한국CAD/CAM학회 논문집
Transactions of the Society of CAD/CAM Engineers

Vol. 19, No. 3, pp. 245-252. September 2014
DOI http://dx.doi.org/10.7315/CADCAM.2014.245

〈학술논문〉

pISSN 1226-0606
eISSN 2288-6036

근사 최적화를 활용한 뻐꾸기 탐색법의 성능 개선

이세정†
서울시립대학교 기계정보공학과

Surrogate-Based Improvement on Cuckoo Search for Global Constrained Optimization

Se Jung Lee†
Department of Mechanical and Information Engineering, The University of Seoul

Received 19 April 2014; received in revised form 14 May 2014; accepted 3 June 2014

ABSTRACT

Engineering applications of global optimization techniques are recently abundant in the literature and it may be caused by both new methodologies arising and faster computers coming out. Many of the optimization techniques are based on natural or biological phenomena. This study put focus on enhancing the performace of Cuckoo Search (CS) among them since it has the least number of parameters to tune. The proposed enhancement can be achieved by applying surrogate-based optimization at every cycle of CS, which fortifies the exploitation capability of the original method. The enhanced algorithm has been applied several engineering design problems with constraints. The proposed method shows comparable or superior performance to the original method.

Key Words: Approximation, Cuckoo Search (CS), Design of experiments (DOE), Global optimization, Metamodel, Surrogate-based optimization (SBO)

1. 서 론

최근 들어 전역 최적화 기법들의 공학적 활용에 대한 연구가 활발하게 이루어지고 있는데, 효율적인 새로운 방법론의 개발과 컴퓨터 처리속도의 비약적 증가가 주 요인이다. 전역 최적화 기법 중 자연 현상 또는 생물학적 현상에 기반한 Metaheuristcs 알고리즘(Metaheuristic Algorithm, MA)들이 많이 개발되었다[1,2]. 그 중 Yang[2]의 책에는 저자가 작성한 매트랩 코드가 수록되어 있어서 바로 실제 설계 문제에 적용해 볼 수 있다.

본 연구에서는 많은 MA 중에서 뻐꾸기 탐색법(Cuckoo Search, CS)[3,4]을 기초로 하여 근사 최적화 개념을 도입하여 그 효율성을 제고하기 위한 제안을 하고자 한다. 선택 이유는 다른 많은 방법들에 비하여 미리 설정하여야 하는 매개 변수(parameter)의 수가 가장 적으므로 공학적으로 적용하기가 비교적 수월하다는 장점 때문이다. 즉 이 방법을 쓰기 위하여 매개 변수 조정 때문에 사용자가 곤란을 겪지 않아도 되는 것이다. 물론 최적설계 문제를 효율적으로 해결하기 위해서는 당면한 문제의 특성을 파악하는 것이 가장 중요한 작업임에는 틀림이 없으나 여기서는 아무런 선입견

†Corresponding Author, selee@uos.ac.kr

없이 블랙박스 형태의 문제가 닥쳤을 때 잘 처리할 수 있는 방법을 개발하고자 함이 그 목적이며, 특정 문제에 대한 사전 정보는 없다고 가정한다.

CS의 최근 연구 경향 및 추세는 Fister[5], Rajabioun[6], Yang[7] 등의 논문을 참조하면 한 눈에 파악할 수 있다. CS의 기본 개념[3,4]을 살펴 보면, 이 알고리즘은 국부 무작위 이동(Local random walk)과 전역 탐색 무작위 이동(Global explorative random walk)을 적절하게 혼합하여 새로운 개체들을 만들고, 현 세대의 개체들 중 좋지 않다고 판단되는 개체들과 대체하는 과정을 반복하여 좋은 개체 들만 남기게 되는 원리를 구현한 것 이다. 이 알고리즘은 현재 다양한 분야에 적용되고 있으며, 그 동안 풀기 어려웠던 문제들에 대하여 해결 가능성을 보여 주고 있다[8-12].

