文章编号: 1000-0887(2007) 12-1433-08

二层线性规划的自适应遗传算法*

王广民[1], 王先甲[2], 万仲平[3], 贾世会[4]

(1. 中国地质大学 管理学院, 武汉 430074;
2. 武汉大学 系统工程研究所, 武汉 430072;
3. 武汉大学 数学与统计学院, 武汉 430072;
4. 武汉科技大学 理学院, 武汉 430081)

(张石生推荐)

摘要: 提出了一种自适应遗传算法来求解二层线性规划问题. 该方法克服了难以确定合适的交叉概率和变异概率的困难. 另外,在该方法中还采用了其它一些技巧不仅解决了在采用遗传算法经常出现的有些个体不可行的问题,而且还改进了算法的效率.

关 键 词: 二层线性规划; 遗传算法; 适应值; 自适应算子概率; 交叉和变异
中图分类号: O221.5 文献标识码: A

引 言

二层线性规划是具有二层递阶结构的系统优化问题,上层和下层规划问题都有各自的目标函数和约束条件,并且这些目标函数和约束条件都是线性的. 上层规划问题的目标函数和约束条件不仅与上层规划问题的决策变量有关,而且还依赖于下层规划问题的最优解,而下层规划问题的最优解又受上层规划问题的决策变量的影响.

这里考虑如下形式的二层线性规划模型(BLP)[1]:

$$(\mathrm{BLP})\begin{cases}\max\limits_{\boldsymbol{x}} F(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y},\\ \text{其中 } \boldsymbol{y} \text{ 是下面问题的解:}\\ \max\limits_{\boldsymbol{y}} f(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{d}^{\mathrm{T}}\boldsymbol{y}\\ \text{s.t. } \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} \leqslant \boldsymbol{r}, \quad \boldsymbol{x},\boldsymbol{y} \geqslant 0,\end{cases} \tag{1}$$

其中 $F, f: R^{n_1} \times R^{n_2} \to R$ 分别是二层线性规划问题(BLP)的上层和下层规划问题的目标函数. $\boldsymbol{a}, \boldsymbol{c} \in R^{n_1}$, $\boldsymbol{b}, \boldsymbol{d} \in R^{n_2}$, $\boldsymbol{r} \in R^m$. $\boldsymbol{A} \in R^{m\times n_1}$, $\boldsymbol{B} \in R^{m\times n_2}$. $\boldsymbol{x} \in R^{n_1}$, $\boldsymbol{y} \in R^{n_2}$ 分别是(BLP)的上层和下层规划问题控制的决策变量.

二层线性规划问题结构的复杂性给它的求解造成了极大的困难,可以证明求解该问题是NP-难的[2, 3],然而由于其广泛的应用,许多学者对它做了深入的研究. 并且一些作者对二层

* 收稿日期: 2005-11-30; 修订日期: 2007-10-31
基金项目: 国家自然科学基金资助项目(60574071; 70771080)
作者简介: 王广民(1978—),男,河南鲁山人,博士(Tel: +86-27-62351793; E-mail: wgm97@163.com);
王先甲(联系人. Tel: +86-27-68775208; E-mail: wangxj@whu.edu.cn).

