ISSN 1229-800X(Print) / ISSN 2586-7792(Online)
The Transactions of the Korean Institute of Electrical Engineers Vol. 67P, No. 3, pp. 131~136, 2018
http://doi.org/10.5370/KIEEP.2018.67.3.131

최대 반복 횟수 없이 튜닝에 기반을 둔 HS 최적화 구현

HS Optimization Implementation Based on Tuning without Maximum Number of Iterations

이 태 봉†
(Tae-bong Lee)

Abstract - Harmony search (HS) is a relatively recently developed meta-heuristic optimization method imitating the music improvisation process where musicians improvise their instruments' pitches searching for a perfect state of harmony. In the conventional HS algorithm, it is necessary to determine the maximum number of iterations with some algorithm parameters. However, there is no criterion for determining the number of iterations, which is a very difficult problem. To solve this problem, a new method is proposed to perform the algorithm without setting the maximum number of iterations in this paper. The new method allows the algorithm to be performed until the desired tuning is achieved. To do this, a new variable bandwidth is introduced. In addition, the types and probability of harmonies composed of variables is analyzed to help to decide the value of HMCR. The performance of the proposed method is investigated and compared with classical HS. The experiments conducted show that the new method generally outperformed conventional HS when applied to seven benchmark problems.

Key Words : HS(harmony search), Meta-heuristic, HMCR, Tuning.

1. 서 론

자연현상을 모방한 최적화 알고리즘은 기존 고전적 수치 알고리즘이 갖는 복잡함과 초기 값에 대한 민감성 등과 같은 단점을 극복하고자 1970년대 이 후 많이 고안되었다. 그 중 Harmony search(HS) 알고리즘은 즉흥적 음악 연주 시 음악인들의 화음 개선 방식을 모방하여 비교적 최근에 개발된 메타 휴리스틱 최적화 기법이다[1]. 고전적 기법과 달리 HS는 비수학적 알고리즘으로 설계변수에 대한 초기 값을 요구하지 않으며 경사법(gradient search) 대신 확률변수 기법 (stochastic random search)을 기반으로 하고 있다. 또한 두 부모 벡터만 고려하는 GA와 달리 수집된 모든 벡터를 고려하여 새로운 벡터를 생성한다.

비교적 구현이 쉽고 수렴이 빠르며 탐색(exploration)과 활용(exploitation)의 균형이 좋은 HS는 다양한 최적화 문제 해결에 성공적으로 적용되어 왔다.

그러나 많은 장점에도 불구하고 알고리즘 매개변수를 고정된 값으로 설정한다거나 매개변수 값을 잘못 설정한 경우 국지 값으로 수렴하게 되는 단점이 있다. 이러한 단점을 극복하고 알고리즘 성능을 향상시키기 위한 많은 노력들이 있었다. 그 중 [2]는 알고리즘 매개변수를 총 알고리즘 반복 횟수와 현재 반복에 따라 바뀌도록 하여 HS의 성능을 개선하였다. 이 후 이 방식에 대한 여러 진전이 있어왔다[3-5].

† Corresponding Author : School of Electronic Engineering, Gachon University, Korea
E-mail : tblee@gachon.ac.kr
접수일자 : 2018년 8월 9일
최종완료 : 2018년 8월 16일

또 다른 방법은 기존 HS의 구조나 요소를 변경하거나 이에 더해 다른 최적화 결합하는 방식이다[6-10]. 그러나 이들의 공통점은 알고리즘 수행을 위해 사전에 알고리즘 총 반복 횟수가 정해져야 한다는 것이다.

본 논문에서는 이러한 단점을 극복하고자 횟수 기반에서 튜닝 기반으로 HS를 구현하고자 한다. 튜닝 기반이란 튜닝 대역폭 크기를 가변적으로 설정하고 이 값이 원하는 값이 될 때까지 알고리즘을 수행하도록 하는 것이다. 이와 더불어 튜닝 속도를 조절하는 매개변수를 도입하여 원하는 튜닝 값에 도달 되는 속도를 조절하여 알고리즘 반복을 제어 할 수 있도록 하였다.

