Neural Comput & Applic
DOI 10.1007/s00521-016-2328-2

ORIGINAL ARTICLE

β-Hill climbing: an exploratory local search

Mohammed Azmi Al-Betar[1]

Received: 11 December 2015 / Accepted: 13 April 2016

Abstract Hill climbing method is an optimization technique that is able to build a search trajectory in the search space until reaching the local optima. It only accepts the uphill movement which leads it to easily get stuck in local optima. Several extensions to hill climbing have been proposed to overcome such problem such as Simulated Annealing, Tabu Search. In this paper, an extension version of hill climbing method has been proposed and called β-hill climbing. A stochastic operator called β-operator is utilized in hill climbing to control the balance between the exploration and exploitation during the search. The proposed method has been evaluated using IEEE-CEC2005 global optimization functions. The results show that the proposed method is a very efficient enhancement to the hill climbing providing powerful results when it compares with other advanced methods using the same global optimization functions.

Keywords Hill climbing · Metaheuristics · Global optimization · Local search methods · Operations research

1 Introduction

Optimization is a field of operations research concerned with finding a best configuration of problem variables to optimize its objective function $f(\boldsymbol{x})$. The optimization problems are conventionally categorized into two groups due to the value range of their variables: continuous and discrete. In general, the continuous optimization problem is formulated as follows:

$$\min\{\mathrm{f}(\boldsymbol{x}) \,|\, \boldsymbol{x} \in \mathbf{X}\},$$

where $f(\boldsymbol{x})$ is the objective function; $\boldsymbol{x} = \{x_i \,|\, i = 1, \ldots, N\}$ is the set of decision variables. $\mathbf{X} = \{\boldsymbol{X}_i \,|\, i = 1, \ldots, N\}$ is the possible value range for each decision variable, where $\boldsymbol{X}_i \in [LB_i, UB_i]$, and LB_i and UB_i are the lower and upper bounds for the decision variable x_i , respectively, and N is the number of decision variables.

Definition 1 The search space $\mathcal{S} = \{\boldsymbol{x} = \{(x_i, v_i)|i \in 1 \cdots N \wedge v_i \in \boldsymbol{X}_i\}\}$ is the set of all possible solutions for a given problem with a dimension N, where each member is a solution of a given problem. In continuous optimization, $S \subseteq \mathbb{R}^N$.

The main purpose of solving optimization problems is to find a solution $\boldsymbol{x}^* \in \mathcal{S}$ with a minimum value of the objective function such that $f(\boldsymbol{x}^*) < f(\boldsymbol{x}^j), \forall \boldsymbol{x}^j \in \mathcal{S}$. This solution (e.g., $\boldsymbol{x}^*$) is called a *global optimal solution*.

The emergence of *metaheuristics* for solving difficult global optimization problems has been one of the most notable accomplishments over the last two decades [16, 18]. Metaheuristic-based method is an iterative improvement process that uses its operators and combines intelligently the problem specific knowledge for exploring and exploiting the search space ($\mathcal{S}$) in order to reach a good-quality solution [3, 22, 31].

The key research issue in applying metaheuristic-based methods to any global optimization problem is to make an attempt to *strike a balance between exploration and exploitation* during the search. It should be emphasized that during the *exploration* stage the search is encouraged to

✉ Mohammed Azmi Al-Betar
mohbetar@bau.edu.jo

[1] Department of Information Technology, Al-Huson University College, Al-Balqa Applied University, P.O. Box 50, Al-Huson, Irbid, Jordan

Published online: 27 April 2016

explore the not-yet-visited search space regions if need be. During the *exploitation* stage, however, the search concentrates on the already-visited search space regions [29].

It is widely accepted in the literature that the metaheuristic methods can be classified into two classes: (1) local search-based and (2) population-based methods. However, some researchers have classified the metaheuristic into nature-inspired versus non-nature-inspired methods; single versus multiobjective algorithms; one versus various neighborhood structures, and memory or memory-less [8].

Population-based methods normally begin with a population of random solutions. They iteratively recombine the properties of current population to come up with a new population [6]. Besides, they normally use a random method to diversify the search when necessary. The main advantage of population-based algorithms is that they are able to widely scan many search space regions at the same time. However, they are disadvantaged in their incapacity to provide a precise local optimal solution for each visited region. That is why the problem of slow convergence might occur.

Local search-based methods, the key issue of this paper, initiated with a single provisional solution, $\boldsymbol{x}$. That solution iteratively undergoes changes using a neighborhood structure (i.e., $\mathcal{N}(\boldsymbol{x})$) until a locally optimized solution, which is usually in the same region of the search space as the initial solution, is obtained. Therefore, the search space $\mathcal{S}$ can be seen as a set of regions $\widehat{s}$ where $\widehat{s} \subseteq S$ and $\cup\widehat{s} = S$. Local search-based methods are able to fine-tune the search space region to which they converge and find a precise local optimal solution. However, they go through a trajectory without doing a wider scan of the entire search space.

Definition 2 A neighborhood structure is a function $\mathcal{N} : \mathbf{S} \leftarrow 2^{\mathbf{S}}$ which assigns to each $\boldsymbol{x} \in \mathbf{S}$ a set of neighbors $\mathcal{N}(\boldsymbol{x}) \leftarrow \mathbf{S}$. Note that $\mathcal{N}(\boldsymbol{x})$ is called the neighborhood of $\boldsymbol{x}$.

Function $\mathcal{N}$ has different operations to calculate the neighboring solution for the solution $\boldsymbol{x}$. Any local search-based method can carry out one or more operations to move from the current solution $\boldsymbol{x}$ to the neighboring solution $\boldsymbol{x}' \in \mathcal{N}(\boldsymbol{x})$ as follows:

1. *First improvement* Explore the $\mathcal{N}(\boldsymbol{x})$ and select the first solution $\boldsymbol{x}'$ better than $\boldsymbol{x}$ such as $f(\boldsymbol{x}') < f(\boldsymbol{x})$.
2. *Best improvement* Explore all solutions in $\mathcal{N}(\boldsymbol{x})$ and select the solution with the lowest objective function. Formally, select $\boldsymbol{x}' \in \mathcal{N}(\boldsymbol{x})$ where $f(\boldsymbol{x}') < f(\boldsymbol{x}'')$, $\forall \boldsymbol{x}'' \in \mathcal{N}(\boldsymbol{x})$.
3. *Random walk* Select randomly $\boldsymbol{x}' \in \mathcal{N}(\boldsymbol{x})$. Note that this strategy does not check whether the value of $f(\boldsymbol{x}')$ is lower or higher than $f(\boldsymbol{x})$.
4. *Side walk* Explore $\mathcal{N}(\boldsymbol{x})$ and select the first solution $\boldsymbol{x}'$ that has the same value of objective function as $\boldsymbol{x}$ such as $f(\boldsymbol{x}') = f(\boldsymbol{x})$. Note that the side walk improvement strategy is very useful to enable the local search-based method in jumping from search space region to another in the same level with hope to empower it to escape local optima efficiently.

Definition 3 The local optimal solution with respect to the neighborhood structure $\mathcal{N}$ is a solution $\boldsymbol{x}$ such that $\forall x \in \widehat{s} : f(\widehat{s}) \longleftarrow f(x)$. The solution $\boldsymbol{x}$ is a local optimal solution if $f(\boldsymbol{x}) < f(x), \forall(x) \in \widehat{s}$.

Hill climbing (HC) is the simplest form of local search-based methods where the search nature only accepts the 'downhill'[1] move. In this method, the exploitation concept is activated where the search is improved using the accumulative search without any stochastic component. Therefore, HC is normally used to empower other EAs in terms of exploitation capability (see memetic algorithm [3]). The main shortcoming with classical version of HC is that it can easily get stuck in local optima [8]. With this dilemma in mind, several extensions to HC have been introduced to overcome such problem. The common idea of these extensions lies in utilizing an intelligent stochastic operator that may help in avoiding the local optima trap. The most popular extensions are Simulated Annealing (SA) [17], Tabu Search (TS) [12], Greedy Randomize Adaptive Search Procedure (GRASP) [10], Variable Neighborhood Search (VNS) [14], and Iterated Local Search (ILS) [20].

The main objective of this paper is to produce an extension variation of HC that utilizes an intelligent stochastic operator able to avoid the local minima for global optimization. The proposed method is called β-hill climbing. In the improvement loop of β-hill climbing, two consecutive operators have been triggered: neighborhood navigation ($\mathcal{N}$ operator) and β operator. In $\mathcal{N}$ operator, a single move to the neighboring solution using a random walk acceptance rule is adopted. Thereafter, β operator constructs a new solution selecting the majority of the new values from the current solution while other values are selected randomly from the range controlled by parameter β that tunes the balance between exploration and exploitation according to the nature of the search space. This idea stems from the uniform mutation operator of GA.

The performance of β-hill climbing has been intensively evaluated using IEEE-CEC2005 [27] functions of total number of 25 functions as used in [1, 2, 4, 5, 7, 11, 19, 21, 23–26, 30]. The comparative results show the efficiency of the proposed method.

The rest of the paper is organized as follows: The related work of hill climbing and its extensions is overviewed in

[1] The downhill move is used for minimization problem.

Sect. 2. The β-hill climbing algorithm is described in Sect. 3. Experiments and comparative results are presented in Sect. 4. Finally, the conclusion and some promising future research directions are provided in Sect. 5.

