韓國 컴퓨터情報學會 論文誌
第11卷 第4號, 2006. 9.

2006-11-4-1-14

동적 네트워크 환경하의 분산 에이전트를 활용한 병렬 유전자 알고리즘 기법

백 진 욱*, 방 정 원**

Applying Distributed Agents to Parallel Genetic Algorithm on Dynamic Network Environments

Jin-Wook Baek*, Jeon-Won Bang**

요 약

네트워크를 통하여 서로 연결된 컴퓨팅 자원들의 집합을 분산 시스템이라고 정의할 수 있다. 최적화 문제 영역에서 가장 중요한 해결 기법 중에 하나인 병렬 유전자 알고리즘은 분산 시스템을 기반으로 하고 있다. 인터넷과 이동 컴퓨팅과 같은 동적 네트워크 환경 하에서 네트워크의 상태는 가변적으로 변할 수 있어 기존의 병렬 유전자 알고리즘을 분산 시스템에서 최적화 문제를 해결하기 위하여 그대로 사용하기에는 비효율적이다. 본 논문에서는 동적 네트워크 환경 하에서 분산 에이전트를 사용하여 병렬 유전자 알고리즘을 효율적으로 사용할 수 있는 기법을 제시한다.

Abstract

Distributed Systems can be defined as set of computing resources connected by computer network. One of the most significant techniques in optimization problem domains is parallel genetic algorithms, which are based on distributed systems. Since the status of dynamic network environments such as Internet and mobile computing, can be changed continually, it must not be efficient on the dynamic environments to solve an optimization problem using previous parallel genetic algorithms themselves. In this paper, we propose the effective technique, in which the parallel genetic algorithm can be used efficiently on the dynamic network environments.

▸ Keyword : 분산 시스템(Distributed System), 분산 에이전트 시스템(Distributed Agent System), 병렬 유전자 알고리즘(Parallel Genetic Algorithm)

• 제1저자 : 백진욱
• 접수일 : 2006.08.23, 심사일 : 2006.09.01, 심사완료일 : 2006.09.23
* 안산1대학 인터넷상거래과교수 ** 청강문화산업대학 컴퓨터소프트웨어과교수

Ⅰ. 서 론

분산 시스템(Distributed System)은 네트워크(Network)에 의해 서로 연결된 컴퓨팅 자원(Resource)들의 집합이라고 정의할 수 있다. 분산 시스템을 사용하는 장점은 네트워크에 연결된 컴퓨터, 파일, 프로그램 등의 자원들을 공유할 수 있다는 것이다. 따라서 분산 시스템에 존재하는 자원들을 효율적으로 이용하여 최적화 문제와 같은 특별한 문제들을 해결하는 것이 분산 시스템 분야에서 중요한 이슈 중의 하나가 된다.

유전자 알고리즘(Genetic Algorithm)은 생물 진화의 이론을 기본으로 하여 개발되어 주로 최적화 문제에 많이 적용되고 있는데, 그 이유는 많은 문제들이 최적화 문제(Optimization Problem)로 표현될 수 있기 때문이다. 특히, 병렬 처리(Parallel Processing)에 의하여 고속화(High Speed Up)를 이루고 전역해(Global Solution)에 근접한 해를 구하려는 병렬 유전자 알고리즘은 분산 시스템을 기반으로 한 가장 강력한 최적화 문제 해결 기법 중의 하나이다[1].

최근 들어서, 저 비용과 고 효율을 위해서 많은 컴퓨팅 문제들을 인터넷과 이동 컴퓨팅 환경 하에서 해결하려는 노력들이 있어 왔다. 하지만, 인터넷과 이동 컴퓨팅과 같은 동적 네트워크의 경우에 네트워크를 이루는 노드와 링크의 상태가 가변적으로 변할 수 있다. 따라서 기존의 병렬 유전자 알고리즘(Parallel Genetic Algorithm)을 동적 네트워크 환경하의 분산 시스템에서 최적화 문제를 해결하기 위하여 그대로 사용하기에는 매우 비효율적이다.

