最优臂取样的启发式算法研究

（申请清华大学工学硕士学位论文）

培 养 单 位：计 算 机 科 学 与 技 术 系
学　　科：计 算 机 科 学 与 技 术
研　究　生：胡　巍
指 导 教 师：李　建

二〇一八年六月

Heuristic Algorithms for Best Arm Identification Problem

Thesis Submitted to

Tsinghua University

in partial fulfillment of the requirement

for the professional degree of

Master of Engineering

by

Wei Hu

(Computer Science and Technology)

Thesis Supervisor : Associate Professor Jian Li

June, 2018

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；(2) 为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容。

本人保证遵守上述规定。

（保密的论文在解密后应遵守此规定）

作者签名：_____　　导师签名：_____

日　　期：_____　　日　　期：_____

摘 要

本论文研究了最优臂选择问题。学习了几个已知的著名算法，如连续删除算法和上置信边界算法。从这些算法中总结出一个差值方程来控制算法所需的样例数目。基于这个差值方程，我们设计了一个启发式的类似与上置信边界算法的取样方法，并且通过模拟数据实验选取平方根方程作为这个差值方程。另外，我们还设计了短尾停止条件和动态取样方法使得这个启发式算法获得更好的结果。最后我们用模拟数据测试这个启发式算法的正确性，并且和已知的算法进行比较，用实验数据验证这些启发式的步骤确实能再实际数据中带来效能上的提升。

关键词：最优臂取样；启发式算法

Abstract

We studies the 最有臂选择问题 problem and formalize gap function from the 逐次删除算法 and lil'UCB 算法 to control the number of samples. Using gap functions, we designed a heuristic UCB-style sample algorithm using a square root gap function determined from a serial experiment results. We also design short tail stopping condition and demand sampling scheme to make a heuristic algorithm gain better performance. Then we test the correctness of the algorithm and the efficiency of the algorithm by comparing it with some known algorithms in the simulation experiment.

Key words: Best Arm Identification Problem; Heuristic Algorithm

Contents

主要符号对照表

A	手臂的集合
D_i	手臂关联的分布
A_i	第 i 个手臂
μ_i	手臂A_i 的期望均值
$\hat{\mu}_i$	手臂A_i 的经验均值
Δ_i	手臂A_1 和A_i 的期望均值差
$g(t)$	差值方程
$t(\Delta)$	尾方程

第 1 章　导论

1.1　多臂老虎机问题

多臂老虎机问题是一个经典且被深度研究的决策理论问题。在很多不同的领域，诸如统计控制，统计学，机器学习和理论计算机科学等等，都有大量的关于多臂老虎机模型的研究文献。

定义 1.1 (多臂老虎机问题)：假设给定一个大小为 n 手臂的集合 $A = \{A_1, A_2, \ldots, A_n\}$，每个手臂的收益是一个未知的平均值在 $\mu_i \in [0,1]$ 上的分布。每当摇动一个手臂，该手臂会从对应的分布中独立的进行一个抽样并将抽样结果作为这次摇臂的收益。我们称这样的摇臂过程为一个多臂老虎机问题的算法。

多臂老虎机问题可以想象为如下场景：假设你要从一排的老虎机选出最好的一台，你可以决定你怎么玩这些老虎机，比如按什么顺序来玩这些老虎机，在特定的阶段对于特定的老虎机该玩多少次，什么时候结束。在你的决策过程中，你需要平衡勘探问题（认知所有的机器）和开发问题（认知最优的机器）。多臂老虎机问题的目标很多种，除了上述的选择最好的一台，还包括最小化损失之和，最大化收益之和。（比如 [1,2]）

在大量的应用中，多臂老虎机问题有过广泛的研究。比如，在医疗领域的重症治疗过程中，治疗手段不当的代价非常的高，所以需要最小化治疗的整体损失，这个过程可以模拟为多臂老虎机问题，其中每一种治疗方法为其中的一个手臂。另一个例子，在手机通讯领域，在通讯前一台手机需要从诸多通讯频道中选择一个，其通过对于信道的多次尝试后选择一个最优的，这也可以认为是多臂老虎机问题。（[3,4]）还有很多其他的例子，比如在线广告选择[5,6]，众包问题[7,8]，金融投资组合设计[9] 等等。

1.2　最有臂选择问题

定义 1.2 (最有臂选择问题)：在定义 1.1中，假设 $\delta \in (0,1)$，最有臂选择问题是设计算法，使得算法满足以至少 $1-\delta$ 概率返回均值最大的手臂。固定δ，我们的目标是最小化摇臂的次数。

最有臂选择问题是多臂老虎机问题的一个特例，它首先进行取样过程然后根据取样结果直接返回最大的手臂，这可以看作纯粹的勘探问题。本文中最有臂选

择问题的目标是最小化摇臂次数，被称为固定置信度 (δ) 设定。还有一种固定预算设定，给定了摇臂的次数，需要最小化最小化置信度。本文中我们只研究固定置信度设定。

为了简化定义，我们使用如下的约定和记号：

- 手臂的平均值是以降序排列的：$\mu_1 > \mu_2 > \cdots > \mu_n$。
- 最大的经验均值：$\hat{\mu}^*$ 。
- 手臂A_i 的经验均值：$\hat{\mu}_i$ 。
- 最大手臂和第 i 大手臂的均值差：$\Delta_i = \mu_1 - \mu_i$。

1.3　经典结果调研

最有臂选择问题可以回溯到 1950 年代 [10,11]，早期的结果在专著 [12] 总结。在近一二十年，这个问题引起了极大的关注 [4,13–23]。最优 k 个手臂选择问题作为最有臂选择问题的一个扩展，需要选取 k 个最大的手臂，该问题近年来也引起了广泛的研究 [7,23–31].

