Journal of KIIT. Vol. 23, No. 12, pp. 51-66, Dec. 31, 2025. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2025.23.12.51

협력 공진화 알고리즘에서 최적 부분 문제 선택을 위한 Non-Stationary UCB-Tuned 알고리즘의 활용 방법 연구

김 경 수*

Study on Utilizing the Non-Stationary UCB-Tuned Algorithm to Choose Optimal Subproblems in Cooperative Coevolutionary Algorithms

Kyung-Soo Kim*

이 논문은 2022년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No.2022R1I1A3065378)

요 약

대규모 차원의 정의역 공간을 갖는 목적함수의 전역 최적화를 위한 협력 공진화 알고리즘에서 부분 문제 선택 과정은 주어진 목적함수의 부분 최적해를 탐색할 해 공간을 결정하는 역할을 수행하며, 이는 전역 최적해 탐색 성능에 영향을 미친다. 그러므로, 협력 공진화 알고리즘의 최적해 탐색 성능을 강화하기 위해서는 탐색과 활용의 균형을 유지하면서 최적해 발견에 중요하게 기여하는 부분 문제를 식별하는 것이 중요하다. 이에 따라, 본 논문에서는 탐색과 활용의 균형을 정교하게 유지하면서 최적의 부분 문제를 선택하기 위해 슬라이딩 윈도 기반의 Non-Stationary UCB-Tuned 알고리즘을 활용하는 방법을 제안한다. 실제 1,000차원의 벤치마크 함수를 활용한 최적화 성능평가 실험에서 제안하는 부분 문제 선택 방법을 적용한 협력 공진화 알고리즘이 가장 우수한 전역 최적해 탐색 성능을 보여주었다.

Abstract

In Cooperative Coevolutionary (CC) algorithms used to optimize an objective function with a high-dimensional domain, the subproblem selection task serves to restrict the solution space in which the sub-optimal solution is searched, thereby significantly affecting the performance of the solution search. Therefore, it is important to identify a subproblem that can significantly contribute to finding an optimal solution by balancing exploration and exploitation. Accordingly, in this paper, we propose a subproblem selection method that utilizes the sliding window-based non-stationary UCB-Tuned algorithm to effectively maintain the exploration-exploitation trade-off in the subproblem selection task. In the practical experiments with 1,000-dimensional benchmark functions, the CC algorithm with our subproblem selection method exhibited the best performance in terms of optimal solution search.

Keywords

global optimization, non-linear optimization, cooperative coevolution, upper confidence bound, non-stationary mechanism

* 국립금오공과대학교 컴퓨터공학부 교수
- ORCID: https://orcid.org/0000-0002-1044-3089

· Received: Aug. 14, 2025, Revised: Oct. 21, 2025, Accepted: Oct. 24, 2025

· Corresponding Author: Kyung-Soo Kim
Dept. of Computer Engineering, Kumoh National Institute of Technology, Gumi, Gyeongbuk 39177, Korea
Tel.: +82-54-478-7521, Email: kyungskim@kumoh.ac.kr

Ⅰ. 서 론

주어진 목적함수에 대한 전역 최적해를 탐색하는 전역 최적화 문제(Global optimization problem)는 컴퓨터 공학을 비롯한 정보통신공학, 산업공학, 경제학 등 다양한 분야에서 폭넓게 활용된다[1]-[4]. 특히, 대규모 차원의 정의역 공간을 갖는 목적함수에 대한 전역 최적화 문제는 해당 목적함수를 구성하는 결정 변수들의 복잡한 상호 종속성으로 인하여 전통적인 분석적, 해석적 최적화 방법을 이용해서는 그것의 전역 최적해를 발견하는 데 많은 한계점이 존재한다. 이러한 문제점을 극복하기 위해서 유전 알고리즘(Genetic algorithm)[5], 차분 진화(Differential evolution)[6]와 같은 진화 알고리즘을 활용한 다양한 대규모 전역 최적화 알고리즘이 제안되었다[7][8]. 이 중에서, 분할 정복법과 진화 알고리즘을 복합적으로 활용하여 대규모 전역 최적화 문제를 해결하는 협력 공진화(CC, Cooperative Coevolution) 알고리즘이 특히 주목받고 있다[9][10].

협력 공진화 알고리즘은 목적함수의 정의역 공간을 보다 작은 차원의 여러 개의 부분 공간, 즉 부분 문제로 나누고 이들 각각에 대한 부분 최적해를 탐색함으로써 전역 최적해를 발견하는 것을 목표로 한다[11]. 이 과정에서, 협력 공진화 알고리즘은 주어진 목적함수를 더 작은 차원의 부분 문제로 나누는 "문제 분할 함수"와 여러 개의 부분 문제 중에서 하나를 선택하여 해당 부분 문제에 대해서만 제한적으로 부분 최적해를 탐색하는 "부분 문제 선택 함수"를 갖는다[10]. 구체적으로, 문제 분할 함수는 목적함수의 정의역 공간을 구성하는 결정 변수 간의 상호 종속성을 식별하고 이를 토대로 정의역 공간을 여러 개의 부분 공간, 즉 부분 문제로 분할한다[12]. 이때, 하나의 부분 공간을 구성하는 결정 변수들은 상호 종속성이 존재하는 변수들이며, 서로 다른 부분 공간을 구성하는 결정 변수들은 상호 종속성이 존재하지 않는 특성을 갖는다.

문제 분할 함수에 의해 목적함수의 정의역 공간이 여러 개의 부분 문제로 분할된 후 협력 공진화 알고리즘은 각각의 부분 문제에 대한 부분 최적해를 탐색한다. 이때, 각 부분 공간을 생성(Span)하는 결정 변수들은 해당 목적함수 내에서 서로 다른 수치적 가중치를 가지므로, 각 부분 문제 역시 목적함수의 전역 최적해를 형성하는 과정에서 고유의 영향력을 갖는다. 즉, 어떤 부분 문제를 먼저 선택하고 얼마만큼 반복적으로 최적해 탐색을 수행하는가에 따라 최종적으로 발견되는 전역 최적해의 품질인 적합도(Fitness)가 달라질 수 있다.

이처럼 목적함수의 정의역 공간을 구성하는 다수의 부분 공간이 존재하는 경우 최적해를 탐색할 부분 문제를 선택하는 과정에서 탐색(Exploration)과 활용(Exploitation)의 균형을 유지하는 것이 필수적이다[13]. 만약 전역 최적해를 탐색하는 데 낮은 기여도를 갖는 부분 문제에 대해서 빈번하게 최적해 탐색을 진행한다면 전역 최적해 탐색에 소비되는 계산 비용이 불필요하게 증가하는 문제점이 발생한다. 한편, 전역 최적해 생성에 높은 기여도를 갖는 부분 문제만을 집중적으로 선택하여 탐색을 수행한다면 해의 다양성이 감소하게 되어 궁극적으로 국소 최적점(Local minimum) 또는 안장점(Saddle point)으로 조기 수렴하여 전역 최적해 탐색에 실패하는 문제가 발생할 수 있다[14].

이에 따라, 본 연구에서는 협력 공진화 알고리즘에서 탐색과 활용의 균형을 유지하면서 전역 최적해 탐색에 중요하게 기여할 수 있는 최적의 부분 문제를 선택하는 방법으로 슬라이딩 윈도(Sliding window) 기반의 Non-Stationary UCB-Tuned(Upper Confidence Bound-Tuned) 알고리즘[15][16]을 활용하는 방법을 제안한다. 실제 1,000차원의 벤치마크 함수[17]를 이용한 성능평가 결과, 제안하는 슬라이딩 윈도 기반의 Non-stationary UCB-Tuned 알고리즘을 활용한 부분 문제 선택 알고리즘을 협력 공진화 알고리즘에 적용했을 때, 더욱 우수한 성능으로 전역 최적해를 탐색할 수 있었다.

본 논문의 구성은 다음과 같다. 제2장에서는 협력 공진화 알고리즘과 관련한 선행 연구 내용을 설명한다. 제3장에서는 슬라이딩 윈도 기반의 Non-Stationary UCB-Tuned 알고리즘을 활용한 부분 문제 선택 알고리즘을 제안한다. 제4장에서는 제안하는 방법의 성능평가 결과를 설명한다. 마지막으로 제5장에서는 본 연구의 내용과 결론을 요약하고 차기 연구 계획을 소개하면서 마무리 짓는다.

II. 관련 연구

2.1 협력 공진화 알고리즘

협력 공진화 알고리즘[9]은 컴퓨터 공학 분야에서 전통적으로 활용되는 분할 정복법 메커니즘을 진화 알고리즘에 적용하여 대규모 차원의 정의역 공간을 갖는 목적함수의 전역 최적해를 효율적으로 탐색하기 위해 개발된 대표적인 진화 알고리즘의 한 종류이다. 표 1은 기본적인 협력 공진화 알고리즘의 의사코드를 보여준다.

표 1. 기본 협력 공진화 알고리즘의 의사코드
Table 1. Pseudocode of the basic CC algorithm

Algorithm basicCC

- **Inputs**: An objective function $f : R^n \rightarrow R$
- **Parameters**: Number of individuals m, $maxFEs$

$P \leftarrow$ Initialize an $m \times n$ population matrix randomly;
bestInd $\leftarrow \arg\min_{P[i,:]} f(P)$;
bestFit $\leftarrow f$(bestInd); $FEs \leftarrow m$;
$V_1, ..., V_K \leftarrow$ **ProblemDecomposer**(f);
$FEs \leftarrow FEs$ + "FEs used for Problem Decomposition";
while $FEs < maxFEs$ **do**
  for $i \leftarrow 1$ **to** K **do**
    $P[:, V_i] \leftarrow$ **Evolution**($P[:, V_i]$);
    $FEs \leftarrow FEs$ + "FEs used for Evolution";
    bestInd $\leftarrow \arg\min_{P[i,:]} f(P)$;
    bestFit $\leftarrow f$(bestInd); $FEs \leftarrow m$;
  end
end
return bestInd;

표 1에 기술된 바와 같이 협력 공진화 알고리즘은 우선 문제 분할 함수를 이용하여 주어진 목적함수 $f : R^n \rightarrow R$의 정의역 공간을 구성하는 결정 변수 간의 상호 종속성을 식별하고, 이를 토대로 정의역 공간을 K 개의 부분 공간, 즉, 부분 문제로 분할한다. 이때, 목적함수 f의 정의역 공간을 구성하는 임의의 두 결정 변수 x_i와 x_j의 상호 종속성은 아래의 식 (1)을 이용하여 식별할 수 있다[11][12]. 이때, δ_i와 δ_j는 임의의 작은 상수이며, $\Omega(x_i,x_j) > 0$이면 x_i와 x_j는 상호 종속성이 존재하는 것으로, $\Omega(x_i,x_j) = 0$이면 상호 종속성이 존재하지 않는 것으로 판단한다.

