유도된 이진난수 생성법을 이용한 uDEAS의 Multi-start 성능 개선

논 문
58-4-26

Performance Improvement of Multi-Start in uDEAS Using Guided Random Bit Generation

김 은 숙* · 김 만 석* · 김 종 욱†
(Eunsu Kim · Manseak Kim · Jong-Wook Kim)

Abstract - This paper proposes a new multi-start scheme that generates guided random bits in selecting initial search points for global optimization with univariate dynamic encoding algorithm for searches (uDEAS). The proposed method counts the number of 1 in each bit position from all the previously generated initial search matrices and, based on this information, generates 0 in proportion with the probability of selecting 1. This rule is simple and effective for improving diversity of initial search points. The performance improvement of the proposed multi-start is validated through implementation in uDEAS and function optimization experiments.

Key Words : Global optimization, Heuristic algorithm, Multi-start, DEAS, uDEAS

1. 서 론

시간 제어와 식별(identification)은 산업 기술이 고도화되고 각종 네트워크와 시스템이 복잡해질수록 그 중요성이 날로 증대하고 있다. 그러나 비용함수의 미분 정보가 필요한 기존의 해석적 최적화기법(analytical optimization method)으로는 실시간 제어/식별은 거의 불가능하다고 할 수 있다. 이에 대한 해결책으로 컴퓨터 프로그램을 이용한 연산적 최적화 기법(computational optimization method)에 대한 연구가 1990년대 이후로 활발히 이루어지고 있으며 그 대표적인 예는 genetic algorithm(GA)[1], ant colony optimization (ACO)[2], particle swarm optimization(PSO)[3], generating set search(GSS)[4], mesh adaptive direct search(MADS)[5], dynamic encoding algorithm for searches(DEAS)[6]-[11] 등이다. 앞에서 GA, ACO, PSO는 집단기반(population-based) 전역최적화 알고리듬의 범주에 속하고, GSS와 MADS는 지역탐색(local search) 알고리듬, DEAS는 지역탐색 기반 전역최적화 알고리듬에 해당된다.

집단기반 최적화 알고리듬은 여러 탐색점을 탐색영역에 산포시킨 후 주어진 규칙을 이용해서 점차적으로 전체 집단을 전역해(global minimum)를 향해 수렴하게 하는 탐색 원리를 가지고 있다. 그러므로 하나의 집단 전체를 매번 갱신하는데 있어 탐색 시간이 개체수에 비례해서 증가한다. 이 때문에 연산 프로세서의 속도가 상대적으로 느린 저사양 PC나 임베디드 시스템에서 이러한 알고리듬을 구현할 경우 실시간에 적용하기에는 한계가 있다. 지역탐색 알고리듬은 이와는 달리 임의의 단일 탐색점으로부터 주변의 더 나은 해로 순차적으로 이동하는 방식이기 때문에 탐색 시간은 상대적으로 짧다고 할 수 있다. 그러나 초기 탐색점에 따라 수렴하는 지역해(local minimum)가 다르기 때문에 가급적 전역해 주변에서 탐색을 시작해야 하는 단점이 존재한다. 이러한 지역탐색 알고리듬의 단점을 보완하기 위해 가장 많이 사용되는 전역최적화 알고리듬이 multi-start 기법인데, 무작위 혹은 정해진 규칙에 의해 추출한 초기 탐색점으로부터 반복적으로 지역탐색을 수행하여 얻은 지역해로부터 가장 좋은 것을 전역해로 간주하는 알고리듬이다.

DEAS는 메타휴리스틱 알고리듬의 일종으로서, 이진 행렬을 이용해서 규칙적으로 생성되는 이웃 점의 비용함수 값들을 순차적으로 비교함으로써 지역해의 근사값을 찾고, 이를 임의의 시작점으로부터 반복함으로써 최종적으로 전역해를 찾는 원리로 전역최적화를 수행한다[6]-[11]. DEAS는 GA[1]처럼 이진 스트링으로 실수 해를 표현하지만 다변수 문제의 경우 GA와는 달리 이진 행렬로써 실수 벡터를 표현하며 각 행은 실수 벡터의 각 원소를 나타낸다. GA와 또 한가지 다른 점은 DEAS는 탐색 정밀도를 높이기 위해 이진 행렬의 각 행 최하위비트(least significant bit, LSB) 오른쪽에 계속적으로 이진 비트를 삽입함으로써 점차적으로 행의 길이가 길어진다는 점이다.

uDEAS(univariate DEAS)는 DEAS의 계열 알고리듬으로서 지역탐색 시 이진 행렬에서 한 번에 한 행씩 각 행의 LSB에 0 또는 1을 추가하고 연이어 증가/감소 연산을 수행함으로써, 한 번의 지역해 천이(transition)를 위해 각 변수당 최대 2번씩 비용함수를 계산한다[9]-[11]. 그러므로 n개의 변수 전체에 대해 행별 연산이 끝나면 약 $2n$번의 비용함수 계산 횟수가 소요된다. uDEAS는 탐색 변수의 수가 최대 30인 테스트 함수 최적화[9],[10]와 13개의 유도전동기 파라미터 추정[11]에서 탁월한 탐색 속도와 정확도를 보였다. 그러나 지금까지 uDEAS에는 전역최적화를 위해 등분포 확률

* 준 회 원 : 동아대학 전자공학과 석사과정
† 교신저자, 정회원 : 동아대학 전자공학과 조교수 · 공박
E-mail : kjwook@dau.ac.kr
접수일자 : 2009년 3월 5일
최종완료 : 2009년 3월 24일

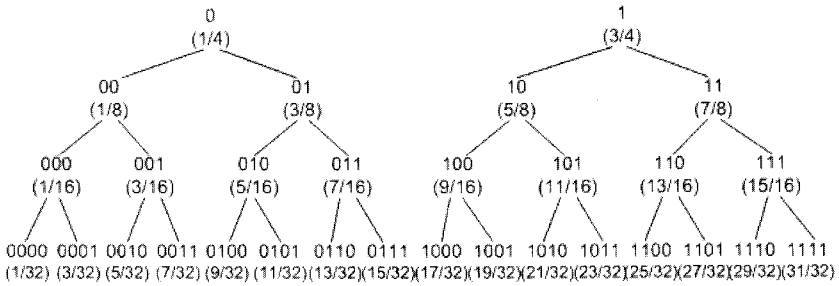


그림 1 uDEAS에서 유전자형으로 사용하는 이진 스트링 트리 구조(괄호 안의 값은 0과 1사이에서 복호화된 실수값)
Fig. 1 Tree structure of binary strings used as genotypes in uDEAS (real numbers are decoded values)

값(uniform distribution random number)을 이용한 단순한 multi-start 기법이 사용되어 왔다.

