Cognitive Computation (2018) 10:816–826
https://doi.org/10.1007/s12559-018-9561-1

A Novel Cognitively Inspired State Transition Algorithm for Solving the Linear Bi-Level Programming Problem

Zhaoke Huang[1] · Chunhua Yang[1] · Xiaojun Zhou[1] · Weihua Gui[1]

Received: 20 September 2017 / Accepted: 2 May 2018 / Published online: 12 May 2018

Abstract

Linear bi-level programming (LBLP) is a useful tool for modeling decentralized decision-making problems. It has two-level (upper-level and lower-level) objectives. Many studies have shown that the LBLP problem is NP-hard, meaning it is difficult to find a global solution in polynomial time. In this paper, we present a novel cognitively inspired computing method based on the state transition algorithm (STA) to obtain an approximate optimal solution for the LBLP problem in polynomial time. The proposed method is applied to a supply chain model that fits the definition of an LBLP problem. The experimental results indicate that the proposed method is more efficient in terms of solution accuracy through a comparison to other metaheuristic-based methods using four problems from the literature in addition to the supply chain distribution model. In this study, a cognitively inspired STA-based method was proposed for the LBLP problem. In the future, we expect to extent the proposed method for linear multi-level programming problems.

Keywords Linear bi-level programming · State transition algorithm · Cognitively inspired computing · Global optimization

Introduction

Bi-level programming is a mathematical programming model for decentralized decision-making problems that contain nested optimization problems. In these problems, the upper level is referred to as the leader and the lower level represents the objective of the followers. Bi-level programming has been applied in many different fields, including transportation engineering, organizational design, facility location, production planning, and supply chain management [23, 25]. In the bi-level programming problem, the lower-level optimization problem can be viewed as a constraint for the upper-level optimization problem, meaning only an optimal solution of the lower-level optimization problem is a feasible candidate solution to the upper-level optimization problem. Furthermore, an optimal solution to the upper-level optimization problem may be an infeasible solution to the lower-level optimization problem. This hierarchical optimization structure introduces various difficulties, including non-convexity and disconnectedness. For these reasons, bi-level programming problems can be very difficult to solve. The linear bi-level programming (LBLP) problem is a special type of bi-level programming problem. It is well known that the LBLP problem is an NP-hard problem [8].

Many methods have been proposed to solve the LBLP problem [4, 10]. These methods can be divided into four categories: (1) methods based on vertex enumeration, (2) methods based on Karush–Kuhn–Tucker (KKT) conditions, (3) fuzzy approaches, and (4) methods based on metaheuristics. Because our work is based on a novel metaheuristic algorithm, other metaheuristic-based methods will be discussed in detail in the following sections. In [20], Mathieu et al. proposed a genetic algorithm-based technique for solving the LBLP problem. Their algorithm is able to solve the LBLP problem using only mutations, alleles of decimal numbers, and a survival strategy. Kuo et al. [15] developed an efficient method based on the particle swarm optimization (PSO) algorithm for solving the LBLP problem. Later, in [14], Kuo et al. developed an efficient method based on the hybridization of a genetic algorithm (GA) and PSO for solving the LBLP problem and applied it to a supply chain distribution problem. Zheng et al. [27] presented an exact penalty method based on the classical Kth-best algorithm to

✉ Xiaojun Zhou
michael.x.zhou@csu.edu.cn

[1] School of Information Science and Engineering, Central South University, Changsha 410083, China

solve the weak LBLP problem. Safaei et al. [21] proposed a novel method to find the fuzzy optimal solution of a fully fuzzy linear bi-level programming (FFLBLP) problem by representing all parameters with triangular fuzzy numbers. The given FFLBLP problem was decomposed into three crisp linear programming (CLP) problems with bounded constraints. The three CLP problems were solved separately and the fuzzy optimal solution to the given FFLBLP was obtained by combining the results. He et al. [9] proposed a recurrent neural network (RNN) characterized by differential inclusion for solving the LBLP problem. This model has a low number of state variables and simple structure. It can approximately converge to an optimal solution of the LBLP problem under certain conditions by using non-smooth analysis, the theory of differential inclusions, and a Lyapunov-like method. In [19], Lv et al. proposed a neural network model containing fewer neurons for LBLP programs.

Inspired by the research on natural cognition, many nature-inspired optimization algorithms have emerged in the past few decades [12, 13, 16, 17, 22]. Among them, genetic algorithm and particle swarm optimization have been broadly studied. Genetic algorithm is a global optimization algorithm based on the principles of natural selection and "survival of the fittest." In nature, the stronger the adaptability of an individual, the higher its chances of survival and reproduction. In genetic algorithm, a set of genes in chromosomes surviving in the competitive environment will be inherited by three operating process that are selection, crossover, and mutation. Particle swarm optimization is another computational method for optimization that can mimic the social behavior of a flock of birds and a school of fish. In PSO, each individual flies in the search space with a velocity which is dynamically adjusted according to its own flying experience and its companions' flying experience. There are some advantages to solve the linear bi-level programming problem with the GA and the PSO method, such as favorable global searching ability and fast convergence. However, because of the characteristics of GA and PSO, their solution accuracy is not high enough. Recently, a novel cognitively inspired metaheuristic, state transition algorithm (STA) [31], has been proposed as a powerful tool for global optimization. It aims to imitate human thinking to solve global optimization problem. As is known to us, with respect to global optimization, the goal is to find a global optimal solution as soon as possible. In view of this, in STA, it has deliberately proposed both global (like expansion transformation) and local search (rotation transformation) operators to guarantee the globality and optimality, as well as heuristic operators to accelerate the search process. In STA, a solution to an optimization problem can be treated as a state, and in the meanwhile, an update of current solution using certain state transformation operator is considered as a state transition. Additionally, the strong global search ability and adaptability of the STA have been demonstrated through comparisons to other global optimization algorithms in several real-world applications [5–7, 11, 26, 28–32]. As is known to us, human beings are good at dealing with high-level social cognition problem such as LBLP. Hence, it is important to associate human intelligence with social cognition that brings enormous benefits to practical applications.

In this paper, we propose a novel cognitively inspired computing method based on the state transition algorithm (STA) [31] for solving LBLP problems. We base our method on the STA because of its excellent performance for global search. Several benchmark instances taken from existing studies and an example supply chain model are used to test the performance of the proposed method. The experimental results indicate that the proposed method outperforms other nature-inspired methods in terms of solution accuracy and stability.