规划问题的理论、方法和应用作了综述[1,4-8]. 求解二层线性规划问题的方法很多, 主要可以分为以下几类[9]: 极点搜索法、转化方法、下降法和启发式算法、智能算法及内点法. 然而这些传统的方法一般都依赖于问题的搜索空间, 所以对于求解实际问题就不很方便[10]. 遗传算法是一种启发式算法, 因为其简单, 对函数的可微性没有特别要求, 而且具有全局收敛性、鲁棒性、简单通用、适于并行处理等优点越来越多地被应用于求解二层规划问题. Mathieu 等人首先提出了一种遗传算法来求解二层规划问题[10]. 后来, Yin 也提出了一种基于遗传算法的二层规划算法[11], 在该方法中, 仅仅对上层决策变量进行编码, 并通过求解下层变量来得到每个个体的适应值, 然后通过复制、交叉和变异操作得到最优的个体. Hejazi 等人用下层规划问题的互补条件将二层线性规划问题转化为带有互补约束的单层规划问题. 然后用遗传算法来求解该转化后的问题. 这个方法避免了在用遗传算法求解优化问题中经常遇到的困难——产生的个体可能是不可行的[12]. Oduguwa 等人提出了一种二层规划的遗传算法, 该方法是一种精英优化算法, 鼓励上下两层的决策者进行有限的对称合作, 并且可以在一个框架中求解各种不同种类的二层规划模型[13]. 我们也提出了求解二层线性规划问题的遗传算法[14]. 近来, Calvete 等人[15]把极点搜索法和遗传算法结合起来提出了求解二层线性规划问题的遗传算法. 上述这些算法在求解二层线性规划问题时都展示了良好的性能, 但是在这些算法中交叉概率和变异概率都是随机选取的. 由于交叉和变异是遗传算法中两个主要的操作, 对算法的性能和收敛有很大的影响, 因此选取合适的交叉概率和变异概率是一个非常重要的工作, 但是对于不同种类的却很难确定合适的交叉概率和变异概率. 为了克服这个困难, 本文提出了一种自适应遗传算法来求解二层线性规划.

1 二层线性规划问题的基本概念

任意给定上层规划问题的决策变量 $\boldsymbol{x} \geqslant 0$, $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$ 是常数, 不失一般性, 在求解下层规划问题的时候可以忽略该项, 即可以假设 $\boldsymbol{c} = 0$, 因此下层规划问题可以简化为如下形式:

$$(\mathrm{BLP})\begin{cases}\max\limits_{\boldsymbol{y}} f(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{d}^{\mathrm{T}}\boldsymbol{y}\\ \text{s.t. } \boldsymbol{B}\boldsymbol{y} \leqslant \boldsymbol{r} - \boldsymbol{A}\boldsymbol{x}, \quad \boldsymbol{y} \geqslant 0.\end{cases} \tag{2}$$

下面给出二层线性规划问题的一些基本概念:

i) (BLP) 问题的约束域为

$$S = \{(\boldsymbol{x},\boldsymbol{y}) \mid \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} \leqslant \boldsymbol{r}, \boldsymbol{x}, \boldsymbol{y} \geqslant 0\};$$

ii) 对于任意给定的 $\boldsymbol{x} \geqslant 0$, (BLP) 问题的下层规划问题的约束域为

$$S(\boldsymbol{x}) = \{\boldsymbol{y} \geqslant 0 \mid \boldsymbol{B}\boldsymbol{y} \leqslant \boldsymbol{r} - \boldsymbol{A}\boldsymbol{x}\};$$

iii) S 在上层决策空间的投影为:

$$S(\boldsymbol{X}) = \{\boldsymbol{x} \geqslant 0 \mid \exists \boldsymbol{y} \geqslant 0, \quad \text{使得}(\boldsymbol{x},\boldsymbol{y}) \in S\};$$

iv) 对于任意给定的 $\boldsymbol{x} \in S(\boldsymbol{X})$, (BLP) 问题的下层规划问题的合理反应集为

$$Y(\boldsymbol{x}) = \{\boldsymbol{y} \mid \boldsymbol{y} \in \arg\max\{f(\boldsymbol{x},\boldsymbol{y}), \boldsymbol{y} \in S(\boldsymbol{x})\}\};$$

v) (BLP) 问题的诱导域为

$$IR = \{(\boldsymbol{x},\boldsymbol{y}) \mid (\boldsymbol{x},\boldsymbol{y}) \in S, \boldsymbol{y} \in Y(\boldsymbol{x})\}.$$

为了保证问题(1)有解, 假定 S 和 $S(\boldsymbol{x})$ 是非空的有界紧集. 但是, 对于 $\boldsymbol{x} \in S(\boldsymbol{X})$, 如果问题(2)的解不唯一, 那么即使 IR 非空, 二层线性规划问题(1)也可能不存在最优解[4]. 因

此, 对于任意的 $\boldsymbol{x} \in S(\boldsymbol{X})$, 本文只考虑下层规划问题(2) 有且仅有唯一的解, 记为 $\boldsymbol{y}(\boldsymbol{x})$• 所以, 问题(1) 可以记为:

$$\max\{F(\boldsymbol{x}, \boldsymbol{y}) \mid (\boldsymbol{x}, \boldsymbol{y}) \in IR\}, \tag{3}$$

