특징 선택을 위한 혼합형 유전 알고리즘과 분류 성능 비교

(Hybrid Genetic Algorithms for Feature Selection and Classification Performance Comparisons)

오 일 석 † 이 진 선 †† 문 병 로 †††
(Il-Seok Oh) (Jin-Seon Lee) (Byung-Ro Moon)

요 약 이 논문은 특징 선택을 위한 새로운 혼합형 유전 알고리즘을 제안한다. 탐색을 미세 조정하기 위한 지역 연산을 고안하였고, 이들 연산을 유전 알고리즘에 삽입하였다. 연산의 미세 조정 강도를 조절할 수 있는 매개 변수를 설정하였으며, 이 변수에 따른 효과를 측정하였다. 다양한 표준 데이타 집합에 대해 실험한 결과, 제안한 혼합형 유전 알고리즘이 단순 유전 알고리즘과 순차 탐색 알고리즘에 비해 우수함을 확인하였다.

키워드 : 특징 선택, 혼합형 유전 알고리즘, 순차 탐색 알고리즘, 지역 탐색 연산

Abstract This paper proposes a novel hybrid genetic algorithm for the feature selection. Local search operations are devised and embedded in hybrid GAs to fine-tune the search. The operations are parameterized in terms of the fine-tuning power, and their effectiveness and timing requirement are analyzed and compared. Experimentations performed with various standard datasets revealed that the proposed hybrid GA is superior to a simple GA and sequential search algorithms.

Key words : feature selection, hybrid genetic algorithm, sequential search algorithm, local search operation

1. 서 론

특징 선택(feature selection)은 주어진 최적 함수 하에서, D개의 특징 중 쓸모가 없거나 중복되거나 덜 유용한 것들을 제거하여 d개를 선택하는 문제이다. 특징 선택의 목적은 가능한 한 적은 성능 희생을 감수하며 보다 간결한 분류기를 설계하는 것이다. 이 선택 문제는 지수적 계산 복잡도를 가지므로, 특징 벡터의 크기(즉, D)가 큰 경우 최적해(optimal solution)를 구할 수 없고, 부최적해(sub-optimal solution)를 구하는 알고리즘을 사용해야 한다. 특징 선택에 관한 튜토리얼 논문[1,2]과 비교 연구 논문[3-5]이 발표되어 있다.

특징 선택을 위한 유전 알고리즘(genetic algorithm)은 최근에 개발되었다. 유전 알고리즘은 생물학에서 영감을 얻어 자연 진화를 흉내낸 방법론으로, 과학과 공학의 많은 최적화 탐색 문제를 푸는데 활용되고 있다[6]. 특징 선택은 지수적 탐색 공간을 갖는 탐색 문제의 하나이므로 자연스럽게 유전 알고리즘을 적용할 수 있다. Siedlecki와 Sklansky의 선구적인 연구에서는 기존의 순차 탐색(sequential search) 알고리즘에 비해 유전 알고리즘이 우수함을 실험으로 보여주었다[7]. 그 후 특징 선택 문제에서의 유전 알고리즘의 장점을 주장하는 많은 논문들이 발표되었다[8-11]. 이들 논문에서는 유전 알고리즘의 우수성을 보여주는 실험 데이타를 제시하였지만, 비교 연구를 수행한 다른 논문들에서는 성능에 관한 주장이 일치하지 않는다. 이들 비교 연구와 우리 실험 결과에 의하면 [12]에서 제시된 SFFS(sequential floating forward search) 알고리즘이 순차 탐색 알고리즘들 중에서는 가장 좋은 성능을 제공하였다. 그러나 SFFS와 유전 알고리즘의 비교에 대해서는 일치하지 않는다. SFS, PTA, SFFS와 같은 순차 알고리즘에 대해서는 튜토리얼 논문을 참고할 수 있다[1].

Ferri등의 연구에서는 순차 탐색 알고리즘들 중에서 SFFS가 최상이었고, SFFS와 유전 알고리즘이 비슷하

† 종신회원 : 전북대학교 전자정보공학부 교수
isoh@chonbuk.ac.kr
†† 정 회 원 : 우석대학교 컴퓨터공학과 교수
jslee@woosuk.ac.kr
††† 비 회 원 : 서울대학교 컴퓨터공학부, 교수
moon@soar.snu.ac.kr
논문접수 : 2003년 6월 9일
심사완료 : 2004년 5월 29일

지만 차원(즉, D 값)이 증가함에 따라 유전 알고리즘이 SFFS에 비해 열등하게 되는 경향을 보였다[3]. Jain과 Zonker는 SFFS가 유전 알고리즘을 포함하여 다른 모든 알고리즘들에 비해 우세하다는 결론을 제시하였다[4]. 이들의 실험에서는 유전 알고리즘은 7~8번째 세대에서 최고 성능에 도달하는 조기 수렴 현상을 보였고, 유전 알고리즘을 구현하는데 있어 매개변수 설정의 어려움을 토로하였다. Kudo와 Sklansky도 최근 연구에서 성능 비교 결과를 제시하였다[5]. 이들은 문제를 크기에 따라 세가지 범주, 즉 $0<D\leq19$은 작은 문제, $20\leq D\leq49$은 중간 문제, 그리고 $50\leq D$은 큰 문제로 나누었다. 이들은 SFFS가 순차 탐색 알고리즘들 중에서 최상이고, SFFS가 작은 크기와 중간 크기의 문제들에 적합한 반면 유전 알고리즘은 큰 문제에 적합하다는 결론을 내렸다. 이들의 주장은 [3]에서의 주장과 일치하지 않는다. Kudo와 Sklansky는 이러한 불일치가 유전 알고리즘의 구현 과정에서의 차이에 기인한다고 주장하였다.