2. Harmony Search

모든 메타 휴리스틱 알고리즘은 다각화(diversification)를 위한 해 공간에 대한 탐색(exploration)과 잠재적 최적 값들의 활용(exploitation)을 통한 해의 강화(intensification)라는 두 가지 과정으로 구성된다. 두 과정의 적절한 조화는 해의 전역성과 정확성 및 빠른 수렴에 있어 매우 중요한 요소이다[10].

HS는 즉흥 연주 시 연주자들이 좀 더 좋은 하모니를 얻기 위해 악기의 피치를 조정해가는 것을 모방하여 비교적 최근에 개발된 메타 휴리스틱 알고리즘 이다. 연주자가 하나의 음을 즉흥적으로 낼 때는

① 연주자의 기억에 있는 음을 내거나
② 기억 속 음을 기준으로 그에 이웃한 음을 연주하거나
③ 기억에 의존하지 않고 악기의 음역에서 임의로 선택하여 연주를 한다.

HS 알고리즘은 이를 '하모니 기억', '톤 조정' 및 '무작위'

라는 세 가지 방식으로 모방하여 결정변수 값을 선택해 가며 해 벡터(solution vector)를 개선해 나간다. HS는 구체적으로 다음과 같은 단계별 과정을 통해 구현된다.

Step 1. 최적화 문제와 알고리즘 매개변수를 초기화 한다.
Step 2. HM(harmony memory)를 초기화 한다.
Step 3. 새로운 하모니 즉 해 벡터를 생성한다.
Step 4. HM을 최신화 한다.
Step 5. 중단 요건을 검사 한다.

2.1 최적화 문제와 알고리즘 매개변수 초기화

첫 단계에서 기술해야 하는 최적화 문제는 다음과 같이 표현 할 수 있다.

$$\text{Min.(or Max.)} f(x),\ x=[x_1 x_2 \cdots x_n],\ x_i \in B_i \tag{1}$$

식 (1)에서 $f(x)$는 목적함수이고 x_i는 결정변수 이며 연속이거나 이산적 이다. n은 결정변수의 개수이고 B_i는 각 결정변수 범위를 나타내는 집합으로 다음과 같이 정의 된다.

$$B_i=\{x_i | L_i \le x_i \le U_i\}.(i=1,\cdots,n) \tag{2}$$

최적화 문제와 더불어 초기화해야 할 HS 알고리즘 매개변수는 해 벡터를 저장하는 HM의 크기 HMS와 확률변수 $HMCR$과 PAR 그리고 기존 해의 강화를 위해 사용되는 대역폭 b 및 알고리즘 반복횟수 이다.

2.2 HM(harmony memory) 초기화

HM의 초기화는 균일 분포 확률함수를 이용해 무작위로 하모니를 생성하여 HM에 저장하는 것이다. 즉, j번째 해 벡터의 i번째 목적변수 값은

$$x_i^j = L_i + rand(0,1)\cdot(U_i - L_i),\ i=1,\cdots,n\ \ j=1,2,\cdots,HMS \tag{3}$$

와 같이 생성되어 HM에 저장된다. 식 (3)에서 $rand(0,1)$은 0과 1사이 균일분포를 갖는 확률함수 이다. HM에 저장된 초기 해 벡터는 다음과 같이 나타낼 수 있다.

$$HM=\begin{bmatrix} x^1 \\ \vdots \\ x^{HMS} \end{bmatrix},\ x^j=[x_1^j x_2^j \cdots x_n^j]\quad ,\ j=1,\cdots,HMS \tag{4}$$

2.3 새로운 하모니 생성

새로운 하모니 $x^N=\{x_1^N, x_2^N, \cdots, x_n^N\}$는 원소 x_i^N를 기억회상, 피치조정 및 무작위 세 가지 방식 중 하나가 확률적으로 결정되어 생성된다. 기억회상은 x_i^N를 기존에 HM에 저장된 값 중 하나를 선택하는 방식으로 이러한 방식을 선택할 확률이 $HMCR$ 이다. 반면 $(1-HMCR)$은 x_i^N를 기존 값이 아닌 변수 범위 내에서 임의의 값을 선택하는 무작위 방식에 대한 확률이다. 즉, 기억회상과 무작위 방식에 대한 선택은 다음과 같이 식 (5)로 표현 할 수 있다.

$$x_i^N \leftarrow \begin{cases} x_i^N \in \{x_i^1, x_i^2, \cdots, x_i^{HMS}\}, & \text{w.p.}\quad HMCR \\ x_i^N = L_i + rand(0,1)\cdot(U_i - L_i) & \text{w.p.}\quad (1-HMCR) \end{cases} \tag{5}$$