$$\begin{aligned}\Omega(x_i,x_j) = | & f(..., x_i+\delta_i, ..., x_j, ...) \\ & - f(..., x_i+\delta_i, ..., x_j+\delta_j, ...) \\ & - f(..., x_i, ..., x_j, ...) \\ & + f(..., x_i, ..., x_j+\delta_j, ...) |\end{aligned} \quad (1)$$

목적함수 f에 대한 부분 문제 분할이 완료된 후 협력 공진화 알고리즘의 부분 문제 선택 함수는 매 진화 단계에서 최적해 탐색이 진행될 부분 문제를 선택한 후 진화 알고리즘을 이용하여 부분 최적해를 탐색한다. 이때, n차원의 개체(Individual) m개로 구성된 집단(Population)을 활용하여 최적해를 탐색하며, 이 과정에서 해당 부분 문제를 생성하는 결정 변수와 연관된 차원을 제외한 나머지 차원들을 상수로 고정한 후 최적해 탐색을 진행한다[18].

i번째 부분 문제에 대한 부분 최적해 탐색이 완료된 후 협력 공진화 알고리즘은 아래와 같이 i번째 부분 문제에 대한 부분 집단 P_i의 진화된 결과 P_i'를 기존 집단 P에 반영한다.

$$\forall_{1 \le j \le |V_i|} P[:,v_j] = P_i'[:,j] \quad (2)$$

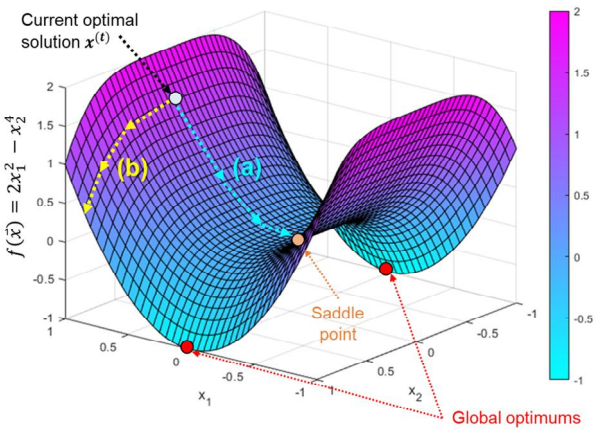
상기 식 (2)에서 P는 n차원의 개체 m개로 구성된 $m \times n$ 행렬이며, $P[:,j]$는 행렬 P의 j번째 열벡터이다. 즉, 식 (2)는 진화된 부분 문제에 대한 행렬 P' 내의 j번째 열벡터를 기존 집단 P에서 i번째 부분 문제 V_i의 v_j번째 차원에 해당하는 열벡터에 반영한다. 이렇게 갱신된 집단 내 각 개체에 대하여 적합도(Fitness)를 평가하고 가장 좋은 적합도를 갖는 개체를 새로운 전역 최적해로 지정한다. 이러한 과정은 적합도 계산을 위해 목적함수를 호출한 횟수(FEs)가 최대 허용 호출 횟수($maxFEs$)를 초과할 때까지 반복적으로 수행한다. 이후, 가장 좋은 적합도를 갖는 개체를 해당 목적함수의 전역 최적해로서 반환한다.

2.2 부분 문제 선택

표 1에 기술된 바와 같이 협력 공진화 알고리즘

은 매 진화 단계에서 K개의 부분 문제 중 하나의 부분 문제를 선택하고 해당 부분 문제에 대한 부분 최적해를 탐색한다. 이는 해당 부분 문제를 구성하는 결정 변수들이 생성하는 부분 공간에 한정하여 부분 최적해를 탐색하는 것과 동일하다.

그림 1은 목적함수 $f(x_1,x_2)=2x_1^2-x_2^4$의 최적해 공간을 그래프로 나타낸 것이다. 협력 공진화 알고리즘의 부분 문제 분할 함수는 식 (1)을 활용하여 $f(x_1,x_2)$의 정의역 공간을 두 부분 문제 $V_1=\{x_1\}$과 $V_2=\{x_2\}$로 분할한다[12]. 이 경우, 그림 1의 경로 (a)에 표현된 바와 같이 첫 번째 부분 문제 V_1을 선택하는 경우 두 번째 부분 문제 V_2에 해당하는 차원, 즉 x_2에 대해서는 탐색이 이루어지지 않고 V_1과 관련된 x_1 차원에 대해서만 제한적으로 최적해를 탐색한다. 이와 반대로, 경로 (b)와 같이 부분 문제 V_2가 선택될 때는 x_1차원은 고정된 채로 x_2차원에 대해서만 최적해 탐색을 수행한다.



(a) Solution search path on x_1 when x_2 is fixed.

(b) Solution search path on x_2 when x_1 is fixed.

그림 1. 목적함수 $f(x_1,x_2)=2x_1^2-x_2^4$의 최적해 표면
Fig. 1. Optimal solution surface of the objective function $f(x_1,x_2)=2x_1^2-x_2^4$

즉, 협력 공진화 알고리즘의 매 진화 단계에서 특정 부분 문제를 선택하면 해당 부분 문제를 구성하는 결정 변수가 생성하는 차원에 대해서만 제한적으로 최적해 탐색을 수행한다. 따라서, 부분 문제 선택 방법은 협력 공진화 알고리즘의 전역 최적해 탐색 성능은 물론 최종적으로 발견되는 전역 최적해의 품질에도 유의미한 영향을 미칠 수 있다.

2.3 부분 문제 선택 과정에서 탐색과 활용 균형의 필요성

앞서 2.2절에서 설명한 바와 같이 목적함수를 구성하는 결정 변수들은 해당 함수의 적합도를 계산할 때 고유의 영향력을 갖는다. 이는 곧 해당 목적함수의 최적해를 탐색하는 데 각 부분 문제가 고유의 기여도를 가짐을 의미한다. 예를 들어 그림 1에서 보인 목적함수 $f(x_1,x_2)=2x_1^2-x_2^4$의 경우 $x^{(t)}$ 지점에서 부분 문제 V_1을 선택하면 더 빠른 속도의 최적해 탐색을 수행할 수 있다. 이처럼, 최적해를 탐색하는 과정에서 빠른 수렴을 달성할 수 있는 부분 문제를 중점적으로 선택하여 해 탐색을 수행하는 방법을 "활용 기반의 부분 문제 선택 (Exploitation-based subproblem selection)"이라 한다. 이 방법은 최적해 탐색의 수렴 속도를 강화할 수 있지만 자칫 지나치게 빠른 최적해 탐색으로 인하여 국소 최적점 또는 안장점에 조기 수렴하는 현상이 발생할 수 있으며, 이는 곧 전역 최적해를 발견하는 데 실패하는 현상을 초래할 수 있다[14].

이러한 문제점을 완화하기 위해서는 비록 낮은 기여도를 갖는 부분 문제에 대해서도 주기적으로 해 탐색을 수행하여 해의 다양성을 강화하는 것이 중요하며, 이를 "탐색 기반의 부분 문제 선택 (Exploration-based subproblem selection)"이라 한다. 예를 들면, 그림 1의 $x^{(t)}$ 지점에서 안장점을 피하여 최적해 탐색을 진행하기 위해서는 경로 (a)에 표현된 x_1 차원보다는 경로 (b)의 x_2 차원 방향으로 해 탐색을 진행하는 것이 바람직하다. 즉, $x^{(t)}$ 에서는 비록 수렴 속도는 느려질지라도 부분 문제 V_2를 선택하는 것이 궁극적으로는 최적해를 탐색하는 데 더욱 효과적이다.

이처럼 목적함수의 전역 최적해 탐색을 성공적으로 수행하기 위해서는 부분 문제를 선택할 때, 이들의 기여도에 기반하여 탐색과 활용의 균형을 적절하게 유지하는 것이 중요하다. 그러나 종래의 협력 공진화 알고리즘은 표 1과 같이 순차적으로 부분 문제를 선택하는 방법을 사용하기 때문에 탐색과 활용의 균형을 유지하면서 최적해를 탐색하는 데 한계점이 존재한다. 이에 따라, 다음 장에서는 협력

공진화 알고리즘에서 탐색과 활용의 균형을 유지하면서 부분 문제를 선택하기 위해 슬라이딩 윈도 기반의 Non-Stationary UCB-Tuned 알고리즘을 활용하는 새로운 부분 문제 선택 알고리즘을 제안한다.

III. 슬라이딩 윈도 기반의 Non-Stationary UCB-Tuned 알고리즘을 활용한 최적의 부분 문제 선택 방법

본 연구에서 제안하는 부분 문제 선택 알고리즘은 협력 공진화 알고리즘의 매 진화 단계에서 모든 부분 문제에 대한 적합도 향상의 정도를 평가하고, 이를 토대로 슬라이딩 윈도 기반의 Non-Stationary UCB-Tuned 알고리즘을 활용하여 각 부분 문제의 최종적인 기여도를 적응적으로 평가한다. 이를 다이어그램으로 나타내면 그림 2와 같다.

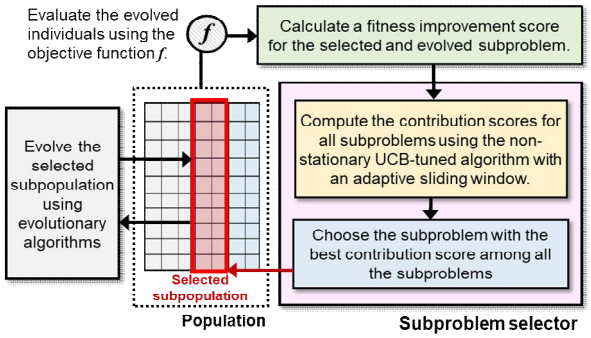


그림 2. 제안하는 슬라이딩 윈도를 활용한 Non-Stationary UCB-Tuned 알고리즘 기반 부분 문제 선택 방법의 전체 구성

Fig. 2. Overall architecture of the proposed subproblem selection algorithm based on non-stationary UCB-tuned algorithm with sliding window

3.1 UCB-Tuned 알고리즘 기반의 부분 문제 선택 방법

함수 $f: R^n \rightarrow R$의 해 공간은 모든 가능한 해의 집합인 실수 공간 R^n의 부분 집합이며, f의 해 공간을 생성하는 n개의 결정 변수들의 집합을 V라 할 때, V는 결정 변수 간의 상호 종속성에 따라 아래와 같이 K개의 분리 집합으로 표현할 수 있다.

$$V = \cup_{i=1}^{K} V_i \quad \text{s.t.} \quad \forall_{i \neq j} V_i \cap V_j = \varnothing \quad \text{and}$$

$$V_i = \left\{ v_{i,l} \mid 1 \leq l \leq |V_i| \ \text{and} \ \forall_{q \neq l} \Omega(v_{i,l}, v_{i,q}) > 0 \right\} \tag{3}$$

식 (3)에서 $\forall_{q \neq l} \Omega(v_{i,l}, v_{i,q}) > 0$은 집합 V_i 내 임의의 두 변수 $v_{i,l}$와 $v_{i,q}$ (단, $l \neq q$) 사이에 종속성이 존재함을 의미한다. 즉, 하나의 부분 문제를 구성하는 모든 변수는 상호 종속적이다.