우리는 이들 논문으로부터 다음과 같은 사실을 유도하였다. 이들은 우리 연구에 대한 동기를 제공하였으며, 이러한 문제에 대한 해결책 제시가 우리 논문의 중요한 공헌이라 할 수 있다.

- SFFS가 순차 탐색 알고리즘들 중에서 최상이나, 유전 알고리즘과 SFFS간에는 주장이 일치하지 않는다. 여러 연구에서 유전 알고리즘에 대한 우수함을 주장하였지만 보다 엄격한 성능 분석이 필요하다.
- 유전 알고리즘은 구현하는 데 있어 여러 변형과 매개변수가 존재하나, 구현에 필요한 알고리즘 명세가 충분하지 않다. 따라서 독자들이 같거나 비슷한 성능을 얻을 수 있도록 알고리즘의 세부 사항과 매개변수 설정 정보가 제공되어야 한다.
- 특징 선택 문제에 존재하는 영역 지식을 유전 알고리즘에 혼합하여 성능을 향상시키려는 시도가 없었다.

단순 유전 알고리즘(simple genetic algorithm)의 성능 한계는 여러 응용에서 밝혀져 있다. 무제한의 계산 자원을 허용하면 최적에 아주 가까운 해를 얻을 수 있지만, 이러한 가정은 비현실적이다. 이러한 한계를 극복하는 효과적인 방법은 문제 영역에 존재하는 지식을 연산화하여 유전 알고리즘에 삽입하는 것이다. 이러한 혼합형 유전 알고리즘(hybrid genetic algorithm)은 다양한 응용 분야에서 개발되어 성공적인 성능을 제공하고 있다[13-15].

이 논문에서는 특징 선택 문제를 위한 혼합형 유전 알고리즘을 제안한다. 기본적인 아이디어는 지역 탐색 연산(local search operation)을 단순 유전 알고리즘에 삽입하는데 있다. 지역 탐색 연산은 해를 지역 최적점(local optimum)을 향해 이동시킨다. 이러한 이동이 전 세대를 통해 누적되고 결국 상당한 정도의 성능 향상을 가져오게 된다. 혼합형 유전 알고리즘을 위해 몇 가지 지역 탐색 연산을 고안하였다. 다양한 데이타 집합에 대한 실험을 통해, 제안한 혼합 유전 알고리즘과 기존 알고리즘들의 성능을 분류 정확성(classification accuracy)면에서 비교하였다. 결론적으로, 제안한 혼합 유전 알고리즘이 기존 알고리즘들에 비해 월등히 우세하였다.

2장에서는 특징 선택 문제를 정의한 후, 단순 유전 알고리즘의 상세한 명세를 제시한다. 3장은 지역 탐색 연산을 정의한 후, 이를 이용한 혼합형 유전 알고리즘을 제시한다. 4장에서는 실험 결과를 토대로 성능 비교를 제시한다. 결론은 5장에서 기술한다.

2. 특징 선택을 위한 단순 유전 알고리즘

특징 선택은 주어진 최적 함수 하에서 D개의 특징을 갖는 집합으로부터 d개의 특징을 갖는 최적의 부분 집합을 선택하는 문제이다. D개의 특징들을 1에서 D까지의 서로 다른 정수를 사용하여 표기하고, 전체 특징 집합은 $U=\{1,2,\cdots,D\}$로 표기한다. 특징 선택 과정에서 X는 선택되어 있는 특징들의 집합을 나타내고 Y는 이때 선택되지 않은 나머지 특징들의 집합이라 하자. 어떤 순간에서든 $U=X\cup Y$이다. $J(X)$는 X의 성능을 측정하는 평가 함수를 나타낸다. 평가 함수 J는 응용 분야에 따라 다를 수 있다. 특정 데이타 집합에 대한 특정 분류기의 정확도일 수도 있고, 어떤 통계적 척도일 수도 있다.

안정형(steady-state) 유전 알고리즘의 처리 과정은 다음과 같다.

```
steady_state_GA()
{
  initialize population P ;
  repeat {
    select two parents p1 and p2 from P ;
    offspring = crossover(p1,p2) ;
    mutation(offspring) ;
    replace(P, offspring) ;
  } until (stopping condition) ;
}
```

2.1 염색체 표현

하나의 해(즉 염색체)는 D개의 비트를 갖는 이진 스트링으로 표현한다. 하나의 비트는 특징 하나를 나타내며, 값 1과 0은 각각 선택된 상태와 선택되지 않은 상태를 의미한다. 예를 들어 D=8일 때 염색체 00101000은 세 번째와 다섯 번째 특징이 선택되어 있는 해를 나타내며, 이는 X={3,5}와 Y={1,2,4,6,7,8}과 동일하다.

2.2 초기 해 집단

해 집단(population)의 초기화는 아래 코드로 수행한다. 함수 random_uniform()은 [0,1] 사이의 임의의 실

수를 생성한다. 어떤 염색체에서 선택된 특징의 평균 개수는 d임에 주목하자.

