基于 Pareto 支配的双目标优化求解非线性双层规划问题

吴 军[1] 严丽娜[2]

[1](宁夏大学新华学院 宁夏 银川 750021)

[2](北方民族大学医学影像技术系 宁夏 银川 750021)

摘 要 双层规划问题是一类具有双层递阶结构的系统优化问题。采用 Pareto 支配的双目标优化策略求解非线性双层规划问题。利用 K-T 条件把双层规划问题等价转化单层规划问题,进而结合约束部分建立可行性度量目标形成双目标规划问题。在基本的差分进化算法框架中融入非负的最小二乘曲线拟合判断候选解的可行性,构造基于动态概率的 Pareto 支配选择策略挑选下一代个体,解决种群容易陷入局部最优的缺陷。15 个标准函数的测试结果对比显示,该算法在求解非线性双层规划问题中具有较好的全局寻优能力、较低的计算复杂度、较强的稳定性和适用性,可以获得全局最优解。

关键词 非线性双层规划 双目标规划 差分进化 Pareto 支配 K-T 条件

中图分类号 TP301.6 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2020.03.046

BI-OBJECTIVE OPTIMIZATION FOR SOLVING NONLINEAR BILEVEL PROGRAMMING PROBLEMS BASED ON PARETO DOMINATION

Wu Jun[1] Yan Li'na[2]

[1](*Xinhua College*, *Ningxia University*, *Yinchuan 750021*, *Ningxia*, *China*)

[2](*Department of Medical Imaging Technology*, *North Minzu University*, *Yinchuan 750021*, *Ningxia*, *China*)

Abstract The bilevel programming problem is a kind of system optimization problem with bilevel hierarchical structure. In this paper, bi-objective optimization strategy dominated by Pareto is used to solve the nonlinear bilevel programming problem. We used the K-T condition to convert the bi-level programming problem into a single-level planning problem, and then combined the constraint part to establish a feasible measurement target to form a bi-objective programming problem. In the basic differential evolution algorithm framework, non-negative least squares curve fitting was used to judge the feasibility of the candidate solution. The Pareto domination selection strategy based on dynamic probability preference was constructed to select the next generation of individuals, which solved the defect that the population was easy to fall into local optimum. The comparison of test results of 15 standard functions shows that the proposed algorithm has better global optimization ability, lower computational complexity, stronger stability and applicability in solving nonlinear bilevel programming problems, and it can obtain the global optimal solution.

Keywords Nonlinear bilevel programming Bi-objective programming Differential evolution Pareto domination K-T condition

0 引 言

一般的非线性双层规划(Nonlinear Bilevel Programming,NBLP)问题通常是非凸不可微的,文献[1]证明了即使搜索局部最优解,NBLP 问题仍是 NP 难问题。传统的方法求解 NBLP 问题主要有以下几类:关于线性规划的极点法、分支定界算法、下降算法、罚函数法和信赖域法[2-7]等。当 NBLP 问题变得复杂时,这些传统算法将失去作用,在求解时很难获得全局最

收稿日期:2019-03-23。2017 宁夏高等学校科学研究项目(NGY2017224)。**吴军**,讲师,主研领域:智能优化算法及其应用。**严丽娜**,助教。

优解,而智能优化算法对函数要求较低且具有较强的全局搜索能力,因此被逐渐用于求解此类问题。

Mathieu 等[8]首次提出将遗传算法应用到求解双层线性规划问题,在算法中使用嵌套策略,用线性规划和遗传算法分别求解下层和上层规划问题。在解决双层规划问题中嵌套策略是一个非常受欢迎的方法,然而,当遇见大规模、复杂问题时,这种方法将出现耗时高甚至不能求解的情况。为了改善这个问题,Sinha 等[9]将二次逼近嵌套在进化算法中,使下层规划问题的解无限接近最优解,在有限的计算量下有效地解决了复杂的双层规划问题。何胜学等[10]针对建立的停车设施选择和出行路线选择的双层规划模型,上下层问题通过设施选择概率函数实现了有效关联,在嵌套策略下设计了有效的算法求解模型。刘丹等[11]针对离散交通网络设计的大规模双层规划问题,提出一种机器学习-优化混合算法,上下层问题利用不同算法分别求解。Wang 等[12]针对同样的问题,利用新的约束处理机制提高了算法的搜索能力,改善了个体的质量和算法的收敛速度。但这个方法对于下层问题的非凸性无能为力,为了解决这个问题,Wang 等[13]在算法中融合了传统的确定性方法求解下层问题,实验结果显示,新的进化算法可以求解各种双层规划问题。Han 等[14]依据多层规划问题的求解思路,设计出层次粒子群算法求解双层和三层规划问题,实验结果显示,层次粒子群算法对于非线性和大规模多层规划问题效果显著。Kuo 等[15]将遗传算法和粒子群算法融合构造混合算法,拥有同样思路的还有文献[16],他们将粒子群算法和差分进化算法混合求解实际定价和批量决策问题。Li 等[17]将一般的双层规划模型转变为双目标双层规划模型,他们认为在双目标规划中,两个决策者经过“讨价还价”而达到双方都满意的结果并不违背双层规划模型的求解理论,反而可以给出一种形象直观的解帮助决策者在贸易中找到最好的权衡。但是区别于双目标优化问题的 Pareto 最优集,双层规划问题只需得到一个最优解,文献[17]中该问题并没有得到解决。