한편, 진화 알고리즘에서는 함수 f의 전역 최적해를 탐색하기 위해 한 개 이상의 개체로 이루어진 집단을 활용하며 $m \times n$의 실수 행렬 P로 표기한다 (즉, $P \in R^{m \times n}$). 이때, 행렬 P의 i번째 행벡터는 $P[i,:]$와 같이 표현하며 이는 i번째 개체를 의미한다. 이와 유사하게, P의 j번째 열벡터는 $P[:,j]$로 표현하며 이는 행렬 P의 j번째 차원을 의미한다. 그렇다면, i번째 부분 문제 V_i에 대한 집단 P의 부분 집단 P_i는 다음과 같이 표현할 수 있다.

$$P_i = \oplus_{j=1}^{|V_i|} P[:, v_{i,j}] \tag{4}$$

식 (4)에서 $v_{i,j}$는 V_i 내 j번째 원소이며(즉, $v_{i,j} \in V_i$) $\oplus$는 두 열벡터를 왼쪽에서 오른쪽으로 연결(Concatenation)하는 이항 연산자이다. 예를 들어, $P \in R^{m \times n}$이고 $\{v_1, v_5, v_7\} \in V_i$일 때, P_i는 행렬 P의 v_1번째 열벡터 $P[:,v_1]$과 v_5번째 열벡터 $P[:,v_5]$, v_7번째 열벡터 $P[:,v_7]$을 순차적으로 연결하여 생성되는 $m \times 3$ 행렬이다.

즉, 협력 공진화 알고리즘에서 i번째 부분 문제를 선택하면 진화 알고리즘은 전체 집단 P에서 부분 집단 P_i를 구성하는 차원들에 대해서만 최적해 탐색을 수행하고 나머지 차원들에 대해서는 고정된 값을 유지한다. 이 경우, i번째 부분 문제를 선택하여 최적해 탐색을 수행하기 전과 후에 발견된 두 전역 최적해의 적합도를 비교함으로써 해당 부분 문제를 선택하였을 때 얼마만큼 전역 최적해의 적합도를 향상시킬 수 있는가를 정량적으로 측정할 수 있다. 현재 시점 t를 기준으로 직전에 관찰된 전역 최적해의 적합도를 f_{t-1}^*, 현재 선택된 부분 문제에 대하여 최적해 탐색을 수행한 후 발견된 전역 최적해의 적합도를 f_t^*라 할 때, 아래와 같이 적합도 향상의 정도를 계산할 수 있다.

$$\frac{\left|f^*_{t-1}-f^*_t\right|}{\left|f^*_{t-1}+\tau\right|} \tag{5}$$

상기 식 (5)에서 τ는 분모가 0이 되는 현상을 예방하기 위해 사용되는 Smoothing Factor이다. 식 (5)를 응용하여 i번째 부분 문제를 j번째로 선택하여 최적해 탐색을 수행하였을 때의 적합도 향상 점수를 정의하면 다음과 같다.

$$\Delta_{i,j}=\frac{\left|f^*_{prev}-f^*_{i,j}\right|}{\left|f^*_{prev}+\tau\right|} \tag{6}$$

식 (6)에서 $f^*_{i,j}$는 i번째 부분 문제를 j번째로 선택하여 최적해 탐색을 수행한 후 발견된 전역 최적해의 적합도이고 f^*_{prev}는 직전 단계에서 관찰된 전역 최적해의 적합도이다.

한편, 앞서 2.3절에서 설명한 바와 같이 부분 문제를 선택하는 과정에서 탐색과 활용의 균형을 적절하게 유지하기 위해서는 전역 최적해 탐색 과정에서 비록 낮은 기여도를 갖는 부분 문제에 대해서도 주기적으로 선택하여 탐색을 수행하는 과정이 필요하다. 동시에, 수렴 속도가 지나치게 느려지는 현상을 예방하기 위해서는 높은 기여도를 갖는 부분 문제에 대한 탐색 역시 필수적으로 수행해야 한다. 이처럼, 탐색과 활용의 균형을 유지하면서 부분 문제를 선택하기 위해서는 식 (6)에 기반하여 부분 문제를 선택하는 것보다는 UCB-Tuned 알고리즘[15]의 Arm 선택 전략을 활용하여 각 부분 문제의 기여도 점수를 계산하고 이를 토대로 부분 문제를 선택하는 것이 효과적이다. 즉, i번째 부분 문제의 기여도 점수는 다음과 같이 계산할 수 있다.

$$C_i=\mu_i+P_i \tag{7}$$

식 (7)에서 μ_i는 i번째 부분 문제에 대한 적합도 향상 점수들의 평균으로 다음과 같이 계산한다.

$$\mu_i=\frac{1}{n_i}\sum_{j=1}^{n_i}\Delta_{i,j} \tag{8}$$

식 (8)에서 n_i는 i번째 부분 문제가 선택되어 최적해 탐색이 진행된 횟수를 의미한다. 한편, 식 (7)의 P_i는 i번째 부분 문제에 대한 기여도 점수를 계산할 때 탐색과 활용의 균형을 조정하는 Padding Function으로 아래와 같이 계산한다.

$$P_i=\sqrt{\frac{\ln n}{n_i}\min\left(\frac{1}{4},\xi_i+\sqrt{\frac{2\ln n}{n_i}}\right)} \tag{9}$$

식 (9)에서 n은 지금까지 수행된 진화 단계의 총 횟수를 의미하며, K개의 모든 부분 문제에 대해서 $\sum_{i=1}^{K}n_i=n$을 만족한다. 또한, ξ_i는 i번째 부분 문제에 대한 최적해 탐색을 수행한 후 측정된 적합도 향상 점수들의 분산으로 아래의 식 (10)을 이용하여 계산할 수 있다.

$$\xi_i=\frac{1}{n_i}\sum_{j=1}^{n_i}\Delta_{i,j}^2-\mu_i^2 \tag{10}$$

식 (7)에 의해서 계산된 i번째 부분 문제에 대한 기여도 점수를 활용하여 부분 문제를 선택하면 탐색과 활용의 균형을 효과적으로 유지할 수 있다. 예를 들어, 초기 진화 단계일수록 P_i는 상대적으로 큰 값을 가지므로 식 (7)에서 P_i의 영향력 또한 커진다. 이 경우, 활용보다는 탐색의 원리에 기반하여 부분 문제를 선택한다. 반대로, 진화 과정이 진행됨에 따라 i번째 부분 문제에 대한 적합도 향상 점수의 분산이 지속적으로 감소하므로, P_i의 값은 점차 작아진다. 이 경우, 식 (7)에서 μ_i의 영향력이 상대적으로 증가하여 부분 문제를 선택할 때 활용의 효과가 강화된다. 이에 따라, 진화 단계의 후반부로 진행될수록 부분 문제의 평균 적합도 향상 점수에 기반하여 부분 문제를 선택한다. 즉, 초기 진화 단계에서는 해 탐색의 다양성을 강화하기 위해 탐색 중심의 부분 문제 선택을 수행한다면, 이후에는 빠른 수렴을 위해 점진적으로 "활용"에 기반하여 부분 문제를 선택한다.

3.2 UTSWSP: 슬라이딩 윈도 기반 Non-Stationary UCB-Tuned 알고리즘을 활용한 부분 문제 선택 알고리즘

협력 공진화 알고리즘에서 최적해 탐색이 진행될수록 집단 내의 개체들은 전역 또는 국소 최적점으로 수렴한다. 이에 따라, 각 개체의 적합도 역시 국소 또는 전역 최적해의 적합도와 점차 가까워지고, 결과적으로 각 부분 문제를 선택하여 최적해를 탐색하였을 때 전역 최적해의 적합도 향상의 정도 역시 달라진다. 즉, 부분 문제별 적합도 향상 점수의 분포는 동적으로 변화하며 이는 MAB 알고리즘에서 시간이 지남에 따라 각 Arm이 반환하는 보상의 분포가 변화하는 현상과 동일하다. 이와 같이, 협력 공진화 알고리즘은 최적해 탐색이 진행됨에 따라 부분 문제의 적합도 향상 정도의 분포가 동적으로 변화하는 Non-Stationary 특성을 갖는다.

이에 따라, 진화 알고리즘의 Non-Stationary 특성을 고려하여 부분 문제에 대한 적합도 향상 점수의 평균을 계산하면, 각 부분 문제를 선택하였을 때 얼마나 유의미하게 전역 최적해의 적합도를 향상시킬 수 있었는가를 더욱 정확하게 측정할 수 있다. 즉, 식 (8)~(10)에 적용된 적합도 향상 점수의 평균과 분산을 계산할 때 Non-Stationary 메커니즘을 적용하기 위해 다음과 같이 슬라이딩 윈도[16]를 활용할 수 있다.