$F(\boldsymbol{x}, \boldsymbol{y})$ 是点$(\boldsymbol{x}, \boldsymbol{y})$ 的目标函数值• 于是我们可以给出二层线性规划问题的可行解和最优解的定义[1]:

定义 1 如果 $(\boldsymbol{x}, \boldsymbol{y}) \in IR$, 则称点$(\boldsymbol{x}, \boldsymbol{y})$ 为二层线性规划问题(BLP) 的可行解•

定义 2 对于可行点 $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, 如果对于所有的 $\boldsymbol{y}^* \in Y(\boldsymbol{x}^*)$, $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}^* + \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y}^*$ 是唯一的, 并且 $F(\boldsymbol{x}^*, \boldsymbol{y}^*) \geqslant F(\boldsymbol{x}, \boldsymbol{y})$, $\forall(\boldsymbol{x}, \boldsymbol{y}) \in IR$, 则称点$(\boldsymbol{x}^*, \boldsymbol{y}^*)$ 为二层线性规划问题(BLP) 的最优解•

下面我们基于上面的定义来讨论二层线性规划问题的数值解法•

2 二层线性规划问题的自适应遗传算法

2.1 遗传算法简介

遗传算法(简称 GA) 是一种基于生物自然选择和基因遗传学原理的优化搜索方法, 它通过应用选择算子、交叉算子和变异算子来模拟群体遗传, 具有优良的自适应能力和优化能力, 而且还有全局收敛性、鲁棒性、简单通用、适于并行处理等优点, 在求解非凸和非可微的问题也具有良好的鲁棒性, 因此已经被广泛地应用于工程系统、科学计算和商业系统以及在物理、生物、经济和社会中的优化问题•

一般来说, 基本遗传算法由下面的要素构成:

1) 个体编码方法• 编码是应用遗传算法时要解决的首要问题, 也是设计遗传算法时一个关键步骤• 遗传算法一般用二进制编码和浮点数编码• 相对于二进制编码, 浮点数编码更加接近问题空间, 并且数值试验表明浮点数编码收敛速度更快, 而且具有更高的计算精度[16]• 因此, 本文的遗传算法应用浮点数编码•

2) 个体适应值评价• 基本遗传算法按与个体适应值成正比的概率来决定当前群体中每个个体遗传到下一代群体中的机会多少• 为了正确计算这个概率, 这里要求所有个体的适应值必须大于或等于 0• 如果所有个体的在可行域上的目标函数值都大于或等于 0, 那么个体的适应值可以用其目标函数值来表示, 否则用下式来确定目标函数值到个体适应值之间的转换关系[16]:

$$F_{\mathrm{fit}}(\boldsymbol{x}, \boldsymbol{y}(\boldsymbol{x})) = uF(\boldsymbol{x}, \boldsymbol{y}(\boldsymbol{x})) + v, \tag{4}$$

其中 u 和v 是系数•

3) 基本遗传算法的运行参数•

• M: 种群大小, 即群体中所能含个体的数量•

• T: 终止遗传运算的最大进化代数•

• P_c: 交叉概率•

• P_{m}: 变异概率•

4) 遗传算子• 基本遗传算法使用下述 3 种遗传算子: 选择运算、交叉运算和变异算子•

选择运算是用轮盘赌的方式从群体 $P(t)$ 中选择个体来得到群体 $P(t+1)$[16], 具体过程如下:

• 首先计算出群体 $P(t)$ 中所有个体的适应值的和;

• 然后计算群体 $P(t)$ 中每个个体的相对适应值的大小, 它即为各个个体被遗传到下一代群体中的概率;

