

一个基于插值的解非线性双层规划的遗传算法

李和成^{1),2)} 王宇平¹⁾

¹⁾(西安电子科技大学计算机学院 西安 710071)

²⁾(西安电子科技大学数学科学系 西安 710071)

摘 要 非线性双层规划问题是一类递阶优化问题,相关的算法往往需要对每一个上层变量值求一个下层优化问题才能得到一个可行点,这使得算法的计算量很大.目前文献中的算法通常都是基于对每个确定的上层变量,下层最优解唯一的条件,这就意味着每个下层变量的分量都可以看成是上层变量的函数.基于这个思想,同时为了避免频繁计算下层优化问题,文中提出了一种新的方法.这种方法与已有方法的主要不同之处在于,它不需频繁求解下层规划,而是用插值函数近似下层最优解函数.其主要思想如下:首先,取一些上层变量值作为插值节点,计算它们对应的下层问题的最优解,这些最优解的第 i 个分量作为第 i 个插值函数的函数值,利用这些节点和函数值计算插值函数;其次,将插值函数代入上层问题,得到一个近似原问题的单层规划;最后用一个新的遗传算法求解该单层规划.由于插值节点和相应的插值函数在进化过程中自适应修正和更新,这样可使得该单层规划问题的最优解逐步逼近原问题的最优解,并且可减少计算量.对25个测试问题的仿真结果表明,该文所提出的算法能以较少的计算量找到这些问题的最好解.

关键词 非线性双层规划;插值函数;遗传算法;最优解

中图法分类号 TP18

An Interpolation Based Genetic Algorithm for Solving Nonlinear Bilevel Programming Problems

LI He-Cheng^{1),2)} WANG Yu-Ping¹⁾

¹⁾(School of Computer Science and Technology, Xidian University, Xi'an 710071)

²⁾(Department of Mathematics Science, Xidian University, Xi'an 710071)

Abstract Nonlinear bilevel programming problems are hierarchical optimization problems. For each fixed value of leader's variables the existing algorithms are required to solve the follower's optimization problem to obtain a feasible point for the whole problem, which results in a large amount of computation. Note that in the existing research works, the condition that the follower's optimal solution is unique for each leader's variable value is usually adopted. This condition means that each follower's variable can be seen as a function of the leader's variables although this function is unknown. Based on this observation and to avoid solving the follower's problem frequently, a different skill from that of the existing works is used to tackle this difficulty, i. e., the interpolation functions are adopted to approximate these unknown functions. First, the values of the interpolation function are gotten by solving the follower's problem for some given leader's variable values (i. e., the interpolation points), and the interpolation polynomials (functions) are calculated by using these interpolation points. Then, the follower's variables can be replaced by the corresponding interpolation polynomials in the leader's problem. As a result, the original nonlinear bilevel programming can be approximated by a single-level programming.

Finally, a specifically designed genetic algorithm is proposed for the single-level programming, and the interpolation points and the corresponding interpolation functions are adaptively modified and updated during the evolution so that the optimal solutions of the single-level programming can well approach to those of original nonlinear bilevel programming. Moreover, the computation amount will be decreased. The simulations on 25 test problems indicate the proposed algorithm can find the best solutions with a relatively small amount of computation for these test problems.

Keywords nonlinear bilevel programming; interpolation polynomials; genetic algorithm; optimal solutions

1 引言

双层规划问题(Bilevel Programming Problems, BLPP)是一种具有双层递阶结构的系统优化问题,它包含一个上层规划问题(Leader's programming problem)和一个或多个下层规划问题(Follower's programming problem(s)).上层问题和下层问题都有各自的目标函数和约束条件.上层问题的目标函数和约束条件不仅与上层决策变量有关,而且还依赖于下层问题的最优解.下层问题的最优解则受上层决策变量的影响.这类问题最初在 1952 年由 von Stackelberg 提出^[1],其数学模型为

$$\begin{aligned} \min_{x \in X} \quad & F(x, y) \\ \text{s. t.} \quad & G(x, y) \leq 0; \text{ 其中 } y \text{ 求解} \\ \min_{y \in Y} \quad & f(x, y) \\ \text{s. t.} \quad & g(x, y) \leq 0; \end{aligned} \quad (1)$$

其中, $F(f): R^n \times R^m \rightarrow R$, $G(g): R^n \times R^m \rightarrow R^p (R^q)$; X, Y 为其它约束,如界约束、整数约束等.

这一模型的决策机制是上层决策者首先宣布他(她)的决策 x , 这一决策将影响下层决策问题的约束集和目标函数,然后下层决策者在这一前提下选择使自己的目标函数达到最优的决策 y . 这一过程反过来也影响上层决策问题的约束域和目标函数;进一步,上层决策者再调整他(她)的决策 x , 直到他(她)的目标函数达到最优为止. 这类问题广泛出现于管理、经济规划、优化控制和运输问题等工程实践领域^[2-3]. 一般来讲,问题(1)是一个非凸优化问题.

双层规划问题按所涉及函数是否都为线性函数,可分为线性和非线性双层规划问题.线性双层规划是双层规划问题中最简单的一种,其全局最优解能在约束域的极点上达到^[3]. 尽管如此,线性双层规划问题依然是强-NP 难问题^[3]. 基于线性双层规划问题的特点,已有一些有效的算法,如 k -best 算法、

K-K-T 方法、分支定界法、罚函数法等^[2-3]. 对于非线性双层规划,获得其全局最优解极为困难. 目前存在的一些算法可概括为分支定界法、罚函数法、割平面法等^[2-7], 这些基于梯度的传统优化方法在上下层函数满足相应的可微和凸性要求时求解效率较高,但当函数不可微或结构较复杂(如多峰)时,往往得不到全局最优解. 进化算法和其它一些智能优化算法(如禁忌搜索算法),由于其较强的全局搜索能力和对函数要求较低的优点,已被用于求解非线性双层规划问题^[8-12]. 但与基于梯度的算法(如最速下降法等)相比,一方面对一些问题的计算结果并不理想^[10]. 另一方面,需要对算法产生的每个上层变量值计算下层问题^[8,11-12],这导致了很大的计算量. 文献[9]根据下层为凸规划的特点,利用 K-K-T 条件将问题转化为一个等价的单层规划. 这使得问题的结构简化,但由于 Lagrange 乘子的引入,使得搜索空间的维数增多,因而计算量依然很大. 那么,有没有既简单又计算量不大的方法能使得问题简化呢? 本文试图尝试一种新的途径来解决这个问题.