The main contributions of our research are fourfold: (1) the STA method is introduced to solve the LBLP problem for the first time; (2) the state transformation operators are only used to generate upper-level candidate solutions, which promotes diversity of solutions while ensuring an optimal solution for the lower-level optimization problem; (3) proper bounds are predetermined for the upper-level variables, which can reduce the search space of the problem; and (4) the proposed metaheuristic is successfully applied to benchmark tests and a supply chain problem.

The remainder of this paper is organized as follows. Section "Background" discusses the basic concepts of the LBLP problem and provides a brief description of the STA. The proposed method based on the STA for solving LBLP problems is presented in Sections "A Novel Cognitively Inspired STA for the LBLP Problem" and "Computational Experiments" contains a thorough discussion of the computational results. Finally, our concluding remarks are presented in Section "Conclusion".

Background

This section briefly discusses the background of the LBLP problem and STA.

Linear Bi-Level Programming (LBLP) Problem

The LBLP problem can be mathematically formulated as follows:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{X}} \quad & F(\mathbf{x},\mathbf{y}) = \mathbf{c_1}\mathbf{x} + \mathbf{d_1}\mathbf{y} \\ \min_{\mathbf{y}\in\mathbb{Y}} \quad & f(\mathbf{x},\mathbf{y}) = \mathbf{c_2}\mathbf{x} + \mathbf{d_2}\mathbf{y} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b}, \end{aligned} \tag{1}$$

where $\mathbf{c_1}, \mathbf{c_2} \in \mathbb{R}^{1\times n}$, $\mathbf{d_1}, \mathbf{d_2} \in \mathbb{R}^{1\times m}$, $\mathbf{A} \in \mathbb{R}^{p\times n}$, $\mathbf{B} \in \mathbb{R}^{p\times m}$, $\mathbf{b} \in \mathbb{R}^{p}$. $F(\mathbf{x}, \mathbf{y})$ is the upper-level optimization problem and $f(\mathbf{x}, \mathbf{y})$ is the lower-level optimization problem. The variables $\mathbf{x}$ and $\mathbf{y}$ are column vectors of size n and m, respectively. Their proper compact convex set are denoted by $\mathbb{X}$ and $\mathbb{Y}$.

The LBLP problem is a special type of bi-level programming problem that is divided into two levels (upper level and lower level). The goal for the upper-level optimization problem is to find an optimal solution with respect to $\mathbf{x}$, with the variables $\mathbf{y}$ acting as parameters, and vice versa for the lower-level optimization problem. A different $\mathbf{x}$ generates a different lower-level optimization problem whose optimal solution must be computed.

Next, we provide several definitions related to the LBLP problem [1]:

Definition 1 The relaxed feasible region (or constraint region) is defined as

$$\Omega = \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^m : \mathbf{x} \in \mathbb{X}, \mathbf{y} \in \mathbb{Y}, \mathbf{Ax} + \mathbf{By} \leq \mathbf{b}\}. \quad (2)$$

Definition 2 For a given (fixed) vector $\bar{\mathbf{x}} \in \mathbb{X}$, the lower-level feasible set is defined as:

$$\Omega(\bar{\mathbf{x}}) = \{\mathbf{y} \in \mathbb{R}^m : \mathbf{By} \leq \mathbf{b} - \mathbf{A}\bar{\mathbf{x}}\}, \quad (3)$$

while the lower-level reaction set (or rational reaction set) is defined as

$$R(\bar{\mathbf{x}}) = \{\mathbf{y} \in \mathbb{R}^m : \mathbf{y} \in \arg\min\{f(\bar{\mathbf{x}}, \hat{\mathbf{y}}) : \hat{\mathbf{y}} \in \Omega(\bar{\mathbf{x}})\}\}. \quad (4)$$

Definition 3 The set

$$\mathrm{IR} = \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^m : \mathbf{x} \in \mathbb{X}, \mathbf{Ax} + \mathbf{By} \leq \mathbf{b}, \mathbf{y} \in R(\mathbf{x})\}, \quad (5)$$

which regroups the feasible points of the LBLP problem, corresponds to the feasible set of the leader, and is known as the induced region (or inducible region).

Definition 4 $(\mathbf{x}^*, \mathbf{y}^*)$ is an optimal solution of the LBLP problem if

$$\begin{aligned} &\mathbf{x}^* \in \mathbb{X}, \\ &\mathbf{Ax}^* + \mathbf{By}^* \leq \mathbf{b}, \\ &\mathbf{y}^* \in R(\mathbf{x}^*), \\ &F(\mathbf{x}^*, \mathbf{y}^*) \leq F(\mathbf{x}^*, \mathbf{y}) \quad for\ all\ \mathbf{y} \in R(\mathbf{x}^*). \end{aligned} \quad (6)$$

To solve the LBLP problem, once the leader has chosen an $\mathbf{x}$, the $\mathbf{x}$ of the follower objective function becomes a constant. Therefore, the objective function of the follower can be simplified to $\min_{\mathbf{y}\in\mathbb{Y}} f(\mathbf{y}) = \mathbf{d_2 y}$. However, according to [23], the solution is different in different situations when solving the LBLP problem. In other words, based on Definition 4, when both levels cooperate completely with each other, the LBLP problem may have an optimal solution.

Cognitively Inspired State Transition Algorithm (STA)

The STA is a novel cognitively inspired global optimization method [5–7, 11, 26, 28–32]. The STA starts with a random solution and searches for optima by generating candidates and comparing them. Unlike a GA or PSO, the STA is an individual-based optimization approach that generates candidates by using state transformations with both local and global operators in alternating fashion. It aims to imitate human thinking to solve global optimization problem. As a result, it has deliberately proposed expansion transformation for global search and rotation transformation for local search to guarantee both the globality and optimality. Furthermore, the STA has also adopted a lot of cognitively inspired strategies. For example, when executing certain transformation operator, a sampling technique is adaptively accommodated to percept the geometrical shape of space formed by candidate solutions. The past experiences are accumulated in the incumbent best solution which is utilized to produce candidate solutions for next generation.

