다측면 유전자 알고리즘을 이용한 시뮬레이션 최적화 기법

A Simulation Optimization Method Using the Multiple Aspects-based Genetic Algorithms

박성진* · 백두권*
Seong-Jin Park · Doo-Kwon Baik

Abstract

For many optimization problems where some of the system components are stochastic, the objective functions cannot be represented analytically. Therefore, modeling by computer simulation is one of the most effective means of studying such complex systems.

Many, if not most, simulation optimization problems have multiple aspects. Historically, multiple aspects have been combined ad hoc to form a scalar objective function, usually through a linear combination(weighted sum) of the multiple attributes, or by turning objectives into constraints. The genetic algorithm(GA), however, is readily modified to deal with multiple aspects. In this paper we propose a MAGA(Multiple Aspects-based Genetic Algorithm) as an algorithm for finding the Pareto optimal set. We demonstrate its ability to find and maintain a diverse "Pareto optimal population" on two problems.

1. 서 론

일반적으로 최적화 문제란 시스템을 구성, 운영, 이용하여 주어진 제약식에 따라서 원하는 목적함수를 최대 또는 최소화함으로써 최적의 해를 구하는 문제이다. 이러한 최적화 문제에 대한 해를 구하기 위해서는 시스템 자체에 대한 충분한 이해를 바탕으로 문제 자체를 수학적인 식으로 먼저 표현되어야한다. 그러나, 많은 최적화 문제들은 시스템 요소가 복잡하고 확률적 특성을 내재함으로써 정확한 분석적 표현이 어렵다는 문제점이 있다. 이에 대한 해결책으로 최근 컴퓨터 시뮬레이션에 의한 모델링이 복잡한 시스템에 대한 최적화 연구의 효과적인 수단으로 고려되고 있다[11, 12, 13].

그 결과 시뮬레이션을 이용한 다양한 최적화 기법이 제시되었으며 생태계의 적자생존에 의한 진화 과정을 모델링한 유전자 알고리즘(GA)도 이러한 기법중의 하나로 많은 연구가 수행되었다. GA는 다양한 문제에 적용 가능하면서도 전역적인 최적해를 찾는 효과적인 최적화 알고리즘으로 최근 많은 분야에 적용되고 있다[3]. 그러나, 기존의 유전자 알고리즘은 주어진 문제에

* 고려대학교 전산학과 소프트웨어시스템연구실

대해 단일 측면만을 고려하여 특정 해에 수렴함으로써 다양한 측면을 고려해야하는 최적화 문제에는 그대로 적용할 수 없다는 문제점이 있다. 특히, 시뮬레이션 최적화를 해야하는 실세계의 많은 문제들이 다양한 측면에서 다수의 목적이 서로 상충하는 복잡한 시스템이라는 점을 고려하면 다측면을 고려한 시뮬레이션 최적화 방안으로서의 새로운 유전자 알고리즘이 요구된다.

본 논문에서는 단순 유전자 알고리즘(SGA;Simple Genetic Algorithm)을 파레토-박스-선택 방식과 파레토 대치 구조를 가지고 확장함으로써 다측면에 대한 파레토 최적해를 생성하는 MAGA(Multiple Aspects Genetic Algorithm)를 제안하였다.

본 논문에서 제시하는 다측면을 고려한 유전자 알고리즘은 다양한 측면의 트레이드-오프(trade-off)에 따른 파레토 경계를 따라 모든 해들에 대한 대표적인 해들을 찾는 것을 목적으로 한다.

2. 기존 연구

2.1 시뮬레이션 최적화 기법

일반적으로 시뮬레이션 최적화에서 다루는 문제들은 목적 함수나 제약식 혹은 두가지 다 컴퓨터 시뮬레이션에 의해 평가될 수 밖에 없는 복잡한 시스템의 반응을 다루게 됨으로써 최적해를 찾는 절차가 복잡하여 많은 노력이 요구된다. 따라서, 복잡한 시스템의 최적화 도구로서 시뮬레이션을 사용하기 위해서는 적절한 최적화 기법이 필요하다. 특히, 다중 측면을 고려한 최적화 문제에 컴퓨터 시뮬레이션을 이용할 경우 다양한 측면들을 적절히 고려함으로써 각 측면간의 트레이드-오프를 고려할 수 있는 효율적인 최적화 알고리즘이 절대적으로 요구된다.

단일 측면을 고려하는 경우에 적용할 수 있는 최적화 기법으로는 Gradient Based Search Method, Stochastic Approximation Method, Response Surface Method 그리고 Heuristic Search Method 등이 있다[1, 5, 10]. 이중에서 Heuristic Search Method에는 Complex Search기법이나 Simulated Annealing, Genetic Algorithm 등이 포함된다. 이러한 최적화 기법을 다측면을 고려해야하는 최적화 문제에 적용하기 위해서는 기존 기법의 확장 및 보완이 필요하며 경우에 따라서는 적용이 불가능한 기법들도 있다.

그 중에서 유전자 알고리즘(GA)은 다양한 문제에 적용 가능한 전역 최적화 알고리즘으로서 그 수행 과정 자체가 병렬적으로 수행되며 여러 개의 복수 해들에 대한 연산을 반복 수행한다는 점에서 다수의 해를 필요로 하는 다중 최적화 문제에 적용이 용이하다는 이점이 있다[2, 3].

본 논문에서는 이러한 유전자 알고리즘의 특성을 이용하여 MAGA를 이용한 시뮬레이션 최적화 기법을 제안하였다.

2.2 다측면을 고려한 유전자 알고리즘

유전자 알고리즘은 생물학적 진화 과정을 모델링함으로써 전역적인 공간 해의 탐색에 유용한 최적화 알고리즘으로 초기에 임의로 선정한 초기 개체들로부터 정보를 수집하여 그 다음 탐색을 어떠한 방식으로 진행시켜 나가야할 지를 결정한다. 이러한 정보 수집은 다수의 개체들을 통해서 전역적인 공간에 대해 수행되며 따라서, 새로운 개체의 결정은 전역적인 탐색에 기반을 두게 된다.

유전자 알고리즘을 최적화 기법으로 이용한 연구들은 대부분 단일-측면(single aspect) 문제들에 주로 적용되어왔다. 그러나, 많은 실세계의 GA 적용 문제들은 그 목적함수가 실제로는 다중-측면(multi aspect)을 갖는 경우가 많으며 시뮬레이션 최적화 문제도 마찬가지이다.

일반적으로 다중 측면을 고려하기 위해 기존의 최적화 기법을 사용하는 경우 취할 수 있는 접근 방법은 스칼라 적합도 함수를 생성하기 위해 임의의 특별 함수(ad-hoc function)를 사용하는 방법, 또는 제한치(threshold)와 벌칙 함수(penalty function)를 이용한 제한조건으로 고려하는 방법, 그리고 평가한 측면 값들의 선형 결합에 가중치를 주는 방법 등이 가능하다[9]. 그러나, 벌칙 함수를 사용하거나 가중치를 주는 방법은 많은 문제가 있는 것으로 지적되었다. 특히, 유전자 알고리즘을 사용하는 경우, GA의 최종해는 벌칙 함수의 계수 또는 가중치 요소의 작은 변화에도 매우 민감하기 때문이다[7].