초기 해 집단 :
for (i=1 to $|P|$)
  for (each gene g in i-th chromosome)
    if (random_uniform()<d/D) g=1 ; **else** g=0 ;

2.3 적합도 계산, 선택, 대치, 그리고 중단

하나의 염색체는 선택된 특징 집합 X를 나타내므로, 평가 함수 J(X)로 적합도(fitness) 계산은 쉽게 이루어진다. 부분 집합 크기가 d여야 한다는 사실을 제약 조건으로 설정하고, 이 조건을 만족하지 않는 염색체에는 벌점(penalty)을 준다. 염색체 C의 적합도는 다음과 같이 정의한다. 여기서 X_C는 C에 해당하는 부분 집합이고, w가 벌점 계수일 때 penalty(X_C) = w*||X_C|-d| 이다. 여기서 |X_C|는 집합 X_C의 크기이다.

$$\text{fitness}(C) = \text{J}(X_C) - \text{penalty}(X_C)$$

염색체 선택에서 적합한 염색체일수록 생존 확률이 높도록 하기 위해, 순위에 기반한 룰렛 휠 방법을 사용한다. 해 집단에 있는 염색체들을 적합도에 따라 정렬하고 i-번째 염색체에게 비선형 함수 $P(i)=q(1-q)^{i-1}$로 선택 확률을 부여한다. q 값이 클수록 더 강한 선택압(selection pressure)을 갖게 된다. 선택 과정을 아래 코드가 보여준다.

룰렛-휠에 의한 염색체 선택 :
1. i-번째 누적 확률을 $p_i=\sum_{j=1,i}P(j)$ (i=1,..,n이고 p_0= 0)로 계산. (n은 해 집단 크기)
2. [0,p_n] 사이에 있는 임의의 수 r을 생성.
3. $p_{i-1}<r<p_i$인 i-번째 염색체를 선택.

이러한 과정을 통해 두 개의 부모 염색체를 선택한다. 교배 연산은 두 부모로부터 새로운 자손 염색체를 생성하고, 돌연변이 연산은 이 염색체에 약간의 변형을 가한다. 이렇게 만들어진 염색체가 두 부모에 비해 모두 우세하면 두 부모 중에서 자신과 비슷한 부모를 대치한다. 만일 두 부모 사이라면 열세한 부모를 대치한다. 그렇지 않으면 해 집단에서 가장 열세한 염색체를 대치한다. 이러한 대치 방법은 [14]에서 제시되었다. 유전 과정은 전체 세대 수가 미리 정한 최대 세대 수 T에 도달하면 중단한다.

2.4 교배와 돌연변이

일반적으로 사용하는 교배(crossover)와 돌연변이(mutation) 연산자를 약간 변형하여 사용한다. 그림 1에 예시한 것처럼 m개 자름 점을 임의로 선택한 후, 두 부모의 부분 염색체를 서로 교차하여 자식 염색체를 만드는 m-점 교배 연산자를 사용한다. 교차한 부분 염색체의 1의 개수가 서로 다를 수 있으므로, 선택된 특징 개수 측면에서 자식 염색체는 부모와 다를 수 있다.

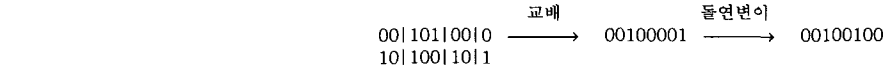

그림 1 3-점 교배와 돌연변이의 예

교배 후에는 돌연변이 연산을 적용한다. 돌연변이는 선택된 특징 개수를 큰 폭으로 다르게 할 수 있으므로 1-0 변환과 0-1 변환의 개수를 비슷하도록 조절할 필요가 있다. 아래 코드는 이러한 조절을 염두에 두고 있다. p_m은 돌연변이 확률이다.

돌연변이 :
1. n_0와 n_1을 각각 염색체에 있는 0-bit와 1-bit의 개수라 하자.
2. $p_1=p_m$; $p_0=p_m \cdot n_1/n_0$;
3. **for** (each gene g in the chromosome)
4. [0,1] 사이의 임의 수 r을 생성하라.
5. **if**(g=1 and $r<p_1$) g를 0으로 변환 ;
   else if(g=0 and $r<p_0$) g를 1로 변환 ;

2.5 매개변수

체계적인 매개변수 최적화 과정은 시도하지 않았으나, 5장의 실험에서 다음과 같은 설정을 모든 데이타 집합에 동일하게 적용하였다. 특정 데이타 집합에 적합하도록 값들을 조절하면 향상된 성능을 가져올 수 있다.

파라메터 설정 :
해 집단 크기 = 20
p_c (교배 확률) = 1.0 (항상 적용)
p_m (돌연변이 확률) = 0.1
q (순위 기반 선택에서) = 0.25
w (벌점 계수) = 0.5

3. 혼합형 유전 알고리즘

유전 알고리즘은 교배와 돌연변이 연산에 의해 지역 최적점(local optimum)에 빠지는 것을 극복할 수 있고, 선택압이 적절할 때 넓은 범위의 공간을 효과적으로 탐색한다. 하지만 지역 최적점 근처에서의 미세 조정력은 약하며, 따라서 많은 실행 시간을 필요로 한다. 단순 유전 알고리즘의 미세 조정력을 향상시키기 위해, TSP 문제[13], 그래프 분할 문제[14], 그리고 영상 압축[15] 등을 포함한 많은 응용에서 혼합형 유전 알고리즘이 개발되었다. 혼합형 유전 알고리즘은 염색체를 적절한 지역 탐색 연산을 이용하여 약간 개선하여 해 집단에 넣는다.