MA의 일반적인 특징은 최적화 과정 초기에 빠르게 해 근처로 접근할 수 있지만 해 근처에서 오랫동안 수렴하지 않는 단점들이 있을 수 있다. 이 연구에서는 MA의 매 사이클마다 가장 좋은 해 근처에서 국부적 근사모델(Surrogate, Metamodel)을 만들어 이를 국부 최적화하여 새로운 실험 점으로 추가하는 방법을 제안한다. 이 방법이 전형적인 근사 최적화(Surrogate-Based Optimization, SBO) 방법과 다른 점은 근사모델을 지속적으로 개선하는 과정이 없으며 다만 매 사이클마다 가장 나쁜 실험 점 대신에 근사 최적 점을 대치하여 세대를 구성하는 점이다. MA와 SBO 를 적절하게 조합하는 목적은 국부 및 전역 탐색의 장점을 모두 취하고자 함이며, 이를 통하여 빠른 수렴 성을 확보하고자 하는 의도를 구현하고자 하는 것이다. 근사 최적화의 궁극적인 목적은 가능한 적은 수의 시뮬레이션 계산을 하여 근사모델을 개선시키고, 점차적으로 다 나은 최적 해를 찾아 가는 데 있다. 근사 최적화에 대한 개념과 개발 현황은 저자의 이전 논문[13]을 참조하면 파악할 수 있을 것이다.

전형적인 CS 절차는 Fig. 1과 같다. 초기 개체 수를 정하고 목적, 구속 함수의 계산을 한 후, 국부 및 전역 무작위 이동에 의하여 새로운 개체를 생성하여 좋은 개체들을 유지하는 과정을 반복 하는 것이 알고리즘의 핵심이다.

CS는 원래 구속 함수(constraint function)가 없는 전역 최적화 문제를 풀기 위하여 개발된 알고리즘이었다. 아직까지 방법론 자체에서 구속 함수를 다룰 수는 없기 때문에 이번 연구에서 구속 함수의 처리는 벌칙 함수(Penalty function) 방법을 이용하겠다. 또한 본 연구에서 다루고자 하는 문제는 일반적인 공학적 설계 문제로 전역 최적화 문제이고, 목적 함수나 구속 함수가 비선형성을 가질 수 있는 경우이다. 예제에 따라서 설계 변수가 이산 변수인 경우도 포함할 수 있다.

```
Generate initial population of n host nests
  x_i, i=1,2,... ,n
While (< MaxGeneration) or (stop criterion)
    Get a cuckoo randomly by Levy flights
    Evaluate its quality F_i
    Randomly choose a nest, j among n
    If F_i is better F_j, replace j by the new solution
    A fraction, P_a of worse nests are abandoned
        and new ones are built
    Keep the best solutions
    Rank the solutions and find the current best
end while
```

Fig. 1 Pseudo code of the original Cuckoo search

CS의 기본 개념 및 구현에 관한 개괄적인 주제들에 대하여 논의를 하였으므로, 다음 장부터 본 연구에서 제안하는 근사 최적화를 활용한 CS의 개념을 설명하겠다. 수정된 CS를 본래의 방법과 구별하기 위하여 Cuckoo Search with Approximation (CSA)로 부르기로 한다. 이어서 제안한 CSA와 원래 방법인 CS의 성능을 비교하기 위하여 사용할 설계 문제들을 소개하고, 수치 실험 방법을 설명하겠다. 마지막에 수치 실험 결과의 분석과 더불어 연구의 결론을 맺도록 한다.

2. 근사 최적화를 활용한 CS

근사 최적화 방법의 기본 개념은 해석이나 실험에 소요되는 시간이 긴 경우에 소수의 실험 점들로 근사 모델을 만들고, 최적화 알고리즘에 따라 원래 함수의 값이 필요할 때마다 근사 모델을 계산하는 방식으로 진행하게 된다. 근사 최적화의 개론적 소개는 Wang and Shan[14] 또는 Kazemi 등[15]에 구속 조건이 있는 경우까지 잘 정리되어 있다.

근사 최적화를 수행할 때 근사 최적 점을 어떻게 이용하는 가에 따라서 다양한 방법들이 존재할 수 있다. CS 방법론이 국부 및 전역 이동을 통하여 전체적으로 전역 최적 해를 찾아가는 데, 본 연구에서는 근사 최적 해를 국부 이동에 의하여 얻

어지는 개체들을 개선하는 데 사용한다. 즉 국부 이동에 의하여 개체들이 만들어 지면, 현재 최적점(Current best) 근처에서 일정 수의 실험 점을 선택하여 근사 모델을 만들고, 이를 최적화하여 최적 점을 찾는다. 그리고 이 최적 점을 기존 개체들과 비교하여 이 보다 열악한 개체가 있으면 최악의 개체를 이 최적 점으로 대체한다. 만일 구한 최적 점보다 열악한 개체가 발견되지 않거나 최적화 과정에서 실패하는 경우, 구한 최적 점은 포기한다.