이에 따라 다측면 최적화를 위해 GA를 이용한 연구들이 제시되었으며 이러한 GA를 이용한 연구 결과들은 서로 상충하는 다중 측면을 고려했을 때 발생할 수 있는 모든 가능한 트레이드-오프를 찾아내는 것을 목적으로 하고 있다. 이러한 경우 최적화 결과로써 얻어지는 해들은 모든 측면에서 보다 우수한 다른 해들이 존재하지 않는다는 점에서 비열성(nondominated)이라고 할 수 있다. 한편, 상충하는 각 측면간의 트레이드-오프 경계를 파레토 최적 경계(pareto optimal frontier)라 하며 각 해들을 해 공간에 표현할 경우 비열성인 해들의 집합은 이 파레토 경계를 형성한다[6, 7].

다측면 최적화에 GA를 적용한 첫번째 연구로는 VEGA(Vector Evaluated Genetic Algorithm)가 있다[6]. VEGA는 다중 측면에 대한 다수의 해를 찾기 위한 연구로서 비용이나 신뢰성 같은 여러 측면중 각 측면에서의 우성인 해들을 다음 세대의 일부로 선택하는 방법을 제시하였다. 이 방법은 초기 단계에서 일부 성공적인 결과가 제시되기는 했지만 세대 교체가 진행될수록 특정 측면에서만 최적이 되는 해가 증가함으로써 파레토 경계의 극점에 수렴하는 경향이 있다. 이와 같은 문제점은 측면들간의 트레이드-오프에 따라 다음 세대를 선택하지 않기 때문에 발생한다.

한편, Horn은 다중 측면 문제에 있어서 각 후보해들을 파레토 경계 쪽으로 이동시키기 위해서 비우성 순서화 및 선택(nondomination ranking & selection) 기법을 제안하였다[7]. Horn이 제시한 Niched Pareto GA에서는 GA가 파레토 경계의 한 점으로 수렴하는 것을 방지하기 위해 유사한 해들이 많은 후보해에 대한 선택 확률을 감소시킴으로써 후보해의 다양성을 유지하는 니이칭(niching) 기법을 사용하였다. 니이칭 기법은 각 개체들을 파레토 경계를 따라 일정한 간격으로 유지시키도록 한다. 그러나, 이 기법은 세대 교체가 이루어질 때마다 후보해간의 니이칭을 평가함으로써 너무 많은 연산을 요구한다는 단점이 있다.

3. GA를 이용한 다측면 최적화 기법

3.1 파레토 최적해

일반적으로 다중 측면을 고려하는 경우 최적(optimality)의 개념을 정의하는 것은 상당히 어렵다. 특히, 다양한 측면에 의한 상대적인 값들 사이의 관련성을 정의하기가 어려운 경우, 새로운 형태의 최적 개념이 필요하며 이 경우 19세기 프랑스 수학자 파레토가 제안한 파레토 최적(pareto optimality)의 개념은 아주 적절하다[3, 7]. 파레토는 동일한 크기의 두 벡터들에 대한 equality, less-than, greater-than, partially-less-than 관계 등을 제안하였는데 4번째 'partially-less-than' 관계는 다음과 같이 정의되며 파레토 최적해를 정의하기에 유용한 개념이다.

벡터 x, y가 다음과 같이 주어졌을 때

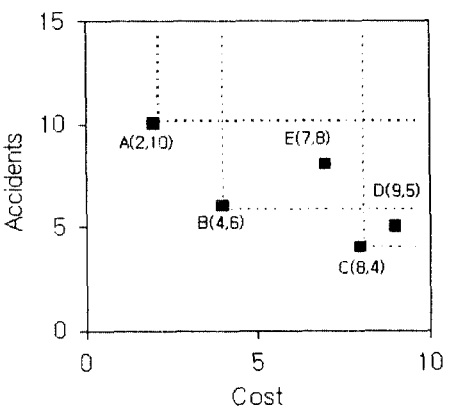
$x = (x_1, x_2, \ldots, x_n)$

$y = (y_1, y_2, \ldots, y_n)$

이진관계 x is partially-less-than y

$\Leftrightarrow \forall i : x_i \leq y_i, \ \exists i : x_i < y_i$

이때, 만약, 목적 함수가 최소화 함수이고 x가 y보다 부분적으로 작다면 "y는 x에 의해 지배된다(dominated)" 혹은 "y는 x에 비해 열성(inferior)이다"라고 부르며 이때 파레토-최적 집합(P-optimal set)은 어떤 다른 벡터에 의해서도 지배되지 않는 비열성인 벡터들을 가리킨다.



〈그림 1〉 파레토 최적의 예

예를 들어, 비용(cost)과 사고발생수(accidents) 두 측면에서 최소화를 목표로 하는 경우, 〈그림 1〉과 같은 5개의 제품 생산 시나리오가 주어진다면 P(파레토) 최

적해 집합은 {A, B, C}로 결정된다.

왜냐하면, 〈그림 1〉에서 시나리오 E의 경우 시나리오 B에 비해서 두 측면 모두에서 열성이며, 시나리오 D 또한 시나리오 C에 대해 두 측면 모두에서 열성이기 때문에 P 최적해 집합에서 제외된다. P 최적해 집합에 속한 시나리오 A, B, C는 서로간에 두 측면에서 트레이드-오프가 발생하며 다른 어떠한 시나리오에 대해서도 모든 측면에서 열성은 아니다. 즉, A, B, C 각각에 대해 모든 측면에서 그들보다 우수한 해는 없기 때문에 A, B, C 3개의 시나리오는 파레토 최적이라고 할 수 있다.

3.2 MAGA 기법

파레토 최적해를 찾기 위해 GA를 적용하는 경우 해결해야할 문제점은 크게 두 가지로 요약할 수 있다.

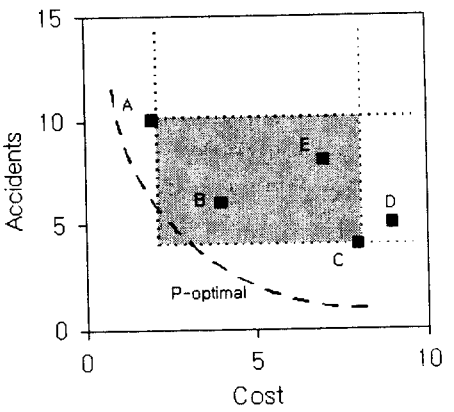
첫번째는 세대 교체를 통해서 지속적으로 파레토 경계를 향해서 개체가 진화되어야한다는 점이다.

두번째는 개체들이 파레토 경계를 향해 진화되는 동안 한 지점으로 수렴해서는 안되고 파레토 경계를 따라 균등하게 분산되어야 한다는 점이다.

첫번째 문제점에 대한 해결책으로 본 논문에서는 파레토-박스-선택(pareto-box-selection) 방식을 사용하였다. 파레토-박스-선택 방식은 VEGA에서 사용한 파레토-토너먼트(pareto-tournament) 방식이 양 극단에 치우쳐 진화되는 문제를 해결하기 위해 제시된 기법이다. N개의 개체를 랜덤하게 선택한 뒤 각 측면에서 가장 우성인 개체들을 선정하고 선정되지 않은 개체들 중에서 선정된 개체들에 대해서 비열성인 개체들이 있다면 추가로 선정하는 선택 방식으로 전체 개체수를 채울때까지 반복된다.

예를 들어, 〈그림 1〉의 예제와 같은 5개의 개체를 선정하여 파레토-박스-선택 방식을 적용한 경우, 그 결과는 〈그림 2〉와 같다.

