Search Economics for Single-Objective Real-Parameter Optimization

Chun-Wei Tsai
Department of Computer Science and Engineering
National Sun Yat-sen University
Kaohsiung, Taiwan
cwtsai@mail.cse.nsysu.edu.tw

Shin-Jui Liu
Department of Computer Science and Engineering
National Chung Hsing University
Taichung, Taiwan
knight31241@gmail.com

***Abstract*—An effective search algorithm for solving the continuous global optimization problem based on a new metaheuristic algorithm named search economics is presented in this paper. The proposed algorithm works by first dividing the solution space into a set of subspaces and then assigning searchers to these subspaces based on the so-called "expected value" of each subspace. The expected value contains not only objective values of searched solutions but also objective values of new probe solutions and computation costs invested in each subspace. This makes it possible for the proposed algorithm to dynamically adjust the computing resources invested in each subspace during the convergence process. To evaluate the performance of the proposed algorithm, we compare it with four state-of-the-art search algorithms; namely, jSO, EBOwithCMAR, ELSHADE-SPACMA, and L-SHADE-RSP, for solving 30 CEC 2017 benchmark functions. The experimental results show that the proposed algorithm can find better results than all the other search algorithms compared in this study, especially for complex, high-dimensional functions, hybrid functions, and composition functions.**

***Index Terms*—Optimization problem, search economics, and metaheuristic algorithm.**

I. Introduction

The single-objective real-parameter optimization problem (SOP) is a well-known global optimization problem, which can be found in many real-world scientific and engineering applications in our daily life [1]–[3]. Since the solution space of this problem is continuous. Formally, a single-objective real-parameter optimization problem [4] can be formulated as:

$$s^* := \min_{s \in \mathbb{A}} f(s), \quad s = [s_1, s_2, \ldots, s_D],$$

where D is the number of dimensions; $\mathbb{A}$ is the feasible set of decision vectors; s^* the global optimum the d-th dimension of which is in the range $[L_d, U_d]$ where L_d and U_d represent, respectively, the lower and upper bounds of the d-th dimension, e.g., $L_d = -100$ and $U_d = 100$.

Being complex and time-consuming, how to find a "good solution" for the single-objective real-parameter optimization problem more effectively and efficiently has become an active research topic in recent years. Most deterministic search algorithms are built on the divide-and-conquer principle for solving the SOP which can be dated back to the late 1960s [5] or even earlier. One of the typical deterministic search algorithms based on the divide-and-conquer principle in the early stage of the search algorithm development for this kind of optimization problem is the so-called branch-and-bound algorithm [5]–[7].

In addition to the deterministic search algorithm, several recent studies [8], [9] have attempted to develop a more efficient way to search for better solutions based on metaheuristic algorithms due to the fact that most metaheuristic algorithms are capable of finding better solutions than deterministic search algorithms for solving the SOP, for each of them uses a different way to avoid falling into local optimum at the early stage of the convergence process. However, most metaheuristic algorithms for solving the SOP inherit the search that would move the searches toward particular directions or specific regions at the later stage of the convergence process. This is unavoidable because most metaheuristic algorithms require the searches to converge to regions that have a higher chance to find better results instead of a random walk in the solution space. Unfortunately, when the searches are limited to specific directions or regions, it is very likely to fall into local optimum, the search diversity for most metaheuristic algorithms will then be decreased; thus, the possibility of finding better solutions will also be degraded at the later stage of the convergence process.

To avoid this phenomenon from happening, an effective metaheuristic algorithm, called search economics for single-objective real-parameter optimization problem (SE-SOP), is presented in this paper. As the name suggests, the SE-SOP is built on the search economics (SE) [10]. The basic idea of SE-based algorithms is to take into account the potentials and investments of each region in the solution space to make every search more meaningful. The main difference between the proposed algorithm and SE is in that we first modify the way the solution space is divided, the way the expected value is computed, and the way the transition of SE is performed. We then add the self-adaptive parameter and linear population size reduction (LPSR) to the proposed algorithm to further improve its performance.

The remainder of this paper is organized as follows: Section II begins with the basic idea of the proposed algorithm, followed by a detailed description of the proposed algorithm for the single-objective real-parameter optimization problem. In Section III, the datasets and parameter settings are first given, and then the simulation results are addressed. Section IV

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

provides a brief introduction to search algorithms for the single-objective real-parameter optimization problem. Finally, the conclusion and some possible future research directions are given in Section V.

II. The Proposed Algorithm

A. The Basic Idea

The serial investigation of SE [10]–[12] is inspired by the observation of [13] which says that most metaheuristic algorithms at the later stage of the convergence process all typically face the dilemma of converging to particular regions to stabilize the search. Motivated by this observation, the basic idea of SE is to use the return on investment (ROI) to measure every search during the convergence process; that is, the aim in the design of the SE is to make every search as meaningful as possible, such as avoiding searching similar or same candidate solutions in a region for a long time.