CS에서 사용자가 지정해야 하는 매개 변수는 두 가지가 있다. 하나는 초기 개체 수이고 다른 하나는 국부 이동과 전역 이동의 조합 비율, 즉 일정한 확률, P_a이다. Yang이 정한 초기 개체 수는 기본(default) 값이 25인데 모든 문제에 대하여 변경 없이 사용하였다. 국부 이동과 전역 이동의 조합 비율, P_a는 0.25가 기본 값인데 이도 변경 없이 사용하였다. 그러면 CS에서는 초기 개체 수만큼 국부 이동과 전역 이동을 하여 매 세대마다 50개씩의 개체 수를 유지하여 최적화를 진행한다. 또한 국부 이동은 기존의 개체들 사이의 거리를 이용하여 무작위로 시행하고, 전역 이동은 Levy 이동[3]을 이용하여 탐색 영역을 전역으로 확장하여 수행한다.

CSA에서는 알고리즘이 수정되어 매 사이클마

```
Generate initial population of n host nests
Evaluate qualities and store them
      in the solution repository
While (< MaxGeneration) or (stop criterion)
   Get a cuckoo, i randomly by Levy flights
   Evaluate its quality, F_i
   Randomly choose a nest, j among n
   If F_i is better than F_j, replace j by the new solution, i
   A fraction, P_a of worse nests are abandoned
     and new ones are built
   Keep the best solutions
   Rank the solutions and find the current best
   Select k points near the current best
     out of the repository
   Build a surrogate with k nearest neighbor points
     and find its optimum
   If the optimum found is better than anyone
     in the population,
     replace the worst solution with it
   Store all the solutions kept in the repository
end while
```

Fig. 2 Pseudo code of the Cuckoo search with approximation (CSA)

다 국부 근사 모델을 만들 때 필요한 실험 점 수를 미리 정하여야 한다. 물론 근사 모델 종류도 정하여야 하는 데, 본 논문에서는 크리깅을 사용한 결과만을 논의하기로 한다. 참고로 저자는 다양한 다른 종류의 근사 모델도 사용하였었는데, 회귀 법보다는 보간 법이 더 좋은 근사 최적 점을 찾아 주었고, 보간 법 중에서도 크리깅과 Radial Basis 함수를 이용한 보간 법이 좋은 결과를 주었다. 근사 모델을 만들기 위한 실험 점 수는 이차 함수로 근사화하기 위하여 필요한 최소 숫자의 20배로 정하였다. 이 숫자도 여러 가지로 변화시켜 보았는데, 해당 사이클에서의 가장 좋은 점 근처에서 국부 근사 모델을 만드는 것이라서 너무 적은 숫자만 아니면 항상 일관성 있는 결과를 주었다. 결국 근사화 하는데 걸리는 계산 시간을 고려하여 정하면 되는데 대부분의 공학 설계의 경우 반응(response) 값들을 계산하는 시간이 근사화에 비하여 월등하게 오래 걸린다고 가정하고 충분히 큰 숫자로 정하였다. CSA의 계산 절차를 Fig. 2에 나타내었다.

3. 수치 실험

3.1 수치 실험 환경

제안한 알고리즘의 성능 평가를 위하여 구속 조건의 유무와 상관없이 전역 최적화 함수들을 문헌에서 선택하였다. 수치 실험은 모든 함수들에 대하여 수행하였으나 본 논문에서는 구속 조건이 있는 경우만을 언급하기로 한다. 이 구속 최적화 문제들은 많은 연구자들이 다루었던 문제들이며, 매우 다양한 알고리즘들이 적용되어 성능 비교를 하였다[4,16-19]. 다만 같은 문제라도 정식화 하면서 이산 변수를 뺀다거나, 설계 영역이 다르거나, 일부 구속 조건을 생략하는 등 수정을 가한 경우들도 발견되므로 성능 비교를 하려면 정식화된 문제를 면밀하게 살펴보아야만 한다. 본 연구에서 선택된 최적화 문제들은 모두 최소화 문제로 변환되었으며, 최적화는 최소화를 의미한다. 선택한 문제들의 목적 함수, 구속 함수, 그리고 그 설계 영역을 부록에 정리하였으며 네 문제의 특징을 Table 1에 요약하였다. 한편 구속 함수 처리를 위하여 벌칙 매개 변수는 10^{20}으로 일정하게 하였다.