우리는 특징 선택을 위한 혼합형 유전 알고리즘을 제안한다. 기본 아이디어는 유전 알고리즘에 지역 탐색 연산을 삽입하는 것이다. 이러한 아이디어를 포함한 안정형(steady-state) 유전 알고리즘은 다음과 같다.

```
HGA()
{
        initialize population P ;
        repeat {
           select two parents p1 and p2 from P ;
           offspring = crossover(p1,p2) ;
           mutation(offspring) ;

           local_improvement(offspring) ;

           replace(P, offspring) ;
        } until (stopping condition) ;
}
```

교배와 돌연변이 연산을 거친 자손은 부모의 좋은 형질을 포함할 가능성이 높으며, 부모보다 우수할 수도 열세할 수도 있다. 제안한 혼합형 유전 알고리즘에서는 자손 염색체는 대치 직전에 지역 탐색 연산에 의해 성능을 개선시킬 기회를 갖는다. 이러한 아이디어를 실현하는데 중요한 점은 지역적 개선을 위한 적절한 연산을 정의하는 것이다.

기존의 순차 탐색 알고리즘들이 지역 최적 해를 찾아가는데 사용하는 기본 연산들을 소개한다. 연산을 기술할 때 X와 Y의 크기는 암시적으로 명확하므로 표시하지 않는다.

기본적인 지역 탐색 연산 :

rem: X에서 가장 의미 없는 특징 x를 찾아 (x=argmax$_{a\in X}$J($X-\{a\}$)) Y로 옮긴다.

add: Y에서 가장 의미 있는 특징 y를 찾아 (y=argmax$_{a\in Y}$J($X\cup\{a\}$)) X로 옮긴다.

REM(k): rem 연산을 k 번 반복한다.

ADD(k): add 연산을 k 번 반복한다.

혼합형 유전 알고리즘을 위한 지역 탐색 연산 :

ripple_rem(r) ≡ {REM(r) ; ADD($r-1$) ;}, $r\geq 1$

ripple_add(r) ≡ {ADD(r) ; REM($r-1$) ;}, $r\geq 1$

위의 연산들은 X에서 가장 의미 없는 특징을 제거하거나 가장 의미 있는 특징을 추가함으로써 현재의 해 주위에서 지역 탐색을 수행한다. 이들을 적당한 순서로 적용함으로써 염색체의 지역적 개선을 얻을 수 있다. 또 다른 효과로서 연산 적용 순서를 적절하게 구성하여 X를 원하는 크기의 집합으로 만들 수 있다. 이러한 특징 제거나 추가를 수선(repairing) 연산자로 간주할 수 있다.

염색체 C의 지역적 개선은 함수 local_improvement (C)로 수행한다. 먼저 염색체 C를 두 개의 특징 집합 X와 Y로 변환한다. 위에 기술한 기본적인 지역 탐색 연산을 바탕으로 또 다른 두 가지 지역 탐색 연산을 정의한다. ripple_rem(r)은 가장 의미 없는 특징 제거를 r 번 수행하고 가장 의미 있는 특징 추가를 r-1번 수행한다. 따라서 전체적으로는 한 개의 특징을 제거하는 효과가 있다. 이와 비슷하게 ripple_add(r)에서는 X의 크기가 1만큼 증가한다. 여기서 사용하는 변수 r을 물결 인자(ripple factor)라 한다. 그림 2에서 지역 탐색 연산의 두 가지 예를 보여준다.

```
local_improvement(C) {   // C : 염색체
   transform C into X and Y ;

   switch {
      case |X|=d : ripple_rem(r) ; ripple_add(r) ;
      case |X|<d : repeat ripple_add(r) d-|X| times ;
      case |X|>d : repeat ripple_rem(r) |X|-d times ;
   }

   transform X and Y into C ;
}
```

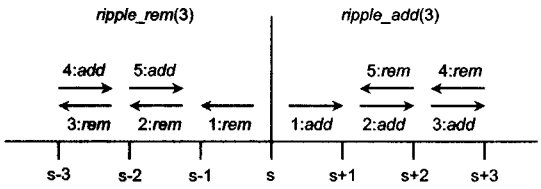

그림 2 ripple_rem(3)과 ripple_add(3)에 의한 add와 rem의 수행 순서

X의 크기와 우리가 원하는 부분 집합 크기 d에 따라, 세가지 경우를 다르게 고려해야 한다.

(1) 크기 요구가 만족될 때 : ripple_rem(r)과 ripple_add(r)를 한번씩 수행하여 X를 개선하고 크기를 그대로 유지한다.

(2) X의 특징 개수가 부족할 때 : ripple_add(r)을 d-|X| 번 적용하여 X를 개선함과 동시에 크기를 키운다.

(3) X의 특징 개수가 과잉일 때 : ripple_rem(r)을 |X|-d 번 적용하여 X를 개선함과 동시에 크기를 줄인다.

물결 인자 r은 개선의 정도를 조절하는데 사용한다. ripple_rem(r)에 의해 줄어드는 특징 개수는 r값에 상관없이 1이지만, r은 실제 실행되는 rem과 add연산 횟수에 직접적인 영향을 미친다. r이 클수록 지역적 개선 정도가 더 크다. 예를 들어, ripple_rem(2)는 두 번의 rem과 한 번의 add로 이루어지며, 한 번의 rem만을 수행하는 ripple_rem(1)보다 개선 정도가 강하다.