• 最后再使用轮盘赌(即 0 到 1 之间的随机数) 来确定各个个体被选中的次数•

交叉是产生新个体的主要方法之一, 算术交叉的过程可以描述如下[17]: 假定个体 $(\boldsymbol{x}_1, \boldsymbol{y}(\boldsymbol{x}_1))$ 和 $(\boldsymbol{x}_2, \boldsymbol{y}(\boldsymbol{x}_2))$ 被选中作为进行交叉运算的父个体• 为了保证产生的子个体是可行的, 这里只对上层决策变量作如下形式的算术交叉运算:

$$\boldsymbol{x}_{o1} = \beta\boldsymbol{x}_1 + (1-\beta)\boldsymbol{x}_2,\quad \boldsymbol{x}_{o2} = (1-\beta)\boldsymbol{x}_1 + \beta\boldsymbol{x}_2, \tag{5}$$

其中 $\boldsymbol{x}_1$、$\boldsymbol{x}_2$ 是进行交叉运算的父个体$(\boldsymbol{x}_1, \boldsymbol{y}(\boldsymbol{x}_1))$ 和$(\boldsymbol{x}_2, \boldsymbol{y}(\boldsymbol{x}_2))$ 中的上层决策变量, $\beta \in [0, 1]$ 随机生成, 并且保证 $\boldsymbol{x}_{o1}, \boldsymbol{x}_{o2} \in S(\boldsymbol{X})$• 对于 $\boldsymbol{x}_{o1}, \boldsymbol{x}_{o2}$, 求解下层规划问题(2), 可以得到点 $(\boldsymbol{x}_{o1}, \boldsymbol{y}(\boldsymbol{x}_{o1}))$, $(\boldsymbol{x}_{o2}, \boldsymbol{y}(\boldsymbol{x}_{o2}))$, 根据定义 1 可知它们是(BLP) 问题的可行解•

变异算子的目的是为了改善遗传算法的局部搜索能力并且维持群体的多样性• 非均匀操作的步骤如下[18]: 假定通过变异 $\boldsymbol{x}$ 得到$\boldsymbol{x}_o$, 那么 $\boldsymbol{x}$ 的变异单元$\boldsymbol{x}_o(k)$ $(k \in [1, n_1])$ 是进行如下形式的非均匀变异运算:

$$\boldsymbol{x}_o(k) = \begin{cases} \boldsymbol{x}(k) + \beta(\boldsymbol{x}_{ub}(k) - \boldsymbol{x}(k)), & \text{当 random}(0,1) = 0, \\ \boldsymbol{x}(k) - \beta(\boldsymbol{x}(k) - \boldsymbol{x}_{lb}(k)), & \text{当 random}(0,1) = 1, \end{cases} \tag{6}$$

其中 $\boldsymbol{x}_{ub}$, $\boldsymbol{x}_{lb}$ 分别表示变量 $\boldsymbol{x}$ 的上界和下界, $\beta \in [0, 1]$ 随机生成, 并保证 $\boldsymbol{x}_o \in S(\boldsymbol{X})$• 对于 $\boldsymbol{x}_o$, 求解下层规划问题(2), 于是可以得到点$(\boldsymbol{x}_o, \boldsymbol{y}(\boldsymbol{x}_o))$, 根据定义 1 可知该点是(BLP) 问题的可行解•

2.2 二层规划问题的自适应遗传算法

在遗传算法中, 选择合适的 P_c 和 P_m 是影响算法的性能和收敛的最主要的因素• 如果交叉率 P_c 过大, 那么交叉会破坏群体中好的模式, 这样就会影响进化的速度; 如果 P_c 过小, 那么生成新个体的速度又会较慢• 如果变异率 P_m 过大, 就会有较多的新个体产生, 但是这样会破坏更多的较好个体, 因此遗传算法的性能就类似于随机搜索的效率了; 如果 P_m 过小, 那么生成新个体以及抑制早熟的能力就很差• 所以, 选择合适的 P_c 和 P_m 是非常重要的• 理论上, P_c 一般选择为$[0.4, 0.99]$, P_m 选择为$[0.000\,1, 0.1]$, 但是通过试验来确定合适的 P_c 和 P_m 却是一件非常麻烦的事情, 而且也很难得到一个适合于不同种类问题的 P_c 和 P_m 的值• 因此 Srinivas 等人[19] 提出了自适应遗传算法, 在该算法中, P_c 和 P_m 的值可以根据个体的适应值来自动地变化: 在群体将要陷入局部最优的时候增加 P_c 和 P_m 的值, 当群体在解空间中比较分散时减少 P_c 和 P_m 的值• 即 P_c 和 P_m 按照下面的形式进行选取:

