第3章 基于共轭先验分布的单模分布估计算法

3.1 概　　述

本章主要介绍基于共轭先验分布的两层分布估计(Two-level Hierarchical Estimation of Distribution Algorithm,THEDA)算法。分布估计算法作为演化算法的一个分支,其全局搜索与局部搜索的关系仍是一个基本的问题[28]。全局搜索(exploration)是探索新空间的过程,强调搜索的广度;而局部搜索(exploitation)是探索已搜索空间邻域的过程,强调搜索的深度。全面而深入地搜索整个决策空间是最理想的方式。但是有限的计算资源要求算法合理地平衡这两种搜索策略。例如,在传统的遗传算法中,交叉算子主要体现了全局搜索策略,而变异算子则主要体现了局部搜索策略。仅强调全局搜索会产生质量较差的解,而过分强调局部搜索会使算法陷入近优解,因此一个演化算法需要合理平衡这两方面,以利用最少的计算资源获得最好的搜索效果。如果算法可以在计算的开始尽可能地探索更多的区域,并且随着计算的进行逐步过渡到局部搜索上去,就可以获得较好的性能。由于分布估计算法是根据已经选择的个体估计种群分布的概率模型,这可以看作一个根据已有样本推断其分布的过程。受到贝叶斯推断过程的启发,利用 Beta 分布是二项分布的共轭先验分布这一性质,本章提出了基于 Beta 分布的两层分布估计算法。与其他单变量分布估计算法相同,THEDA 也使用概率向量作为基本模型。但是 THEDA 不直接从种群估计对应的概率向量,而是估计概率向量可能的分布,即先估计一个 Beta 向量,其中每个元素表示对应的概率向量中元素的分布。通过 Beta 分布采样得到概率向量,最后通过概率向量采样得到新种群。使用“分布的分布”这一思想使得这种算法框架有两个主要的优点:首先,利用“分布的分布”有助于直接提高种群的多样性,并且通过限制 Beta 分布中参数的取值范围可以避免变量的收敛,这可以帮助已经陷入局部最优状态的算法获得改变状态的机会;其次,在这种框架下,通过控制 Beta 向量的参数更新过程,可以很好地控制运算中全局搜索和局部搜索的权重。在 0-1 背包问题和 NK-Landscapes 问题上的测试实验说明 THEDA 的性能显著优于其他单变量分布估计算法。

本章内容安排如下:3.2 节给出共轭先验分布的概念和基本性质,主要介绍 Beta 分

布和二项分布的关系；3.3 节完整描述两层分布估计算法，包括算法框架以及模型的学习和演化过程；3.4 节利用 3 个测试问题测试算法的性能；3.5 节总结本章的内容。

3.2　Beta 分布与二项分布

频率推断和贝叶斯推断[29]是统计推断的两种主要方法。在频率推断方法中，未知参数被看作具有固定的取值。为了获得准确的结果，使用频率推断的方法需要较大规模的样本。在贝叶斯推断中，未知参数被看作一个随机变量，并且这个变量具有一个概率分布。这种方法可以避免极端情况下错误推断的产生。贝叶斯推断的核心是贝叶斯公式[30]：

$$p(\theta \mid X, H) = \frac{p(X \mid \theta) p(\theta \mid H)}{\int_{\theta} p(X \mid \theta) p(\theta \mid H) \mathrm{d}\theta} \tag{3-1}$$

其中，$p(\theta \mid X, H)$是后验概率，即根据数据 X 更新后的 θ 的概率。式(3-1)等号右侧的分子包含两部分：似然概率 $p(X \mid \theta)$和先验概率 $p(\theta \mid H)$。在 $p(\theta \mid H)$中，H 为超参数。理论上，先验分布本可以取任意形式的函数。然而选择共轭先验分布会减少计算量。假设一个 n 重伯努利实验具有未知参数 θ。可以通过做实验并利用实验结果推断参数 θ 的取值。在实验开始前，假设参数 θ 服从某个参数为 H 的概率分布，即 θ 服从 $p(\theta \mid H)$，这个分布也称作先验分布。通过 n 次实验，获得了实验结果 X，利用式(3-1)可以计算出在获得实验结果的情况下参数 θ 的分布情况。Beta 分布[31]是一个定义在区间[0,1]的连续概率分布，其概率密度函数定义如下：

$$p(\theta; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} \theta^{\alpha-1} (1-\theta)^{\beta-1} \tag{3-2}$$

其中，$B(\alpha, \beta)$是 Beta 函数。Beta 分布中的两个参数 α 和 β 决定了概率密度函数的形状，如图 3-1 所示。

Beta 分布的一个重要性质是它为二项分布的共轭先验分布。假设在上述 n 重伯努利实验中出现 s 次正例和 $f = n - s$ 次反例，那么这组实验的采样分布为

$$p(s, f \mid \theta) = \binom{n}{s} \theta^{s} (1-\theta)^{f} \tag{3-3}$$

如果选取参数为 α 和 β 的 Beta 分布作为 θ 的先验分布。利用式(3-1)计算其后验概率：

$$\begin{aligned} p(\theta \mid s, f, \alpha, \beta) &= \frac{p(s, f \mid \theta) p(\theta \mid \alpha, \beta)}{\int_{\theta} p(s, f \mid \theta) p(\theta \mid \alpha, \beta) \mathrm{d}\theta} \\ &= \frac{\frac{\binom{s+f}{s} \theta^{s+\alpha-1} (1-\theta)^{f+\beta-1}}{B(\alpha, \beta)}}{\int_{0}^{1} \frac{\binom{s+f}{s} \theta^{s+\alpha-1} (1-\theta)^{f+\beta-1}}{B(\alpha, \beta)} \mathrm{d}\theta} \end{aligned}$$