接下来介绍最有臂选择问题已知算法的上界和下界。

1.3.1　算法下界

[13] 的中的算法的期望下界为 $\Omega(\sum_{i=2}^{n}\Delta_i^{-2}\log\delta^{-1})$。[30] 也提供了一个算法有如上所述的下界，但在常数上有所改进。[19] 中证明了当置信度δ 趋近 0 的渐近复杂度的解析解。[17] 对于特定的 n，证明了另一种类型的算法 $\Omega(\sum_{i=2}^{n}\Delta_i^{-2}\log\log n)$ 这个结果以前的下界进行了细分。[16] 证明了对于两个手臂问题有 $\Delta^{-2}\log\log\Delta^{-1}$ 下界，这个结果是 [32] 的一个推论，对于任意的取样方法，我们都有如下不等式:

$$\lim_{\Delta\to 0}\sup\frac{E_\Delta[T]}{\Delta^{-2}\log\log\Delta^{-1}} \geq 2-4\delta$$

。能不能把两个手臂的结论推广到通用的情况下仍然是一个开放的问题。[15] 通过实验结果断言下界中的 $(\log\log\Delta^{-1})$ 项是无法被消除的。[20] 也提出一个差值-熵断言称 $(\log\log\Delta^{-1})$ 项是必须的。

1.3.2　算法上界

近二十年来，最有臂选择问题的上界有很大的改进 [15–17,27,30,33]. [33] 中提供了逐次删除算法，该算法的上界为

$$O\left(\sum_{i=2}^{n}{\Delta_i}^{-2}\left(\log\delta^{-1}+\log n+\log{\Delta_i}^{-1}\right)\right)$$

[15] 中提供了指数差值删除算法，该算法的上界为

$$O\left(\sum_{i=2}^{n}{\Delta_i}^{-2}\left(\log\delta^{-1}+\log\log{\Delta_i}^{-1}\right)\right)$$

[16] 中提供了 lil'UCB 算法，这个算法的上届和 [15] 一样。[17] 提供一个算法，将上界改进到

$$O\left({\Delta_2}^{-2}\log\log{\Delta_2}^{-1}+\sum_{i=2}^{n}{\Delta_i}^{-2}\left(\log\log\min({\Delta_i}^{-1},n)+\log\delta^{-1}\right)\right)$$

到目前为止上下界之间的仍然有空隙。

第 2 章　经典算法回顾

2.1　全取样算法

Algorithm 1 全取样算法 (ϵ, δ)

每个手臂都取样 $\frac{4}{\epsilon^2}\log(\frac{2n}{\delta})$ 次。

让$\hat{\mu}_i$ 手臂A_i 的经验均值。

输出 $\arg\max_{A_i \in A} \hat{\mu}_i$.

PAC 模型。给定 $\delta, \epsilon \in (0,1)$，最有臂选择问题满足以置信度$\delta$ 返回一个 ϵ-最优的手臂，它就是一个 (δ, ϵ)-PAC 算法。

全取样算法是一个 PAC 模型算法。这个算法用暴力取样的方式，对每个手臂进行等量的取样，然后返回经验均值最大的手臂。全取样算法并不是一个高效的算法，但是这个算法的非常简单，很适合进行启发式的改变。事实上，很多高级算法都可以视为是对全取样算法算法的改进。

2.2　中位数删除算法

Algorithm 2 中位数删除算法 (ϵ, δ)

Let $S_1 = A, \epsilon_1 = \epsilon/4, \delta_1 = \delta/2, r = 1$。

while $|S_r| > 1$。 **do**

对于 S_r 中的每一个手臂取样 $\frac{4}{\epsilon_r}\log\frac{3}{\delta_r}$ 次。

设$\hat{\mu}_i$ 手臂A_i 的经验均值。

从 $\{\hat{\mu}_i\}$ 找出经验均值的中位数 m_r。

设 $S_{r+1} = S_r / \{A_i : m_r \geq \hat{\mu}_i, \forall A_i \in S_r\}$。

设 $\epsilon_{r+1} = \frac{3}{4}\epsilon_r, \delta_{r+1} = \delta_r/2, r = r+1$.

end while

输出 S_r 中的手臂。

中位数删除算法同样是一个 PAC 模型下的算法 [14]。每一轮次，该算法都像全取样算法一样对每个存活的手臂进行取样，然后删除经验均值低于中位数的

手臂。当删到只有一个手臂的时候，算法停止。[14] 证明该算法在 PAC 模型下有 $\Omega((n/\epsilon)\log\delta^{-1})$ 上界，同时证明在 PAC 模型下，这也是下界，所以中位数删除算法是 PAC 模型下最有臂选择问题的最优的理论算法，但是常数非常的大。

2.3　逐次删除算法

Algorithm 3 逐次删除算法 (δ)

设 $r = 1, S_1 = A, c \geq 4$。

while $|S_r| > 1$。 **do**

对于 S_r 中的每个手臂取样一次。

设$\hat{\mu}_i$ 是A_i 的经验均值。

设 $\hat{\mu}^* = \max_{A_i \in S_r} \hat{\mu}_i, \alpha_r = \sqrt{\frac{\log(cnr^2/\delta)}{r}}$。

设 $S_{r+1} = S_r / \{A_i : \hat{\mu}^* - \hat{\mu}_i \geq 2\alpha_r, \forall A_i \in S_r\}$。

