双层规划问题的粒子群算法研究①

李相勇，田 澎
（上海交通大学安泰经济与管理学院，上海 200052）

摘要：提出一种求解一般双层规划问题的层次粒子群算法. 和传统的针对特定类型的问题或者基于特定假定假设条件所设计的算法不同，所提出的算法是一个层次算法框架，它通过模拟双层规划的决策过程来直接求解一般双层规划问题. 层次粒子群算法将求解一般双层规划问题转化为通过两个变形粒子群算法的交互迭代来求解上下两层规划问题. 同其它算法的实验结果比较表明层次粒子群算法是一个有效的求解一般双层规划问题的方法.

关键词：粒子群算法；现代启发式算法；双层规划问题；约束优化
中图分类号：O221 **文献标识码**：A **文章编号**：1007－9807(2008)05－0041－12

0 引 言

双层规划问题（bilevel programming problem，BLPP）是具有主从递阶结构的层次优化问题，该问题将一个参数化的优化问题作为其约束条件[1]. 在 BLPP 的层次决策框架中，下层决策者在给定上层决策者参数条件下去优化自己的目标函数，而上层决策者基于下层决策者可能的最优反应，来确定其自己的决策参数以优化其自己的目标函数. 尽管每一个决策者试图在不考虑其它决策者的目标函数的情况下优化其自身的目标函数，但是个人的可能选择是相互依存的，一个决策者的选择将影响其它决策者的目标函数和决策空间. 双层规划模型的主从递阶的决策思想已经应用到很多实际问题中，比如博弈论中的 Stackelberg 博弈就是 BLPP 的雏形，高速公路收费问题[2]，城市交通网络设计问题等. 一个双层规划问题可以表示为[2]

$$\min_{x} \quad F(x,y) \tag{1}$$

$$\text{s.t.} \quad G(x,y) \leqslant 0 \tag{2}$$

$$H(x,y) = 0 \tag{3}$$

对于每一个给定的 x,y 为下式所示优化问题的最优解

$$\min_{y} f(x,y) \tag{4}$$

$$\text{s.t.} \quad g(x,y) \leqslant 0 \tag{5}$$

$$h(x,y) = 0 \tag{6}$$

式中，$x \subseteq R^n$ 和 $y \subseteq R^m$ 分别表示上层和下层规划问题的决策变量. F 和 f: $R^n \times R^m \to R$ 为上层和下层规划问题的目标函数. $G(x,y) \leqslant 0$（$g(x,y) \leqslant 0$），以及 $H(x,y) = 0$（$h(x,y) = 0$）分别为上层和下层规划问题的约束条件. 对于 BLPP 的其它的一些概念，可以参考文献[1].

在构建有效的求解 BLPP 算法方面已经开展了大量的研究工作，目前求解 BLPP 的算法可以分为两类：精确算法和现代启发式算法. 由于其固有的复杂性，大多数精确算法都是针对一些具有良好问题特性的 BLPP，比如求解只有一个下层约束条件，没有上层约束条件的 BLPP 的极值点法[3]、求解线性和二次 BLPP 的分枝定界法[4]、求解没有上层规划约束、下层规划问题为严格凸且约束为线性函数的 BLPP 的信任域法[5] 等. 目前最通用的算法都是基于库恩－塔克条件的，在这些方法中，用库恩－塔克条件代替下层规划问题，将 BLPP 转化为一个带有附加条件的单层规划问题，然后再用其它的算法求解该问题. 但是库

① 收稿日期：2006－04－17；修订日期：2008－07－08.
作者简介：李相勇（1978—），男，江苏建湖人、博士. Email：lixiangyong@163.com

恩 - 塔克条件的推导都必须基于目标函数或者约束函数的可微性或者凸性等条件. 因此这种类型的算法很难运用到一般的应用问题中,特别是带有不可微目标函数或者非凸搜索空间的 BLPP. 与传统的精确算法相比,现代启发式算法(如进化算法、模拟退火算法等)在求解优化问题时,不需要目标函数的可微性、梯度信息和搜索空间的凸性等条件. 现代启发式算法已经显示出其在求解复杂优化问题方面的潜力,文献中已经提出一些求解 BLPP 的现代启发式算法:求解线性 BLPP 的遗传算法[6] 和混合禁忌搜索算法[7]、求解非线性 BLPP 的进化算法[8]、模拟退火算法[9] 等. 虽然现代启发式算法已经被证明是有效的求解 BLPP 的算法,但是大多数已知的算法都是针对特定类型的问题或者基于特定的假设的. 特别是对于具有非可微非凸目标函数和约束函数的一般 BLPP,现代启发式算法还很少.

BLPP 已经被证明为 NP-hard 问题[1],即使最简单的线性 BLPP 也很难求解,更不用说复杂的非线性 BLPP. 本文将粒子群算法(particle swarm optimization, PSO)的应用拓展到求解 BLPP,提出了一个层次粒子群算法. 与传统的精确算法和其它的启发式算法相比,本文提出的算法不是针对特定类型的 BLPP,也不是基于或强或弱的假设条件来求解 BLPP,而是可以用来求解一般 BLPP,它不需要任何关于求解问题的可微和凸条件,也不要对目标函数或者约束条件进行任何形式的转化.

1 求解双层规划问题的层次粒子群算法

本节将详细介绍本文所构建的用来求解一般 BLPP 的层次粒子群算法. 由求解 BLPP 算法的简单综述可知,目前大多数求解算法都是用来求解特定类型的 BLPP(比如线性 BLPP),或者是基于特定的假设条件(比如目标函数可微、决策空间是凸集)等. 这里基于粒子群算法提出了一种求解一般 BLPP 的算法框架,该算法框架集成了两个标准粒子群算法的变形算法分别用来求解上层和下层规划问题,通过两个变形算法之间的交互迭代来直接求解 BLPP. 该算法的最大特点是不需要借助任何假设条件,可以直接来求解一般 BLPP.

1.1 标准粒子群算法

粒子群算法是由美国社会心理学家 Kennedy 和电气工程师 Eberhart 于 1995 年首先提出[10],它是一种新的群体智能优化算法. 自提出以来,粒子群算法以其易实现性以及算法求解的高效性,已经成为一种广泛应用的连续空间优化问题的有效算法. 在粒子群算法中,每个个体动态和随机地调整它们的飞行轨迹,飞向它们自己所经历过的最好位置以及各自邻居中个体所经历过的最好位置. 每个个体的飞行方向是个体的飞行经验以及群体的飞行经验动态交互作用的结果.

在粒子群算法中,每一个粒子的位置向量 $\boldsymbol{x}_i$ 表示求解问题所确定的搜索空间中的 n 维向量,$\boldsymbol{x}_i = (x_{i1}, x_{i2}, \cdots, x_{in})$,其性能通过预设的与求解问题相关的适应度函数来评价. 粒子 i 的速度则定义为粒子位置的改变,$v_i = (v_{i1}, v_{i2}, \cdots, v_{in})$. 每一个粒子的位置是下述几个变量的函数:粒子当前所处的位置和速度,粒子自身所经历过的最好位置,粒子邻居中其它粒子所经历过的最好位置. 粒子自身所经历过的最好位置定义为:$p_i = (p_{i1}, \cdots, p_{in})$,也就是粒子 i 所经历过的具有最小适应度值的位置(以最小化问题为例). 粒子 i 速度和位置的进化方程为[10]

$$v_i(t+1) = v_i(t) + c_1 \cdot rand1 \cdot (p_i - \boldsymbol{x}_i(t)) + c_2 \cdot rand2 \cdot (p_g - \boldsymbol{x}_i(t)) \quad (7)$$

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + v_i(t+1) \quad (8)$$

式中,p_g 表示粒子 i 的邻居中其它粒子所发现的最好的位置(局部版 PSO)或者整个群体中所有粒子发现的最好位置(全局版 PSO). t 是迭代计数器. c_1 和 c_2 为加速因子,它是粒子飞向全局和局部最好位置的速度权重. $rand1$ 和 $rand2$ 为均匀分布于区间[0,1]的随机数. 上述粒子群算法称为基本粒子群算法.