显然,双层规划问题的难点和计算量主要集中在对每一个上层变量,都需求解一个下层优化问题,从而获得双层规划的一个可行点. 为了不增加搜索空间的维数和减少计算量,若将这一过程用一个简化的技术近似而不影响全局最优解的计算,则双层规划问题的计算难点即可得到解决. 注意到若对上层变量的每一个确定的值,下层变量都有唯一确定的最优解,则下层变量最优解的每一个分量都可看成是上层变量的多元函数,我们称其为下层最优解函数. 若能求出这些最优解函数,并将他们代入上层规划而省去求解下层规划,则双层规划便可化为单层规划问题,从而使双层规划问题的求解变得相对比较容易. 然而遗憾的是,这些最优解函数是隐函数,几乎不可能求出他们的解析表达式. 因此,直接将双层规划以这样的方式转化为单层规划问题几乎是不可能的. 但是,随着函数逼近理论的发展,对于

多维函数的逼近方法越来越成熟.若能近似这些最优解函数,则可能极大地降低计算量.本文就是基于这种思想,尝试用插值函数来近似下层最优解函数,从而将双层规划问题转化为单层规划问题,简化问题并降低计算量.然后设计遗传算法进行求解.另外,在遗传算法求解的过程中,对产生的最好点进行修正,并将其作为新的插值节点更新原插值函数.这一过程使得插值函数在最好点附近的逼近程度越来越好.同时为了保证算法全局收敛,在种群中随机选择一个或几个点进行修正并作为新的插值节点.这样做不仅可保证全局收敛性,而且数值实验的结果也表明算法是有效的.

2 基本概念与假设

本文考虑 X, Y 为如下界约束的情形:

$$\begin{aligned} X &= \{(x_1, x_2, \dots, x_n)^T \in R^n \mid l_i \leq x_i \leq u_i, \\ &\quad i=1, 2, 3, \dots, n\}, \\ Y &= \{(y_1, y_2, \dots, y_m)^T \in R^m \mid \bar{l}_j \leq y_j \leq \bar{u}_j, \\ &\quad j=1, 2, 3, \dots, m\}, \end{aligned}$$

其中, $l_i, u_i, \bar{l}_j, \bar{u}_j$ ($i=1, 2, \dots, n; j=1, 2, 3, \dots, m$) 为常数,给出与问题(1)相关的基本概念如下^[3]:

(1) 搜索空间: $\Omega = \{(x, y) \mid x \in X, y \in Y\}$;

(2) 约束域: $S = \{(x, y) \in \Omega \mid G(x, y) \leq 0, g(x, y) \leq 0\}$;

(3) 对固定的 x , 下层可行域为 $S(x) = \{y \in Y \mid g(x, y) \leq 0\}$;

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

(5) 对每个 $x \in S(X)$, 下层合理反应集:

$$M(x) = \{y \in Y \mid y \in \arg \min \{f(x, v), v \in S(x)\}\};$$

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

本文对问题作如下假设:

① S 为非空紧集;

② 对每一个上层变量 $x \in X$, 下层存在唯一的最优解 $y(x)$;

③ 对每个上层变量, 下层问题关于下层变量是凸规划.

值得指出的是, ① 主要是为了保证问题有最优解, 这是目前研究通常要求的条件^[3]; 假设②也是目前研究非线性双层规划问题通常所作的假设(如文献[2-6, 8]等), 这主要是为了使问题变得确定和简单. 当下层问题存在多个最优解时, 可按不同的要求, 利用上层问题选择一个合适的下层最优解^[3]. 从

这个意义上讲, 本文假设下层最优解唯一是合理的. 假设③是限定本文所考虑问题的范围.

3 模型的转化

对于双层规划问题(1), 要获得一个可行点 (x, y) , 必须求解一次以 x 为参数的下层优化问题. 这一过程使得算法的计算复杂性剧增. 对于大规模问题, 则需要求解很多次下层优化问题, 这甚至使得算法无法在有限时间内完成求解, 因此考虑将原问题进行转化. 在问题(1)中, 对每个 $x \in X$, 下层规划为

$$\begin{cases} \min_{y \in Y} f(x, y) \\ \text{s. t. } g(x, y) \leq 0 \end{cases} \quad (2)$$

由假设②, 对任意 $x \in X$, 存在唯一的下层问题最优解 $y(x) = (y_1(x), y_2(x), \dots, y_m(x))^T$, 即每个下层变量 y_j 均可看成上层变量的 n 元函数; 将 $y(x)$ 代入上层函数, 可将问题(1)转换为如下单层规划问题

$$\begin{cases} \min_{x \in X} F(x, y(x)) \\ \text{s. t. } G(x, y(x)) \leq 0 \end{cases} \quad (3)$$

但 $y_j = y_j(x)$ 均为隐函数, 在一般情况下, 无法写出表达式, 因此对问题(3)直接求解是不现实的. 但是, 若能通过某种简单的计算获得下层近似解, 则能解决这个问题. 本文尝试利用插值函数逼近最优解函数, 即构造 $y_j = \varphi_j(x)$ 来近似 $y_j = y_j(x)$, $j=1, 2, \dots, m$; 从而得到问题(1)的一个近似单层规划

$$\begin{cases} \min_{x \in X} F(x, \varphi(x)) \\ \text{s. t. } G(x, \varphi(x)) \leq 0 \end{cases} \quad (4)$$

其中 $\varphi(x) = (\varphi_1(x), \varphi_2(x), \dots, \varphi_m(x))$. 这样对大量在进化过程中产生的上层变量值, 可以利用插值函数值给出下层问题的近似解. 因此在进化过程中, 避免了求解大量的下层优化问题, 从而达到了减少计算量的目的.

4 插值函数

用插值函数近似有如下特点: 局部插值点越密集, 逼近程度越好; 且在插值样本点处, 下层变量值是相应上层变量值对应的下层最优解, 即满足下层优化问题. 考虑到线性插值计算简单且易于更新, 本文采用基于三角划分的分段(片)线性插值函数逼近下层最优解函数^[13].