2 Related work

2.1 Hill climbing

The hill climbing (HC) algorithm (sometimes called simple local search) is the simplest form of local improvement methods. It starts with a random initial solution, iteratively moves from a current solution to a better neighboring solution until it arrives at a local optima (i.e., local optimal solution does not have a better neighboring solution). It only accepts the downhill moves where the quality of neighboring solution should be better than the current one. Therefore, it has the ability of converging to the local optima quickly. However, it can easily get stuck in local optima which in most cases is not satisfactory. Algorithm 1 provides the pseudo-code of the HC algorithm. The global optimization problems are considered to describe the classical version of HC. After generating the initial solution $\boldsymbol{x}$ and during the iterative improvement loop, a set of neighboring solutions are generated using the procedure $Improve(\mathcal{N}(\boldsymbol{x}))$. This procedure tries to find the improved neighboring solution from the set of the neighbors using any adopted acceptance rule such as first improvement, best improvement, random walk, or side walk. However, all of these strategies are stopped in local optima.

HC clearly depends on the definition of the search space S, the objective function $f(\boldsymbol{x})$ and the neighborhood structure $\mathcal{N}(\boldsymbol{x})$. The resulting solutions from HC tend to be highly unsatisfactory for most optimization problems since they often get stuck in local optima. Therefore, researchers have extended HC to have intelligent stochastic components that is able to escape the trap of local optima, and sometimes HC used as an exploiter engine hybridized with other evolutionary algorithms [3].

Algorithm 1 Hill climbing()

```
x_i = LB_i + (UB_i − LB_i) × U(0,1), ∀i = 1,2,...,N {The initial solution x}
Calculate(f(x))
while (no improvement can be reached) do
    x' = Improve(N(x))
    if (f(x') ≤ f(x)) then
        x = x'
    end if
end while
```

2.2 The variations of hill climbing

Several extensions of HC have been proposed in a bid to avoid getting stuck in local optimal solution. These extensions are commonly introduced through utilizing some stochastic mechanisms to accept the uphill move and empower the search to explore other regions of problem search space. The most popular HC extensions are Simulated Annealing (SA), Tabu Search (TS), Greedy Randomize Adaptive Search Procedure (GRASP), Variable Neighborhood Search (VNS), Guided Local Search (GLS), and Iterated Local Search (ILS) which will be described in the following sections.

2.2.1 Simulated Annealing (SA)

SA might be considered the oldest explorative method which was developed by Kirkpatrick et al. [17]. It has an explicit stochastic strategy to escape the local optimal problem. Algorithm 2 shows the pseudo-code of SA. It moves from the current solution $\boldsymbol{x}$ to a neighboring solution $\boldsymbol{x}' \in \mathcal{N}(\boldsymbol{x})$ based on the objective function of the current solution $f(\boldsymbol{x})$ and the objective function of a neighboring solution $f(\boldsymbol{x}')$ and the temperature parameter T which should be initialized in the initial search. In each iteration, SA moves from $\boldsymbol{x}$ to $\boldsymbol{x}'$ based on two acceptance criteria:

1. $f(\boldsymbol{x}') < f(\boldsymbol{x})$.
2. Monte Carlo acceptance criteria: $f(\boldsymbol{x}') > f(\boldsymbol{x})$ but $U(0,1) < e^{(-\frac{(f(x')-f(x)}{T})}$.

where $U(0, 1)$ is a uniform distribution random generator rating from 0 to 1. The latter acceptance option performs 'uphill moves' in the search space. Note that SA uses the *first improvement* acceptance rule.

SA initially sets a large value for T ($T = T_0$) where the exploration is at its prime. During the search, the value of T is dramatically decreased by the value $T = \alpha \times T$ where the chance of accepting uphill move is reduced until the *equilibrium* state (or steady state) is reached. The ratio of reducing the temperature over time is referred to as the 'cooling schedule.' The success of SA depends on the cooling schedule and the initial temperature T_0 which are subjected to an optimization problem each. Recently, two variations of SA have appeared in the literature: (1) The Threshold Accepting method. (2) The Great Deluge method. For more detailed information about SA, see [28].

Algorithm 2 Simulated Annealing()

```
x = Initial_Solution()
T = T_0
while (not stopping criterion is met) do
    Select x' Where x' ∈ N(x)
    if f(x') < f(x) then
        x = x'
    else if U(0,1) < e^(−(f(x')−f(x))/T) then
        x = x'
    end if
    T = αT
end while
```

2.2.2 Tabu Search (TS)

Tabu Search (TS) which was developed by Glover [12] uses the short-term memory to escape the local optimal solution and avoid the cyclic search. This short-term memory is represented by 'Tabu list' which keeps track of the recently visited solutions (or *solution attributes*) which will not be revisited by TS in the predefined following iterations. TS uses the *best improvement* acceptance criteria in which the best neighboring solution is not necessarily better than the current one. However, the movement from the current solution to a neighboring solution is accepted based on two conditions: (1) the neighboring solution is not included in the Tabu list, or (2) the neighboring solution is included in the Tabu list and accepted by 'inspiration criteria.' The inspiration criteria let the TS move to the Tabu solution when it has a better quality than the best solution found so far. In practice, Tabu list is a 'queue' (e.g., FIFO principle) with a specific size called 'Tabu tenure (TN)' which the success of TS depends on: e.g., larger Tabu tenure means that the TS will keep track of many visited solutions, and thus, the cyclic solutions are reduced. However, if the Tabu neighboring solutions are large in number, this leads to complicated moves.

Algorithm 3 shows the pseudo-code of the basic TS. The function which navigates the set of neighboring solutions of the current one and selects the best solution which is not in the Tabu list is $Select\_Best\_Nigbour(.)$. Undoubtedly, the best solution is adopted for the next iteration without checking whether the current solution is improved. By this method, the TS can accept the uphill moves and thus minimize the chance of getting stuck in local minima.

In the latest developments of TS [13], the long-term memory has been added to TS structure in order to keep track of the whole search. Basically, the long-term memory includes information on each solution. This information includes *recency* (the recent iterations that used this solution as a neighboring one), *frequency* (the number of times the solution was visited during the search), *quality* (which keeps track of the good elements included in this solution), and *influence* (which keeps track of the most influential solution visited in the accumulative search). For a detailed explanation on TS see [13].

Algorithm 3 Tabu Search()

1: x = Initial_Solution()
2: $Tabu\_list = \phi$
3: **while** (not stopping criterion is met) **do**
4: $\quad \boldsymbol{x}' = Select\_Best\_Nigbour(\mathcal{N}(\boldsymbol{x})) \wedge \boldsymbol{x}' \notin Tabu\_list$
5: $\quad \boldsymbol{x} = \boldsymbol{x}'$
6: $\quad$ Update($Tabu\_list$)
7: **end while**

2.2.3 Greedy Randomize Adaptive Search Procedure (GRASP)

It is developed in [10] which is a simple local search metaheuristic method. It can be seen as a multistart two-phase metaheuristic: the construction phase and the local improvement phase. The best solution obtained in the whole iteration is recorded and returned using MemoriseBestSolutionFound() procedure as shown in Algorithm 4.

In the construction phase, the initial solution $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is constructed where each value of any decision variable $x_i, \forall i = \{1, 2, \ldots, n\}$ is assigned based on two strategies: a dynamic constructive heuristic and randomization. Initially, it constructs a list (called RCL_α) of potential decision variables that should be assigned values in the next loop. The variables in RCL_α are selected based on a heuristic algorithm which ranks the variables based on their effect on the partial constructed solution. The parameter α is the length of the *RCL* which should be correctly selected. The next decision variable to be assigned in $\boldsymbol{x}$ is selected randomly from the list RCL_α and assigned values. The RCL_α elements are updated dynamically until a complete solution is constructed as shown in Algorithm 5.

In the local improvement phase, the solution $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ built in the construction phase is improved using hill climbing or any local search algorithms such as SA and TS. Notably, the GRASP can multistart from different search space regions sequentially starting each time by a different potential solution generated by a constructive phase. More details of GRASP can be found in [10].

Algorithm 4 Greedy Randomize Adaptive Search Procedure()

1: $itr = 0$
2: **while** ($itr \leq$ Max_Itr) **do**
3: $\quad \boldsymbol{x}$= ConstructGreedyRandomizedSolution()
4: $\quad \boldsymbol{x}$'= Hill_climbing($\boldsymbol{x}$)
5: $\quad$ MemoriseBestSolutionFound()
6: **end while**

Algorithm 5 ConstructGreedyRandomizedSolution() function

1: Set $\boldsymbol{x} = \phi$
2: Set α {The length of restricted candidate list RCL_α.}
3: Let $RCL_\alpha = \phi$ be the restricted candidate list.
4: **for** j= 1 to n **do**
5: $\quad$ Updated the restricted candidate list RCL_α using a greedy heuristic method.
6: $\quad$ Select randomly a decision variable $x_i \in RCL_\alpha$.
7: $\quad$ Assign the x_i by value in $\boldsymbol{x}$.
8: **end for**
9: return $\boldsymbol{x}$.

2.2.4 Variable Neighborhood Search (VNS)

VNS, developed in [14] and [15], uses multiple neighborhood structures rather than a single one. It is another version of hill climbing algorithm that systematically changes

the neighborhood structures $\mathcal{N}(\boldsymbol{x})$ during the search process which facilitates switching between different search space regions.