Unlike all the other metaheuristic algorithms, the SE relies on three important mechanisms to enhance its search performance. They are: (1) divide the solution space into a set of *regions*, (2) use the *expected value of region* instead of the objective and/or fitness value of solution to determine later search directions, and (3) use a new transition operator to generate new probe solutions based on the searched solution of searcher and the sampling solution of region. Because most SE-based algorithms [10]–[12] are designed for combinatorial single-objective real-parameter optimization problem; that is, problems the solution space of which are discrete, the proposed algorithm thus has to first modify the way the solution space is divided, the way the expected value is computed, and the way the transition is performed in SE and then add the self-adaptive parameter and linear population size reduction to SE-SOP to make it not only work, but also enhance its performance, for the single-objective real-parameter optimization problem.

B. The SE-SOP

To simplify the discussion that follows, the following notation is used throughout the rest of the paper for the proposed algorithm.

- r set of regions; i.e., $r = \{r_1, r_2, \ldots, r_h\}$, where r_j is the j-th region and h the number of regions.
- s set of solutions (also known as searchers); that is, $s = \{s_1, s_2, \ldots, s_n\}$, where s_i is the i-th searcher and n the number of searchers.
- m set of solutions (called goods) in all regions; that is, $m = \{m_{11}, \ldots, m_{1w}, \ldots, m_{h1}, \ldots, m_{hw}\}$, where m_{jk} is the k-th goods in the j-th region and w the number of goods in each region.
- r_j^b best so far solution in the j-th region r_j.
- v_j^i set of possible probe solutions (called investments) of the i-th searcher s_i, generated by applying the transition operator to s_i and all goods in the j-th region m_j. That is, $v_j^i = \{v_{j1}^i, \ldots, v_{jm}^i\}$ where $v_{jk}^i = s_i \otimes m_{jk}$ where $\otimes$ is the transition operator.
- e set of expected values, where e_j^i indicates the expected value of the i-th searcher in the j-th region.
- t_j^a number of times the j-th region has been searched. Initially, t_j^a is set equal to 1 and will be increased by one every time the j-th region is searched by a searcher.
- t_j^b number of times the j-th region has not been searched. Initially, t_j^b is set equal to 1 and will be increased by one every time the j-th region is not searched; otherwise, it will be reset to 1 when the region is searched by a searcher.

Figure 1 gives a simple example to illustrate the concept of searchers, goods, and regions of the SE. In this example, the solution space is divided into 4 regions; i.e., $h = 4$. Four solutions are then randomly created as the searchers each of which is assigned to a region at the very beginning. Another type of solution is also randomly generated as the goods. In this example, two goods are randomly generated for each region; thus, there are total 8 goods, which can be regarded as the sampling points to depict the landscape of all the regions for the proposed algorithm.

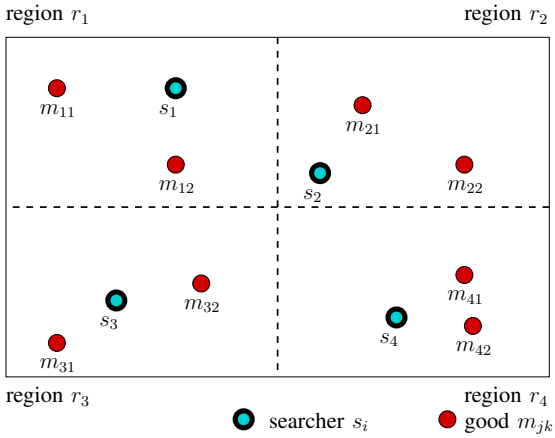

Fig. 1: A simple example to illustrate the basic idea of searchers, goods, and regions of SE.

Algorithm 1 SE-SOP

```
1 Initialization()
2 ResourceArrangement()
3 While the termination criterion is not met
4     VisionSearch()
5     MarketingResearch()
6     LinearPopulationReduction()
7 End
8 Output the end results.
```

As shown in Algorithm 1, in addition to the Initialization() operator, the proposed algorithm contains the ResourceArrangement(), VisionSearch(), MarketingResearch(), and LinearPopulationReduction() operators. The Initialization() operator plays the role of initializing the parameters of search economics for single-objective real-parameter optimization problem and generating a set of solutions as searchers. The ResourceArrangement() operator is responsible for dividing

the solution space into a set of regions. Then, it will assign searchers regions and generate a set of goods for each region. The VisionSearch() operator takes care of transiting the searched solutions to generate new candidate solutions, evaluating the solutions generated, and determining the search directions and regions at later iterations. The MarketingResearch() operator is used to record the status of searches (i.e., investments) in each region to provide more complete information to the VisionSearch() operator to avoid the searches from moving toward particular directions and regions. The LinearPopulationReduction() operator is used to reduce the number of regions, searchers, and goods to increase the local search ability as the number of searches increases, especially at the later stage of the convergence process.