최근 분산 에이전트 개념은 인터넷과 이동 컴퓨팅 분야에 적합하게 사용할 수 있는 새로운 분산 컴퓨팅 개념으로 자리를 잡고 있다. 분산 에이전트는 동작하는 방법에 따라서 정적 에이전트와 이동 에이전트로 분류할 수 있다. 정적 에이전트는 노드 자원을 제공할 수 있는 기능을 가지고 있는 반면에 이동 에이전트들은 노드들 사이를 이동하면서 작업을 수행 한다[2]. 특히, 이동 에이전트는 인터넷과 이동 컴퓨팅에서 흔히 사용하는 원격 프로시저 호출(Remote Procedure Call)을 기반으로 한 직접 연결(Direct Connection)의 대체 개념으로 중요하게 여겨진다. 네트워크 관리, 이동 무선 통신, 전자 및 이동 상거래, 분산 정보 검색, 그리고 Peer to Peer (P2P) 같은 여러 분산 애플리케이션에 분산 에이전트를 활용하려는 많은 연구가 이루어져왔다. 특히, 이동 에이전트가 가지는 이동 특성은 분산 에이전트 기술이 네트워크 통행량(Network Traffic)을 줄이고, 네트워크 지연 시간(Latency)을 극복하고, 분산 애플리케이션(Application)의 견고성(Robustness)과 결함 내성(Fault Tolerant)을 향상시켜준다[3].

본 논문에서는 인터넷과 이동 컴퓨팅과 같은 동적 네트워크 환경 하에서 분산 에이전트를 사용하여 병렬 유전자 알고리즘의 효율적인 사용기법을 제시한다.

Ⅱ. 관련 연구

이 장에서는 본 연구와 관련된 기존의 연구 내용들에 대해서 소개한다. 먼저, 분산 에이전트에 대해서 설명하고, 다음으로, 유전자 알고리즘과 병렬 유전자 알고리즘에 대한 과거의 연구들에 대해서 간략하게 언급한다.

2.1 분산 에이전트

최근 분산 컴퓨팅 분야에서 분산 에이전트의 연구들이 많이 이루어졌다. 지능성, 이동성 등과 같은 많은 에이전트가 가지는 특성으로 인하여 분산 애플리케이션에 분산 에이전트를 적용하려는 많은 노력들이 있어왔다. 특히, 많은 에이전트 연구자들은 분산 에이전트를 분산 애플리케이션에 적용하는데 있어서의 시스템 환경이 낮은 네트워크 대역폭(Bandwidth)과 자주 끊김 현상이 있는 인터넷과 이동 컴퓨팅 환경에서 매우 효율적이라고 주장하였다[4]. 많은 에이전트 연구 그룹들은 이런 형태의 실행 환경들, 즉 Aglets [5] 그리고 D'agents [6]와 같은 것들을 개발하였다. 분산 에이전트는 최근 수년간 매우 중요한 분산 컴퓨팅 기법의 하나가 되었으며, 그 들은 광범위한 자원 공유를 목적으로 하는 이들 실행 환경의 출현으로 더욱 더 중요성을 가지게 되었다.

2.2 유전자 알고리즘

1975년 홀랜드의 저서에서 소개된 유전자 알고리즘은 생물 진화의 원리로부터 착안된 알고리즘으로 확률적 탐색이나 학습 및 최적화를 위한 한 가지 기법으로서 1985년 미국의 카네기 멜론 대학에서 첫 번째 국제 학회가 열린 후부터 유전자 알고리즘에 대한 연구가 본격화되었다.