Algorithm 6 provides the pseudo-code of VNS. Initially, the set of neighborhood structures should be defined where each one of them defines a solution landscape differently from the others. These neighborhood structures can be randomly chosen, yet an ordered series of increasing cardinality for these neighborhood structures (i.e., $\mathcal{N}_1(\boldsymbol{x}) < \mathcal{N}_2(\boldsymbol{x}) < \cdots < \mathcal{N}_k(\boldsymbol{x})$) has to be defined. Then the initial solution is randomly generated.

As described in [8], the improvement loop consists of three phases: *shaking*, hill climbing and move. In the shaking phase, a neighboring solution $\boldsymbol{x}'$ of the current one based on particular neighborhood structures $\mathcal{N}_i(\boldsymbol{x})$ is randomly selected. This shaking process is randomly done regardless of the objective function of the current solution. This is to perturb the solution $\boldsymbol{x}$) as to provide a good starting point for the hill climbing. The second phase (i.e., hill climbing) begins with $\boldsymbol{x}'$ and navigates its neighboring solutions until the local optima is reached. Note that the hill climbing can utilize any neighborhood structure but is not restricted to the neighborhood structures of $\mathcal{N}_1(\boldsymbol{x}), \mathcal{N}_2(\boldsymbol{x}), \ldots, \mathcal{N}_k(\boldsymbol{x})$.

The move phase decides whether the local optima obtained by hill climbing should be replaced by the current one. As can be seen from Algorithm 6, if the obtained solution replaces the current solution, then the process is repeated starting from the first neighborhood structure in the list. If not, the next neighborhood structure will be selected. Note that if the shake phase is done frequently without improvement, the current solution will be dispersed more than needed and might be affected by high diversity.

Recently, other variations of VNS with the same search structure have been proposed such as Variable Neighborhood Descent (VND),Variable Neighborhood Decomposition Search (VNDS), Skewed VNS (SVNS). More detailed can be found in [8].

Algorithm 6 Variable Neighborhood Search()

```
1: Select a set of neighborhood structures N_1(x), N_2(x), ..., N_k(x)
2: x = Initial_Solution()
3: while (not stopping criterion is met) do
4:    i = 1
5:    while (i ≤ k) do
6:       x' = SelectRandomSolution(N_i(x)) { Shaking phase}
7:       x'' = Hill_climbing(x')
8:       if f(x'') < f(x) then
9:          x = x''
10:         i = 1
11:      else
12:         i++
13:      end if
14:   end while
15: end while
```

2.2.5 *Iterated Local Search (ILS)*

It was developed in [20] for quadratic assignment problem which can be considered the most general local search framework among the explorative methods. Note that the VNS can be counted as an extracted type of ILS. As pseudo-code is shown in Algorithm 7, it starts with a solution $\boldsymbol{x}$ which is randomly generated. That solution is input to the hill climbing until local optimal solution $\boldsymbol{x}'$ is reached. During the improvement loop, the solution $\boldsymbol{x}'$ is perturbed where some values of the decision variables keep unchanged while small values should be randomly changed from the possible range. The resulting solution $\boldsymbol{x}''$, thereafter, enters again into hill climbing until it reaches the local optimal $\boldsymbol{x}'''$ solution. The obtained solution is evaluated using ApplyAcceptanceCriterion($\boldsymbol{x}'$, $\boldsymbol{x}'''$, history) that decides which of the two local optimal solution has to be chosen as a new candidate for the next improvement loop. In summary, Table 1 provides a summary for local search-based methods and their main concepts.

Algorithm 7 Iterated Local Search()

```
1: x = Initial_Solution()
2: x' = Hill_climbing(x)
3: while (not stopping criterion is met) do
4:    x'' =Perturbation(x', history)
5:    x''' = Hill_climbing(x'')
6:    x' = ApplyAcceptanceCriterion(x', x''', history)
7: end while
```

3 Proposed method

Hill climbing is the simplest form of the local search-based methods. It does not have an explorative strategy which leads it to get easily stuck in local optima. As aforementioned, the previous extensions of hill climbing utilize intelligent stochastic strategies to overcome such problem. These explorative strategies are often embedded in the neighborhood functions to define systematically different search space regions. Other methods like SA or GD used a relaxed acceptance method to deal with uphill moves. In β-hill climbing, a new explorative operator called β has been utilized based on an idea inspired by the uniform mutation operator of GA. At each iteration, the search space of the current solution will be defined based on the function $\mathcal{N}$ and unbounded search space is defined based on the β operator.

The β-hill climbing (see Algorithm 8 and the flowchart in Fig. 1) starts with an arbitrary solution $x = (x_1, x_2, \ldots, x_N)$. It iteratively generates a new solution $x' = (x'_1, x'_2, \ldots, x'_N)$ based on two operators: neighborhood navigation (i.e., $\mathcal{N}$-operator) and β operator.

In the $\mathcal{N}$-operator stage, the function *improve*($\mathcal{N}(\boldsymbol{x})$) is used with the 'random walk' acceptance rule where in each

Table 1 A summary table for the local search-based methods

Method	Exploitation process	Exploration process	Parameters	Category of neighborhood used
HC	Neighborhood Search	NA	NA	Random improvement, first improvement, best improvement, side walk
SA	Neighborhood Search	Cooling scheduling	α and temperature (T)	First improvement
TS	Neighborhood Search	Short-term memory	Aspiration criteria and Tabu tenure (TN)	Best improvement
GRASP	Neighborhood Search	RCL_α	α	Random improvement, first improvement, best improvement, side walk
VNS	Neighborhood Search	Shaking	k	Random improvement, first improvement, best improvement, side walk
ILS	Neighborhood Search	Perturbation	Number of perturbation elements	Random improvement, first improvement, best improvement, side walk

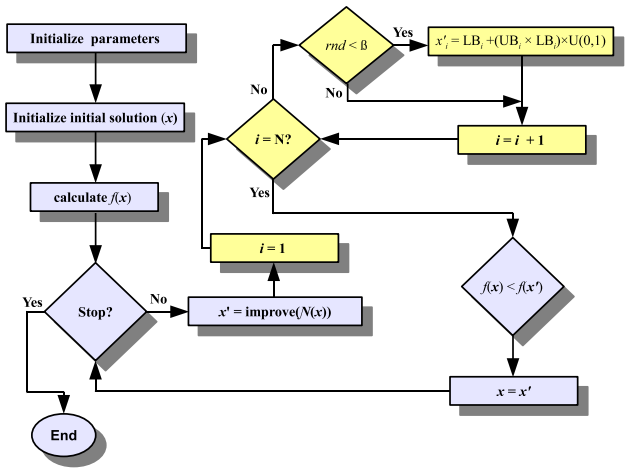

Fig. 1 Flowchart of β-hill climbing

iteration a random neighboring solution of the solution $\boldsymbol{x}$ is adopted as follows:

$$x'_i = x_i \pm U(0,1) \times bw \quad \exists! i \in [1, N].$$

Note that i is randomly selected from the dimensionality range, $i \in [1, 2, \ldots, N]$. The parameter bw is the bandwidth between the current value and the new value.

In β operator stage, the variables of the new solution are assigned values based on the existing values of the current solution or randomly from available range with a probability of β where $\beta \in [0, 1]$ as follows:

$$x'_i \leftarrow \begin{cases} x_r & rnd \leq \beta \\ x_i & \text{otherwise.} \end{cases}$$

where $x_r \in \boldsymbol{X}_i$ is the possible range for the decision variable x'_i and rnd generates a uniform random number between 0 and 1.

The convergence toward the optimal solution in the β-hill climbing is achieved by two operators: neighborhood navigation ($\mathcal{N}$ operator) and β operator. $\mathcal{N}$-operator navigates the neighboring solutions of the current one and selects randomly one with a better objective value. In β-operator, the convergence can be achieved by constructing a controlled portion of the current solution, and therefore, the convergence rate could be accelerated. The β-operator can be the source of exploration while the $\mathcal{N}$-operator can be considered as the source of exploitation.

Algorithm 8 β–hill climbing pseudo-code

```
x_i = LB_i + (UB_i − LB_i) × U(0,1), ∀i = 1,2,...,N {The initial solution x}
Calculate(f(x))
itr = 0
while (itr ≤ Max_Itr) do
    x' = improve(N(x))
    for i = 1, ··· , N do
        if (rnd ≤ β) then
            x'_i = LB_i + (UB_i − LB_i) × U(0,1)
        end if  { rnd ∈ [0,1]}
    end for
    if (f(x') ≤ f(x) then
        x = x'
    end if
    itr = itr + 1
end while
```

In terms of search space navigation, β-hill climbing is able to jump from a search space region to another using β operator that can be thought as a source of exploration. To visualize such idea, Fig. 2 shows a simple search space with multiple regions. Let assume that the β-hill climbing navigates the search space from solution labeled 1 to solution 7 (i.e., the global minima). Notably, the β operator allows the algorithm to move from a region to another at the same level or less (i.e., see the movement from solution 2 to 3 or from solution 4 to 5). By means of this ability, the β-hill climbing can escape the trap of local minima by trying stochastic values for some decision variables.