C. Initialization

Like the SE, the proposed algorithm will first initialize the following parameters; namely, the number of regions h, the number of searchers n, and the number of goods in a region w. Unlike the SE, the additional parameters added to SE-SOP and initialized by this operator are s_p, m_p $\mathcal{T}_p$, $\mathcal{H}$, $\mathtt{s}_i^c$, $\mathtt{s}_i^f$, $\mathtt{m}^c$, and $\mathtt{m}^f$. SE-SOP will generate $n = s_p \times D$ searchers, where s_p is a predefined number for adjusting the number of searchers, and D the number of dimensions. When $n > h$, each region will be assigned either $\lceil n/h \rceil$ or $\lfloor n/h \rfloor$ searchers. m_p denotes the ratio of goods and $w = n \times m_p/h$. $\mathcal{T}_p$ is the rate of tournament selection for the creation of the new investment v_j^i. $\mathcal{H}$ is the size of the historical memory. This means that the proposed algorithm will keep a set of feasible strategies (i.e., $\mathtt{s}_i^c$ and $\mathtt{s}_i^f$) for generating the new investments. $\mathtt{m}^c = \{\mathtt{m}_1^c, \ldots, \mathtt{m}_{\mathcal{H}}^c\}$ is a set of transition rates for using either searcher or goods to create new solutions in the historical memory while $\mathtt{m}^f = \{\mathtt{m}_1^f, \ldots, \mathtt{m}_{\mathcal{H}}^f\}$ is a set of weights for the searcher and goods.

Inspired by SHADE [14], the way a new solution is created makes it possible for the proposed algorithm to find a better transition rate at the early stage of the convergence process and to adjust the search strategy at the later stage of the convergence process by itself. As shown in Figure 2, the transition strategies—best/random, $\mathtt{s}_i^c$, and $\mathtt{s}_i^f$—are associated with the i-th searcher s_i.

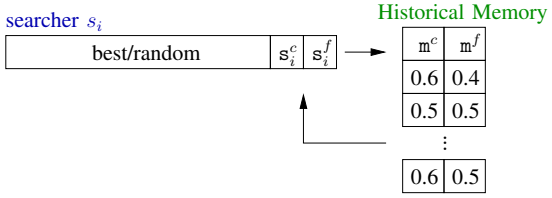

Fig. 2: A simple example to illustrate the encoding of searcher and the relationship between the historical memory and success parameters.

The strategies that have been used before are kept in the historical memory, and those that eventually improve the results are stored in the table of success parameters. This means that the strategies $\mathtt{s}_i^c$ and $\mathtt{s}_i^f$ will be saved in the historical memory by the MarketingResearch() operator if they eventually improve the solution. The VisionSearch() operator can then randomly select a pair of strategies out of the historical memory to increase the possibility of getting a better solution at later iterations.

D. Resource Arrangement

As shown in Figure 3, unlike the SE for which none of the regions are allowed to overlap, the SE-SOP allows all the regions to overlap to avoid searchers from getting stuck at the boundary. In this example, the solution space is divided into four regions. The width and height of each region, say r_1, are extended from 50% to 60% so that each of which has a 20% overlap with regions next to it. As mentioned in Section II-C, this operator will randomly create a certain number of searchers and assign them to all the regions. Besides, the proposed algorithm will randomly create a certain number of goods for each region.

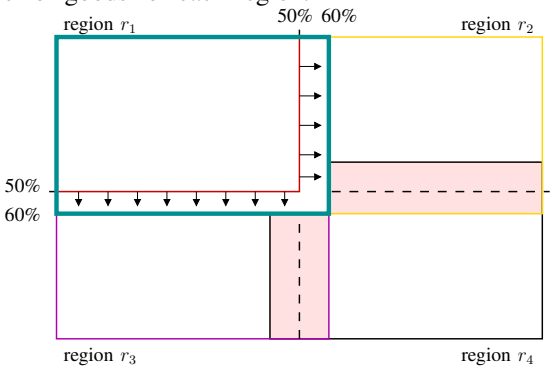

Fig. 3: The basic idea of SE.