设 $r = r + 1$。

end while

输出 S_r 中的手臂。

每一轮，逐次删除算法 [13] 都对存活的手臂进行一次取样，然后删除经验均值比最大的低太多的手臂，这个过程由一个差值参数 α_r 删除的速度。当只有一个手臂留下时，算法停止。该算法的复杂度为 $O\left(\sum_{i=2}^{n} {\Delta_i}^{-2}\left(\log\delta^{-1} + \log n + \log {\Delta_i}^{-1}\right)\right)$

2.4　指数差值删除算法

每一轮，指数差值删除算法 [15] 使用了中位数删除算法作为子过程，其参数以指数的方式进行设置 2^{-r-2}，然后把这个子过程的结果作为参考手臂，删除所有均值不是 ϵ_r-最优的手臂。当只剩一个手臂时，算法停止。该算法的复杂度为 $O\left(\sum_{i=2}^{n} {\Delta_i}^{-2}\left(\log\delta^{-1} + \log\log {\Delta_i}^{-1}\right)\right)$。指数差值删除算法有着最好的上界但是常数非常大。

2.5　lil'UCB 算法

每一轮，lil'UCB 算法 [16] 都根据一个叫做置信上界的值进行取样。每次取样后，这个置信上界就会降低，从而让别的手臂得到取样的机会。

Algorithm 4 指数差值删除算法 (δ)

设 $S_1 = A, r = 1$。

while $|S_r| > 1$ **do**

设 $\epsilon_r = \frac{2^{-r}}{4}, \delta_r = \frac{\delta}{50r^3}$。

对于 $A_i \in S_r$ 中的手臂取样 $\frac{2}{\epsilon_r^2}\log\frac{2}{\delta_r}$ 次。

设$\hat{\mu}_i$ 手臂A_i 的经验均值。

设 A_{i_r} = 中位数删除算法 $(S_r, \epsilon_r/2, \delta_r)$。

设 $S_{r+1} = S_r/\{A_i : \hat{\mu}_i < \hat{\mu}_{i_r} - \epsilon_r, \forall A_i \in S_r\}$。

设 $r = r + 1$。

end while

输出 S_r 中的手臂。

Algorithm 5 lil'UCB(δ)

设 $\epsilon, \lambda, \beta, \sigma > 0$

设$T_i(r)$ 为第 r 轮手臂A_i 的总取样次数。

对每个手臂进行一次取样并初始化$T_i(r)$ 。

while $T_i(r) < 1 + \lambda \sum_{A_j \neq A_i} T_j(r)$ 对于所有的手臂A_i **do**

设$T_i(r)$ 第 r 轮A_i 的取样次数。

设$\hat{\mu}_i$ 第 r 轮A_i 的经验均值。

对手臂 $\arg\max_{A_i \in A}\left\{\hat{\mu}_i + (1+\beta)(1+\sqrt{\epsilon})\sqrt{\frac{2\sigma^2(1+\epsilon)\log\left(\frac{\log((1+\epsilon)T_i(r))}{\delta}\right)}{T_i(r)}}\right\}$ 进行一次取样。

设 $r = r + 1$。

end while

输出手臂 $\arg\max_{A_i \in A} \hat{\mu}_i$.

第 3 章　启发式算法

我们希望用启发式的算法设计来减少取样的次数。

3.1　算法要求

启发式算法可以看作算法设计中的一种捷径，其牺牲了算法的最优性，完整性，精确性以获得更好实践结果，因此在现实世界中有着广泛的应用。由于启发式算法的设计维度过于宽泛，其衡量标准也很模糊，因此我们固定一些常识性的要求用来测试我们设计的启发式算法的正确性。

- 正确性。给定一个现实的数据集合，启发式算法失败的数据个数不能比已知理论算法更多。虽然启发式算法没有理论的正确性证明，但是要保证实际运行的正确性。
- 效率。给定一个现实的数据集合，启发式算法需要比理论算法用更少的取样次数。这个要求是我们使用启发式的初衷，同时也是判断一个启发式算法好坏的重要标准。

3.2　手臂的难度

3.2.1　差值，难度和核心集

定义 3.1 (**差值**)：　两个手臂A_i 和A_j 的差值 $\Delta_{i,j}$ 定义为他们的均值差 $|\mu_i - \mu_j|$。

$\Delta_i = \mu_1 - \mu_i$ 最有意义的差值，因为我们算法的目的是找出最大的手臂，需要把最大的手臂和剩余的手臂区分开来。一般而言，差值的大小可以衡量当前问题的难易程度。对于一个大差值的两个手臂，我们需要较少的取样次数就能区分，对于一个小差值的手臂，我们则需要较多的取样的次数。根据定义，最难的一对手臂是最大的两个手臂A_1 and A_2 。

定义 3.2 (**难度**)：　手臂A_i 的难度定义为 $H_i = \Delta_i{}^{-2} \log\log \Delta_i{}^{-1}$。

当一个手臂对应的差值Δ_i 非常小，对应的难度H_i 非常大，则这个手臂则被称为难的手臂。可以看出，难度的和 $H = \sum_{i=2}^{n} H_i$ 和算法理论上界非常相似。实际上，我们用难度来估算把一个手臂A_i 和最大的手臂A_1 手臂区分开来所需的取样次数，所以我们用算法复杂度上界中的 $\Delta_i{}^{-2} \log\log \Delta_i{}^{-1}$ 项，常数则被我们省略了。根据