综上所述,智能优化算法在复杂的优化问题中得到了成功的应用,大多数文章都利用 KT 条件转化双层规划为单层规划求解,以双目标优化思想解决双层规划问题的文献却很少见。文献[17]虽提出了双目标双层规划模型,但对 Pareto 最优集在双层规划问题中的处理并没有给出合理的解决方案。本文将非线性双层规划问题转换为双目标规划问题,在基本的差分进化(Differential Evolution,DE)算法框架中融合非负的最小二乘曲线拟合判定解的可行性,构造了基于动态概率偏好的 Pareto 支配选择策略。以动态的概率偏好保护候选解朝最优目标前进的同时避免了算法陷入局部最优,提高全局寻优能力,成功地将多目标优化算法的思想用于求解双层规划问题。

1 双层规划问题的等价转换

1.1 双层规划模型

考虑如下非线性双层规划问题:

$$\begin{cases}\min\limits_{x\in X} & F(x,y) & G(x,y)\leqslant 0\\ \min\limits_{y\in Y} & f(x,y) & g(x,y)\leqslant 0\end{cases} \tag{1}$$

式中:$F,f:\mathbf{R}^n\times\mathbf{R}^m\to\mathbf{R}$,$G:\mathbf{R}^n\times\mathbf{R}^m\to\mathbf{R}^p$,$g:\mathbf{R}^n\times\mathbf{R}^m\to\mathbf{R}^q$,上层目标函数 $F(x,y)$ 和约束函数 $G(x,y)$ 是非凸不可微的,下层目标函数 $f(x,y)$ 和约束函数 $g(x,y)$ 是凸并且可微的。

x 为上层决策变量,y 为下层决策变量。$\boldsymbol{X}$、$\boldsymbol{Y}$ 分别代表上层变量和下层变量的搜索空间:

$$\boldsymbol{X}=\{(x_1,x_2,\cdots,x_n)^{\mathrm{T}}\in\mathbf{R}^n\mid\mu_i^l\leqslant x_i\leqslant\mu_i^u,i=1,2,\cdots,n\} \tag{2}$$

$$\boldsymbol{Y}=\{(y_1,y_2,\cdots,y_n)^{\mathrm{T}}\in\mathbf{R}^m\mid\xi_j^l\leqslant\xi_j\leqslant\xi_j^u,j=1,2,\cdots,m\} \tag{3}$$

式中:μ_i^l、μ_i^u、ξ_j^l 和 ξ_j^u($i=1,2,\cdots,n,j=1,2,\cdots,m$)都是实数。

双层规划问题最优解的相关概念如下所述:

定义 1[18-19]

1)约束空间:$S=\{(x,y)\mid G(x,y)\leqslant 0,g(x,y)\leqslant 0\}$;

2)对于固定的 $x\in\boldsymbol{X}$,下层问题的可行域:$S(x)=\{y\mid g(x,y)\leqslant 0\}$;

3)S 在上层决策空间的投影:$S(X)=\{x\mid\exists y,(x,y)\in S\}$;

4)对每个 $x\in S(x)$,下层问题的合理反应集:$M(x)=\{y\mid y\in\arg\min\{f(x,y),y\in S(x)\}\}$;

5)诱导域:$IR=\{(x,y)\mid(x,y)\in S,y\in M(x)\}$;

6)双层规划问题的最优解集:$OS=\{(x,y)\mid(x,y)\in\arg\min F(x,y),(x,y)\in IR\}$。

1.2 双层规划模型的等价形式

对任意 $x\in S(x)$,下层规划模型如下:

$$\begin{aligned}&\min_{y\in Y}\quad f(x,y)\\ &\text{s.t.}\quad g(x)\leqslant 0\end{aligned} \tag{4}$$

对于固定的 x,根据最优性条件可知,式(4)等价于求解 Kuhn-Tucher 点问题[20-21]:

$$\begin{cases}\nabla_y f(x,y)+(\nabla_y g(x,y))^{\mathrm{T}}\boldsymbol{U}=0\\ \boldsymbol{U}^{\mathrm{T}}g(x,y)=0\\ \boldsymbol{U}\geqslant 0\end{cases} \tag{5}$$