为了提高基本粒子群算法的收敛性能,Shi 和 Eberhart 在基本粒子群算法中,引入了参数惯性权重,在新的算法框架下,粒子 i 速度和位置的进化方程为[11]

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot rand1 \cdot (p_i - \boldsymbol{x}_i(t)) + c_2 \cdot rand2 \cdot (p_g - \boldsymbol{x}_i(t)) \quad (9)$$

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + v_i(t+1) \quad (10)$$

式中,w 称为惯性权重,基本粒子群算法是惯性权重取值为 $w = 1$ 的特例. 惯性权重的作用是控制

粒子以前的速度对当前速度的影响，进而使得粒子群算法在全局探索（exploration）和局部开发（exploration）之间取得平衡[11]，上述粒子群算法在大多数文献中被称为标准粒子群算法，后来其它有关粒子群算法的改进以及混合粒子群算法的研究大多也是基于标准粒子群算法来展开的. 本文提出的算法也是基于标准粒子群算法来构建的，所用的粒子群算法为局部版 PSO.

1.2　求解一般双层规划问题的层次粒子群算法

如前文所述，双层规划问题是一个序列和层次优化问题. 考虑到 BLPP 的序列决策的特点，可以构建一个基于标准粒子群算法的变形算法的层次算法框架来求解 BLPP（以下简称 HPSOBLP）. HPSOBLP 算法通过两个标准粒子群算法的变形算法（变形算法的详细内容见 1.3 节）来模拟 BLPP 的序列决策的过程，从而来求解一般的 BLPP. 本文所提出的算法的主体是一个标准粒子群算法的变型算法（简称 PSO_L），它的作用是在给定下层规划模型最优反应 y 时，求解上层规划问题，获得上层决策者的最佳反应. 别外一个标准粒子群算法的变体（简称 PSO_F）嵌入在层次算法框架中，用来求解给定上层决策变量 x（由算法 PSO_L 求得）条件下的下层规划问题，以获得下层决策者的最佳反应 y. 然后这个下层问题的最优解 y，又传递到上层规划模型作为算法 PSO_L 执行的基础. PSO_L 算法仅仅用来基于 PSO_F 的最优解来优化上层规划问题.

在本文所提出的算法 HPSOBLP 中，将求解双层规划问题转化为在给定上层或下层决策变量的条件下分别求解下层和上层规划问题. 但是整个算法的解信息在两个标准粒子群算法的变形算法之间实现共享，两个变形算法的各自的求解结果互为另外一个算法的输入. 这样就形成了一个由两个变形粒子群算法所组成的序列层次算法框架，图 1 给出了本文所提出的层次粒子群算法的求解一般 BLPP 的策略框架. 由图 1 中算法策略过程可知，HPSOBLP 通过两个变形粒子群算法之间的反复的交互迭代作用和协作过程来模拟和反映 BLPP 的决策过程，从而序列地求解 BLPP.

本文所提出的层次粒子群算法可以用来求解任何类型的双层规划问题，比如具有不可微或非凸目标函数和约束条件的 BLPP、非线性 BLPP 等. 本文第三部分将选择不同类型的测试函数来检验算法 HPSOBLP 的性能. 与其它已知的 BLPP 的求解算法相比，HPSOBLP 算法最大的特点是：它能有效地求解一般 BLPP，不需要借助于关于 BLPP 的特定的假设条件，比如库恩 - 塔克条件、目标函数的梯度信息、约束区域为凸域等，它通过两个变形粒子群算法的交互迭代来直接求解一般 BLPP. 算法 HPSOBLP 的详细过程叙述如下：

1）参数设置

同标准粒子群算法一样，需要首先确定算法 HPSOBLP 的参数：惯性权重 ω，加速因子 c_1 和 c_2，种群规模 $N_{\max}$，最大允许迭代次数 $G_{\max}$，粒子的最大速度 V_{max} 等. 此外需要设置参数 $Worst\_fit$（定义详见 1.3 节）和约束违背度容限 ε（详见 2.2 节）.

2）种群初始化

根据上层规划问题的决策变量的取值范围，为算法 PSO_L 随机初始化一个种群，算法 HPSOBLP 在调用程序 PSO_F 时，根据下层规划问题的决策变量的取值范围则作同样的初始化过程.

3）主程序

若算法终止条件满足，转到第 4 步，否则算法重复执行过程 a - f

（a）更新参数 $Worst\_fit$ 的值

（b）执行算法 PSO_F

对于每一个给定的 $\boldsymbol{x}_i$，算法 PSO_F 求解下层规划问题，获得对应的下层规划问题的最优解 y_i，该最优解作为 HPSOBLP（其主体是 PSO_L）算法执行的基础.

（c）评价粒子适应度值

根据 1.3 节式（11）来计算每一个粒子的适应度值.

（d）更新式（9）中 p_i 和 p_g 的值

更新粒子 i 发现的最优解 p_i 的值，以及粒子 i 的邻居中其它粒子所发现的最优解 p_g 的值.

（e）根据式（9），更新粒子 i 的空间速度 v_i.

（f）根据式（10），更新粒子 i 的空间位置 $\boldsymbol{x}_i$.

4）算法终止

一般优化算法的终止条件主要有以下几种：（1）给定最大允许迭代次数或最长计算时间；（2）算法达到预设的计算精度；（3）连续一定次数的迭代中最优解没有改进或改进低于给定的水平. 本文选择最大迭代次数作为算法终止条件. 在算

法 HPSOBLP 迭代达到最大允许迭代数时，输出其所发现的最优解.

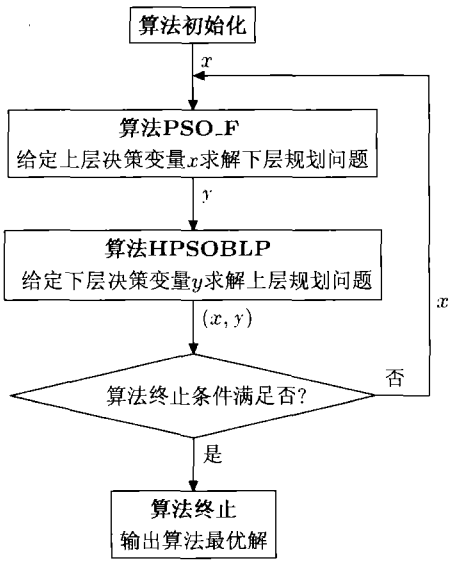


图 1 HPSOBLP 算法框架

Fig. 1 The scheme procedure of HPSOBLP

1.3 求解上层和下层约束优化问题的算法

由式(1)—(6) 可知，在不考虑上下层决策者之间信息交互以及相互依存作用时，上层和下层规划问题均为标准的约束优化问题. 本节提出一个带有新的约束处理机制的改进粒子群算法(即为1.2节的 PSO_L 和 PSO_F 算法) 来求解上层和下层规划问题[12]，为叙述方便下文简称该方法为 PSOCO.