본 논문에서는 multi-start 기법에 있어서 새로운 시작점의 각 비트들을 랜덤하게 정할 때 현재까지 생성된 시작점들의 비트별 정보를 이용하여 최대한 산포도를 증가시키는 기법을 새롭게 제안하고, 이를 uDEAS에 적용하여 잘 알려진 테스트 함수에 대해 적용함으로써 전역최적화 성능을 향상시킬 수 있음을 보인다.

2. uDEAS

uDEAS는 크게 지역탐색과 전역탐색 루틴으로 나뉜다. 지역탐색은 탐색점 주변을 한 변수에 대해 양분해서 좀 더 면밀히 탐색하는 bisectional search(BSS)와 BSS에서 얻은 정보를 기반으로 개선된 해가 존재할 가능성이 높은 영역을 확장 탐색하는 unidirectional search(UDS)로 구성된다. uDEAS에서는 한 변수에 한번씩 BSS-UDS 조합을 수행하고 모든 변수에 대해 이를 완료한 것을 session으로 정의하고, session이 진행됨에 따라 이진 행렬의 열의 길이가 1씩 증가한다. 그러므로 변수가 n개인 최적화 문제에 대해 탐색 시작점이 $n\times m$ 행렬로 표현되고, k번의 session이 지역탐색에 의해 수행되었다면 결과적으로 $n\times(m+k)$ 이진행렬이 현재 지역해로 표현된다.

전역탐색을 위해서 지금까지 uDEAS는 이진행렬에 대해 가장 용이하면서도 간단하게 구현할 수 있는 multi-start 기법을 채용해 왔다. Multi-start 기법은 시작점을 무작위로 정하고 그 시작점으로부터 지역최적화 기법을 수행해서 지역해를 찾은 후, 이 작업을 다시 임의의 시작점으로부터 주어진 횟수만큼 재시작(restart) 함으로써 최종적으로 찾은 지역해 중 가장 좋은 지역해를 전역해로 간주하는 전역최적화 기법이다.

본 절에서는 uDEAS의 지역탐색과 전역탐색 전략에 대해 간략히 설명한다.

2.1 지역탐색

uDEAS는 지역탐색을 위해 변수 별로 BSS와 UDS를 수행한다. 이는 exhaustive DEAS(eDEAS)[6]-[8]가 전체 n개의 변수에 대해 BSS 2^n개의 이웃점 행렬을 생성하여 비교 탐색 후 그 중 하나를 골라 그 방향으로 2^n-1번의 UDS를 수행하는 것과는 크게 다르다고 할 수 있다.

BSS는 하나의 변수를 나타내는 이진 스트링의 LSB에 0 또는 1을 붙임으로써 현재 탐색점 주변의 이진 스트링을 생성한다. 최근에 개선되어 사용되고 있는 복호화 함수를 이용하면 0을 붙인 경우는 그 실수 값이 동일한 폭만큼 감소하고, 1을 붙인 경우는 그 실수 값이 동일한 폭만큼 증가하는 특성을 갖는다[10],[11]. 그림 1은 uDEAS에서 사용하는 이진 스트링 구조와 해당 실수값 들을 트리 구조로 나타낸 것으로 0과 1을 LSB에 붙였을 때 해당 스트링의 실수 값들이 감소하고 증가하는 것을 확인할 수 있다. 그리고 0 또는 1을 붙이기 전의 스트링의 길이가 m일 때, 이에 대해 BSS 수행 시 탐색 폭(step length)은 $\epsilon=\dfrac{u-l}{2^{m+2}}$과 같다[11]. 여기에서 u와 l은 해당 변수 탐색범위의 최대값과 최소값이다. BSS는 이진 스트링이 1 길어질수록 탐색 폭이 반감하므로 빨리 수렴하는 장점이 있지만, 주변을 넓게 탐색할 수 없는 단점이 있다. 이를 보완하기 위해 스트링 길이는 유지한 채 1씩 더하거나 빼면서 변수를 증가와 감소시킬 수 있는데 이를 UDS라 한다. UDS의 탐색 폭은 바로 직전에 행해진 BSS의 탐색 폭의 2배가 되며, 비용함수가 감소하는 한 동일한 탐색 폭으로 계속 수행될 수 있다.

이해를 돕기 위해 2개의 변수 x_1, x_2의 최적해를 구하는 2차원 문제를 예로 들어 uDEAS의 지역탐색 원리를 설명하고자 한다. 예를 들어 직전 session의 결과로서 구해진 지역해의 행렬이 다음과 같다고 하자.

$$M=\begin{bmatrix}1&0&1\\0&1&1\end{bmatrix}$$

uDEAS는 행렬 M의 x_1에 대해 먼저 BSS와 UDS를 한다. 즉, BSS를 위해 아래와 같이 2개의 이웃행렬을 생성한다:

$$M_B^{(1)-}=\begin{bmatrix}M&0\end{bmatrix}=\begin{bmatrix}1&0&1&0\\0&1&1&\end{bmatrix},\quad M_B^{(1)+}=\begin{bmatrix}M&1\end{bmatrix}=\begin{bmatrix}1&0&1&1\\0&1&1&\end{bmatrix}.$$

여기에서 윗첨자 중 (1)은 첫 번째 원소 x_1에 대해 BSS-UDS를 행하는 것을 의미하고, '+'와 '-'는 양과 음의 탐색방향을 각각 나타낸다. 그리고 아래첨자 B는 BSS를 의미한다. 그 다음 단계로 비용함수 계산을 위해 이웃행렬의 각 행 별로 복호화하여 실수 벡터를 다음과 같이 구한다.