〈그림 2〉에서 A, B, C, D, E 5개의 후보해를 선정하였을 경우, 비용측면에서 최소인 A와 사고 발생수에서 최소인 C는 우선적으로 선택된다. 다음으로 두 측면에서 최적인 A와 C에 의해 지배되지 않는 영역(표시된 부분)에 속한 해가 있는지를 평가하고 영역안에 해가 존재한다면 추가적으로 선택한다. 따라서, 〈그림 2〉에서는 5개의 시나리오중 A, B, C, E 4개의 후보해가 선택된다.



〈그림 2〉 파레토-박스-선택의 적용 예

위의 파레토-박스-선택 기법은 VEGA 기법[6]이 중간해를 유지하지 못하고 양 극점에 수렴하는 것을 개선하기 위한 방법으로 위의 경우 시나리오 E는 정확히 평가하면 시나리오 B에 의해 지배되지만 양극단으로의 수렴을 방지하고 다음 세대에 보다 우수한 파레토 해의 생성 가능성을 위해 선택한다. 즉, VEGA 기법이 각 측면에서의 최적해를 하나씩만 선택함으로써 세대 교체가 진행될수록 양극점으로 수렴해가는 것에 비해 파레토-박스-선택 기법은 후보해들을 파레토 경계쪽으로 균등하게 진화시킬 수 있다.

매번 파레토-박스-선택 기법에 의해 선정되는 개체수는 랜덤하게 선정된 N개의 개체들의 분포에 따라 다양하다. 최대 N개의 개체가 모두 선정될 수 있으며 최악의 경우에도 각 측면에서 가장 우수한 해들은 하나씩 선정된다. 선정된 개체들은 랜덤하게 짝을 지어 단일점(one-point) 교배 연산을 수행한다. 짝을 지은 개체들의 비트열중 특정 위치를 기준으로 앞의 비트열과 뒤의 비트열을 바꾸어 새로운 비트열을 만듦으로써 기존 개체의 우성 인자를 유지하면서 보다 우수한 개체의 생성을 기대할 수 있다.

두번째 문제점에 대해 본 연구에서는 DeJong이 GA의 조기 수렴(premature convergence) 문제를 해결하기 위해 제시한 crowding을 확장한 파레토 대치(Pareto Replace) 구조를 사용하였다[3]. 파레토 대치 구조는 새

로운 개체가 기존 세대 속에 어떻게 동화될 것인지를 결정하기 위해 세대간의 오버래핑을 허용하는 기법이다.

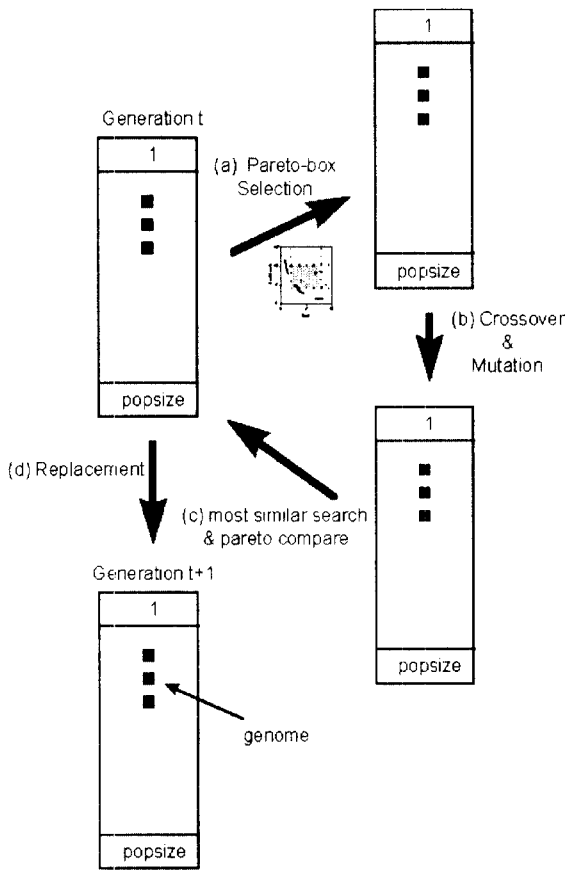
본 연구에서는 〈그림 3〉과 같은 파레토 대치 구조를 통해서 개체간의 다양성(diversity)을 유지하도록 한다. 파레토 대치 구조에서는 부모 세대로부터 파레토-박스-선택을 통해 선정된 후보 파레토 세대들에 대해 단일점 교배 연산 및 돌연변이 연산을 수행한다. 그리고, 그 결과 생성된 개체들 각각에 대해서 가장 유사한 부모 개체를 선정하고 만약, 파레토 최적 측면에서 자식 개체가 부모 개체보다 비열성일 경우 해당 부모 개체를 대체하는 과정을 반복 수행한다.

한편, 자식과 부모 개체와의 유사성(similarity)은 다음과 같은 척도에 의해 평가된다.

$$Ssimilarity(P,O) = \frac{1}{\left(\sum_{i=1}^{n} | fit(P_i) - fit(O_i) |^2 \right)^{1/2}}$$

위의 유사성 척도는 자손 개체 O, 부모 개체 P에 대해서 n개의 측면에서 적합도 fit() 차이를 계산한다. 다양한 측면을 고려한 해공간을 가정할 때 이 척도는 해들간의 거리에 기반한다. 따라서, Ssimilarity(P, O) 척도값이 높을수록 두 해의 유사성이 높은 것으로 평가된다.

다음은 제안한 MAGA 기법에 대한 개략적인 알고리즘을 표현한 것이다. MAGA는 기존 GA에 비해 파레토-박스-선택 기법과 파레토 대치 구조를 사용한다는 점에서 구별된다.



〈그림 3〉 파레토 대치구조

```
MAGA
begin
Generate initial population of N individuals
For i = 1 to MAX_GEN do
  Use Pareto-Box-Selection to produce candidate offsprings;
  For j = 1 to Pop_Size do
      Crossover;
      Mutation;
  For j = 1 to Pop_Size do
      Use Pareto Crowding to replace a parent
        with offspring j;
end

Pareto-Box-Selection
begin
For i = 1 to Pop_Size do
    Create candidate group by randomly picking Cp parent;
  For j = 1 to Num_Aspect do
      Select best-fit parent from candidate group
        by evaluation in jth aspect;
      Append best-fit parent to ith offspring;
      i=i+1;
  For k = 1 to Cp do
      if (.NOT. ( ∀ best-fit parents ≺ k-th parent))
        Select parent k from candidate group ;
        Append parent k to i+1th offspring;
        i=i+1 ;
end

Pareto Replacement
begin
  For i = 1 to Pop_Size do
      Create replacement group by randomly picking Cf parent;
      Select offspring i;
      Select best-similar individual with offspring i by
        similarity factor;
      if (.NOT. best-similar parent ≺ offspring i)
        Replace best-similar parent with offspring i;
end
```

MAGA에서 파레토 대치 구조는 생성된 후보해들이 가장 유사한 개체만을 대치하도록 함으로써 세대 교체를 통한 특정 해의 손실을 최소화하였다. 또한, 생성된 후보해가 파레토 측면에서 열성이 아닌 경우엔 유사해를 대치하도록 허용함으로써 활발한 세대 교체를 통한 새로운 후보해의 생성을 유도하였다.