기억회상이 선택 된 경우 HM의 값을 그대로 사용 할 것인지 피치 조정을 할 것인지를 다시 한 번 확률적으로 결정하게 되는데 이 때 사용되는 매개 변수가 PAR 이며 그 구조는 식 (6)과 같다.

$$x_i^N \leftarrow \begin{cases} x_i^N \pm rand(0,1)\cdot b_i, & \text{w.p. } PAR \\ x_i^N, & \text{w.p. } (1-PAR) \end{cases} \tag{6}$$

식 (6)에서 b_i는 변수 x_i의 대역폭 이다. 결과적으로 세 가지 방식에 대한 확률은 다음과 같다.

$$x_i^N \leftarrow \begin{cases} x_i^N = L_i + rand(0,1)\cdot(U_i - L_i), & \text{w.p.}\quad (1-HMCR) \\ x_i^N = x_i^N + rand(-1,1)\cdot b_i, & \text{w.p.}\quad HMCR\cdot PAR \\ x_i^N \in \{x_i^1, x_i^2, \cdots, x_i^{HMS}\}, & \text{w.p.}\quad HMCR\cdot(1-PAR) \end{cases} \tag{7}$$

2.4 HM 최신화

단계 3에서 결정된 새로운 하모니는 목적함수에 적용되고 그 결과 HM 내 가장 나쁜 하모니 보다 더 좋은 결과를 가질 때 새로운 하모니는 해당 하모니를 대체 한다. 이러한 해벡터 즉 하모니 개선 과정을 그림으로 나타내면 다음 그림 1과 같다.

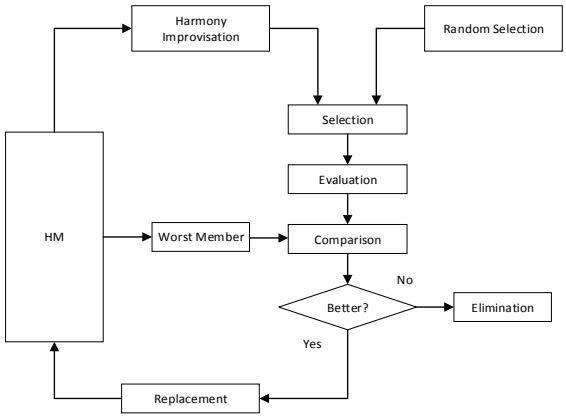

그림 1 HS의 하모니 개선과정
Fig. 1 HS process of harmony improvisation

2.5 중단 요건 검사

이 단계에서는 미리 설정된 종료 기준이 만족하는지를 검사하여 안 된 경우 단계 3과 4를 반복하고 아니면 알고리즘을 종료한다. 지금까지 HS의 단계별 알고리즘 전계 과정을 그림 2에 나타냈다..

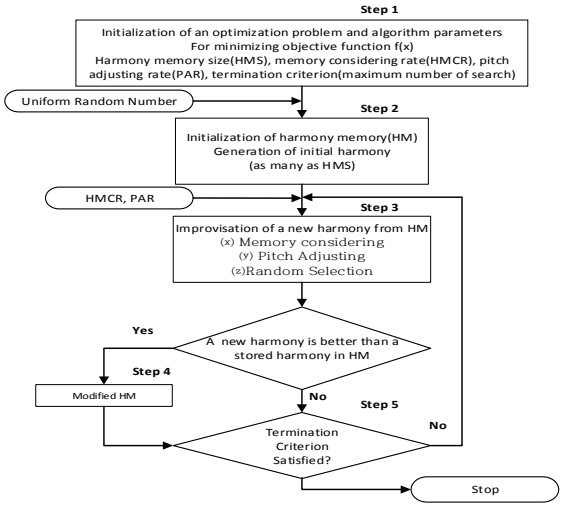

그림 2 HS 알고리즘의 단계별 최적화 절차

Fig. 2 Optimization step procedure of the harmonysearch algorithm

3. 하모니 종류 및 확률

식 (1)의 하모니는 n개의 변수 조합으로 구성되며 각 변수는 앞에서 서술한 세 가지 방식 중 하나로 선택된다. 그러나 선택된 변수의 조합으로 구성되는 하모니 종류는 매우 다양하고 확률 또한 복잡하다. 앞에서 살펴 본 하모니 구성을 위한 변수 선택 과정을 순서도로 나타내면 다음 그림 3과 같다.