$$P_c = \begin{cases} \dfrac{k_1(F_{fit}^{M} - F'_{fit})}{(F_{fit}^{M} - F_{fit}^{A})}, & F'_{fit} \geqslant F_{fit}^{A}, \\ k_2, & F'_{fit} < F_{fit}^{A}, \end{cases} \tag{7}$$

$$P_m = \begin{cases} \dfrac{k_3(F_{fit}^{M} - F_{fit})}{(F_{fit}^{M} - F_{fit}^{A})}, & F_{fit} \geqslant F_{fit}^{A}, \\ k_4, & F_{fit} < F_{fit}^{A}, \end{cases} \tag{8}$$

其中 F_{fit}^{M} 是群体中最大的个体适应值, F_{fit}^{A} 群体中所有个体的平均适应值, F'_{fit} 是要进行交叉的两个个体中适应值较大的个体的适应值, F_{fit} 是要进行变异的个体的适应值, k_1, k_2, k_3, $k_4 \in [0, 1]$ 则为 $k_1 = 1.0$, $k_2 = 0.5$, $k_3 = 1.0$, $k_4 = 0.5$(参考文献[19])•

所以, 求解二层线性规划问题(BLP) 的自适应算法(SGA) 的算法步骤如下:

步骤 1 初始化• 设置自适应遗传算法的参数 (M, T)•

步骤 2 初始化初始群体• 群体 $P(0)$ 按照下面的步骤进行初始化:

对于随机选择的 $\boldsymbol{x} \in S(\boldsymbol{X})$, 求解下层规划问题(2), 得到最优解为 $\boldsymbol{y}(\boldsymbol{x})$, 那么根据定义 1 可知点 $(\boldsymbol{x}, \boldsymbol{y}(\boldsymbol{x}))$ 是(BLP) 问题的可行解, 并且令该点为初始种群中的个体• 当生成足够多这样个体后, 置迭代代数的计算器为 $t = 0$, 并转下一步•

步骤 3 评价群体中的个体适应值• 根据 2.1 节中的方法, 确定群体中每个个体的适应值• 然后计算群体 $P(t)$ 中所有个体的平均适应值和最大适应值•

步骤 4 保留当前最好个体• 如果当前进化代数 $t = 0$, 那么保留当代种群最好个体为当前最好个体; 否则如果群体 $P(t)$ 中的最大适应值大于当前最好个体的适应值, 那么保留群体 $P(t)$ 中具有最大适应值的个体为当前最好个体, 否则当前最好个体不变化•

步骤 5 计算交叉和变异概率• 分别按照式(7) 和(8) 来得到 P_c 和 P_m 的值•

步骤 6 选择运算• 用轮盘赌的方式从群体 $P(t)$ 中选出个体得到群体 $P(t+1)$•

步骤 7 交叉运算和变异运算• 根据 2.1 节中的方法, 对群体 $P(t+1)$ 进行交叉运算和变异运算•

步骤 8 终止条件• 当迭代代数 $t+1$ 大于或等于最大迭代代数 T 时, 算法终止, 且保留当前最好个体即为由算法得到二层线性规划问题的最优解• 否则, 令 $t = t+1$, 转步骤 3•

3 数值试验

为了验证本文提出的二层线性规划问题的自适应遗传算法的可行性和有效性, 我们应用该方法来求解参考文献中的一些问题•

首先用本文提出的算法来求解由文献[20] 中的例 1 改写而成的例子:

例

$$\max_{x} F(\boldsymbol{x}, \boldsymbol{y}) = -\boldsymbol{x}_1 + 2\boldsymbol{x}_2 + \boldsymbol{x}_3 + 3\boldsymbol{y},$$