수치 실험을 위하여 연산장치가 네 개 이상 되는 컴퓨터에 윈도우7환경에서 MATLAB R2013a를 사용하였다. 근사 모델을 만들기 위하여 Viana[20]

Table 1 Characteristics of the chosen engineering design problems

Problem	Number of design variables	Number of discrete variables	Number of nonlinear objectives	Number of nonlinear constraints	Number of linear constraints
EP1	3	1	1	3	1
EP2	4	2	1	1	3
EP3	4	0	1	5	2
EP4	7	1	1	11	0

가 공개한 툴박스에 구현되어 있는 근사화 방법을 그대로 사용하였으며, 매개 변수들은 모두 기본 값을 사용하였다. 그리고 크리깅에서 전역 경향 함수로 상수를, 상관 관계 함수로 Gaussian을 사용하고, 국부 최적화 기법은 MATLAB 툴박스가 제공하는 fmincon을 사용하였다.

기존 CS와 제안한 CSA 알고리즘에서 난수 발생기를 사용하므로 매번 같은 입력으로 실행하여도 다른 결과가 나올 수 밖에 없다. 그래서 본 논문에서는 난수 발생 종자(Seed)를 고정시키지 않고, 같은 문제에 대하여 20회 이상 반복 계산시켜 그 평균 값을 비교 대상으로 하였다.

CS의 원시 코드는 여러 가지를 구할 수가 있는데 본 연구에서는 Yang[21]이 가장 최근에 개선하여 공개한 코드를 사용하였다. 이 코드는 함수 계산 횟수에 있어서 보다 정교하게 조절하여 성능을 획기적으로 개선할 수도 있는데, 본 논문에서는 이 방법과 제안한 방법을 비교하는 것이 목적이라서 코드를 개선하지 않고 그대로 사용하였으며, 제안한 알고리즘을 거기에 적용하였다.

CS와 CSA 모두 초기 개체 수는 25로, 무작위 변이 확률 P_a는 0.25로 기본 값을 사용하였다. 이 경우, CS는 초기 개체 수 25에 매 사이클마다 국부 개체 25개, 전역 개체 25개로, 총 50개씩 개체를 유지하여 진행한다. CSA의 경우에도 CS와 공정하게 함수 계산 횟수를 비교하기 위하여 국부 개체를 24개로 하고, 근사 최적 점을 1개 추가하여 동일한 함수 계산 횟수를 유지하도록 하였다. 즉, N_{cycle}번째 단계에서 총 함수 계산 횟수는 $N_{fe} = 25 + 50 \times N_{cycle}$이 된다. 따라서 이후 논의에서는 함수 계산 횟수 대신에 최적화 사이클로 계산 시간을 나타내도록 한다.

3.2 수치 실험 결과

제안한 CSA는 구속 조건이 없는 문제에 먼저 적용하여 그 효율성을 확인하였는데, 본 논문의 주제가 공학적인 구속 조건이 있는 문제에 국한하다 보니, 여기에는 수록하지 않았다.

본 논문의 설계 문제에서 사용했던 근사 모델은 크리깅, Support Vector를 이용한 회귀 법, Radial Basis 함수를 이용한 보간 법, 반응 표면 법 등 Viana[21]가 제공하는 다양한 기법을 시도하였는데, 이 논문에서는 크리깅에 대한 비교 자료만을 수록하였다. 근사 모델의 종류에 따라서 수렴 속도가 다르게 나타나기는 하지만 수렴 하는 경향은 동일하였으며 크리깅이 그 중 빠르게 수렴하였다. 다만 Shepard 기법의 몇몇 경우에는 CS보다 별 다른 우위를 보여주지 못하였다.

풀었던 설계 문제들에 대해서, 현재까지 문헌에 알려진 가장 좋은 해를 최적 해로 간주하였으며, CS나 CSA에 의하여 구한 최적 해의 상대 오차를

$$\varepsilon = \frac{\left|Y_e^* - Y^*\right|}{1 + \left|Y_e^*\right|}$$

의 식으로 계산하여 정확도를 비교하였다. 여기서 Y^*는 CS 또는 CSA의 한 사이클이 끝난 다음 갱신한 Y 값들 중에서 최소 값이며, Y_e^*는 미리 알고 있는 전역 최적 해의 참 값이다. 즉 이렇게 계산된 상대 오차는 매 사이클이 끝난 후 구한 최적 해의 정확도를 평가하는 척도가 된다.

모든 예제 문제를 풀 때, 알고리즘에서 난수를 이용하므로 최소한 20회 이상 반복적으로 해를 구했으며 네 가지 공학 설계 문제에 대한 최적화 결과를 요약하여 Table 2와 3에 나타내었다. 수록한 네 경우에는 20회 반복 실행 시 모두 동일한 해로 수렴하였는데, 그 기준은 두 방법으로 구한 해의 상대 오차가 1% 이하이면 동일한 해로 수렴한 것으로 판단하였다.