4. 적용 예

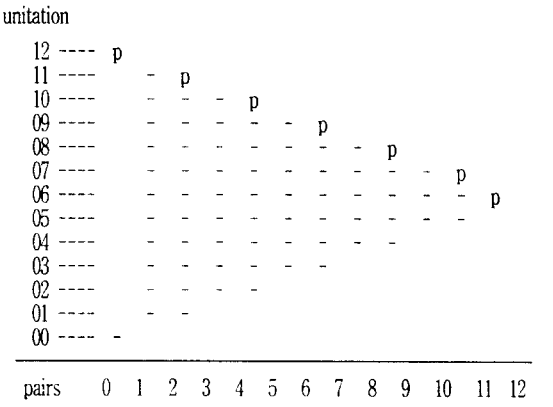
4.1 [문제1] unitation versus pairs

먼저, 제안한 MAGA 최적화 기법을 평가하기 위하여 쉽게 계산될 수 있는 파레토 최적해 경계를 가진 간단한 문제에 적용시켜보았다.

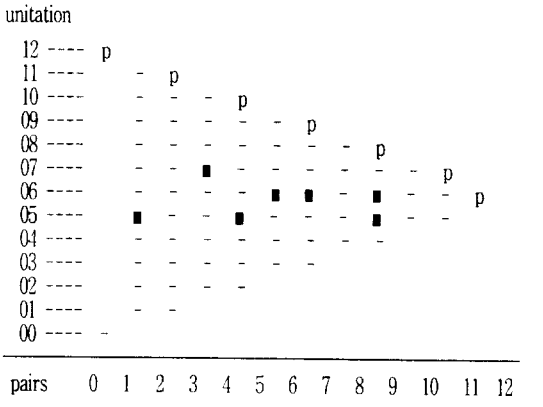
[문제 1]은 "unitation versus pairs"라 부를 수 있으며 주어진 이진 문자열에 대한 두 가지 측면에서 최적화를 요구한다. 먼저, 한 측면은 이진 문자열에서 비트 1의 갯수인 unit[s]를 최대화시키는 것이고 다른 한 측면은 이진 문자열중에서 인접한 비트 값이 1에서 0으로 혹은 0에서 1로 반전되는 갯수인 prs[s]를 최대화시키는 것이다. 따라서, unit[0011010010] 값은 4, prs [0011010010] 값은 6이 되며 이 두 측면은 서로간에 트레이드-오프 관계를 갖는다.

〈그림 4〉는 12-비트 문제에 대해 unit와 prs 두 측면을 고려할 경우에 생성 가능한 해의 2차원 공간을 보여준다. 〈그림 4〉에서 'p'는 파레토 최적해를 표시하며 '-'는 생성 가능한 모든 후보해를 표시한 것으로 파레토 경계에 위치한 최적해 p에 비해 모든 측면에서 열성인 후보 해들을 표시한다. 즉, unit[01010]과 unit [00110]은 모두 2로 같지만 prs[01010]은 4로 prs[00110]가 2인 경우에 비해 우성이므로 후보해 01010이 후보해 00110보다 파레토 최적이라고 할 수 있다.

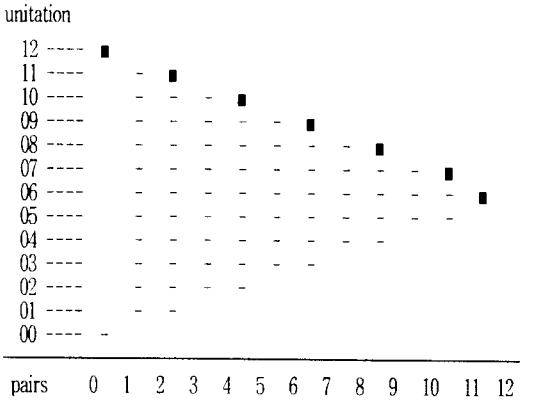
한편, 〈그림 5〉는 랜덤하게 생성된 12-비트 개체 7개로 이루어진 초기 세대를 2차원 공간에 표시한 것이다. 그리고, 교배 연산율 0.9, 돌연변이율 0.1에 의해 7개의 개체들에 대한 10000번의 세대교체 이후에 발생한 개체들을 그래프에 표시한 결과가 〈그림 6〉이다. 이때 적용한 매개변수들은 GA의 특성상 여러 번의 실험을 통하여 가장 적절한 값을 선정한 것으로 특히, 파레토 최적화의 특성상 각 개체간의 수렴을 최소화하면



〈그림 4〉 파레토 최적해



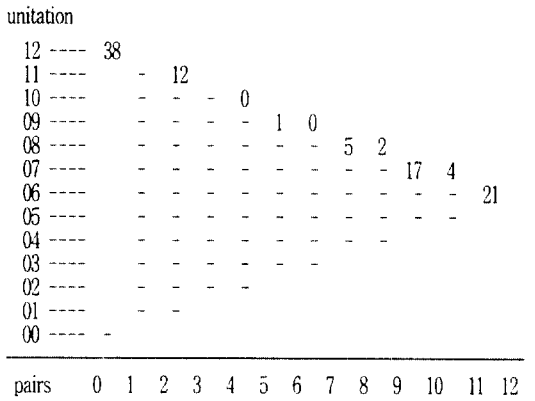
〈그림 5〉 초기해



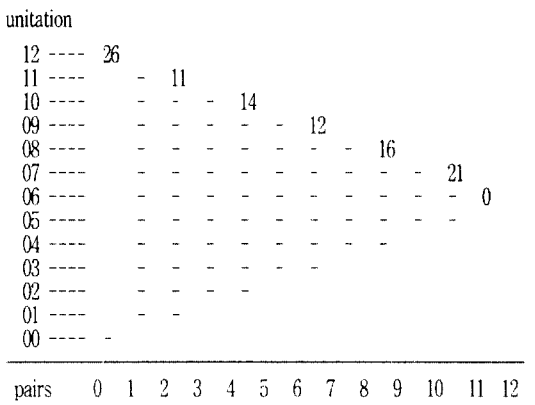
〈그림 6〉 파레토 최적해로의 수렴(최중해)

서 우수한 개체로의 진화를 위해 교배 연산율은 높게, 돌연변이율은 낮게 설정하였다. 〈그림 6〉을 통해서 제안한 MAGA 최적화 기법이 파레토 최적 집합의 모든 멤버들을 찾는 데 성공했음을 확인할 수 있으며 다른 연구 결과에 비해 최소의 개체들을 이용하여 모든 가능한 해를 찾아낼 수 있음을 알 수 있다.

〈그림 7〉과 〈그림 8〉은 [문제 1]의 동일한 실험을 기존 기법들이 100개의 개체를 가지고 수행한 결과를 표시한 것이다. 〈그림 7〉의 VEGA 기법[6]은 파레토 최적을 모두 찾지 못했으며 양극으로 편중되어 있음을 볼 수 있다.