定义可以得出，排在前面的手臂的难度比后面的手臂的难度大很多，我们用核心集来描述这个比较难的手臂集合。

定义 3.3 (核心集)：　手臂的核心集 S_{core} 定义为最小的 k 个手臂，满足这些手臂的难度和 $H_{core} = \sum_{i=2}^{k} H_i$ 占据大部分 (90%) 的难度总和 H。

可以看出，核心集的手臂是一个算法要处理的最难的部分，从某种意义上（下界）来说，任何保证正确率的算法都不可避免的需要对这些难度的手臂进行取样和取那么多的取样次数。

3.2.2　差值方程，置信上界和置信下界

任意一个最有臂选择问题的算法都是由一次次的取样构成的，每次取样都是从一个未知的分布上返回一个样本。根据大数定律，当一个手臂的经验均值随着取样次数的增多而趋近于期望均值。因为期望均值未知，大部分算法都用经验均值来估算期望均值，所以算法的目的可以视为使经验均值越来越精确。我们用*差值方程*来描述这精确的程度。

定义 3.4 (差值方程)：　给定一个手臂 A_i ，方程 $g(t) : N \to R^+$ 是 A_i 的一个*差值方程*，当对于任意数量的 A_i 的样本个数 $t \in N$，其经验均值都以很高的概率满足 $|\hat{\mu}_i - \mu_i| < g(t)$。

给定一个差值方程 $g(t)$ ，可以导出经验均值 $\hat{\mu}_i$ 的上界和下界。

定义 3.5 (置信上界和置信下界)：　给定一个差值方程 $g(t)$ 和手臂 A_i 的取样次数 t，A_i 当前的经验均值*置信上界* $U_i(t)$ 定义为 $U_i(t) = \mu_i + g(t)$；其*置信下界* $L_i(t)$ 定义为 $L_i(t) = \mu_i - g(t)$。

根据定义 3.4我们得出 $L_i(t) \le \hat{\mu}_i \le U_i(t)$，所以我们只要对每个手臂取样足够的次数使得对于所有的其他手臂都满足 $L_1(t) > U_i(t)$，那么我们可以确信最大经验均值的手臂就是最优的手臂。然而手臂的概率分布是未知的，所以我们只能通过取样估算手臂的实际差值。一个合适的差值方程对于一个算法来说很重要，因为其能够缩小经验均值和期望均值的范围，可以帮助算法减少不必要的取样次数。相反，一个很差的差值方程则可能会导致错误的结果，比如，一个下降非常快的差值方程 (2^{-t})，会导致经验均值被错误的估计。下面是一些比较“好”的差值方程。

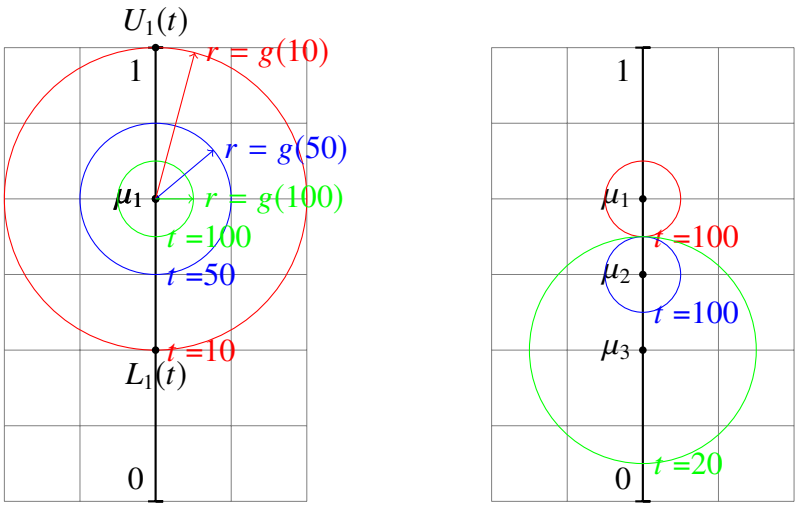
定义 3.6 (逐次删除差值方程)：　$g_{se}(t) = \sqrt{t^{-1} \log t}$

定义 3.7 (UCB 差值方程)：　$g_{ucb}(t) = \sqrt{t^{-1} \log \log t}$

定义 3.8 (平方根差值方程)：　$g_{sqrt}(t) = \sqrt{t^{-1}}$

定义 3.9 (多项式差值方程)：　$g_{poly} = \frac{1}{t^c}$, for a constant $0 < c < 1$

3.2.3　直观化取样过程：取样圆



(a) 红色的圆代表手臂A_1 在样本数量是 $t = 20$ 的时候的取样圆；$U_1(t)$ 和$L_1(t)$ 是对应的置信上届和置信下界。类似的，蓝色和绿色的圆分别代表样本数量是 $t = 50, 100$ 时候的取样圆。

(b) 当A_1 和A_2 的样本数量是 100，A_3 的是 20 的时候，红色的圆和绿色的圆完全分离了。这个时候绿色的圆则不需要获得更多样本。

Figure 3.1　直观化：取样圆

给定一个差值方程 $g(t)$，我们可以把取样过程直观化为一个变化的圆，参考图 3.1。一个手臂A_1 可以用一个位于 y-轴的圆来表示，其半径为该手臂当前的差值方程的值 $g(t)$。通过这种表现方式，该手臂的置信上界和置信下界分别为取样圆与 y-轴的上下交点。当取样数目增加的时候，对应的取样圆的半径则变小，参考图 3.1(a)。如果两个手臂的取样圆完全不相交，我们可以断定为上面的圆对应的手臂的期望均值一定比下面的圆大，同时不管下面的手臂再获得多少样本，其取样圆永远不会高于上面的圆，即我们可以安全的删除下面的手臂，参考图 3.1(b)。