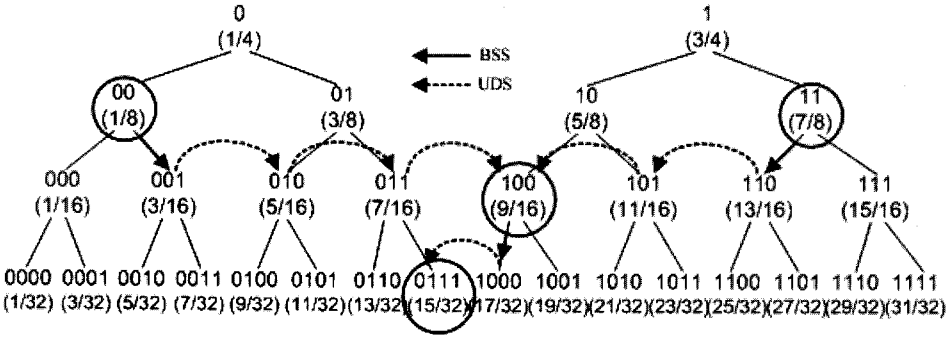


그림 2 서로 다른 스트링으로부터 시작된 uDEAS 지역탐색의 예(1차원 단봉함수)

Fig. 2 Example of uDEAS local searches carried out from different initial strings for a one-dimensional problem

$$X_B^{(1)-}=\begin{bmatrix} f_d([1\,0\,1\,0],x_1^u,x_1^l) \\ f_d([0\,1\,1\ ],x_2^u,x_2^l)\end{bmatrix},\quad X_B^{(1)+}=\begin{bmatrix} f_d([1\,0\,1\,1],x_1^u,x_1^l) \\ f_d([0\,1\,1\ ],x_2^u,x_2^l)\end{bmatrix}.$$

여기에서 x_1^u, x_1^l과 x_2^u, x_2^l은 x_1과 x_2변수 탐색영역의 최대값과 최소값을 각각 의미한다. uDEAS에서는 복호화를 위해 길이가 m인 이진 스트링 $[b_m b_{m-1}\cdots b_1]$을 $[r_l, r_u]$ 범위에서 실수로 변환시키는 다음 함수를 사용한다[10],[11].

$$f_d([b_m b_{m-1}\cdots b_1],r_u,r_l)=r_l+\frac{r_u-r_l}{2^{m+1}}\left(\sum_{j=1}^{m} b_j 2^j+1\right),\quad b_i\in 0,1. \tag{1}$$

이 후 각 변수에 대해 계산된 비용함수 값 $J(X_B^{(1)-})$과 $J(X_B^{(1)+})$를 상호 비교한다. 만약 $J(X_B^{(1)+})<J(X_B^{(1)-})$라면 $M_B^{(1)+}$의 첫 번째 행(x_1)에 대해 양의 방향으로 UDS가 수행된다. 이를 위해 아래와 같이 이웃점이 차례로 생성되며 비용함수 값의 감소여부가 그 때마다 확인된다.

$$X_{U(1)}^{(1)+}=\begin{bmatrix} f_d([1\,1\,0\,0],x_1^u,x_1^l) \\ f_d([0\,1\,1\ ],x_2^u,x_2^l)\end{bmatrix},\quad X_{U(2)}^{(1)+}=\begin{bmatrix} f_d([1\,1\,0\,1],x_1^u,x_1^l) \\ f_d([0\,1\,1\ ],x_2^u,x_2^l)\end{bmatrix},\quad X_{U(3)}^{(1)+}=\begin{bmatrix} f_d([1\,1\,1\,0],x_1^u,x_1^l) \\ f_d([0\,1\,1\ ],x_2^u,x_2^l)\end{bmatrix},\cdots \tag{2}$$

여기에서 아래첨자 U는 UDS를 나타내고 뒤의 숫자는 UDS가 일어난 횟수를 의미한다.

매회의 비용함수 감소 확인에서 $X_U^{(1)*}=X_{U(k)}^{(1)+}$ $\arg\min\{J(X_{U(1)}^{(1)+}),\cdots,J(X_{U(k)}^{(1)+}),J(X_{U(k+1)}^{(1)+})\}$임이 판명되었다면, x_1에 대해서 현 스트링에 대한 BSS와 UDS가 완료된다.

이번에는 구해진 $X_U^{(1)*}$를 기준점으로 삼아 x_2에 대해 BSS를 행한다. 예를 들어 식 (2)에서 UDS의 두 번째 행렬이 최적으로 판명되었다면 $\begin{bmatrix}1\,1\,0\,1\\0\,1\,1\end{bmatrix}$로부터 두 번째 행을 변화시킨 두 개의 이웃행렬이 다음과 같이 생성된다.

$$M_B^{(2)-}=\begin{bmatrix}1\,1\,0\,1\\0\,1\,1\,0\end{bmatrix},\quad M_B^{(2)+}=\begin{bmatrix}1\,1\,0\,1\\0\,1\,1\,1\end{bmatrix}$$

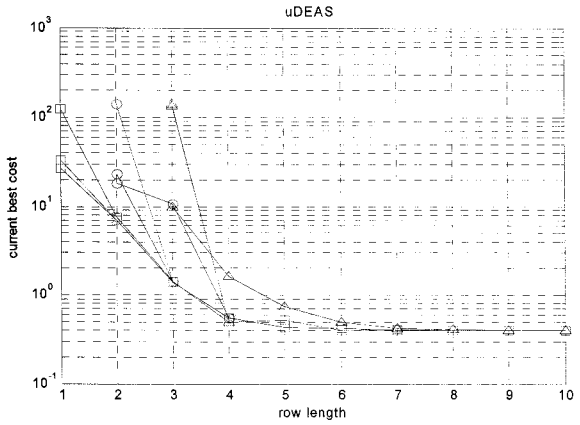
동일한 방법으로 x_2에 대해 UDS를 수행하고 최적의 행렬을 찾아내면 한 번의 session이 완료된다.

상기 예제는 행 길이(row length)가 3인 행렬에 대해 session을 1회 수행한 과정을 보인 것으로, 한 번의 지역탐색은 행 길이가 *finRowLen* 값이 되면 종료된다. 그러므로 초기 행렬의 행 길이가 *initRowLen*라면 총 session 횟수는 *finRowLen-initRowLen*가 된다. 지역탐색의 종료조건으로 이렇게 *finRowLen* 값을 지정해 주는 이유는 행의 길이가 탐색 정밀도(resolution)와 직접적으로 연관되어 있으며 사용자가 원하는 정밀도 이상의 탐색을 수행하지 않게 하기 위함이다.