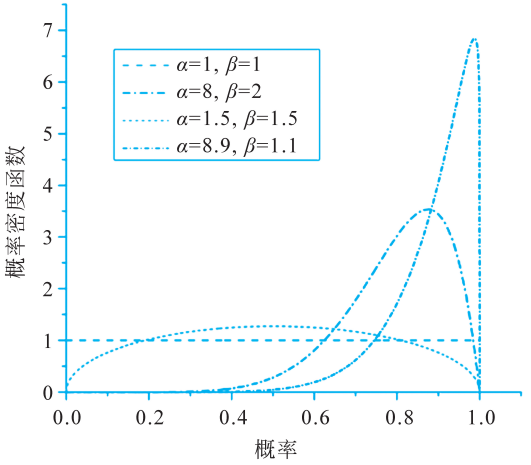

图 3-1　不同参数下 Beta 分布的概率密度函数

$$= \frac{\theta^{s+\alpha-1}(1-\theta)^{f+\beta-1}}{B(s+\alpha, f+\beta)}$$
$$= p(\theta \mid \alpha+s, \beta+f) \tag{3-4}$$

θ 的后验分布也是一个 Beta 分布，只是两个参数变为 $\alpha+s$ 和 $\beta+f$。在这个问题上，Beta 分布的两个参数具有直观的物理意义，分别表示实验中正例和反例出现的次数。需要指出的是，这两个参数可以取非整数值。Beta 分布的概率函数的方差为

$$\mathrm{var}(\theta) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \tag{3-5}$$

方差表明参数 θ 的不确定性。参数 α 和 β 同时增加会导致方差减小。其物理意义是，当实验结果中正例和反例的数量都增加时，参数的不确定性减小。参数 $\alpha=1$、$\beta=1$ 的 Beta 分布被称作贝叶斯先验分布，这时 Beta 分布为区间[0,1]上的均匀分布。这时熵最大，表明没有关于 θ 的信息，θ 在区间[0,1]上的取值是均匀的。

3.3　两层分布估计算法

3.3.1　算法框架

基于共轭先验分布的两层分布估计算法与其他算法的最大区别是采用了层次化模型。分布估计算法一般直接利用选择的个体估计概率模型并用以产生新个体。在两层分布估计算法中，选择的个体用来估计一个 Beta 向量，其定义如下。

定义 3-1　Beta 向量是由 Beta 分布组成的，每个分量包含定义在[1,+∞)区间上的

两个实数。如下所示：

$$\mathrm{BetaV}=\begin{bmatrix}\alpha_1 & \alpha_2 & \cdots & \alpha_n\\ \beta_1 & \beta_2 & \cdots & \beta_n\end{bmatrix}$$

其中，(α_i,β_i)确定了一个 Beta 分布。因此，Beta 向量表示由 n 个 Beta 分布组成的概率分布数组。Beta 向量中每个参数的定义域为$[1,+\infty)$，这是因为，Beta 分布的参数的定义域为$[0,+\infty)$，并且当参数大于或等于 1 时，可以保证概率密度的最大值不会出现在两端。可以通过对 Beta 向量采样产生一个概率向量，再对这个概率向量采样得到新解。图 3-2 是 THEDA 的基本框架。其中，Pop 代表种群(Population)，Sel 代表选择(Selection)，Best 代表最优的或最好的个体。由图 3-2 可以看出，该算法的基本框架是一个两层结构，第一层是从 BetaV 到 PV，第二层是从 PV 到种群，因此该算法称为两层分布估计算法。在利用子种群估计下一代 BetaV 时，学习速率显式地由学习速率函数控制。模型更新和学习速率函数将在 3.3.2 节详细介绍。

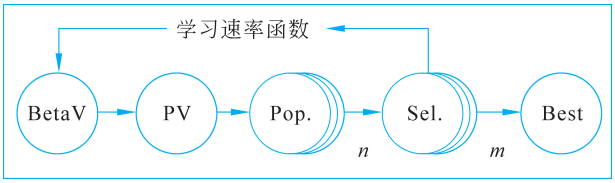

图 3-2　THEDA 的基本框架

计算过程中适应度最高的解将保留下来并作为算法最后的输出。算法 3-1 给出了 THEDA 的伪代码。算法具体过程描述如下。第 1～4 行为算法的初始化过程。首先，变量 t 为算法当前的迭代次数，初始化为 0。更新过程中的学习速率函数将根据 t 确定学习速率。算法开始时，学习速率较小，这时的算法倾向于随机搜索；随着 t 的增大，通过增大学习速率使模型尽可能多地进行局部搜索。第 2～4 行的循环用于将概率向量 PV 的每个元素初始化为 0.5。在这个条件下，算法采样得到的初始种群是符合均匀分布的。第 5～18 行是迭代的主循环。循环开始时，首先通过对 PV 采样得到本轮迭代的初始种群。对每个采样得到的个体 P_j，利用目标函数评价得到其适应度 F_j。在 THEDA 中，使用截断选择(truncation selection)算子从种群中选取最优秀的 $p_s\times n=m$ 个个体组成子种群 S。

算法 3-1　THEDA

```
1:  t←0
2:  for i=1,2,…,l do
3:      PV_i←0.5
```

```
4:  end for
5:  while 未达到终止条件 do
6:      for j=1,2,…,n do
7:          for i=1,2,…,l do
8:              P_ji←sampleBernoulli(PV_i)
9:          end for
10:         F_j←evaluate(P_j)
11:     end for
12:     S←select(P,F)
13:     BetaV←updateModel(S,t)
14:     for i=1,2,…,l do
15:         PV_i←sampleBernoulli(BetaV_i)
16:     end for
17:     t←t+1
18: end while
```