$(\nabla_y g(x,y))^{\mathrm{T}}$是一个 $m\times q$ 的矩阵,它的第 i 列是 $\nabla_y g_i(x,y)$(也就是函数 g 关于 y 在点(x,y)上的 Jacobian 矩阵的转置矩阵),$\boldsymbol{U}$ 是 q 维列向量,表示 Lagrange 算子。因此,双层规划问题可以转换为单层规划问题:

$$\begin{aligned}&\min_{x,y,U}\quad F(x,y)\\ &\text{s.t.}\quad E_k(x,y,U)=0\quad k=1,2,\cdots,m+1\\ &\qquad I_r(x,y,U)\leqslant 0\quad r=1,2,\cdots,p+q\\ &\qquad U\geqslant 0\end{aligned}\tag{6}$$

式中:$E_k(x,y,\boldsymbol{U})$是式(5)中前两个方程左边的所有函数;$I_r(x,y,\boldsymbol{U})$是 $G(x,y)$和 $g(x,y)$所有的不等式约束的部分。

1.3　双目标优化模型

我们将式(6)中 $E_k(x,y,\boldsymbol{U})$、$I_r(x,y,\boldsymbol{U})$和 $\boldsymbol{U}$ 结合组成约束违反函数,建立一个特殊的优化问题——双目标优化。根据两个目标函数值,依据动态概率偏好的 Pareto 支配策略选取优秀个体。

设 $Q=(\boldsymbol{x}^{\mathrm{T}},\boldsymbol{y}^{\mathrm{T}},\boldsymbol{U}^{\mathrm{T}})^{\mathrm{T}}$,$\boldsymbol{Q}^{*}=(\boldsymbol{x}^{*\mathrm{T}},\boldsymbol{y}^{*\mathrm{T}},\boldsymbol{U}^{*\mathrm{T}})^{\mathrm{T}}$,$\boldsymbol{Q}_1=(\boldsymbol{x}_1^{\mathrm{T}},\boldsymbol{y}_1^{\mathrm{T}},\boldsymbol{U}_1^{\mathrm{T}})^{\mathrm{T}}$,$\boldsymbol{Q}_2=(\boldsymbol{x}_2^{\mathrm{T}},\boldsymbol{y}_2^{\mathrm{T}},\boldsymbol{U}_2^{\mathrm{T}})^{\mathrm{T}}$,记 $I(\boldsymbol{Q})=I(\boldsymbol{x},\boldsymbol{y},\boldsymbol{U})$,令:

$$\begin{cases}I_{p+q+k}(\boldsymbol{Q})=|E_k(\boldsymbol{Q})|\\ I_{p+q+k+r}(\boldsymbol{Q})=-\boldsymbol{U}(r)\end{cases}\tag{7}$$

式中:$k=1,2,\cdots,m+1$;$r=1,2,\cdots,q$;$I_t=\max\{0,I_t\}$;$t=1,2,\cdots,p+2q+m+1$。当 $I_t=0$ 时,表示个体(x,y)为可行解,否则为不可行解。显然,对于式(6),一个可行解优于不可行解;两个可行解中,目标函数值小的解更优秀;但两个不可行解比较时,就无法确认谁好谁坏。因此构造如下双目标优化问题:

$$\min\{F(\boldsymbol{Q}),I(\boldsymbol{Q})\}\tag{8}$$

式中:$I(\boldsymbol{Q})=\max\{0,I_1,I_2,\cdots,I_{p+2q+m+1}\}$,表示约束违反中最大的一个,记为可行性度量函数。同时优化目标函数 $F(\boldsymbol{Q})$和可行性度量函数 $I(\boldsymbol{Q})$,使种群朝向可行区域和全局最优解前进。

2　双目标优化策略的差分进化算法

2.1　差分进化算法

差分进化[22](Differential Evolution,DE)是一种基于种群的启发式全局搜索算法,它主要包括变异和交叉操作。

(1) 变异操作。在进化中的每一代 in,变异操作会被应用到每一个个体 $\boldsymbol{Q}_i^{in}$(目标向量),产生一个相关的变异个体 $\boldsymbol{V}_i^{in}$,本文采用了常用的 DE/best/2 变异策略,具体如下:

$$\boldsymbol{V}_i^{in}=\boldsymbol{Q}_{\mathrm{best}}^{in}+F\cdot(\boldsymbol{Q}_{r1}^{in}-\boldsymbol{Q}_{r2}^{in})+F\cdot(\boldsymbol{Q}_{r3}^{in}-\boldsymbol{Q}_{r4}^{in})\tag{9}$$

式中:$\boldsymbol{Q}_{\mathrm{best}}^{in}$表示在第 in 代中最好的个体;$\boldsymbol{Q}_{r1}^{in}$、$\boldsymbol{Q}_{r2}^{in}$、$\boldsymbol{Q}_{r3}^{in}$和 $\boldsymbol{Q}_{r4}^{in}$是随机选取的四个个体;F 是缩放因子,是控制进化的重要参数。