혼합형 유전 알고리즘은 두 가지 유용한 효과를 제공한다. 첫째, 염색체의 지역적 개선을 통해 최종 성능을 상당히 향상시킬 수 있다. 둘째, 염색체가 우리가 원하는 개수의 특징을 포함하도록 조절하는 기능을 제공한다. 물결 인자 값이 클수록 한 세대를 거치는데 더 많은

시간이 소요되지만, 수렴하는데 보다 적은 세대를 필요로 한다.

4. 실험과 토론

4.1 실험 환경

표 1은 실험에 사용한 11개의 데이타 집합을 보여준다. 앞의 열 개의 데이타 집합은 특징 개수가 열 개 이상이고 누락 값을 갖는 특징이 없으며 모든 특징이 수치값을 갖는 조건 하에서 UCI repository [16]에서 선택하였다. 마지막 데이타 집합은 CENPARMI 필기 숫자 데이타베이스의 샘플에서 메쉬 특징을 추출하여 얻었다. 메쉬 특징은 샘플 영상을 10*10 크기로 정규화한 후, 각 화소의 값을 특징 값으로 취하였다. 이들 데이타 집합은 패턴 인식 분야에서 널리 사용되는 것들이며, 이들은 문자 인식, 음성 인식, 물체 인식, 그리고 의료 진단 분야를 포함하여 다양하다. 또한 특징 벡터의 크기(즉 D)가 10부터 100까지 고루 분포하고 있다.

평가를 위해 분류 정확도를 이용하였는데, 기각을 허용하지 않는 1-NN 분류기의 인식률로 측정하였다. 총 여덟 개의 알고리즘을 평가하였다. 이들은 세가지 순차 탐색 알고리즘, SFS(sequential forward search), PTA (l=2,r=1) (plus-l-and-take-away-r), SFFS(sequential forward floating search)를 포함하며, SGA (단순 유전 알고리즘)와 다른 물결 인자를 갖는 네 개의 HGA (혼합형 유전 알고리즘)이다. 혼합형 유전 알고리즘은 물결 인자 r을 갖는 HGA(r)로 표시한다.

유전 알고리즘 간의 공정한 평가를 위해, 이들이 비슷한 시간을 사용하도록 멈춤 조건의 최대 세대 수 T를 조절하였다. 데이타 집합 각각에 대해, 네 가지 d 값에 대한 인식률을 측정하였다. 모든 유전 알고리즘은 다섯 번 독립적으로 수행하였으며, 이에 대한 평균과 최대 성능을 제시한다.

4.2 성능 분석

표 2는 11 개의 데이타 집합에 대한 여덟 개 알고리즘의 성능을 비교해서 보여준다. 각 데이타 집합에 대해 네 가지 d 값, 즉 D/5, 2D/5, 3D/5, 그리고 4D/5에 대해 성능 측정을 수행하였다. D가 작은 데이타 집합에 대해서는 전체 (exhaustive) 탐색으로 최적해를 찾아 제시하였다. 유전 알고리즘에서는 다섯 번 실행에 대한 평균 인식률(x)과 최고 인식률(y)을 x(y)로 제시한다. 여덟 개 알고리즘 중에 가장 좋은 해를 찾은 알고리즘에 대해서는 알아보기 쉽도록 굵은 글씨체로 표시하였다.

우선 다섯 개의 유전 알고리즘을 비교함으로써 혼합 유전 알고리즘의 효과를 분석한다. 편의상 [5]에서 분류한 대로, $0<D\leq19$의 작은 문제, $20\leq D\leq49$의 중간 문제, 그리고 $50\leq D$의 대용량 문제로 나누어 생각해 보자. 작은 문제의 경우, Vehicle 데이타에 대해 d=11일 때의 SGA를 제외하고 모든 경우 최적해를 찾았다. HGA는 모든 경우 최적해를 찾았다. 중간 크기의 데이타 집합 (WDBC, Ionosphere, 그리고 Satellite)에서는 과다한 계산 시간으로 최적해를 제시하지 못했다. 찾은 해 중에서는 HGA가 항상 가장 좋은 해를 제공하였으며, SGA도 몇 가지 경우에서 같은 해를 찾아내었다.

대용량 문제(Sonar와 Numeral)에서는 모든 경우 HGA가 SGA를 능가하였다. HGA($r\geq2$)가 HGA(1)에 비해 우수함을 알 수 있다. Sonar에서는 HGA($r\geq2$)는

표 1 실험에 사용한 데이타 집합

	샘플 개수	특징 개수	부류 개수	평가 방법	
				분류기	x in leave-x-out
Glass*	214	10	7	1-NN	1
Vowel*	990	10	11	1-NN	1
Wine*	178	13	3	1-NN	1
Letter*	20000	16	26	1-NN	5000
Vehicle*	846	18	4	1-NN	1
Segmentation*	210/2100***	19	7	1-NN	-
WDBC*	569	30	2	1-NN	1
Ionosphere*	351	34	2	1-NN	1
Satellite*	4435/2000***	36	6	1-NN	-
Sonar*	208	60	2	1-NN	1
Numerals**	4000/2000***	100	10	1-NN	-

* UCI repository [16]에서 다운로드 하였음.