3.2.1 슬라이딩 윈도의 설계 방법

목적함수 f의 최대 허용 호출 횟수를 $maxFEs$, 진화 알고리즘 내에서 허용되는 최대 반복 횟수를 ρ, 집단 내 개체의 수를 m, f의 부분 문제의 개수를 K라 하자. 그렇다면, 최적해 탐색 과정에서 각 부분 문제별 평균 선택 횟수 $\tilde{n}$는 다음과 같은 근사식으로 표현할 수 있다.

$$\tilde{n} \cong \left\lceil \frac{maxFEs}{(\rho+1)mK} \right\rceil \tag{11}$$

즉, $maxFEs$가 커짐에 따라 $\tilde{n}$ 또한 이에 비례하여 증가한다. 이 경우 슬라이딩 윈도의 크기 역시 이에 비례하여 확장하는 것이 합리적이다. 반면에, K, m, 또는 ρ의 값이 증가하는 경우 동일한 $maxFEs$ 하에서 $\tilde{n}$는 감소하게 된다. 이 경우에는 각 부분 문제별로 측정되는 적합도 향상 점수의 개수 역시 줄어들므로 슬라이딩 윈도의 크기를 축소하는 것이 바람직하다. 결과적으로, 슬라이딩 윈도의 평균 크기 W는 아래와 같이 $\tilde{n}$에 비례한다.

$$W \propto \tilde{n} \cong \left\lceil \frac{maxFEs}{(\rho+1)mK} \right\rceil \tag{12}$$

따라서, 슬라이딩 윈도의 크기 W의 근사 공식은 $maxFEs$와 ρ, m, K에 관한 식으로 다음과 같이 표현할 수 있다.

$$W \cong \left\lceil \alpha \times \frac{maxFEs}{(\rho+1)mK} \right\rceil \tag{13}$$

식 (13)에서 $\alpha \in (0,1)$는 비례 상수로 슬라이딩 윈도의 실제 크기를 결정하는 제어 파라미터의 역할을 수행한다. 만약 α를 0에 근접한 임의의 작은 값으로 설정한다면 기여도 점수 계산 시 가장 최근의 적합도 향상 점수들이 반영된다. 반면에 1에 가까운 값으로 설정하는 경우, 슬라이딩 윈도의 크기가 이에 비례하여 증가하므로 더 많은 과거 적합도 점수들을 포함하여 기여도 점수를 계산할 수 있다. 다만, α를 1에 근접한 값으로 설정하면 기여도 점수 계산 시 대부분의 과거 적합도 향상 점수들이 포함되어 결과적으로 슬라이딩 윈도를 사용하지 않는 기존의 UCB 또는 UCB-Tuned 기반의 부분 문제 선택 알고리즘[18]과 동일해지는 효과를 초래할 수 있음에 유의해야 한다.

한편, i번째 부분 문제가 선택된 횟수를 n_i라 할 때, $n_i < W$인 경우 슬라이딩 윈도의 크기는 n_i로 설정된다. 이후 진화 알고리즘이 진행됨에 따라 n_i 역시 점차 증가하게 되고 $n_i = W$가 되면 i번째 부분 문제에 대한 슬라이딩 윈도의 최종 크기는 W로 고정된다. 이에 따라, i번째 부분 문제에 대한 슬라이딩 윈도의 크기 W_i를 결정하는 방법을 수식으로 표현하면 다음과 같다.

$$W_i = \begin{cases} n_i & \text{if } n_i < W \\ W & otherwise \end{cases} \quad (14)$$

그림 3은 슬라이딩 윈도를 이용하여 i번째 부분 문제에 대한 적합도 향상 점수의 평균을 계산하는 방법을 설명한다. 그림 3에 나타낸 바와 같이 i번째 부분 문제가 총 n_i번 선택되어 적합도 향상 점수가 계산 및 누적되었다고 가정할 때, 가장 최근에 계산된 결과(리스트의 가장 오른쪽에 표시된 값)를 포함한 최신 W_i개의 결과만을 평균 계산 시 활용한다.

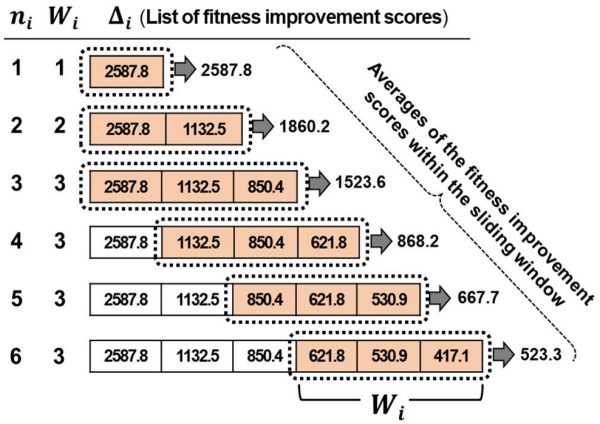


그림 3. 슬라이딩 윈도를 활용하여 적합도 향상 점수의 평균을 계산하는 방법의 예

Fig. 3. Example of a method to average the fitness improvement score by utilizing the sliding window

3.2.2 슬라이딩 윈도를 활용한 Non-Stationary UCB-Tuned 기반 부분 문제 선택 알고리즘의 구현

식 (14)에서 설명한 슬라이딩 윈도 W_i를 활용하여 i번째 부분 문제에 대한 적합도 향상 점수의 평균을 다음과 같이 계산할 수 있다.

$$\mu_i^{(W)} = \frac{1}{W_i} \times \sum_{j=n_i-W_i+1}^{n_i} \Delta_{i,j} \quad (15)$$

즉, i번째 부분 문제에 대한 평균 적합도 향상 점수를 계산하기 위해서 총 n_i개의 적합도 향상 점수 중 가장 최근에 계산된 W_i개의 점수들을 활용한다. 이와 동일한 원리로, i번째 부분 문제에 대한 Padding Function $P_i^{(W)}$는 다음과 같이 계산한다.

$$P_i^{(W)} = \sqrt{\frac{\Psi_i}{W_i} \ln \sum_{r=1}^{K} W_r} \quad (16)$$

식 (16)에서 Ψ_i는 부분 문제 선택 시 진화 단계에 따른 탐색과 활용의 균형을 조정하기 위한 함수로써 아래와 같이 계산할 수 있다.

$$\Psi_i = \min\left(\frac{1}{4}, \xi_i^{(W)} + \sqrt{\frac{2}{W_i} \ln \sum_{r=1}^{K} W_r}\right) \quad (17)$$

상기 식 (17)에서 $\xi_i^{(W)}$는 슬라이딩 윈도를 이용하여 계산되는 적합도 향상 점수의 분산으로 아래의 식을 이용하여 계산한다.

$$\xi_i^{(W)} = \frac{1}{W_i} \times \sum_{j=n_i-W_i+1}^{n_i} \Delta_{i,j}^2 - \left[\mu_i^{(W)}\right]^2 \quad (18)$$

결과적으로 i번째 부분 문제에 대한 최종 기여도 점수 $C_i^{(W)}$의 계산 공식을 정의하면 다음과 같다.

$$C_i^{(W)} = \mu_i^{(W)} + P_i^{(W)} \quad (19)$$

식 (19)를 이용하여 협력 공진화 알고리즘은 매 진화 단계에서 모든 부분 문제에 대한 기여도 점수를 계산한다. 그다음에, 가장 높은 기여도 점수를 갖는 부분 문제를 다음 진화 단계에서 최적해 탐색을 진행할 부분 문제로 선택한다. 즉, 식 (19)에 의해서 계산된 K개의 부분 문제에 대한 기여도 점수를 $C_1^{(W)},...,C_K^{(W)}$라 할 때, 아래의 식 (20)을 이용하여 다음 단계에서 최적해 탐색이 진행될 부분 문제의 인덱스 $i^{(next)}$를 결정한다.

$$i^{(next)} = \arg\max_i \left(C_1^{(W)}, \ldots, C_i^{(W)}, \ldots, C_K^{(W)}\right) \quad (20)$$

식 (14)~(20)을 종합하여 실제 협력 공진화 알고리즘에 적용할 수 있는 부분 문제 선택 방법인 UTSWSP(Non-Stationary UCB-Tuned with Sliding Window-based Subproblem Selector) 알고리즘을 구현할 수 있다. 표 2는 제안하는 UTSWSP 알고리즘의

의사코드를 보여준다. UTSWSP 알고리즘은 먼저 모든 부분 문제를 한 번씩 순차적으로 선택하고 그것의 인덱스를 반환하는 Round-Robin Subproblem Selection을 수행한다. 이는 협력 공진화 알고리즘에서 부분 문제별 기여도를 계산하기 위해서는 최소 하나 이상의 적합도 향상 점수가 필요하기 때문이다. 이후 UTSWSP 알고리즘은 모든 부분 문제에 대한 기여도 점수에 기반하여 다음 단계에서 최적해 탐색을 진행할 부분 문제를 선택한다. 이때, 기여도 점수를 효율적으로 계산하기 위해 각 부분 문제별로 측정된 적합도 향상 점수들을 인덱싱하는 리스트 Δ와 Γ를 크기가 $K \times W$인 K개의 원형 큐(Circular queue)로 설계할 수 있다. 이 경우, 만약 $n_i < W$라면, i번째 부분 문제에 대한 슬라이딩 윈도의 크기 W_i는 식 (14)에 의해 n_i가 되어야 한다. 이를 효율적으로 구현하기 위해서 Δ와 Γ의 모든 성분을 "NULL"로 초기화한다. 그렇다면, $n_i < W$인 경우 Δ와 Γ의 i번째 행에 인덱싱된 값들의 합인 sum($\Delta[i,:]$)와 sum($\Gamma[i,:]$)는 n_i개의 NULL이 아닌 요소들의 합을 계산하는 것과 동일하다. 이처럼 원형 큐를 이용함으로써 식 (15)와 (18)을 계산하는 데 필요한 최신 W개의 적합도 향상 점수만을 효율적으로 유지할 수 있다. 마지막으로, 모든 부분 문제에 대한 기여도 점수 계산을 완료한 후 UTSWSP 알고리즘은 식 (20)을 이용하여 최대 기여도 점수를 갖는 부분 문제의 인덱스를 반환한다.

표 2의 UTSWSP 알고리즘은 진화 알고리즘의 동적 수렴 특성을 반영하기 위해 각 부분 문제별로 식 (14)의 슬라이딩 윈도를 활용하여 식 (15)와 (18)을 계산하며 이를 위해 W 크기의 원형 큐를 이용한다. 이를 통해, 최근 W개의 적합도 향상 점수들을 효율적으로 인덱싱하고 이를 기반으로 기여도 점수를 Incremental 하게 계산할 수 있다. 이러한 방법은 과거의 모든 적합도 향상 점수를 활용하여 기여도를 계산하는 기존의 부분 문제 선택 알고리즘인 [18], [19], [20]과 명확히 구별되는 UTSWSP 알고리즘만의 특징이다. 이를 통해 UTSWSP 알고리즘을 적용한 협력 공진화 알고리즘은 기존의 전통적인 부분 문제 선택 알고리즘을 적용했을 때보다 더욱 높은 품질의 전역 최적해를 효과적으로 탐색할 수 있다. UTSWSP 알고리즘을 실제 협력 공진화 알고리즘에 적용하여 최적해 탐색을 수행하는 방법에 대해서는 이어지는 3.3절에서 자세히 설명한다.