유전자 알고리즘은 기본적으로 생성과 시험형의 알고리즘으로 일반적으로 세 가지 종류의 유전자 조직을 사용한다. 즉, 선택(Selection), 교차(Crossover), 돌연변이(Mutation)로, 처리 순서는 먼저 초기 모집단(Population)을 생성하며, 다음으로 종료 조건을 만족할 때까지 적응도를 평가하

면서 선택, 교차, 돌연변이 같은 조작을 반복 수행한다. 여기서 각각의 반복은 세대(Generation)를 의미한다. 이 알고리즘은 유전자 연산자를 사용하여 문제의 해를 구하는 예비해(Chromosome)의 집합을 표현하는 모집단에서부터 수행되어진다[7][8].

유전자 알고리즘은 조합 최적화, 설계 문제, 스케줄링 문제, 그리고 신경 망 설계 등의 많은 문제에 적용하여 풀 수 있으나, 특히 최적화 문제에 많이 적용되고 있다. 그 이유는 많은 문제들이 최적화 문제로 표현될 수 있기 때문이다.

2.3 병렬 유전자 알고리즘

병렬 유전자 알고리즘은 기본적으로 두 가지 사고방식이 있다. 한 가지는 각 처리기(Processor)가 하부 모집단(Sub Population)을 가짐으로써 지역해(Local Solution)가 전 집단으로 퍼지는 것을 피하고, 보다 좋은 해를 얻고자 하는 사고방식이고, 또 다른 한 가지는 병렬 처리에 의해 고속화를 꾀하는 사고방식이다. 그러나 어느 쪽이든 한 가지 방법만으로 고려하는 경우는 거의 없고 실제로는 해의 질을 향상시키고 고속화를 동시에 얻고자 하고 있다[9][10].

해의 질을 향상시키는 것에 관해서는 테네시(Tenese)에 의한 연구가 있다. 테네시는 월시(Walsh) 다항식을 사용하여 병렬 유전자 알고리즘과 종래의 유전자 알고리즘으로 얻어진 해의 질을 비교하였다. 병렬 유전자 알고리즘으로서 복수개의 하위 모집단을 두고 그 사이에 이주(Migration)가 정기적으로 발생하는 모델과 단일 집단을 비교하였다. 그 결과, 종래의 단일 집단에 의한 유전자 알고리즘에 비하여 병렬 유전자 알고리즘을 통해서 보다 질 높은 해를 얻을 수 있음을 알 수 있었다[11].

이것은 분할된 모델에서는 지역해가 전체로 퍼지는 것이 제한되어, 각각의 하위 모집단으로 다른 지역해를 얻는 것이 가능하게 되었기 때문이다. 일반적으로 작은 모집단에서는 초기에 지역해로 빠르게 수렴하기 때문에 나중에 적응 값(Fitness Value)이 향상되지 않는다. 그러나 병렬 유전자 알고리즘처럼 일정한 간격으로 이주를 수행함에 의해서 각 집단의 진화가 정체되는 것을 방지할 수 있을 것이다.

유전자 알고리즘은 많은 방식으로 병렬성을 도입할 수 있다. 실제로, Holland가 제시한 유전자 알고리즘으로부터 매우 다양한 병렬 유전자 알고리즘이 만들어져 왔다. 병렬 유전자 알고리즘의 기본적인 적용은 모집단을 분리한 하부 집단들에서 각각의 기존의 유전자 알고리즘을 수행하고, 다음으로 해를 검색하는 것을 도와주기 위해 각 하부 모집단 사이에서 정보의 주기적 교환을 허용해 준다. 하부 모집단 사이에서 교환되는 정보는 각각 하부 모집단의 적응 개체(Fitness Entity)들의 부분 집합이며 개체의 교환을 이주라고 부른다. 이러한 모델은 몇 가지 방식으로 구현되는 데 하나는 병렬 유전자 알고리즘을 항상 분산 기억장치 MIMD (Multi-Instruction Multi-Data) 기계에서 구현하는 것이다. 다른 하나는 모집단 유전자 항목을 사용하는 것으로 Island 모델이라 부르며 개체의 다소 주기적 교환을 제외하고는 상대적으로 고립된 하부 모집단을 가지고 있다[12].