(2) 交叉操作。为了提高种群的多样性,交叉操作针对变异后的个体 $\boldsymbol{V}_i^{in}$和目标向量 $\boldsymbol{Q}_i^{in}$进行交叉操作产生子代候选个体 $\boldsymbol{S}_i^{in}$,具体操作如下:

$$\boldsymbol{S}_{i,j}^{in}=\begin{cases}\boldsymbol{V}_{i,j}^{in} & rand(0,1)\leqslant CR\ \text{或}\ j=j_{\mathrm{rand}}\\ \boldsymbol{O}_{i,j}^{in} & \text{其他}\end{cases}\tag{10}$$

式中:$rand(0,1)$是$[0,1]$区间的随机数;CR 是交叉概率;j_{rand}是$[1,D]$区间内随机产生的整数。

2.2　可行性的度量

约束优化问题最关键的步骤是判定解的可行性。对于式(6)中的约束部分,很多情况下满足 $E_k(x,y,U)=0$ 是极其困难的,而容忍误差的设定没有具体的标准去衡量。本文依据最小二乘曲线拟合原理,在有限的计算量中,寻找每个候选解对应的 Lagrange 算子 $U'>0$,使函数$|E_k(x,y,U')|$的值尽可能小。

对于 $E_k(x,y,U)=0$,将 U 看作方程中的未知量,将其改写成方程组形式 $\boldsymbol{A}U-\boldsymbol{B}=0$,其中 $\boldsymbol{A}$ 是系数矩阵,$\boldsymbol{B}$ 是常数向量,那么非负的最小二乘曲线拟合就相当于求解如下最小化问题:

$$Ie=\min_{U}\|\boldsymbol{A}\cdot U-\boldsymbol{B}\|_2^2\tag{11}$$

若 $Ie=0$,证明使用最小二乘曲线拟合的程度最高,等式约束的误差为零,得到了精确的候选解;Ie 的值越大,说明拟合程度越低,约束违反值越大。

2.3　动态概率偏好的 Pareto 支配选择

定义 2

(1) Pareto 支配:对于解 $\boldsymbol{Q}_1$ 和 $\boldsymbol{Q}_2$ 满足 $F(\boldsymbol{Q}_1)\leqslant F(\boldsymbol{Q}_2)$和 $I(\boldsymbol{Q}_1)\leqslant I(\boldsymbol{Q}_2)$时,且至少存在 $F(\boldsymbol{Q}_1)<F(\boldsymbol{Q}_2)$和 $I(\boldsymbol{Q}_1)<I(\boldsymbol{Q}_2)$中的一项,则称解 $\boldsymbol{Q}_1$ 支配 $\boldsymbol{Q}_2$;

(2) Pareto 最优:设 Ω 是一解集,如果解 $\boldsymbol{Q}_1$ 是当前 Ω 集合中 Pareto 最优的,当且仅当不存在解 $\boldsymbol{Q}\in\Omega$,使得解 $\boldsymbol{Q}$ 支配 $\boldsymbol{Q}_1$;

(3) Pareto 最优集:设 Ω 是算法目前为止所有解的集合,那么关于 Ω 中所有 Pareto 最优解被称为 Pareto 最优集。

在通常情况下,双层规划问题的求解理论并不等同于双目标优化问题,因此它不能直接转换为双目标优化问题(上下两层的目标函数作为平行的两个目标函数),也就是说,双层规划问题的最优解不能等同于双目标优化问题的 Pareto 最优解。

根据以上定义,对于双目标优化问题(式(8)),如

果解 $\boldsymbol{Q}_1$ 支配 $\boldsymbol{Q}_2$，那么解 $\boldsymbol{Q}_1$ 优于 $\boldsymbol{Q}_2$。如果解 $\boldsymbol{Q}_1$ 和 $\boldsymbol{Q}_2$ 互不支配，那将依概率选择目标函数或可行性度量值较大的解。本文中的双目优化问题（式(8)）等价于单层规划问题（式(6)），因此式(8)的 Pareto 最优解是式(6)的最优解，也就是式(1)的解，具体内容可参照文献[12]中定理 1。

设置如下的动态概率 Pareto 支配选择（DpPa）方案：

（1）两个都是可行解或者两个不可行解。按照 Pareto 支配选择一个解进入下一代种群。

（2）可行解支配不可行解。*P*、*NewP* 分别代表两代种群，*I*、*NewI* 分别代表两代解的可行性度量值，*F*、*NewF* 分别代表两代解的目标函数值，*randp* 代表[0,1]的随机数，$p_d \in [p_l, p_u]$ 代表动态概率值，它的取值随着迭代次数的增加而减小。p_d 的计算公式如下：

$$p_d = \frac{in_{\max} - in}{in_{\max}} \times (p_u - p_l) + p_l$$

具体的伪代码如下所示：

```
if I = 0 & New I≠0 & F < NewF
    if rand p < p_d
      Next P←P
    else
      Next P←NewP
    end
  end
```