표 2. 제안하는 UTSWSP 알고리즘의 의사코드
Table 2. Pseudocode of proposed UTSWSP algorithm

Algorithm UTSWSP

▪ **Inputs**: &isRoundRobin, &C[], i, K, Δ[], Γ[], n[]

if isRoundRobin = True **then**
  $i \leftarrow i + 1$;
  if $i = K$ **then** isRoundRobin ← False **end**;
else
  $N \leftarrow$ **sum**(n[:]);
  for $k \leftarrow 1$ **to** K **do**
    $\mu \leftarrow$ **sum**($\Delta[k,:]$;) / $n[k]$;
    $v_1 \leftarrow$ **sum**($\Gamma[k,:]$;) / $n[k] - \mu^2$;
    $v \leftarrow v_1 + \sqrt{(2\ln N)/n[k]}$;
    $P \leftarrow \sqrt{((\ln N)/n[k]) \times \min(0.25, v)}$;
    $C[k] \leftarrow \mu + P$;
  end
  $i \leftarrow \arg\max_i(C)$;
end
return i;

• The notation "&" means that the variable is handled by the call by reference.

3.3 실제 협력 공진화 알고리즘 내 UTSWSP 알고리즘의 활용 방법

표 2에서 설명한 UTSWSP 알고리즘은 실제 협력 공진화 알고리즘 내에서 부분 문제 함수로써 활용할 수 있다. 이에 대한 예제 방법을 의사코드로 표현하면 표 3과 같다.

협력 공진화 알고리즘은 주어진 목적함수에 대한 전역 최적해를 탐색하기에 앞서 집단 내 개체들에 대한 초기화를 수행한 후 가장 좋은 적합도를 갖는 개체를 초기 전역 최적해로 지정한다. 다음으로, 식 (13)을 이용하여 슬라이딩 윈도의 크기 W를 결정하고, 부분 문제별 기여도 계산에 필요한 적합도 향상 점수와 부분 문제 선택 횟수 등을 인덱싱할 변수들을 초기화한다. 이후 표 2의 UTSWSP 알고리즘을 이용하여 부분 문제를 선택한 후 해당 부분 문제에 대한 최적해 탐색을 진행하고, 새롭게 진화된 부분 문제의 모든 개체를 기존 집단에 반영(갱신)한

다. 이어서, 식 (6)을 이용하여 해당 부분 문제에 대한 새로운 적합도 향상 점수를 계산하고 변수 $\Delta[i,j]$에 계산 결과를 인덱싱한다. 이 과정에서 해당 부분 문제의 선택 횟수와 새로운 전역 최적해의 적합도 정보도 함께 갱신한다. 이러한 과정은 목적함수의 호출 횟수(FEs)가 최대 허용 호출 횟수($maxFEs$)를 초과할 때까지 반복적으로 수행한다. 마지막으로, 전역 최적해 탐색 과정이 종료되면 현재까지 발견된 전역 최적해를 최종적인 해로서 반환한 후 알고리즘을 종료한다.

표 3. 협력 공진화 알고리즘에서 UTSWSP 알고리즘을 활용하는 예
Table 3. Example of utilizing the UTSWSP Algorithm in the CC framework

Algorithm example CC with the UTSWSP algorithm

- **Inputs**: An objective function $f : R^n \rightarrow R$
- **Parameters**: m, $maxFEs$, α, τ, ρ

$P \leftarrow$ Initialize an $m \times n$ population matrix randomly;
bestInd $\leftarrow \arg\min_{P[i,:]} f(P)$;
bestFit $\leftarrow f$(bestInd); $FEs \leftarrow m$;
$V_1, ..., V_K \leftarrow$ **ProblemDecomposer**(f);
$FEs \leftarrow FEs$ + "FEs used for problem decomposition";
$W \leftarrow \lceil \alpha \times maxFEs / ((\rho+1)mK) \rceil$;
$C, T, n \leftarrow$ Initialize K-dimensional vectors to zeros;
$\Delta, \Gamma \leftarrow$ Initialize $K \times W$ matrices to NULL;
isRoundRobin $\leftarrow$ True; $i \leftarrow 1$;
while $FEs < maxFEs$ **do**
  prevFit $\leftarrow$ bestFit;
  $P[:, V_i] \leftarrow$ **Evolution**($P[:, V_i]$; ρ);
  $FEs \leftarrow FEs$ + "FEs used for Evolution";
  bestInd $\leftarrow \arg\min_{P[i,:]} f(P)$;
  bestFit $\leftarrow f$(bestInd); $FEs \leftarrow m$;
  $T[i] \leftarrow T[i] + 1$;
  $j \leftarrow T[i] \bmod W$;
  if $j = 0$ **then** $j \leftarrow W$ **end**;
  $\Delta[i,j] \leftarrow$ (prevFit - bestFit) / (prevFit + τ);
  $\Gamma[i,j] \leftarrow \Delta[i,j]^2$;
  if $T[i] > W$ **then** $n[i] \leftarrow W$;
  else $n[i] \leftarrow n[i] + 1$ **end**;
  $i \leftarrow$ **UTSWSP**(&isRoundRobin, &C, i, K, Δ, Γ, n);
end
return bestInd;

• The notation "&" means that the variable is handled by the call by reference.

IV. 실험 및 성능평가

4.1 실험 환경 구축 및 성능평가 방법

제안하는 UTSWSP 알고리즘을 실제 협력 공진화 알고리즘에 적용하였을 때, 최적해 탐색 성능을 강화하는데 얼마만큼 기여할 수 있는가를 확인하기 위해 아래와 같이 실험을 설계하여 성능평가를 진행하였다.

먼저, 협력 공진화 알고리즘에서 활용될 기반 진화 알고리즘과 문제 분할 함수로 SaNSDE[21]와 ERDG[22]를 각각 적용하였다. 이와 함께, 실제 협력 공진화 알고리즘의 최적화 성능을 평가하기 위한 벤치마크 목적함수로 CEC'2010 벤치마크 함수 세트[17]에 포함된 17개의 테스트 함수를 채택하였다. CEC'2010 벤치마크 함수 세트에는 1,000차원의 테스트 함수 20개가 포함되어 있으며, ERDG를 이용하여 이들 각각에 대한 문제 분할을 시행한 결과 함수 f_3, f_{19}, f_{20}에 대해서는 2개 이상의 부분 문제로 분할되지 않았다. 따라서, 해당 함수들을 제외한 나머지 17개의 함수(f_1, f_2, f_4~f_{18})에 대해서만 성능평가를 수행하였다.

한편, 제안하는 UTSWSP 알고리즘을 실제 협력 공진화 알고리즘에 적용했을 때 얼마만큼 최적화 성능을 향상시킬 수 있는가를 평가하기 위해 아래의 절차에 따라 실험을 진행하였다.

첫째, UTSWSP 알고리즘과 기존 부분 문제 선택 알고리즘 각각을 협력 공진화 알고리즘에 적용하였다. 이때, 성능 비교를 위한 기존의 협력 공진화 알고리즘으로 Round-robin 기반의 부분 문제 선택 방법을 활용하는 BasicCC 알고리즘[9]과 임의로 부분 문제를 선택하는 Random CC 알고리즘(RandomCC)[9], ε-Greedy 알고리즘을 이용하여 부분 문제를 선택하는 BBCC 알고리즘[19], 그리고 적합도 향상의 정도에 기반하여 부분 문제를 선택하는 CBCC1과 CBCC2 알고리즘[20]을 채택하였다. 표 4는 각각의 부분 문제 선택 알고리즘을 활용하는 협력 공진화 알고리즘들의 명칭을 보여준다.

둘째, 표 4의 협력 공진화 알고리즘 8개를 이용하여 CEC'2010 벤치마크 함수 17개에 대한 전역 최적해 탐색을 수행하였다. 이때, 하나의 테스트 함

수에 대해서 25번 반복적으로 전역 최적해 탐색을 수행하고 이를 통해 발견된 전역 최적해들에 대한 평균 적합도를 계산하였다.

표 4. 8개의 부분 문제 선택 알고리즘을 활용하는 협력 공진화 알고리즘들의 명칭
Table 4. Names of the CC algorithms utilizing each of eight subproblem selection algorithms

Subproblem selection algorithms	CC algorithms
Proposed subproblem selection algorithm (UTSWSP)	UTSWSPCC
UCB algorithm-based subproblem selection algorithm (UCBSP) [18]	UCBSPCC
UCB-tuned-based subproblem selection algorithm (UCBTSP) [18]	UCBTSPCC
Round-robin-based subproblem selection nethod [9]	BasicCC
Random subproblem selection method [9]	RandomCC
ε-Greedy-based subproblem selection algorithm [19]	BBCC
Contribution-based subproblem selection algorithm (Version 1) [20]	CBCC1
Contribution-based subproblem selection algorithm (Version 2) [20]	CBCC2

셋째, 각각의 테스트 함수에 대해서 UTSWSPCC가 발견한 전역 최적해의 평균 적합도와 나머지 7개의 협력 공진화 알고리즘이 발견한 전역 최적해의 평균 적합도를 상호 비교하였다. 이를 위해, 대표적 분산 분석 방법의 하나인 윌콕슨 순위 합 검정(Wilcoxon rank-sum test)을 수행하였으며, 이 과정에서 유의 수준(Significance level) p는 0.05로 설정하고 비교 과정에서 1종 오류의 발생을 예방하기 위해 Bonferroni p-correction을 수행하였다.

마지막으로, UTSWSP와 UTSWSPCC 알고리즘에서 개체의 개수 m은 100, Smoothing Factor τ는 10^{-8}, 목적함수의 최대 허용 호출 횟수 $maxFEs$는 3×10^{6}, SaNSDE에서 사용하는 최대 반복 횟수 ρ는 100으로 설정하고 성능평가를 진행하였다.

4.2 슬라이딩 윈도 크기에 따른 협력 공진화 알고리즘의 최적화 성능 비교 평가

식 (12)에서 설명한 바와 같이 UTSWSP 알고리즘은 슬라이딩 윈도의 크기를 제어하는 파라미터로써 α를 사용한다. 이에 따라, 본 실험에서는 파라미터 α에 대한 최적의 설정값을 찾고, 파라미터 설정에 따른 최적해 탐색 성능을 분석하기 위한 실험을 진행하였다. 이를 위해, α를 0.1에서 1.2까지 설정한 12종의 UTSWSP 알고리즘을 구현하고, 이들 각각을 적용한 협력 공진화 알고리즘의 최적해 탐색 결과를 비교하였다. 이때, 슬라이딩 윈도를 활용한 Non-Stationary 메커니즘이 부분 문제를 선택할 때 얼마만큼 유의미하게 기여할 수 있는가를 평가하기 위해 Non-Stationary 메커니즘을 적용하지 않은 UCB-Tuned 기반의 부분 문제 선택 알고리즘을 활용하는 UCBTSPCC[18]의 최적해 탐색 결과와 비교 평가를 수행하였다.