不失一般性，选择下层规划问题作为单个独立的约束优化问题来描述算法 PSOCO，并且同时假设已经给定了上层规划问题的决策变量 x(对于求解上层规划问题可以作类似的处理) 这里假设下层规划问题的约束集由 q 个不等式约束 ($g_i(x,y) \leqslant 0$, for $i = 1,\cdots,q$) 和 $m-q$ 个等式约束 ($h_i(x,y) = 0$, for $i = q+1,\cdots,m$) 组成. 在本文接下来的部分中称所有满足约束条件的粒子为可行粒子，否则称为非可行粒子. 受文献[13] 所提出的用遗传算法求解约束优化问题思想的启发，本文假设算法 HPSOBLP 中每一个可行解均比任何非可行解好，即非可行解总是比任何可行解具有较大的目标函数值(本文以最小化目标函数为例) 在此假设条件下，所有违背了约束条件(至少一个约束条件) 的粒子将比那些满足所有约束条件的粒子被赋予较大的适应度值. 除了上述假设，本文同时假设，在算法执行过程中，每一代的非可行粒子都比算法迭代到当前代所发现的最差的可行粒子(也就是具有最大适应度值的可行粒子) 要差，也就是说这些非可行粒子具有更大的目标函数值. 本文引入一个控制参数 *Worst_fit* 来动态记录粒子群中最差可行粒子的适应度值的变化过程.

在求解约束优化问题的粒子群算法 PSOCO 中，在给定上层规划问题决策变量 x 的情况下，所有的粒子(包括可行的和非可行的粒子) 通过目标函数 $P(x,y)$ 来评价其适应度

$$P(x,y) = \begin{cases} f(x,y) & \text{如果 } y \in \Omega(x) \\ f(x,y) + r\sum_{i=1}^{m} f_i(x,y) + \varphi(y,t) & \\ & \text{如果 } y \in S\backslash\Omega(x) \end{cases} \quad (11)$$

式中，S 表示搜索空间；$f(x,y)$ 表示下层规划目标函数；$\Omega(x)$ 为下层规划问题的可行集；r 是惩罚因子；$f_i(x,y)$ 表示非可行粒子对第 i 个约束条件的约束违背测度；$\varphi(y,t)$ 表示算法执行过程中对非可行粒子的附加启发式值；$f_i(x,y)$ 和 $\varphi(y,t)$ 分别通过下式来定义

$$f_i(x,y) = \begin{cases} \max\{0, g_i(x,y)\} & \text{如果 } 1 \leqslant i \leqslant q \\ |h_i(x,y)| & \text{如果 } q+1 \leqslant i \leqslant m \end{cases} \quad (12)$$

$$\varphi(y,t) = Worst\_fit(t) - \min_{y \in S\backslash\Omega(x)} \{f(x,y) + r\sum_{i=1}^{m} f_i(x,y)\} \quad (13)$$

$$Worst\_fit(t) = \max\{Worst\_fit(t-1), \max_{y \in \Omega(x)} \{f(x,y)\}\} \quad (14)$$

式中，$\varphi(y,t)$ 为算法第 t 代对非可行粒子的附加启发式值，它的作用是通过给非可行粒子一个附加的惩罚值来保证算法当前代所有非可行粒子的适应度值都大于 *Worst_fit* 的值. 控制变量 *Worst_fit* 记录了算法进化到第 t 代所获得的具有最大适应度值的可行粒子，其值在算法的执行过程中，根据式(14) 动态地更新. 通过使用参数 $Worst\_fit(t)$，算法 PSOCO 可以保证在算法迭代过程中所有可行粒子均优于非可行粒子，从而提高算法的收敛速度. 在粒子群算法的初始化阶段，只需要将 $Worst\_fit(t)$ 的值设置为一个较大的值(比如 1 000 000)，并且在算法迭代过程中动态更新其值. 该算法不需要初始种群中所有粒子均为可

行粒子或者至少包含有一个可行粒子.

为了保证粒子在可行区域内搜索以及提高算法收敛速度，在执行上下两层两个 PSOCO 算法时，要求所有的粒子必须同时满足上下两层规划的约束条件，否则就给这些粒子附加惩罚项. 图2给出了算法 PSOCO 的策略框架.

2　实验和结果

2.1　测试函数

为了检验算法 HPSOBLP 的求解质量，并比较 HPSOBLP 和其它算法的性能，本节选择一个包含13个基准问题的测试函数集来检验算法的求解质量. 这些测试函数包含了不同类型的 BLPP，比如具有线性、二次型、或非线性目标函数的 BLPP. 测试函数的详细数学表达式参见附录.

算法 PSOCO
STEP 1　种群初始化
STEP 2　参数设置
STEP 3　当算法满足终止条件时，输出结果，否则执行下列过程：
　STEP 3.1 更新变量 *Worst_fit* 的值
　STEP 3.2 根据式(11) 评价粒子适应度值
　STEP 3.3 根据式(9) 更新粒子速度向量
　STEP 3.4 根据式(10) 更新粒子位置向量

图2　算法 PSOCO 策略框架

Fig. 2 The scheme procedure of PSOCO

2.2　参数设置

算法 HPSOBLP 和标准的 PSO 一样，其性能受一些参数设置的影响. 为了获得比较好的算法求解效果，这些参数需要进行合理的设置，本节给出具体的参数设置过程. 对于部分参数，通过实验来研究参数选择对算法性能的影响，并以此来确定算法最优的参数选择. 在接下来的参数选择实验中，算法 PSO_L 和 PSO_F 的种群规模分别设置为一个较小的值20和40，惩罚因子设置为1 000，选择 T3，T5，T6，T11 和 T13 五个测试函数作为基准函数来研究在不同的参数选择下算法的性能，并确定最优的参数组合.

鉴于计算机本身的计算精度问题，本文引入约束违背度容限 ε 参数来控制约束条件的精度，其值为一个较小的正常数. 例如，对于约束条件 $g_i(X) \leqslant 0$，当且仅当 $g_i(X) > \varepsilon$ 时，解 X 才被视为不满足约束条件，否则算法认为解 X 满足约束条件. 此外对于等式约束 $h_i(X) = 0$，在实验中，用两个不等式来代替，即 $h_i(X) \leqslant 0$ 和 $h_i(X) \geqslant 0$. 在本文的所有实验中，取约束违背度容限值为 7×10^{-5}.

本文所有实验均在 Pentium IV(2.8GHz，256M 内存) 的台式电脑上执行，并以 Matlab7.0 为实验仿真平台. 下面通过实验研究“最大允许迭代次数”、“加速因子” 和“惯性权重” 三个参数对算法 HPSOBLP 性能的影响，并确定最优参数值.