Originally, the framework for the STA was designed by Zhou [31] in 2012 and was inspired by state-space representation in control theory. In the STA, a state represents a solution to an optimization problem and a state transition represents an update to the current solution using state transformation operators. Generally, in the continuous STA, the unified form for the generation of a solution can be defined as follows:

$$\begin{cases} \mathbf{s_{k+1}} = \mathbf{A_k s_k} + \mathbf{B_k u_k}, \\ y_{k+1} = F(\mathbf{s_{k+1}}), \end{cases} \quad (7)$$

where $\mathbf{s_k} \in \mathbb{R}^n$ is a state that corresponds to an optimization problem's candidate solution. $\mathbf{A_k}$ and $\mathbf{B_k}$ are state transition matrices with suitable dimensions, $\mathbf{u_k}$ is a function of $\mathbf{s_k}$, as well as historical states, and F is considered to be the evaluation function.

There are four special state transformation operators for generating candidates for both local and global search.

(1) Rotation transformation

$$\mathbf{s_{k+1}} = \mathbf{s_k} + \alpha \frac{1}{n\|\mathbf{s_k}\|_2} \mathbf{R_r s_k}, \quad (8)$$

where α is called the rotation factor and is a positive constant, $\mathbf{R_r} \in \mathbb{R}^{n\times n}$ is a random matrix whose elements are restricted to the range $[-1, 1]$, and $\|.\|_2$ is the $L_2 - norm$ of a vector.

(2) Translation transformation

$$\mathbf{s_{k+1}} = \mathbf{s_k} + \beta \mathbf{R_t} \frac{\mathbf{s_k} - \mathbf{s_{k-1}}}{\|\mathbf{s_k} - \mathbf{s_{k-1}}\|_2}, \quad (9)$$

where β is called the translation factor and is a positive constant, and $\mathbf{R_t} \in \mathbb{R}$ is a random variable whose elements are restricted to the range [0, 1].

(3) Expansion transformation

$$\mathbf{s_{k+1}} = \mathbf{s_k} + \gamma \mathbf{R_e} \mathbf{s_k}, \quad (10)$$

where γ is called the expansion factor and is a positive constant, and $\mathbf{R_e} \in \mathbb{R}^{n \times n}$ is a random diagonal matrix whose elements follow a Gaussian distribution.

(4) Axesion transformation

$$\mathbf{s_{k+1}} = \mathbf{s_k} + \delta \mathbf{R_a} \mathbf{s_k}, \quad (11)$$

where δ is called the axesion factor and is a positive constant, and $\mathbf{R_a} \in \mathbb{R}^{n \times n}$ is a random diagonal matrix whose elements follow a Gaussian distribution, where only one random position has a nonzero value.