** 메쉬 특징을 CENPARMI 필기 숫자에서 추출하였음.

*** x/y는 x개의 훈련 샘플과 y개의 테스트 샘플을 나타냄.

표 2 11개 데이타 집합에 대한 인식률 (단위: %)

Datasets	d*	Optimum	SFS	PTA	SFFS	SGA	HGA(1)	HGA(2)	HGA(3)	HGA(4)
Glass (*D*=10)	2	**99.07**	**99.07**	**99.07**	**99.07**	**99.07**(99.07)	**99.07**(99.07)	**99.07**(99.07)	NA	NA
	4	**100**	**100**	**100**	**100**	**100**(100)	**100**(100)	**100**(100)	**100**(100)	**100**(100)
	6	**100**	**100**	**100**	**100**	**100**(100)	**100**(100)	**100**(100)	**100**(100)	**100**(100)
	8	**100**	**100**	**100**	**100**	**100**(100)	**100**(100)	**100**(100)	**100**(100)	NA
Vowel (*D*=10)	2	**62.02**	**62.02**	**62.02**	**62.02**	**62.02**(62.02)	**62.02**(62.02)	**62.02**(62.02)	NA	NA
	4	**92.83**	92.63	**92.83**	**92.83**	**92.83**(92.83)	**92.83**(92.83)	**92.83**(92.83)	**92.83**(92.83)	**92.83**(92.83)
	6	**98.79**	98.28	**98.79**	**98.79**	**98.79**(98.79)	**98.79**(98.79)	**98.79**(98.79)	**98.79**(98.79)	**98.79**(98.79)
	8	**99.70**	**99.70**	**99.70**	**99.70**	**99.70**(99.70)	**99.70**(99.70)	**99.70**(99.70)	**99.70**(99.70)	NA
Wine (*D*=13)	3	**93.82**	**93.82**	**93.82**	**93.82**	**93.82**(93.82)	**93.82**(93.82)	**93.82**(93.82)	**93.82**(93.82)	NA
	5	**95.51**	94.38	94.38	94.94	**95.51**(95.51)	**95.51**(95.51)	**95.51**(95.51)	**95.51**(95.51)	**95.51**(95.51)
	8	**95.51**	**95.51**	**95.51**	**95.51**	**95.51**(95.51)	**95.51**(95.51)	**95.51**(95.51)	**95.51**(95.51)	**95.51**(95.51)
	10	**92.70**	92.13	92.13	**92.70**	**92.70**(92.70)	**92.70**(92.70)	**92.70**(92.70)	**92.70**(92.70)	**92.70**(92.70)
Letter (*D*=16)	3	**47.09**	**47.09**	**47.09**	**47.09**	**47.09**(47.09)	**47.09**(47.09)	**47.09**(47.09)	**47.09**(47.09)	NA
	6	**87.60**	86.20	**87.60**	**87.60**	**87.60**(87.60)	**87.60**(87.60)	**87.60**(87.60)	**87.60**(87.60)	**87.60**(87.60)
	10	**96.35**	96.12	**96.35**	**96.35**	**96.35**(96.35)	**96.35**(96.35)	**96.35**(96.35)	**96.35**(96.35)	**96.35**(96.35)
	13	**96.42**	**96.42**	**96.42**	**96.42**	**96.42**(96.42)	**96.42**(96.42)	**96.42**(96.42)	**96.42**(96.42)	**96.42**(96.42)
Vehicle (*D*=18)	4	**69.74**	62.77	64.78	69.15	69.50(69.74)	**69.74**(69.74)	**69.74**(69.74)	69.62(69.74)	69.39(69.74)
	7	**73.52**	69.15	70.09	**73.52**	72.97(73.52)	**73.52**(73.52)	**73.52**(73.52)	**73.52**(73.52)	**73.52**(73.52)
	11	**72.46**	69.50	71.75	71.87	71.84(71.87)	**72.46**(72.46)	**72.46**(72.46)	**72.46**(72.46)	72.29(72.46)
	14	**70.80**	68.20	**70.80**	**70.80**	**70.80**(70.80)	**70.80**(70.80)	**70.80**(70.80)	**70.80**(70.80)	**70.80**(70.80)
Segmen-tation (*D*=19)	4	**92.81**	**92.81**	**92.81**	**92.81**	**92.81**(92.81)	**92.81**(92.81)	**92.81**(92.81)	**92.81**(92.81)	**92.81**(92.81)
	8	**92.95**	**92.95**	**92.95**	**92.95**	**92.95**(92.95)	**92.95**(92.95)	**92.95**(92.95)	**92.95**(92.95)	**92.95**(92.95)
	11	**92.95**	**92.95**	**92.95**	**92.95**	**92.95**(92.95)	**92.95**(92.95)	**92.95**(92.95)	**92.95**(92.95)	**92.95**(92.95)
	15	**92.57**	**92.57**	**92.57**	**92.57**	**92.57**(92.57)	**92.57**(92.57)	**92.57**(92.57)	**92.57**(92.57)	**92.57**(92.57)
WDBC (*D*=30)	6	**94.90**	93.15	93.15	94.20	93.67(93.67)	93.92(94.90)	**94.38**(94.90)	93.99(94.20)	93.99(94.20)
	12	NA	92.62	92.97	94.20	93.95(94.38)	94.06(94.38)	94.06(94.38)	94.06(94.38)	**94.27**(94.38)
	18	NA	94.02	**94.20**	**94.20**	93.85(93.85)	93.92(94.20)	93.99(94.20)	94.13(94.20)	93.99(94.20)
	24	NA	92.44	93.50	**93.85**	**93.85**(93.85)	**93.85**(93.85)	**93.85**(93.85)	**93.85**(93.85)	**93.85**(93.85)
Ionosphere (*D*=34)	7	NA	93.45	93.45	93.45	94.70(95.44)	95.38(95.73)	95.50(95.73)	**95.56**(95.73)	95.50(95.73)
	14	NA	90.88	92.59	93.73	94.30(94.87)	94.93(95.73)	**95.56**(95.73)	95.21(95.73)	95.21(95.73)
	20	NA	90.03	92.02	92.88	93.79(94.30)	93.90(94.30)	**94.19**(94.30)	93.73(94.02)	94.13(94.30)
	27	NA	89.17	91.17	90.88	91.17(91.45)	**91.45**(91.45)	**91.45**(91.45)	**91.45**(91.45)	**91.45**(91.45)
Satellite (*D*=36)	7	NA	86.85	88.20	**88.55**	87.89(88.10)	88.25(88.55)	88.44(88.55)	**88.55**(88.55)	88.46(88.55)
	14	NA	89.45	89.85	90.10	90.61(90.85)	**90.88**(90.95)	90.87(91.00)	90.80(90.95)	90.82(90.95)
	22	NA	90.45	91.10	**91.45**	91.36(91.45)	91.37(91.45)	91.44(91.45)	**91.45**(91.45)	91.41(91.45)
	29	NA	90.40	90.70	90.95	91.10(91.25)	91.21(91.25)	**91.24**(91.25)	**91.24**(91.25)	91.18(91.25)
Sonar (*D*=60)	12	NA	87.02	89.42	92.31	92.40(93.75)	93.65(94.71)	94.71(95.67)	94.61(95.19)	**94.81**(95.67)
	24	NA	89.90	90.87	93.75	95.49(95.67)	95.86(96.63)	95.96(96.63)	**96.34**(97.12)	96.15(97.12)
	36	NA	88.46	91.83	93.27	95.09(95.67)	95.67(96.15)	**95.82**(96.15)	95.67(96.15)	95.67(96.15)
	48	NA	91.82	92.31	91.35	92.02(92.79)	92.60(92.79)	**93.17**(93.27)	**93.17**(93.27)	93.08(93.27)
Numeral (*D*=100)	20	NA	86.05	89.05	89.45	88.86(89.70)	89.75(90.20)	90.05(90.20)	90.16(90.60)	**90.18**(90.50)
	40	NA	94.40	94.90	95.15	94.82(95.00)	95.03(95.10)	95.38(95.75)	**95.74**(96.00)	95.53(95.80)
	60	NA	95.50	96.05	96.05	96.05(96.20)	96.39(96.55)	96.53(96.70)	**96.62**(96.70)	96.57(96.65)
	80	NA	95.55	95.80	95.80	96.09(96.30)	96.18(96.35)	**96.41**(96.50)	96.34(96.40)	96.38(96.55)