2.2.1　最大允许迭代次数

由算法 HPSOBLP 序列层次结构的特点可知：随着最大允许迭代次数的增加，算法的计算时间可能增加或者算法所获得的最优解可能进一步改善. 假设 PSO_L 和 PSO_F 的种群规模分别为 N_1 和 N_2，两个算法的最大允许迭代次数分别为 G_1 和 G_2，则整个算法的总的迭代次数为 $N_1 \times N_2 \times G_1 \times G_2$. 本节在给定其它参数情况下设计实验研究最大允许迭代次数对算法性能的影响. 实验中将 G_1 和 G_2 设置为从30到120的10个不同的水平，步长为10，因此共有100个不同的水平组合，

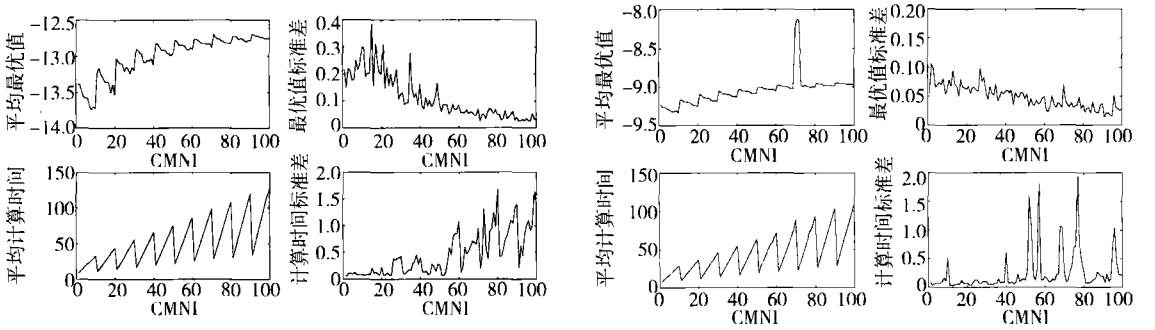
实验中其它参数设置如下：惯性权重线性地从1.2减小到0.1，加速因子 $c_1 = c_2 = 2$，粒子的最大速度 V_{max} 设置为相应的决策变量的取值范围，参数 *Worst_fit* 初始值设置为 100 000. 对于每一个测试函数，实验中算法 HPSOBLP 运行30次.

图3—图4给出了100个不同的 G_1 和 G_2 参数水平组合下，测试函数 T3、T5、T6 和 T11 上层规划问题的最优值以及对应的计算时间的平均值和标准差的变化过程. 图中横坐标“CMNI” 表示 G_1 和 G_2 组合水平，比如“1 ~ 10” 表示的10个组合水平为：$G_1 = 30$，G_2 以步长10从30变化到120. “11 ~ 20” 表示的10个组合水平为：$G_1 = 40$，G_2 以步长10从30变化到120. “99 ~ 100” 表示的10个组合水平为 $G_1 = 120$，G_2 以步长10从30变化到120. 其它值的含义以此类推.

从图中的实验结果可以得出以下一些结论：

(1) 在不同的 G_1 和 G_2 参数组合水平下，对于所有的测试函数 HPSOBLP 都能够发现比较好的解. 所有测试函数30次实验最优解的标准差均比较小，这表明 HPSOBLP 是一个具有较高鲁棒性的算法.

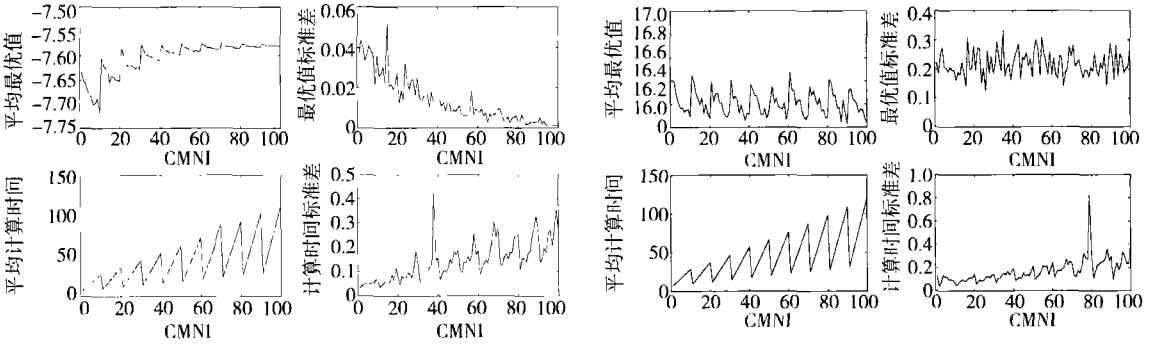
(2) 实验结果表明在不同的最大迭代数的组合水平下，最优解的目标函数值变化较小.



(a) T3　　(b) T5

图 3　测试函数 T3 和 T5 的实验结果

Fig. 3 Computational results of T3 and T5 under different combinations of G_1 and G_2



(c) T6　　(d) T11

图 4　测试函数 T6 和 T11 的实验结果

Fig. 4 Computational results of T6 and T11 under different combinations of G_1 and G_2

(3) 尽管不同的最大迭代次数组合对算法最优值的影响不显著,但是其对算法计算时间有显著性影响. 由图中可知,对于所有的测试函数,在选择较高的 G_1 和 G_2 组合时,算法的计算时间显著性增加. 算法计算时间对于不同的 G_1 和 G_2 组合值比较敏感,实验中发现,一个较大的 G_1 和较小 G_2 组合水平是一个较好的选择,该组合水平下算法能够在求解质量和计算时间之间取得较好的平衡. 因此在本文其它实验中将 G_1 和 G_2 分别设置为 $G_1 = 120$ 和 $G_2 = 30$.

(4) 现代启发式算法的最大优点是算法能够在相对比较中等的计算时间内,找到优化问题的最优解或者近似最优解. 算法 HPSOBLP 在将最大迭代次数设置为较小值时能够获得鲁棒性比较高的最优解. 因此,为了减少计算时间,在实际应用时可以将 G_1 和 G_2 设置为较小的值.

2.2.2　加速因子

在标准 PSO 算法中,粒子的飞行轨迹受由加速因子 c_1 和 c_2 标定的两个随机部分控制,因此合适的选择加速因子对于算法获得比较好的求解质量具有重要影响. 本节通过实验来研究加速因子对 HPSOBLP 算法性能的影响. 实验中约束违背度容限、种群规模、最大迭代次数以及惩罚因子都设置为前面章节确定的最优值. 目前文献对于加速因子的设置有两种策略,即固定加速因子和时变加速因子[14]. 为了更加合理的比较这两种策略对算法性能的影响,本节实验中将加速因子设置为包含上述两种策略的 12 种不同的水平组合(见表 1). 表中"2 - 1" 表示加速因子值从 2 线性减少到 1,其它依此类推.

表 1 给出了不同加速因子组合水平下,算法 HPSOBLP 的 30 次实验结果. 表中给出了对应的上层规划问题目标函数值的平均值和标准差. 从实验结果可以得出如下结论:

(1) 实验结果显示算法 HPSOBLP 的性能对加速因子的取值是非常敏感的. 除了测试函数 T5,对于其它测试函数,比较好的加速因子设置是采用固定参数策略,并且两个加速因子取值都为 2.5,在接下来的实验中将两个加速因子都取值为 2.5.

(2) 对于 T5 以外的测试函数,固定加速因子策略下所有测试函数总的平均值均小于时变加速因子策略下的值(具体见表 1).

(3) 在将算法 HPSOBLP 应用到实际问题时，时变加速因子策略可能不是一个合适的设置. 实验结果表明虽然固定加速因子策略是一个较好的选择，但是具体值的设置却是问题依赖的.