在 X 上均匀抽取 N' 个点 $x^i = (x_1^i, x_2^i, \dots, x_n^i)$, $i=1, 2, \dots, N'$ (包括顶点), 并进行三角划分; 对每个

相对固定的 \mathbf{x}^i , 计算优化问题(2), 得最优解 $\mathbf{y}^i = (y_1^i, y_2^i, \dots, y_m^i)$; 考虑每个下层分量 y_j , 以 (\mathbf{x}^i, y_j^i) ($i=1, 2, \dots, N'$), 为样本点(\mathbf{x}^i 为插值节点), 计算函数 $y_j = y_j(\mathbf{x})$ 的插值函数 $y_j = \varphi_j(\mathbf{x})$, $j=1, 2, \dots, m$. 分段(片)线性插值函数的特点是计算简单, 在每个分段(片)上的插值函数通过求解一个 $n+1$ 元线性方程组即可确定; 且当产生新的插值节点时, 只需更新原插值函数在新插值节点附近段(片)上的表达式即可, 计算量小. 以二维函数为例: 对 N' 个样本点采用 Delaunay 三角划分, 相关算法的运算量能降到 $O(N' \log N')$ (N' 表示随机样本点的个数); 进一步, 设 \mathbf{x}_i ($i=1, 2, 3$) 为任一划分三角形的三个顶点, 则

对该三角形内的任一点 $\mathbf{x} = \sum_{i=1}^3 \alpha_i \mathbf{x}_i$, 插值函数值

$$\varphi(\mathbf{x}) = \sum_{i=1}^3 \alpha_i y(\mathbf{x}_i), \text{ 其中 } \alpha_i \geq 0, \sum_{i=1}^3 \alpha_i = 1, y(\mathbf{x}_i) \text{ 为 } \mathbf{x}_i$$

对应的下层解函数值. 求解一个三元线性方程组即可得到 α_i , 其计算复杂性为 $O(n^3)$, 其中 $n=3$ 为变量个数.

5 提出的算法

遗传算法是求解复杂优化问题的一种新型有效方法, 具有全局收敛性、鲁棒性. 为了有效求解问题(3), 首先对上层变量进行实数编码, 给出了一种基于上层目标函数和约束函数的适应度函数, 它能有效辨别不同个体的质量好坏; 其次在杂交算子的设计上, 考虑了进化方向, 使进化后代尽可能向全局最优解趋近. 基于以上考虑, 给出基于插值函数的遗传算法 (An Interpolation Based Genetic Algorithm, IBGA).

算法 1. IBGA.

1. 插值函数: 在 X 上均匀抽取 N' 个点 $\mathbf{x}^i = (x_1^i, x_2^i, \dots, x_n^i)$, $i=1, 2, \dots, N'$, 并计算对应的下层最优解 $\mathbf{y}^i = (y_1^i, y_2^i, \dots, y_m^i)$ 和以 \mathbf{x}^i 为节点的插值函数 $\varphi(\mathbf{x})$;

2. 初始种群: 从所有的 N' 个点 $(\mathbf{x}^i, \mathbf{y}^i)$ (即 $(\mathbf{x}^i, \varphi(\mathbf{x}^i))$) 中, 随机选择 N ($N \leq N'$) 个尽量均匀分布的点, 构成规模为 N 的初始种群 $pop(0)$. 计算种群中每个点的适应度值 $R(\mathbf{x}, \varphi(\mathbf{x})) = F(\mathbf{x}, \varphi(\mathbf{x})) + M \max\{G(\mathbf{x}, \varphi(\mathbf{x})), 0\}$ (其中 M 为充分大的正常数); 记 NON 为当前适应度最小的点集. 令 $k=0$.

3. 杂交: 任取 $\mathbf{p}_{best} \in NON$, 令 \mathbf{x}_{best} 为其对应的上层分量, $\mathbf{p} = (\mathbf{x}, \varphi(\mathbf{x}))$ 是一个从 $pop(k)$ 中按杂交概率 p_c 选择的父代个体. 杂交后代为 $\mathbf{o}_p = (\mathbf{o}_x, \varphi(\mathbf{o}_x))$, 其中

$$\mathbf{o}_x = \mathbf{x} + \mu(\mathbf{x}_{best} - \mathbf{x}) \quad (5)$$

$\mu \in [0, \eta]$ 是一个随机数, η 是使得 $\mathbf{x} + \eta(\mathbf{x}_{best} - \mathbf{x})$ 达到 X 边界的常数. 所有杂交后代的集合记为 OL .

4. 变异: 设 $\hat{\mathbf{p}} = (\hat{\mathbf{x}}, \varphi(\hat{\mathbf{x}}))$ 是按变异概率 p_m 从 $pop(k)$ 中选择参加变异的父代个体, 变异后代为 $\mathbf{o}_{\hat{p}} = (\mathbf{o}_{\hat{x}}, \varphi(\mathbf{o}_{\hat{x}}))$, 其中 $\mathbf{o}_{\hat{x}} = \hat{\mathbf{x}} + \mathbf{\Delta}$; $\mathbf{\Delta} \sim N(0, \sigma^2)$. 所有变异后代的集合记为 $O2$.

5. 修正: 从 $OL \cup O2$ 中选择最好的一个个体和任意其它 $t \geq 1$ 个个体, 通过求解问题(2)修正其对应的下层最优解. 将这些点作为新的插值节点更新插值函数 $\varphi(\mathbf{x})$ 和问题(4). 比较这些节点处的适应度值, 更新 NON .

6. 选择: 任选 $\mathbf{p}_{best} \in NON$ 作为下一代种群中的一个点, 并在 $pop(k) \cup OL \cup O2$ 中选择其它 $N-1$ 个最好的个体组成下一代种群 $pop(k+1)$.

7. 判断: 如果终止条件成立, 则停止, 输出最好解集 NON ; 否则令 $k=k+1$, 转步 3.