* D의 1/5, 2/5, 3/5, 그리고 4/5의 네 개 값 사용.

** 유전 알고리즘의 x(y)에서 x와 y는 각각 다섯 번 수행의 평균과 최대 인식률을 나타냄.

*** 굵은 글씨체는 각 행에서 가장 좋은 해를 나타냄.

HGA(1)보다 약 0.5~1.0% 정도 좋은 결과를 보였다. 또한 평균과 최고의 차이 측면에서 HGA($r \geq 2$)는 HGA(1)보다 편차가 적었다. 이는 HGA($r \geq 2$)이 HGA(1)보다 해의 품질과 안정성 면에서 장점이 있음을 뜻한다.

세가지 순차 탐색 알고리즘을 비교해 보면, SFFS는 몇 가지 예외를 제외하고 전체적으로 가장 좋은 해를 생성하였다. PTA는 항상 SFS를 능가하였으며, SFFS는 Ionosphere에서 d=27인 경우와 Sonar에서 d=48인 경우를 제외하고 PTA의 성능을 능가하였다. 작은 문제인 Glass와 Segmentation에 대해 SFS도 최적 해를 생성하였다. 하지만 문제 크기가 커짐에 따라 세 알고리즘 간의 성능 차이가 커짐을 알 수 있다. Sonar 예에서, SFFS는 PTA보다 약 1.5~3% 나은 해를 생성하였으며, PTA는 SFS보다 약 0.5~3.5% 정도 나음을 알 수 있다.

이제 순차 탐색 알고리즘과 유전 알고리즘을 비교해 보자. SFFS가 PTA와 SFS를 능가하므로 SFFS만을 고려한다. 다섯 번 수행한 결과의 평균 성능을 보면 SGA는 SFFS와 비슷한 해를 보인다. 하지만 최고 성능을 고려하면, SGA가 SFFS보다 우수하다. HGA는 SGA를 크게 향상시켰으며, HGA는 모든 경우에 SFFS보다 우수하다.

위의 실험 결과와 분석에 따라 다음 네 가지 결론을 이끌어 내었다. 이 결론은 넓은 범위의 문제 크기를 갖는 표준 데이타 집합으로부터 도출한 것이다.