表 1　不同加速因子组合水平下算法 HPSOBLP 实验结果

Table 1 Computational results of HPSOBLP under different acceleration coeffcients

函数	上层目标函数值的平均值和标准差											
	$c_1=0.5$ $c_2=0.5$	$c_1=1$ $c_2=1$	$c_1=1.5$ $c_2=1.5$	$c_1=2$ $c_2=2$	$c_1=2.5$ $c_2=2.5$	$c_1=2-0$ $c_2=0-2$	$c_1=2-0.5$ $c_2=0.5-2$	$c_1=2-1$ $c_2=1-2$	$c_1=2.5-0$ $c_2=0-2.5$	$c_1=2.5-0.5$ $c_2=0.5-2.5$	$c_1=2.5-1$ $c_2=1-2.5$	$c_1=2.5-1.5$ $c_2=1.5-2.5$
T3	-13.791 3 0.393 9	-13.303 3 0.157 0	-13.366 0 0.155 4	-13.694 6 0.222 5	-14.066 2 0.288 3	-13.085 6 0.066 1	-13.178 4 0.136 4	-13.311 4 0.138 3	-13.143 9 0.134 6	-13.248 6 0.110 2	-13.437 2 0.154 2	-13.710 4 0.260 6
T5	-9.203 3 0.055 4	-9.141 7 0.048 1	-9.153 8 0.050 9	-9.189 1 0.054 8	-9.146 8 0.048 9	-9.141 2 0.036 9	-9.260 4 0.109 5	-9.439 7 0.104 9	-9.194 7 0.189 8	-9.095 0 0.030 3	-9.149 3 0.056 2	-9.127 1 0.041 5
T6	-7.705 6 0.049 1	-7.641 4 0.023 4	-7.639 3 0.017 3	-7.702 9 0.033 9	-7.781 2 0.060 9	-7.615 1 0.013 5	-7.617 6 0.014 2	-7.635 7 0.019 6	-7.616 6 0.011 0	-7.636 7 0.017 8	-7.661 5 0.036 6	-7.713 4 0.027 3
T11	16.173 6 0.216 6	16.129 9 0.187 4	16.206 7 0.163 7	16.017 8 0.208 7	16.036 5 0.283 3	16.149 0 0.211 8	16.188 3 0.173 0	16.144 4 0.218 2	16.147 5 0.181 5	16.167 2 0.196 3	16.133 9 0.161 2	16.117 0 0.204 4
T13	2.748 7 0.000 5	2.749 5 0.000 1	2.749 0 0.000 5	2.744 4 0.002 3	2.738 7 0.006 1	2.749 6 0.000 2	2.749 4 0.000 2	2.749 0 0.000 3	2.749 1 0.000 4	2.748 5 0.001 2	2.747 2 0.001 4	2.744 6 0.001 6

2.2.3　惯性权重

Eberhart 和 Shi[11] 在实验中发现惯性权重取值于区间[0.9,1.2]，算法在平均意义上具有较好的性能. 同时他们也发现，在实验中将惯性权重线性的减少，算法的性能能够大大改善. 但是他们的实验结论只是基于一个测试函数的实验结果而得出的. 用同样的测试函数，他们在文献[15] 中指出，与使用固定惯性权重策略相比，将惯性权重在算法迭代过程中从 1 线性减少到 0.4 能够取得更好的实验结果[15]. 然而正如 Shi 和 Eberhart 在文献[15] 指出的惯性权重的选择是问题依赖的. 文献中通常通过实验来确定惯性权重的取值. 本节也通过实验来确定最优的惯性权重值. 实验中将惯性权重值设置为 10 个不同的水平(见表 2)，表中“1.2 - 0.1” 表示惯性权重值从 1.2 线性地减少到 0.1，其它依此类推. 表 2 给出了算法 HPSOBLP30 次实验所发现的上层规划问题的目标函数值的平均值和标准差.

表 2　不同惯性权重下的 HPSOBLP 实验结果

Table 2 Computational results of HPSOBLP under different inertia weights

函数	上层目标函数值的平均值和标准差								
	1.2	1	0.8	0.6	0.5	1.2 - 0.1	1 - 0	0.9 - 0.4	0.8 - 0.1	0.6 - 0
T3	-14.248 0 0.266 0	-14.060 0 0.285 0	-13.932 0 0.302 0	-13.809 0 0.218 0	-13.742 0 0.194 0	-14.028 0 0.289 0	-13.861 0 0.256 0	-13.862 0 0.264 0	-13.886 0 0.255 0	-13.852 0 0.350 0
T5	-9.332 9 0.190 0	-9.090 5 0.117 0	-8.974 2 0.091 0	-8.890 8 0.146 0	-8.874 6 0.127 0	-9.022 0 0.145 0	-8.955 9 0.146 0	-8.922 5 0.116 0	-8.884 5 0.116 0	-8.927 1 0.133 0
T6	-7.828 3 0.181 0	-7.717 3 0.111 0	-7.752 5 0.071 0	-7.759 2 0.049 0	-7.758 8 0.056 0	-7.781 2 0.062 8	-7.771 7 0.082 0	-7.754 7 0.036 0	-7.739 6 0.043 8	-7.763 1 0.052 6
T11	16.306 0 0.195 0	16.183 0 0.241 0	16.247 0 0.173 0	16.262 0 0.173 0	16.390 0 0.147 0	16.009 0 0.320 0	16.194 0 0.153 0	16.124 0 0.182 0	16.212 0 0.183 0	16.359 0 0.137 0
T13	2.741 6 0.004 0	2.742 6 0.006 0	2.742 9 0.004 0	2.743 6 0.004 0	2.743 5 0.003 0	2.740 3 0.005 0	2.741 6 0.004 0	2.742 0 0.005 0	2.741 9 0.004 0	2.742 2 0.004 0

实验数据表明：

1）算法 HPSOBLP 的性能对惯性权重是比较敏感的. 因此为了获得比较好的求解质量，如何合理的确定惯性权重的值是 HPSOBLP 执行的一个关键因素.

2）实验结果同时也表明时变惯性权重策略并不是对于所有的测试函数都是适当的选择，比如，对于测试 T3，T5 和 T6，算法就是在固定惯性权重策略下获得了最好的解，并且惯性权重的取值是 1.2. 相反对于测试函数 T11 和 T13，算法却是惯性权重设置为从1.2线性减少到0.1时，取得最好的实验结果.

3）实验结果证实了文献[15]的结论：惯性权重的选择是问题依赖的，综合考虑最优解以及对应的计算时间，本文选择采用时变的惯性权重策略，并且在算法执行中将其值从 1.2 线性减少到0.1.

2.3 HPSOBLP 算法的计算结果

根据 2.2 节确定的最优参数，本节通过实验研究算法 HPSOBLP 对于所有测试函数的求解性能. 对于每一个测试函数，算法执行100次，表3中给出了实验结果，对于每一个测试函数给出如下统计指标值：上层规划问题的目标函数值的最差值（用“Worst_L” 表示），平均值（用“Avg.” 表示），最优值（用“Best_L” 表示），100 次实验的最优值的标准差（用“Std.” 表示）表 3 同时给出了与上层规划问题目标函数最优值以及最差值所对应的下层规划问题的目标函数值，分别用“Best_F” 和“Worst_F” 表示.