6 算法性能分析

目前求解双层规划的大部分算法都需要对每个上层变量计算下层的最优解^[8, 11-12], IBGA 的优势在于不需要对每个上层变量都解下层优化问题, 而只需对个别上层变量(较优解)来求解下层问题, 具体作法是用插值函数的值代替了求解下层优化问题. 现在来比较 IBGA 与已有算法的计算量: 设遗传算法的最大代数 g_{max} , 则已有算法需计算下层优化问题的平均次数为 $N(p_c + p_m)g_{max}$, 而 IBGA 仅需要 $(t+1)g_{max}$ 次, 其中 t 通常取很小的正整数, 一般 $t=1 \sim 3$, 本文的数值实验中取 $t=1$. 因为在遗传算法中, 一般有 $N(p_c + p_m) > 3$, 因此当种群规模 N 较大时, 已有算法每代求解下层问题的次数 $N(p_c + p_m)$ 会远远大于 IBGA 求解下层问题的次数 $(t+1)$. 同时当进化代数 g_{max} 较大时, 即使考虑为获得初始插值函数而计算的 N' 个优化问题, IBGA 求解下层问题的总次数也会远远少于已有算法. 因此, 本文方法可节省大量计算量.

值得指出的是, 在 IBGA 的每一代修正个体时, 由插值函数得到的近似解能为下层优化算法提供一个接近最优解的初始点. 这使得优化算法能很快找到相应的下层最优解, 特别是当下层函数关于变量 (\mathbf{x}, \mathbf{y}) 凸时, 可以证明这个近似解是下层可行解. 不失一般性, 以二维为例说明: 令 $(\mathbf{x}, \varphi(\mathbf{x}))$ 是待修正的种群个体, 且 $\mathbf{x} = \sum_{i=1}^3 \alpha_i \mathbf{x}_i$ ($\alpha_i \geq 0, \sum_{i=1}^3 \alpha_i = 1, \mathbf{x}_i$ 是插值节点). 因为界约束 $\mathbf{y} \in Y$ 可以化为凸函数的零上界约束, 因此只须说明 $g(\mathbf{x}, \varphi(\mathbf{x})) \leq 0$ 即可. 因为有

$$g(\mathbf{x}, \varphi(\mathbf{x})) = g\left(\sum_{i=1}^3 \alpha_i \mathbf{x}_i, \sum_{i=1}^3 \alpha_i y(\mathbf{x}_i)\right) = g\left(\sum_{i=1}^3 \alpha_i \mathbf{A}_i\right) \leq \sum_{i=1}^3 \alpha_i g(\mathbf{A}_i) \leq 0,$$

其中, $\mathbf{A}_i = (\mathbf{x}_i, \mathbf{y}(\mathbf{x}_i))$, $\mathbf{y}(\mathbf{x}_i)$ 是 \mathbf{x}_i 对应的下层最优解. 因此 $\boldsymbol{\varphi}(\mathbf{x})$ 是下层问题的可行解. 这使得一些基于初始点的内点算法能很快找到最优解 $\mathbf{y}(\mathbf{x})$, 这也是本文选择线性插值的一个原因.

7 收敛性分析

上面的方法能否保证收敛性呢? 现在来讨论 IBGA 的全局收敛性. 尽管有些遗传算法是全局收敛算法, 但对非线性程度较高的非凸优化问题, 很多遗传算法往往陷入局部最小. 双层规划是一类复杂的非凸优化问题, 因此对 IBGA 的全局收敛性给出一个理论证明是必要的.

定义 1. 设 $\{\xi_i\}$ 是概率空间的一个随机向量序列, 若存在一个随机向量 ξ 使得 $p(\lim_{m \rightarrow \infty} \xi_m = \xi) = 1$ 或对 $\forall \epsilon > 0$, 有 $p(\bigcap_{m=1}^{\infty} \bigcup_{k \geq m} \{\|\xi_k - \xi\| \geq \epsilon\}) = 0$, 则称 $\{\xi_m\}$ 以概率 1 收敛于 ξ .

引理 1 (Borel-Cantelli). 设 $A_1, A_2, \dots, A_m, \dots$ 为概率空间的一个随机事件序列, 记 $p_k = p\{A_k\}$, 若 $\sum_{k=1}^{\infty} p_k < \infty$, 则 $p(\bigcap_{m=1}^{\infty} \bigcup_{k \geq m} A_k) = 0$; 若 $\sum_{k=1}^{\infty} p_k = \infty$, 且 $A_1, A_2, \dots, A_m, \dots$ 相互独立, 则 $p(\bigcap_{m=1}^{\infty} \bigcup_{k \geq m} A_k) = 1$;

进一步假设:

④ $S(X)$ 是一个有界闭集, 且 $F(\mathbf{x}, \mathbf{y})$ 在 Ω 上连续;

⑤ 至少存在一个全局极小点 \mathbf{x}^* , 使得对 $\forall \delta > 0$, 集合 $S(X) \cap \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{x}^*\| < \delta\}$ 的 Lebesgue 测度大于 0;

⑥ 问题(1)中, $f(\mathbf{x}, \mathbf{y})$, $g(\mathbf{x}, \mathbf{y})$ 关于 $\mathbf{y} \in S(\mathbf{x})$ 二次连续可微, 且 $f(\mathbf{x}, \mathbf{y})$ 是 \mathbf{y} 的严格凸函数.

假设⑥保证了下层最优解函数 $\mathbf{y} = \mathbf{y}(\mathbf{x})$ 的连续性^[14]. 由假设④知,

$$S_0 = \{\mathbf{x} \mid \arg \min_{\mathbf{y} \in S(\mathbf{x})} F(\mathbf{x}, \mathbf{y})\} \neq \emptyset.$$

对 $\forall \epsilon > 0$, 令

$$Q_1 = \{\mathbf{x} \in S(X) \mid \|F(\mathbf{x}, \mathbf{y}(\mathbf{x})) - F^*\| < \epsilon\},$$

$$Q_2 = S(X) \setminus Q_1,$$

其中, $F^* = \min\{F(\mathbf{x}, \mathbf{y}) : (\mathbf{x}, \mathbf{y}) \in IR\} = \{F(\mathbf{x}^*, \mathbf{y}(\mathbf{x}^*)); \mathbf{x}^* \in S(X)\}$.

于是, 种群 $p(k)$ 可分为两种状态:

S_1 : 若 $p(k)$ 中至少有一点属于 Q_1 , 则称 $p(k)$ 处于状态 S_1 ;

S_2 : 若 $p(k)$ 中所有点都不属于 Q_1 , 则称 $p(k)$ 处于状态 S_2 .