E. Vision Search

The basic idea of the transition operator VisionSearch() of SE is to create new solutions v_j^i by exchanging information between the i-th searcher s_i and the goods m in the j-th regions. A pair of strategies $\mathtt{s}_i^c$ and $\mathtt{s}_i^f$ will first be randomly selected from the historical memory. Then, $\mathtt{s}_i^c$ will be fine-tuned in the range $[0, 1]$ based on a normal distribution the standard deviation of which is 0.1 while $\mathtt{s}_i^f$ will be fine-tuned in the range $[0.5, 0.6]$ based on a Cauchy distribution the standard deviation of which is 0.01. Moreover, for $\mathtt{s}_i^f$, when the number of evaluations used so far t_{curr} is larger than $t_{\max} \times c_1$, the range will be changed to $[0, 1]$ and the standard deviation to 0.1, where $t_{\max}$ is the maximum number of evaluations and c_1 is a predefined number. If this operator chooses to select the best solution, then the new investment (solution) v_j^i will be created as follows:

$$v_{jd}^i = \begin{cases} s_{id} + \mathtt{s}_i^f (r_j^{b_1} - s_{id}) + \mathtt{s}_i^f (r_j^{b_2} - m_{jk}^d) & \text{if } \mathtt{r} \leq s_i^c \text{ or } d = l, \\ s_{id} & \text{otherwise,} \end{cases} \tag{1}$$

where v_{jd}^i is the d-th dimension of the new investment solution created from the i-th searcher and the goods in the j-th region. $r_j^{b_1}$ and $r_j^{b_2}$ are, respectively, the best and second best goods selected by tournament selection with $\mathcal{T}$ players where $\mathcal{T}$ is defined as

$$\mathcal{T} = \begin{cases} \lfloor \mathcal{T}_p \times n \times m_p / w \rfloor, & \text{if } t_{\text{curr}} < 0.8 \times t_{\max}, \\ \lfloor 2 \times \mathcal{T}_p \times n \times m_p / w \rfloor, & \text{otherwise,} \end{cases} \tag{2}$$

where t_{curr} is the number of evaluations used so far, and $t_{\max}$ the maximum number of evaluations. m^d_{jk} represents the k-th goods randomly selected from the j-th region; $\mathtt{r}$ represents a random number uniformly distributed in the range $[0, 1]$; and l denotes a particular dimension randomly selected from $\{1, 2, \ldots, D\}$.

If this operator chooses to randomly select a solution, then the new investment (solution) v^i_j will be created as follows:

$$v^i_{jd} = \begin{cases} m^d_{jk_1} + \mathtt{s}^f_i(r^{b_1}_j - m^d_{jk_1}) + \mathtt{s}^f_i(r^{b_2}_j - m^d_{jk_2}) & \text{if } \mathtt{r} \le s^c_i \text{ or } d = l, \\ m^d_{jk_1} & \text{otherwise,} \end{cases} \tag{3}$$

where $m^d_{jk_1}$ and $m^d_{jk_2}$ are two different goods randomly selected from the j-th region; that is, $k_1 \neq k_2$.

SE-SOP uses a way that is similar to SE to compute the expected value e^i_j of the i-th searcher in the j-th region, as follows:

$$e^i_j = \mathcal{N}(t_j) \cdot \mathcal{N}(v^i_j) \cdot (1 - \mathcal{N}(m_j)), \tag{4}$$

where $\mathcal{N}(x)$ is a normalization function that will normalize x so that its value will fall in the range $[\epsilon, 1]$, where ϵ is set equal to 0.001 as far as this study is concerned; $t_j = t^b_j / t^a_j$; and $v^i_j = (\sum_{k=1}^{w} f(v^i_{jk}))/w$. Moreover, the proposed algorithm will also use the tournament selection operator as the determination operator to determine the region to which the i-th searcher should be moved based on the expected value, and the number of players is set equal to half the number of regions h the current iteration has.

F. Marketing Research

This operator is responsible for keeping track of and updating the search (investment) status of each region, such as t^a_1 and t^a_2, once every iteration. This information is extremely useful in depicting the landscape of the solution space. This makes it possible for the SE-SOP to determine the search directions or regions that have a higher chance to find better results at later iterations based not only on the objective value of the new candidate solution but also on the landscape of the solution space. This operator will also update the historical memory and success parameters. The new investment solutions will then be updated if they are better than the current goods.

G. Linear Population Reduction

Figure 4 shows that the proposed algorithm also adopts the population size reduction for improving its performance because this strategy, which has also be used in several recent studies [14], [15], can be used to remove some of the similar searched solutions at the later stage of the convergence process. The example shows that the numbers of searchers, goods, and regions can reduced and merged.