利用选取的个体更新 Beta 向量,其具体方法如算法 3-2 所示。模型更新过程在 3.3.2 节介绍。主循环的最后一步是通过 Beta 向量采样得到一个新的概率向量。

算法 3-2 THEDA 的模型更新过程

```
1:  procedure updateModel(S,t)
2:      for i=1,2,…,l do
3:          num1←0
4:          num0←0
5:          for j=1,2,…,m do
6:              if S_ji=1 then
7:                  num1←num1+1
8:              else
9:                  num0←num0+1
10:             end if
11:         end for
12:         (BetaV_i).α←1+num1×f_L(t)
13:         (BetaV_i).β←1+num0×f_L(t)
14:     end for
15:     return BetaV
16: end procedure
```

3.3.2　模型更新过程

如 2.3 节所述，模型与相应的模型更新策略是一个分布估计算法区别于其他分布估计算法的主要部分。理想的模型具有相对简单的形式并且可以通过较少的计算资源实现更新。模型更新的过程就是通过选取的子种群学习模型的过程。这个过程与贝叶斯推断有相似性。THEDA 从选取的个体推断其模型的分布。

Beta 分布的两个参数具有直观的物理意义。在贝叶斯推断中，参数 α 和 β 表示观测结果中正例和负例出现的次数。在二进制编码条件下，α 和 β 可以看作 0 和 1 的个数，因此可通过统计每个决策变量上所有个体中 0 和 1 的个数作为 Beta 向量对应元素的参数。但是直接将统计结果赋予参数 α 和 β 并不能产生理想的更新结果，因此要使用学习速率函数显式控制这个更新过程。学习速率函数的定义如下。

定义 3-2　学习速率函数是一个定义域为正整数、值域为实数的单调函数，记为 $f_{\mathrm{L}}(t)$。

在 THEDA 中，学习速率函数的自变量为当前迭代的次数 t。理想的演化算法应该随着计算的进行由全局搜索逐步过渡到局部搜索。由 3.2 节可知，可以通过同时增加 Beta 分布的两个参数减小其方差，直接减小第一层采样的不确定性，进而加强局部搜索。因此，学习速率函数应该为单调增函数。这个性质可以保证算法随着计算的进行而逐渐加强局部搜索。

算法 3-2 为模型更新过程的伪代码。在更新过程中，首先统计选取的子种群中每个决策变量上所有个体中 0 和 1 的个数。为了利用学习速率函数控制算法的搜索，将 0 和 1 的个数乘以学习速率作为对应的 Beta 分布的参数，这样，Beta 分布的方差会随着计算的进行而逐步减小，算法逐渐加强局部搜索。在定义 3-2 中，Beta 分布的参数定义域为 $[1,+\infty)$，这本质上是一种加一平滑[32](add-one Smoothing)方法，广泛应用于自然语言处理等领域。当 num0 或者 num1 为 0 时，对于一个决策变量来说，种群中的所有个体都具有相同的取值。在这种情况下，若不使用平滑技术，那么这个变量会收敛到一个值上。从此以后，这个变量不会发生改变，进而使算法陷入近优解而无法跳出。例如，当 Beta 分布的一个参数为 0 时，不妨假设 $\alpha=0$，这时 Beta 分布退化为一个单点分布，即 $p(x=1)=1$。在随后的采样中，概率向量中的对应分量只会为 1，这个分量已经收敛，随后所有产生的个体都具有相同的取值。所以，α 和 β 的值要再加上 1 以避免这种情况的发生。通过这种限制，可以保证 Beta 分布是在区间[0,1]上的连续分布，且概率最大值不会出现在区间两端。在 THEDA 中，学习速率函数为简单的线性函数：

$$f_{\mathrm{L}}(t)=c\times t \tag{3-6}$$

其中，c 为决定学习速率的系数，t 为当前迭代的次数。

3.4 测试与实验

3.4.1 测试问题

1. Onemax 问题

Onemax 问题是简单的与变量无关的单峰问题。其搜索空间为$\{0,1\}^n$,其中 n 是问题的规模。目标函数定义如下:

$$f_{\text{Onemax}}(\boldsymbol{z})=\sum_{i=1}^{n}z_i \tag{3-7}$$

优化目标是求解最大值,全局最优解为 $\boldsymbol{z}=[1\ 1\ \cdots\ 1]$,对应的目标函数值为 n。在本节中使用问题规模 $n=1000$ 的实例,记为 $P_{\text{Onemax1000}}$。这个简单的问题用于测试算法参数对算法性能的影响。

2. 0-1 背包问题

0-1 背包问题是著名的组合优化问题[33]。问题描述如下:有一个物品的集合,其中每个物品具有重量和价值两个属性;同时有一个背包,具有一定容量。问题的求解目标是找到一个选取物品的方案,即选取一个物品集合的子集,最大化所有选取物品的价值并且满足总重量不超过背包的容量这一限制条件。0-1 背包问题的形式化定义如下:

$$\max\ f(x)=\sum_{i=1}^{N}p_i x_i$$

$$\text{subject to}\ \sum_{i=1}^{N}w_i x_i \tag{3-8}$$

其中,w_i 和 p_i 分别是第 i 个物品的重量和价值。这里使用了文献[22]中所采用的问题实例生成方法,即

$$w_i=\text{uniformly\_random}[1,10] \tag{3-9}$$

$$p_i=w_i+5 \tag{3-10}$$

在测试中,使用了两种不同类型的背包容量:一种是将背包容量设定为所有物品总重量的一半;另一种是将背包容量设定为固定值 20。这两种容量分别记为 C_{h} 和 C_{f}。

$$C_{\text{h}}=\frac{1}{2}\sum_{i=1}^{N}w_i \tag{3-11}$$

$$C_{\text{f}}=20 \tag{3-12}$$

在计算中生成的解可能不满足问题的限制条件,即不在可行域内。为了保证解满足限制条件,需要使用随机修复算子[22]。对于一个方案,随机修复算子的作用如下:当物

品总重量超过背包容量时，随机去除一些物品，直到满足限制条件；在物品总重量不超过背包容量时，随机增加一些物品以尽可能提高解的适应度。其具体过程如算法 3-3 所示。