Table 2의 결과를 보면, 네 문제에 대하여 두 방법, 모두 가장 좋은 해는 찾았다. 그러나 표준편차를 보면 CSA가 훨씬 작은 것을 알 수 있는데, 이는 CSA가 CS 보다 안정적으로 해를 찾는다는 것을 의미한다. 그래서 해의 평균 값을 비교해 보면

Table 2 Statistics of the objective function values of 20 trials after a given number of cycles (The number in the parenthesis denotes the number of cycles taken and sd means the standard deviation)

Problem		CS	CSA
EP1 (1000) spring	best	0.012666	0.012666
	worst	0.012680	0.012670
	mean	0.012668	0.012666
	sd	3.18E-6	9.02E-7
EP2 (2000) pressure vessel	best	6059.7	6059.7
	worst	6090.7	6090.7
	mean	6067.4	6062.8
	sd	13.7	9.48
EP3 (1000) welded beam	best	1.7249	1.7249
	worst	1.7249	1.7249
	mean	1.7249	1.7249
	sd	6.23E-6	4.35E-6
EP4 (500) speed reducer	best	2996.3	2996.3
	worst	2996.3	2996.4
	mean	2996.3	2996.3
	sd	3.80E-4	4.54E-4

CSA가 우수함을 보인다. 이는 Table 3에서 모든 계산을 마친 후 상대 오차를 비교하였는데, CSA가 훨씬 작은 오차를 보여주었다. 이 두 표를 통하여 CSA가 안정성과 정확성 면에서 우수함을 알 수 있다.

Table 3 The Mean values of the relative errors at the last cycle (The number in the parenthesis is the best-known optimum value of the objective function)

Problems	Maximum number of cycles	Mean values of relative errors	
		CS	CSA
EP1 (0.01266602)	1000	1.91E-06	3.95E-07
EP2 (6059.714)	2000	1.90E-06	6.90E-07
EP3 (1.724852)	1000	1.65E-06	1.23E-06
EP4 (2996.348)	500	1.60E-07	1.40E-07

Fig. 3은 네 문제에 대하여 CS와 CSA에 의한 상대 오차를 매 사이클 마다 계산하여 표시한 그림이다. 그림의 진한 선이 CSA이고, 연한 선이 CS를 표시한다. EP1~3에서는 CSA가 훨씬 빨리 상대 오차가 줄어드는 것을 볼 수 있으며, EP4에서는 근소하게 CSA가 빨리 수렴함을 볼 수 있다. 네 경우 모두 CSA의 수렴 속도가 빠르게 나타났다.

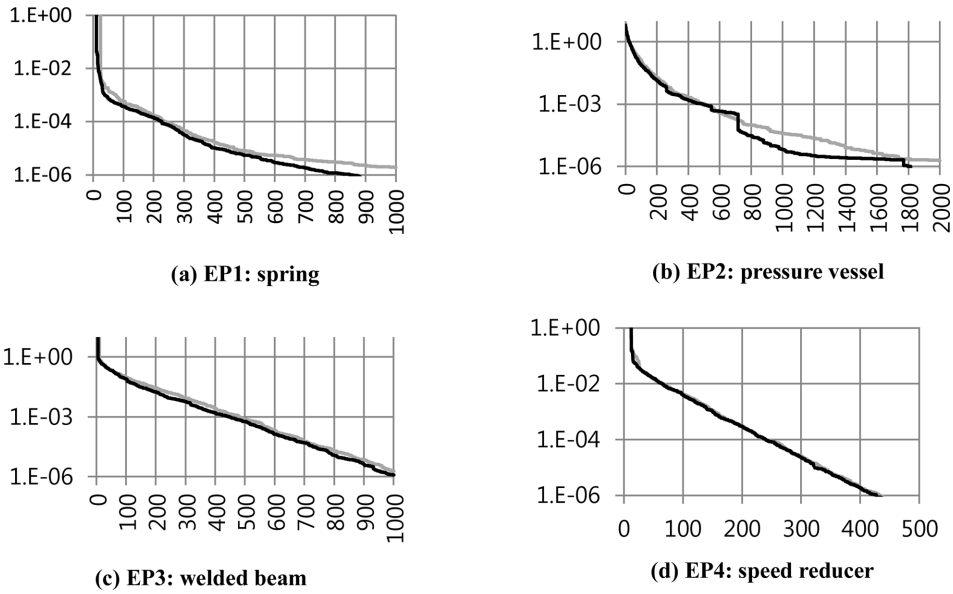

Fig. 3 The Mean values of the relative errors as the cycle proceeds (The horizontal axis depicts the number of cycles and the vertical axis expresses the mean value of the relative errors. The dark line is for CSA and the pale line is for CS)