The proposed algorithm will first sort the searchers and goods based on their objective values so as to remove the worse of them. The number of searchers can be calculated as follows:

$$n_{\text{curr}} = (n_{\min} - n_{\text{init}}) \times \frac{\mathrm{t}_{\text{curr}}}{\mathrm{t}_{\max}} + n_{\text{init}}, \tag{5}$$

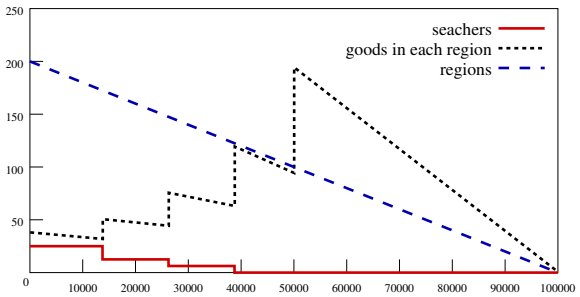

Fig. 4: A simple to illustrate the way for reducing the numbers of goods, regions, and searchers.

where n_{curr}, n_{init}, and $n_{\min}$ represent the number of searchers at the current iteration, the number of searchers in the first iteration, and the minimum number of searchers, respectively. The number of goods can be reduced by

$$w_{\text{total}} = w_{\text{init}} \times m_p \times w_\gamma, \tag{6}$$

where w_{total} is the total number of goods at the current iteration; w_{init} is the number of goods in each region; w_γ is the reduction rate of the goods; and m_p is ratio of goods. The example described in Figure 4 shows that the number of goods may not always decrease linearly because the number of goods in a region will be increased in the merge process, but the long-term trend is to reduce the number of goods as an aged curve. The proposed algorithm will randomly select the neighbor regions for the merge. The number of regions can be calculated as follows:

$$h_{\text{curr}} = \frac{h_{\text{init}}}{\left\lfloor \frac{\mathrm{t}_{\text{curr}} \times \log_2(h_{\text{init}})}{\mathrm{t}_{\max} \times c_1} \right\rfloor}, \tag{7}$$

where h_{curr} represents the number of regions at the current iteration; h_{init} the number of regions at the first iteration; and c_1 the acceleration factor to reduce the number of regions to 1.

III. The Experimental Results

A. Experimental Environment and Parameter Settings

The experimental analysis is conducted on a PC with two Intel Xeon E5-2620v4 (2.1 GHz, 20 MB cache, and 8 cores) and 78 GB of memory running Ubuntu 18.04 LTS, and all the programs are written in C++. The benchmark is a set of problems appeared in the CEC2017 competition on the real-parameter single objective optimization [16], which include 30 test functions that are tested for 10D, 30D, 50D, and 100D. Functions 1–3 are unimodal; functions 4–10 are multimodal; functions 11–20 are hybrid; and functions 21–30 are composition functions. Note that the results of function 2 is numerically unstable; therefore, it is not considered in the measurements [16]. To evaluate the performance of the proposed algorithm SE-SOP for solving the single-objective real-parameter optimization problem, we compare it to four state-of-the-art search algorithms; namely, jSO [17], EBOwithCMAR [18], ELSHADE-SPACMA [19], and L-SHADE-RSP

[20]. The parameter settings of these algorithms are the same as the original settings. As for the parameter setting, s_p is set equal to 21, h_{init} to 4, m_p to 3.5, $\mathcal{H}$ to 7, $\mathcal{T}_p$ to 0.011, and c_1 to 0.45.

As far as this study is concerned, the simulation is carried out for 51 runs for all the test functions. The number of dimensions is set equal to 10, 30, 50, and 100. The maximum number of evaluations is set equal to 100,000, 300,000, 500,000, and 1,000,000. To evaluate the search results of these algorithms, the lower and upper bounds of each dimension are set equal to -100 and 100, respectively; so, if the solution of a test function is out of the scope, it will not be used to evaluate the performance of the search algorithm. If the difference between the searched solution and the optimum solution is less than 10^{-8}, it can be regarded as 0. The first measurement is based on the total rank [16] that is defined as

$$\mathcal{R} = \mathcal{R}^1 + \mathcal{R}^2 \tag{8}$$

that takes into account (1) the difference between the searched result and the optimum solution (i.e., error rate) $\mathcal{R}^1$, (2) the rank of a search algorithm $\mathcal{R}^2$ defined as follows:

$$\mathcal{R}^1 = 1 - \frac{\mathcal{R}^{\text{err}} - \mathcal{R}^{\text{err}}_{\text{min}}}{\mathcal{R}^{\text{err}}}, \tag{9}$$

$$\mathcal{R}^2 = 1 - \frac{\mathcal{R}^{\text{rank}} - \mathcal{R}^{\text{rank}}_{\text{min}}}{\mathcal{R}^{\text{rank}}}, \tag{10}$$

