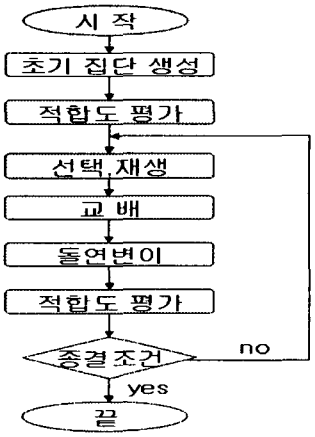

그림 1. 유전알고리즘 과정도

유전자 알고리즘은 자연계의 진화 현상을 구현하는 알고리즘으로, 다윈이 주장한 적자생존과 자연도태의 원리를 이용하여 개발된 알고리즘이다. 유저 알고리즘에서는 생물학적 유전인자인 염색체에 해당하는 문자열을 가지고 생물과 같은 재생산(reproduction), 교배(crossover), 돌연변이(mutation)를 거쳐서 다음 세대의 새로운 자손(offspring)을 만들어 낸다. 자연의 생물유전을 모방한 연산자들을 반복적으로 적용하여 적합한 해를 탐색 한다[3,4]. 유전자 알고리즘은 (그림 1)과 같은 과정들로 구성된다.

4. 유전자 알고리즘을 이용한 스케쥴링

4.1 코딩 방법

각 사용자가 원하는 프리터 출력의 질(Quality), 사용자와 프린터와의 거리(Distance), 출력까지의 대기 시간(Time)을 프린터 서버한테 보내면, 프린터 서버에서는 이 세 가지 값을 고려해서 자동으로 적절한 프린터를 선택하며, 그 프린터의 대기큐에 몇 번째로 들어가야 하는지를 결정한다. 이를 위하여 염색체는 프린터와 그 프린터 내에서 대기 순서를 위해서 두 string으로 구성되었다. 예를 들어 프린터가 3개(P_1, P_2, P_3)있고 프린트 요청을 한 사용자가 10명 있는 경우 한 가지 인코딩 예는 다음과 같다.

	1	2	3	4	5	6	7	8	9	10
printer	P_3	P_2	P_1	P_1	P_3	P_3	P_2	P_3	P_1	P_2
sequence	4	2	5	2	1	3	5	2	5	6

그림 2. 코딩 예

스트링의 인덱스는 사용자의 번호를 의미하며 첫 번째 printer string은 사용자가 요구한 프린팅 작업을 어느 프린터로 보낼 것인지를, 두 번째 sequence string은 그 프린트 작업은 선택한 프린터의 대기큐에서의 우선순위를 나타난다.

예를 들어 첫 번째 사용자를 살펴보자. 첫 번째 사용자 U_1의 printer는 P_3이고 sequence는 4이다. 이 의미는 첫 번째 사용자 U_1이 요구한 프린팅 작업은 P_3프린터의 대기큐에서 4의 우선순위를 갖도록 넣으라는 것이다. 대기큐에서는 작은 우선순위가 먼저 프린트 되도록 하며, 같은 우선순위일 경우 먼저 요청된 것이 먼저 프린트 되도록하였다. 예를 들어 프린터 P_3에 할당된 사용자는 U_1, U_5, U_6, U_8사용자인데, 이들의 sequence 우선순위는 각각 4, 1, 3, 2이다. 따라서 의 P_3 대기큐에는 U_5, U_8, U_6, U_1의 순서로 들어가게 된다. 이것을 모든 프린터에

대해서 스케쥴링을 한 결과는 (표 1)와 같다.

	User(sequence)			
P_1	$U_4(2)$	$U_3(5)$	$U_9(5)$	
P_2	$U_2(2)$	$U_7(5)$	$U_{10}(6)$	
P_3	$U_5(1)$	$U_8(2)$	$U_6(3)$	$U_1(4)$

표 1. (그림 2)의 따른 스케쥴

4.2 적합도 함수(Fitness function)

각 염색체의 적합도(fitness)는 염색체(chromosome)의 속한 각 유전인자(gene)가 표현하는 사용자의 만족도의 합으로 정의된다. 여기서 유전인자가 나타내는 사용자의 만족도는 사용자가 원하는 프린터 출력의 질, 사용자와 프린터와의 거리, 출력까지의 대기 시간과 실제로 사용자에게 할당된 프린터의 질, 프린터와의 거리, 출력까지의 대기 시간을 이용해서 구한다. 여기서 사용자가 원하는 프리터의 거리, 대기 시간, 프린터의 질은 x, y, z로 표현 하고, 실제 거리, 대기 시간, 프린터 질은 α, β, γ로 표현 하자. 그러면 각 유전인자 g와 염색체의 적합도(fitness)의 관계는 다음과 같다.