定理 1. 设 $p_{ij} (i, j=1, 2)$ 表示 $p(k)$ 处于状态 S_i , 而 $p(k+1)$ 处于状态 S_j 的概率, 则在假设下有:

(a) 对任一个处于状态 S_1 的 $p(k)$, 必有 $p_{11} = 1$;

(b) 对任一个处于状态 S_2 的 $p(k)$, 存在一个常数 $c \in (0, 1)$, 使 $p_{22} < c$.

证明. 从算法选择过程知, 若 $p(k) \in S_1$, 则 $p(k+1) \in S_1$, 所以 (a) 成立; 因为 $S_0 \neq \emptyset$, 对满足假设⑤的 $\forall \mathbf{x}^* \in S_0$, 由 F 和 $\mathbf{y}(\mathbf{x})$ 的连续性可知: 对 $\forall \epsilon > 0$, 存在 $\gamma > 0$, 当 $\mathbf{x} \in S(X)$, $\|\mathbf{x} - \mathbf{x}^*\| < \gamma$ 时, 有

$$|F(\mathbf{x}, \mathbf{y}(\mathbf{x})) - F^*| < \epsilon/2 \quad (6)$$

成立.

因此, 当 $\mathbf{x} \in S(X) \cap \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{x}^*\| \leq \gamma\}$ 时, 有 $|F(\mathbf{x}, \mathbf{y}(\mathbf{x})) - F^*| < \epsilon$. 记 $N_\gamma(\mathbf{x}^*) = \{\mathbf{x} \in R^n \mid \|\mathbf{x} - \mathbf{x}^*\| \leq \gamma\}$. 又因为在 IBGA 中, 当 \mathbf{x} 是插值点时, 有 $\boldsymbol{\varphi}(\mathbf{x}) = \mathbf{y}(\mathbf{x})$. 所以要使个体 $\mathbf{x} \in Q_1$, 只需 $\mathbf{x} \in N_\gamma(\mathbf{x}^*) \cap S(X)$ 且为插值点即可.

当 $p(k)$ 处于状态 S_2 时, 令 $\mathbf{x} \in p(k)$ 的变异后代为 $\hat{\mathbf{x}}$, 则 $\hat{\mathbf{x}} = \mathbf{x} + \Delta \mathbf{x}$, 其中, $\Delta \mathbf{x} = (\Delta x_1, \Delta x_2, \dots, \Delta x_n)^T$, $\Delta x_i \sim N(0, \sigma_i^2)$, $i = 1, 2, \dots, n$. 而 Δx_i , $i = 1, 2, \dots, n$, 相互独立. 于是 $\hat{\mathbf{x}} \in S(X) \cap N_\gamma(\mathbf{x}^*)$ 的概率为

$$p\{\hat{\mathbf{x}} \in S(X) \cap N_\gamma(\mathbf{x}^*)\} = \int \cdots \int_{S(X) \cap N_\gamma(\mathbf{x}^*)} \left(\frac{1}{\sqrt{2\pi}\sigma_i} \right)^n \exp\left(-\sum_{i=1}^n \left(\frac{\hat{x}_i - x_i}{2\sigma_i^2} \right)^2 \right) \cdot d\hat{x}_1 d\hat{x}_2 \cdots d\hat{x}_n \quad (7)$$

记 $p_1(\mathbf{x}) = p\{\hat{\mathbf{x}} \in S(X) \cap N_\gamma(\mathbf{x}^*)\}$. 另一方面, 设 $\hat{\mathbf{x}}$ 成为插值点的概率为 p_2 , 从算法的修正过程可知,

$$p_2 = \begin{cases} 1, & \hat{\mathbf{x}} = \arg \min R(\mathbf{x}_i, \boldsymbol{\varphi}(\mathbf{x}_i)) \\ \frac{t}{|O1 \cup O2| - 1}, & \text{其它} \end{cases},$$

其中, $|O1 \cup O2|$ 表示集合 $O1 \cup O2$ 中元素的个数. 因此一个点 \mathbf{x} 能变为 Q_1 中的点的概率为 $p_3(\mathbf{x}) = p_m p_1(\mathbf{x}) p_2$. 注意到 $\frac{t}{2N}$ 是 p_2 的一个下界. 另一方面, $S(X) \cap N_\gamma(\mathbf{x}^*)$ 是有界闭集, 且其 Lebesgue 测度大于 0, 由式(7)知, $0 < p_1(\mathbf{x}) < 1$, 且 $p_1(\mathbf{x})$ 在有界闭集 X 上连续, 故 $\exists \bar{\mathbf{x}} \in X$, 使得

$$p_1(\bar{\mathbf{x}}) = \min\{p_1(\mathbf{x}) \mid \mathbf{x} \in X\} \text{ 且 } 0 < p_1(\bar{\mathbf{x}}) < 1 \quad (8)$$

注意到 p_{21} 表示 $p(k)$ 处于状态 S_2 而 $p(k+1)$ 处于状态 S_1 的概率, 由式(7)及式(8)知, $p_{21} \geq p_3(\mathbf{x}) \geq p_m p_1(\bar{\mathbf{x}}) \frac{t}{2N}$. 记 $c = 1 - p_m p_1(\bar{\mathbf{x}}) \frac{t}{2N}$, 显然, $0 < c < 1$, 由 $p_{21} + p_{22} = 1$ 知, $p_{22} = 1 - p_{21} \leq c$. 于是, 结论 (b)

成立. 证毕.