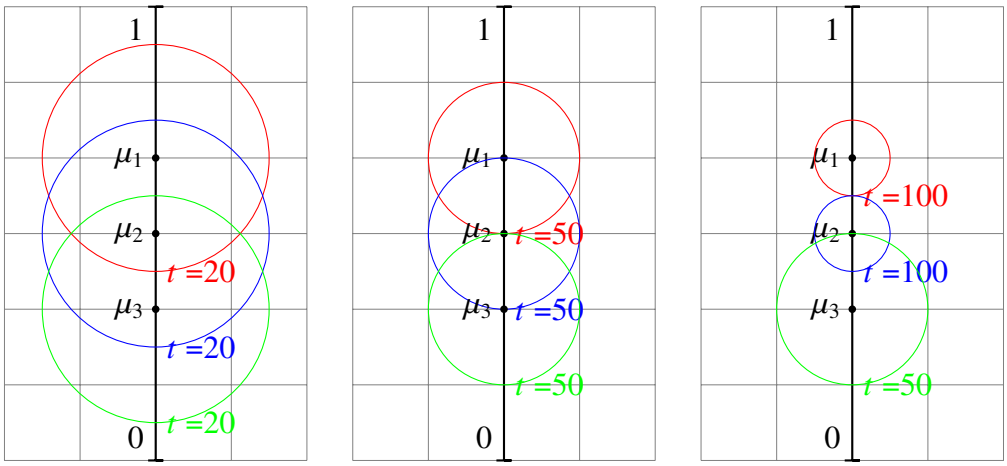
3.3　取样的方法

一个好的取样算法应该同时兼顾勘探和开发问题。对于开发而言，算法应该尽快的找到问题的核心集，然后把样本资源花在核心集上。对于勘探而言，算法需要保证所有的手臂都得到充分的取样，否则算法容易陷入局部最优从而输出错

误的结果。这就是关于*效率*和*覆盖率*之间的平衡。比如，全取样算法就是一个覆盖率很高但效率很低的算法。

在第2章中，我们回顾了一系列已知的算法。我们用差值方程和取样圆的这样更加直观的方式重新描述逐次删除算法和 lil'UCB 算法，这样可以更深入的理解算法的覆盖率和效率问题。

3.3.1　逐次删除的取样方法



(a) 当样本数量是 $t = 20$ 的时候，所有手臂获得相同的取样次数，故而其取样圆的半径相等。

(b) 当样本数量是 $t = 50$ 的时候，绿色的圆和红色的圆分离，所以对应的手臂被删除。

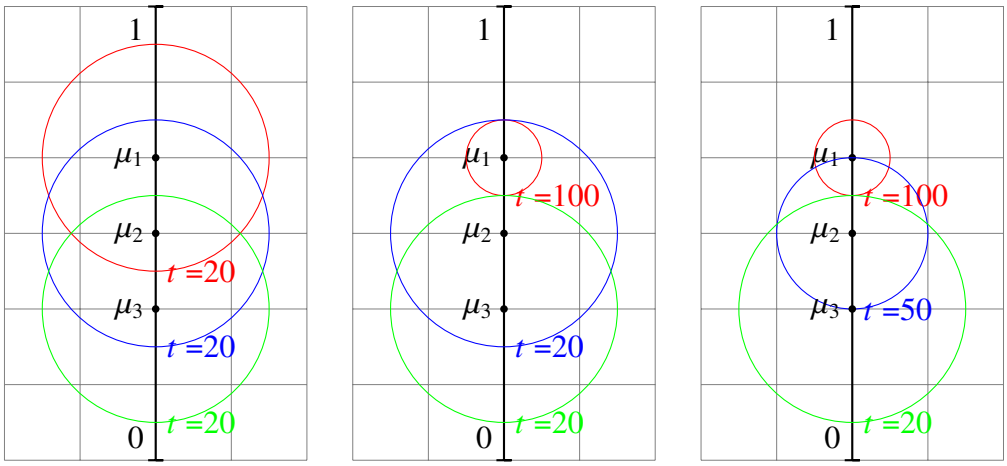
(c) 当样本数量是 $t = 100$ 的时候，绿色的圆因为被删除了，所以没有变化；蓝色的圆开始和红色的圆分离，同理也要被删除。

Figure 3.2　逐次删除取样方法

逐次删除取样方法是在算法 3[13] 中使用的一个取样方法。因为 逐次删除算法和全取样算法很相似，所以其有比较好的覆盖率。对于效率问题，该算法通过删除和最大的手臂的取样圆不相交的的手臂来避免过多的对那些比较容易的手臂取样，其差值方程是$g_{se}(t)$ (定义 3.6)。这个删除过程可以参见图 3.2。

3.3.2　置信上界取样方法

置信上界取样法在算法 5[16] 被使用。不像删除的机制，lil'UCB 算法每次选取具有最大的置信上界的手臂进行一次取样，其差值方程为$g_{ucb}(t)$ (定义 3.7)。这个算法的效率非常高，因为每次都只对一个最大的手臂进行取样。而对于那些比较的低的手臂，其获得取样机会比较小，但是由于其取样圆和最大的圆不相交，因