其中 $\boldsymbol{y}$ 是下面问题的解

$$\begin{aligned}
&\max_{y} f(\boldsymbol{x}, \boldsymbol{y}) = 2\boldsymbol{x}_1 - \boldsymbol{x}_3 - 4\boldsymbol{y},\\
&\text{s.t. } 0.2\boldsymbol{x}_1 + \boldsymbol{x}_3 + \boldsymbol{y} \leqslant 12,\ -2\boldsymbol{x}_2 + \boldsymbol{y} \leqslant 10,\\
&\quad -3\boldsymbol{x}_1 - \boldsymbol{x}_2 + \boldsymbol{x}_3 \leqslant 12,\ -\boldsymbol{x}_1 + \boldsymbol{y} \leqslant -2,\\
&\quad -2\boldsymbol{x}_1 - \boldsymbol{x}_3 \leqslant -2,\ \boldsymbol{x}_2 \leqslant 15,\ \boldsymbol{y} \geqslant 2,\\
&\quad \boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)^{\mathrm{T}} \geqslant 0,
\end{aligned}$$

在该算法中, 令最大迭代代数 $T = 50$, 种群大小 $M = 30$• 独立运行该算法 10 次, 得到的最好解为 $(\boldsymbol{x}^*, \boldsymbol{y}^*) = (4.0138, 14.9994, 9.1964, 2.0000)$, 上层规划问题的目标函数值为 $F(\boldsymbol{x}^*, \boldsymbol{y}^*) = 41.1814$, 下层规划问题的目标函数值为 $f(\boldsymbol{x}^*, \boldsymbol{y}^*) = -9.1688$, 可以看出它们分别与其精确值 $(\boldsymbol{x}, \boldsymbol{y}) = (4, 15, 9.2, 2)$, $F(\boldsymbol{x}, \boldsymbol{y}) = 41.2$ 和 $f(\boldsymbol{x}, \boldsymbol{y}) = -9.2$ 非常接近• 图 1 给出了在计算过程中每一代群体中最好的适应值以及所有个体的平均适应值•

为了进一步测试算法的效果, 令群体大小为 $M = 30$, 最大迭代代数 $T = 50$, 对于文献中的每个问题, 用本文的算法在微机上独立计算 50 次并记录下面的数据:

ⅰ) 在 50 次计算中所得到的最差解记为 $(\boldsymbol{x}, \boldsymbol{y})$, 在相应于该点的上层和下层规划问题的目标函数值分别记为 $F(\boldsymbol{x}, \boldsymbol{y})$ 和 $f(\boldsymbol{x}, \boldsymbol{y})$;

ⅱ) 在 50 次计算中所得到的最好解记为 $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, 在相应于该点的上层和下层规划问

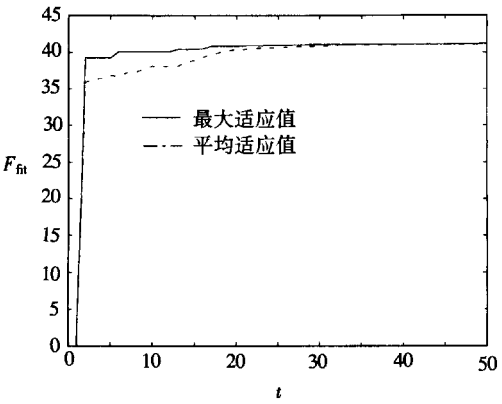

图 1 适应值随代数的变化情况

题的目标函数值分别记为 $F(\boldsymbol{x}^*, \boldsymbol{y}^*)$ 和 $f(\boldsymbol{x}^*, \boldsymbol{y}^*)$;

iii) 对于每个问题 50 次计算的平均 CPU 时间(简记为 t_{CPU})•

另外, 计算

$$\Delta_1 = \left| \frac{(F(\boldsymbol{x}, \boldsymbol{y}) - F(\boldsymbol{x}, \boldsymbol{y}))}{F(\boldsymbol{x}, \boldsymbol{y})} \right| \times 100\%$$

和 $$\Delta_2 = \left| \frac{(F(\boldsymbol{x}^*, \boldsymbol{y}^*) - F(\boldsymbol{x}, \boldsymbol{y}))}{F(\boldsymbol{x}, \boldsymbol{y})} \right| \times 100\%$$

来评价由本文的算法得到的解的优劣, 其中 $F(\boldsymbol{x}, \boldsymbol{y})$ 是文献中二层线性规划问题的上层规划问题的目标函数值• 计算结果分别列于表 1 和表 2 中•