4. 결 론

본 연구는 구속 조건이 있는 전역 최적화 문제를 해결하기 위한 MA중에서 CS를 선택하여 근사 최적화 개념을 도입하여 알고리즘의 성능을 제고하고자 하였다. 기존 연구에서와 다른 독창적인 점은 CS 알고리즘 중 국부 이동에 의하여 생성되는 개체에 현 최적 점 근처에서 국부 근사 모델을 만들어 국부 최적 점을 구하여 개체에 편입시켜 국부 탐색을 강화하는 개념을 제안하였다.

기존의 CS 알고리즘과 본 연구에서 제안한 CSA의 성능을 비교하기 위하여 구속 조건이 정의되어 있는 공학 설계 문제를 네 종류 선택하여 풀어 보았으며, 그 결과 CSA가 네 경우에서 모두 안정성, 정확성, 수렴 속도 면에서 우수한 것으로 나타났다. 다만 본 연구가 기존 방법의 개선을 목적으로 하였기에 CS 이외의 다른 방법과의 비교는 하지 않았다. CS는 전역 최적화 기법 중에서도 최근에 제안되어 계속 발전되어 가고 있는 알고리즘이므로 향후 성능 면에서 많은 개선이 있을 것으로 기대하고 있다.

향후 연구 방향은 현 알고리즘을 더욱 강화하는 작업을 생각해 볼 수 있다. 국부 근사 최적화에서 하나의 근사 최적 점을 개체에 편입시켰는데, 이와 더불어 국부 최적 점 근처에서 일정 수의 무작위 개체를 생성하여 추가할 수도 있을 것이다. 완전히 무작위 한 개체 추가보다는 확률이 높은 최적 점 근처에서의 개체 추가가 국부 이동을 훨씬 강화 시킬 수 있을 것으로 판단된다. 한편 최근의 최적화 기법의 연구가 새로운 방법론의 개발뿐 아니라 기존 방법들의 효과적인 조합, 재구성 등으로 좋은 결과들을 내는 경우가 많이 보고 되고 있다. 따라서 MA의 다른 방법과 조합하여 알고리즘을 새로 구성하여 성능 개선을 볼 수도 있을 것이다.

본 연구에서는 구속 함수를 단순한 벌칙 함수로 처리하여 구속 최적화 문제를 비구속 최적화 문제로 변환하여 해결하였는데, 최근 구속 함수를 다루는 효율적인 방법들이 많이 개발되었다. 그 중 매 사이클마다 벌칙 매개 변수를 자동적으로 조절하여 처리하는 법, 확률론적인 규칙에 의한 처리 방법, 보간이나 회귀에 의한 근사화 방법의 활용 등을 CSA에 적용시킬 수 있을 것으로 사료된다. 또한 CS는 기본적으로 무작위 이동들이 서로 독립적이므로 병렬 처리를 쉽게 도입할 수 있어서 비약적인 성능 개선을 기대할 수도 있을 것이다.

감사의 글

이 논문은 2013년도 서울시립대학교 교내학술연구비에 의하여 연구되었음.

References

1. Beheshti, Z. and Shamsuddin, S.M.H., 2013, A Review of Population-based Meta-Heuristic Algorithm, *Int. J. Advances in Soft Computing and Its Appications*, 5(1), pp.1-35.
2. Yang, X.-S., 2010, *Nature-Inspired Metaheuristic Algorithms*, 2nd Edition, Luniver Press.
3. Yang, X.-S. and Deb, S., 2009, Cuckoo Search via Levy Flights, in: Proc. of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), December 2009, India, IEEE Publications, USA, pp. 210-214.
4. Yang, X.-S. and Deb, S., 2010, Engineering optimization by cuckoo search, *Int. J. Mathematical Modelling and Numerical Optimisation*, 1(4), pp.330-343.
5. Fister, I., Jr, Yang, X.-S., Fister, D. and Fister, I., 2014, Cuckoo Search: A Brief Literature Review. In *Cuckoo Search and Firefly Algorithm*, Springer, pp.49-62.
6. Rajabioun, R., 2011, Cuckoo Optimization Algorithm, *Applied Soft Computing Journal*, 11(8), pp.5508-5518.
7. Yang, X.-S. and Deb, S., 2013, Cuckoo Search: Recent Advances and Applications, *Neural Computing and Applications*, 24(1), pp.169-174.
8. Bhargava, V., Fateen, S.E.K. and Bonilla-Petriciolet, A., 2013, Cuckoo Search: A New Nature-inspired Optimization Method for Phase Equilibrium Calculations, *Fluid Phase Equilibria*, 337, pp.191-200.
9. Bulatovic, R.R., Dordevic, S.R. and Dordevic, V.S., 2013, Cuckoo Search Algorithm: A Metaheuristic Approach to Solving the Problem of Optimum Synthesis of a Six-bar Double Dwell Linkage, *Mechanism and Machine Theory*, 61(C), pp.1-13.
10. Burnwal, S. and Deb, S., 2012, Scheduling Optimization of Flexible Manufacturing System Using Cuckoo Search-based Approach, *The International Journal of Advanced Manufacturing Technology*, 64(5-8), pp.951-959.
11. Panda, R., Agrawal, S. and Bhuyan, S., 2013, Edge Magnitude Based Multilevel Thresholding