III. 분산 에이전트 시스템 구조

본 장은 병렬 유전자 알고리즘의 수행을 지원하기 위한 분산 에이전트의 구조를 개략적으로 기술한다. 모든 노드들은 정적 및 이동 에이전트를 수행할 수 있다. 정적 에이전트는 이동 에이전트를 지원하는 기능을 가지며, 이동 에이전트는 실행, 복제(Cloning), 의사소통(Communication), 이동(Migration) 등을 할 수 있으며, 노드들 사이를 이동하면서 작업을 수행한다. 분산 에이전트를 지원하는 전체 시스템의 구조는 그림 1과 같다.

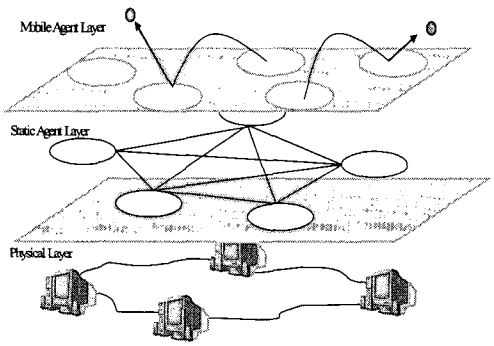

그림 1. 시스템 구조
Fig 1. System Architecture

다음은 각 계층에 대한 설명이다. 이동 에이전트 계층에서 이동 에이전트가 실행된다. 정적 에이전트 계층은 컴퓨팅 또는 데이터 자원을 이동 에이전트에게 제공한다. 또한, 이 계층은 하부의 물리적 계층을 도움을 받아서 다양한 서비스를 제공한다. 예를 들어, 네트워크 상태를 감시하거나 고급 목록 서비스(Advanced Directory Service) 등을 수행한다. 물리적 계층은 실제 컴퓨터 노드들이 연결되어 물리적으로 통신 등을 담당한다.

IV. 동적 병렬 유전자 알고리즘 방법

4.1 병렬 유전자 알고리즘 개요

그림2는 병렬 유전자 알고리즘의 방법을 보여주고 있다. 그림에서 노드는 컴퓨팅 자원을 말하며 노드 사이의 링크를 통하여 통신을 하여 예비해들이 전이할 수 있도록 한다. 대부분 각 노드에서 수행되는 알고리즘은 UNIX 프로세스에 의해 관리되며 하부 모집단에서의 통신은 Berkeley Socket에 의해서 수행되어 질 수 있도록 구현된다. 노드에 분배되는 집단은 하부 개체 집단일 수도 있고 하부 해 집단일 수도 있다.

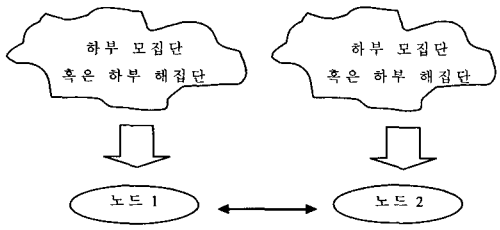

그림 2. 병렬 유전자 알고리즘
Fig 2. Parallel Genetic Algorithm

4.2 인코딩(Encoding)

인코딩이란 유전자 알고리즘이 이해하도록 문제를 표현하는 것으로, 이진 인코딩(Binary Encoding) 방법을 사용한다. 그림 3은 한 예를 보여준다.

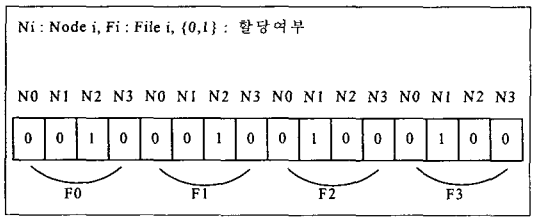

그림 3. 예비해의 인코딩 구조
Fig 3. Encoding Structure of Chromosome

4.3 초기 모집단 생성

모집단이란 유전자 알고리즘이 동작하기 위해서 초기에 구성하고 있는 시스템 인수들의 인코딩된 예비해들을 일컫는다. 실시간 시스템 등의 특별한 환경을 제외하면 대체로 모집단은 30에서 100개 정도로 구성된다[13].