(a) 当样本数量是 $t = 20$ 的时候，具有最高置信上界的圆（红色）获得取样机会。

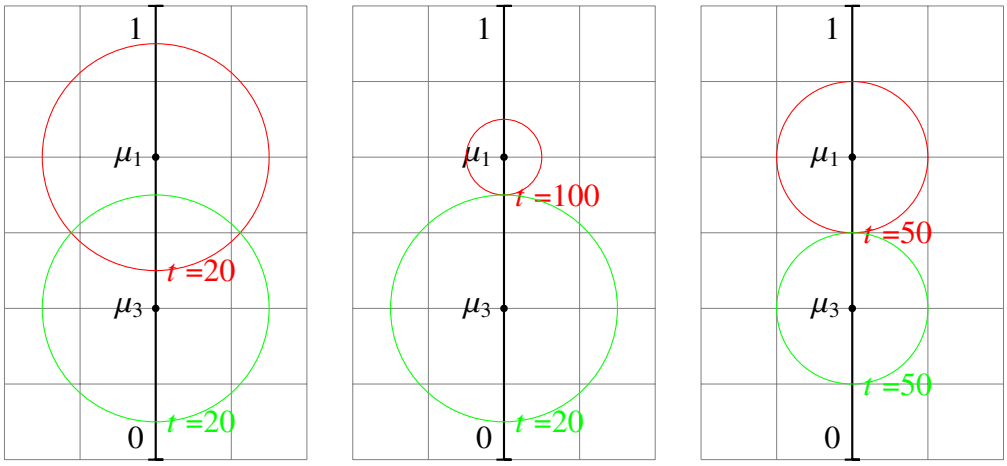
(b) 当样本数量是 100 的时候，蓝色的圆变成最高的圆，从而获得取样的机会。

(c) 当蓝色圆的样本数量是 50，红色的圆重新成为最高的圆而获得取样机会。

Figure 3.3　置信上界取样法

此保证的覆盖率。图 3.3描述这个过程。图中我们可以注意到，绿色的手臂一直得不到取样，因为他已经和最高的圆完全分离了。

3.3.3　在 UCB 取样法选取两个手臂



(a) 一开始的样本数量都是 20

(b) 每次只对最高的手臂进行取样，这种情况它需要大量样本是其取样圆足够小，使得两个圆不相交。

(c) 每次对两个圆同时进行取样，这样两个圆更容易分离。

Figure 3.4　在 UCB 算法中对两个手臂进行取样

这个技巧在很多 UCB 类的取样方法中被使用[16]，即其不但对具有最大置信

上界的手臂取样，还对具有最大经验均值的手臂进行取样，如果两个是同一个手臂，则对具有第二大经验均值的手臂进行取样。直观而言，一个算法的目的是选出最大的手臂来，不可避免需要对两个手臂进行比较，所以我们需要了解两个手臂的信息，所以需要同时对两个手臂进行取样。图 3.4直观的解释这种取样方式的好处。用这种取样方式的算法被称为 LUCB 算法。

3.4　终止条件

3.4.1　核心集的大小

部分的算法都需要找出核心集来，比如删除类的算法。如果核心集足够小了，通常来说是核心集只包含一个手臂，那么这个算法就可以终止，输出其中最大的手臂。逐次删除算法[13] 就是这样的例子。

3.4.2　足够数量的样本

在一个取样的过程中，如果某一个手臂获得远远多于其他手臂的样本数量，那么这个手臂很有可能就是最优的手臂。比如 lil'UCB 算法[16] 的终止条件就是具有最大经验均值的手臂的样本数量是剩下的手臂样本数量之和的 $\lambda = 9$ 倍 (算法 5)。

3.4.3　Lil'UCB 终止条件

如果最高的取样圆和剩余的其他的取样圆都分离了，那么这个算法就能直接终止。换言之，如果最大的手臂的置信下界比其余手臂的置信上界都要大，那么这个算法就可以终止。这种终止条件被成为 Lil' 终止条件 [16]。

3.4.4　快速终止条件

许多算法都需要把大量的样本数量分配到核心集的手臂上，这个过程通常发生算法的后半段。也就是说，这个算法在某个时刻之后，大部分的时间都只是对最大的两个手臂进行取样。另一方面，在算法取样过程中，具有最大经验均值的手臂通常不会变化。我们用稳定阶段来描述这种现象。

定义 3.10 (**稳定阶段**)：　一个算法在第 x-个样本的之后，其最大的和第二大的经验均值的手臂都保持不变，那么我们称这个算法进入稳定阶段。

一个算法在稳定阶段但没有停止的原因是其不满足终止条件，通常来说终止条件都是比较苛刻的。假设我们在算法进入稳定阶段后直接终止算法输出结果，我

们可以节省很多的样本数量。但是我们并不知道算法什么开始进入稳定阶段。下一节我们会对稳定阶段进行合理的启发式的猜测。

3.5　启发式的步骤

我们设计的启发式的算法借用了逐次删除算法和 lil'UCB 算法算法的框架，每一轮取样过程中，我们测试终止条件，如果不满足则继续取样。前面章节从许多方面介绍了这个过程的，下面我们对这些方面进行启发式的改进。

3.5.1　快速的差值方程

差值方程在很多算法中都有非常重要的作用，从某些方面而言，其决定了一个算法的复杂度。有理论上界的算法通常都会小心翼翼的选择差值方程来满足其理论证明，但对于启发是算法而言，差值方程的选择则可以更加随意。我们可以用下降更快的差值方程比如$g_{sqrt}(t)$ (定义 3.8)，这个方程把 $(\log\log\Delta^{-1})$ 项去掉了。注意到很多断言说 $(\log\log\Delta^{-1})$ 项是必须的，我们可以用实际的数据看看结果。我们甚至可以选择更加激进的差值方程 $g(t)=\sqrt{t^{-x}}$ 但是正确性需要用实验来证明。