表 3 算法 HPSOBLP 实验结果

Table 3 Summary of the results of HPSOBLP

函数	上层目标函数值 $F(x,y)$			下层目标函数值 $f(x,y)$		
	Best_L	Worst_L	Avg.	Std.	Best_F	Worst_F
T1	0	0	0	0	100	100
T2	225	225	225	0	100	100
T3	-14.757 8	-13.453 8	-14.039 1	0.261 1	0.206 7	-0.590 95
T4	-36.000 3	-36	-36.000 1	0.000 1	0.250 0	0.250 0
T5	-9.276 9	-8.699 9	-8.950 5	0.126 8	-4.756 1	-10.015 7
T6	-7.956 5	-7.661 0	-7.783 1	0.066 8	-1.515 2	-0.958 7
T7	-11.998 5	-11.997 8	-11.998 1	0.000 2	-459.224 8	-137.194 3
T8	-3.602 8	-3.598 9	-3.600 7	0.000 7	-1.984 9	-2.005 5
T9	-3.920 1	-3.916 6	-3.919 5	0.000 6	-1.992 9	-2.081 9
T10	88.775 7	88.785 2	88.783 5	0.001 6	-0.769 8	-0.769 8
T11	15.440 0	16.370 0	16.031 8	0.217 3	2.728 0	2.258 1
T12	1.999 7	2.060 8	2.001 6	0.006 5	24.019 0	23.870 2
T13	2.703 9	2.749 7	2.741 5	0.005 0	0.560 2	0.559 7

由表 3 中结果可知，对于大部分测试函数来说，100 次实验最优解的标准差都非常小，接近于0，这表明本文提出的算法是个鲁棒性比较高的算法，对于文中的测试，算法都能找到比较好的解. 表 4 给出了对应的 100 次实验的计算时间的统计信息. 对于每一个测试函数表 4 给出了 CPU time 的平均值（Avg.）、最小值（Min）、最大值（Max）以及标准差（Std.）这里所给出的计算时间是指算法迭代到终止条件时，整个算法的执行时间. 从实验数据可以看出，算法 HPSOBLP 均能在比较短的时间内发现比较好的解，并且计算时间的标准差也比较小. 综上所述，最优解的质量以及相应的计算时间都表明 HPSOBLP 是一个具有较高鲁棒性的求解一般 BLPP 的算法. 这里需要指出的是由于文献中没有合适的较大规模 BLPP 的测试函数，本文从文献中选择的这些基准测试函数只是具有较小决策变量和约束条件，算法 HPSOBLP 的性能需要在将来的研究工作中通过大规模的测试

函数以及实际应用问题进一步来检验.

2.4　与其它算法比较结果

本节比较了算法 HPSOBLP、文献[8]中的遗传算法 GA，文献[2]中的信任域方法（简称为 TRM），以及每一个测试函数所对应的原始文献中的算法（简称为 Original）的性能. 选择上述几个算法作为比较对象是因为本文所选择的测试函数，文献中其它的传统算法没有给出算法结果，此外文献中不同的算法使用了不同的测试函数集，没有一个共同的测试函数集.

表5给出了算法性能的比较结果. 从表5中数据可以看出，算法 HPSOBLP 发现的最优解优于其它几个算法，特别对于测试函数 T3，T4，和 T11，HPSOBLP 发现了较好的解. 需要指出的，本文没有比较不同算法获得各自最优解所需要的计算时间，因为对不同算法的计算时间作一个合理的评价是一件非常困难的事：(1) 不同的作者使用不同的机器来执行各自的程序，而机器的计算速度很大程度上影响算法执行的时间；(2) 算法的程序编写是一件很有技巧性的工作，由于不同的作者各自具有不同的编程能力，使用不同的编程语言、不同的算法结构和数据结构.

表 4　算法 HPSOBLP 的计算时间

Table 4 Summary of computation cost of HPSOBLP

时间	不同函数 CPU time 统计值												
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13
Avg.	26.63	25.98	30.35	38.37	27.13	26.44	27.51	26.42	27.38	26.63	28.75	25.66	27.99
Min	26.55	25.55	30.13	38.17	27.02	26.13	26.88	26.14	27.05	26.55	28.44	25.5	26.91
Max	27.66	26.53	30.69	39.59	27.58	26.8	27.89	26.81	27.72	27.66	29.00	26.14	29.14
Std.	0.110 7	0.186 3	0.102 7	0.170 7	0.079 3	0.124 8	0.097 3	0.127 1	0.134 8	0.110 7	0.124 0	0.084 8	0.490 6

表 5　算法 HPSOBLP 和其它算法性能比较

Table 5 Comparison of the optimums of HPSOBLP with other algorithms

函数	上层目标函数值 $F(x,y)$				下层目标函数值 $f(x,y)$			
	HPSOBLP	GA	TRM	Original	HPSOBLP	GA	TRM	Original
T1	0	0		5	100	100		0
T2	225	225		225	100	100		100
T3	-14.757 8	-12.68	-12.68	-12.68	0.206 7	-1.016	-1.02	-1.016
T4	-36.000 3	-29.2		-29.2	0.250 0	0.314 8		0.314 8
T5	-9.276 9	-8.92		-8.92	-4.756 1	-6.14		-6.05
T6	-7.956 5	-7.58		-7.56	-1.515 2	-0.574		-0.580
T7	-11.998 5	-11.999		-12	-459.225 0	-163.42		-112.71
T8	-3.602 8	-3.6		-3.6	-1.984 9	-2		-2
T9	-3.920 1	-3.92		-3.15	-1.992 9	-2		-16.29
T10	88.775 7		88.79	88.79	-0.769 8		-0.77	-0.77
T11	15.440 0		17	17	2.728 0		2	2
T12	2.000 0		2	2	24.019 0		24.02	24.02
T13	2.703 9		2.75	2.75	0.560 2		0.57	0.57

2.5　结果分析

对于测试函数 T1，除了最优解 $(x,y)=(0,30,-10,10)$，算法 HPSOBLP 还发现了另外一个可行解 $(x,y)=(0,0,-10,-10)$，这个解和最优解具有相同的上层规划问题的目标函数值 100，但是具有一个较大的下层规划目标函数值.

因此尽管解$(x,y)=(0,0,-10,-10)$是一个可行解,但是其不是一个帕累托最优解.

函数 T3,HPSOBLP 发现了一个比较好的解(0, 1.958 112, 2.826 687,1.556 686),其对应的上层规划问题的目标函数值为 - 14.757 8,该值明显优于遗传算法[8] 和信任域方法[2] 所发现的最优解.

对于测试函数 T4,算法发现了一个具有约束违背度容限 $\varepsilon=0$ 的最优解,(0,0,1,1,0,1,0,0),其对应的上层和下层规划问题的最优解分别为 - 36 和 - 0.25. 然而当约束违背度容限 $\varepsilon=7e-5$ 时,算法发现了一个比文献[8,16] 中算法更好的解,这就证明其它算法所发现的最优解不是全局最优解. 本文发现的最优解为(0,0.093 64, 0.618 728,1,1,1,0,0.500 005,0.312 714),其所对应的第二和第三个约束条件的约束违背度分别为:5e - 6 和 6e - 6,最优解所对应的上层规划目标函数值为 - 40.374 559.

测试函数 T5,算法 HPSOBLP 与其它算法相比发现了一个较好的解,上层规划问题的目标函数值为 - 9.276 89,对应的解为(0,2.665 131, 2.595 505, 1.794 999).