$$fitness = \sum_{i=1}^{m} g_i \qquad (1)$$

식 (1)에서 사용된 m는 염색체의 길이가 되며 g_i는 아래와 같다.

$$g_i = \alpha_{ij} x_i + \beta_i y_j + \gamma_i z_i \qquad (2)$$

- g_i: i 번째 사용자(유전인자)의 만족도 값.
- a_{ij}: i 번째 사용자와 j 번째 프린터와의 거리(D)를 0과1 사이 값으로 표현한 값.
- β_i: i 번째 사용자 대기 시간(T_i)을 0과1사이 값으로 표현한 값.
- γ_i: i 번째 사용자에게 배치 된 프린터 질.

4.3 선택(Selection)

선택 연산은 염색체에 적용되며 다음 세대의 집단을 만들기 위하여 부모가 될 염색체 쌍들(pairs)을 결정하는 단계로 일반적으로 많이 이용되는 룰렛휠 선택(Roulette wheel selection) 기법을 사용하였다.

4.4 교배(Crossover)

교배 연산은 새로이 만들어진 염색체에 적용되며 교배연산 결과 염색체에 대한 평가가 이루어진다. 이 단계에서는 다음 세대 염색체에 대한 부모들의 유전 형질 전이(Genetic materials transfer)가 일어난다. 이러한 전이는 교배 연산자에 의해 이루어지는데 본 논문에서는 " One-point" 교배 연산자를 선정하여 적용한다. 여기서는 염색체는 (표 1)과 같이 분배된 프린터 번호와 프린터에 분배된 우선순위로 구성 된다. (그림 3)는 교배연산의 예이다.

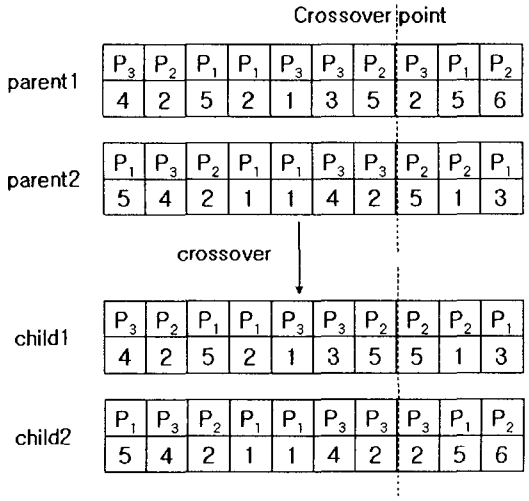

그림 3 교배 연산

4.5 돌연변이(Mutation)

돌연변이 연산은 부모해에 없는 속성을 도입하여 탐색 공간을 넓히려는 목적을 가진 연산이다. 본 논문에서 균등돌연변이(uniform mutation) 방법을 적용하였다. 돌연변이는 printer string과 sequence string에 대해서 모두 적용하였다.

5. 실험 및 결과

본 논문에서는 다음과 같은 상황을 가정하여 실험 하였다. 프린터는 두 대가 있고(P_1, P_2), 사용자는 다섯 명이 있다(C_1, C_2, C_3, C_4, C_5). 각 사용자는 (표 2)과 같이 요청하며 프린터 P_1, P_1가 각 사용자에게 제공하는 거리, 시간, 질은 (표3, 4)와 같다. (표 3,4)에서 t_i는 i번째 사용자가 요청한 프린트 요청을 각 프린터가 처리하는데 걸리는 시간이라 이번실험에서는 모두 1.0으로 동일하게 하였다. 여기에서는 위와 같이 5개의 프린트 요구가 있을 때, 프린트 요구가 생성되는 순서에 따라서 GA가 어떤 성능을 보이는지 실험하였다. 실험결과와 라운드 로빈(Round-

Robin) 방법으로 나온 결과를 비교해서 대표적인 10개 결과들이 (표 5)과 같다. 라운드 로빈(Round Robin)[5] 방식을 이용하는 경우는 사용자의 프린터 요청을 차례대로 각 프린터에 균등하게 분배하는 방식으로 사용자의 새로운 요청을 순차적으로 균등하게 분배를 수행한다.