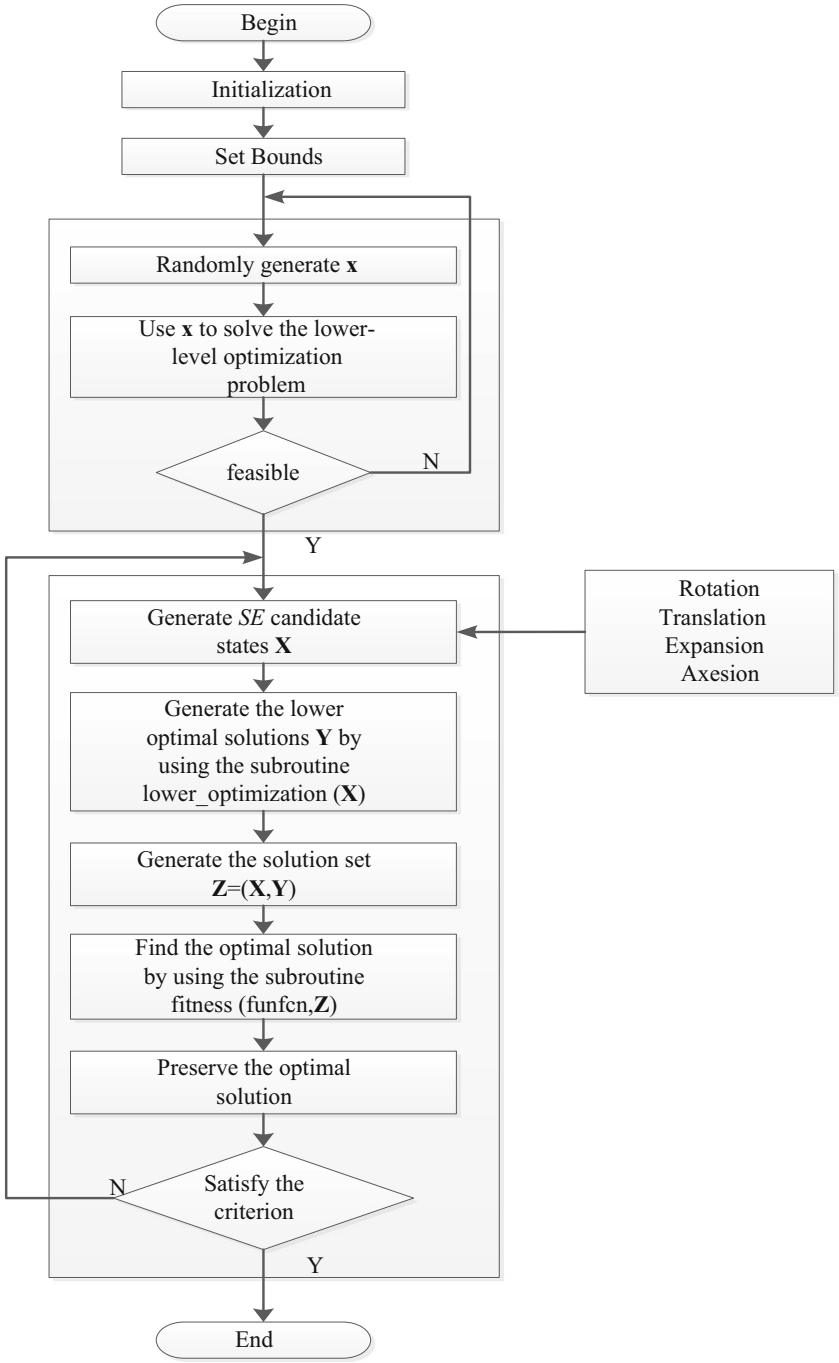

Fig. 1 Flow chart of the proposed method

Table 1 Description of the test problems

Problem 1	Problem 2
$\min\limits_{x_1} f_1 = 2x_1 - 11x_2$	$\min\limits_{x_1} f_1 = -x_2$
$\min\limits_{x_2} f_2 = x_1 + 3x_2$	$\min\limits_{x_2} f_2 = x_2$
$s.t.\ x_1 - 2x_2 \le 4$	$s.t.\ -x_1 - 2x_2 \le 10$
$2x_1 - x_2 \le 24$	$x_1 - 2x_2 \le 6$
$3x_1 + 4x_2 \le 96$	$2x_1 - x_2 \le 21$
$x_1 + 7x_2 \le 126$	$x_1 + 2x_2 \le 38$
$-4x_1 + 5x_2 \le 65$	$-x_1 + 2x_2 \le 18$
$x_1 + 4x_2 \ge 8$	$x_1, x_2 \ge 0$
$x_1, x_2 \ge 0$	

Problem 3	Problem 4
$\min\limits_{x_1} f_1 = -x_1 - 3x_2$	$\min\limits_{x_1,x_2} f_1 = -8x_1 - 4x_2 + 4y_1 - 40y_2 - 4y_3$
$\min\limits_{x_2} f_2 = x_2$	$\min\limits_{y_1,y_2,y_3} f_2 = x_1 + 2x_2 + y_1 + y_2 + 2y_3$
$s.t.\ -x_1 + x_2 \le 3$	$s.t.\ y_1 - y_2 - y_3 \ge -1$
$x_1 + 2x_2 \le 12$	$-2x_1 + y_1 - 2y_2 + 0.5y_3 \ge -1$
$4x_1 - x_2 \le 12$	$-2x_2 - 2y_1 + y_2 + 0.5y_3 \ge -1$
$x_1, x_2 \ge 0$	$x_1, x_2, y_1, y_2, y_3 \ge 0$

The detailed geometrical properties of these state transformation operators can be found in [31]. The STA has both local and global search operators. For example, rotation transformation can be regarded as local search and expansion transformation can be regarded as global search. A multiplication operation called search enforcement (*SE*) is performed on each state transformation operator. As a result, a set of *SE* candidate states are generated, and the state with the best fitness value between the incumbent state and candidate set is selected for the next iteration.

A Novel Cognitively Inspired STA for the LBLP Problem

In the STA, the incumbent state is treated as the best solution achieved so far for an optimization problem in a given problem space. Additionally, the strong global search ability and adaptability of the STA have been demonstrated through comparisons to other global optimization algorithms in several real-world applications [5–7, 11, 26, 28–32]. For these advantages, the STA is used to solve the LBLP problem in our proposed method. A flowchart of the proposed method is presented in Fig. 1. The proposed method based on STA for solving the LBLP problem is described in detail below.

Table 2 Parameter setup

Method	STA	GA	PSO
Parameter	Iterations: 10	Population: 20	Population: 20
	SE: 10	Crossover rate: 0.8	Vmax: 10
	α_{min}:10^{-4} , α_{max}:1	Mutation rate: 0.1	Inertial weight: 1.2–0.2
	α:α_{max} , γ:1 , δ:1	Generations: 200	Iterations: 200

Table 3 The best solutions for Problem 1

	STA	GA	PSO	Lingo
x_1	17.4545	17.4528	17.4535	17.4545
x_2	10.9090	10.9055	10.9070	10.90909
f_1	− 85.0909	− 85.0551	− 85.0700	− 85.0909
f_1 error rate	0%	0.04%	0.02%	N/A
f_2	50.1818	50.16937	50.17450	50.18182
f_2 error rate	0%	0.038%	0.015%	N/A

Step 1 (Parameter setting) *Specify parameters, including search enforcement SE, rotation factor α, translation factor β, expansion factor γ, axesion factor δ, and the maximum number of iterations.*

Step 2 (Set Bounds) *Generate an upper bound (and lower bound) for each $x_i \in \mathbf{x}$ by solving the following problem for $i = 1, \ldots, n$:*

$$\begin{aligned} &\max_{\mathbf{x},\mathbf{y}}(\min) \quad x_i \\ &\text{s.t.} \quad \mathbf{Ax} + \mathbf{By} \le \mathbf{b} \end{aligned} \tag{12}$$

*We use the subroutine set_bounds(**X**) for this process in the pseudocode.*

Step 3 (Generate Initial Solution) *Generate an initial solution $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ by randomly generating $\mathbf{x}$ and using it to solve the lower-level optimization problem. The detailed process is as follows:*

Table 4 Averages and standard deviations for Problem 1

	STA	GA	PSO
$\overline{x_1}$	17.4545	17.43289	17.4417
$\overline{x_2}$	10.9090	10.86578	10.8788
$\overline{f_1}$	−85.0909	−84.657812	−84.85119
f_1 error rate	0%	0.51%	0.28%
$\overline{f_2}$	50.1818	50.03023	50.0781
f_2 error rate	0%	0.30%	0.21%
f_1 standard deviation	0	0.38650	0.189965

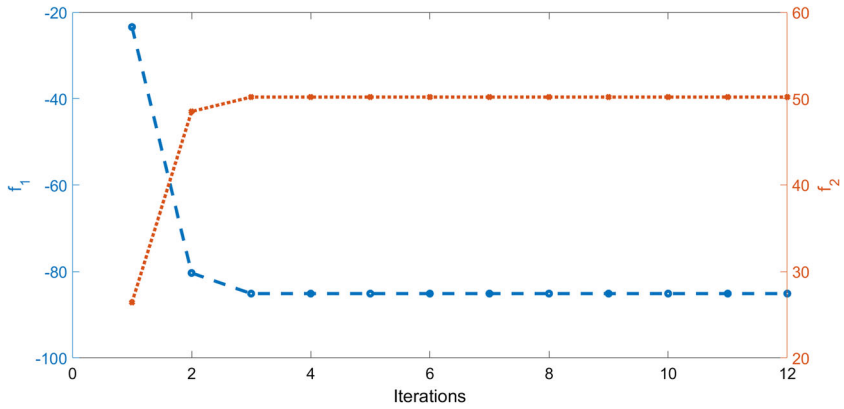

Fig. 2 Iterative curves of the objective function values obtained by the STA for Problem 1