测试函数 T6,HPSOBLP 改进了文献[17,8] 中算法所发现的最优解,本文发现的最优解为(0.642 288,0.644 529,2.441 662,1.331 785),对应的上下两层目标函数值分别为 - 7.956 49 和 - 1.515 178.

测试函数 T7,算法能够发现许多可行解,这些可行解具有相同的上层规划目标函数值 - 11.999,但是具有不同的下层规划目标函数值,比如说表 5 给出的 - 459.225. 这主要是因为上层规划问题的目标函数值完全由下层规划问题的决策变量 y 来标定,而上层规划问题的决策变量不影响上层规划问题的目标函数值,它只影响下层规划问题的目标函数值. 算法可以发现许多可行解,这些可行解具有相同的下层决策变量值 $y=(2.998\ 52,2.998\ 52)$,但是具有不同的 x 值.

测试函数 T9,算法同样发现了一个较好的解(- 0.434 399,0.781 03,2.000 007,0),对应的上下两层目标函数值分别为 - 3.920 14 和 - 1.992 92.

对于非线性双层规划问题,HPSOBLP 发现的最优解也优于信任域方法[2] 例如 T10,算法发现的最优解为(0,0.577 914),与信任域方法相比具有较小的上层规划问题的目标函数值. 函数 T11,算法 HPSOBLP 发现的上层规划问题的目标函数值为 15.440 04,对应的最优解为(1.999 94,0),明显优于信任域方法[2] 对于测试函数 T12 和 T13,算法 HPSOBLP 同样改进了文献中其它算法所发现的最优解. 对于其它测试函数,算法 HPSOBLP 也同样发现和其它算法一样好的最优解.

综上所述,表 3 和表 5 中的数据表明,算法 HPSOBLP 对于大多数测试函数都发现了比其它几种算法好的最优解. 最重要的一点是 HPSOBLP 可以求解不同类型的 BLPP,算法的运用不需要借助于任何特定的假设条件(比如可微、库恩 - 塔克条件等),也不需要对目标函数作任何形式的改变和处理. 需要指出的是:上述实验结果是在约束违背度容限 $\varepsilon=7\times10^{-5}$ 下得到的,通过进一步减小约束违背度容限的值来提高算法的计算精度.

3 结 论

双层规划问题是一个 NP - hard 问题,针对特定类型的问题,目前文献中已经提出了许多求解算法. 作为一个新的群体智能优化算法,粒子群算法已经被大量的应用证明是一个比较有竞争力的优化算法,本文将其应用拓展到求解 BLPP,并构建了求解一般 BLPP 的层次粒子群算法(HPSOBLP). 本文所提出的算法是一个由两个 PSO 变形算法所组成的层次算法框架,文中所设计的变形算法分别用来求解上层和下层约束优化问题. 层次粒子群算法通过两个 PSO 算法的变形算法之间的交互迭代作用,来反映 BLPP 的决策过程,从而有效地求解一般 BLPP. 与其它优化算法的实验比较结果表明本文所提出的算法是一个可选择的鲁棒性较高的求解一般 BLPP 的算法,它不需要借助于任何假设条件,也不需要对目标函数或者约束条件作任何形式的转换和处理. 需要指出的是尽管 HPSOBLP 能够序列地求解一般的双层规划问题,但是由于其迭代求解问题的机制可能会造成算法在求解大规模实际问题时计算时间会很高. 可以通过在并行框架下执行 HPSOBLP 来减少算法的执行时间. 此外将进一步研究算法中参数选择问题,并且通过大规模的实际应用问题来进一步检验算法的性能.

参考文献：

[1]Dempe S. Foundations of Bilevel of Programming[M]. Volume 61 of Nonconvex optimization and its application Boston: Kluwer Academic Publisher, 2002.

[2]Colson B, Marcotte P, Savard G. Bilevel programming: A survey[J]. 4OR: Quarterly Journal of the Belgian, French and Italian Operations Research Societies, 2005, 3: 87—107.

[3]Candler W, Townsley R. A linear two-level programming problem[J]. Computers and Operations Research, 1982, 9: 59—76.

[4]Bard J F, Moore J T. A branch and bound algorithm for the bilevel programming problem[J]. SIAM Journal of Scientific and Statistical Computing, 1990, 1: 281—292.

[5]Liu G, Han J, Wang S. A trust region algorithm for bilevel programming problems[J]. Chinese Science Bulletin, 1998, 43: 820—824.

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

[7]Genderau M, Marcotte P, Savard G. A hybrid tabu-ascent algorithm for the linear bilevel programming problem[J]. Journal of Global Optimization, 1996, 8: 217—233.

[8]Wang Y, Jiao Y, Li H. An evolutionary algorithm for solving ninlinear bilevel programming based on a new constraint-handling scheme[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C, 2005, 35: 221—232.

[9]Sahin K H, Ciric A R. A dual temperature simulated annealing approach fo r solving bilevel programming problems[J]. Computers and Chemical Engineering, 1998, 23: 11—25.

[10]Kennedy J, Eberhart R C. Particle Swarm Optimization[C]. Proceeding of IEEE International Conference on Neural Networks. Piscataway, NJ: IEEE Service Center, 1995. 1942—1948.

[11]Shi Y H, Eberhart R C. A Modified Particle Swarm Optimizer[C]. Proceeding of IEEE World Congress on Computational Intelligence. Anchorage, AK: IEEE Press, 1998. 69—73.

[12]*Li X Y, Tian P, Kong M. A Novel Particle Swarm Optimization for Constrained Optimization Problems*[C]. Proceeding of 18th Australian Joint Conference on Artificial Intelligence. Sydney, Australia: Springer Berlin / Heidelberg, 2005. 1305—1310.

[13]Powell D, Skolnick M. Using Genetic Algorithms in Engineering Design Optimization with Nonlinear Constraints[C]. Proceeding of the Fifth International Conference on Genetic Algorithms. Los Altos, CA: Morgan Kaufmann Publisher, 1993. 424—431.

[14]Ratnaweera A, Halgamuge S, Watson H. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients[J]. IEEE Transactions on Evolutionary Computation, 2004, 8: 240—255.

[15]Shi Y H, Eberhart R C. Parameter Selection in Particle Swarm Optimization[C]. Proceeding of 1998 Annual Conference on Evolutionary Programming. San Diego: Springer-Berlin, 1998. 591—600.

[16]Calvete H I, Gale C. Theory and methodology: The bilevel linear/linear fractional programming problem[J]. European Journal of Operation Research, 1999, 114: 188—197.

[17]Outrata J V. On the numerical solution of a class of stackelberg problems[J]. Zeitschrift Fur Operation Research, 1990, 34: 255—278.

[18]Aiyoshi E, Shimizu K. A solution method for the static constrained stackelberg problem via penalty method[J]. IEEE Transactions on Automatic Control, 1984, 29: 1111—1114.

[19]Shimizu K, Aiyoshi E. A new computational method for syackelberg and min-max problems by use of a penalty method[J]. IEEE Transaction on Autommatic Control, 1981, AC-26: 460—466.

[20]Bard J. Convex two-level optimization[J]. Mathematical Programming, 1988, 40: 15—27.