定理 2. 设 $\{p(k)\}$ 是由算法产生的种群序列, 且 $p(0)$ 中至少有一个属于 S 的点. 用 $(\mathbf{x}^*(k), \mathbf{y}(\mathbf{x}^*(k)))$ 表示 $p(k)$ 中已修正的最好点, 则有

$$p\{\lim_{k \rightarrow \infty} F(\mathbf{x}^*(k), \mathbf{y}(\mathbf{x}^*(k))) = F^*\} = 1.$$

证明. 对 $\forall \varepsilon > 0$, 记 $p_k = p\{ |F(\mathbf{x}^*(k), \mathbf{y}(\mathbf{x}^*(k))) - F^*| \geq \varepsilon \}$, 则

$$p_k = \begin{cases} 0, & \exists l \in \{1, 2, \dots, k\}, \mathbf{x}^*(l) \in Q_1 \\ \bar{p}_k, & \mathbf{x}^*(t) \notin Q_1, t=1, 2, \dots, k \end{cases}.$$

由定理 1 知, $\bar{p}_k = p_{22}^k \leq c^k$,

于是 $\sum_{k=1}^{\infty} p_k \leq \sum_{k=1}^{\infty} c^k = \frac{c}{1-c} < \infty$, 由 Borel-Canteli

引理知,

$$p\{\bigcap_{m=1}^{\infty} \bigcup_{k \geq m} \{|F(\mathbf{x}^*(k), \mathbf{y}(\mathbf{x}^*(k))) - F^*| \geq \varepsilon\}\} = 0,$$

定理成立. 证毕.

8 数值实验

本文选取了 25 个广泛使用的测试问题进行测试(文献[7, 9, 12]). 这 25 个问题除了 T07 和 T25 外, 都是非线性双层规划问题, 且其中 T03~T05 是

含不可微目标函数的非线性双层规划. 对于初始插值节点, 一般建议 $N' = 50 \times n \times \varpi$, 其中 n 为上层变量的维数, $\varpi (=1 \sim 2)$ 为一个修正系数, 当下层函数非线性程度较高时取较大值, 反之, 取接近 1 的值. 为计算方便, 本文统一取 $N' = 100$. 算法的相关参数取值如表 1.

表 1 IBGA 参数表

N', N	p_c	p_m	t	M
100	0.8	0.2	1	10000

IBGA 的停止准则为: 若算法连续运行 20 代不改善适应度值或达到 100 代, 则停止.

对每个测试问题独立运行 30 次, 记录最好解的位置、最好和最差的上层目标函数值; 并计算了 30 次运行的最好目标函数的平均值、中值和均方差、平均 CPU 时间(CPU)、适应度函数计算的平均次数(MNI)及算法停止时的平均代数(gen); 所有数据见表 2 和表 3. 表 2 列出了算法所用的搜索空间 Ω 和目标函数值的比较; 表 3 给出了 CPU 时间、gen、MNI 及最优解的位置.

表 2 目标函数值计算结果比较

序号	搜索空间	IBGA- $F(\mathbf{x}, \mathbf{y})$					文献- $F(\mathbf{x}, \mathbf{y})$
		最好值	最差值	平均值	中值	均方差	最好值
T01[9,1]	$[-50, 50]^2$	225	225	225	225	1.4e-13	225
T02[9,2]	$[0, 50]^2$	0	0	0	0	0	0
T03[9,20]	$[0, 50]^2$	0	0	0	0	0	0
T04[9,21]	$[0, 50]^2$	0	0	0	0	0	0
T05[9,22]	$[0, 50]^2$	0	0	0	0	0	0
T06[9,3]	$[0, 2]^2$	-12.6787	-12.6760	-12.6770	-12.6787	1.2e-6	-12.68
T07[9,4]	$[0, 10]^2$	-29.2	-29.2	-29.2	-29.2	1.1e-8	-29.2
T08[9,5]	$[-10, 20]^2$	-8.9172	-8.9171	-8.9172	-8.9172	9.0e-6	-8.92
T09[9,6]	$[-10, 20]^2$	-7.5781	-7.5775	-7.5779	-7.5780	4.7e-5	-7.58
T10[9,7]	$[-10, 50]^2$	-11.9991	-11.9989	-11.9990	-11.9990	1.8e-9	-11.999
T11[9,8]	$[-10, 20]^2$	-3.6	-3.6	-3.6	-3.6	0	-3.6
T12[9,9]	$[-10, 20]^2$	-3.92	-3.9146	-3.9179	-3.9187	0.001	-3.92
T13[9,10]	%	-6600	-6600	-6600	-6600	0	-6600
T14[9,12]	$[-50, 50]$	81.3279	81.3278	81.3278	81.3278	1.2e-5	81.3279
T15[9,13]	$[0, 15]$	100.0000	100.0000	100.0000	100.0000	2.4e-7	100.0001
T15[12,7]	$[0, 15]$	100.0000	100.0000	100.0000	100.0000	2.4e-7	100.324
T16[9,14]	$[0, 3]$	-1.2098	-1.2097	-1.2097	-1.2098	4.1e-6	-1.2098
T17[9,17]*	$[0, 10]^2$	1.9802	1.98002	1.98008	1.98007	5.3e-5	1.9802
T18[7, Dempe]	$[-50, 50]$	28.25	28.25	28.25	28.25	0	31.25
T19[7, BIPA2]	$[0, 50]$	17	17	17	17	0	17
T19[12,6]	$[0, 50]$	17	17	17	17	0	17.071
T20[7, BIPA3]	$[0, 10]$	2	2	2	2	0	2
T21[7, BIPA4]	$[0, 10]$	88.7871	88.7871	88.7871	88.7871	1.4e-6	88.79
T22[7, BIPA5]	$[0, 10]$	2.7497	2.7500	2.7498	2.7498	1.1e-4	2.75
T23[12,4]*	$[0, 20]$	85.0909	85.0897	85.0905	85.0909	1.1e-4	84.898
T24[12,5]*	$[0, 30]$	11	11	11	11	0	10.990
T25[12,1]*	$[0, 8]$	13	13	13	13	0	12.953

注: 1. %表示 $[0, 10] \times [0, 5] \times [0, 15] \times [0, 20]$; 2. 序号列中 Tu[v,w]表示第 u 个测试函数选自文献 v 的第 w 个算例;

3. 序号带*表示极大化模型.