3.5.2　激进的参数

启发式算法可以重新选择一些参数来减少取样，如

- 在删除类的算法中，当算法删掉 $n-k$ 个手臂时就终止算法，直接输出最大的经验均值的手臂。
- 我们可以通过选择不同的常数，比如逐次删除算法中的 ϵ 等
- 对于 lil'UCB 算法，我们可以选择 $\lambda=1$ 或者更小的数。

3.5.3　短尾停止条件

我们可以通计数的方式来猜测算法进入稳定阶段的时间来，从而加速算法停止条件。

定义 3.11 (**尾方程**)：　设A_1 和A_2 是同一个分布D_i ，$\hat{\mu}_1$ 是当A_1 的样本个数为 t_1 时的经验均值，$\hat{\mu}_2$ 是当A_2 的样本数量为 t_2 时候的经验均值。一个方程 $t(\Delta):R^+\to N$ 如果满足当 $t_1>t(\Delta_2)$ 和 $t_2>t(\Delta_2)$ 时，以很大的概率 (1 - δ) 满足 $\hat{\mu}_1>\hat{\mu}_2$，则这个方程被成为这个概率分布下的一个*尾方程*。

尾方程可以看作估算区分开两个手臂需要的最少样本数量。利用尾方程，我们可以假设，当一个算法取样过程中，其经验均值最大的两个手臂的样本数量都

大于对应的尾方程，则该算法进入稳定阶段。

定义 3.12 (**短尾停止条件**)：　设 $t(\Delta)$ 是一个尾方程，t_1 和 t_2 分别是最大的和第二大的手臂的样本数量。短尾停止条件被定义为同时满足 $t_1 \geq t(\Delta_2)$ 和 $t_2 \geq t(\Delta_2)$。

这个短尾终止条件显示了核心集的重要性，特别的核心集中最大的两个手臂，所以这个终止条件最好用在能够快速定位核心集的算法中，这个特性恰好是 UCB 类算法的一个特性。由于未知的手臂分布，我们只能用经验均值去估算其均值的差值，这个差值被用于尾方程的计算。同时我们还可以对尾方程的种类进行进行猜测：

定义 3.13 (**平方尾方程**)：　$t_{square} = \frac{1}{\Delta^2}$

定义 3.14 (**多项式尾方程**)：　$t_{poly} = \frac{1}{\Delta^c}$，给定常数 $1 < c < 2$

3.5.4　需求取样

在 LUCB 算法中，我们需要每次都对两个手臂进行取样。如果我们用短尾停止条件，我们会发现最大的手臂的取样次数远远大于第二大的手臂，根据短尾停止条件的思想，当一个手臂的样本数量超过其尾方程的数值，他的样本数量就足够了，所以如果还是对两个手臂同时进行取样的话，我们将浪费很多样本。所以在每一轮的取样过程中，如果某个手臂的样本数量超过其尾方程预测的数值时，我们就停止对该手臂进行取样，这种取样过程称之为需求取样。需求取样对于手臂数量比较大的问题比较有效率，因为对于每一个手臂都能节省取样数目。在极端条件下，比如所有的手臂都是一样的，这样可以节省一半的样本数目。

3.5.5　动态短尾停止条件

在 UCB 类取样的算法的稳定阶段，最大的和第二大的手臂都不会发生变化，但是算法不一定会对第二大的手臂进行取样，因为其置信上界不一定是最大或者第二大的，这种情况下，短尾停止条件可能需要等待更长的时间来让第二大手臂获取足够的样本数目来满足短尾停止条件。为了改进这种情况，我们可以假设如果一个手臂占据第二的位置足够的久（并不表示其样本数目有这么多），那么这个短尾停止条件就满足了，这种改进的短尾停止条件被称为动态短尾停止条件。

定义 3.15 (**第 i 位连续长度**)：　假设整个算法已经进行了 t 次取样，A_i^* 是经验均值第 i 位的手臂，第 i 位连续长度定义为到最大的 c_i 满足从 $t - c_i$ 个取样到第 t 个取样的过程中，均值第 i 大的手臂保持不变。

定义 3.16 (**动态短尾停止条件**)：　设 $t(\Delta)$ 是一个尾方程，c_1 和 c_2 分别是第 1 位和第 2 位连续长度，动态短尾停止条件定义为同时满足 $c_1 \geq t(\Delta_2)$ 和 $c_2 \geq t(\Delta_2)$。