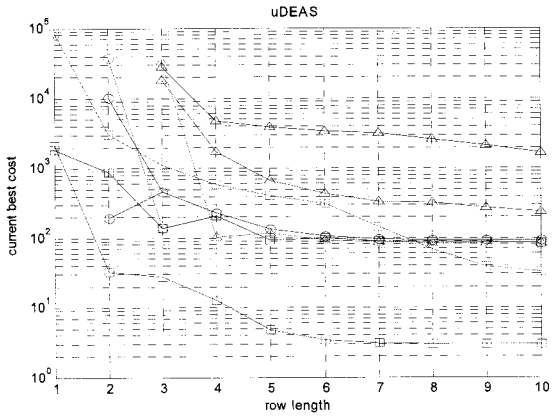
2.2 전역탐색

uDEAS로 지역탐색 시 session이 진행되는 동안 탐색 폭은 지속적으로 반감하며, 탐색 궤적은 비용함수의 형상(landscape)에 좌우된다. 그러므로 탐색 변수의 순서가 바뀌지 않는 한 동일한 행렬로부터 시작된 지역탐색은 동일한 탐색 궤적을 생성하고 결과적으로 동일한 지역해를 찾게 된다. 그림 2는 1차원 단봉함수를 탐색하는 경우를 예로든 것으로 서로 다른 시작 스트링 (00과 11)에서 지역탐색이 시작되었지만 등간격으로 복수 회 시행되는 UDS에 의해 하위 스트링(100)에서 다시 만날 수 있음을 보인다. 이후 100 스트링에서 시작되는 지역탐색은 동일한 탐색결과를 얻으므로 00이나 11 스트링 중 나중에 지역탐색을 시작한 스트링은 더 이상 탐색을 진행할 필요가 없다

uDEAS에서는 이러한 재탐색(revisit) 방지를 위해 매 session이 끝날 때 마다 이력확인(history check)을 실행한다[9]. 이력확인이란 매 session이 끝난 후 얻게 된 최적 이진행렬을 하나의 대표값으로 변환하여 이를 메모리에 저장시키고, 이후의 재시작에서 해당 행 길이에 대해 동일한 이진행렬이 탐색되었는지를 확인하는 것이다. 가장 간단한 이력확인 방법은 이진행렬을 이진스트링으로 이어 붙인 후 이를 하나의 정수로 복호화하고 이 값을 이력테이블(history table)에 저장한 후 동일한 행길이의 이전 값들과 비교하는 것이다. 이는 탐색 변수로서 이진행렬을 다루는 uDEAS에서만 가능한 효율적인 재탐색 방지 기법이다.



(a) Branin 함수



(b) Goldstein-Price 함수

그림 3 두 개의 벤치마크 함수에 대한 uDEAS 예비탐색 양상
Fig. 3 Preliminary search aspects of uDEAS for two benchmark functions

전역최적화시 추가적으로 고려되어야 할 사항은 탈출계획(escaping scheme)이다. 즉, 비용함수 값이 아주 미미하게 감소할 때 일반적으로 현 지역해의 인력영역(region of attraction)에 빠졌다고 간주하고 현재 지역탐색의 지속 여부를 결정해야 한다. 이 때 비용함수 감소정도가 얼마 이하이거나, 수렴하고 있는 지역해가 어느 값 정도이면 현 인력영역으로부터 탈출해야 하는지를 결정해야 하는데 이는 문제마다 조금씩 다르므로 쉽지가 않다. uDEAS에서는 예비탐색(preliminary search)이라는 특별한 과정을 통해 비용함수의 형상을 대략적으로 조망할 수 있으며 이를 탈출계획에 적용한다. 예비탐색이란 초기행렬의 행 길이를 최소값인 1부터 3~5까지 변화시키며 지역탐색을 수행하고, 동일한 초기행렬 행 길이에 대해 3~5회씩 무작위 이진행렬을 생성한 후 지역탐색을 수행하는 것을 의미한다. 만약 다수의 지역해가 존재하고 넓게 분포되어 있다면 예비탐색 패턴을 관찰함으로써 쉽게 알 수 있다.

탈출계획과 관련된 세 가지 파라미터들은 *colIndRestart*와 *costIndRestart*, 그리고 *optInitRowLen* 들이며 uDEAS의 탈출계획은 다음과 같이 설명될 수 있다.

> "현 지역해 행렬의 행 길이가 *colIndRestart*가 될 때의 비용함수 값이 *costIndRestart*보다 큰(최대화 문제의 경우, 작은) 경우 지역탐색을 멈추고, 행 길이가 *optInitRowLen*인 임의의 행렬을 생성해서 재탐색(restart)을 수행한다."

그림 3은 잘 알려진 테스트 함수 중 지역해가 하나인 Branin 함수와 다수의 지역해를 갖는 Goldstein-Price(GP)함수에 대해 uDEAS 예비탐색을 수행한 결과를 보인다. Branin 함수와는 달리 GP 함수는 여러 지역해로 수렴하며 대략 행 길이가 6비트 이상이 되면 비용함수가 수렴하기 시작함을 알 수 있다.

uDEAS의 예비탐색을 통해 비용함수의 대략적인 형상정보를 얻어내고, 탈출 관련 파라미터를 구했다면 본탐색(main search)을 수행한다. 본탐색에서는 초기점 행렬의 행 길이는

```
Initialize the history table.
For iRL = iLen1:iLen2
    Randomly generate an initial binary matrix T_{n×iRL}.
    HISTORY CHECK for T_{n×iRL}.
    For m = 1:numMaxRestart
        For k = iRL:mRL
            T_{n×(k+1)} = SESSION(T_{n×k}).
            RESTART CHECK for T_{n×(k+1)}.
            HISTORY CHECK for T_{n×(k+1)}.
        end for
    end for
end for
```

그림 4 uDEAS의 전역최적화 의사코드
Fig. 4 Pseudocode for global optimization of uDEAS

*optInitRowLen*로 고정시키고 재탐색을 *numMaxRestart* 횟수만큼 수행한다. 이 때 행 길이의 최대값은 *mRL*로 설정해 주어야 한다. 그림 4는 상기 예비탐색과 본탐색을 의사코드로 표현한 것으로 uDEAS의 전역최적화 루틴을 나타낸다. 코드에서 *iLen1*과 *iLen2*는 초기점 행렬의 행 길이를 나타내며, 예비탐색의 경우 앞서 든 예에 대해서는 *iLen1*=1과 *iLen2*=3으로 설정해 주면 된다. 본탐색은 초기행렬 행 길이가 고정되어 있으므로 *iLen1*=*iLen2*=*optIntRowLen*로 설정해 준다.