算法 3-3　0-1 背包问题的随机修复算子

```
procedure repairOperator(x)
    if Σ_{i=1}^{N} w_i x_i ≤ C then
        feasible = true
    else
        feasible = false
    end if
    while !feasible do
        随机将 x 中的一个为 1 的元素改写为 0
        if Σ_{i=1}^{N} w_i x_i ≤ C then
            feasible = true
        end if
    end while
    while feasible do
        随机将 x 中的一个为 0 的元素改写为 1，假设改写的元素为 x_i
    if Σ_{i=1}^{N} w_i x_i > C then
            feasible = false
            x_i = 0
        end if
    end while
end procedure
```

修复算子首先判断给定解是否满足限制条件。当不满足限制条件时，随机将解向量中值为 1 的元素改写为 0，直到满足限制条件为止。这时再尝试增加 1 的个数，以尽可能提高解的质量。当解满足限制条件时，随机将解向量中值为 0 的元素改写为 1，以尽可能提高解的质量。

在章节实验中，问题的规模分别为 250、500、1000 和 2000。将实例记为 $P_{\text{knp\_}D\text{\_Ch}}$ 或者 $P_{\text{knp\_}D\text{\_Cf}}$。其中，$D$ 为问题规模的大小，Ch 和 Cf 分别表示 C_{h} 和 C_{f} 两种背包容量。

3. NK-Landscapes 问题

NK-Landscapes 是一个产生不同目标函数的模型[34,35]。它由 Kauffman 提出并且广泛应用在演化算法性能测试上[26,36-38]。其适应度函数定义如下：

$$F(x)=\frac{1}{N}\sum_{i=1}^{N}F_i(x_i;x_{i_1},x_{i_2},\cdots,x_{i_K}) \tag{3-13}$$

其中,$\{i_1,i_2,\cdots,i_K\}\subset\{1,\cdots,i-1,i+1,\cdots,N\}$。$N$ 是问题的规模;K 为每个变量邻域的大小,表示变量之间的相关程度。邻域的生成有两种方法:一种是选取位置相邻的 K 个基因位作为邻域;另一种是随机选取 K 个基因位组成邻域。Kauffman 研究了生成邻域的两种方法的性质。适应度贡献(fitness contribution)函数 $F_i(x_i;x_{i_1},x_{i_2},\cdots,x_{i_K})$ 是一个从二进制字符串到实数的映射,具体的对应关系由模型生成时随机产生。有研究从理论上已经证明[39],当 $K\geqslant 3$ 并且采用随机邻域生成方法时,NK-Landscapes 模型产生的优化问题实例是 NP-完全的。本章实验使用随机邻域生成方法。问题的规模设定为 256、512、1024、2048 和 4096,K 取 0～8 的所有整数,因此,总共有 $5\times 9=45$ 个实例用于测试算法。

3.4.2 参数研究

THEDA 包含 3 个参数,分别为种群规模 n、选择压力 P_s 和学习速率 $f_L(t)$。本节将在 Onemax 问题上探究 3 个参数的选取对算法性能的影响。根据单一变量原则,本节的实验分为 3 部分。

为了测试种群大小的影响,将选择压力和学习速率函数设定为 $P_s=0.1$,$f_L(t)=0.5t$,参见式(3-6),问题的规模设定为 1000,测试中种群规模 n 设定为 20、100、200 和 500。记录算法在 50 次独立运算中的最好结果并求其均值和方差,结果如图 3-3 所示。实验结果表明,当种群规模 n 设定为 20、100 和 200 时,算法性能相近;当 $n=500$ 时,算法性能出现下降。这说明种群规模对算法性能的影响有限。

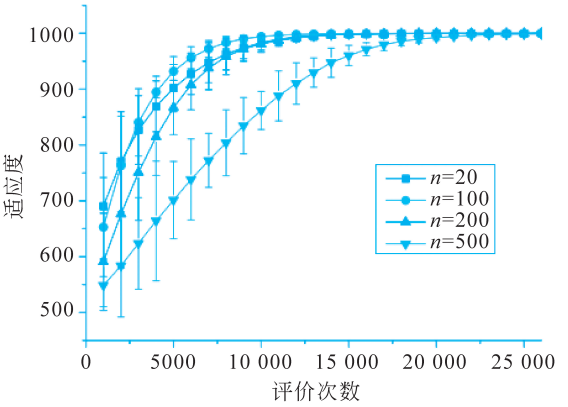

图 3-3 种群规模的影响

为了测试选择压力的影响，将种群规模 n 设定为 200，将学习速率函数设定为 $f_L(t)=0.5t$，将选择压力设定为 0.05、0.1、0.2 和 0.5，结果如图 3-4 所示。当 P_s 等于 0.05、0.1 和 0.2 时，算法性能相近；当 P_s 等于 0.5 时，算法性能较差。导致这个结果有两个原因：首先，选择压力过小的时候无法有效驱使种群移动；其次，较小的选择压力会产生较大的子种群。模型的参数增长很快，Beta 向量中每个 Beta 分布的方差快速减小。这不利于多样性保持，影响了算法对搜索空间的搜索能力。