where $\mathcal{R}^{\text{err}}$ and $\mathcal{R}^{\text{rank}}$ are defined as follows:

$$\mathcal{R}^{\text{err}} = 0.1\sum_{i=1}^{29} f^{\text{err}}_{i,10} + 0.2\sum_{i=1}^{29} f^{\text{err}}_{i,30} + 0.3\sum_{i=1}^{29} f^{\text{err}}_{i,50} + 0.4\sum_{i=1}^{29} f^{\text{err}}_{i,100}, \tag{11}$$

$$\mathcal{R}^{\text{rank}} = 0.1\sum_{i=1}^{29} f^{\text{rank}}_{i,10} + 0.2\sum_{i=1}^{29} f^{\text{rank}}_{i,30} + 0.3\sum_{i=1}^{29} f^{\text{rank}}_{i,50} + 0.4\sum_{i=1}^{29} f^{\text{rank}}_{i,100}; \tag{12}$$

$\mathcal{R}^{\text{err}}_{\text{min}}$ and $\mathcal{R}^{\text{rank}}_{\text{min}}$ are defined as the minimum error rate and rank of all the search algorithms compared herein. The second one is Wilcoxon signed-ranks test that is used to understand the difference between the proposed algorithm and all the other search algorithms compared in this paper.

B. Simulation Results

To understand the impact of the parameters on the performance of the proposed algorithm, we first set the parameters s_p, h_{init}, m_p, $\mathcal{H}$, $\mathcal{T}_p$, and c_1 equal to 21, 4, 3.5, 7, 0.009, and 0.4, respectively, as the default values. We then adjust the value of each parameter so as to understand its impact defined as follows:

$$\mathbb{S} = 4 \cdot \mathbb{S}_0 - \sum_{i=1}^{4} \mathbb{S}_i, \tag{13}$$

where $\mathbb{S}$ represents the score of the proposed algorithm with the given parameter settings, and $\mathbb{S}_0$, $\mathbb{S}_1$, $\mathbb{S}_2$, $\mathbb{S}_3$, and $\mathbb{S}_4$ represent, respectively, the total rank $\mathcal{R}$ of the proposed algorithm, EBOwithCMAR, jSO, ELSHADE-SPACMA, and L-SHADE-RSP with the given parameter settings. Figure 5 shows the impact of the parameters s_p, h_{init}, m_p, $\mathcal{H}$, $\mathcal{T}_p$, and c_1. It can be easily seen that the best parameter settings are 21, 4, 3.5, 7, 0.011, and 0.45. That is why SE-SOP uses these parameter settings to carry out all the simulations in this study.

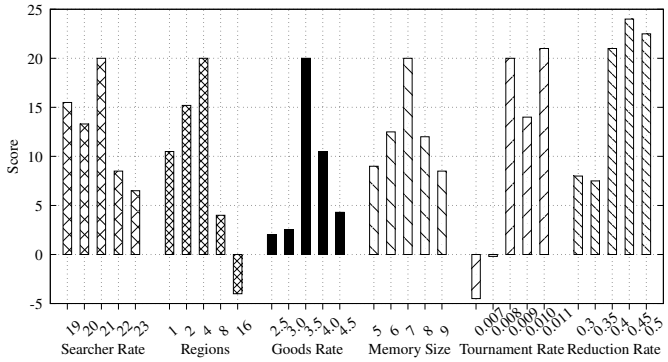

Fig. 5: The impact of parameters on the SE-SOP.

As shown in Table I, although the ELSHADE-SPACMA gives the best $\mathcal{R}^1$, its rank score $\mathcal{R}^2$ is not as good. Our observation shows that this is because ELSHADE-SPACMA is able to find quite good results than the others for some of the test functions, such as 50 dimensional f_{30} and 100 dimensional f_{22}. However, its rank score shows that it is not able to find the best results than the others for most of the test functions. L-SHADE-RSP provides a more stable way to find the solution compared with ELSHADE-SPACMA, jSO, and EBOwithCMAR in terms of $\mathcal{R}^1$ and $\mathcal{R}^2$. That is why L-SHADE-RSP is ranked the second. The proposed algorithm can not only find the best results in terms of the rank score $\mathcal{R}^2$ but also the second best results in terms of the error score $\mathcal{R}^1$, among all the search algorithms compared in this study. This implies that the proposed algorithm can not only minimize the error rate but also find the best results for most test functions.

TABLE I: COMPARISON OF SE-SOP AND ALL THE OTHER SEARCH ALGORITHMS IN TERMS OF $\mathcal{R}^1$, $\mathcal{R}^2$, AND $\mathcal{R}$.