예비해들을 이진 인코딩 방법으로 표현하며 모집단에서 서로 인접한 해들 사이의 해밍 거리(Hamming Distance)가 클수록 최적의 전역해를 찾는 것이 효율적이지 않다고 알려져 있다. 본 연구에서는 병렬 알고리즘 사용하기 위하여 모집단을 몇 개의 하부 모집단으로 나누거나 하부 해집단으로 나눈다. 이것은 컴퓨터 노드들에게 할당하여 병렬로 수행시켜 최적의 전역해를 빠르게 효율적으로 찾고자 함이다. 그리고 인접한 해들의 간의 해밍 거리를 상수로 만들기 위해 회색 인코딩(Gray Encoding)과 같은 방법을 사용할 수 있다. 그림4는 해밍 거리를 상수로 만든 모집단 한 예이다. 이 모집단 내의 인접한 각각의 해들 간의 해밍 거리는 2가된다.

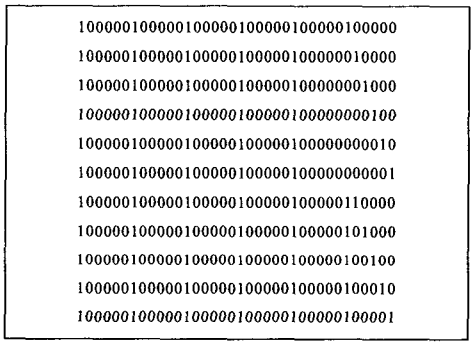

그림 4. 모집단
Fig 4. A Population

마지막으로 모집단에 포함되는 해들은 여러 가지 제약 조건을 두어서 문제 공간을 줄일 수 있다. 이렇게 하는 이유는 모집단에 포함되는 가능한 모든 예비해들 중에서는 실제 분산 시스템에서 의미 없는 할당이 있을 수 있다. 그래서 제약 조건을 둠으로써 그러한 해를 근본적으로 제거함으로써 문제 공간을 줄일 수 있다.

4.4 선택

선택은 현재의 세대에서 다음 세대의 구조를 선택하는 절차이다. 선택 절차에서 가장 많이 사용하는 것은 룰렛 바퀴(Roulette Wheel) 방법이다.

이 방법은 각 예비해의 적응도에 비례한 확률로 자손을 남길 가능성이 있는 모델이다. 이러한 확률적인 선택은 개체의 수가 충분히 많지 않은 경우에 난수의 발생 형태에 의해서 적응도를 정확히 반영하지 않는 선택이 이루어질 가능성이 있는 문제점이 있다.

적응도에 근거한 선택 방법은 각 개체가 적응도 별로 잘 분포 화되어 있지 않을 경우 낮은 적응도를 가진 개체가 선택될 확률이 아주 희박해 지는 단점이 있다. 그래서 이 문제를 해결하기 위해 순위 전략(Rank Based) 방법을 사용할 수 있는 데, 이것은 적응도에 따라서 각 개체의 순위를 매기고 사전에 각 순위에 대해서 결정된 확률로 자손을 남기는 모델이다. 하지만, 순위 전략의 경우에도 확률적 선택에 비해서 눈에 뛰게 개선되지는 않는다고 알려져 있다[14].
또한 낮은 순위의 개체도 선택될 확률을 좀 더 향상시킨 적응도 공유(Fitness Sharing) 방법을 사용할 수 있는데, 이 방법은 모집단에서 비슷한 유형의 개체들에게 벌점(Penalty)을 부과하여 적응도를 제어한다.