〈그림 7〉 VEGA의 결과

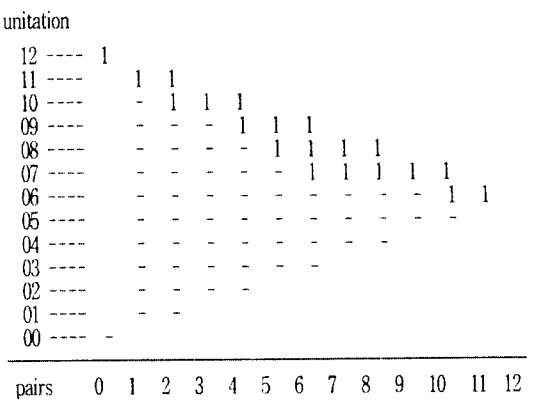


〈그림 8〉 Niched Pareto GA의 결과

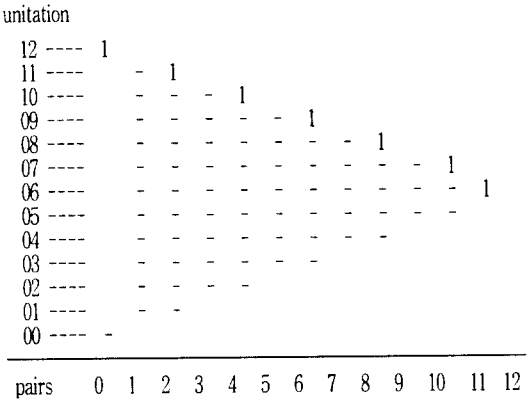
〈그림 8〉의 Niched Pareto GA 기법[7]은 비교적 균등하게 각 개체를 분산시키면서 파레토 최적해만을 찾았으나 7개의 파레토 최적해중 하나를 찾지 못했다. 이것은 각 개체간의 일정 거리 정보에 의존해서 개체를 진화시킴으로써 [문제 1]처럼 파레토 최적해간의 거리가 일정하지 않은 경우 문제점이 있기 때문이다.

한편, 제안한 MAGA 기법은 [문제 1]과 같이 파레토 최적해의 개수를 미리 알 수 있는 경우 〈그림 6〉에서처럼 최소(7개)의 개체만으로 모든 가능한 해를 찾아낼 수 있다. 그러나, 주어진 문제에 대해서 파레토 최적해의 정확한 개수를 사전에 알 수 있는 경우는 많지 않고 특히, 많은 최적화 문제가 예측할 수 없는 많은 파레토 최적해를 갖기 때문에 이에 대한 고려가 필요하다. [문제 1]에 대해서 한 세대의 개체수를 20개로 증가시켜 MAGA를 적용한 실험 결과는 〈그림 9〉와 같다.

〈그림 9〉 (A)의 결과는 MAGA가 crowding 기법을 사용한 파레토 대치 구조를 적용함으로써 한 세대안에 동일한 개체가 생성될 수 없기 때문이다. 즉, 7개의 파레토 최적해를 제외한 나머지 개체들은 파레토 최적해에 인접한 해로 수렴될 수밖에 없다. 그러나, 〈그림 9〉 (A)의 20개의 최종해들에 대해 상호간의 파레토 최적 비교를 수행할 경우, 7개의 파레토 최적해에 대해 나머지 13개의 해들은 열성이므로 〈그림 9〉 (B)처럼 7개의 파레토 최적해를 최종적으로 얻을 수 있다. 따라서,



(A) MAGA 적용 결과(pop_size=20)

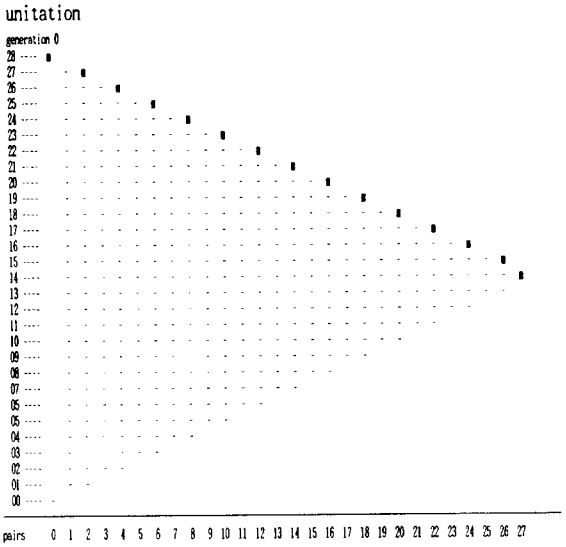


(B) 최종해 (파레토 최적 비교를 수행한후)

〈그림 9〉 MAGA 결과(개체수 〉 파레토최적해수)

서, 제안한 MAGA는 개체수가 파레토 최적해보다 많을 경우에는 파레토 최적 비교를 통해서 원하는 파레토 최적해만을 얻을 수 있으며 초기 개체수가 파레토 최적해보다 적거나 같을 경우에도 주어진 개체들만으로 다양한 파레토 최적해를 생성함으로써 주어진 문제에 대한 파레토 최적 경계를 파악할 수 있도록 한다. 또한, 0.1-0.9 사이의 다른 랜덤 씨드(seed)를 이용하여 초기 개체를 다르게 생성시킨 실험에서도 7개의 파레토 최적해를 모두 찾을 수 있었으며 〈그림 10〉처럼 비트의 길이를 늘여서 28-비트 문제에 적용했을 경우에도 만족스러운 결과를 얻을 수 있었다.

[문제 1]의 경우 GA를 적용함으로써 파레토 경계에 속한 한 해를 찾아내는 것은 어렵지 않다. 하지만, 후보해의 생성 비율을 고려할 경우 경계의 중간 부분에 대부분의 해가 수렴하고 양 극단에는 하나 혹은 둘 정도의 개체만 생성될 수 있으므로 후보해의 상대적인 생성 비율을 벗어나 양 극단의 해까지 찾아내고 지속적으로 유지한다는 것은 매우 어렵다. 따라서, [문제 1]에 대한 적용을 통하여 제안한 MAGA 기법은 파레토 최적해를 찾기위한 적절한 수렴성(convergence)과 다양성(diversity)을 가지고 있다고 평가할 수 있다. 또한, 유사한 다양성을 갖는 Niched Pareto GA와 MAGA를 시간 복잡도(time complexity)를 기본적인 연산 수행 횟수 측면에서 평가한 결과, Niched Pareto GA는 $O(n^2)$, MAGA는 대치 구조의 cf(crowding factor)값에 따라 최



〈그림 10〉 28-비트 Unitation & Paris 문제의 결과

대 $O(n^2)$, 최소 $O(n)$ 시간을 가짐을 분석할 수 있다. 이로써 MAGA는 Niched Pareto GA보다 적은 연산 시간을 갖거나 최악의 경우에도 더 많은 연산 시간을 요구하지는 않는다는 것을 알 수 있다. [문제1]의 결과를 토대로 기존 연구들과의 비교를 개략적인 표로 표시하면 다음과 같다.

비교항목 \ 기법	VEGA	Niched Pareto GA	MAGA
수렴성	○	○	○
다양성	×	○	○
연산량	$O(n)$	$O(n^2)$	최대 : $O(n^2)$ 최소 : $O(n)$

4.2 [문제2] Data Allocation Problem

적용하고자 하는 두 번째 문제는 보다 실제적인 문제로서 분산 시스템의 최적 운영 환경을 위한 데이터 할당 최적화 문제이다. 데이터 할당 문제는 NP-complete 문제로 그동안 많은 연구가 수행되어 다양한 휴리스틱이 제시되었지만 특정한 단일 측면만을 고려하고 있다는 문제점이 있다.

분산 시스템에서 적절한 데이터 할당은 시스템 자원들을 적절히 공유함으로써 데이터에 대한 접근 비용이나 기억 장소를 크게 줄여줄 수 있을 뿐만 아니라 전체 시스템의 가용성도 향상시킨다. 데이터 할당시에 중복을 많이 하게 되면 읽기 트랜잭션의 경우 지역 처리만으로 수행이 가능하므로 성능이 향상되지만, 쓰기 트랜잭션은 복사본을 가지고 있는 모든 노드에 접근하여 데이터에 대한 갱신을 수행해야하므로 통신 오버헤드가 증가하여 성능이 저하된다. 또한, 데이터의 중복도를 향상시킬 경우 통신선이나 노드에 고장이 발생할 경우에도 중복된 데이터를 이용한 처리가 가능함으로써 데이터에 대한 가용성이 증대된다. 따라서, 적정 수준으로 최적의 노드에 데이터를 할당할 수 있는 최적화 기법이 필요하다.