通过这种选择策略，一个可行解支配一个不可行解，我们以较大概率选择可行解。随着迭代的进行，种群趋于集中，选择可行解的概率逐渐减小到最低下限值 p_l。相反以较小的概率选择不可行解，随着迭代的增加选择不可行解的概率逐渐增大，最大概率限定为 $p_u - p_l$。以大概率保护优秀可行解进入下一代种群，以小概率保留不可行解进入下一代种群，增加种群的多样性以及全局搜索能力。

（3）一个可行解和一个不可行解互不支配。这种情况下近似于第二种，不可行解的目标函数值优于可行解的，在动态概率的设置上更倾向于不可行解，因此 $p_d \in [p'_l, p_u]$，$p'_l < p_l$。

（4）两个不可行解互不支配。以大概率选择可行性度量值小的，使种群快速朝着可行区域进化，本文设置定值 $p_d = 0.7$。

2.4 基于 Pareto 支配的双目标差分进化算法

基于 Pareto 支配的双目标差分进化算法（DPDE）求解非线性双层规划问题，具体的算法流程如下：

步骤 1 设置初始参数，种群规模 p^s，交叉概率 *cr*，动态概率上限和下限 p_u、p_l，当前迭代次数 *in*，缩放因子 *F*，最大迭代次数 $in_{\max}$。

步骤 2 初始化种群，在变量范围内随机产生初始种群 $P = \{\boldsymbol{Q}_i \mid i = 1, 2, \cdots, ps\}$，计算每个个体的目标函数值 $F(\boldsymbol{Q})$ 和约束违反值 $I(\boldsymbol{Q})$。

步骤 3 依据约束违反值 $I(\boldsymbol{Q})$ 判断个体的可行性，将可行解中目标函数值 $F(\boldsymbol{Q})$ 最小的个体作为最好个体 Q_{best}，如果不存在可行解，那么将不可行解中约束违反值 $I(\boldsymbol{Q})$ 最小的个体作为最好个体 Q_{best}。

步骤 4 随机选择 4 个个体 Q_{r1}、Q_{r2}、Q_{r3} 和 Q_{r4}，根据式(9)进行变异操作，产生变异个体 $\boldsymbol{V}_i$，根据式(10)进行交叉操作，产生子代候选个体 $\boldsymbol{S}_i$，计算 $\boldsymbol{S}_i$ 目标函数值 $F(\boldsymbol{S}_i)$ 和可行性度量值 $I(\boldsymbol{S}_i)$，依据 DaPa 策略比较 $\boldsymbol{Q}_i$ 与 $\boldsymbol{S}_i$ 的目标函数值和可行性度量值，选择较好个体。

步骤 5 更新种群，判断算法终止条件，如果 $in < in_{\max}$，返回步骤 3，否则终止算法运行，输出结果。

3 实验结果及分析

算法 DPDE 在文中的参数设置如下：种群规模 $ps = 20$，交叉概率 $cr = 0.6$，动态概率上限 $p_u = 1$，下限 $p_l = 0.9$，$p'_l = 0.8$，初始迭代次数 $in = 0$，缩放因子 $F = 0.6$，最大迭代次数 $in_{\max} = 3\ 000$。本文选用文献[12,14]中 15 个测试函数，如表 1 所示。实验测试环境为：处理器 Intel(R) Core(TM) i3-2350，CPU @2.30 GHz，内存为 6 GB，编程软件为 MATLAB R2010a。

表 1 测试函数及其相关来源

函数	函数来源文献[12]	函数来源文献[14]
1	函数 1	函数 19
2	函数 2	函数 20
3	函数 3	函数 1
4	函数 5	函数 21
5	函数 6	函数 22
6	函数 7	函数 23
7	函数 8	函数 24
8	函数 9	函数 25
9	函数 11	函数 3
10	函数 12	函数 17
11	函数 13	函数 4
12	函数 14	函数 5
13	函数 17	函数 6
14	函数 23	函数 9
15	函数 26	函数 12

对于每个测试函数，DPDE 算法独立运行 50 次，记录如下数据：中的最好解 (x^*, y^*) 和最差解 $(\bar{x}, \bar{y})$，

上层目标函数值 $F(x^*,y^*)$ 和 $F(\bar{x},\bar{y})$,下层目标函数值 $f(x^*,y^*)$ 和 $f(\bar{x},\bar{y})$,以及最好解 (x^*,y^*) 的约束违反值 $I(x^*,y^*)$。最终记录如下实验数据:

(1) 最优解 (x^*,y^*);

(2) 上层目标函数 $F(x,y)$ 的最优值、最差值、均值、方差;