	U_1	U_2	U_3	U_4	U_5
x	1.0	0.5	1.0	1.0	0.5
y	1.0	0.6	0.1	1.0	0.5
z	0.5	1.0	1.0	1.0	1.0

표 2. 사용자 원하는 선호도

P_1	U_1	U_2	U_3	U_4	U_5
α_{ij}	1.0	1.0	1.0	0.5	0.5
t_i	1.0	1.0	1.0	1.0	1.0
γ_i	0.5	0.5	0.5	0.5	0.5

표 3 P_1이 각 사용자 제공하는 정도

P_2	U_1	U_2	U_3	U_4	U_5
α_{ij}	0.5	0.5	0.5	1.0	1.0
t_i	1.0	1.0	1.0	1.0	1.0
γ_i	1.0	1.0	1.0	1.0	1.0

표 4 P_2와 각 사용자 제공하는 정도

프린트 요청순서	GA	RR
U_2, U_1, U_4, U_3, U_5	8.54	7.74
U_5, U_4, U_3, U_1, U_2	8.17	8.09
U_3, U_1, U_4, U_5, U_2	7.97	7.06
U_3, U_5, U_4, U_2, U_1	8.33	6.01
U_1, U_4, U_3, U_5, U_2	8.65	6.97
U_1, U_5, U_3, U_2, U_4	8.88	5.96
U_1, U_2, U_3, U_4, U_5	8.96	9.01
U_1, U_3, U_2, U_5, U_3	9.83	9.83
U_2, U_4, U_3, U_5, U_1	9.23	6.83
U_4, U_3, U_2, U_5, U_1	8.51	5.89

표 5. GA와 RR 방법 비교

(표 5)에서는 (표 2, 3, 4)이 값을 이용해서 프린트 요청순서로 GA와 RR 값을 비교하여 GA는 RR보다 더 좋은 값을 보여준다. GA는 유전자 알고리즘을 적용한 만족도 값, RR는 라운드 로빈 방법을 적용한 만족도 값이다. 예를 들어 첫 번째 경우 U_2, U_1, U_4, U_3, U_5는 사용자 U_2가 제일 먼저 프린터요청을 한 후 그다음 U_1, U_4, U_3, U_5의 순서로 요청을 한 경우 인데, GA는 전체 만족도 8.54인 U_1, U_3, U_5, U_2, U_4로 스케쥴을 하였지만 RR의 경우는 U_2, U_1, U_4, U_3, U_5로 만족도가 7.54인 성능을 보였다. GA이용한 경우는 프린터 요청을 적절한 순서로 프린터에 배치되고, RR방법을 이용한 경우는 (표 5) 순서대로 배치하였다. 본 실험을 보면 RR방법을 이용해서 최적화한 값을 구할 수 있지만 좋은 값을 유지하기 어렵다. GA를 이용하면 최적화한 값을 구할 수 있다는 것이 보다 평균적으로 좋은 값을 유지할 수 있다.

6. 결론

본 논문에서는 유비쿼터스 환경에서 사용자가 요구하는 프린트 요청을 GA를 이용하여 적절히 배분하여 스케쥴하는 방법을 제안하였다. 제안된 방법은 라운드 로빈(Round Robin) 방법에 비하여 좋은 성능을 보여주었다. 또한 본 논문에서 제안한 프린터 스케쥴링 방법은 많은 사용자를 위한 최적 스케쥴 생성에도 적용할 수 있다. 향후 연구과제로 많은 사용자 경우의 성능 평가와 더 빠른 수렴을 위한 지역최적화 (local optimization)기법 적용 등의 연구가 있다.

참 고 문 헌

[1] M. Weiser.“ The computer for the 21st century, Scientific American,” Vol.265, No.3 pp.94-104, September 1991.

[2] M. Weiser.“ Some computer science issues in ubiquitous computing,” CACM, Vol.36, No.7, pp.74-83, In Special Issue, Computer-Augmented Environments, July 1993.

[3] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Co.,1989.

[4] A. Y. Zomaya, Y. H. Teh, "Observations on Using Genetic Algorithms for Dynamic Load-Balancing," IEEE Transactions on Parallel and Distributed systems, Vol. 12, No.9, Sep. 2001.

[5] D. Mendonça, M. Raghavachari, "Comparing the Efficacy of Ranking Methods for Multiple Round-robin Tournaments," European Journal of Operational Research, Vol.123, pp. 593-605, 2000.