두 번째 문제인 데이터 할당 문제는 이진문자열로 표현된 데이터 할당 해에 대한 비용을 최소화하면서 가용성을 최대화하는 문제로 다음과 같이 정의한다.

데이터들의 집합(D), 노드들의 집합(N), 트랜잭션 집합(T)을 각각

$$D = \{ D_1, D_2, \ldots, D_d \}$$
$$N = \{ N_1, N_2, \ldots, N_n \}$$
$$T = \{ T_1, T_2, \ldots, T_t \}$$

와 같이 정의하고 데이타 할당 해 x를

$$x_{ij} = \begin{cases} \text{데이터 } D_i\text{가 노드 } N_j\text{에 있으면} & 1 \\ \text{그렇지 않으면} & 0 \end{cases}$$

과 같이 정의하자. 그러면 데이터 할당 모델은 D를 N에 할당시키는 "최적의 해" x^*를 찾아내는 문제로 정의할 수 있다. 해의 좋고 나쁨은 목적 함수(objective function) f로 평가되며 "최적의 해집합" X^*는 비용 및 가용성 목적 함수에 대해 우수한 값을 나타내는 파레토 해집합이다. 즉, S를 데이터 할당의 해공간(solution space)이라 할때, 파레토 해집합은

$$\forall x^* \in X^*, \forall x \in S\text{-}X^*:$$
$$(f_C(x) < f_C(x^*) \wedge f_A(x) > f_A(x^*))$$

를 만족하는 X^*를 의미한다.

데이터 할당 문제에서는 두가지 측면에서의 최적화를 고려한다. 첫 번째는 비용을 최소화하는 것이며 두 번째는 가용성을 최대화하는 것이다. 비용과 가용성 척도는 다음과 같이 정의된다.

■ 비용(cost) : 임의 트랜잭션의 수행 비용

$$f_C(x) = \sum_{k \in T} (\text{CPU 비용} + \text{통신비용} + \text{IO비용}) + \sum_{t \in N} \sum_{i \in D} \text{저장비용}_{it}$$

■ 가용성 : 임의 트 잭션의 수행 완료 확률

$$f_{A(x)} = \sum_{k \in T} (\text{CPU 가용성} * \text{통신 가용성} * \text{IO 가용성})$$

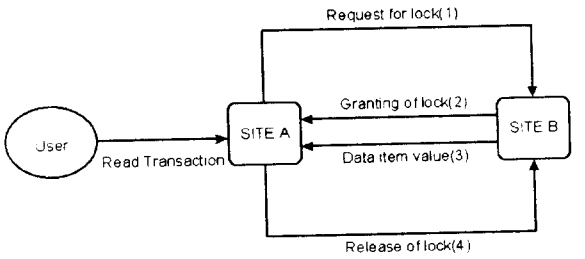
비용을 목적 함수로 하는 경우, 트랜잭션을 처리하는데 드는 총 비용(CPU비용,통신비용, IO비용의 합)과 데이터를 각 노드에 저장하는 비용(저장 비용)을 최소로 하는 해를 탐색한다. 반면에 가용성을 목적 함수로 하는 경우, 트랜잭션의 처리를 위해 필요로 하는 모든 자원들이 가용할 확률(CPU 가용성, 통신선 가용성, IO 가용성의 곱)을 최대로 하는 해를 탐색한다.

분산 시스템에서의 데이터 할당 시뮬레이션을 위해서 〈그림 11〉과 같은 2단계 로킹(2-phase locking) 기법을 사용하는 트랜잭션 모델을 사용한다. 〈그림 11〉의 트랜잭션 처리 모델은 읽기 트랜잭션의 경우 오직 한 데이터만을 판독하고, 쓰기 트랜잭션의 경우 모든 데이터를 갱신하는 ROWA(Read One Write All) 규칙을 모델링한 것이다.

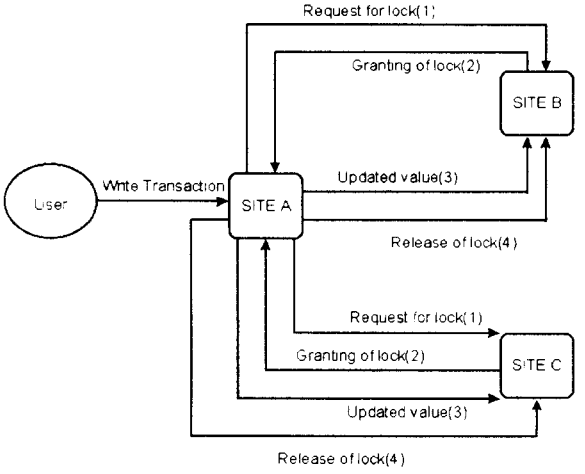
시뮬레이션에서는 〈그림 12〉와 같은 분산 환경을 가정하였으며 이때 사용한 매개 변수들(노드, 저장장치, 통신선 등에 대한 비용 및 가용성)은 〈표 1〉, 〈표 2〉, 〈표 3〉에 제시되어 있다.

각 통신선의 가용성 값은 [14]에 근거하여 0.95 이상의 값들을 랜덤하게 할당하였으며 노드 및 저장 장치의 가용성 및 비용은 동일하게 가정하였다. 각 통신선에 대한 단위 비용은 0.25로 고려하였으며, 〈표 3〉은 〈그림 12〉와 같은 분산 환경에서 특정한 두 사이트간의 단위 통신 비용을 계산한 것이다.

한편, 발생하는 트랜잭션의 타입은 6개로 각각의 트랜잭션이 S_1부터 S_6에서 일정 비율로 반복 발생하는 것을 가정하였다. 발생되는 트랜잭션은 〈표 4〉에서 보는

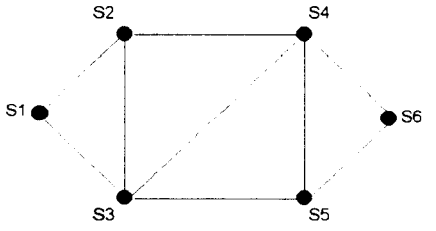


(A) 읽기 트랜잭션



(B) 쓰기 트랜잭션

〈그림 11〉 시뮬레이션 트랜잭션 모델링



〈그림 12〉 분산 환경

바와 같이 4개의 읽기 트랜잭션과 2개의 쓰기 트랜잭션으로 나누어진다. 읽기 트랜잭션의 경우에는 필요로 하는 데이터 저장 노드중 하나만을 접근하고 쓰기 트랜잭션의 경우에는 데이터 저장 노드들 모두에 대해 접근해야 한다.