Using Cuckoo Search Technique, *Expert Systems With Applications*, 40(18), pp.7617-7628.
12. Yang, X.-S. and Deb, S., 2014, Cuckoo Search: Recent Advances and Applications, *Neural Computing & Applications*, 24, pp.169-174.
13. Lee, S.J., 2012, An Efficient Heuristic Algorithm of Surrogate-Based Optimization for Global Optimal Design Problems, *Transactions of the Society of CAD/CAM Engineers,* 17(5), pp.375-386.
14. Wang, G.G. and Shan, S., 2007, Review of Metamodeling Techniques in Support of Engineering Design Optimization, *Journal of Mechanical Design*, 129, pp.370-380.
15. Kazemi, M., Wang, G., Rahnamayan, S. and Gupta, K., 2011, Metamodel-Based Optimization for Problems with Expensive Objective and Constraint Functions, *Journal of Mechanical Design*, 133(1), pp.1-7.
16. Akay, B. and Karaboga, D., 2010, Artificial Bee Colony Algorithm for Large-scale Problems and Engineering Design Optimization, *Journal of Intelligent Manufacturing*, 23(4), pp.1001-1014.
17. Bui, T., Pham, H. and Hasegawa, H., 2013, Improve Self-Adaptive Control Parameters in Differential Evolution for Solving Constrained Engineering Optimization Problems, *Journal of Computational Science and Technology*, 7(1), pp.59-74.
18. Rao, R.V., Savsani, V.J. and Vakharia, D.P., 2011, Teaching-learning-based Optimization: A Novel Method for Constrained Mechanical Design Optimization Problems, *Computer-Aided Design*, 43(3), pp.303-315.
19. Sadollah, A., Bahreininejad, A., Eskandar, H. and Hamdi, M., 2013, Mine Blast Algorithm: A New Population Based Algorithm for Solving Constrained Engineering Optimization Problems, *Applied Soft Computing Journal*, 13(5), pp.2592-2612.
20. Viana, F.A.C., 2011, SURROGATES Toolbox User's Guide, Version 3.0, available at http://sites.google.com/site/felipeacviana/surrogates-toolbox.
21. Yang, X.-S., 2013, Cuckoo Search (CS) Algorithm, available at http://www.mathworks.com/matlabcentral/fileexchange/29809-cuckoo-search--cs—algorithm.

부 록

EP1: Tension/compression spring design[2,4,16-19]

Min $f(\mathbf{x}) = (x_3+2)x_2x_1^2$

Subject to

$g_1(\mathbf{x}) = 1-x_2^3x_3/71785x_1^4 \le 0$

$g_2(\mathbf{x}) = (4x_2^2-x_1x_2)/12566(x_1^3x_2-x_1^4) + 1/5108x_1^2-1 \le 0$

$g_3(\mathbf{x}) = 1-140.45x_1/x_2^2x_3 \le 0$

$g_4(\mathbf{x}) = (x_1+x_2)/1.5-1 \le 0$

$-0.05 \le x_1 \le 2,\ 0.25,\ 2 \le x_3 \le 15$

where x_3 is an integer.

EP2: Pressure vessel design[16-19]

Min $f(\mathbf{x}) = 0.6224x_1x_3x_4+1.7781x_2x_3^2+ 3.1661x_1^2x_4+19.84x_1^2x_3$

Subject to

$g_1(\mathbf{x}) = -x_1+0.0193x_3 \le 0$

$g_2(\mathbf{x}) = -x_2+0.00954x_3 \le 0$

$g_3(\mathbf{x}) = 1-(\pi x_3^2x_4+4/3\,\pi x_3^3)/129600 \le 0$

$g_4(\mathbf{x}) = x_4/240-1 \le 0$

$1 \times 0.0625 \le x_1,\ x_2 \le 99 \times 0.0625,$
$10 \le x_3,\ x_4 \le 200,$

where x_1 and x_2 are integer multiples of 0.0625.