(1) 순차 탐색 알고리즘 중에는 SFFS가 최상이다.
(2) SGA와 HGA를 포함하여 유전 알고리즘은 문제 크기에 관계없이 SFFS보다 우수하다.
(3) HGA는 SGA보다 우수하다. HGA에서 물결 인자 r ≥2를 사용하는 것이 좋다.
(4) HGA는 대용량 크기 문제에서 보다 효과적이다.

5. 결 론

특징 선택을 위한 새로운 혼합형 유전 알고리즘을 제안하였다. 물결 인자를 매개변수로 갖는 지역 탐색 연산을 고안하였으며, 이를 유전 알고리즘에 삽입하여 혼합 유전 알고리즘을 개발하였다. 또한 구현 상에서 차이를 최소화하기 위해 구체적인 알고리즘 명세와 실험에서 사용한 각종 매개변수 설정을 제공하였다. 다양한 데이타 집합에 대한 실험 결과로 제안한 알고리즘의 우수성을 입증하였다. 앞으로 염색체 표현을 위한 유전자 재배열 기법이나 보다 적절한 유전 연산자 개발을 통해 추가적인 성능 향상을 모색할 예정이다.

참 고 문 헌

[1] J. Kittler, "Feature selection and extraction," in *Handbook of Pattern Recognition and Image Processing*, Academic Press (Edited by T.Y. Young and K.S. Fu), pp.59-83, 1986.

[2] W. Siedlecki and J. Sklansky, "On automatic feature selection," *International Journal of Pattern Recognition and Artificial Intelligence*, Vol2. No.2, pp.197-220, 1988.

[3] F.J. Ferri, P. Pudil, M. Hatef, and J. Kittler, "Comparative study of techniques for large-scale feature selection," in *Pattern Recognition in Practice IV* (Edited by E.S. Gelsema and L.N. Kanal), Elsevier Science, pp.403-413, 1994.

[4] A. Jain and D. Zongker, "Feature selection: evaluation, application, and small sample performance," *IEEE Tr. PAMI*, Vol.19, No.2, pp.153-158, February 1997.

[5] M. Kudo and J. Sklansky, "Comparison of algorithms that select features for pattern recognition," *Pattern Recognition*, Vol.33, No.1, pp.25-41, 2000.

[6] J. Holland, *Adaptation in Nature and Artificial Systems*, MIT Press, 1992.

[7] W. Siedlecki and J. Sklansky, "A note on genetic algorithms for large-scale feature selection," *Pattern Recognition Letters*, Vol.10, pp.335-347, 1989.

[8] F.Z. Brill, D.E. Brown, and W.N. Martin, "Fast genetic selection of features for neural network classifiers," *IEEE Tr. Neural Networks*, Vol.3, No.2, pp.324-328, March 1992.

[9] J.H. Yang and V. Honavar, "Feature subset selection using a genetic algorithm," *IEEE Intelligent Systems*, Vol.13, No2, pp.44-49, 1998.

[10] L.I. Kuncheva and L.C. Jain, "Nearest neighbor classifier: simultaneous editing and feature selection," *Pattern Recognition Letters*, Vol.20, pp.1149-1156, 1999.

[11] M.L. Raymer, W.F. Punch, E.D. Goodman, L.A. Kuhn, and A.K. Jain, "Dimensionality reduction using genetic algorithms," *IEEE Tr. Evolutionary Computation*, Vol.4, No.2, pp.164-171, July 2000.

[12] P. Pudil, J. Novovicova, and J. Kittler, "Floating search methods in feature selection," *Pattern Recognition Letters*, Vol.15, pp.1119-1125, 1994.

[13] P. Jog, J. Suh, and D. Gucht, "The effect of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem," *Proc. of International Conference on Genetic Algorithms*, pp.110-115, 1989.

[14] T.N. Bui and B.R. Moon, "Genetic algorithm and graph partitioning," *IEEE Tr. Computers*, Vol.45, No.7, pp.841-855, July 1996.

[15] X. Zheng, B.A. Julstrom, and W. Cheng, "Design of vector quantization codebooks using a genetic algorithm," *Proc. of IEEE International Conf. on Evolutionary Computation*, pp.525-529, 1997.

[16] P.M. Murphy and D.W. Aha, "UCI repository for

machine learning databases(http://www.ics.uci.edu/~mlearn/MLRepository.html)," Irvine, CA: University of California, Department of Information and Computer Science, 1994.

오 일 석

정보과학회논문지 : 소프트웨어 및 응용
제 31 권 제 2 호 참조

이 진 선

1985년 전북대학교 전산통계학과 학사. 1988년 전북대학교 전산통계학과 석사. 1988년~1992년 한국전자통신연구원 연구원. 1995년 전북대학교 컴퓨터공학과 박사. 1995년~현재 우석대학교 컴퓨터공학과 교수. 관심분야는 패턴인식, 영상처리, 멀티미디어

문 병 로

1985년 서울대학교 계산통계학과, 학사 1987년 KAIST 전산학과, 석사. 1994년 펜실바니아 주립대, 박사. 1987년~1991년 (주)LG전자 중앙연구소, 선임연구원 1994년~1995년 UCLA VLSI CAD Lab, 박사후연구원. 1996년~1997년 8월 (주)LG반도체 DT연구소, 책임연구원. 1997년 9월~현재 서울대학교 컴퓨터공학부, 부교수. 관심분야는 최적화, 유전알고리즘, 알고리즘 디자인 및 분석, 스케줄링 등