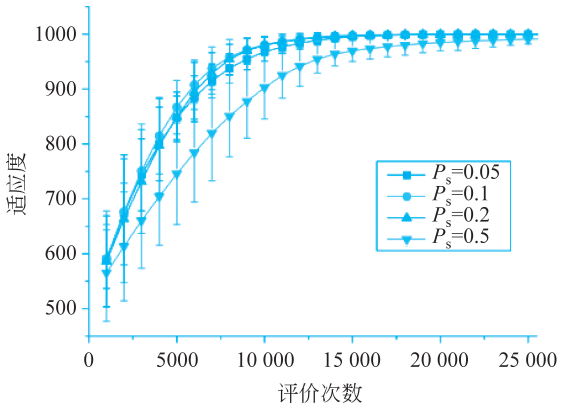

图 3-4　选择压力的影响

为了测试学习速率的影响，将种群大小设定为 200，将选择压力设定为 0.1。学习速率函数的系数 c 分别设定为 0.1、0.2、0.5、1.0 和 2.0。实验结果如图 3-5 所示。

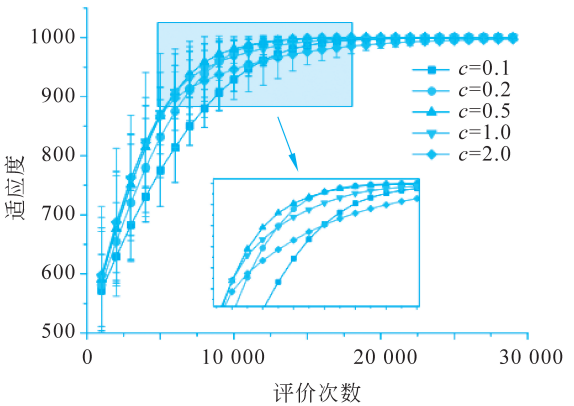

图 3-5　学习速率的影响

首先将 $c=0.5$ 的实验结果作为基线。当系数减小时，算法的整体性能下降；当系数增加时，计算的开始阶段和基线的表现相近，评价次数超过 5000 次时算法的性能下降。

3.4.3 实验设置

本节将 THEDA 与 cGA、PBIL 和 QEA 进行对比。算法的性能都受到其参数设定的影响。

后 3 种算法的参数设定如下：

- 对于 cGA，其唯一的参数为虚拟种群规模 n。在本节实验中采用文献[40]中推荐的设置。在文献[26]中也使用了这个设置。
- PBIL 具有 4 个参数：学习速率 L、变异概率 P_m、变异偏移量 S 和种群规模 n。前 3 个参数分别设定为 0.1、0.02 和 0.05，这与文献[26]的设置相同。在本节所涉及两个实验中，PBIL 的性能受到种群规模的影响较大。通过在实例 $P_{knp\_500\_Ch}$ 测试了 10，20，…，100 共 10 个参数的结果，最终选取效果最好的取值，$n=20$。
- QEA 的所有参数采用其原始文献中的推荐设置[22]。

在 THEDA 中，根据 3.4.2 节的实验结果，种群规模、选择压力和学习速率 3 个参数分别设置为 $n=200$，$P_s=0.1$ 以及 $L=0.5$。本节涉及的 4 个算法的参数如表 3-1 所示。

表 3-1 4 个算法的参数设置

算　法	参　数
cGA	$n=\frac{\sqrt{\pi}}{2}\sqrt{D}\log_2 D$
PBIL	$n=20, L=0.1, P_m=0.02, S=0.05$
QEA	$n=10, \Delta\theta=0.01\pi, S_g=100, S_l=1$
THEDA	$n=200, P_s=0.1, f_L(t)$使用式(3-6)

算法在每个实例上独立运行 50 次。为了实现公平的比较，终止条件设定为最大的目标函数调用次数。对于所有 0-1 背包问题的实例，最大评价次数为 40 000；而对于 NK-Landscapes 问题的实例，最大评价次数设定为 10 000。算法发现的最好结果都被记录以用于计算统计信息(包括均值和方差)并用来比较显著性。在本节实验中，使用了 $\alpha=0.05$，即显著性为 95%的威尔科克森符号秩检验(Wilcoxon signed-rank test)计算算法得到结果是否存在的显著差异。

3.4.4 实验结果

对于背包容量为 C_h 的实例，实验结果如图 3-6～图 3-9 所示。对于背包容量为 C_f

的实例，实验结果如图 3-10～图 3-13 所示。表 3-2 给出了 4 个算法在 NK-Landscapes 问题上的实验结果。

在图 3-6～图 3-9 中，均值用虚线表示，标准差用围绕在均值附近的实线表示。在实例 $P_{\text{knp\_250\_Ch}}$ 和 $P_{\text{knp\_500\_Ch}}$ 上，cGA、PBIL 和 THEDA 的性能相近。对于 $P_{\text{knp\_250\_Ch}}$ 实例，cGA 和 PBIL 的收敛速度较 THEDA 快，但是 THEDA 得到的最终结果较好。