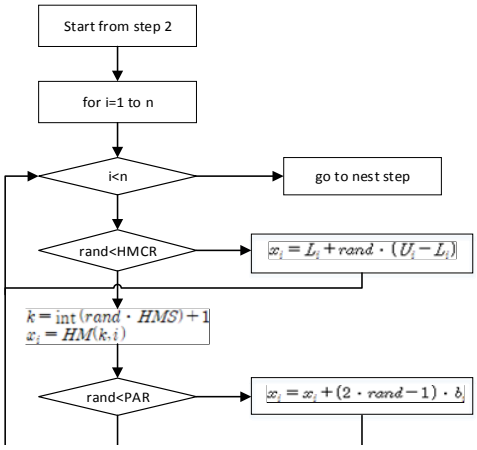

그림 3 변수 선택 순서도

Fig. 3 flow chart of variable selection

표 1 변수 조합 및 하모니 확률

Table 1 variable combination and harmony probability

harmony combination		x_1		
		RS	MS	PA
x_2	RS	$(1-HMCR)^2$	$(1-HMCR)\cdot HMCR\cdot(1-PAR)$	$(1-HMCR)\cdot HMCR\cdot PAR$
	MS	$(1-HMCR)\cdot HMCR\cdot(1-PAR)$	$HMCR^2\cdot(1-PAR)^2$	$HMCR^2\cdot(1-PAR)\cdot PAR$
	PA	$(1-HMCR)\cdot HMCR\cdot PAR$	$HMCR^2\cdot(1-PAR)\cdot PAR$	$HMCR^2\cdot PAR^2$

순서도에 따라 결정되는 하모니 종류는 3^n으로 지수 적으로 증가한다. 두 변수의 경우 하모니 종류 및 확률은 다음 표 1과 같다.

표 1에서 보듯이 하모니 종류는 모두 아홉($3^2=9$) 가지이며 크게 세 부분으로 구분된다. 첫째는 두 변수 모두 무작위로 선택되는 경우이고 두 번째는 HM을 기반으로 하는 두 가지 기억회상 선택 간 조합이며 나머지 하나는 무작위 선택과 기억회상의 하이브리드 조합 이다. 이들 세 부류의 하모니 생성 확률은 다음과 같다.

1) 무작위 확률 : $RSH=(1-HMCR)^2$
2) 하이브리드 확률 : $HYH=2(1-HMCR)\cdot HMCR$
3) 기억회상 확률 : $MCH=HMCR^2$

보는 바와 같이 세 부류의 하모니 확률은 PAR과 무관하며 각 하모니의 확률 값이 개별변수와 달리 비선형적으로 결정된다. 서로 다른 하모니는 해의 다각화와 강화 측면에서 역할의 정도가 다르다 할 수 있다. 특히 해의 다각화와 가장 깊은 관련이 있는 무작위 하모니에 대한 확률이 $(1-HMCR)$이 아닌 $(1-HMCR)^2$ 라는 것은 매우 중요한 내용이다. 변수의 수가 n인 경우 그 확률은

$$RSH=(1-HMCR)^n \tag{8}$$

이 되어 $HMCR$에 반비례하여 기하급수적으로 작아진다. 따라서 $HMCR$ 값을 결정할 때는 변수의 수에 따라 위 식을 고려하여 정해야 한다. 한편 하이브리드 하모니는 적어도 하나의 변수가 무작위 값을 가지므로 이 또한 해의 다양성과 관계가 있으므로 무작위 하모니와 더불어 다각화 하모니라 부르기로 한다. 무작위 하모니와 하이브리드 하모니 확률을 더한 다각화 하모니 확률은 다음과 같다.

$$DIVH=RSH+HYH=1-HMCR^2 \tag{9}$$

식 (9)는 해의 다각화를 살펴 볼 수 있는 지표로 사용 할 수 있을 것이다. 식 (9)에 따르면 두 변수 문제의 경우 $HMCR=1/\sqrt{2}$ 일 때 $DIVH=0.5$가 되어 튜닝 하모니와 다각화 하모니 확률이 같게 된다.