	$\mathcal{R}^1$	$\mathcal{R}^2$	$\mathcal{R}$	Total Rank
jSO	48.62	39.80	88.42	5
EBOwithCMAR	48.93	44.42	93.35	4
ELSHADE-SPACMA	**50.00**	43.73	93.73	3
L-SHADE-RSP	49.03	48.88	97.91	2
SE-SOP	49.21	**50.00**	**99.21**	**1**

Table II provides additional comparison to show that the proposed algorithm is capable to find significantly better results than all the other search algorithms compared in this paper. The terms $+$, $\approx$, and $-$ represent, respectively, that the SE-SOP is significantly better than the others, has no difference from the others, and is significantly worse than the others. Some of the results of low-dimensional test functions show that the proposed algorithm is no difference from the other search algorithm; some of the results show that the proposed algorithm is even worse than the other search algorithms. Although not obvious for functions in low-dimensional space, the proposed algorithm outperforms all the other search algorithms compared in this study as the number of dimensions of the test functions increases. This implies that the number of times the proposed algorithm beats the others will increase significantly, especially when the number of dimensions increases to 100 or even higher. That is why

TABLE II: COMPARISON OF SE-SOP AND ALL THE OTHER SEARCH ALGORITHMS IN TERMS OF WILCOXON SIGNED-RANKS TEST.

	$D = 10$			$D = 30$			$D = 50$			$D = 100$		
	$+$	$\approx$	$-$	$+$	$\approx$	$-$	$+$	$\approx$	$-$	$+$	$\approx$	$-$
EBOwithCMAR	7	12	10	9	10	10	11	7	11	16	4	9
jSO	9	10	10	10	11	8	12	9	8	20	4	5
ELSHADE-SPACMA	10	12	7	12	15	2	12	10	7	15	4	10
L-SHADE-RSP	8	10	11	7	12	10	4	12	13	15	7	7

the proposed algorithm provides the best results in terms of the total rank $\mathcal{R}$.

IV. RELATED WORK

As we mentioned in Section I, the branch-and-bound algorithms were developed in the early stage for solving the single-objective real-parameter optimization problem [5]–[7]. Since some successful results in recent studies [21]–[23] have shown that metaheuristic algorithms are capable of finding better results than deterministic search algorithms in solving the SOP, using particle swarm optimization (PSO) to solve such a problem has attracted the attention of many researchers. More recently, the covariance matrix adaptation evolution strategy (CMA-ES) [8] and success-history based parameter adaptation for differential evolution (SHADE) [9] have become two promising ways for solving the SOP. The basic idea of CMS-ES is to use the Gaussian distribution with mean, step-size and covariance estimation to create new candidate solutions. The basic idea of SHADE is to use the historical memory to record the parameter settings that have been used to improve the search results before to adjust the parameters of differential evolution (DE). In [14], Tanabe and Fukunaga presented an improved version of SHADE that has the linear population size reduction added to reduce the similar solutions at the later stage of the convergence process, called L-SHADE. Still, another improved version of L-SHADE, called L-SHADE-RSP, that uses a rank-based selection to increase the exploitation capabilities of mutation strategy has been presented in [20].

In [17], Brest et al. presented an improved variant of the L-SHADE, by using a new weighted version of mutation strategy. In [18], Kumar presented a hybrid search algorithm, called EBOwithCMAR, that integrates butterfly optimizer and covariance matrix and won the computation of CEC 2017. Another hybrid search algorithm, called ELSHADE-SPACMA, can be found in [19], which adopts the technologies of CMA-ES and L-SHADE. In summary, the development of search algorithms for solving the single-objective real-parameter optimization problem has undergone several improvements, from branch-and-bound algorithm, PSO-based algorithms, CMS-ES, SHADE, all the way to the hybrid algorithm of them. Most of them strongly impact the development of search algorithms for solving the global optimization which means that an effective search algorithm for solving the SOP may also be used to solve other kinds of optimization problems in the continuous space.

V. CONCLUSIONS

This paper presents an improved version of (SE) for solving the single-objective real-parameter optimization problem. This is motivated by the SE-based algorithms that have successfully solved many combinatorial optimization problems but not for optimization problems in the continuous space. The main differences between the proposed algorithm (SE-SOP) and SE are in the way the solution space is divided, the expected value is computed, and the transition operator is designed. Two promising technologies in solving the SOP—self-adaptive parameter and linear population size reduction (LPSR)—are also added in the proposed algorithm. The simulation results show that the proposed algorithm is capable of finding out better results than other state-of-the-art algorithms for solving the single-objective real-parameter optimization problem for most functions. It can be easily seen that the proposed algorithm provides a stable way to solve the SE-SOP. In the future, our focus will be on developing an efficient way to accelerate the speed of the proposed algorithm.