〈표 1〉 노드 및 저장 장치의 가용성 & 비용

	S_1	S_2	S_3	S_4	S_5	S_5
node_av	0.95	0.95	0.95	0.95	0.95	0.95
IO_av	0.95	0.95	0.95	0.95	0.95	0.95
CPU_co	0.3	0.3	0.3	0.3	0.3	0.3
IO_co	0.2	0.2	0.2	0.2	0.2	0.2

〈표 2〉 통신선의 가용성

link_av	S_1	S_2	S_3	S_4	S_5	S_6
S_1	0	0.95	0.96	0	0	0
S_2	0.95	0	0.98	0.98	0	0
S_3	0.96	0.98	0	0.97	0.99	0
S_4	0	0.98	0.97	0	0.99	0.95
S_5	0	0	0.99	0.99	0	0.96
S_6	0	0	0	0.95	0.96	0

〈표 3〉 통신선의 비용

link_co	S_1	S_2	S_3	S_4	S_5	S_6
S_1	0.0	0.25	0.25	0.5	0.5	0.75
S_2	0.25	0.0	0.25	0.25	0.5	0.5
S_3	0.25	0.25	0.0	0.25	0.25	0.5
S_4	0.5	0.25	0.25	0.0	0.25	0.25
S_5	0.5	0.5	0.25	0.25	0.0	0.25
S_6	0.75	0.5	0.5	0.25	0.25	0.0

〈표 4〉 트랜잭션 정보

트랜잭션	타입	발생 갯수	발생 사이트	접근 데이터
T1	R(읽기 트랜잭션)	(0 - 250)	S_1	F1, F3
T2			S_2	F1, F6
T4			S_3	F1, F5
T5			S_4	F3, F5
T3	W(쓰기 트랜잭션)	(0 - 500)	S_5	F2, F4
T6			S_6	F2, F3
총 6개	읽기 : 4 쓰기 : 2	총 1000개	총 6 사이트	총 6 데이터

트랜잭션이 접근하는 데이터의 전체 갯수는 6개로 읽기 트랜잭션, 쓰기 트랜잭션들은 모두 데이터 2개씩을 필요로 한다. 6개의 데이터 각각의 크기는 모두 10 Kbyte로 동일하게 고려하였다.

〈표 4〉의 트랜잭션들은 다양한 트랜잭션 형태를 모델링한 것으로 발생 비율을 조정함으로써 분산 트랜잭션 환경을 위한 실험 데이터로 활용할 수 있다. 즉, 읽기 트랜잭션간의 경쟁, 읽기/쓰기 트랜잭션간의 충돌, 쓰기 트랜잭션간의 충돌들을 모두 포함하고 있어 데이터 할당시에 발생하는 모든 경우를 고려할 수 있다.

위와 같은 시뮬레이션 환경을 가정한 경우 데이터 할당해는 다음과 같이 표현될 수 있다.

x = 110001 000100 010001 100001 000011 010000

위의 후보해 x는 데이터 F1은 사이트 S_1,S_2,S_6에, F2는 S_4, F3은 S_2,S_6, F4는 S_1,S_6, F5는 S_3,S_5, F6은 S_2에 저장됨을 나타낸다. 이중에서 트랜잭션 T3의 처리 비용 및 가용성만을 평가해보면 다음과 같다.

$f_c(T3)$ = F2 갱신비용 + F4 갱신비용
= (4*0.3+4*0.25+0.2) + (4*0.3+4*0.25+4*0.5+2*0.2)

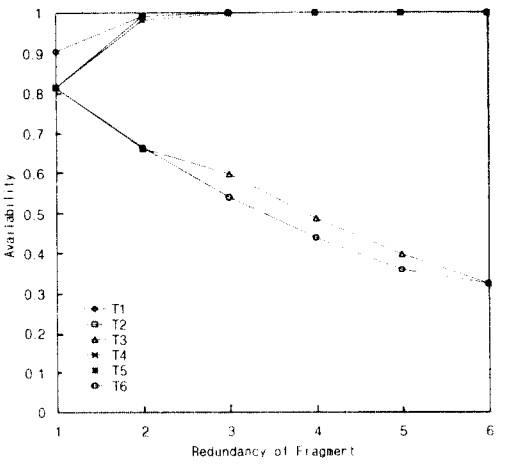
$f_A(T3)$ = F2 갱신가용성*F4 갱신가용성
= (4*0.95*4*0.97*0.95) * (4*0.95*4*0.96*4*0.99*0.96*2*0.95)

트랜잭션 T3은 사이트 S_2에서 발생하는 쓰기 트랜잭션으로 위의 비용은 ROWA 규칙에 따라 데이터 F2(S_2에 위치)와 F4(S_1과 S_6에 위치)를 갱신함에 따른 비용을 계산한 것이다. 동일한 방법으로 다른 트랜잭션들에 대한 비용 및 가용성을 구하고 각 사이트의 저장 비용을 추가적으로 고려함으로써 전체 비용과 가용성 값을 계산할 수 있다.

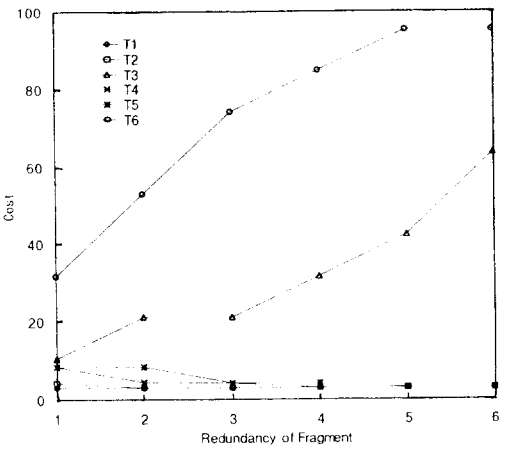
한편, MAGA는 위와 같은 방법을 통해 평가된 복수 해들의 비용 및 가용성값을 비교하여 비용은 낮으면서 가용성은 높은 쪽으로 해를 유도해간다. 예를 들어, 한 세대의 전체 개체수가 50인 경우 N(N≤50)개의 개체들을 랜덤하게 선택하여 〈그림 2〉와 같은 파레토-박스-선택을 50개의 개체가 선택될때까지 반복한 뒤 이진 문자열로 표현된 해들에 대해 단일점 교배 연산과 돌연변이 연산을 적용한다. 이후, 유연자 연산을 통해 생성된 50개의 새로운 개체는 파레토-대치구조를 통해서 이전 세대의 개체들과의 적합도 비교를 하게 된다. 랜덤하게 선택된 새로운 개체가 이전 세대중 가장 유사한 개체보다 파레토 최적이면 이전 개체를 대치하는 과정을 수행하는 파레토-대치구조를 개체수만큼 반복함으로써 한 세대간의 전이를 완료하며 전체적으로는 주어진 generation값만큼 위의 전과정을 반복한다.

먼저, 데이터 할당에 관한 시뮬레이션에서 평가 요소로 삼은 가용성과 비용 변화를 분석하기 위해 읽기/쓰기 트랜잭션에 대한 동일한 빈도를 가정했을 경우의 데이터 중복에 따른 가용성 및 비용의 변화를 평가하였다.