4. 튜닝 기반 제어

지금까지 HS의 알고리즘 수행 종료는 최대 반복 횟수를 알고리즘 수행 전에 정하여 이루어져 왔다. 그러나 이 값을

어떻게 정하고 또 얼마로 해야 할지는 매우 어려운 문제이다. 이와 관련 가장 큰 영향을 미치는 요소가 대역폭 b_i 이다. 대역폭 b_i는 최적 값으로 수렴하는 알고리즘 수렴속도와 그 정밀도에 큰 영향을 미치는 값으로 알고리즘 초반에는 빠른 수렴을 위해 큰 값이 유리하고 후반으로 갈수록 보다 정밀한 최적 값 튜닝을 위해 작은 값이 요구된다. 그러나 고정된 b_i를 사용하는 고전적인 HS에서는 최적 값의 정밀성을 중시하여 비교적 작은 b_i을 사용하였으며 그 결과 보다 큰 알고리즘 반복 횟수를 필요로 했다. 이러한 문제점을 해결하고자 [2]는 알고리즘 초반에는 비교적 큰 대역폭을 갖다가 알고리즘 반복에 따라 대역폭이 작아지는 가변 대역폭을 제안하여 HS 성능 향상에 기여하였다. 그러나 이 또한 알고리즘 최대 반복 횟수가 사전에 결정된 것을 전제로 하며 따라서 이 값을 얼마로 할 것인가 하는 것은 여전히 큰 문제이다. 본 연구에서는 HS의 성능 향상을 위해 가변 대역폭을 사용하되 알고리즘 최대 반복 횟수를 사용하는 대신 정의된 가변 대역폭을 통해 알고리즘 반복을 제어 하고자 한다. 이를 위해 가변 대역폭을 다음과 같이 정의 하도록 한다.

$$b_i(j) = b_{i0}\exp\left(-\frac{(j-1)}{DI}\right),\ j=1\text{ to }\max(b_i(j))<\epsilon\ ,\ i=1,\cdots,n \quad (10)$$

식에서
b_{i0}:x_i의 초기 대역폭
$DI \gg 1$:튜닝 감속 지수
$\epsilon \ll 1$:튜닝정밀도

식 (10)은 알고리즘 반복 횟수를 미리 정하지 않고 원하는 튜닝에 사용되는 대역폭의 정밀도가 원하는 값, ϵ이 될 때까지 알고리즘을 반복 수행 하는 구조를 갖는다. 대역폭 ϵ과 더불어 이에 도달할 때 까지 속도(시간)를 DI값을 통해 알고리즘 반복 횟수를 이 중으로 제어 할 수 있게 된다. 예를 들어 $b_0=1$이고 $\epsilon=\exp(-10)$인 경우 DI 값에 따라 알고리즘 반복 횟수, NoI는 다음 표 2와 같다.

초기 값 b_0는 비교적 큰 값을 선택하면 알고리즘 수렴속도가 향상되고 알고리즘 진행에 따라 대역폭 값이 작아져 해의 원하는 정도까지 세밀한 튜닝이 가능하게 된다. 본 논문에서는 초기 값으로 $(U_i-L_i)/2 \sim (U_i-L_i)/4$를 사용 할 예정이다.

5. 수치 예

본 연구에서 제안한 방법의 성능을 확인하고 그 우수성을 평가하기 위해 다음과 같은 7개의 최적화 함수를 선정하였다[1]. 각 함수에 대해 HS의 알고리즘 매개변수는 $HMS=15$, $HMCR=0.95$, $PAR=0.95$로 설정하였다. 단 많은 국지 값을 갖고 있는 4)에 대해서는 $HMCR=0.35$로 하였다. 본 논문에서 제안한 가변 대역폭의 초기 값은 $b_{i0}=(U_i-L_i)/2(i=1,2)$, 튜닝 정밀도는 $\epsilon=10^{-5}$ 및 $\epsilon=10^{-7}$ 두 가지 값을 사용하여 그 결과를 비교하였다. DI는 주어진 문제에 따라 적절한 값을 선택하였다. 논문에서 제시된 방법에 대한 해의 정확성과 신뢰성을 평가하기 위해 각 함수를 100번 수행하여 해에 대한 평균, 표준 편차 및 최대 오차를 살펴보았다. 이와 더불어 오차 허영 범위 $tol=10^{-6}$에 대한 성공률도 표시 하였다.