(3.1) *Randomly generate an* $\mathbf{x}$*. This* $\mathbf{x}$ *must fall within a specified field between an upper and lower boundary.*

(3.2) *Use the* $\mathbf{x}$ *to solve the following optimization problem:*

$$\begin{aligned} &\min \quad \mathbf{d_2 y} \\ &\text{s.t.} \quad \mathbf{By} \leq \mathbf{b} - \mathbf{Ax} \end{aligned} \tag{13}$$

This optimization problem is a linear programming problem. Many tools are available to solve linear programming problems. In this study, the interior point algorithm was applied to solve this problem. We use the subrountine $lower\_optimization(\mathbf{X})$ *for this process in the pseudocode.*

(3.3) *If* $\mathbf{y}$ *is an infeasible solution, proceed to Step 3.1. Otherwise, skip to Step 4.*

Step 4 (**STA phase**) *In this stage, we begin the STA phase that can be divided into two steps. The first is state transformation and the second is state updating.*

(4.1) *State transformation*

This step generates new states from the incumbent state using state transformation operators. For example, the $\mathbf{x}$ *in the initial solution* $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ *is treated as the incumbent state. Then, each state transformation operator (rotation, expansion, etc.) is used to generate SE candidate states* $\mathbf{X}$*. It should be noted that* $\mathbf{X}$ *is a* $SE \times |\mathbf{x}|$ *matrix. Next, the subroutine* $lower\_optimization(\mathbf{X})$ *is used to generate lower-optimal solutions* $\mathbf{Y}$*. However, the generated* $\mathbf{x}$ *must be regenerated if* $\mathbf{y}$ *is infeasible. Finally, the next solution set* $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$ *is generated.*

(4.2) *State updating*

We use the subroutine $fitness(funfcn, \mathbf{Z})$ *to find an optimal solution according to the upper-level objective. The best solution thus far is preserved.*

The process for the STA phase is outlined in Algorithm 1.

For the sake of detailed explanation, Algorithm 2 illustrates the process of the expansion function in Algorithm 1.

SE is search enforcement value that represents the size of the candidate solution set. A new best solution is chosen by using a greedy criterion. Additionally, there are three other important parameters that must be set appropriately: rotation factor α, expansion factor γ, and axesion factor δ. Funfcn, Best, and State represent the upper-level objective function, current best solution, and candidate solution set, respectively. The specified termination condition is the maximum number of iterations (Maxiter) in this study.

Table 5 The best solutions for Problem 2

	STA	GA	PSO	Lingo
x_1	16	15.9984	15.9999	16
x_2	11	10.9968	10.9998	11
f_1	− 11	− 10.9968	− 10.9998	− 11
f_1 error rate	0%	0.03%	0.002%	N/A
f_2	11	10.9968	10.9998	11
f_2 error rate	0%	0.03%	0.002%	N/A

Table 6 Averages and standard deviations for Problem 2

	STA	GA	PSO
$\overline{x_1}$	16	15.9041	15.99806
$\overline{x_2}$	11	10.8082	10.9961
$\overline{f_1}$	− 11	− 10.8082	− 10.9961
f_1 error rate	0%	1.74%	0.04%
$\overline{f_2}$	11	10.8082	10.9961
f_2 error rate	0%	1.74%	0.04%
f_1 standard deviation	0	0.168034	0.004014

Algorithm 1 Pseudocode for the STA phase

Input:
maxiter: the maximum number of iterations
SE: search enforcement
Best: the initial solution
Output:
$Best^*$: the optimal solution

repeat
  if $\alpha < \alpha_{min}$ **then**
    $\alpha \leftarrow \alpha_{max}$
  end if
  $Best \leftarrow rotation(funfcn, Best, SE, \alpha, \cdots)$
  $Best \leftarrow expansion(funfcn, Best, SE, \gamma, \cdots)$
  $Best \leftarrow axesion(funfcn, Best, SE, \delta, \cdots)$
  $\alpha \leftarrow \frac{\alpha}{f_c}$
until the specified termination criterion is met
$Best^* \leftarrow Best$

Computational Experiments

Benchmark Tests

We utilized the four benchmark instances listed in Table 1 to assess the proposed method. These instances were selected to illustrate the feasibility and capability of the proposed method, which is able to solve a variety of LBLP problems. Additionally, the performance of the proposed method is compared to that of the GA and PSO methods in [15].

For our representative problems, four benchmark instances were selected from different sources. Thirty independent runs were executed for each problem. All experiments were implemented on a personal computer with an Intel Core i5 Duo 2.6 GHz CPU and 8 GB of RAM using MATLAB.

Table 7 The best solutions for Problem 3 for different methods

	STA	GA	PSO	Lingo
x_1	4	3.9994	4	4
x_2	4	3.9974	4	4
f_1	− 16	− 15.9917	− 16	− 16
f_1 error rate	0%	0.05%	0	N/A
f_2	4	3.94636	4	4
f_2 error rate	0%	1.34%	0	N/A

Algorithm 2 Pseudocode for the expansion transformation in Algorithm 1

Input:
Best: the best solution from the last transfomation
Output:
$Best^*$: the best solution

$\mathbf{X} \leftarrow op\_expand(Best, SE, \gamma)$
if $\mathbf{x}_i$ is out of range **then**
  $\mathbf{x}_i \leftarrow set\_bounds(\mathbf{x}_i)$
end if
$\mathbf{Y} \leftarrow lower\_optimization(\mathbf{X})$
$\mathbf{Z} \leftarrow [\mathbf{X}, \mathbf{Y}]$
$Best^* \leftarrow fitness(funfcn, \mathbf{Z})$

For fair comparison, the solution obtained from Lingo was treated as the best solution. Because each problem was 30 times, an error rate could be calculated as follows:

$$error\ rate = \frac{|f^* - f_M^*|}{f^*} \times 100\% \tag{14}$$

where f^* is the optimal solution based on Lingo, and f_M^* is the optimal solution obtained from the M method.

Table 2 lists the parameter settings for the three test algorithms: STA, PSO, and GA.