3.1 Illustrative example

In order to further understand the convergence behavior of β-hill climbing, the 'Shifted Schwefel's Problem 1.2'

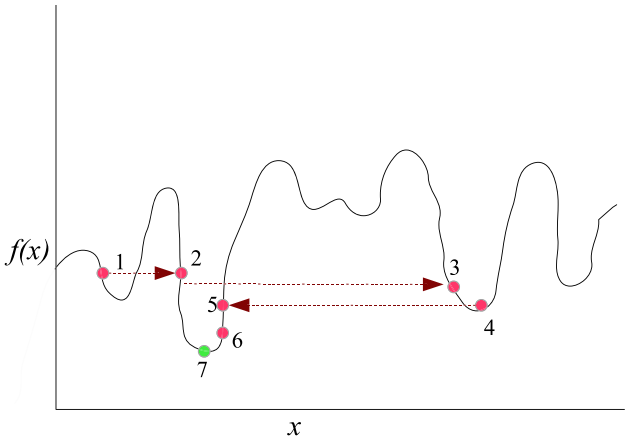

Fig. 2 Example of Search space navigation using β-hill climbing

discussed in Table 3 is used with $N = 5$ to provide a step-by-step example of how the proposed algorithm is performed.

Initially, as given in Table 2, the initial solution is generated randomly with $f(x) = 108{,}780.5649$. Thereafter, this solution undergoes improvement using two operators: $\mathcal{N}$ operator and β operator. The $\mathcal{N}$ operator is used for neighboring navigation where only one decision variable is adjusted to its neighboring value at each iteration. This can move the solution to its neighboring solution using a trajectory in the local search space without jumping to another region. However, the β operator is the source of exploration controlled by $\beta = 0.05$ parameter where all decision variables are checked whether or not to be stochastically replaced by a new value from the value range (i.e., $[-100, 100]$). By means of this process, the β operator can navigate other regions of the search space and ensure some diversity for the solution.

Apparently, from the values reported in Table 2 where iteration details have been provided, the β-hill climbing utilized the two operators for convergence. The source of improvement either acquire from local exploitation done by $\mathcal{N}$ operator or global exploration achieved by β operator. Eventually, the β-hill climbing is able to converge to the global optima [i.e., $f(x) = -450$] by means of striking a right balance between exploration and exploitation during the search.

4 Experimental evaluation

To provide a comprehensive comparison and assess the different aspects of the proposed β-hill climbing, we divide the presentation of the experimental evaluation into the following subsections. We firstly describe the characteristics of the test functions used in Sect. 4.1. The configuration to the all upcoming experiments and convergence cases have been introduced in Sect. 4.2. The performance of the proposed algorithm and the sensitivity to its parameters β and bw have been analyzed in Sect. 4.3. In Sect. 4.4, the classic version of HC in comparison with the proposed β-hill climbing is demonstrated and the results produced by β-hill climbing is compared against those obtained by other local search-based methods. Subsequently, the comparative evaluation with a total of 11 comparative methods using the same test functions participated in the IEEE-CEC2005 competition is provided in Sect. 4.5.

Note that the time required to solve any optimization problem using β-hill climbing depends on the objective function complexity as well as the number of iteration to converge. Therefore, the time complexity of this type of problems cannot be easily measured in advance as others algorithmic solutions built for classical problem like sorting or searching. For example, the time required to find the optimal solution for f1: Shifted Sphere Function of decision variable $= N$ and with number of iterations $= Max\_Itr$, thus, the time complexity is $\mathcal{O}(N \times Max\_Itr)$. Therefore, the time complexity of the proposed algorithm cannot be generalized.

4.1 Test functions

In order to evaluate the proposed β-hill climbing, 25 test functions with different characteristics have been used (f_1–f_{25}) selected from the set of test functions of IEEE-CEC2005 [27]. A brief summary of the test functions used in our experiments is provided in Table 3. In Table 3, each test function is abbreviated and its variables' value range are recorded as appears in the original source [27]. The dimension N of each solution vector used in the experiments is also recorded together with the fitness value of the optimal solution $f(x*)$.

The set of IEEE-CEC2005 test functions (f_1–f_{25}) can be characterized as follows: the first five functions (f_1–f_5) are unimodal and shifted; the second seven test functions (f_6–f_{12}) are basic multimodal and shifted; the third two functions (f_{13}–f_{14}) are expanded multimodal; and the fourth 11 functions (f_{15}–f_{25}) are hybrid composition (i.e., all of them are non-separable, rotated, and multimodal functions containing a large number of local minima). The detailed principle of the IEEE-CEC2005 is given in [27].

4.2 Test design

The β-hill climbing is experimentally evaluated and compared with the basic hill climbing. The test functions have been experimented with nine cases of different combinations to show the sensitivity of β-hill climbing to its parameters. The main two parameters of β-hill climbing

Table 2 Illustrative example

Iteration	Process stage	Solution	$f(\boldsymbol{x})$
1	Current	85.634820, 30.589992, 26.489684, −87.581556, −80.798977	108,780.5649
	After $\mathcal{N}$ operator	2.369930, 30.589992, 26.489684, −87.581556, −80.798977	36,285.2874
	After β operator	2.369930, 30.596535, 26.489684, −87.581556, −80.798977	36,288.6955
	Decision		Replace the current
2	Current	2.369930, 30.596535, 26.489684, −87.581556, −80.798977	36,288.6955
	After $\mathcal{N}$ operator	2.369930, 30.596535, 26.489684, −87.581556, −80.798977	36,288.6955
	After β operator	2.369930, 30.596535, 26.486056, −87.581556, −80.798977	36,287.3879
	Decision		Replace the current
3	Current	2.369930, 30.596535, 26.486056, −87.581556, −80.798977	36,287.3879
	After $\mathcal{N}$ operator	2.369930, 30.596535, 26.486056, −87.581556, −80.798977	36,287.3879
	After β operator	2.369930, 30.596535, 26.486056, −87.581556, −80.804117	36,287.9078
	Decision		Do Not Replace the current
10	Current	−6.219165, 30.596535, 26.488431, −87.589369, −80.798977	32,753.6911
	After $\mathcal{N}$ operator	−6.219165, 30.596535, 26.488431, −87.589369, −80.798977	32,753.6911
	After β operator	−6.219165, 30.589252, 26.488431, −87.589369, −80.798977	32,750.3980
	Decision		Replace the current
100	Current	−6.206341, −38.132448, −30.833368, −87.545767, 92.026644	2208.0565
	After $\mathcal{N}$ operator	−6.206341, −38.132448, −30.833368, −87.545767, −75.800356	34,514.9487
	After β operator	−6.206341, −38.132448, −30.827927, −87.545767, −75.800356	34,512.5697
	Decision		Do Not Replace the current
1000	Current	6.419687,−38.413372, −36.248243, −69.971576, 84.322711	774.0564
	After $\mathcal{N}$ operator	6.419687, −38.413372, −36.248243, −69.971576, 84.322711	774.0564
	After β operator	6.419687, −38.413372, −36.248243, −69.971576, 84.315710	773.9939
	Decision		Replace the current
10,000	Current	25.113352, −65.678317, −17.885534, −83.014658, 81.781912	−292.2612
	After $\mathcal{N}$ operator	25.113352, −65.678317, −17.885534, −83.014658, 81.781912	−292.2612
	After β operator	25.119389, −65.678317, −17.885534, −83.014658, 81.781912	−292.3286
	Decision		Replace the current
100,000	Current	35.626604, −82.912141, −10.642348, −83.581521, 83.155210	**−450.0000**
	After $\mathcal{N}$ operator	35.626604, −66.108827, −10.642348, −83.581521, 58.163361	464.1085
	After β operator	35.626604, −66.108827, −10.642348, −83.584144, 58.163361	464.0633
	Decision		Do not replace the current

Bold value indicates best solution achieved (lowest is best)

are β which is responsible for managing the size of diversity within the constructed solution and *bw* which determines the distance bandwidth of any variable value to its neighboring value. The experimented cases are given in Table 4. Each case is set to show the convergence behavior of β-hill climbing using different values of β and *bw*. This is to clarify the effect of their interactions.

For the 25 test functions of IEEE-CEC2005, we follow the parameters and conditions of the CEC competition [27] where the 25 repeated runs have been performed for each test function. The 25 runs have been summarized in terms of the Average of the Error of the best individual (i.e., $AE = |f(\boldsymbol{x}^*) - f(\boldsymbol{x}^{best})|$). Note that $\boldsymbol{x}^*$ is a given optimal solution while the $\boldsymbol{x}^{best}$ is the average best solution obtained in 25 runs. The dimension $N = 10$ and the β-hill climbing iterates 100,000 evaluations of the fitness function. Table 5 provides summary (i.e., *AE*) of the IEEE-CEC2005 results.