表 1 最好解和最差解与文献中解的比较

例	由本文算法得到的解		文献中的解
	$(\boldsymbol{x}, \boldsymbol{y})$	$(\boldsymbol{x}^*, \boldsymbol{y}^*)$	$(\boldsymbol{x}, \boldsymbol{y})$
1[21]	(15.996 6, 10.993 3)	(15.996 6, 10.993 3)	(16, 11)
2[22]	(17.444 8, 10.889 6)	(17.454 2, 10.908 5)	(17.45, 10.91)
3[23]	(0.996 1, 0, 0.496 2, 1.003 9)	(0.999 3, 0, 0.499 3, 1.000 8)	(1, 0, 0.5, 1)
4[24]	(0.003 5, 0.897 9, 0, 0.596 3, 0.399 1)	(0, 0.899 3, 0, 0.599 5, 0.398 1)	(0, 0.9, 0, 0.6, 0.4)

表 2 相应于最好解与最差解的目标函数值与文献中目标函数值的比较

例	由本文算法得到的解						文献中的解	
	$F(\boldsymbol{x}, \boldsymbol{y})$	$f(\boldsymbol{x}, \boldsymbol{y})$	$\Delta_1/(\%)$	$F(\boldsymbol{x}^*, \boldsymbol{y}^*)$	$f(\boldsymbol{x}^*, \boldsymbol{y}^*)$	$\Delta_2/(\%)$	t_{CPU}/s	$F(\boldsymbol{x}, \boldsymbol{y})$	$f(\boldsymbol{x}, \boldsymbol{y})$
1[21]	48.976 4	− 16.983 3	0.098	48.976 4	− 16.983 3	0.098	63.54	49	− 17
2[22]	84.896 5	− 50.113 6	0.25	85.084 7	− 50.179 7	0.03	71.68	85.11#	− 50.18
3[23]	1.744 2	− 0.015 4	0.33	1.748 8	− 0.002 8	0.067	79.32	1.75	0
4[24]	29.067 0	− 3.193 7	0.46	29.170 9	− 3.180 5	0.10	88.07	29.2	− 3.2

注 有“#”的值应为 85.09[22]•

表中的结果表明, 由本文算法得到的解是全局最优解, 而且无论从与文献中的最优解的比较还是从与文献中的最优值的比较都说明, 由本文算法得到的结果与文献中的结果非常接近•

4 小 结

本文提出了一个自适应遗传算法来求解二层线性规划问题• 该方法根据个体的适应值来自适应地确定交叉概率和变异概率, 从而避免了难于确定合适的交叉概率和变异概率的麻烦• 另外, 根据采用的策略, 初始群体中的个体都是可行的, 而且相应的交叉算子和变异算子都能保证产生的新个体也都是可行的, 这样就大大缩小了搜索空间并且避免了处理不可行个体• 为了进一步提高算法的效率, 在选择和变异过程中用较好的个体来代替父个体• 最后对试验的结果作了分析, 表明本文提出的算法是可行的• 然而, 遗传算法是一种基于自然选择的自适应启发式搜索算法• 因此在我们以后的研究工作中还需要用更多的大规模的数值例子来验证

本文算法的有效性•

[参 考 文 献]

[1] Wen U P, Hsu S T. Linear bilevel programming problem—a review[J]. Journal of the Operational Research Society, 1991, **42**(2): 125-133.

[2] Bard J F. Some properties of the bilevel linear programming[J]. Journal of Optimization Theory and Applications, 1991, **68**(2): 146-164.

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

[4] Ben-Ayed O. Bilevel linear programming[J]. Comuters and Operations Research, 1993, **20**(5): 485-501.

[5] Vicente L N, Calamai P H. Bilevel and multibilevel programming: a bibliography review[J]. Journal of Global Optimization, 1994, **5**(3): 291-306.

[6] Dempe S. Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints[J]. Optimization, 2003, **52**(3): 333-359.

[7] Colson B, Marcotte P, Savard G. An overview of bilevel optimization[J]. Annals of Operations Research, 2007, **153**(1): 235-256.

[8] 王广民, 万仲平, 王先甲. 二(双)层规划综述[J]. 数学进展, 2007, **36**(5): 513-529.