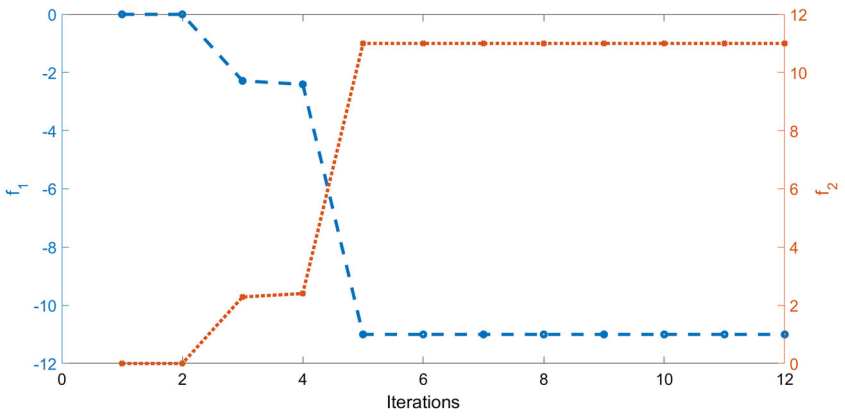

Fig. 3 Iterative curves of the objective function values obtained by the STA for Problem 2

Table 8 Averages and standard deviations for Problem 3

	STA	GA	PSO
$\overline{x_1}$	4	3.986583	3.99908
$\overline{x_2}$	4	3.946363	3.996343
$\overline{f_1}$	− 16	− 15.82567	− 15.98811
f_1 error rate	0%	1.09%	0.07%
$\overline{f_2}$	4	3.946363	3.996343
f_2 error rate	0%	1.34%	0.09%
f_1 standard deviation	0	0.192652	0.009664

Table 9 Results for Problem 4

	STA	GA	PSO	Lingo
x_1	0	0.000	0.0004	0
x_2	0.9	0.898	0.8996	0.9
y_1	0	0.000	0	0
y_2	0.6	0.599	0.5995	0.6
y_3	0.4	0.399	0.3993	0.4
f_1	− 29.2	− 29.1480	− 29.1788	− 29.2
f_1 error rate	0%	0.18%	0.07%	N/A
f_2	3.2	3.1930	3.1977	3.2
f_2 error rate	0%	0.22%	0.07%	N/A

(1) Problem 1

Problem 1 is an example taken from Wen and Hsu [24] and is a standard linear BLP problem. As shown in Tables 3 and 4, we found that the STA provides better results than GA and PSO both for upper-level and lower-level objectives, f_1 and f_2. The GA and PSO do have small error rates on f_1 and f_2, but the STA has no error. Figure 2 presents the iterative curves of the objective function values obtained by the STA for problem 1.

(2) Problem 2

Problem 2 is an example taken from [3]. In problem 2, a major characteristic is that the upper-level and lower-level optimization problems are conflicting. Tables 5 and 6 show that the solutions of STA are better than that of GA and PSO, and without error. Figure 3 presents the iterative curves of the objective function values obtained by the STA for problem 2.

(3) Problem 3

Problem 3 is an example taken from [18]. In problem 3, the lower level does not contain the upper-level decision variables. The accuracy of the STA was significantly higher than that of the GA and PSO. Detailed results are presented in Tables 7 and 8. One can see that the STA is more stable than the GA and PSO. Figure 4 presents the iterative curves of the objective function values obtained by the STA for problem 3.

(4) Problem 4

Problem 4 is an example taken from [2]. In problem 4, the number of variables in the upper level and lower level is different. The experimental results are listed in Tables 9 and 10. Again, the STA achieved better performance than the GA and PSO. Figure 5 presents the iterative curves of the objective function values obtained by the STA for problem 4.

Application to a Supply Chain Problem

For further testing, a supply chain problem was considered to verify the practicability of the proposed method. The problem involves two distribution centers and one manufacturer [14]. The distribution center aims to maximize overall profits and the manufacturer aims to minimize

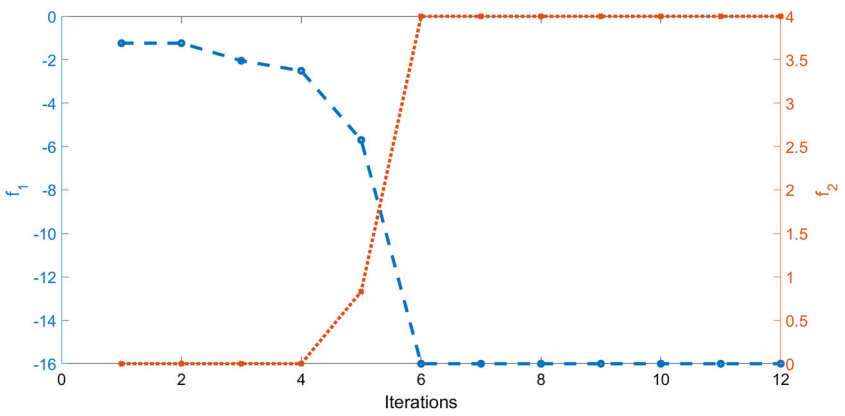

Fig. 4 Iterative curves of the objective function values obtained by the STA for Problem 3

Table 10 Averages and standard deviations for Problem 4

	STA	GA	PSO
$\overline{x_1}$	0	0.15705	0.02192
$\overline{x_2}$	0.9	0.86495	0.86693
$\overline{y_1}$	0	0	0
$\overline{y_2}$	0.6	0.47192	0.56335
$\overline{y_3}$	0.4	0.51592	0.34108
$\overline{f_1}$	− 29.2	− 21.52948	− 24.81256
f_1 error rate	0%	17.19%	4.57%
$\overline{f_2}$	3.2	3.39072	3.1977
f_2 error rate	0%	5.96%	6.21%
f_1 standard deviation	0	3.14432	1.55374

Table 11 Notation for the supply chain problem

Variable	Description
i	The number of distribution centers
j	The number of manufacturers
X_{ij}	The demand of the jth manufacturer from the ith distribution center
$P(X_{ij})$	The product price of product X_{ij}
Y_i	The total amount of products that the ith distribution center has
$U(Y_i)$	The unit cost of product Y_i
C_i	The capacity constraint for the ith distribution center
N_j	The total amount of products that the jth manufacturer needs
$O(X_{ij})$	The costs of other products that the manufacturer must purchase X_{ij}

total costs. Based on the notation in Table 11, the supply chain problem can be mathematically formulated using the following constraints:

The distribution center is subject to the following constraints:

(a) The capacity of the ith distribution center is subject to C_i.

$$\begin{aligned} Y_1 &\leq C_1 \\ Y_2 &\leq C_2 \end{aligned} \tag{15}$$

(b) The total amount of products from all distribution centers should not be lower than that required by the manufacturer.

$$Y_1 + Y_2 \geq N_1 \tag{16}$$

The objective of the distribution center is typically formulated as follows:

$$\begin{aligned} \max_{Y_1,Y_2} \; f_1 = {} & P(X_{11}) \times X_{11} + P(X_{21}) \times X_{21} - U(Y_1) \\ & \times Y_1 - U(Y_2) \times Y_2 \end{aligned} \tag{17}$$

The manufacturer is subject to the following constraints:

(a) The demand of the jth manufacturer from the ith distribution center should not be lower than the amount of products that the ith distribution center has.

$$\begin{aligned} X_{11} &\leq Y_1 \\ X_{21} &\leq Y_2 \end{aligned} \tag{18}$$

The objective of the manufacturer is typically formulated as follows:

$$\begin{aligned} \min_{X_{11},X_{21}} \; f_2 = {} & (P(X_{11}) + O(X_{11})) \times X_{11} \\ & + (P(X_{21}) + O(X_{21})) \times X_{21} \end{aligned} \tag{19}$$

The related parameters are set as $C_1 = 1000$, $C_2 = 500$, $N_1 = 750$, $U(Y_1) = 40$, $U(Y_2) = 50$, $P(X_{11}) =$

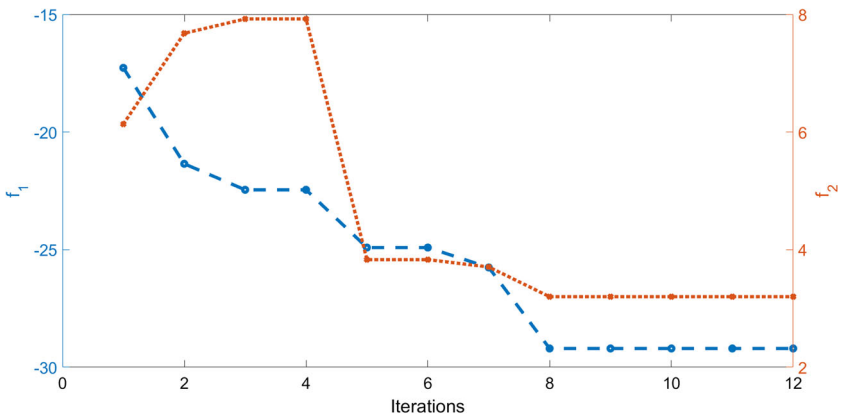

Fig. 5 Iterative curves of the objective function values obtained by the STA for Problem 4

Table 12 Results of the supply chain model

	STA	GA	PSO	Lingo
X_{11}	1000	997.3735	997.7685	1000
X_{21}	500	496.5783	497.1162	500
Y_1	1000	999.5942	999.3908	1000
Y_2	500	498.0463	498.2986	500
f_1	105000	104414.4	104517.9	105000
$f_1$error rate	0%	0.5577%	0.4591%	N/A
f_2	202500	201662.4	201791.7	202500
$f_2$error rate	0%	0.4136%	0.3498%	N/A

110, $P(X_{21}) = 120$, $O(X_{11}) = 20$, and $O(X_{21}) = 25$. Therefore, the supply chain problem is formulated as follows:

$$\begin{aligned} &\max_{Y_1,Y_2} \quad f_1 = 110X_{11} + 120X_{21} - 40Y_1 - 50Y_2 \\ &\min_{X_{11},X_{21}} \quad f_2 = 130X_{11} + 145X_{21} \\ &s.t. \quad \begin{cases} X_{11} \le Y_1 \\ X_{21} \le Y_2 \\ Y_1 \le 1000 \\ Y_2 \le 500 \\ Y_1 + Y_2 \ge 750 \\ X_{ij} \ge 0 \\ Y_i \ge 0 \\ i = 1, 2 \\ j = 1 \end{cases} \end{aligned} \tag{20}$$

This problem is a typical supply chain problem. In this problem, the lower level does not contain upper-level decision variables. This leads to solutions becoming easily trapped in local optima. Therefore, the global solution is difficult to obtain.

In this problem, the proposed method was compared to the GA and PSO. The corresponding solutions are listed in Table 12. The results reveal that using both the GA and PSO is worse than using the proposed method. Among the three methods, the STA has the smallest f_1 and f_2 error rates. Additionally, its standard deviation is also the smallest among all methods. The corresponding converging curve is illustrated in Fig. 6. Using only six iterations can find an optimal solution. The STA can also provide more feasible solutions compared to the other two methods.

Conclusion

In this study, a cognitively inspired STA-based method was proposed for solving LBLP problems. Experimental results demonstrated that the performance of the proposed method is superior to other nature-inspired techniques, such as the GA and PSO, for most problems. This indicates that the STA has better stability than its competitors. Additionally, this study has demonstrated that a supply chain model can be formulated using the LBLP and that the proposed method is suitable for practical applications, particularly supply chains. However, it should be noted that the proposed method has not been applied to large or multi-level instances. In the future, we expect to extend the proposed method for large-scale and linear multi-level programming problems. Additionally, the virtues of the STA will be enhanced to improve learning efficiency so the proposed method can be employed in a variety of applications, such as resource allocation, production plan, and pricing and lot-sizing decisions.