4.5 교차

교차는 두 부모의 염색체의 일정 부분을 바꾸어 자식의 염색체를 만드는 조작이다. 가장 단순한 방법은 교차하는 위치를 하나 결정하고 그 앞과 뒤에서 어느 쪽 부모의 유전자형을 받을 것인가를 결정하는 방법이다. 이것을 단순 교차(Simple Crossover), 또는 1점 교차(One-Point Crossover)라고 부른다.
다음으로, 교차 위치가 복수인 방법을 복수 점 교차(Multi-Point Crossover)라고 하며 그림5에서는 파일 단위의 2점 교차 방법의 한 예를 보여주고 있다. 일정 교차(Uniform Crossover) 방법은 교차 시 마스크(Mask)를 사용하여 그것에 의해 어느 쪽 부모의 유전자를 받아들일 것인지를 결정하는 방법이다. 이와 같이 많은 교차 기법이 제안되어 왔는데 특정한 문제 영역마다 다른 교차 방법들을 적용하고 있다. 예를 들면, 순환 방문 판매원 문제 등에 대해서는 부분 일치 교체(Partially Matched Crossover: PMX) 등이 제안되었다.

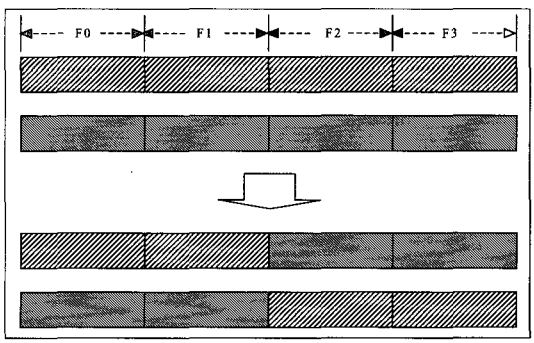

그림 5. 2점 교차
Fig 5. Two-Point Crossover

4.6 돌연 변이

돌연 변이는 유전자를 일정한 확률로 변화시키는 조작이다. 돌연변이를 적용하는 이유는 더 좋은 해가 존재함에도 초기 유전자의 조합 이외의 공간을 탐색할 수 없어 너무 빨리 지역해(Local Solution)에 근접해버리는 Genetic Drift라 불리는 현상을 막기 위함이다. 하지만, 돌연 변이를 너무 큰 확률로 변이 시키면 임의 탐색(Random Search)로 변하게 된다. 따라서 전형적인 돌연변이 조작은 0.0333, 0.015, 0.01, 0.001 등의 아주 미약한 확률을 적용한다. 일반적으로 유전자의 다양성을 확보하기 위하여 고정된 확률이 아닌 동적으로 확률을 조정하는 방법들도 있다.

4.7 할당 및 동적 네트워크 환경 적응

초기 하부 모집단의 집합 또는 하부 해집단의 집합을 적절한 수의 컴퓨터 노드들에게 할당을 한다. 할당된 하부 집단들은 전역해를 도출하기 위하여 선택, 교차, 그리고 돌연 변이 연산을 반복적으로 수행한다. 이때 홈 노드는 모 집단을 쪼개어 하부 모집단을 만들어 이동 에이전트에 실어서 보내는 노드라고 한다. 각 노드들은 다음과 같은 수행을 통하여 동적 네트워크에 적응을 한다.

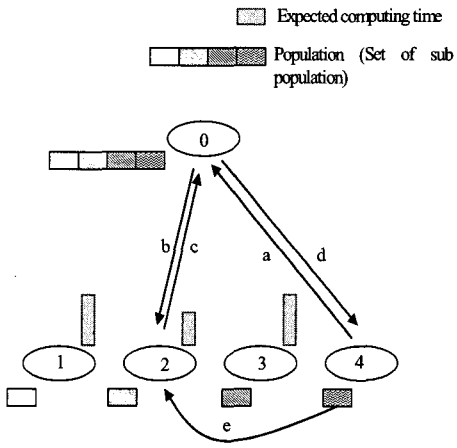

그림 6. 동적 적응 (노드)
Fig 6. Dynamic Adaptation (Node)