(3) 下层目标函数 $f(x,y)$ 的最优值、最差值、均值、方差;

(4) 算法 50 次独立运行适应度函数的平均计算次数(MNI)和平均 CPU 运行时间(CPU)。

表 2 记录了 DPDE 算法的最好解数值及与文献[12]和文献[14]的比较,可以看出,DPDE 算法在大多数问题中可以得到与算法 NEA 和 BL-PSO 基本一致的结果,而测试函数 6 的结果和其他两种算法有较大不同。由于小数位的限制,我们也不能比较谁好谁坏。

表 2　最好解的比较

函数	DPDE	NEA[12]	BL-PSO[14]
1	(20,5.0,10.0,5.0)	(20,5,10,5)	(20,5,10,5)
2	(0.0,30.0, −10.0,10.0)	(0,30, −10,10)	(0,30, −10,10)
3	(1.32e −8,2.0, 1.875,0.906)	(4.4e −7,2, 1.875,0.906)	(0,2,1.875, 0.906)
4	(1.033,3.098, 2.597,1.793)	(1.03, 3.097, 2.59,1.79)	(1.031,3.098, 2.597,1.793)
5	(0.279,0.475, 2.344,1.032)	(0.27,0.49, 2.34,1.036)	(0.281,0.475, 2.344,1.033)
6	(61.31,25.77, 2.999,2.999)	(12.47,67.511, 2.999,2.999)	(38.09,60.52, 2.999,2.999)
7	(2.000,0.000, 2.000,0.000)	(2, −2.84e −8, 2,0)	(2,0,2,0)
8	(−0.40,0.800, 2.00,0.00)	(−0.381, 0.809 5,2,0)	(−0.40,0.802, 1.999,0)
9	(0.000,1.000, 0.000)	(1.4e −12,1, 7.07e −13)	(0,1,0)
10	(10.016,0.820)	(10.016 4, 0.819 7)	(1,0)
11	(9.999 96, 9.999 96)	(10.000,10.000)	(9.999 8, 9.999 8)
12	(1.889,0.889, 0.000)	(1.889,0.889, 0)	(2.034 5, 0.884,0)
13	(7.106,7.036, 6.825,7.036)	(7.071,7.071, 7.071,7.071)	(7.070,7.070, 6.928,6.928)
14	(0.00,30.00, −10.00,10.00)	(0,30, −10,10)	(0,30, −10,10)
15	(20.00,5.00, 10.00,5.00)	(20,5,10,5)	(20,5,10,5)

表 3 和表 4 统计了 DPDE 算法的最好解的上层和下层函数值及与文献[12]和文献[14]的比较,对比文献[14]可以看出,DPDE 算法对于函数 3、函数 5、函数 11 和函数 12 的解绝对优于算法 NEA[12],其他函数的解在精度允许范围内也和其相等;对比文献[14]可以看出,DPDE 算法对于函数 5 和函数 8 得到的解绝对优于算法 BL-PSO[14]。对于函数 10,算法 BL-PSO 得到了 $(F,f)=(1,0)$,从数值上看明显优于 DPDE 算法,但是经过验证,当 $x=1$ 时,下层目标函数最小值应该为 $f(x,y)=-101\ 250$。因此,算法 BL-PSO 得到的解 $(x,y)=(1,0)$ 不是全局最优解。剩余函数中,除过函数 12 本文算法稍显劣势外,其他函数都与 BL-PSO 算法得到的函数值相等。

表 3　最好解 $F(x^*,y^*)$ 函数值的比较

函数	DPDE	NEA[12]	BL-PSO[14]
1	225.0	225	225
2	0.0	0	0
3	−18.678 7	−12.65	−18.678 7
4	−8.917 2	−8.92	−8.917 2
5	−7.578 5	−7.575	−7.577 4
6	−11.998 5	−11.999	−11.998 5
7	−3.60	−3.6	−3.6
8	−3.920	−3.918	−3.919 4
9	1 000.0	1 000	1 000
10	81.327 9	81.327 9	1
11	99.999 6	100.014	99.996
12	−1.209 9	−1.209 1	−1.231 2
13	1.980 1	1.98	1.98
14	0	0	0
15	2.2e −16	2.2e −16	0

表 4　最好解 $f(x^*,y^*)$ 函数值的比较

函数	DPDE	NEA[12]	BL-PSO[14]
	$f(x^*,y^*)$	$f(x^*,y^*)$	$f(x^*,y^*)$
1	100	100	100
2	100.0	100	100
3	−1.015 6	−1.021	−1.015 6
4	−6.137 0	−6.14	NA
5	−0.571 9	−0.579 2	−0.577 7
6	−250.288 8	−163.42	−219.261 8
7	−2.0	−2	−2
8	−2.0	−1.956	−2.010 9