表 3 最好解的比较

序号	MNI	CPU/s	gen	最好解	
				IBGA	文献算法
T01[9,1]	5251	7.5	48	(20,5,10,5)	(20,5,10,5)
T02[9,2]	2358	3.7	21	(0,0,−10,−10), (0,30,−10,10)	(0,30,−10,10)
T03[9,20]	2307	3.2	21	(0,0,−10,−10)	(0,30,−10,10)
T04[9,21]	2500	3.8	23	(0,0,−10,−10)	(0,30,−10,10)
T05[9,22]	2516	3.0	23	(0,0,−10,−10)	(0,30,−10,10)
T06[9,3]	3150	3.7	30	(0,2,1.8750,0.9062)	(4.4e-7,2,1.875,0.9063)
T07[9,4]	4965	12.0	54	(0,0.9,0,0.6,0.4)	(1.25e-13,0.9,0,0.6,0.4)
T08[9,5]	4343	4.9	41	(1.0312,3.0977, 2.5970,1.7928)	(1.03,3.097,2.59,1.79)
T09[9,6]	3039	4.8	39	(0.2759,0.4839,2.3445,1.0346)	(0.27,0.49,2.34,1.036)
T10[9,7]	2042	2.3	20	(42.1432,35.6496,2.9985,2.9985)	(12.47,67.511,2.999,2.999)
T11[9,8]	5296	6.5	48	(2,0,2,0)	(2,−2.8e-8,2,0)
T12[9,9]	3274	3.5	31	(−0.4115,0.7944,2, 0)	(−0.381,0.8095,2, 0)
T13[9,10]	5768	14.0	53	(7,3,12,18,0,10,30,0)	(7.034,3.122,11.938,17.906, 0.25,9.906,29.844,0)
T14[9,12]	4142	2.9	38	(10.0163, 0.8195)	(10.0164, 0.8197)
T15[9,13]	2372	1.7	25	(10,10)	(10.000,10.000)
T16[9,14]	2138	2.5	22	(1.8888, 0.8888,0)	(1.8888, 0.8889,0)
T17[9,17]*	5267	11.0	51	(7.0665,7.0741,7.0571,7.0641)	(7.0709,7.0713,7.0709,7.0713)
T18[7, Dempe]	2049	3.8	20	(0,0)	NA
T19[7, BIPA2]	2023	2.5	20	(1,0)	NA
T20[7, BIPA3]	1909	3.3	20	(4,0)	NA
T21[7, BIPA4]	3292	6.9	38	(0,0.5773)	NA
T22[7, BIPA5]	2828	13.0	26	(1.9402,0,1.2114)	NA
T23[12,4]*	2682	5.6	24	(17.4545,10.9090)	NA
T24[12,5]*	2110	3.8	20	(16,11)	NA
T25[12,1]*	1833	4.4	20	(5,4,2)	NA

注：1. NA 表示相关文献没有给出对应数值；2. *MNI*, *CPU*, *gen* 分别表示算法终止时适应度函数计算的平均次数、平均 CPU 时间和平均进化代数；3. 其它符号同表 2.

从表 2 可以看出,IBGA 对问题 T15,T18,T19, T21~T25 的计算结果优于文献中提供的值. 这说明文献中的算法没有找到这些问题的全局最优解. 而对于问题 T06,T08,T09, IBGA 的计算结果略差于文献提供的结果,但从最优值的差别和最优解的位置来看,已经非常接近已知最好解了. 对于其它问题,IBGA 找到了与文献一致的最优值. 对含不可微函数的问题 T03~T05, IBGA 找到了与文献一致的最优解,这说明 IBGA 对上层函数不可微的问题,求解也是有效的. 另外,从表 2 可以看出,除问题 T12 外,其它问题的均方差都不超过 10^{-4} 数量级,这说明 IBGA 是鲁棒的.

表 3 给出了在 30 次运算中,IBGA 所需的平均 CPU 时间、适应度函数计算的平均次数(*MNI*)和求得最好解时 IBGA 所需的平均代数 *gen*. 不难看出,IBGA 所需的 CPU 时间、*MNI* 和运算代数是很少的.

为了考察插值函数的拟合情况,本文随机从 T01~T25 中抽取 T14、T15 和 T06 进行测试. 首先

从每个问题的上层变量 x 的搜索空间 X 内均匀抽取 s 个样本点. 计算每个样本点对应的下层最优解和插值函数值(算法终止时的插值函数),并考察在这些样本点上两者的最大误差 $error_{\max}$ 和最优解处的误差 $error_{\text{opt}}$ (用两点间的欧氏距离表示),相关结果见表 4 和图 1(a)~(d). 其中 T06 的下层变量 $y=(y_1,y_2)^T$ 是二维的,在图 1(c),(d)中分别表示了 $y_i=y_i(x_1,x_2)$ 的拟合情况($i=1,2$).

表 4 通过插值函数获得的下层最优解误差

算例	变量维数	样本点数 s	$error_{\max}$	$error_{\text{opt}}$	对应图例
T14	$n=1, m=1$	25	0	0	图 1(a)
T15	$n=1, m=1$	50	0.0014	0	图 1(b)
T06	$n=2, m=2$	400	0.0490	0	图 1(c),(d)

从表 4 和图 1 可以看出,插值函数的整体逼近效果和最优解处的局部近似效果是很好的. 这表明在进化过程中修正部分个体时,插值函数获得的近似最优解可以为下层优化算法提供很好的初始点,这能有效减少迭代次数.