EP3: Welded beam design[4,16-19]

Min $f(\mathbf{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4\ (14+x_2)$

Subject to

$g_1(\mathbf{x}) = \tau - 13600 \le 0$

$g_2(\mathbf{x}) = \sigma - 30000 \le 0$

$g_3(\mathbf{x}) = x_1 - x_4 \le 0$

$g_4(\mathbf{x}) = 0.10471x_1^2 + 0.04811x_3x_4(14+x_2) - 5 \le 0$

$g_5(\mathbf{x}) = 0.125 - x_1 \le 0$

$g_6(\mathbf{x}) = \delta - 0.25 \le 0$

$g_7(\mathbf{x}) = 6000 - P_c \le 0$

Where

$\tau = \sqrt{\tau_1^2+2\tau_1\tau_2x_2/2R+\tau_2^2}$

$\tau_1 = 6000 / (\sqrt{2}\,x_1x_2),\ \tau_2 = MR/J$

$M = 6000(14+x_2/2)$

$R = \sqrt{x_2^2/4+\{(x_1+x_3)/2\}^2}$

$$J = 2\sqrt{2}x_1x_2(x_2^2/12 + \{(x_1+x_3)/2\}^2)$$

$$\sigma = 504000/x_3^2x_4$$

$$\delta = 65856000/(30\times10^6 x_3^3x_4)$$

$$P_c = 4.013(30\times10^6)/196\sqrt{(x_3^2x_4^6)/36}\ \times$$
$$(1-(x_3/28)\sqrt{30\times10^6/4(12\times10^6)})$$

$$-0.1 \le x_1,\ x_4 \le 2 \text{ and } 0.1 \le x_2,\ x_3 \le 10$$

EP4: Speed reducer design[16,17,19]

Min

$$f(\mathbf{x}) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2+x_7^2) + 7.4777(x_6^3+x_7^3) + 0.7854(x_4x_6^2+x_5x_7^2)$$

Subject to

$$g_1(\mathbf{x}) = 27/(x_1x_2^2x_3) - 1 \le 0$$

$$g_2(\mathbf{x}) = 397.5/(x_1x_2^2x_3^2) - 1 \le 0$$

$$g_3(\mathbf{x}) = 1.93x_4^3/(x_2x_3x_6^4) - 1 \le 0$$

$$g_4(\mathbf{x}) = 1.93x_5^3/(x_2x_3x_7^4) - 1 \le 0$$

$$g_5(\mathbf{x}) = 1/(110x_6^3)\sqrt{(745x_4/x_2x_3)^2 + 16.9\times10^6} - 1 \le 0$$

$$g_6(\mathbf{x}) = 1/(85x_7^3)\sqrt{(745x_5/x_2x_3)^2 + 157.5\times10^6} - 1 \le 0$$

$$g_7(\mathbf{x}) = x_2x_3/40 - 1 \le 0$$

$$g_8(\mathbf{x}) = 5x_2/x_1 - 1 \le 0$$

$$g_9(\mathbf{x}) = x_1/12x_2 - 1 \le 0$$

$$g_{10}(\mathbf{x}) = (1.5x_6 + 1.9)/x_4 - 1 \le 0$$

$$g_{11}(\mathbf{x}) = (1.1x_7 + 1.9)/x_5 - 1 \le 0$$

$$2.6 \le x_1 \le 3.6,\ 0.7 \le x_2 \le 0.8,$$
$$17 \le x_3 \le 28,\ 7.3 \le x_4 \le 8.3,$$
$$7.8 \le x_5 \le 8.3,\ 2.9 \le x_6 \le 3.9,\ 5.0 \le x_7 \le 5.5,$$

and x_3 is an integer.

이 세 정

1976년~1980년 서울대학교 기계공학 학사
1980년~1982년 한국과학기술원 기계공학 석사
1982년~1992년 현대건설, 삼성중공업, 한국생산성본부
1986년~1989년 Pennsylvania State University 기계공학 박사
2000년~2001년 Georgia Institute of Technology, Aerospace Engineering, 교환교수
2009년~2010년 Emory University, Biomedical Engineering, 교환교수
1993년~현재 서울시립대학교 기계정보공학과 교수