3. Multi-start 기법

3.1 Multi-start의 기본 원리

Multi-start는 지역탐색 기법을 전역탐색 기법으로 확장하기 위해 사용된 방법으로서 임의의 시작점으로부터 지역탐색을 수행하여 지역해를 발견한 후 새로운 탐색점으로부터 지역해를 반복적으로 찾는 알고리듬이다. 그러므로 MS는 두 가지 단계를 갖는다. 첫째는 새로운 탐색점을 생성하는 단계이고, 둘째는 지역탐색을 통해 점차 향상된 탐색점을

```
Initialize i = 1
while (Stopping condition is not satisfied.)
{
    Step 1. (Generation)
            Construct solution x_i.
    Step 2. (Search)
            Apply a search method to improve x_i.
            Let x_i' be the solution obtained.
    if (x_i' improves the best)
            Update the best.
    i = i + 1
}
```

그림 5 Multi-start의 의사코드
Fig. 5 Pseudo-code of multi-start

발견하는 단계이다. 그림 5는 MS의 두 가지 단계를 표현하는 의사코드(pseudo-code)를 나타낸다[12].

본 논문에서는 그림 5에서 Step 2에 uDEAS를 적용한 형태가 되며, Step 1의 시작점 생성에 있어서 보다 효과적으로 분산시키는 방법을 제안한다.

3.2 제안된 Multi-start 기법

전역최적화 시 가급적 넓은 영역을 골고루 탐색하는 것이 일반적으로 효과적이다. 이를 위해서는 multi-start에 의해 생성되는 초기 탐색점들이 상호 최대한 멀리 위치해야 하는데, 본 논문에서는 uDEAS의 초기 탐색점이 이진수로 구성되어 있음을 활용하여 효과적으로 이를 구현하는 방법을 새롭게 제안한다. 이를 위해 기존의 uDEAS에서 초기 탐색점 생성 시 사용된 난수 추출 방법을 분석하고, 전 탐색영역에 초기점을 골고루 분산시키는 새로운 난수 추출방법을 제안한다.

기존의 방법은 각 비트에 대해 0과 1 사이에서 균일분포(uniform distribution) 확률변수를 생성 후 0.5 이하의 값이 나오면 0, 그 이상의 값이 나오면 1로 결정했다. 다음 식은 이러한 flipping의 원리를 확률분포 함수로 표현한 것이다.

$$f(x)=\begin{cases}0 & \text{for } 0\leq x<0.5\\ 1 & \text{for } 0.5\leq x<1\end{cases} \tag{3}$$

식 (3)은 GA에서 비트별로 난수를 추출할 때도 많이 사용하는 범용 확률함수이다. 그러나 현재까지의 이력(history) 정보는 담고 있지 않으므로 해밍 거리(Hamming distance)의 개념을 적용한 '유도된 이진난수(Guided Random Bit, GRB) 생성법'을 본 논문에서 다음과 같이 새롭게 제안한다.

먼저 상호간 최대한 멀리 분포하는 시작점을 만들기 위해서 해밍 거리의 개념을 도입한다. 즉, 직전 시작점의 한 변수의 특정 비트가 0이었다면 해밍 거리를 멀게 하기 위해 그 비트에 대해 현재 1이 생성될 확률을 그만큼 높이는 것이다. Multi-start에서는 복수 개의 시작점을 생성하므로 이를 구현하는 가장 효과적인 방법은 특정 비트에서 현재까지 1이 생성된 횟수를 세거나 총합을 구해서 그 값으로부터 그 다음 비트 값을 확률적으로 정하는 것이다. 이 때 다음과 같

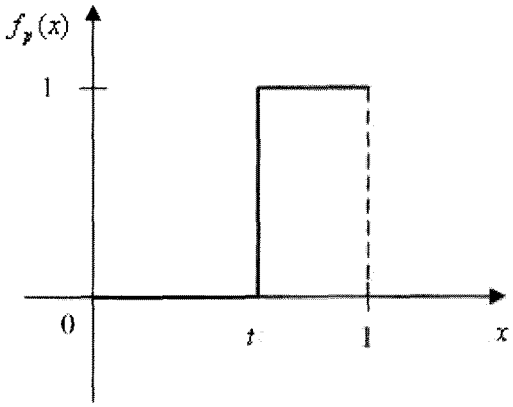


그림 6 문턱값을 갖는 균일분포 확률함수
Fig. 6 Uniform distribution function with threshold value

이 문턱값(threshold value)을 갖는 균일분포 확률함수를 이용한다.

$$f_t(x)=\begin{cases}0 & \text{for } 0\leq x<t\\ 1 & \text{for } t\leq x<1\end{cases},\quad 0<t<1 \tag{4}$$

여기에서 t는 문턱값을 나타낸다. 그림 6은 식 (4)를 나타낸 것으로 문턱값 t가 1에 가까울수록 0이 나올 확률이 비례해서 커짐을 알 수 있다.