We decided to compare the proposed algorithm with the classical version of hill climbing using various values of *bw* where the β value is constant in the β-hill climbing. The comparative Table 6 described in Sect. 4.4. Furthermore, in order to provide more in depth explanations and better evidence to the convergence efficiency of the proposed β-hill climbing, the extensions of hill climbing algorithms such as SA, TS, VNS, and ILS have been also experimented with same IEEE-CEC2005 dataset and compared against the β-hill climbing. The summary of the

Table 3 Test functions

Abb.	Name	Search range	N	$f(x*)$
f_1	Shifted Sphere Function	$[-100, 100]$	10	−450
f_2	Shifted Schwefel's Problem 1.2	$[-100, 100]$	10	−450
f_3	Shifted Rotated High Conditioned Elliptic Function	$[-100, 100]$	10	−450
f_4	Shifted Schwefel's Problem 1.2 with Noise in Fitness	$[-100, 100]$	10	−450
f_5	Schwefels Problem 2.6 with Global Optimum on Bounds	$[-100, 100]$	10	−310
f_6	Shifted Rosenbrock's Function	$[-100, 100]$	10	390
f_7	Shifted Rotated Griewank's Function without Bounds	[0, 600]	10	−180
f_8	Shifted Rotated Ackley's Function with Global Optimum on Bounds	$[-32, 32]$	10	−140
f_9	Shifted Rastrigin's Function	$[-5, 5]$	10	−330
f_{10}	Shifted Rotated Rastrigin's Function	$[-5, 5]$	10	−330
f_{11}	Shifted Rotated Weierstrass Function	$[-0.5, 0.5]$	10	90
f_{12}	Schwefel's Problem 2.13	$[-\pi, \pi]$	10	−460
f_{13}	Expanded Extended Griewank's plus Rosenbrock's Function (F8F2)	$[-5, 5]$	10	−130
f_{14}	Shifted Rotated Expanded Schaffer's F6	$[-100, 100]$	10	−300
f_{15}	Hybrid Composition Function	$[-5, 5]$	10	120
f_{16}	Rotated Hybrid Composition Function	$[-5, 5]$	10	120
f_{17}	Rotated Hybrid Composition Function with Noise in Fitness	$[-5, 5]$	10	120
f_{18}	Rotated Hybrid Composition Function	$[-5, 5]$	10	10
f_{19}	Rotated Hybrid Composition Function with a Narrow Basin for the Global Optimum	$[-5, 5]$	10	10
f_{20}	Rotated Hybrid Composition Function with the Global Optimum on the Bounds	$[-5, 5]$	10	10
f_{21}	Rotated Hybrid Composition Function	$[-5, 5]$	10	360
f_{22}	Rotated Hybrid Composition Function with High Condition Number Matrix	$[-5, 5]$	10	360
f_{23}	Non-Continuous Rotated Hybrid Composition Function	$[-5, 5]$	10	360
f_{24}	Rotated Hybrid Composition Function	$[-5, 5]$	10	260
f_{25}	Rotated Hybrid Composition Function without Bounds	[2, 5]	10	260

Table 4 Nine experiment cases to evaluate the sensitivity of β-hill climbing to its parameters

Experiment cases	β	bw
Case 1	0.5	0.5
Case 2		0.05
Case 3		0.005
Case 4	0.05	0.5
Case 5		0.05
Case 6		0.005
Case 7	0.005	0.5
Case 8		0.05
Case 9		0.005

comparative results has been reported in Table 7 and shows the significant of the proposed method. Note that GRASP can be considered as a special case of ILS and thus is not considered in these experiment [8].

Finally, the results of β-hill climbing are compared against those of some advanced Evolutionary Algorithms using the same functions characterized in Table 8 and the comparative results are summarized in Table 9.

All the experiments are run using a computer with 2.66 Intel Core 2 Quad with 4 GB of RAM. The operating system used is Microsoft Windows Vista Enterprize Service Pack 1. The source code is implemented using MATLAB version 7.6.0.324 (R2008a).

4.3 A sensitivity analysis of the β-hill climbing parameters (β and bw)

This section provides the results of β-hill climbing using IEEE-CEC2005 test functions (f_1–f_{25}) over the nine experimented cases given in Table 4.

In Table 5 where the results of the nine cases using the functions (f_1–f_{25}) is summarized in terms of AE, some best results have been obtained with variable values of β. For example, the best results for f_{13} are obtained when the β is lower in its value. An interesting discussion is based on the nature of search space. This type of functions might have a *rugged* landscape and the best results are obtained with less focus on exploration. On the other hand, some functions like f_5, f_7, f_9–$f_{11}, f_{14}, f_{15}, f_{16}$–$f_{25}$ reach the best results when the value of β has a higher rate. Setting β with higher rate

Table 5 Results of β-hill climbing of nine cases using f_1–f_{25} functions of IEEE-CEC2005 in terms of *AE*

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9
f_1	2.20E−04	**1.00E−09**	1.20E−02	**1.00E−09**	**1.00E−09**	**1.00E−09**	**1.00E−09**	**1.00E−09**	**1.00E−09**
f_2	9.59E−01	5.23E+00	1.79E+01	**1.00E−09**	**1.00E−09**	**1.00E−09**	**1.00E−09**	**1.00E−09**	**1.00E−09**
f_3	6.36E+05	1.04E+06	8.99E+05	1.84E+05	4.57E+04	5.51E+04	1.42E+05	2.80E+04	**2.22E+04**
f_4	2.61E+01	4.02E+01	3.66E+01	**4.87E−03**	2.47E+01	4.81E+01	7.58E+01	1.64E+03	1.11E+03
f_5	3.05E+02	**2.51E+02**	3.23E+02	8.05E+02	6.73E+02	5.22E+02	3.88E+03	3.24E+03	2.44E+03
f_6	7.02E+01	9.56E+01	1.07E+02	4.09E+01	**4.07E+01**	8.08E+02	5.37E+01	4.94E+01	5.68E+01
f_7	1.37E+00	**4.73E−01**	8.16E−01	2.72E+00	1.96E+00	1.74E+00	6.63E+00	4.70E+00	1.74E+00
f_8	2.60E+02	2.01E+01	2.60E+02	2.01E+01	2.01E+01	2.01E+01	2.02E+01	2.02E+01	**2.00E+01**
f_9	2.28E−01	3.60E−04	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{10}	**2.09E+01**	2.34E+01	2.39E+01	4.23E+01	4.28E+01	4.10E+01	7.31E+01	7.47E+01	7.26E+01
f_{11}	5.52E+00	**5.50E+00**	5.76E+00	6.68E+00	6.49E+00	7.54E+00	7.33E+00	8.42E+00	9.33E+00
f_{12}	9.66E+02	4.86E+02	5.54E+02	**1.77E+02**	2.35E+02	3.45E+02	4.52E+02	3.58E+02	1.86E+02
f_{13}	5.24E−01	5.09E−01	3.76E−01	3.15E−01	3.11E−01	**3.09E−01**	4.37E−01	5.35E−01	4.00E−01
f_{14}	3.06E+00	**2.93E+00**	2.97E+00	3.45E+00	3.63E+00	3.44E+00	3.70E+00	3.98E+00	3.88E+00
f_{15}	**1.14E+02**	3.79E+02	1.64E+02	2.03E+02	3.93E+02	3.96E+02	2.29E+02	3.14E+02	3.36E+02
f_{16}	**1.58E+02**	2.76E+02	1.59E+02	1.99E+02	3.81E+02	2.60E+02	4.74E+02	3.64E+02	2.36E+02
f_{17}	**1.62E+02**	3.05E+02	1.62E+02	2.97E+02	2.61E+02	2.05E+02	3.03E+02	4.19E+02	2.89E+02
f_{18}	8.07E+02	8.41E+02	**6.56E+02**	1.03E+03	9.66E+02	9.40E+02	1.06E+03	1.05E+03	1.02E+03
f_{19}	**7.79E+02**	8.32E+02	8.05E+02	1.01E+03	9.26E+02	9.43E+02	1.12E+03	1.06E+03	9.79E+02
f_{20}	8.14E+02	**6.57E+02**	8.50E+02	9.96E+02	1.02E+03	8.60E+02	1.05E+03	1.05E+03	9.57E+02
f_{21}	**6.40E+02**	1.14E+03	7.56E+02	9.42E+02	1.09E+03	1.03E+03	9.45E+02	1.02E+03	1.06E+03
f_{22}	8.03E+02	1.18E+03	**7.66E+02**	9.38E+02	9.82E+02	8.20E+02	1.05E+03	9.93E+02	1.05E+03
f_{23}	7.61E+02	1.34E+03	**6.60E+02**	1.05E+03	9.38E+02	1.10E+03	1.16E+03	1.13E+03	9.82E+02
f_{24}	3.71E+02	6.11E+02	**2.62E+02**	5.72E+02	4.52E+02	4.76E+02	1.07E+03	9.35E+02	8.18E+02
f_{25}	2.62E+02	**2.60E+02**	2.83E+02	5.48E+02	5.26E+02	4.33E+02	8.48E+02	8.12E+02	8.95E+02

Bold values indicate best solution achieved (lowest is best)

Table 6 Comparison between hill climbing and β-hill climbing for IEEE-CEC2005 test functions in terms of *AE*