1) Six-hump camelback function

$$f_1(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x^6+x_1x_2-4x_2^2+4x_2^4,\quad -10\le x_i\le 10,\ i=1,2$$

2) Rosenbrock function

$$f_2(x)=100(x_2-x_1^2)^2+(1-x_1^2),\ -10\le x_i\le 10,\ i=1,2$$

3) Goldstein and Price function I

$$f_3(x)=\left\{1+(x_1+x_2+1)^2\left(19-14x_1+3x_1^2-14x_2+3x_2^2\right)\right\}\times\left\{30+(2x_1-3x_2)^2\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right\},\quad -5\le x_i\le 5,\ i=1,2$$

4) Goldstein and Price function II

$$f_4(x)=\exp\left\{\frac{1}{2}(x_1^2+x_2^2-2)^2\right\}+\sin^4(4x_1-3x_2)+\frac{1}{2}(2x_1+x_2-10),\quad -5\le x_i\le 5,\ i=1,2$$

5) Eason and Fenton's gear train inertia function

$$f_5(x)=\frac{1}{10}\left\{12+x_1^2+\frac{1+x_2^2}{x_1^2}+\frac{x_1^2x_2^2+100}{(x_1x_2)^4}\right\},\quad 0\le x_i\le 10,\ i=1,2$$

6) Wood function

$$f_6(x)=100(x_2-x_1^2)^2+(1-x_1)^2+90(x_4-x_3^2)^2+x_1x_2)^2+(1-x_3)^2,\ +10.1\left\{(x_2-1)^2+(x_4-1)^2\right\}+19.8(x_2-1)(x_4-1)\quad -5\le x_i\le 5,\ i=1,2$$

7) Powell quartic function

$$f_7(x)=(x_1+10x_2)^2+5(x_3-x_4)^2+(x_2-2x_3)^4+10(x_1-x_4)^4,\quad -5\le x_i\le 5,\ i=1,2$$

먼저 7개 함수에 대하여 $\epsilon=0^{-5}$로 설정하여 제시한 알고리즘을 수행하였다. 다음 표 3은 그 결과이다.

다음 표 4는 $\epsilon=10^{-7}$로 설정하여 수행한 수치 예 결과이다. 함수 4)의 경우 모두 99회 전역 값으로 수렴하였으며 표의 값들은 이들에 대한 데이터 이다. 두 실험에서 같은 DI를 사용했지만 튜닝 정밀도가 다름으로 알고리즘 반복 횟수, NoI가 차이가 나는 것을 확인 할 수 있다. 또한 보다 정밀한 튜닝을 한 결과인 표 5의 값이 더 우수함을 알 수 있다.

표 2 DI와 NoI
Table 2 DI and NoI

DI	100	200	500	1000	3000	5000	10000	100000
$NoI-1$	1000	2000	5000	100000	300000	500000	1000000	10000000

표 3 7 함수에 대한 수치 예 결과, $\epsilon = 10^{-5}$

Table 3 Results for 7 numerical examples with $\epsilon = 10^{-5}$

function	optimal value	*DI*	*NoI*	mean value	standard. deviation	maximum error	success rate(%)
$f_1(x)$	$f_1^* = -1.0316285$	60	829	-1.03162845e+00	2.40015901e-11	4.66210137e-08	100
$f_2(x)$	$f_2^* = 0$	1000	13816	3.37286817e-12	4.09493872e-12	2.12066066e-11	100
$f_3(x)$	$f_3^* = 3.0$	100	1313	3.00000000e+00	6.72664381e-10	3.01707281e-09	100
$f_4(x)$	$f_4^* = 1(99)$	3000	39368	1.00001710e+00	1.70227823e-04	1.69374545e-03	100
$f_5(x)$	$f_5^* = 1.74$	60	788	1.74415201e+00	1.70381433e-12	4.15200560e-03	*
$f_6(x)$	$f_6^* = 0$	8000	104979	5.65285781e-07	8.67545260e-07	5.82602167e-06	81
$f_7(x)$	$f_7^* = 0$	8000	104979	8.33350531e-08	1.07475037e-07	5.38252576e-07	100