표 1 무작위 시작점 생성 예
Table 1 Example of random initial strings

시행 횟수	시작점 스트링
1	0 1 1 1 0
2	1 0 1 0 0
3	0 1 1 0 1
비트별 합	1 2 3 1 1

이해를 돕기 위해 한 변수에 대해 표 1과 같이 5비트 길이의 시작점 스트링이 무작위로 3회 생성되었다고 하자. 비트 위치별로 1이 발생한 횟수는 비트별 합을 구하면 되며, 표에서 보는 것처럼 최상위 비트부터 1, 2, 3, 1, 1이 된다. 이제 GRB 생성법을 이용해서 4회 째에 임의의 이진 난수 스트링을 생성하기 위해 현재까지의 비트별 합을 시행 횟수 4로 나눈 후 이 값들을 각 비트별 문턱값으로 정한다. 이렇게 되면 4회 째에 생성되는 난수는 지금까지 해당 비트에서 1이 발생한 횟수에 비례하는 확률로 0을 발생시키며 이는 역으로 그에 반비례하는 확률로 1을 발생시킴을 의미한다. 그러므로 4회 째 생성된 시작 스트링의 최상위 비트는 3/4 $(=1-\frac{1}{4})$의 확률로 1을 발생시키며, 그 다음 비트는 1/2 $(=1-\frac{2}{4})$, 그리고 그 다음 비트는 1/4$(=1-\frac{3}{4})$, 3/4, 3/4의 확률로 각각 1을 발생시킨다. 다변수 문제의 경우에는 각 변수별 스트링의 비트별 합을 구해 상기한 GRB 생성법을 적용하면 된다.

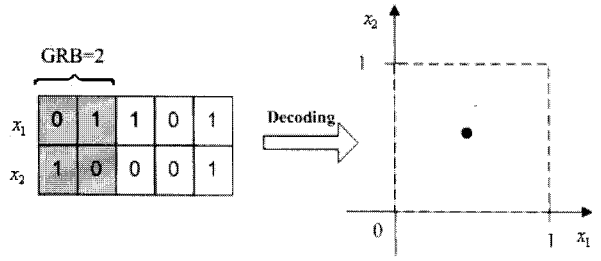


그림 7 GRB 적용비트가 2인 시작행렬의 개념도
Fig. 7 Conceptual diagram in case of GRB=2

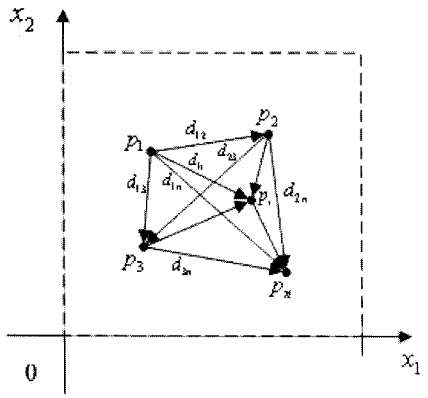


그림 8 시작점 간 유클리드 거리 합
Fig. 8 Sum of Euclidean distances between starting points

이 방법에서 한 가지 고려해야 할 점은 이진코딩 시 두 점이 부호공간에서 해밍 거리가 멀다고 해서 실수값에서도 항상 먼 것은 아니라는 사실이다. 일례로 011과 100은 해밍 거리상 최대값인 3이지만 정수로 복호화하면 1차이의 가장 가까운 값들이기 때문이다. 그러므로 상기 예에서 4회 째 난수 생성시 전체 비트에 대해서 뿐만 아니라 일부 비트에 대해서 GRB법을 적용하는 것도 확인할 필요가 있다.

제안된 GRB법의 효용성을 검증하기 위해 본 논문에서는 다음과 같은 조건 하에서 시뮬레이션을 수행했다.

- 차원: 1차원 ~ 5차원
- 각 변수 값의 실수 범위: 0 ~ 1
- 초기 비트의 총 길이: 5비트
- GRB 적용 비트 범위(MSB 기준): 0 ~ 5비트
- Multi-start 횟수: 10회

그림 7은 2개의 변수 x_1, x_2를 갖는 2차원 문제에 대해 GRB를 앞에서부터 2비트까지 적용하여 초기 행렬을 생성 후 부호화함으로써 0과 1 사이의 한 점으로 변환하는 과정을 나타낸다. 그림에서 회색으로 표현된 GRB 적용 비트들은 제안한 방법을 사용하지만 나머지 비트들은 기존의 flipping 방법을 사용한다.

본 논문에서는 생성된 시작점 들이 얼마나 골고루 분산되어 있는 가를 측정하기 위해, 생성된 모든 초기점들 간의 유클리드 거리의 합을 산포도(diversity measure)로 삼았으며 계산식은 다음과 같다.

$$D=\frac{1}{m}\sum_{i=1}^{m-1}\sum_{j=i}^{m-1}\| p_i - p_{j+1}\| \tag{5}$$

여기에서 m은 Multi-start 횟수를 나타내며, $\| \bullet \|$는 두 점 사이의 유클리드 거리를 나타낸다. 그림 8은 식 (5)를 구하는 과정의 개념을 나타낸 것이다.

표 2는 GRB 적용 비트 길이를 1에서 5비트까지 변화시키며 산포도를 측정한 값을 나타낸다. 난수 발생 특성 상 각 경우에 대해 10000번을 시행한 후 평균 산포도를 계산한 값이며, GRB가 0비트라는 것은 모든 비트에 대해 flipping을 적용한 기존의 방법을 사용했음을 의미한다. 표 3은 표 2의 결과에 대해 기존의 완전 flipping 방법에 대한 산포도의 증가 정도를 백분율로 환산한 값들을 나타낸다. 표에서 차원수가 증가할수록 산포도 증가율이 조금씩 감소함을 알 수 있는데, 이는 GRB가 기본적으로 동일 변수의 스트링들에 대해서만 적용되기 때문에 변수의 숫자가 증가할수록 그 효과가 감소하기 때문이다. 결과적으로 초기점 생성 시 본 논문에서 제안한 GRB를 적용한 비트 길이가 길어질수록 그에 비례해서 산포도가 증가하며, 총 평균 4.56%의 산포도 증가 효과가 있음을 확인할 수 있다. 다음 장에서는 제안한 초기점 생성방법을 uDEAS에 적용하여 함수 최적화한 결과를 소개한다.