Particle swarm optimization for solving bilevel programming problems

LI Xiang-yong, TIAN Peng

Antai College of Economics & Management, Shanghai Jiaotong University, Shanghai 200052, China

Abstract: In this paper we propose a hierarchical particle swarm optimization method to solve general bilevel programming problems. Unlike traditional algorithms designed for solving specific types of bilevel programs, the proposed approach provides a hierarchical algorithm framework, which solves the general bilevel programs by simulating its decision process. In the proposed algorithm, solving for general bilevel programs is transformed into iteratively solving for the upper-level and lower-level programming problems using two variants of particle swarm optimization. The experimental results of the proposed algorithm, as compared with those of other algorithms, show that the proposed hierarchical particle swarm optimization is another effective algorithm for solving general bilevel programming problems.

Key words: particle swarm optimization; metaheuristic; bilevel programming problem; constrained optimization

附录：测试函数

1. 测试函数 T1[18]

$$\min_x \quad F(x,y) = 2x_1 + 2x_2 - 3y_1 - 3y_2 - 60$$

$$\text{s.t. } x_1 + x_2 + y_1 - 2y_2 \leqslant 40, 0 \leqslant x_1 \leqslant 50,$$

$$0 \leqslant x_2 \leqslant 50$$

$$\min_y \quad f(x,y) = (y_1 - x_1 + 20)^2 + (y_2 - x_2 + 20)^2$$

$$\text{s.t. } 2y_1 - x_1 + 10 \leqslant 0, 2y_2 - x_2 + 10 \leqslant 0$$

$$-10 \leqslant y_1 \leqslant 20, -10 \leqslant y_2 \leqslant 20$$

2. 测试函数 T2[19]

$$\min_x \quad F(x,y) = (x_1 - 30)^2 + (x_2 - 20)^2 - 20y_1 + 20y_2$$

$$\text{s.t. } -x_1 - 2x_2 + 30 \leqslant 0, x_1 + x_2 - 25 \leqslant 0, x_2 \leqslant 15$$

$$\min_y \quad f(x,y) = (x_1 - y_1)^2 + (x_2 - y_2)^2$$

$$\text{s.t. } 0 \leqslant y_1 \leqslant 10, 0 \leqslant y_2 \leqslant 10$$

3. 测试函数 T3[20]

$$\min_x \quad F(x,y) = -x_1^2 - 3x_2 - 4y_1 + y_2^2$$

$$\text{s.t. } (x_1)^2 + 2x_2 \leqslant 4, x_1 \geqslant 0, x_2 \geqslant 0$$

$$\min_y \quad f(x,y) = 2x_1^2 + y_1^2 - 5y_2$$

$$\text{s.t. } x_1^2 - 2x_1 + x_2^2 - 2y_1 + y_2 \geqslant -3$$

$$x_2 + 3y_1 - 4y_2 \geqslant 4, y_1 \geqslant 0, y_2 \geqslant 0$$

4. 测试函数 T4[16]

$$\min_x \quad F(x,y) = -8x_1 - 4x_2 + 4y_1 - 40y_2 - 4y_3$$

$$\text{s.t. } x_1 \geqslant 0, x_2 \geqslant 0$$

$$\min_y \quad f(x,y) = \frac{1 + x_1 + x_2 + 2y_1 - y_2 + y_3}{6 + 2x_1 + y_1 + y_2 - 3y_3}$$

$$\text{s.t. } -y_1 + y_2 + y_3 + y_4 = 1,$$

$$2x_1 - y_1 + 2y_2 - \frac{1}{2}y_3 + y_5 = 1$$

$$2x_2 + 2y_1 - y_2 - \frac{1}{2}y_3 + y_6 = 1, y_i \geqslant 0, i = 1, \cdots, 6$$

5. 测试函数 T5—T9[17]

$$\min_x \quad F(x,y) = r(x_1^2 + x_2^2) - 3y_1 - 4y_2 + 0.5(y_1^2 + y_2^2)$$

$$\min_y \quad f(x,y) = 0.5[y_1 \quad y_2]H[y_1 \quad y_2]^T -$$

$$b(x)^T[y_1 \quad y_2]^T$$

$$\text{s.t. } -0.333y_1 + y_2 - 2 \leqslant 0, y_1 - 0.333y_2 - 2 \leqslant 0,$$

$$y_1 \geqslant 0, y_2 \geqslant 0$$

T5) $r = 0.1, H = \begin{bmatrix} 1 & -2 \\ -2 & 5 \end{bmatrix}, b(x) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

T6) $r = 1$, H, and $b(x)$ are the same as in T5

T7) $r = 0, H = \begin{bmatrix} 1 & 3 \\ 3 & 10 \end{bmatrix}, b(x) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

T8) $r = 0.1$, H, and $b(x)$ are the same as in T7

T9) r and H are the same as in T8.

$$b(x) = \begin{bmatrix} -1 & 2 \\ 3 & -3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

6. 测试函数 T10[2]

$$\min_x \quad F(x,y) = x^2 + (y - 10)^2$$

$$\text{s.t. } x + 2y - 6 \leqslant 0, -x \leqslant 0$$

$$\min_y \quad f(x,y) = x^3 - 2y^3 + x - 2y - x^2$$

（下转第 110 页）

Optimal pricing policy for high technology perishable commodity with revenue management

GUAN Zhen-zhong, SHI Ben-shan

School of Economics and Management, Southwest Jiaotong University, Chengdu 610031, China

Abstract: The pricing of the high-tech products, which are the typical perishable ones, is a decisive effect on the profit of the retailers'. After considering customer's substitutable behavior under out of inventory, in order to gain maximum expected profit, the retailer's pricing policy for two perishable commodities are discussed with stochastic demand based on multi-logit consumer choice model and service level with revenue management. The model which applies single stage are numerically analyzed. Optimal policy with different customer arriving rate, initial inventory and different effect degree to customer are discussed, and a series of characters and principles are drawn.

Key words: revenue management; perishable commodity; pricing policy; consumer choice model; high technology product

（上接第 52 页）

s.t. $-x+2y-3\leqslant 0,\ -y\leqslant 0$

7. 测试函数 T11[2]

$$\min_{x} \quad F(x,y)=(x-5)^2+(2y+1)^2$$

s.t. $-x\leqslant 0$

$$\min_{y} \quad f(x,y)=(x-1)^2-1.5xy+x^3$$

s.t. $-3x+y+3\leqslant 0, x-0.5y-4\leqslant 0$

$x+y-7\leqslant 0,\ -y\leqslant 0$

8. 测试函数 T12[2]

$$\min_{x} \quad F(x,y)=(x-5)^4+(2y+1)^4$$

s.t. $x+y-4\leqslant 0,\ -x\leqslant 0$

$$\min_{y} \quad f(x,y)=e^{-x+y}+x^2+2xy+y^2+2x+6y$$

s.t. $-x+y-2\leqslant 0,\ -y\leqslant 0$

9. 测试函数 T13[2]

$$\min_{x} \quad F(x,y)=(x_1-y_2)^4+(y_1-1)^2+(y_1-y_2)^2$$

s.t. $-x_1\leqslant 0$

$$\min_{y} \quad f(x,y)=2x_1+e^{y_1}+y_1^2+4y_1+2y_2^2-6y_2$$

s.t. $6x_1+2y_1^2+e^{y_2}-15\leqslant 0,\ -y_1\leqslant 0,\ y_1-4\leqslant 0$

$5x_1+y_1^4-y_2-25\leqslant 0,\ -y_2\leqslant 0,\ y_2-2\leqslant 0$