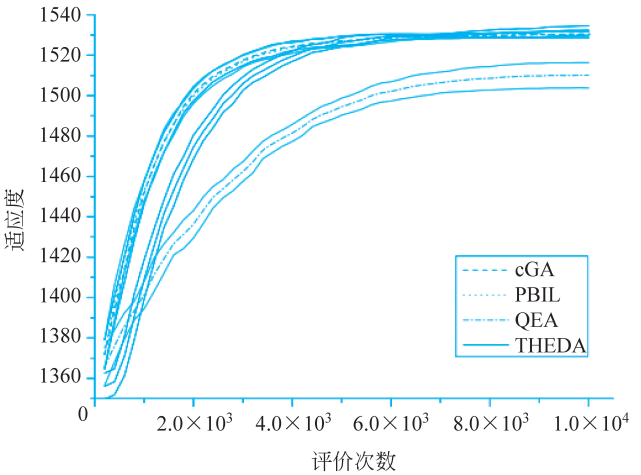

图 3-6　$P_{\text{knp\_250\_Ch}}$ 的实验结果

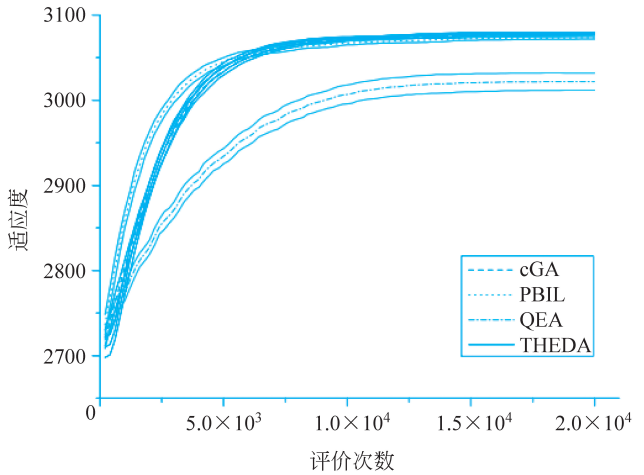

图 3-7　$P_{\text{knp\_500\_Ch}}$ 的实验结果

对于 $P_{\text{knp\_1000\_Ch}}$ 和 $P_{\text{knp\_2000\_Ch}}$，THEDA 具有性能上的优势。cGA 可以得到和

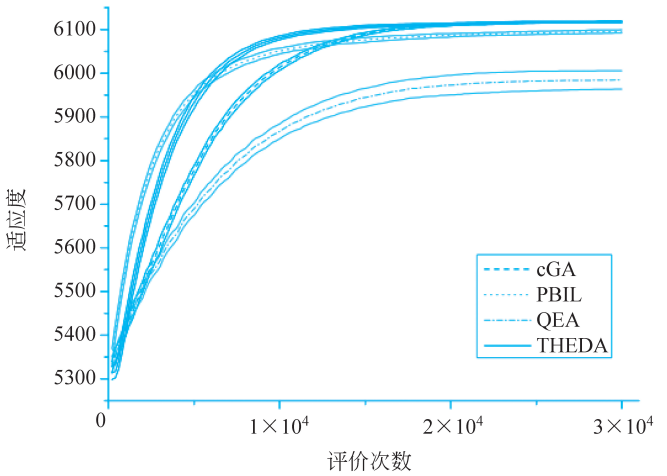

图 3-8 $P_{\text{knp\_1000\_Ch}}$ 的实验结果

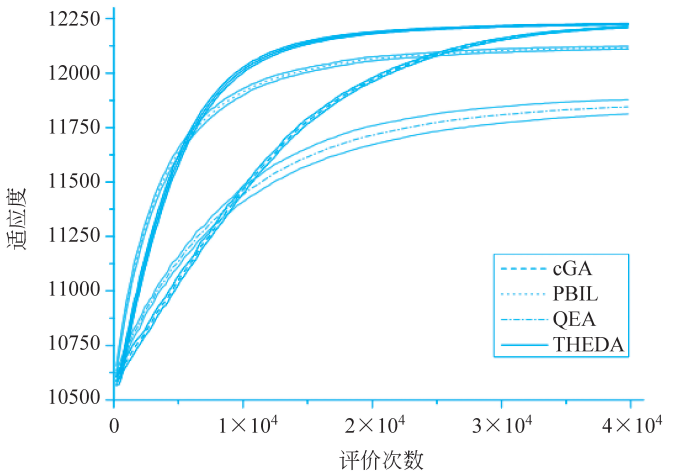

图 3-9 $P_{\text{knp\_2000\_Ch}}$ 的实验结果

THEDA 相近的最终结果，但是 cGA 的收敛速度比 THEDA 慢。PBIL 由于早熟导致最终结果比 THEDA 差。这说明 THEDA 在速度和解的质量上取得了平衡，其原因在于 THEDA 通过学习速率很好地控制了不同阶段的搜索侧重。

对于背包容量为 C_f 的实例，PBIL 和 THEDA 的表现优于 cGA 和 QEA。在实例 $P_{\text{knp\_250\_Cf}}$ 上，PBIL 在前 4000 次函数调用时比 THEDA 的收敛速度快，但是随后其收敛速度减慢。THEDA 的最终结果比 PBIL 好，体现在两方面：首先，其最优结果的均值优于 PBIL；其次，THEDA 计算结果的方差较 PBIL 的小。这说明 THEDA 在这一类问题上