표 2 GRB 적용 비트 길이 변화에 따른 산포도 (각 10000번 시행 후 평균값)
Table 2 Diversity measures according to GRB length

차원	GRB 적용 비트 길이					
	0	1	2	3	4	5
1차원	1.50	1.55	1.57	1.58	1.58	1.59
2차원	2.34	2.41	2.45	2.46	2.47	2.48
3차원	2.97	3.05	3.10	3.12	3.13	3.14
4차원	3.50	3.58	3.64	3.66	3.67	3.68
5차원	3.95	4.05	4.10	4.13	4.15	4.16

표 3 GRB 적용 비트 길이 변화에 따른 산포도의 개선 정도(단위 %)
Table 3 Improvement of diversity according to GRB length

차원	GRB 적용 비트 길이						평균
	0	1	2	3	4	5	
1차원	0	3.36	4.87	5.40	5.88	6.51	5.20
2차원	0	2.94	4.42	4.94	5.48	5.69	4.69
3차원	0	2.77	4.19	4.83	5.35	5.54	4.54
4차원	0	2.36	3.84	4.57	4.92	5.24	4.19
5차원	0	2.42	3.82	4.52	4.98	5.21	4.19
평균	0	2.77	4.23	4.85	5.32	5.64	4.56

표 4 테스트 함수 정보
Table 4 Test functions

테스트 함수명	수 식	알려진 전역최적값
Branin(BR)	$f_{BR}(x)=(x_2-\frac{5.1}{4\Pi^2}x_1^2+\frac{5}{\Pi}x_1-6)^2+10(1-\frac{1}{8\Pi})\cos x_1+10$	0.397887
Six-hump camel-back (CA)	$f_{CA}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	-1.031628
Goldstein-Price(GP)	$f_{GP}(x)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]$ $\times[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	3.0
Rastrigin(RA)	$f_{RA}(x)=x_1^2+x_2^2-\cos 18x_1-\cos 18x_2$	-2.0
3D Hartman (H3)	$f_{H3}(x)=-\sum_{i=1}^{4}c_i\exp[-\sum_{j=1}^{3}\alpha_{ij}(x_j-p_{ij})^2]$	-3.862782
6D Hartman (H6)	$f_{H6}(x)=-\sum_{i=1}^{4}c_i\exp[-\sum_{j=1}^{6}\alpha_{ij}(x_j-p_{ij})^2]$	-3.322368

표 5 본탐색시 함수별 전역 탐색 지수
Table 5 Global search indices for test functions in main search

	BR	CA	GP	RA	H3	H6
optInitRowLen	3	3	3	3	3	3
colIndRestart	7	6	6	6	6	6
costIndRestart	0.5	-0.5	10	-1.9	-3.5	-3.25
전역최적값	0.397887	-1.031628	3.0	-2.0	-3.862782	-3.322368

4. 함수 최적화 실험 결과

제안된 GRB법이 uDEAS의 전역최적화 성능에 미치는 영향을 확인하기 위해 본 논문에서는 잘 알려진 테스트 함수의 전역해를 찾는 실험을 했다. 표 4는 전역해가 이미 알려져 있는 테스트 함수들을 나열한 것이며, 그림 9는 uDEAS로 예비탐색을 수행 시 시작점으로부터 지역해를 찾아가는 모습을 각 함수의 등고선 형상과 함께 보인 것이다. 테스트 함수에 대한 보다 자세한 설명은 [9]에 기록되어 있다.

표 5는 uDEAS를 이용한 본탐색 시 사용된 함수별 전역탐색 지수들을 나타낸 것이다. 시작점의 행 길이인 *optInitRowLen*는 GRB를 적용하기 위해 특별히 모든 문제에 대해 3비트로 통일시켰으며 나머지 지수들은 2.2절에서 설명한 방식대로 예비탐색을 통해 정했다.

그림 9는 네 개의 2차원 함수의 등고선 형상과 예비탐색 시 초기점으로부터 각 지역해로 찾아가는 양상을 보인다. 그리고 표 6은 3비트 길이의 초기 행렬에 대해 제안된 GRB를 적용했을 때 알려진 전역해를 찾기까지 소요된 비용함수 계산 횟수를 정리한 것이다. 비용함수 계산 횟수가 짧을수록 그만큼 빠른 탐색 성능을 가짐을 의미한다. 표 6에서 보면 BR과 H3, H6 함수들은 GRB가 확실히 효율적이지만 나머지 세 함수들은 오히려 탐색성능을 다소 저하시킴을 볼 수 있다. 반면에 GRB 길이 면에서 보면 GRB를 1비트까지 적용했을 때 탐색속도가 평균 4% 향상되었고, 그 이상부터는 점차 향상도가 감소함을 알 수 있다. 이는 3장에서 확인한 것처럼 제안된 GRB가 시작점의 산포도를 향상시키긴 했지만 넓은 지역을 탐색할 수 있는 UDS 성능에 의해 uDEAS의 전역탐색 성능에 직결되지는 않음을 의미한다. 그러나 시작점 생성시 GRB를 최상위비트에 적용하는 것이 전역최적화 성능을 향상시킬 수 있음은 분명하다고 할 수 있다. 왜냐하면 GRB를 최상위비트에 적용한다는 것은 그림 1에서 보여지는 두 트리(시작점이 0과 1인 트리) 중 하나의 트리를 각 변수에서 선택할 확률을 균등하게 조정하는 것을 의미하기 때문이다.