	Hill climbing			β-Hill climbing		
	Case 1	Case 2	Case 3	Case 1	Case 2	Case 3
f_1	**1.00E−09**	**1.00E−09**	7.49E+02	2.20E−04	**1.00E−09**	1.20E−02
f_2	**1.00E−09**	**1.00E−09**	2.53E+03	9.59E−01	5.23E+00	1.79E+01
f_3	8.48E+04	**2.11E+04**	7.61E+06	6.36E+05	1.04E+06	8.99E+05
f_4	1.50E+04	2.68E+05	1.21E+05	**2.61E+01**	4.02E+01	3.66E+01
f_5	5.62E+03	5.51E+03	1.34E+04	3.05E+02	**2.51E+02**	3.23E+02
f_6	**6.65E+01**	7.24E+01	7.51E+08	7.02E+01	9.56E+01	1.07E+02
f_7	6.82E+01	7.99E+01	1.52E+03	1.37E+00	**4.73E−01**	8.16E−01
f_8	2.60E+02	2.02E+01	**2.01E+01**	2.60E+02	**2.01E+01**	2.60E+02
f_9	1.14E+02	1.30E+02	1.32E+02	2.28E−01	3.60E−04	**0.00E+00**
f_{10}	2.77E+02	2.84E+02	3.50E+02	**2.09E+01**	2.34E+01	2.39E+01
f_{11}	8.31E+00	1.26E+01	1.80E+01	5.52E+00	**5.50E+00**	5.76E+00
f_{12}	3.23E+02	**2.37E+02**	2.39E+02	9.66E+02	4.86E+02	5.54E+02
f_{13}	5.08E−01	9.29E+01	1.68E+02	5.24E−01	5.09E−01	**3.76E−01**
f_{14}	4.87E+00	4.84E+00	4.91E+00	3.06E+00	**2.93E+00**	2.97E+00
f_{15}	9.40E+02	1.49E+03	1.60E+03	**1.14E+02**	3.79E+02	1.64E+02
f_{16}	1.06E+03	1.63E+03	1.71E+03	**1.58E+02**	2.76E+02	1.59E+02
f_{17}	7.89E+02	1.53E+03	1.67E+03	**1.62E+02**	3.05E+02	**1.62E+02**
f_{18}	1.39E+03	2.63E+03	3.04E+03	8.07E+02	8.41E+02	**6.56E+02**
f_{19}	1.48E+03	2.59E+03	2.75E+03	**7.79E+02**	8.32E+02	8.05E+02
f_{20}	1.51E+03	2.83E+03	2.98E+03	8.14E+02	**6.57E+02**	8.50E+02
f_{21}	1.67E+03	1.89E+03	1.94E+03	**6.40E+02**	1.14E+03	7.56E+02
f_{22}	2.23E+03	3.93E+03	7.38E+03	8.03E+02	1.18E+03	**7.66E+02**
f_{23}	1.69E+03	2.07E+03	2.09E+03	7.61E+02	1.34E+03	**6.60E+02**
f_{24}	1.65E+03	1.83E+03	1.88E+03	3.71E+02	6.11E+02	**2.62E+02**
f_{25}	1.87E+03	2.05E+03	2.11E+03	2.62E+02	**2.60E+02**	2.83E+02

Bold values indicate best solution achieved (lowest is best)

leads to higher exploration and lower exploitation, thus the *neutral* landscape can be efficiently navigated.

It can be noted that the value of *bw* for global optimization controls the bandwidth amount of move to the neighboring solutions of the current one. The lower the value of *bw* is, the more precise the results will be, which was borne out by the results.

4.4 A comparison of β-hill climbing and other local search-based methods

Initially, this section provides a comparative analysis between hill climbing and β version of hill climbing, i.e., β-hill climbing. Table 6 summarizes the results obtained by both versions in terms of *AE* for all tested functions.

Three cases of varying *bw* values are used in comparison. Note that, in the three cases, the value of β is fixed to 0.5. In general, for all IEEE-CEC2005 functions, the obtained results by β-hill climbing are significantly better than the obtained results of hill climbing. Indeed, adding a β operator in β-hill climbing is evidently a powerful component for hill climbing and can be easily expanded to other local-based algorithms.

To study the convergence behavior of both hill climbing (HC) and β-hill climbing (Bhc), Fig. 3 plots the trend between both algorithms using the six experimented cases. These cases are selected randomly from all functions, i.e., $f_3, f_5, f_{10}, f_{16}, f_{20}$, and f_{25}. The x-axis in the plots represents the iteration number, while y-axis represents the objective function values. In all plots, β-hill climbing converge faster than classical hill climbing with varying values of *bw*.

In order to validate whether the β-hill climbing can be produced a qualitative results as other local search-based methods, the β-hill climbing is compared with SA, TS, ILS, VNS, and the original HC as reported in Table 7. Note that in Table 7, only the best results produced by each method is reported.

In order to guarantee the fairness of the comparison, the rules and parameters of IEEE-CEC2005 are implemented for all comparative methods. For SA, the initial value of $T_0 = 100$ and $\alpha = 85\,\%$ is used as suggested in [9]. For TS, the $TN = N/2$.

Table 7 Average error rate obtained in IEEE-CEC2005 Special Session in dimension 10

Algorithm	f_1	f_2	f_3	f_4	f_5
SA	**1.00E−09**	**1.00E−09**	7.57E+04	1.49E+05	6.46E+03
TS	**1.00E−09**	**1.00E−09**	1.42E+05	4.02E+05	9.27E+03
ILS	**1.00E−09**	**1.00E−09**	7.70E+06	1.17E+04	6.68E+03
VNS	**1.00E−09**	**1.00E−09**	7.21E+04	2.25E+05	1.18E+04
HC	**1.00E−09**	**1.00E−09**	**2.11E+04**	1.50E+04	5.51E+03
βHC	**1.00E−09**	**1.00E−09**	2.22E+04	**4.87E−03***	**2.51E+02***
Algorithm	f_6	f_7	f_8	f_9	f_{10}
SA	6.69E+01	3.98E+02	2.02E+01	1.41E+02	2.60E+02
TS	9.89E+01	1.70E+03	2.01E+01	1.37E+02	2.67E+02
ILS	2.95E+03	2.35E+01	2.02E+01	7.06E+00	1.38E+02
VNS	7.49E+05	6.02E+02	2.01E+01	1.42E+02	2.48E+02
HC	6.65E+01	6.82E+01	2.01E+01	1.14E+02	2.77E+02
βHC	**4.07E+01***	**4.73E−01***	**2.00E+01**	**0.00E+00***	**2.09E+01***
Algorithm	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
SA	1.30E+01	2.73E+02	1.14E+01	4.90E+00	1.32E+03
TS	1.79E+01	**2.20E+00***	1.34E+02	4.90E+00	1.64E+03
ILS	1.05E+01	1.09E+01	1.61E+00	4.26E+00	4.01E+02
VNS	1.24E+01	3.08E+02	4.75E+00	4.85E+00	1.45E+03
HC	8.31E+00	2.37E+02	5.08E−01	4.84E+00	9.40E+02
βHC	**5.50E+00***	1.77E+02	**3.09E−01***	**2.93E+00**	**1.14E+02***
Algorithm	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
SA	1.68E+03	1.53E+03	2.42E+03	2.56E+03	2.53E+03
TS	7.53E+02	1.58E+03	2.74E+03	2.24E+03	2.50E+03
ILS	3.38E+02	3.63E+02	1.07E+03	1.10E+03	1.08E+03
VNS	1.23E+03	1.58E+03	2.11E+03	2.45E+03	2.51E+03
HC	1.06E+03	7.89E+02	1.39E+03	1.48E+03	1.51E+03
βHC	**1.58E+02***	**1.62E+02***	**6.56E+02***	**7.79E+02***	**6.57E+02***
Algorithm	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}
SA	1.90E+03	3.04E+03	1.66E+03	1.86E+03	1.88E+03
TS	1.99E+03	6.60E+03	2.15E+03	1.91E+03	1.67E+03
ILS	9.97E+02	1.14E+03	1.26E+03	9.60E+02	1.22E+03
VNS	1.85E+03	2.62E+03	2.02E+03	1.81E+03	1.74E+03
HC	1.67E+03	2.23E+03	1.69E+03	1.65E+03	1.87E+03
βHC	**6.40E+02***	**7.66E+02***	**6.60E+02***	**2.62E+02***	**2.60E+02***

Bold values indicate best solution achieved (lowest is best)

* Significant difference in the marked results in comparison with others

Notably, as given in Table 7, the *AE* produced by β-hill climbing are better than those produced by other local search-based methods for almost all IEEE-CEC2005 dataset.

It is worth mentioning that the best results are almost produced using the local search-based methods with an exploration capability rather than the classical HC. The exploration process enables the algorithm to jump from a search space region to another in the hope of moving to a better place that might contain a fitter solution. By means of this process, the local search-based algorithm with an exploration process can yield better results than that with no exploration process like HC.

Table 8 Key to comparative methods

Key	Method name	References
BLXGL50	Hybrid real-coded genetic algorithms with female and male differentiation	[11]
BLX-MA	Adaptive local search parameters for real-coded memetic algorithms	[21]
CoEVO	Real-parameter optimization using the mutation step coevolution	[23]
DE	Real-parameter optimization with differential evolution	[25]
DMS-L-PSO	Dynamic multiswarm particle swarm optimizer with local search	[19]
EDA	Experimental results for the Special Session on Real-Parameter Optimization at CEC 2005: a simple, continuous EDA	[30]
K-PCX	A population-based, steady-state procedure for real-parameter optimization	[26]
G-CMA-ES	A restart CMA evolution strategy with increasing population size	[5]
L-CMA-ES	Performance evaluation of an advanced local search evolutionary algorithm	[4]
L-SaDE	Self-adaptive differential evolution algorithm	[24]
SPC-PNX	Real-parameter optimization performance study on the CEC-2005 benchmark with SPC-PNX	[7]
cHS	Cellular harmony search algorithm	[2]

To further evaluate the performance of β-hill climbing, Mann–Whitney U test is conducted to show whether the produced best Average Errors (*AEs*) are significant. It is worth mentioning that the comparison in the Mann–Whitney U test is done using the best *AEs* from β-hill climbing and the best *AEs* from other local search-based methods. The asterisk (*) means there is a significant differences between the obtained results of two samples. Notably, best *AEs* are produced by β-hill climbing and they are significant.

4.5 A comparison of β-hill climbing and state-of-the-art methods

This section is purposed to compare the proposed β-hill climbing with the other state-of-the-art methods abbreviated in Table 8, where 11 of comparative methods are used. *AE* in Table 9 is used as a comparative factor for the IEEE-CEC2005 functions. The best results in this table have been highlighted in bold (lowest is best).

The authors decide to provide an intensive comparison in order to show the power of the new version of hill climbing in comparison with the latest extensions of optimization methods. Note that the dimensionality has been unified for all comparative methods with β-hill climbing which is expressed in Table 4.