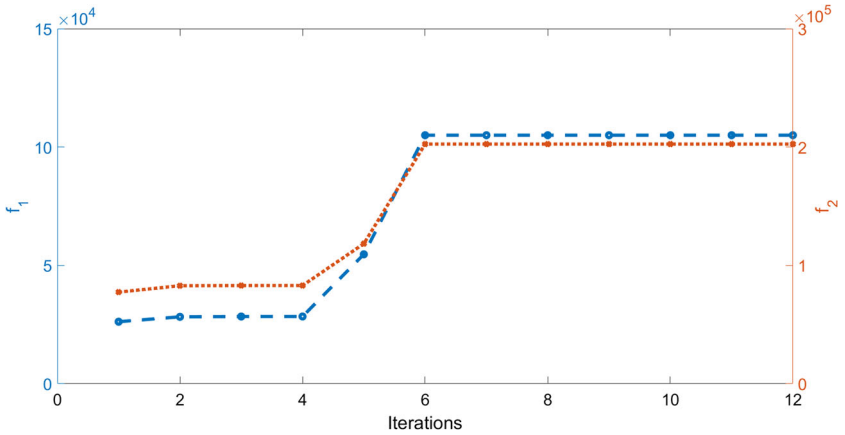

Fig. 6 Iterative curves obtained by the STA for the supply chain model

Funding This study was funded by the National Natural Science Foundation of China (grant numbers 61503416, 61533020, 61533021, 61590921), the 111 Project (grant number B17048), the Innovation-Driven Plan in Central South University (grant number 2018CX012), and Hunan Provincial Natural Science Foundation of China (Grant No. 2018JJ3683).

Compliance with Ethical Standards

Conflict of Interest The authors declare that they have no conflict of interest.

Ethical Approval This article does not contain any experiments with human or animal participants performed by any of the authors.

Informed Consent Informed consent was obtained from all individual participants included in the study.

References

1. Bard JF. Practical bilevel optimization: algorithms and applications. Springer Science & Business Media; 2013. vol. 30.
2. Bard JF, Falk JE. An explicit solution to the multi-level programming problem. Comput Oper Res. 1982;9(1):77–100.
3. Bialas WF, Karwan MH. Two-level linear programming. Manag Sci. 1984;30(8):1004–1020.
4. Dempe S. Foundations of bilevel programming. Springer Science & Business Media. 2002.
5. Han J, Yang C, Zhou X, Gui W. Dynamic multi-objective optimization arising in iron precipitation of zinc hydrometallurgy. Hydrometallurgy. 2017;173:134–148.
6. Han J, Yang C, Zhou X, Gui W. A new multi-threshold image segmentation approach using state transition algorithm. Appl Math Model. 2017;44:588–601.
7. Han J, Yang C, Zhou X, Gui W. A two-stage state transition algorithm for constrained engineering optimization problems. Int J Control Autom Syst 2017:1–13.
8. Hansen P, Jaumard B, Savard G. New branch-and-bound rules for linear bilevel programming. SIAM J Sci Statist Comput. 1992;13(5):1194–1217.
9. He X, Li C, Huang T, Li C, Huang J. A recurrent neural network for solving bilevel linear programming problem. IEEE Trans Neural Netw Learn Syst. 2014;25(4):824–830.
10. Reza Hejazi S, Memariani A, Jahanshahloo G, Sepehri MM. Linear bilevel programming solution by genetic algorithm. Comput Oper Res. 2002;29(13):1913–1925.
11. Huang M, Zhou X, Huang T, Yang C, Gui W. Dynamic optimization based on state transition algorithm for copper removal process. Neural Comput and Applic. 2017:1–13.
12. Javed SG, Majid A, Ali S, Kausar N. A bio-inspired parallel-framework based multi-gene genetic programming approach to denoise biomedical images. Cogn Comput. 2016;8(4):776–793.
13. Kim S-S, McLoone S, Byeon J-H, Lee S, Liu H. Cognitively inspired artificial bee colony clustering for cognitive wireless sensor networks. Cogn Comput. 2017;9(2):207–224.
14. Kuo RJ, Han YS. A hybrid of genetic algorithm and particle swarm optimization for solving bi-level linear programming problem–a case study on supply chain model. Appl Math Model. 2011;35(8):3905–3917.
15. Kuo RJ, Huang CC. Application of particle swarm optimization algorithm for solving bi-level linear programming problem. Comput Math Appl. 2009;58(4):678–685.
16. Li C, Xu Y, Yu X, Ryan C, Huang T. Risk-averse energy trading in multienergy microgrids: a two-stage stochastic game approach. IEEE Trans Indus Inf. 2017;13(5):2620–2630.
17. Li C, Yu X, Yu W, Chen G, Wang J. Efficient computation for sparse load shifting in demand side management. IEEE Trans Smart Grid. 2017;8(1):250–261.
18. Liu Y-H, Hart SM. Characterizing an optimal solution to the linear bilevel programming problem. Eur J Oper Res. 1994;73(1):164–166.
19. Lv Y, Wan Z. Solving linear bilevel programs via a new neural network. Artif Intell Res. 2015;5(1):49.
20. Mathieu R, Pittard L, Anandalingam G. Genetic algorithm based approach to bi-level linear programming. Oper Res. 1994;28(1):1–21.
21. Safaei N, Saraj M. A new method for solving fully fuzzy linear bi-level programming problems. Int J Appl Oper Res. 2014;4(1):39–46.
22. Siddique N, Adeli H. Nature-inspired chemical reaction optimisation algorithms. Cogn Comput. 2017:1–12.
23. Wen U-P, Hsu S-T. Linear bi-level programming problems–a review. J Oper Res Soc. 1991:125–133.
24. Wen U-P, Yang YH. Algorithms for solving the mixed integer two-level linear programming problem. Comput Oper Res. 1990;17(2):133–142.
25. White DJ, Anandalingam G. A penalty function approach for solving bi-level linear programs. J Glob Optim. 1993;3(4):397–419.
26. Zhang F, Yang C, Zhou X, Gui W. Fractional-order pid controller tuning using continuous state transition algorithm 2006:1–10. Neural Comput Applic. 2018;29(10):795–804.
27. Zheng Y, Fang D, Wan Z. A solution approach to the weak linear bilevel programming problems. Optimization. 2016;65(7):1437–1449.
28. Zhou X, Gao DY, Simpson AR. Optimal design of water distribution networks by a discrete state transition algorithm. Eng Optim. 2016;48(4):603–628.
29. Zhou X, Gao DY, Yang C, Gui W. Discrete state transition algorithm for unconstrained integer optimization problems. Neurocomputing. 2016;173:864–874.
30. Zhou X, Shi P, Lim C-C, Yang C, Gui W. A dynamic state transition algorithm with application to sensor network localization. Neurocomputing. 2018;273:237–250.
31. Zhou X, Yang C, Gui W. State transition algorithm. J Indus Manag Optim. 2012;8(4):1039–1056.
32. Zhou X, Yang C, Gui W. Nonlinear system identification and control using state transition algorithm. Appl Math Comput. 2014;226:169–179.