표 4 7 함수에 대한 수치 예 결과, $\epsilon = 10^{-7}$

Table 4 Results for 7 numerical examples with $\epsilon = 10^{-7}$

function	optimal value	*DI*	*NoI*	mean value	standard. deviation	maximum error	success rate(%)
$f_1(x)$	$f_1^* = -1.0316285$	60	1106	-1.03162845e+00	2.89355885e-15	4.65101395e-08	100
$f_2(x)$	$f_2^* = 0$	1000	18421	3.69743585e-16	3.66286541e-16	1.93814761e-15	100
$f_3(x)$	$f_3^* = 3.0$	100	1773	3.00000000e+00	7.94901893e-14	4.13447054e-13	100
$f_4(x)$	$f_4^* = 1(99)$	3000	53183	1.00000000e+00	2.47746257e-16	8.88178419e-16	100
$f_5(x)$	$f_5^* = 1.74$	60	1064	1.74415201e+00	1.63107740e-15	4.15200559e-03	*
$f_6(x)$	$f_6^* = 0$	8000	141821	6.69857990e-08	1.75094370e-07	9.44972668e-07	100
$f_7(x)$	$f_7^* = 0$	8000	141821	7.16015132e-08	7.40685058e-08	3.88138505e-07	100

표 5 7 함수에 대한 고전적인 HS의 결과, $b_i = 0.001$

Table 5 Results for 7 numerical examples with $b_i = 0.001$

function	Optimal value	NoI	mean value	standard. deviation	maximum error	success rate(%)
$f_1(x)$	$f_1^* = -1.0316285$	1106	-9.72877456e-01	1.71827594e-01	8.16182324e-01	2
$f_2(x)$	$f_2^* = 0$	18421	5.28586501e-01	1.01825206e+00	3.79687092e+00	3
$f_3(x)$	$f_3^* = 3.0$	1773	6.04058299e+01	2.09777410e+02	2.07285163e+03	1
$f_4(x)$	$f_4^* = 1(99)$	53183	1.00006636e+00	5.30991763e-04	5.06030015e-03	93
$f_5(x)$	$f_5^* = 1.74$	1064	1.74490765e+00	3.42147642e-03	2.62774489e-02	*
$f_6(x)$	$f_6^* = 0$	141821	1.66049708e-06	2.79627727e-06	2.22012803e-05	57
$f_7(x)$	$f_7^* = 0$	141821	2.20652649e-07	7.20333391e-08	3.88489133e-07	100

한편 제안 된 방식과 기존 방식의 성능 차이를 확인하기 위해 표 4의 *NoI* 값을 적용하여 $b_i = 0.001$을 제외한 모든 알고리즘 매개변수를 같은 값을 사용하여 수치 예를 수행하였다. 표 5는 이에 대한 결과이다.

표 5에서 보듯이 함수 4)의 경우 96회 전역 값으로 수렴하였다. 두 표를 비교해 보면 모든 경우 제안된 방식의 성능이 우수함을 알 수 있다. 특히 함수 1) - 3) 및 6)에 대한 성능 차이가 매우 컸다. 함수 7)의 경우도 허용오차에 대한 성공률은 같으나 세부 항목에 있어서는 제시된 방식이 우수함을 알 수 있다. $b_i = 0.01$을 사용한 경우 성능 차이는 더욱 컸다.

6. 결 론

본 논문에서는 HS 알고리즘을 종래 횟수 기반에서 튜닝 기반으로 구현하였다. 튜닝 기반이란 알고리즘이 실행 전 결정된 일정한 횟수만큼 수행되는 것이 아니라 원하는 튜닝 정밀도에 의해 제어 되는 것이다. 종래 횟수기반에서는 튜

닝에 사용되는 대역폭이 알고리즘 수행 동안 고정된 값을 갖거나 일정한 범위에서 큰 값에서 작은 값으로 변하도록 하였다. 이러한 방법은 대역폭의 크기에 의해 결정되는 튜닝 정밀도에 한계가 있으며 그 값이 정해진 알고리즘 반복 횟수에 영향을 받게 된다. 이러한 문제점을 해결하기 위해 본 논문에서는 횟수 대신 대역폭이 원하는 값이 될 때까지 알고리즘이 수행되도록 구현하였다. 이를 위해 새로운 가변 대역폭 함수를 정의하였다. 새롭게 정의된 대역폭은 알고리즘 반복에 따라 값이 작아질 뿐만 아니라 작아지는 속도 또한 제어 가능하다. 따라서 이 함수를 적용하면 원하는 튜닝 정밀도와 더불어 속도를 제어함으로써 결과적으로 알고리즘 반복 횟수를 제어 하게 된다.