표 5는 파라미터 α의 변화에 따른 협력 공진화 알고리즘의 최적해 탐색 결과를 보여준다. 표 5에서 "W(Win)"은 UTSWSPCC가 UCBTSPCC보다 더 좋은 최적해 탐색 결과를 달성하였음을 보여준다. 반면에, "L(Lose)"는 UTSWSPCC가 UCBTSPCC보다 더 낮은 최적해 탐색 결과를 보여주었음을 나타낸다. 마지막으로, "T(Tie)"는 UTSWSPCC와 UCBTSPCC가 통계적으로 동등한 결과를 달성하였음을 의미한다.

표 5에 기술된 바와 같이 α를 0.2와 0.4로 설정했을 때 총 10개의 테스트 함수에서 UCBTSPCC보다 더 좋은 최적해 탐색 결과를 보여주었다. 이는 총 12개의 UTSWSPCC 모델 중에서 가장 좋은 최적해 탐색 결과이다. 특히, α를 0.2로 설정한 경우 0.4로 설정했을 때보다 17개의 테스트 함수에 대해서 평균적으로 더 좋은 적합도를 가짐을 확인하였다. 또한, α를 0.3으로 설정하였을 때, 총 9개의 함수에서 "Win"을 달성하였으며 이는 두 번째로 좋은 최적해 탐색 결과이다. 한편, α의 크기가 증가할수록 UTSWSPCC와 UCBTSPCC의 결과가 거의 동등해지는 현상을 발견할 수 있는데, 이는 α의 크기가 커질수록 슬라이딩 윈도의 크기 역시 증가함에 따라 대부분의 적합도 향상 점수들이 참조되기 때문이다.

이러한 일련의 실험 결과는 슬라이딩 윈도의 크기가 상대적으로 작을수록 최적화 성능 향상에 유리함을 보여준다. 즉, 부분 문제별로 더욱 정확하게 기여도를 평가하기 위해서는 비교적 최근에 계산된 적합도 향상 점수들을 활용하는 것이 더욱 효과적

표 5. 슬라이딩 윈도 파라미터 α의 설정에 따른 UTSWSPCC의 최적화 성능평가 결과
Table 5. Optimization performance evaluation results of UTSWSPCC based on the configuration of the sliding window's parameter α

Func.	Measures	UTSWSPCC with the parameter α = 0.1 to 1.2												UCBT-SPCC
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2	
f_1	Mean	6.33E-05	6.71E-01	6.00E-01	1.65E-01	1.89E-01	1.46E-02	4.92E-04	4.86E-04	4.96E-05	4.89E-05	2.80E-05	2.57E-05	2.33E-05
	p-value	5.07E-07	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	8.73E-07	1.34E-06	3.00E-01	1.00E+00	-
	W/T/L	L	L	L	L	L	L	L	L	L	L	T	T	-
f_2	Mean	4.10E+02	3.93E+02	3.88E+02	3.71E+02	3.74E+02	3.12E+02	2.28E+02	2.25E+02	1.20E+02	1.21E+02	1.21E+02	1.17E+02	1.19E+02
	p-value	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.00E+00	1.00E+00	1.00E+00	1.00E+00	-
	W/T/L	L	L	L	L	L	L	L	L	T	T	T	T	-
f_4	Mean	6.76E+09	1.27E+10	1.20E+10	2.61E+10	4.78E+10	5.24E+10	1.49E+11	1.91E+11	3.61E+11	9.22E+11	1.00E+12	1.01E+12	1.59E+12
	p-value	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	9.38E-08	5.95E-03	9.14E-03	4.75E-02	-
	W/T/L	W	W	W	W	W	W	W	W	W	W	W	W	-
f_5	Mean	2.36E+08	1.85E+08	1.98E+08	1.43E+08	1.29E+08	1.37E+08	1.51E+08	1.52E+08	3.22E+08	3.42E+08	3.43E+08	3.31E+08	3.18E+08
	p-value	1.84E-06	1.60E-08	3.69E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.00E+00	4.13E-03	7.39E-03	6.42E-01	-
	W/T/L	W	W	W	W	W	W	W	W	T	L	L	T	-
f_6	Mean	1.11E+01	9.77E+00	1.09E+01	1.10E+01	1.17E+01	1.18E+01	1.27E+01	1.12E+01	1.24E+01	1.26E+01	1.26E+01	1.25E+01	1.38E+01
	p-value	2.21E-02	3.19E-04	2.21E-02	1.29E-02	1.60E-01	3.84E-01	1.00E+00	3.70E-02	1.00E+00	1.00E+00	1.00E+00	9.90E-01	-
	W/T/L	W	W	W	W	T	T	T	W	T	T	T	T	-
f_7	Mean	4.14E+01	1.82E-04	3.89E-01	2.64E-02	1.76E-04	1.82E-04	1.49E-05	1.84E-05	1.41E-05	2.10E-05	1.38E-05	1.74E-05	1.57E-05
	p-value	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.00E+00	1.00E+00	1.00E+00	7.34E-01	1.00E+00	1.00E+00	-
	W/T/L	L	L	L	L	L	L	T	T	T	T	T	T	-
f_8	Mean	3.22E+05	6.38E+05	6.38E+05	4.78E+05	7.97E+05	3.19E+05	7.20E+05	4.79E+05	5.48E+07	5.90E+07	5.96E+07	5.00E+07	3.87E+07
	p-value	1.60E-08	1.58E-08	1.58E-08	1.59E-08	1.57E-08	1.60E-08	1.59E-08	1.60E-08	1.11E-03	2.69E-04	2.69E-04	9.44E-04	-
	W/T/L	W	W	W	W	W	W	W	W	L	L	L	L	-
f_9	Mean	7.05E+06	7.83E+06	8.44E+06	1.01E+07	1.23E+07	1.48E+07	1.48E+07	2.04E+07	2.45E+07	2.66E+07	2.63E+07	2.66E+07	2.59E+07
	p-value	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	7.83E-07	9.90E-01	1.00E+00	1.00E+00	1.00E+00	-
	W/T/L	W	W	W	W	W	W	W	W	T	T	T	T	-
f_{10}	Mean	3.68E+03	3.53E+03	3.44E+03	3.48E+03	3.46E+03	3.44E+03	3.46E+03	3.56E+03	3.65E+03	3.60E+03	3.68E+03	3.65E+03	3.71E+03
	p-value	1.00E+00	4.44E-03	1.66E-06	7.74E-05	1.11E-04	8.57E-06	5.73E-06	1.81E-02	1.00E+00	2.86E-01	1.00E+00	1.00E+00	-
	W/T/L	T	W	W	W	W	W	W	W	T	T	T	T	-
f_{11}	Mean	1.08E+01	1.07E+01	1.05E+01	1.11E+01	1.12E+01	1.10E+01	1.13E+01	1.15E+01	1.12E+01	1.14E+01	1.14E+01	1.15E+01	1.16E+01
	p-value	6.44E-02	3.06E-02	1.52E-03	4.44E-01	1.00E+00	5.87E-01	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	-
	W/T/L	T	W	W	T	T	T	T	T	T	T	T	T	-
f_{12}	Mean	5.50E+00	1.30E+01	1.77E+00	6.38E+00	3.56E+01	1.26E+02	1.24E+02	1.64E+02	7.12E+02	5.95E+02	5.25E+02	5.43E+02	5.62E+02
	p-value	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	2.29E-08	1.59E-02	1.00E+00	1.00E+00	1.00E+00	-
	W/T/L	W	W	W	W	W	W	W	W	L	T	T	T	-
f_{13}	Mean	5.96E+03	4.01E+02	4.35E+02	4.96E+02	5.28E+02	6.14E+02	6.00E+02	7.37E+02	7.04E+02	7.16E+02	7.09E+02	7.14E+02	7.26E+02
	p-value	1.60E-08	4.15E-08	4.67E-08	2.08E-07	9.47E-06	1.70E-02	1.81E-02	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	-
	W/T/L	L	W	W	W	W	W	W	T	T	T	T	T	-
f_{14}	Mean	2.95E+07	2.93E+07	3.05E+07	2.89E+07	3.04E+07	2.99E+07	2.94E+07	3.05E+07	3.00E+07	3.07E+07	3.04E+07	2.95E+07	3.13E+07
	p-value	1.99E-01	1.22E-01	1.00E+00	4.47E-02	1.00E+00	5.87E-01	6.83E-02	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.09E-01	-
	W/T/L	T	T	T	W	T	T	T	T	T	T	T	T	-
f_{15}	Mean	5.54E+03	5.46E+03	5.45E+03	5.40E+03	5.43E+03	5.40E+03	5.39E+03	5.31E+03	5.37E+03	5.38E+03	5.29E+03	5.29E+03	5.39E+03
	p-value	1.60E-01	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	-
	W/T/L	T	T	T	T	T	T	T	T	T	T	T	T	-
f_{16}	Mean	2.39E-01	2.28E-01	1.11E-01	1.93E-01	2.52E-01	1.27E-01	1.77E-01	3.46E-01	2.63E-01	1.11E-01	3.75E-01	3.55E-01	3.38E-01
	p-value	3.47E-02	4.57E-02	5.21E-02	3.82E-02	3.92E-02	5.53E-02	4.19E-02	1.43E-02	2.36E-02	1.00E+00	1.00E+00	1.00E+00	-
	W/T/L	W	W	T	W	W	T	W	L	W	T	T	T	-
f_{17}	Mean	2.60E+02	2.55E+02	2.68E+02	2.31E+02	4.20E+02	3.68E+02	2.47E+02	1.78E+02	1.30E+02	1.23E+02	1.25E+02	1.27E+02	1.27E+02
	p-value	1.60E-08	8.36E-08	1.60E-08	3.26E-07	1.60E-08	1.60E-08	1.60E-08	7.45E-08	1.00E+00	1.00E+00	1.00E+00	1.00E+00	-
	W/T/L	L	L	L	L	L	L	L	L	T	T	T	T	-
f_{18}	Mean	1.13E+03	1.08E+03	1.13E+03	1.12E+03	1.11E+03	1.15E+03	1.16E+03	1.18E+03	1.17E+03	1.12E+03	1.16E+03	1.16E+03	1.13E+03
	p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00	-
	W/T/L	T	T	T	T	T	T	T	T	T	T	T	T	-
Total	Win (W)	7	**10**	9	**10**	8	7	8	7	2	1	1	1	-
	Tie (T)	5	**3**	4	**3**	5	6	6	6	12	13	14	15	-
	Lose (L)	5	**4**	4	**4**	4	4	3	4	3	3	2	1	-

- Win (W): The proposed UTSWSPCC achieved better optimization results than UCBTSPCC.
- Tie (T): Both the UTSWSPCC and UCBTSPCC showed statistically equivalent optimization results.
- Lose (L): The proposed UTSWSPCC produced worse optimization results than UCBTSPCC.