续表 4

函数	DPDE	NEA[12]	BL-PSO[14]
	$f(x^*,y^*)$	$f(x^*,y^*)$	$f(x^*,y^*)$
9	1	1	1
10	-0.335 9	-0.335 9	0
11	1.565 9e-08	4.93e-7	0
12	7.617 3	7.614 5	7.781 8
13	-1.980 1	-1.98	-1.98
14	100.0	100	100
15	7.616 7	7.62	7.616 7

特别说明的是,函数 6 的上层函数中 x 的系数为 0,即上层目标函数只是关于变量 y 的函数,在要求下层目标最小的情况下,DPDE 算法计算出 $x=(61.306, 25.768)$,比较下层函数值可以看出解$(x,y)=(61.306, 25.768, 2.999, 2.999)$明显优于算法 NEA[12] 和 BL-PSO[14] 的解。

表 5 和表 6 分别显示了 DPDE 算法的计算结果在上层和下层函数的最好值、最差值、平均值和标准差,函数 6 的均值和方差出现较大变化,具体原因已解释过。其他函数的均值都非常接近最优值,方差都不超过 10^{-6}数量级,最小的方差值如函数 3 和函数 14 的 0,表明算法 DPDE 具有极强的抗变换性。

表 5 上层函数的最好值、最差值、平均值、标准差

函数	$F(x,y)$			
	best	worst	mean	var
1	225.0	225.0	225.0	8.9e-9
2	1.7e-13	4.0e-7	9.3e-8	2.1e-14
3	-18.678 7	-18.678 7	-18.876 7	0
4	-8.917 2	-8.917 2	-8.917 2	6.5e-13
5	-7.578 5	-7.578 5	-7.578 5	3.8e-9
6	-11.998 5	-11.958 2	-11.997 1	0.007 3
7	-3.60	-3.60	-3.60	3.6e-17
8	-3.920	-3.920	-3.920	1.7e-8
9	1 000	1 000.0	1 000.0	4.6e-13
10	81.327 9	81.327 9	81.327 9	1.4e-7
11	99.999 6	99.999 6	99.999 6	1.2e-10
12	-1.209 9	-1.209 9	-1.209 9	2.3e-16
13	1.980 1	1.980 0	1.980 1	4.4e-8
14	0	0	0	0
15	2.2e-16	2.2e-16	2.2e-16	3.4e-29

表 6 下层函数的最好值、最差值、平均值、标准差

函数	$f(x,y)$			
	best	worst	mean	var
1	100.0	100.0	100.0	1.7e-9
2	100.0	100.0	100.0	1.9e-14
3	-1.015 6	-1.015 6	-1.015 6	3.0e-31
4	-6.137 0	-6.137 0	-6.137 0	3.4e-7
5	-0.571 9	-0.571 9	-0.571 9	7.8e-7
6	-263.1	-221.05	-194.5	68.3
7	-2.0	-2.0	-2.0	4.6e-16
8	-2.0	-2.0	-2.0	3.1e-7
9	1	1	1	7.4e-17
10	-0.335 9	-0.335 9	-0.335 9	2.9e-8
11	1.6e-8	1.6e-8	1.6e-8	2.1e-12
12	7.617 3	7.617 3	7.617 3	5.5e-15
13	-1.980 1	-1.980 0	-1.980 1	2.5e-8
14	100.0	100.0	100.0	5.4e-14
15	7.616 7	7.616 7	7.616 7	2.6e-15

表 7 显示了算法的 CPU 平均运行时间和适应度函数的平均计算次数 MNI。从各个函数的平均 CPU 时间上看,本文算法 DPDE 大多数超过了 NEA 算法,但每个问题的平均 CPU 耗时相差不多,而算法 NEA 则具有较强的波动,15 个函数的 CPU 时间的平均值显示 DPDE 算法优于 NEA 算法,表明本文算法具有较强的稳定性,同样 MNI 值也低于 NEA 算法。

表 7 CPU 时间和计算次数

函数	CPU 时间/s		MNI	
	本文算法	NEA[12]	本文算法	NEA[12]
1	16.5	13.3	57 182	85 499
2	13.1	37.3	32 043	256 227
3	20.5	14.4	64 784	92 526
4	12.3	11.8	31 049	71 273
5	16.4	11.7	54 793	77 292
6	18.8	13.2	60 124	92 492
7	20.2	11.8	66 793	77 302
8	11.5	11.9	32 906	71 302
9	15.9	5.6	49 302	4 276
10	14.8	4.2	46 823	28 468
11	11.1	5.9	30 021	4 339
12	14.2	25.3	43 213	163 701

续表 7

函数	CPU 时间/s		MNI	
	本文算法	NEA[12]	本文算法	NEA[12]
13	16.6	177.7	56 097	1 074 742
14	17.4	15.8	60 329	106 760
15	14.9	27.1	56 743	170 818
平均	15.61	25.79	49 481	158 467