〈그림 13〉과 〈그림 14〉를 통해서 데이터를 중복 할당함에 따라 읽기 트랜잭션 T1, T2, T4, T5는 가용성 및 비용면에서 점차적으로 향상되었으며 쓰기 트랜잭션 T3,T6의 경우 가용성은 하락하고 비용은 큰 폭으로 상승하였다. 즉, 데이터 할당시에 읽기 트랜잭션의 발생 빈도가 상대적으로 높을 경우 점차적으로 데이터의 중복도가 증가될 수 있음을 알 수 있다. 이는 읽기 트랜잭션의 경우, 쓰기 트랜잭션에 비해 접근해야 하는 데이터의 갯수가 데이터의 중복에 영향을 받지 않으므로 데이터의 중복을 높일수록 읽기 트랜잭션의 데이터



〈그림 13〉 데이터 중복에 따른 가용성

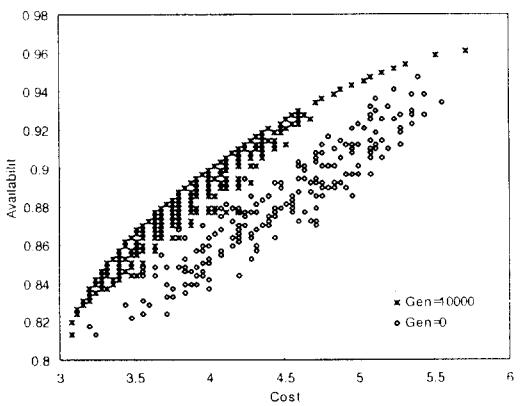


〈그림 14〉 데이터 중복에 따른 비용

에 대한 접근 경로가 그만큼 증가되기 때문이다.

이처럼 분산 시스템에서의 데이타 할당은 트랜잭션의 타입, 데이터의 위치 및 중복도에 따라 비용과 가용성이 다양하게 동적으로 변화하므로 시뮬레이션을 통한 최적화가 필요하다.

〈그림 15〉는 앞에서 기술한 분산 환경에서의 트랜잭션 시뮬레이션을 통한 데이터 할당 최적화에 MAGA를 적용한 결과이다. 실험 결과는 MAGA를 세대 크기 200, 세대 생성 횟수 10000, 교체 연산율 0.9, 돌연변이율 0.1로 설정했을 경우의 얻어진 값이다. 교체 연산율과 돌연변이율을 [문제 1]의 경우와 동일하게 설정하



〈그림 15〉 데이터 할당 문제에 대한 MAGA 적용 결과

였으며 읽기 트랜잭션 비율(0-250)과 쓰기 트랜잭션 비율(0-500)을 조합하여 변화시킴으로써 다양한 트랜잭션 운영 환경을 시뮬레이션 하였다.

〈그림 15〉에서 최적화 하고자 하는 두 측면 비용과 가용성간에는 트레이드-오프가 존재함을 확인할 수 있다. Gen=10000는 최종해들로서 200개의 개체를 랜덤하게 초기화한 후 10000번의 세대 교체동안 파레토-박스-선택 기법과 파레토 대치 구조를 반복 적용함에 의해서 파레토 최적 경계로 추측되는 트레이드-오프 경계로의 수렴 결과를 얻었다. 데이터 할당 문제에서 파레토 최적해의 갯수를 사전에 알기 어렵기 때문에 개체 200개를 임의로 생성하였다. 10000번의 세대 교체를 통한 최종 결과에서 많은 수의 해들이 파레토 최적이 아닌 파레토 경계에 근접한 결과를 보였으나 [문제 1]의 경우처럼 파레토 최적 비교를 통해서 열성인 해들을 제거할 수 있으므로 큰 문제가 되지 않는다.

[문제 1]보다 복잡하고 실제적인 [문제 2]의 경우를 통해서도 제안한 MAGA 기법이 파레토 최적 경계를 따라서 최적해들을 분산 수렴해감을 확인할 수 있었다. 따라서, MAGA를 이용하여 주어진 문제를 최적화할 경우 다양한 파레토 최적해를 얻을 수 있으며 시뮬레이션을 통해 다측면을 고려하면서 최적화해야 하는 많은 문제들에 대해 효과적으로 활용할 수 있음을 알 수 있다.

5. 결 론

본 논문에서는 다측면을 고려하는 시뮬레이션 최적화 기법으로서 MAGA 기법을 제안하였다.

제안한 MAGA 기법은 단순 유전자 알고리즘을 수정 확장한 형태로 파레토 경계에 위치한 최적 해들을 균등하게 생성한다. unitation versus pairs 문제와 data allocation problem 2가지 문제에 대한 실험 결과를 통하여 제안한 MAGA 기법이 파레토 최적해들을 균등하게 트레이드-오프 경계로 분산 수렴시킴으로써 다중 측면을 고려해야하는 모든 최적화 문제에 적용될 수 있음을 보였다.

추후 연구로는 3개 이상의 다측면을 고려해야하는 보다 복잡한 최적화 문제에 대한 평가가 필요하며 최종적으로 얻어진 파레토 해들간의 분포에 있어 균일성

의 향상을 좀더 고려할 필요가 있다.

참고문헌

[1] 양병희, 이영해, "시뮬레이션 최적화 기법과 절삭 공정에의 응용", 한국시뮬레이션학회 논문지, 제3권 제2호(1994), pp. 57-67.

[2] C.M. Fonseca and P.J. Fleming, "Genetic algorithms for multiobjective optimization: formulation, discussion and generalization", Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kauffman(1993), pp. 416-423.

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

[4] D.E. Goldberg and J.J. Richardson, "Genetic algorithms with sharing for multimodal function optimization", Genetic Algorithms and Their Applications: Proceedings of the Second ICGA, Lawrence Erlbaum Associates, Hillsdale, NJ(1987), pp. 41-49.

[5] F. Azadivar, "A Tutorial in Simulation Optimization", Proceedings of the 1992 Winter Simulation Conference(1992), pp. 198-204.

[6] J.D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms", In J. Grefenstette, ed., Proceedings of an International Conference on Genetic Algorithms and their Applications(1985), pp. 93-100.

[7] J. Horn, N. Nafpliotis, and D.E. Goldberg, "A niched pareto genetic algorithm for multiobjective optimization", Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, Vol. 1(1994), pp. 82-87.

[8] J. Horn and N. Nafpliotis, "Multiobjective optimization using the niched Pareto genetic algorithm", IlliGAL Report No. 93005, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 1993.

[9] J.T. Richardson, M.R. Liepins, and M. Hilliard, "Some guidelines for genetic algorithms with penalty function", Proceedings fo the Third International Conference on Genetic Algorithms, Morgan Kauffman(1989), pp. 191-197.

[10] M.H. Safizadeh, "Optimization in Simulation: Current Issues and the Future Outlook," Naval Research Logistics, Vol. 37(1990), pp. 807-825.

[11] M. Syrjakow, H. Szczerbicka, "Simulation and Optimization of Complex technical systems", Proceedings of SCSC'95(1995), pp. 86-95.

[12] M.S. Meketon, "Optimization in Simulation : A Survey of Recent Results", Proceedings of 1987 Winter Simulation Conference(1987), pp. 58-67.

[13] S.J. Park, J.P. Roh, D.K. Baik, "A simulation approach to data partitioning for distributed database design", Proceedings of the SCSC'95(1995), pp. 813-818.

[14] W.W. Yang, "Terminal-Pair Reliability of Tree-Type Computer Communication Networks", IEEE Trans. on Reliability, Vol. 41, No. 1, pp. 49-56, 1992.

● 저자소개 ●

박성진

1991년	고려대학교 전산학과 (이학사)
1993년	고려대학교 전산과학과 (이학석사)
1993년~현재	고려대학교 전산과학과 박사과정
관심분야	모델링 및 시뮬레이션, 분산 데이터베이스

백두권

1974년	고려대학교 수학과 (이학사)
1977년	고려대학교 산업공학과 (공학석사)
1983년	Wayne State Univ. 전산과학과 (이학석사)
1986년	Wayne State Univ. 전산과학과 (이학박사)
1986년~현재	고려대학교 컴퓨터학과 교수
관심분야	모델링 및 시뮬레이션, 데이터 공학