임을 확인할 수 있다. 이에 따라, 0.2를 UTSWSP 알고리즘의 파라미터 α의 최적값으로 채택하였다.

4.3 기존 부분 문제 선택 방법을 적용한 협력 공진화 알고리즘과의 최적화 성능 비교 평가

표 6은 UTSWSPCC와 나머지 7개의 협력 공진화 알고리즘 각각에 의해서 발견된 테스트 함수별 전역 최적해의 평균 적합도와 그들의 비교 결과를 보여준다. 표 5와 유사하게 "W(Win)"는 UTSWSPCC가 비교되는 다른 협력 공진화 알고리즘보다 더 좋은 최적해 탐색 결과를 달성하였음을 의미한다. "T(Tie)"는 UTSWSPCC와 비교되는 협력 공진화 알고리즘이 통계적으로 동등한 최적해 탐색 결과를 달성하였음을 나타내며, "L(Lose)"은 UTSWSPCC가 더 나쁜 탐색 결과를 보여주었음을 의미한다.

표 6의 실험 결과는 제안하는 협력 공진화 알고리즘에 UTSWSP를 적용하였을 때 가장 많은 수의 테스트 함수에서 더욱 향상된 최적해 탐색 결과를 달성하였음을 보여준다. 구체적으로, UTSWSPCC는 UCB 기반의 부분 문제 선택 알고리즘이 적용된 UCBSPCC와 비교했을 때, 9개의 테스트 함수에 대해서 더 좋은 최적해 탐색 결과를 달성하였다. 또한 3개의 테스트 함수에 대해서 통계적으로 동등한 결과를 보여주었다. 이와 유사하게, UCBTSPCC와의 비교에서도 UTSWSPCC가 10개와 3개의 테스트 함수에서 더 좋거나 동등한 최적해 탐색 결과를 달성하였다. 이러한 실험 결과는 협력 공진화 알고리즘에서 최적의 부분 문제를 선택하는 데 있어서 기본적인 UCB 및 UCB-Tuned 알고리즘만을 활용하는 것보다는 슬라이딩 윈도 기반의 Non-Stationary 메커니즘이 적용된 UCB-Tuned 알고리즘을 활용하는 것이 최적해 탐색 성능을 강화하는 데 더욱 효과적임을 보여준다. 이를 통해, 부분 문제별로 기여도를 평가할 때 진화 알고리즘의 동적인 수렴 특성을 고려하여 적합도 향상 점수의 평균과 분산을 계산하는 것이 협력 공진화 알고리즘의 최적해 탐색 성능을 향상하는데 유의미하게 기여할 수 있음을 확인할 수 있다.

한편, 전통적인 협력 공진화 알고리즘과의 비교에서도 UTSWSPCC는 가장 좋은 최적해 탐색 결과를 보여주었다. BasicCC 및 RandomCC와의 비교에서, UTSWSPCC는 11개와 16개의 함수에서 더 좋은 최적화 결과를 달성하였다. 또한, CBCC1 및 CBCC2와의 비교에서도 UTSWSPCC는 12개의 테스트 함수에 대해서 더 좋은 최적해 탐색 성능을 보여주었다. 마지막으로, MAB 알고리즘 중 ε-greedy 알고리즘을 사용하여 부분 문제를 선택하는 BBCC 알고리즘과의 비교에서는 11개의 테스트 함수에 대해서 UTSWSPCC가 더 좋은 최적해 탐색 결과를, 4개의 함수에 대해서 동등한 결과를 달성하였다. 이러한 실험 결과는 협력 공진화 알고리즘에서 부분 문제 선택을 위해 MAB 알고리즘을 활용하는 경우라 할지라도 슬라이딩 윈도 기반의 Non-Stationary UCB-Tuned 알고리즘에 기반하여 부분 문제를 선택하는 것이 실제 최적해 탐색 성능을 강화하는 데 더욱 효과적임을 보여준다.

V. 결 론

협력 공진화 알고리즘에서 부분 문제 선택은 전역 최적해 탐색 성능에 중요한 영향을 미치는 과정으로, 탐색과 활용의 균형을 유지하면서 부분 문제를 선택하는 것이 무엇보다도 중요하다. 이에 따라, 본 논문에서는 협력 공진화 알고리즘에서 부분 문제 선택을 위한 방법으로 슬라이딩 윈도 기반의 Non-Stationary UCB-Tuned 알고리즘을 활용하는 방법을 고안하고 이를 토대로 새로운 부분 문제 선택 알고리즘인 UTSWSP 알고리즘을 제안하였다. UTSWSP 알고리즘은 슬라이딩 윈도를 이용하여 각 부분 문제에 대하여 최근에 측정된 적합도 향상 점수들을 추출하고, 이를 토대로 적합도 향상 점수의 평균과 분산을 계산한다. 이를 통해, 실시간으로 전역 최적해를 탐색하는 동적인 환경에서도 각 부분 문제에 대한 기여도를 효과적으로 계산할 수 있다.

실제 1,000차원의 벤치마크 함수를 대상으로 성능 평가를 진행한 결과 제안하는 UTSWSP 알고리즘이 적용된 협력 공진화 알고리즘이 가장 우수한 성능으로 전역 최적해를 탐색할 수 있음을 확인하였다.

이러한 실험 결과는 부분 문제별로 기여도 점수를 계산하기 위해서 전통적인 UCB-Tuned 알고리즘만을 활용하는 것보다는 슬라이딩 윈도를 이용하여

표 6. UTSWSP 알고리즘 및 다른 부분 문제 선택 방법을 적용한 협력 공진화 알고리즘의 최적화 성능평가 결과
Table 6. Optimization performance evaluation results of the CC algorithms utilizing the UTSWSP algorithm and other subproblem selection methods

Func.	Measures	UTSWSPCC	UCBSPCC	UCBTSPCC	BasicCC	RandomCC	BBCC	CBCC1	CBCC2
f_1	Mean	6.71E-01	1.50E-05	2.33E-05	3.42E-02	9.39E+06	1.70E+07	3.36E-02	3.47E-02
	p-value	-	9.33E-09	9.33E-09	1.13E-02	9.33E-09	9.33E-09	1.13E-02	1.13E-02
	W/T/L	-	L	L	L	W	W	L	L
f_2	Mean	3.93E+02	1.17E+02	1.19E+02	1.18E+02	6.83E+02	5.03E+02	1.58E+02	3.11E+02
	p-value	-	9.33E-09	9.33E-09	9.33E-09	9.33E-09	5.47E-08	9.33E-09	5.67E-07
	W/T/L	-	L	L	L	W	W	L	L
f_4	Mean	1.27E+10	3.43E+12	1.59E+12	1.00E+13	1.17E+13	1.02E+10	4.01E+12	7.69E+09
	p-value	-	9.33E-09	9.33E-09	9.33E-09	9.33E-09	1.00E+00	9.33E-09	5.04E-02
	W/T/L	-	W	W	W	W	T	W	T
f_5	Mean	1.85E+08	3.57E+08	3.18E+08	3.96E+08	4.09E+08	1.09E+08	3.14E+08	1.78E+08
	p-value	-	9.33E-09	9.33E-09	9.33E-09	9.33E-09	9.33E-09	1.19E-08	1.00E+00
	W/T/L	-	W	W	W	W	L	W	T
f_6	Mean	9.77E+00	1.42E+01	1.38E+01	1.53E+01	1.53E+01	1.79E+01	1.54E+01	1.21E+01
	p-value	-	3.12E-05	1.86E-04	1.70E-07	7.72E-08	9.33E-09	1.70E-07	1.29E-02
	W/T/L	-	W	W	W	W	W	W	W
f_7	Mean	1.82E-04	3.43E+07	1.57E-05	2.15E+09	6.00E+09	5.05E+03	1.56E+08	5.18E-03
	p-value	-	9.33E-09	9.33E-09	9.33E-09	9.33E-09	9.33E-09	9.33E-09	2.15E-08
	W/T/L	-	W	L	W	W	W	W	W
f_8	Mean	6.38E+05	6.32E+07	3.87E+07	2.27E+08	1.41E+11	4.84E+05	7.05E+07	4.78E+05
	p-value	-	9.25E-09	9.25E-09	9.25E-09	9.25E-09	9.23E-05	9.25E-09	1.56E-05
	W/T/L	-	W	W	W	W	L	W	L
f_9	Mean	7.83E+06	3.12E+07	2.59E+07	3.79E+07	5.79E+07	1.03E+07	3.61E+07	1.80E+09
	p-value	-	9.33E-09	9.33E-09	9.33E-09	9.33E-09	7.02E-07	9.33E-09	9.33E-09
	W/T/L	-	W	W	W	W	W	W	W
f_{10}	Mean	3.53E+03	3.62E+03	3.71E+03	4.20E+03	4.58E+03	3.40E+03	4.10E+03	3.98E+03
	p-value	-	1.84E-01	2.59E-03	9.33E-09	9.33E-09	4.10E-01	9.33E-09	2.42E-08
	W/T/L	-	T	W	W	W	T	W	W
f_{11}	Mean	1.07E+01	1.18E+01	1.16E+01	1.17E+01	1.17E+01	1.14E+01	1.19E+01	1.23E+01
	p-value	-	3.73E-03	1.79E-02	2.59E-03	1.53E-03	1.00E+00	3.36E-04	2.84E-05
	W/T/L	-	W	W	W	W	T	W	W
f_{12}	Mean	1.30E+01	1.99E+03	5.62E+02	5.28E+03	2.31E+04	4.74E+03	4.57E+03	1.60E+04
	p-value	-	9.33E-09	9.33E-09	9.33E-09	9.33E-09	9.33E-09	9.33E-09	9.33E-09
	W/T/L	-	W	W	W	W	W	W	W
f_{13}	Mean	4.01E+02	8.92E+02	7.26E+02	1.32E+03	1.11E+07	3.67E+03	1.23E+03	2.03E+03
	p-value	-	1.19E-08	2.42E-08	9.33E-09	9.33E-09	9.33E-09	1.19E-08	9.33E-09
	W/T/L	-	W	W	W	W	W	W	W
f_{14}	Mean	2.93E+07	3.00E+07	3.13E+07	3.02E+07	3.74E+07	4.84E+07	3.27E+07	7.33E+09
	p-value	-	1.00E+00	7.10E-02	1.00E+00	4.10E-07	1.09E-07	3.00E-03	9.33E-09
	W/T/L	-	T	T	T	W	W	W	W
f_{15}	Mean	5.46E+03	5.25E+03	5.39E+03	5.34E+03	5.95E+03	5.48E+03	5.42E+03	5.90E+03
	p-value	-	1.29E-02	1.00E+00	4.67E-01	7.10E-05	1.00E+00	1.00E+00	2.79E-03
	W/T/L	-	L	T	T	W	T	T	W
f_{16}	Mean	2.28E-01	1.58E-01	3.38E-01	4.40E-01	1.96E-01	8.46E-01	3.04E-01	3.45E-01
	p-value	-	2.03E-05	2.67E-02	3.58E-01	2.40E-03	2.23E-03	2.76E-02	7.56E-04
	W/T/L	-	L	W	T	L	W	W	W
f_{17}	Mean	2.55E+02	1.33E+02	1.27E+02	1.34E+02	8.36E+02	1.46E+04	1.95E+02	1.55E+03
	p-value	-	1.70E-07	4.88E-08	7.72E-08	3.06E-08	1.19E-08	4.23E-02	9.33E-09
	W/T/L	-	L	L	L	W	W	L	W
f_{18}	Mean	1.08E+03	1.16E+03	1.13E+03	1.19E+03	1.22E+03	3.46E+03	1.15E+03	1.45E+03
	p-value	-	1.67E-01	6.80E-01	2.45E-02	5.33E-03	1.19E-08	2.03E-01	3.30E-07
	W/T/L	-	T	T	W	W	W	T	W
Total	**Win (W)**	-	**9**	**10**	**11**	**16**	**11**	**12**	**12**
	Tie (T)	-	3	3	3	0	4	2	2
	Lose (L)	-	5	4	3	1	2	3	3