From Table 9, the β-hill climbing obtained very impressive results in comparison with 11 comparative methods using the IEEE-CEC2005 functions. β-hill climbing is able to produce three best *AE* results for f_1, f_2, and f_9, while having produced very comparative results for the others.

5 Conclusion and future work

This paper has proposed a local search-based metaheuristic algorithm for global optimization called β-hill climbing. An operator is added to control the diversity of the search space. At each iteration, the new solution is constructed based on elements selected from the current solution and random elements from the range. The β rate controls the size of randomness. The proposed method is evaluated using IEEE-CEC2005 functions of different characteristics and complexity. The sensitivity analysis has been conducted to show the behavior of β-hill climbing with varying values of β rate and the bandwidth parameter *bw* designed to control the distance of movement from the current solution to the new one. The results suggest that for IEEE-CEC2005 functions, the suitable values of β and *bw* depend on the ruggedness of the search space.

The comparative evaluation process has been conducted using three scenarios: Firstly, a comparison between the classical version of hill climbing and the proposed β-hill climbing where the results appeared to be significantly better in favor of the proposed method. Secondly, the performance of the local search-based algorithms are also compared with the proposed algorithm performance and the comparative results shows the efficiency of the proposed method. Finally, the proposed β-hill climbing was compared with the state-of-the-art techniques using the same test functions. Interestingly, The results were very competitive as the proposed method was able to produce some state-of-the-art results for some test functions.

Using the idea of utilizing a stochastic operator to construct the new solution controlled by β rate in the

Table 9 Average error rate obtained in IEEE-CEC2005 Special Session in dimension 10

Algorithm	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
BLX-GL50	**1.00E−09**	**1.00E−09**	5.71E+02	**1.00E−09**	**1.00E−09**	**1.00E−09**	1.17E−02	2.04E+01	1.15E+00
BLX-MA	**1.00E−09**	**1.00E−09**	4.77E+04	2.00E−08	2.12E−02	1.49E+00	1.97E−01	2.02E+01	4.38E−01
COEVO	**1.00E−09**	**1.00E−09**	1.00E−09	**1.00E−09**	2.13E+00	1.25E+01	3.71E−02	2.03E+01	1.92E+01
DE	**1.00E−09**	**1.00E−09**	1.94E−06	**1.00E−09**	**1.00E−09**	1.59E−01	1.46E−01	2.04E+01	9.55E−01
DMS-L-PSO	**1.00E−09**	**1.00E−09**	**1.00E−09**	1.89E−03	1.14E−06	6.89E−08	4.52E−02	2.00E+01	1.00E−09
EDA	**1.00E−09**	**1.00E−09**	2.12E+01	**1.00E−09**	**1.00E−09**	4.18E−02	4.20E−01	2.03E+01	5.42E+00
IPOP-CMA-ES	**1.00E−09**	**1.00E−09**	1.00E−09	**1.00E−09**	**1.00E−09**	**1.00E−09**	**1.00E−09**	**2.00E+01**	2.39E−01
K-PCX	**1.00E−09**	**1.00E−09**	4.15E−01	7.94E−07	4.85E+01	4.78E−01	2.31E−01	**2.00E+01**	1.19E−01
LR-CMA-ES	**1.00E−09**	**1.00E−09**	**1.00E−09**	1.76E+06	**1.00E−09**	1.00E−09	**1.00E−09**	**2.00E+01**	4.49E+01
L-SADE	**1.00E−09**	**1.00E−09**	1.67E−05	1.42E−05	1.23E−02	1.20E−08	1.99E−02	**2.00E+01**	1.00E−09
SPC-PNX	**1.00E−09**	**1.00E−09**	1.08E+05	**1.00E−09**	**1.00E−09**	1.89E+01	8.26E−02	2.10E+01	4.02E+00
cHS	1.00E−09	1.00E−09	1.49E+07	4.66E+03	1.33E+04	5.84E+07	7.85E+01	2.03E+01	2.69E+01
HC	1.00E−09	1.00E−09	2.11E+04	1.50E+04	5.51E+03	6.65E+01	6.82E+01	2.01E+01	1.14E+02
βHC	**1.00E−09**	**1.00E−09**	2.22E+04	4.87E−03	2.51E+02	4.07E+01	4.73E−01	**2.00E+01**	**1.00E−09**

Algorithm	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}
BLX-GL50	4.97E+00	2.33E+00	4.07E+02	7.50E−01	2.17E+00	4.00E+02	9.35E+01	1.09E+02	4.20E+02
BLX-MA	5.64E+00	4.56E+00	7.43E+01	7.74E−01	**2.03E+00**	2.70E+02	1.02E+02	1.27E+02	8.03E+02
COEVO	2.68E+01	9.03E+00	6.05E+02	1.14E+00	3.71E+00	2.94E+02	1.77E+02	2.12E+02	9.02E+02
DE	1.25E+01	**8.47E−01**	3.17E+01	9.77E−01	3.45E+00	2.59E+02	1.13E+02	1.15E+02	4.00E+02
DMS-L-PSO	3.62E+00	4.62E+00	2.40E+00	3.69E−01	2.36E+00	**4.85E+00**	9.48E+01	1.10E+02	7.61E+02
EDA	5.29E+00	3.94E+00	4.42E+02	1.84E+00	2.63E+00	3.65E+02	1.44E+02	1.57E+02	4.83E+02
IPOP-CMA-ES	**7.96E−02**	9.34E−01	2.93E+01	6.96E−01	3.01E+00	2.28E+02	**9.13E+01**	1.23E+02	**3.32E+02**
K-PCX	2.39E−01	6.65E+00	1.49E+02	6.53E−01	2.35E+00	5.10E+02	9.59E+01	**9.73E+01**	7.52E+02
LR-CMA-ES	4.08E+01	3.65E+00	2.09E+02	4.94E−01	4.01E+00	2.11E+02	1.05E+02	5.49E+02	4.97E+02
L-SADE	4.97E+00	4.89E+00	**4.50E−07**	**2.20E−01**	2.92E+00	3.20E+01	1.01E+02	1.14E+02	7.19E+02
SPC-PNX	7.30E+00	1.91E+00	2.60E+02	8.38E−01	3.05E+00	2.54E+02	1.10E+02	1.19E+02	4.40E+02
cHS	6.17E+01	9.74E+00	1.28E+04	3.90E+00	3.97E+00	3.48E+02	2.73E+02	2.89E+02	8.85E+02
HC	2.77E+02	8.31E+00	2.37E+02	5.08E−01	4.84E+00	9.40E+02	1.06E+03	7.89E+02	1.39E+03
βHC	2.09E+01	5.50E+00	1.77E+02	3.09E−01	2.93E+00	1.14E+02	1.58E+02	1.62E+02	6.56E+02

Algorithm	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}
BLX-GL50	4.49E+02	4.46E+02	6.89E+02	7.59E+02	6.39E+02	**2.00E+02**	4.04E+02
BLX-MA	7.63E+02	8.00E+02	7.22E+02	6.71E+02	9.27E+02	2.24E+02	3.96E+02
COEVO	8.45E+02	8.63E+02	6.35E+02	7.79E+02	8.35E+02	3.14E+02	**2.57E+02**
DE	4.20E+02	4.60E+02	4.92E+02	7.18E+02	5.72E+02	**2.00E+02**	9.23E+02
DMS-L-PSO	7.14E+02	8.22E+02	5.36E+02	6.92E+02	7.30E+02	2.24E+02	3.66E+02
EDA	5.64E+02	6.52E+02	4.84E+02	7.71E+02	6.41E+02	**2.00E+02**	3.73E+02
IPOP-CMA-ES	**3.26E+02**	**3.00E+02**	5.00E+02	7.29E+02	**5.59E+02**	**2.00E+02**	3.74E+02
K-PCX	7.51E+02	8.13E+02	1.05E+03	**6.59E+02**	1.06E+03	4.06E+02	4.06E+02
LR-CMA-ES	5.16E+02	4.42E+02	**4.04E+02**	7.40E+02	7.91E+02	8.65E+02	4.42E+02
L-SADE	7.05E+02	7.13E+02	4.64E+02	7.35E+02	6.64E+02	**2.00E+02**	3.76E+02
SPC-PNX	3.80E+02	4.40E+02	6.80E+02	7.49E+02	5.76E+02	**2.00E+02**	4.06E+02
cHS	1.09E+03	1.08E+03	1.09E+03	9.00E+02	1.30E+03	1.06E+03	8.84E+02
HC	1.48E+03	1.51E+03	1.67E+03	2.23E+03	1.69E+03	1.65E+03	1.87E+03
βHC	7.79E+02	6.57E+02	6.40E+02	7.66E+02	6.60E+02	2.62E+02	2.60E+02