4 结　语

本文首先将双层规划问题等价转化为双目标规划问题,采用了非负的最小二乘曲线拟合,利用拟合结果判断候选街的可行性,基于动态概率的 Pareto 支配选择策略挑选下一代个体,解决了 NBLP 问题容易陷入局部最优的缺陷,提高了算法的搜索性能,改善全局寻优能力。15 个标准测试函数的实验数据表明,DPDE 算法在求解 NBLP 问题具有较好的全局寻优能力,较低的计算复杂度,较强的稳定性和适用性,可以获得全局最优解。双目标和双层规划问题深层结合,自适应的 Pareto 支配选择策略是下一步探索的问题。

参 考 文 献

[1] Vicente L, Savard G, Júdice J. Decent approaches for quadratic bilevel programming[J]. Journal of Optimization Theory and Applications, 1994, 81: 379 – 399.

[2] Colson B, Marcotte P, Savard G. An overview of bilevel optimization[J]. Annals of Operations Research,2007,(153): 235 – 256.

[3] 洪云飞,郑跃,陈忠. 弱线性双层规划问题的罚分解方法[J]. 西南师范大学学报,2017(9):11 – 15.

[4] Hansen P, Jaumard B, Savard G. New branch-and-bound rules for linear bilevel programming[J]. SIAM Journal on Scientific and Statistical Computing, 1992, 13(5):1194 – 1217.

[5] Vicente L, Savard G, Judice J. The discrete linear bilevel programming problem[J]. Journal of Optimization Theory and Applications, 1996, 3(89): 597 – 614.

[6] Ren A H, Wang Y P. A novel penalty function method for semivectorial bilevel programming problem[J]. Applied Mathematical Modelling, 2016, 40: 135 – 149.

[7] Colson B, Marcotte P, Savard G. A trust-region method for nonlinear programming algorithm and computational experience[J]. Computational Optimization and Applications, 2005, 3(30): 211 – 227.

[8] Mathieu R, Pittard L, Anandalingam G. Genetic algorithm based approach to bi-level linear programming[J]. Rairo-Operations Research, 1994, 28: 1 – 21.

[9] Sinha A, Malo P, Deb K. Evolutionary algorithm for bilevel optimization using approximations of the lower level optimal solution mapping[J]. European Journal of Operational Research, 2017, 257:395 – 411.

[10] 何胜学,高蕾. 停车需求分布的双层规划模型及算法[J]. 交通运输系统工程与信息,2019(1):83 – 88.

[11] 刘丹,蒲自源,许晓晴,等. 基于机器学习 – 优化混合算法的离散交通网络双层规划模型[J]. 系统工程,2018(8): 114 – 122.

[12] Wang Y P, Jiao Y C, Li H. An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme[J]. IEEE Transaction on Systems, Man and Cybernetics-Part C: Applications and Reviews, 2005, 32(2): 221 – 232.

[13] Wang Y P, Li H, Dang C Y. A new evolutionary algorithm for a class of nonlinear bilevel programming problems and its global convergence[J]. Informs Journal on Computing, 2011, 23(4): 618 – 629.

[14] Han J L, Zhang G Q, Hu Y G, et al. A solution to bi/tri-level programming problems using particle swarm optimization[J]. Information Sciences,2016,370: 519 – 537.

[15] Kuo R J, Lee Y H, Zulvia F E, et al. Solving bi-level linear programming problem through hybrid of immune genetic algorithm and particle swarm optimization algorithm[J]. Applied Mathematics and Computation, 2015, 266: 1013 – 1026.

[16] Ma W M, Wang M M, Zhu X X. Hybrid particle swarm optimization and differential evolution algorithm for bi-level programming problem and its application to pricing and lot-sizing decisions[J]. Journal of Intelligent Manufacturing, 2015, 3(26): 471 – 483.

[17] Li M Q, Lin D, Wang S Y. Solving a type of biobjective bilevel programming problem using NSGA-II[J]. Computers & Mathematics with Applications, 2010, 2: 706 – 715.

[18] Bard J F. Practical bilevel optimization: algorithms and applications[M]. Kluwer Academic Publishers, Netherlands. 1998: 193 – 386.

[19] Amouzegar M A. A global optimization method for nonlinear bilevel programming problems[J]. IEEE Transaction on Systems, Man and Cybernetics-Part B: Cybernetics, 1999, 29(6): 771 – 777.

[20] Ben-Ayed O, Blair C. Computational difficulties of bilevel linear programming[J]. Operational Research. 1990,38(3): 556 – 560.

[21] Bazarar M S, Sherali H D, Shetty C M. Nonlinear programming: Theory and algorithms[M]. 2nd ed. John Wiley and Sons, New York. 1993.

[22] Price K V. Differential evolution: A fast and simple numerical optimizer[C]//IEEE International Conference on Evolutionary Computation, 1997: 153 – 157.