ACKNOWLEDGMENT

This work was supported in part by the Ministry of Science and Technology of Taiwan, R.O.C., under Contract MOST108-2221-E-110-076-MY3 and in part by the Taiwan Information Security Center at National Sun Yat-sen University (TWISC@NSYSU).

REFERENCES

[1] L. Liberti, "Introduction to global optimization," LIX, École Polytechnique, Palaiseau F-91128, France, Tech. Rep., 2006. [Online]. Available: http://www.lix.polytechnique.fr/~liberti/teaching/mpro/pma-12/nonconvex_optimization.pdf

[2] Y. Lu, S. Wang, Y. Zhao, and C. Yan, "Renewable energy system optimization of low/zero energy buildings using single-objective and multi-objective optimization methods," *Energy and Buildings*, vol. 89, pp. 61–75, 2015.

[3] A. Andreotti, G. Carpinelli, F. Mottola, and D. Proto, "A review of single-objective optimization models for plug-in vehicles operation in smart grids part II: Numerical applications to vehicles fleets," in *Proceedings of the IEEE Power and Energy Society General Meeting*, 2012, pp. 1–8.

[4] J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization," Technical Report 201311, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China, and Technical Report, Nanyang Technological University, Singapore, Tech. Rep., 2013. [Online]. Available: http://bee22.com/manual/tf_images/Liang\%20CEC2014.pdf

[5] J. E. Falk and R. M. Soland, "An algorithm for separable nonconvex programming problems," *Management Science*, vol. 15, no. 9, pp. 550–569, 1969.

[6] G. P. McCormick, "Computability of global solutions to factorable nonconvex programs: Part i – Convex underestimating problems," *Mathematical Programmin*, vol. 10, pp. 147–175, 1976.

[7] E. Sandgren, "Nonlinear integer and discrete programming in mechanical design optimization," *Journal of Mechanical Design*, vol. 112, no. 2, pp. 223–229, 1990.

[8] L. Chen, Z. Zheng, H. Liu, and S. Xie, "An evolutionary algorithm based on covariance matrix leaning and searching preference for solving CEC 2014 benchmark problems," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2014, pp. 2672–2677.

[9] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2013, pp. 71–78.

[10] C.-W. Tsai, "Search economics: A solution space and computing resource aware search method," in *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, 2015, pp. 2555–2560.

[11] ——, "An effective WSN deployment algorithm via search economics," *Computer Networks*, vol. 101, pp. 178–191, 2016.

[12] ——, "SEIRA: An effective algorithm for IoT resource allocation problem," *Computer Communications*, vol. 119, pp. 156–166, 2018.

[13] C.-W. Tsai, S.-P. Tseng, C.-S. Yang, and M.-C. Chiang, "PREACO: A fast ant colony optimization for codebook generation," *Applied Soft Computing*, vol. 13, no. 6, pp. 3008 – 3020, 2013.

[14] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2014, pp. 1658–1665.

[15] J. L. J. Laredo, C. M. Fernandes, and J. J. Merelo, "Improving genetic algorithms performance via deterministic population shrinkage," in *Proceedings of the Conference on Genetic and evolutionary computation*, 2009, pp. 819–826.

[16] N. H. Awad, M. Z. Ali, J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective bound constrained real-parameter numerical optimization," Technical Report, Nanyang Technological University, Singapore, Tech. Rep., 2016. [Online]. Available: https://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2017/CEC2017.htm

[17] J. Brest, M. S. Maučec, and B. Bošković, "Single objective real-parameter optimization: Algorithm jSO," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2017, pp. 1311–1318.

[18] A. Kumar, R. K. Misra, and D. Singh, "Improving the local search capability of effective butterfly optimizer using covariance matrix adapted retreat phase," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2017, pp. 1835–1842.

[19] A. A. Hadi, A. W. Mohamed, and K. M. Jambi, "Single-objective real-parameter optimization: Enhanced LSHADE-SPACMA algorithm," King Abdulaziz University, Saudi Arabia and Cairo University, Egypt, Tech. Rep., 2018.

[20] V. Stanovov, S. Akhmedova, and E. Semenkin, "LSHADE algorithm with rank-based selective pressure strategy for solving CEC 2017 benchmark problems," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2018, pp. 1–8.

[21] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[22] J. C. Bansal, P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon, and A. Abraham, "Inertia weight strategies in particle swarm optimization," in *Proceedings of the World Congress on Nature and Biologically Inspired Computing*, 2011, pp. 633–640.

[23] C. Li, S. Yang, and T. T. Nguyen, "A self-learning particle swarm optimizer for global optimization problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 3, pp. 627–646, 2012.