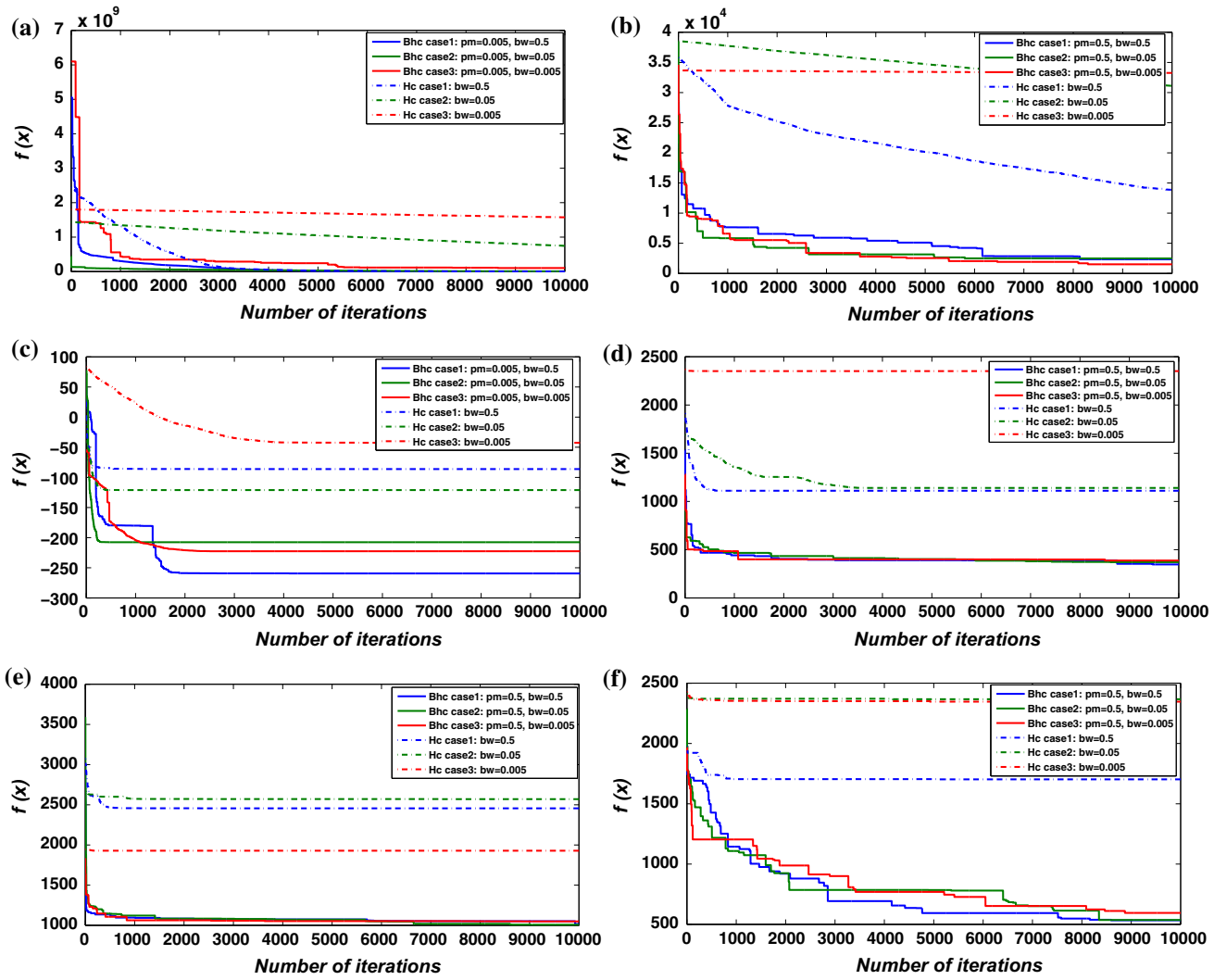

Fig. 3 Plots showing the convergence behavior tendency of both hill climbing and β-hill climbing. **a** f_3 Shifted Rotated High Conditioned Elliptic Function. **b** f_5 Schwefels Problem 2.6 with Global Optimum on Bounds. **c** f_{10} Shifted Rotated Rastrigin's Function. **d** f_{16} Rotated Hybrid Composition Function. **e** f_{20} Rotated Hybrid Composition Function with the Global Optimum on the Bounds. **f** f_{25} Rotated Hybrid Composition Function without Bounds

improvement loop of any local search-based method is a promising research area that is pregnant with fruitful outcomes. It should be emphasized that there needs to be automatic tuning of the β rate for further development to the β-hill climbing. As this research has touched on the global optimization problem, for future research, the combinational optimization problems (COPs) shall be used for evaluation and development of the proposed method.

References

1. Al-Betar MA, Khader AT, Awadallah MA, Abdalkareem ZA (2015) Island-based harmony search for optimization problems. Expert Syst Appl 42:2026–2035
2. Al-Betar MA, Khader AT, Awadallah MA, Alawan MH, Zaqaibeh B (2013) Cellular harmony search for optimization problems. J Appl Math. doi:10.1155/2013/139464
3. Al-Betar MA, Khader AT, Doush IA (2014) Memetic techniques for examination timetabling. Ann OR 218(1):23–50
4. Auger A, Hansen N (2005) Performance evaluation of an advanced local search evolutionary algorithm. In: The 2005 IEEE Congress on Evolutionary Computation (CEC'2005), vol 2. IEEE, pp 1777–1784
5. Auger A, Hansen N (2005) A restart CMA evolution strategy with increasing population size. In: The 2005 IEEE Congress on Evolutionary Computation (CEC'2005), vol 2. IEEE, pp 1769–1776
6. Azar AT, Vaidyanathan S (2015) Computational intelligence applications in modeling and control, vol 575. Springer, Berlin. doi:10.1007/978-3-319-11017-2
7. Ballester PJ, Stephenson J, Carter JN, Gallagher K (2005) Real-parameter optimization performance study on the CEC-2005

benchmark with SPC-PNX. In: The 2005 IEEE Congress on Evolutionary Computation (CEC'2005), vol 1. IEEE, pp 498–505
8. Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput Surv 35(3):268–308
9. Corana A, Marchesi M, Martini C, Ridella S (1987) Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm. ACM Trans Math Softw (TOMS) 13(3):262–280
10. Feo TA, Resende MGC (1995) Greedy randomized adaptive search procedures. J Glob Optim 6(2):109–133
11. García-Martínez C, Lozano M (2005) Hybrid real-coded genetic algorithms with female and male differentiation. In: The 2005 IEEE Congress on Evolutionary Computation (CEC'2005), vol 1. IEEE, pp 896–903
12. Glover F (1986) Future paths for integer programming and links to artificial intelligence. Comput Oper Res 13(5):533–549
13. Glover F, Laguna M (1997) Tabu search. Kluwer, Dordrecht
14. Hansen P, Mladenovic N (1999) An introduction to variable neighborhood search. In: Vo S, Martello S, Osman I, Roucairol C (eds) Metaheuristics: advances and trends in local search paradigms for optimization, chap 30. Kluwer, Dordrecht, pp 433–458
15. Hansen P, Mladenovic N (2001) Variable neighborhood search: principles and applications. Eur J Oper Res 130(3):449–467
16. Hassanien A, Tolba M, Azar A (2014) Advanced machine learning technologies and applications. In: Second international conference, AMLTA, vol 488. Springer. doi:10.1007/978-3-319-13461-1
17. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(1):671–680
18. Lewis R (2008) A survey of metaheuristic-based techniques for university timetabling problems. OR Spectr 30:167–190
19. Liang JJ, Suganthan PN (2005) Dynamic multi-swarm particle swarm optimizer with local search. In: The 2005 IEEE Congress on Evolutionary Computation (CEC'2005), vol 1. IEEE, pp 522–528
20. Loureno HR, Martin OC, Sttzle T (2002) Iterated local search. In: Glover F, Kochenberger G (ed) Handbook of metaheuristics. International series in operations research and management science, vol 57. Kluwer, Dordrecht, pp 321–353
21. Molina D, Herrera F, Lozano M (2005) Adaptive local search parameters for real-coded memetic algorithms. In: The 2005 IEEE Congress on Evolutionary Computation (CEC'2005), vol 1. IEEE, pp 888–895
22. Osman I, Laporte G (1996) Metaheuristics: a bibliography. Ann Oper Res 63(5):511–623
23. Posik P (2005) Real-parameter optimization using the mutation step co-evolution. In: The 2005 IEEE Congress on Evolutionary Computation (CEC'2005), vol 1. IEEE, pp 872–879
24. Qin AK, Suganthan PN (2005) Self-adaptive differential evolution algorithm for numerical optimization. In: The 2005 IEEE Congress on Evolutionary Computation (CEC'2005), vol 2. IEEE, pp 1785–1791
25. Ronkkonen J, Kukkonen S, Price KV (2005) Real-parameter optimization with differential evolution. In: The 2005 IEEE Congress on Evolutionary Computation (CEC'2005), vol 1. IEEE, pp 506–513
26. Sinha A, Tiwari S, Deb K (2005) A population-based, steady-state procedure for real-parameter optimization. In: The 2005 IEEE Congress on Evolutionary Computation (CEC'2005), vol 1. IEEE, pp 514–521
27. Suganthan P, Hansen N, Liang J, Deb K, Chen Y-P, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real parameter optimization. Technical report, Nanyang Technological University
28. Suman B, Kumar P (2006) A survey of simulated annealing as a tool for single and multiobjective optimization. J Oper Res Soc 57:1143–1160
29. Črepinšek M, Liu SH, Mernik M (2013) Exploration and exploitation in evolutionary algorithms: a survey. ACM Comput Surv 45(3):35:1–35:33
30. Yuan B, Gallagher M (2005) Experimental results for the special session on real-parameter optimization at CEC 2005: a simple, continuous EDA. In: The 2005 IEEE Congress on Evolutionary Computation (CEC'2005), vol 2. IEEE, pp 1792–1799
31. Zhu Q, Azar AT (2015) Complex system modelling and control through intelligent soft computations, vol 319. Springer, Berlin. doi:10.1007/978-3-319-12883-2