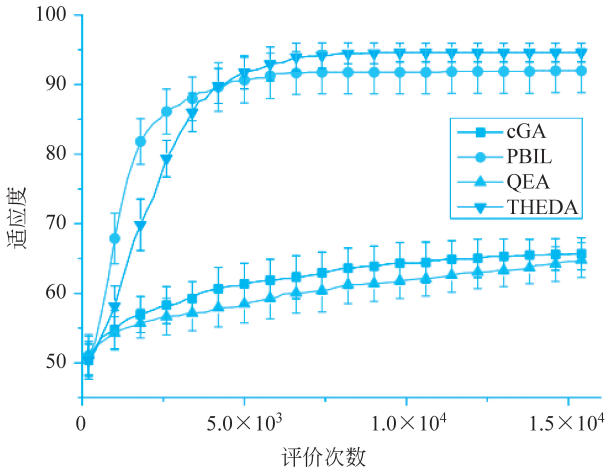

图 3-10　$P_{knp\_250\_Cf}$ 的实验结果

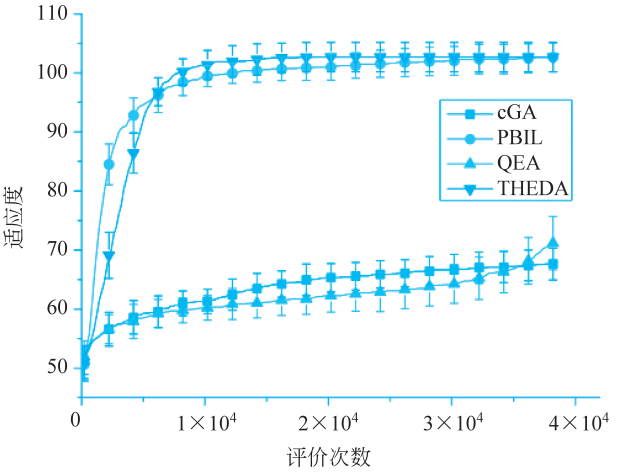

图 3-11　$P_{knp\_500\_Cf}$ 的实验结果

的性能稳定。

整体观察图 3-10～图 3-13，可以发现如下规律：早熟是 PBIL 存在的主要问题，并且这种现象随着问题规模的增大而增大。特别地，在实例 $P_{knp\_2000\_Cf}$ 上，PBIL 在前 10 000 次评价时都具有较快的速度，但是它陷入了局部最优结果中，平均适应度为 75；而 THEDA 的最优结果可以保持持续增长，直到评价次数达到 30 000 次。THEDA 在 $P_{knp\_2000\_Cf}$ 上的结果均值为 109.091，对应标准差为 1.8311。

表 3-2 中每个算法有两行数据，上面的数据是对应问题实例最优结果的均值，下面括

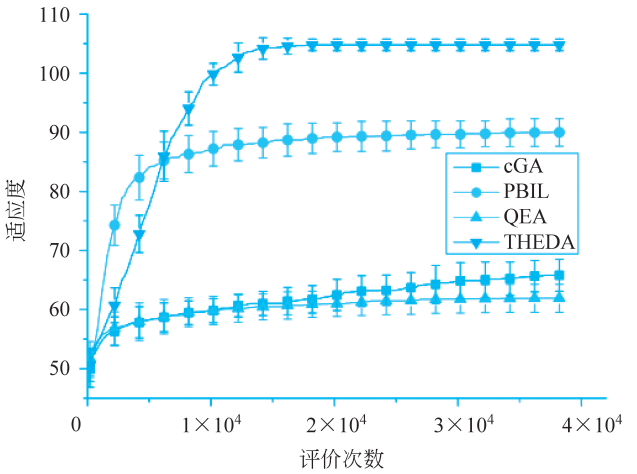

图 3-12 $P_{knp\_1000\_cf}$ 的实验结果

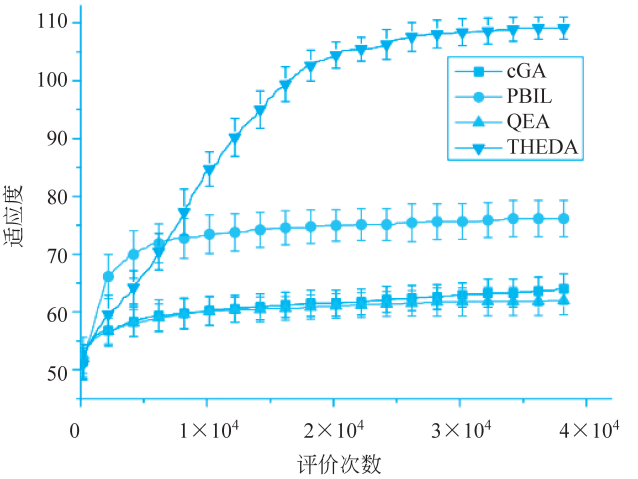

图 3-13 $P_{knp\_2000\_cf}$ 的实验结果

号内的数据为方差。对于问题的每个实例,用威尔科克森符号秩检验对 4 个算法的结果进行两两比较。如果一个算法的结果显著优于其他 3 个算法,则结果用粗体标明。对于 $N=256, K=1,2,4,5$ 的实例,没有算法显著优于其他 3 个算法。对于 $N=512, K=0,1$ 的实例,cGA 得到了最好的结果。在 $N=1024,2048,4096$ 的所有实例上,THEDA 均显著优于 cGA、PBIL 和 QEA。