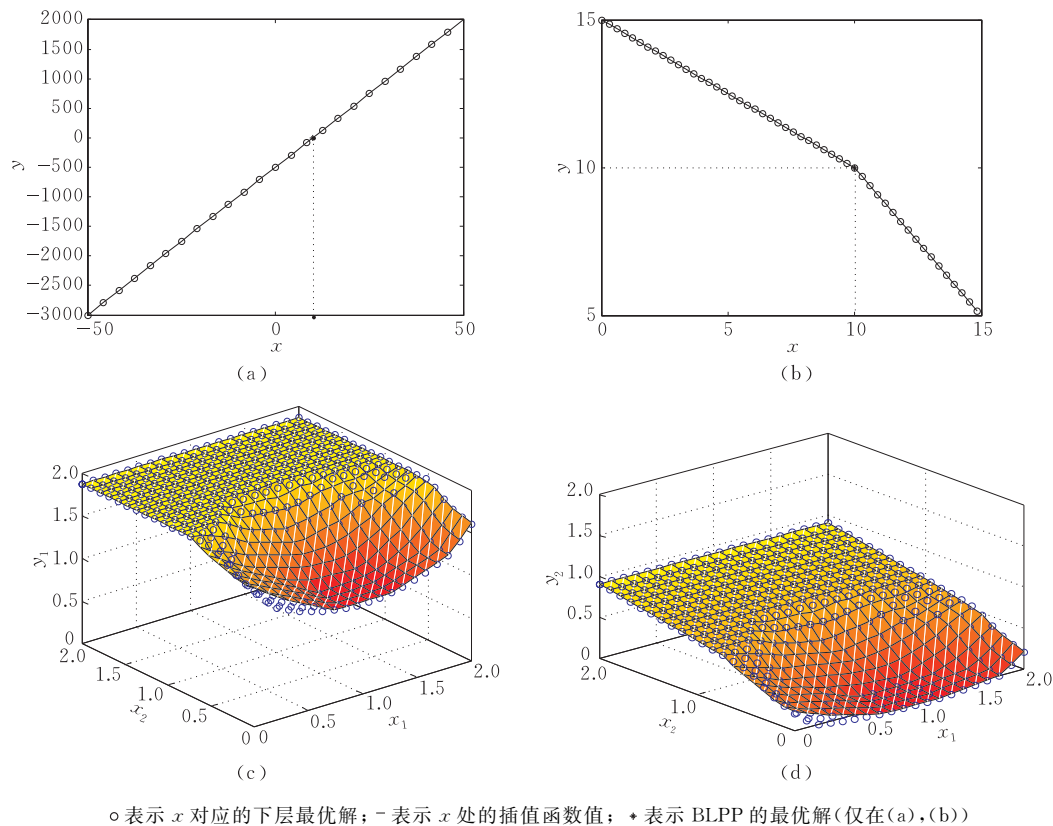


图 1 插值函数拟合情况

9 结束语

本文旨在为一般的非线性双层规划问题提供一种简单易操作的有效算法. IBGA 正是基于这个思想而设计的. 从计算结果可以看出, 算法是稳定有效的. 但对于上层变量为高维的情况, 受插值点数和逼近程度所限, 算法易陷入局部最小. 但对于大多数实际问题, 可以先根据经验估计最优解的大致位置, 使得 IBGA 能在一个小范围内搜索, 这使得算法能快速有效地找到全局最优解. 另外, 假设下层为凸规划, 仅仅是为了有效地获得下层最优解, 对算法本身并无直接影响. 因此对于一些下层易求解的 BLPPs, IBGA 也是有效的. 相信随着函数逼近理论的发展, IBGA 的性能将会进一步得到提高.

参 考 文 献

[1] von Stackelberg H. The Theory of the Market Economy. Oxford, UK: Oxford University Press, 1952

[2] Colson B, Marcotte P, Savard G. Bilevel programming: A survey. A Quarterly Journal of Operations Research(4OR), 2005, 3(2): 87-107

[3] Bard J F. Practical Bilevel Optimization. The Netherlands: Kluwer Academic Publishers, 1998

[4] Muu L D, Quy N V. A global optimization method for solving convex quadratic bilevel programming problems. Journal of Global Optimization, 2003, 26(2): 199-219

[5] Faisca N P, Dua V, Rustem B et al. Parametric global optimization for bilevel programming. Journal of Global Optimization, 2007, 38(4): 609-623

[6] Aiyoshi E, Shimizu K. A solution method for the static constrained Stackelberg problem via penalty method. IEEE Transactions on Automatic Control, 1984, AC-29(12): 1111-1114

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

[8] Li He-Cheng, Wang Yu-Ping. A hybrid genetic algorithm for solving a class of nonlinear bilevel programming problems//Proceedings of Simulated Evolution and Learning — 6th International Conference (SEAL 2006). Hefei, China, 2006: 408-415

[9] Wang Yu-Ping, Jiao Yong-Chang, Li Hong. An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme. IEEE Transactions on Systems, Man, and Cybernetics, Part C, 2005, 35(2): 221-232

- [10] Oduguwa V, Roy R. Bi-level optimization using genetic algorithm//Proceedings of the IEEE International Conference Artificial Intelligence Systems. Divnomorskoe, Russia, 2002: 322-327
- [11] Liu B D. Stackelberg-Nash equilibrium for multilevel programming with multiple followers using genetic algorithms. Computers & Mathematics with Applications, 1998, 36(7): 79-89
- [12] Rajesh J et al. A tabu search based approach for solving a class of bilevel programming problems in chemical engineering. Journal of Heuristics, 2003, 9(4): 307-319
- [13] Wu Zong-Min. Models, Method and Theory of Scattered Data Fitting. Beijing: Science Press, 2007(in Chinese)
(吴宗敏. 散乱数据拟合的模型、方法和理论. 北京: 科学出版社, 2007)
- [14] Edmunds T A, Bard J F. Algorithms for nonlinear bilevel mathematical programs. IEEE Transactions on Systems, Man, and Cybernetics, 1991, 21(1): 83-89



LI He-Cheng, born in 1973, Ph. D. candidate, associate professor. His major research interests include evolutionary computation, and optimization theory and methods.

WANG Yu-Ping, born in 1961, Ph. D., professor. His major research interests include evolutionary computation, data mining, and optimization theory and methods.

Background

The bilevel programming problem (BLPP) can be viewed as a static version of the noncooperative, two-person game introduced by von Stackelberg in the context of unbalanced economic markets. It has a wide variety of applications, such as network design, transport system planning, and management and economics. As an optimization problem with hierarchical structure, bilevel programming problem is intrinsically hard. It is therefore no surprise that most algorithmic research to date has focused on the linear version of the problem. For nonlinear BLPP, most of the existing algorithms can yield a local minimum only. For some specific nonlinear BLPP, for example, all functions in BLPP are twice differential and convex, etc., a few procedures have been proposed to find a global minimum. In recent years, evolutionary algorithm has been developed to solve nonlinear BLPPs with non-differential and nonconvex objective functions, but in order to

obtain feasible points in the populations, for each value of leader's variable, one has to optimize the follower's problem. The process causes a large computational amount. In this paper, authors proposed a new genetic algorithm based on interpolation polynomials, in which the follower's solution functions are given approximately by the interpolation polynomials with respect to leader's variables. It means one can obtain follower's approximately optimal solution only by computing the values of interpolation polynomials at leader's variable value given. As a result, the computational amount is decreased. The experimental results also show that the proposed algorithm can find the best solution with less computation and evolving time.

This work was supported by the National Natural Science Foundations of China (60374063).