새로운 알고리즘 수행 방법과 더불어 변수 선택과 생성 하모니의 관계를 규명하였다. 특히 *HMCR*과 무작위 하모니 생성이 변수의 수에 따라 지수 적 관계임을 알 수 있었다. 이는 변수의 수에 따라 *HMCR* 값 선택이 달라져야 함을 의미 한다.

제시된 알고리즘은 성능 평가를 위해 7개의 평가함수에 적용하였다. 서로 다른 두 개의 튜닝 정밀도를 정하여 수행한 결과 튜닝 기반으로 HS가 구현 될 수 있음을 알 수 있었고 원하는 튜닝 값이 작은 경우 좀 더 최적 값에 근접한 값을 도출하였다. 아울러 튜닝 기반에서 수행된 알고리즘 횟수를 적용해 고전적인 HS를 같은 평가 함수에 적용하여 비교한 결과 함수에 따라 정도의 차이가 있으나 본 연구에서 제시된 방식이 매우 우수함을 보였다.

References

[1] Z. W. Geem, J. H. Kim, and G. V. Loganathan, “A new heuristic optimization algorithm: Harmony search,” *J. Simul.,* vol. 76, no. 2, pp. 60-68, Feb. 2001.

[2] M. Mahdavi, M. Fesanghary, and E. Damangir, “An improved harmony search algorithm for solving optimization problems,” *Appl. Math. Comput.,* vol. 188, no. 2, pp. 1567-1579, May 2007.

[3] F. Zhao, Y. Liu, C. Zhang, and J. Wang, “A self-adaptive harmony PSO search algorithm and its performance analysis,” *Expert Syst. Appl.*, vol. 42, no. 21, pp. 7436-7455, 2015.

[4] H.-B. O. Yang, L.-Q. Gao, S. Li, X. Kong, and D.-X. Zou, “On the iterative convergence of harmony search algorithm and a proposed modification,” *Appl. Math. Comput.*, vol. 247, pp. 1064-1095, Nov. 2014.

[5] S. A. Patil and D. A. Patel, “An overview: Improved harmony search algorithm and its applications in mechanical engineering,” *Int. J. Eng. Sci. Innov. Technol.*, vol. 2, no. 1, pp. 433-444, 2013.

[6] P. Chakraborty, G. G. Roy, S. Das, D. Jain, and A. Abraham, “An improved harmony search algorithm with differential mutation operator,” *Fundam. Inform.*, vol. 95, no. 4, pp. 401-426, 2009.

[7] L. Wang and L.-P. Li, “An effective differential harmony search algorithm for the solving non-convex economic load dispatch problems,” *Int. J. Elect. Power Energy Syst.*, vol. 44, no. 1, p. 832-843, 2013.

[8] B. H. F. Hasan, I. A. Doush, E. Al-Maghayreh, F. Alkhateeb, and M. Hamdan, “Hybridizing harmony search algorithm with different mutation operators for continuous problems,” *Appl. Math. Comput.*, vol. 232, pp. 1166-1182, Apr. 2014.

[9] Edgar Alfredo Portilla-Flores, Álvaro S nchez-M rquez, Leticia Flores-Pulido, Eduardo Vega-Alvarado, Maria Bárbara Calva Yáñez, Jorge Alexander Aponte-Rodríguez, and Paola Andrea Niño-Suárez, “Enhancing the Harmony Search Algorithm Performance on Constrained Numerical Optimization,” *IEEE Access,* vol. 5, pp. 25759-25779, 2017.

[10] A. E. Eiben and C. A. Schippers, “On evolutionary exploration and exploitation,” *Fundamenta Informaticae*, vol. 35, no. 1-4, pp. 1-16, Aug. 1998.

저 자 소 개

이 태 봉 (Tae-bong Lee)

1986년 홍익대학교 전자공학과 졸업.
1989년 동 대학원 전자공학과 (석사/박사)
1995년～현재 : 가천대학교 전자공학과 교수
E-mail : tblee@gachon.ac.kr