表 3-2　NK-Landscapes 问题实验结果

算法	K								
	0	1	2	3	4	5	6	7	8
					$N=256$				
cGA	0.641252 (0.000639)	0.697722 (0.003005)	0.715339 (0.005676)	0.718578 (0.007910)	0.727358 (0.008082)	0.714238 (0.009171)	0.711088 (0.008034)	0.703995 (0.008495)	0.698919 (0.007174)
PBIL	0.642170 (0.000473)	0.697342 (0.003175)	0.719601 (0.006325)	0.724593 (0.006647)	0.735050 (0.007652)	0.725144 (0.010061)	0.725791 (0.010665)	0.720824 (0.008373)	0.715214 (0.007610)
QEA	0.637570 (0.001423)	0.681395 (0.005467)	0.693284 (0.007880)	0.691700 (0.008092)	0.701041 (0.007666)	0.691674 (0.009777)	0.692305 (0.008657)	0.692704 (0.008595)	0.687919 (0.008467)
THEDA	**0.642431** **(0.000044)**	0.697479 (0.003685)	0.719743 (0.006369)	**0.727683** **(0.006923)**	0.737937 (0.008598)	0.728732 (0.009517)	**0.732219** **(0.008500)**	**0.725509** **(0.009094)**	**0.721787** **(0.009589)**
					$N=512$				
cGA	**0.654224** **(0.000286)**	**0.709360** **(0.002152)**	0.709543 (0.004170)	0.713899 (0.005913)	0.699010 (0.006628)	0.681813 (0.007169)	0.661516 (0.007959)	0.633900 (0.009649)	0.614623 (0.007255)
PBIL	0.651499 (0.000761)	0.700830 (0.002868)	0.703134 (0.004996)	0.712613 (0.005582)	0.711476 (0.007017)	0.710398 (0.006873)	0.707592 (0.007741)	0.701077 (0.005883)	0.696201 (0.006571)
QEA	0.634382 (0.003024)	0.662207 (0.005111)	0.655832 (0.004756)	0.658920 (0.007519)	0.656774 (0.007016)	0.655020 (0.005639)	0.656440 (0.005695)	0.652938 (0.005639)	0.652081 (0.006118)
THEDA	0.654077 (0.000277)	0.706179 (0.002751)	**0.712181** **(0.004986)**	**0.720308** **(0.007044)**	**0.717787** **(0.007034)**	**0.719314** **(0.005399)**	**0.714162** **(0.006424)**	**0.708734** **(0.006506)**	**0.703489** **(0.006129)**
					$N=1024$				
cGA	0.648232 (0.001271)	0.640274 (0.002375)	0.635574 (0.003778)	0.614064 (0.004660)	0.586556 (0.004116)	0.568639 (0.003818)	0.556786 (0.003429)	0.547318 (0.003436)	0.541821 (0.002367)
PBIL	0.658673 (0.001170)	0.668541 (0.002988)	0.686582 (0.004120)	0.690710 (0.003966)	0.684507 (0.004650)	0.680655 (0.004773)	0.679084 (0.003772)	0.673617 (0.004636)	0.668580 (0.003397)

续表

算法	K								
	0	1	2	3	4	5	6	7	8
QEA	0.624419 (0.002902)	0.615505 (0.004216)	0.623718 (0.005392)	0.625974 (0.004041)	0.620952 (0.004708)	0.616966 (0.004411)	0.619642 (0.003558)	0.617106 (0.004309)	0.617028 (0.004763)
THEDA	**0.665160** **(0.000975)**	**0.676989** **(0.002180)**	**0.696151** **(0.003314)**	**0.701849** **(0.005103)**	**0.695694** **(0.004094)**	**0.693086** **(0.005711)**	**0.689914** **(0.004151)**	**0.683638** **(0.005314)**	**0.674471** **(0.005202)**
$N=2048$									
cGA	0.589031 (0.001419)	0.581530 (0.002088)	0.567904 (0.003851)	0.553999 (0.002500)	0.545029 (0.002244)	0.535523 (0.002719)	0.529628 (0.002799)	0.527372 (0.001770)	0.526312 (0.001916)
PBIL	0.634739 (0.001598)	0.649805 (0.002447)	0.657396 (0.003429)	0.656802 (0.003943)	0.656324 (0.003731)	0.648725 (0.004304)	0.642937 (0.003742)	0.637816 (0.003936)	0.632434 (0.003667)
QEA	0.591763 (0.003106)	0.592204 (0.003412)	0.592857 (0.003194)	0.591360 (0.003808)	0.591755 (0.002981)	0.589352 (0.003188)	0.588390 (0.002805)	0.587511 (0.003221)	0.586907 (0.002970)
THEDA	**0.645595** **(0.001314)**	**0.662890** **(0.002552)**	**0.670846** **(0.002899)**	**0.669722** **(0.003222)**	**0.667777** **(0.004247)**	**0.660937** **(0.003678)**	**0.652012** **(0.004527)**	**0.645109** **(0.004169)**	**0.637413** **(0.004151)**
$N=4096$									
cGA	0.545996 (0.001289)	0.542634 (0.002375)	0.531850 (0.001602)	0.526075 (0.001424)	0.523887 (0.001516)	0.521163 (0.001032)	0.518870 (0.001396)	0.518327 (0.001143)	0.517876 (0.001321)
PBIL	0.605037 (0.001372)	0.619889 (0.001589)	0.620039 (0.002278)	0.619318 (0.002679)	0.617842 (0.003042)	0.612343 (0.002723)	0.606235 (0.002853)	0.601265 (0.003238)	0.595149 (0.002673)
QEA	0.564602 (0.002484)	0.567269 (0.002312)	0.563595 (0.002332)	0.563918 (0.002364)	0.565783 (0.002344)	0.565418 (0.002466)	0.563724 (0.002315)	0.562946 (0.002133)	0.562668 (0.002418)
THEDA	**0.617069** **(0.001518)**	**0.632464** **(0.002110)**	**0.631715** **(0.002594)**	**0.631146** **(0.002382)**	**0.627607** **(0.002447)**	**0.621313** **(0.002529)**	**0.612567** **(0.002907)**	**0.604486** **(0.003620)**	**0.597990** **(0.002917)**

综合上述实验可以得到几个结论。首先，在 0-1 背包问题上的实验说明总体上 THEDA 和 PBIL 性能优于 cGA 和 QEA。THEDA 和 PBIL 之间存在一个超越点，在大约 6000 次评价之前 PBIL 所得到的结果较好，随后 THEDA 的结果超过了 PBIL，在不同的实例上都有类似的现象。其次，$P_{\text{knp\_2000\_Cf}}$ 上的实验表明 THEDA 可以很好地平衡全局搜索和局部搜索。最后，THEDA 在所有实验中使用了相同的参数，并且随着问题规模的增大，其相对其他算法的优势也越来越显著。

3.5　本 章 小 结

本章主要介绍了两层分布估计算法。利用共轭先验分布构造两层分布估计算法框架，方便地引入了控制模型演化的方法。在 0-1 背包问题和 NK-Landscapes 问题上的实验表明，两层分布估计算法的性能优于其他单变量分布估计算法。并且其性能优势随着问题规模的增大而增加。