- Win (W): The proposed UTSWSPCC achieved better optimization results than the compared CC algorithm.
- Tie (T): Both UTSWSPCC and the compared CC algorithm showed statistically equivalent optimization results.
- Lose (L): The proposed UTSWSPCC yielded worse optimization results than the compared CC algorithm.

가장 최근의 적합도 향상 점수들을 추출하고 이들을 활용하여 부분 문제의 기여도를 측정하는 것이 더욱 합리적임을 보여준다.

그럼에도 불구하고, 본 연구는 여전히 개선해야 할 점이 존재한다. 특히, 슬라이딩 윈도 크기의 비율을 제어하는 파라미터 α의 값을 적응적으로 결정하는 방법에 관한 연구가 우선으로 진행되어야 한다. 이에 따라, 향후 연구에서는 슬라이딩 윈도의 크기를 자동으로 정밀하게 결정하는 적응적 슬라이딩 윈도 제어 방법(Adaptive sliding window control method)을 연구하고, 이를 토대로 더욱 고도화된 부분 문제 선택 알고리즘을 개발하여 최신 부분 문제 선택 알고리즘과 함께 성능 비교 평가를 진행할 예정이다.

References

[1] A. Risanthia, T. Phiboon, S. Bureerat, and A. Pichitkul, "UAV Wing Design via Efficient Global Optimization", Proc. 16th International Conference on Knowledge and Smart Technology (KST), Krabi, Thailand, pp. 62-66, Mar. 2024. https://doi.org/10.1109/KST61284.2024.10499662.

[2] L. D. Austero, A. M. Sison, J. B. Matias, and R. P. Medina, "Solving course timetabling problem using Whale Optimization Algorithm", Proc. 8th International Conference on Information Technology Trends (ITT), Dubai, United Arab Emirates, pp. 160-166, May 2022. https://doi.org/10.1109/ITT56123.2022.9863951.

[3] M. J. Han, "System Development for an Antifreeze Protein Production Using Optimization Analysis of Computer-Based OptimumGene™ Codon", The Journal of The Institute of Internet, Broadcasting and Communication (IIBC), Vol. 25 No. 3, pp. 127-134, Jun. 2025. https://doi.org/10.7236/JIIBC.2025.25.3.127.

[4] K. Shin and Y. A. Min, "Performance Optimization of Reinforcement Learning in Line Tracking Robots using ML-Agents: A Comparative Study of Reward Strategies and Learning Parameters", The Journal of The Institute of Internet, Broadcasting and Communication (IIBC), Vol. 25, No. 3, pp. 57-66, Jun. 2025. https://doi.org/10.7236/JIIBC.2025.25.3.57.

[5] B. Alhijawi and A. Awajan, "Genetic algorithms: Theory, genetic operators, solutions, and applications", Evolutionary Intelligence, Vol. 17, No. 3, pp. 1245-1256, Jun. 2024. https://doi.org/10.1007/s12065-023-00822-6.

[6] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art", IEEE Transactions on Evolutionary Computation, Vol. 15, No. 1, pp. 4-31, Feb. 2011. https://doi.org/10.1109/TEVC.2010.2059031.

[7] F. Wang, X. Wang, and S. Sun, "A reinforcement learning level-based particle swarm optimization algorithm for large-scale optimization", Information Sciences, Vol. 602, pp. 298-312, Jul. 2022. https://doi.org/10.1016/j.ins.2022.04.053.

[8] H. Song, J. Bei, H. Zhang, J. Wang, and P. Zhang, "Hybrid algorithm of differential evolution and flower pollination for global optimization problems", Expert Systems with Applications, Vol. 237, Part A, pp. 121402, Mar. 2024. https://doi.org/10.1016/j.eswa.2023.121402.

[9] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization", Proc. 3rd Conference on Parallel Problem Solving from Nature, Jerusalem, Israel, pp. 249-257, Oct. 1994. https://doi.org/10.1007/3-540-58484-6_269.

[10] X. Ma, X. Li, Q. Zhang, K. Tang, Z. Liang, and W. Xie, "A survey on cooperative co-evolutionary algorithms", IEEE Transactions on Evolutionary Computation, Vol. 23, No. 3, pp. 421-441, Jun. 2019. https://doi.org/10.1109/TEVC.2018.2868770.

[11] K. S. Kim and Y. S. Choi, "An efficient variable interdependency-identification and decomposition by minimizing redundant computations for large-scale global optimization", Information Sciences, Vol. 513, pp. 289-323, Mar. 2020. https://doi.org/10.1016/j.ins.2019.10.049.

[12] M. N. Omidvar, X. Li, Y. Mei, and X. Yao,

"Cooperative co-evolution with differential grouping for large scale optimization", IEEE Transactions on Evolutionary Computation, Vol. 18, No. 3, pp. 378-393, Jun. 2014. https://doi.org/10.1109/TEVC.2013.2281543.

[13] A. H. A. Hanif, S. Das, and I. Ismail, "Fundamental Tradeoffs Between Exploration and Exploitation Search Mechanisms", Into a Deeper Understanding of Evolutionary Computing: Exploration, Exploitation, and Parameter Control, Springer Nature Switzerland, Vol. 50, pp. 101-199, 2024. https://doi.org/10.1007/978-3-031-74013-8_2.

[14] A. Anandkumar and R. Ge, "Efficient approaches for escaping higher order saddle points in non-convex optimization", Proc. 29th Annual Conference on Learning Theory. PMLR, New York, USA, Vol. 49, pp. 81-102, Jun. 2016. https://proceedings.mlr.press/v49/anandkumar16.html/.

[15] P. Auer, N. C. Bianchi, and P. Fischer, "Finite-time analysis of the multi-armed bandit problem", Machine learning, Vol. 47, No. 2, pp. 235-256, May 2002. https://doi.org/10.1023/A:1013689704352.

[16] A. Garivier and E. Moulines, "On upper-confidence bound policies for non-stationary bandit problems", arXiv preprint arXiv:0805.3415, May 2008. https://doi.org/10.48550/arXiv.0805.3415.

[17] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, "Benchmark functions for the CEC'2010 special session and competition on large-scale global optimization", Nature inspired computation and applications laboratory, USTC, China, Technical Report, Jan. 2010. https://github.com/P-N-Suganthan/CEC-2010-LSO-Large-Scale-Opt-/. [accessed: Aug. 21, 2025]

[18] K. S. Kim, "Multi-Armed Bandit-based Cooperative Co-Evolutionary Algorithm for Large-Scale Global Optimization of Non-Linear Functions", The Journal of Korean Institute of Information Technology, Vol. 22, No. 5, pp. 115-129, May 2024. https://doi.org/10.14801/jkiit.2024.22.5.115.

[19] B. Kazimipour, M. N. Omidvar, A. K. Qin, X. Li, and X. Yao, "Bandit-based cooperative coevolution for tackling contribution imbalance in large-scale optimization problems", Applied Soft Computing, Vol. 76, pp. 265-281, Mar. 2019. https://doi.org/10.1016/j.asoc.2018.12.007.

[20] M. N. Omidvar, X. Li, and X. Yao, "Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms", Proc. 13th Annual Conference on Genetic and Evolutionary Computation, Dublin, Ireland, pp. 1115-1122, Jul. 2011. https://doi.org/10.1145/2001576.2001727.

[21] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search", Proc. IEEE Congress on Evolutionary Computation, Hong Kong, China, pp. 1110-1116, Jun. 2008. https://doi.org/10.1109/CEC.2008.4630935.

[22] M. Yang, A. Zhou, C. Li, and X. Yao, "An efficient recursive differential grouping for large-scale continuous problems", IEEE Transactions on Evolutionary Computation, Vol. 25, No. 1, pp. 159-171, Feb. 2021. https://doi.org/10.1109/TEVC.2020.3009390.

저자소개

김 경 수 (Kyung-Soo Kim)

2011년 2월 : 국립목포대학교 사범대학 컴퓨터교육과(공학사)
2020년 8월 : 한양대학교 대학원 전자컴퓨터통신공학과(공학박사)
2020년 9월 ~ 2022년 2월 : 한양대학교 컴퓨테이셔널 사회과학 연구센터 박사후연구원 (Post-Doc.)
2022년 3월 ~ 현재 : 국립금오공과대학교 컴퓨터공학부 컴퓨터공학전공 교수
관심분야 : 인공지능, 인공지능 수학적 모델 고도화 및 응용, 비선형 최적화