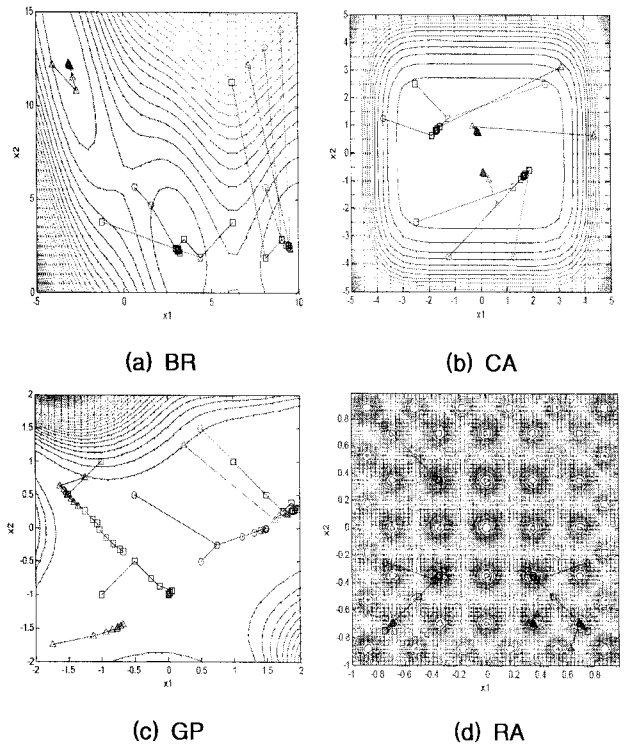
5. 결 론

본 논문에서는 고속 전역최적화 기법인 uDEAS의 전역최적화 성능을 보다 향상시키기 위해 multi-start 시 특정 비트의 이력정보를 이용해서 그 다음 시작점 행렬의 해당 비트를 0 혹은 1로 정하는 GRB 생성법을 새롭게 제안했다. 제안된 방법은 이진수로 해가 표현되는 uDEAS의 장점을 활용해서 각 비트별로 발생된 1의 수를 더하여 그에 반비례하는 확률로 1을 생성하기 때문에 구현이 간단하며 평균 4.5% 정도의 산포도 개선을 이루었다. 그리고 제안된 방법을 이용해서 uDEAS의 함수 최적화에 적용한 결과 6가지 문제에 대해 평균 2.1%, 최대 4.1%의 탐색 속도 개선을 이루었으며 최상위비트에 제안된 방법을 적용하는 것이 가장 효과적임을 확인했다. 제안된 multi-start 법은 향후 uDEAS를 이용한 최적화 문제에 지속적으로 적용되어 성능 개선을

표 6 테스트 함수에 대해 GRB법을 적용했을 때의 비용함수 계산 횟수 비교
(100번 시행한 평균값이며 %는 GRB 길이가 0일 때를 기준으로 한 증감도)

Table 6 Comparison of Test function evaluation numbers according to GRB length

GRB 길이	BR	CA	GP	RA	H3	H6	평균 증감도 (%)
0	95	110	145	267	141	265	0
1	92	114	138	270	118	252	-4.1
	-3.2%	3.6%	-4.8%	1.1%	-16.3%	-4.9%	
2	93	116	145	285	110	254	-2.7
	-2.1%	5.5%	0%	6.7%	-22.0%	-4.2%	
3	91	133	154	286	109	252	0.4
	-4.2%	20.9%	6.2%	7.1%	-22.7%	-4.9%	
평균 증감	-3.2	10.0	0.5	5.0	-20.3	-4.7	-2.1



(a) BR (b) CA

(c) GP (d) RA

그림 9 테스트 함수별 예비탐색 양상

Fig. 9 Preliminary search aspects for test functions

이룰 것으로 기대된다. 아울러 제안된 초기점 생성법은 유사한 구조를 가진 GA에서도 초기점 생성 시 적용될 수 있다.

감사의 글

이 논문은 2006년 동아대학교 학술연구비(신임교원 과제)에 의하여 연구되었습니다.

참 고 문 헌

[1] D. E. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison Wesley, 1989.

[2] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the travelling salesman prolbem," *IEEE Trans. Evolution. Comput.* vol. 1, no. 1, pp. 53-66, April 1997.

[3] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," *Proceedings of the Sixth International Symposium on Micromachine and Human Science*, Nagoya, Japan. pp. 39-43, 1995.

[4] T. G. Kolda, R. M. Lewis, and V. Torczon, "Optimization by direct search: new perspectives on some classical and modern methods," *SIAM Review*, vol. 45, no. 3, pp. 385-482, 2003.

[5] C. Audet and J. E. Dennis JR, "Mesh adaptive direct search algorithms for constrained optimization," *SIAM Journal on Optimization*, vol. 17, no. 1, pp. 188-217, 2006.

[6] J. -W. Kim and S. W. Kim, "Numerical method for global optimization: dynamic encoding algorithm for searches (DEAS)," *IEE Proc.-Control Theory and Appl.*, vol. 151, no. 5, pp. 661-668. Sept. 2004.

[7] J. -W. Kim and S. W. Kim, "Parameter identification of induction motors using dynamic encoding algorithm for searches (DEAS)," *IEEE Trans. Energy Conversion*, vol. 20, no. 1, pp. 16-24, March 2005.

[8] 김태규, 김종욱, "DEAS를 이용한 변압기 코아의 최적설계," 대한전기학회 논문지, 56권, 6호, pp. 1055-1063, June 2007.

[9] J. -W. Kim and S. W. Kim, "A fast computational optimization method: univariate dynamic encoding

algorithm for searches (uDEAS)," *IEICE Trans. Fundamentals*, vol. E90-A, no. 8, pp. 1679-1689, Aug. 2007.

[10] J.-W. Kim, T. Kim, J.-Y. Choi, and S. W. Kim, "On the global convergence of univariate dynamic encoding algorithm for searches (uDEAS)," *International Journal of Control, Automation, and Systems*, vol. 6, no. 4, pp. 571-582, Aug. 2008.

[11] J. -W. Kim, T. Kim, Y. Park, and S. W. Kim, "On-load motor parameter identification using univariate dynamic encoding algorithm for searches," *IEEE Trans. Energy Conversion*, vol. 23, no. 3, pp. 804-813, Sept. 2008.

[12] *Handbook of metaheuristics*, Kluwer Academic Publishers, 2003.Inc. 2006.

저 자 소 개

김 은 수 (金 銀 宿)

1983년 1월 26일생. 2008년 동아대 전자공학과 졸업. 2008년~현재 동 대학원 전자공학과 석사과정

Tel : 051-200-5579

E-mail : tacctics@hotmail.com

김 만 석 (金 萬 錫)

1983년 1월 23일생. 2009년 동아대 전자공학과 졸업. 2009년~현재 동 대학원 전자공학과 석사과정

Tel : 051-200-5579

E-mail : luhyoun@hotmail.com

김 종 욱 (金 鍾 旭)

1970년 10월 24일생. 1998년 포항공대 전자전기공학과 졸업. 2000년 동 대학원 전자전기공학과 졸업(석사). 2004년 동 대학원 전자전기공학과 졸업(박사). 2004년~2006년 포스코 기술연구소 전기강판연구그룹 연구원2, 2006년~현재 동아대학교 전자공학과 조교수

Tel : 051-200-7714

Fax : 051-200-7712

E-mail : kjwook@dau.ac.kr