[9] Shih H S, Wen U P, Lee E S, et al. A neural network approach to multiobjective and multilevel programming problems[J]. Computers and Mathematics With Applications, 2004, **48**(1/2): 95-108.

[10] Mathieu R, Pittard L, Anandalingam G. Genetic algorithm based approach to bilevel linear programming[J]. RAIRO-Operations Research, 1994, **28**(1): 1-21.

[11] YIN Ya-feng. Genetic algorithms based approach for bilevel programming models[J]. Journal of Transportation Engineering, 2000, **126**(2): 115-120.

[12] Hejazi S R, Memariani A, Jahanshanloo G, et al. Linear bilevel programming solution by genetic algorithm[J]. Computers & Operations Research, 2002, **29**(13): 1913-1925.

[13] Oduguwa V, Roy R. Bilevel optimisation using genetic algorithm[A]. In: Proceedings of the 2000 IEEE International Conference on Artificial Intelligence Systems[C]. Washington, DC: IEEE Computer Society, 2002, 322-327.

[14] WANG Guang-min, WANG Xian-jia, WAN Zhong-ping, et al. Genetic algorithms for solving linear bilevel programming[A]. In: Hong Shen, Koji Nakano, Eds. The 6th International Conference on Parallel and Distributed Computing, Applications and Technologies[C]. Washington, DC: IEEE Computer Society, 2005, 920-924.

[15] Calvete H I, Gal C, Mateo P M. A new approach for solving linear bilevel problems using genetic algorithms[J]. European Journal of Operational Research, 2007. DOI: 10.1016/j.ejor.2007.03.034.

[16] Goldberg D E. Genetic Algorithms in Search, Optimization and Machine Learning[M]. Massachusetts: Addison Wesley, 1989.

[17] Michalewicz Z. Genetic Algorithms + Data Structures = Evolution Programs[M]. Berlin: Springer-Verlag, 1992.

[18] Michalewicz Z, Janikow C. A modified genetic algorithm for optimal control problems[J]. Computers and Mathematics With Applications, 1992, **23**(12): 83-94.

[19] Srinivas M, Patnaik L M. Adaptive probabilities of crossover and mutation in genetic algorithms[J]. IEEE Transaction on Systems, Man and Cybernetics, 1994, **24**(4): 656-667.

[20] Bard J F. An efficient point algorithm for a linear two-stage optimization problem[J]. Operations

Research, 1983, **31**(4): 670-684.

[21] Anandalingam G, White D J. A solution for the linear static stackelberg problem using penalty functions[J]. IEEE Transaction on Automatic Control, 1990, **35**(10): 1170-1173.

[22] Wen U P, Hsu S T. Efficient solutions for the linear bilevel programming problem[J]. European Journal of Operational Research, 1992, **62**(3): 354-362.

[23] Bard J F, Falk J E. An explicit solution to the multi-level programming problem[J]. Computers & Operations Research, 1982, **9**(1): 77-100.

[24] Candler W, Townsley R. A linear two-level programming problem[J]. Computers and Operations Research, 1982, **9**(1): 59-76.

Adaptive Genetic Algorithm for Solving Bilevel Linear Programming Problem

WANG Guang-min[1], WANG Xian-jia[2], WAN Zhong-ping[3], JIA Shi-hui[4]

(1. School of Management, China University of Geosciences, Wuhan, 430074, P. R. China;
2. Institute of Systems Engineering, Wuhan University, Wuhan 430072, P. R. China;
3. School of Mathematics and Statistics, Wuhan University, Wuhan 430072, P. R. China;
4. School of Science, Wuhan University of Science and Technology, Wuhan 430081, P. R. China)

Abstract: An adaptive genetic algorithm is proposed for solving the bilevel linear programming problem to overcome the difficulty of determining the probabilities of crossover and mutation. In addition, some techniques are adopted not only to deal with the difficulty that most of the chromosomes may be infeasible in solving constrained optimization problem with genetic algorithm but also to improve the efficiency of the algorithm. The performance of this proposed algorithm is illustrated by the examples from references.

Key words: bilevel linear programming; genetic algorithm; fitness value; adaptive operator probabilities; crossover and mutation