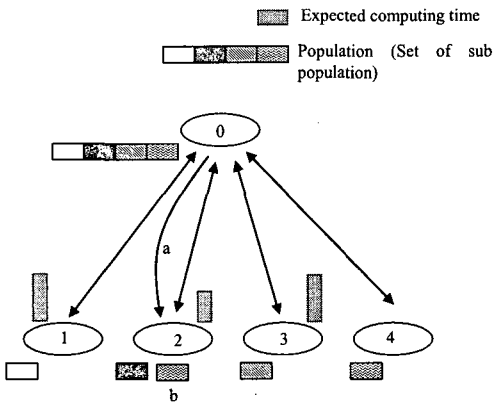

그림 7. 동적 적응 (홈 노드)
Fig 7. Dynamic Adaptation (Home Node)

- 노드가 현재 그 상태가 더 이상 실행을 하기 힘든 상황이 되면 그 노드는 홈 노드에게 보고를 하게 된다(그림 6a). 홈 노드는 가장 적합한 대체 노드를 선택하여 알려준다(그림 6b, 6c, 6d). 그 노드는 대체 노드로 하부 모집단을 보내주는 것이 좋을지 아니면 홈 노드가 보내주는 것이 좋을지 결정하여 자신이 보내는 것이 좋을 것으로 판단되면 그 하부 모집단을 보낸다(그림 6e).
- 홈 노드는 주기적으로 각 노드를 검사하게 된다. 노드가 죽었다는 것을 알게 되면 그 노드가 가진 하부 모집단을 다른 노드(그림 7a)로 보낸다. 그 노드는 받은 하부 모집단을 수행하게 된다(그림 7b).
- 다음 순서의 노드가 홈 노드의 역할을 하면서 각 노드의 해들을 다음 홈 노드에게 전달한다. 홈 노드가 다시 살아나면 현재 홈 노드 역할을 한 노드에서 저장된 전역해를 전달 받는다.

V. 토의 및 구현

기존의 병렬 유전자 알고리즘들은 유전자 알고리즘들이 기본적으로 가지는 병렬성을 이용하여 방대한 해의 영역을 가지는 문제에서 조금이라도 더 빠른 탐색을 하기 위하여 개발되어 사용되고 있다. 하지만, 분산 환경에서 기인한 통신 시간(Communication Time)에 의한 부담(Overhead)은 인터넷과 이동 컴퓨팅과 같은 동적 네트워크 환경에서는 병렬 유전자 알고리즘의 완료 시간(Turnaround Time)을 보장하는데 어려움을 준다. 최악의 경우, 분산 환경을 구성하는 노드들의 정지 및 오류 상태로 인하여 병렬 유전자 알고리즘의 수행은 정상적인 동작을 못하게 될 수 있다. 따라서 본 논문에서 제시하고 있는 분산 에이전트를 활용한 동적 병렬 유전자 알고리즘은 동적 네트워크 환경 하의 분산 시스템에 효율적으로 적용할 수 있다.

기존에 존재하는 대부분의 병렬 유전자 알고리즘들은 대체로 모집단을 나누거나 혹은 인접한 해들로 구성되는 하부 모집단을 나누는 방식을 사용한다. 본 논문에서는 특별한 병렬 유전자 알고리즘은 전제하지 않고 네트워크 적응도에 초점을 맞추고 있기 때문에 본 논문에서 제안한 방법은 사용하는 병렬 유전자 알고리즘의 성능을 높임으로서 전체적인 성능을 높일 수 있는 장점을 가지고 있다.

본 연구에서는 실용성을 위한 목적으로 동적 네트워크 적응성을 위하여 분산 애플리케이션의 테스트베드(Testbed)로 널리 사용되는 플래닛랩(PlanetLab)을 사용하여 소켓(Socket) 기반의 C 프로그래밍으로 동적 네트워크 적응을 할 수 있는 에이전트 플랫폼으로서 구현하였다.

VI. 결론 및 향후 과제

병렬 유전자 알고리즘은 근래에 병렬 분산 처리가 일반화되는 추세에 비추어 상당히 의미 있는 접근 방법으로서 최적해를 구함에 있어 성능 향상을 기할 수 있는 방법이다. 하지만, 인터넷과 이동 컴퓨팅과 같은 동적 네트워크 하에서 그대로 병렬 유전자 알고리즘을 적용하여 최적화 문제를 해결하는 것은 비효율적이며 실현 불가능할 수도 있다. 따라서 본 논문에서는 이에 대한 해결책을 제시하고 있으며, 이를 통하여 효율적으로 분산 시스템의 자원을 사용하여 병렬 유전자 알고리즘을 활용할 수 있다. 향후 과제로서는 최적화 문제만이 아니라 다양한 분산 애플리케이션에서 적용할 수 있는 모델을 개발하는 것이다.

참고문헌

[1] Y. Svirezhev and V. Passekov, "Fundamentals of Mathematical Evolutionary Genetics," Kluwer Academic Publishers, Mathematics and Its Application, vol. 22, 1989.

[2] William R. Cockayne and Michael Zyda, "Mobile agents," Manning Publications Co., 1998.

[3] Alfonso Fuggetta, Gian Pietro Picco, and Giovanni Vigna, "Understanding code mobility," IEEE Transaction on Software Engineering, vol. 24, no. 5, pp. 342-361, January 1998.
[4] Giacomo Cabri, Letizia Leonardi, and Franco Zambonelli, "Agents for information retrieval: issues of mobility and coordination," Journal of Systems Architecture, Elsevier North-Holland, Inc., vol. 46, no. 15, pp. 1419-1433, June 2000.
[5] Gunter Karjoth, Danny B. Lange, and Mitsuru Oshima, "A security model for aglets," IEEE Internet Computing, vol. 1, no. 4, pp. 68-77, August 1997.
[6] Jonathan Bredin, Rajiv T. Maheswaran, Cagri Imer, Tamer Basar, David Kotz, and Daniela Rus, "Computational markets to regulate mobile-agent systems, Autonomous Agents and Multi-Agent Systems," vol. 6, no. 3, pp. 235-263, May 2003.
[7] Holland, John H., "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence," MIT Press, 2nd edition, 1992.
[8] Goldberg, David E., "Genetic Algorithms in Search," Optimization & Machine Learning, Addison Wesley, 1989.
[9] T. Starkweather, D. Whitley, and K. Mathias, "Optimization using distributed genetic algorithm," Parallel Problem Solving from Nature, Springer Verlag, 1991.
[10] L. Tan, D. Taniar and K. A. Smith, "A New Parallel Genetic Algorithm," Proceedings of International Symposium on Parallel Architectures, Algorithms and Networks, pp. 284-289, 2002.
[11] M. S. Ko, T. W. Kang and C. S. Hwang, "Adaptive Crossover Operator based on Locality and Convergence," IEEE International Joint Symposia on Intelligence and Systems, pp. 18-22, 1996.
[12] Erick Cantu-Paz, "Distributed GENESIS Users Guide Version 1.0," ITAM, 1994.
[13] Karr, C. L. "Design of an adaptive fuzzy logic controller using a genetic algorithm," Proceedings of ICGA 4, pp. 450-457, 1991.
[14] John J. Grefenstette, "A Users Guide to GENESIS Version 5.0," 1990.

저 자 소 개

백 진 욱

2006년 2월 : 서울대학교 컴퓨터 공학박사
1998년 ~ 현재 : 안산1대학 인터넷상거래과교수
관심분야 : 분산에이전트, 분산 컴퓨팅

방 정 원

1996년 2월 : 한국과학기술원 정보 및 통신공학과 전산학석사
1998년 ~ 현재 : 청강문화산업대학 컴퓨터소프트웨어과교수
관심분야 : 소프트웨어공학, 분산 컴퓨팅