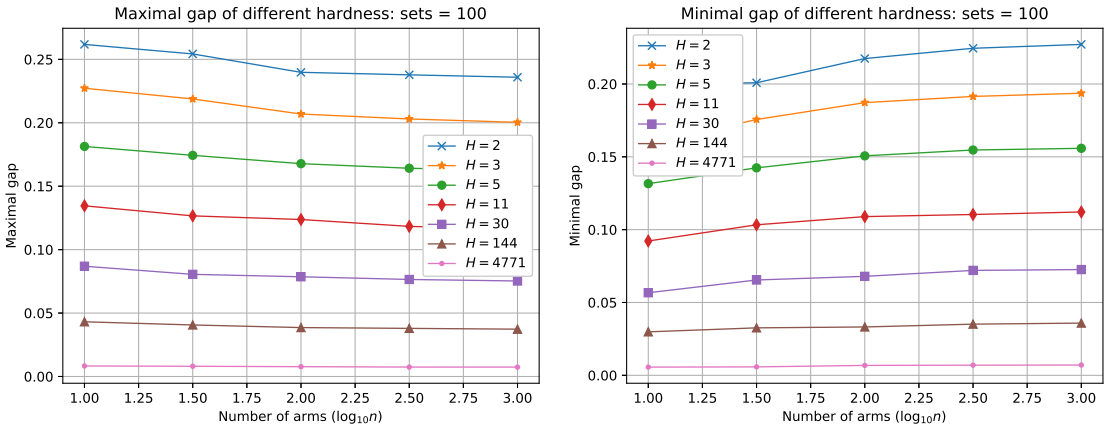
第 4 章　实验

4.1　实验准备

4.1.1　生成随机变量

本次模拟实验是用 python 语言写的，版本 3.6.5，随机数据是由 python 中“numpy” 包的 “numpy.random.RandomState” 类来产生，这个类可以记录随机状态，能够重现相同的随机序列，这样可以保证对于不同的算法都运行在相同的随机序列上。

我们把数据按照难度 (定义 3.2) 进行划分，生成一系列拥有不同大小但平均难度近似的数据集。过程如下：假设平均难度是 h，数据集大小是 n，数据集的总体难度 nh 被平均但随机的分成 $n-1$ 份，然后固定最大的手臂的均值，就能推算出剩余的手臂的均值。

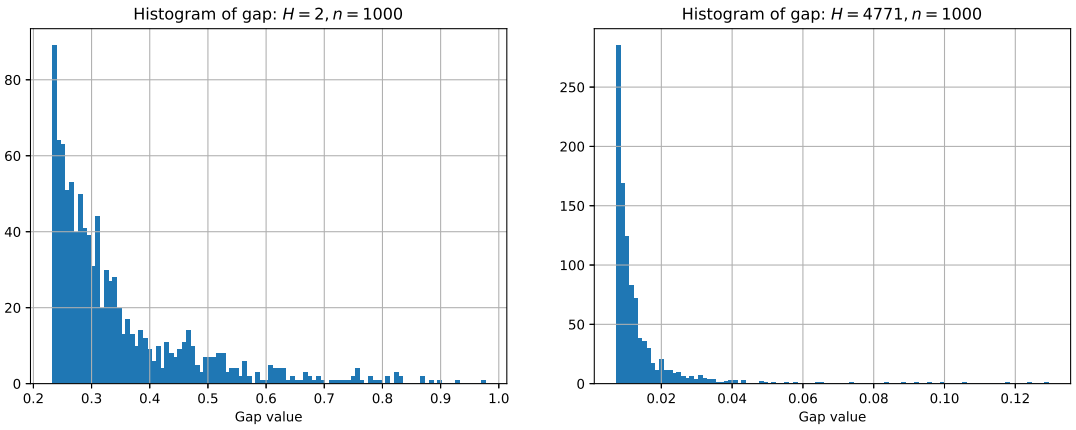


(a) Maximal gap　　(b) Minimal gap

Figure 4.1　对于每个难度和不同大小的数据集，其最大和最小的差值是从 100 个数据集中统计出来的。可以看出，对于不同大小的数据集，其平均难度比较接近。

图 4.1和图 4.2显示，按照上述数据生成的方法产生的数据，其差值都比较稳定：

- 对于固定的难度，不同大小的数据集的差值变化不是很大。参考 图 4.1，每条线几乎都是水平的。
- 对于固定大小的数据集，其不同难度的差值不会相互混淆。参考图 4.1，这些线条都没有相交。



(a) gap = 0.3　　(b) gap = 0.01

Figure 4.2　对于不同难度的数据的差值直方图。可以看出差值都集中在平均难度的位置。

- 对于每一个数据集，其差值分布比较集中。参考图 4.2，差值都集中在平均值附近。

4.1.2　差值方程，尾方程和数据难度

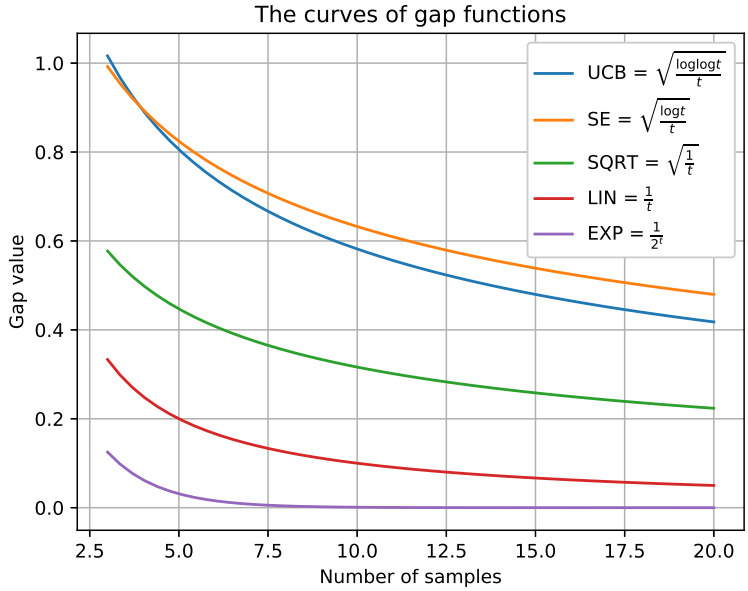


Figure 4.3　差值方程曲线。

图 4.3展示了在模拟实验中用到的差值方程的曲线。可以看出，$\sqrt{t^{-1}\log t}$ 是所示差值方程中最慢的一个，指数差值方程是最快的一个，但其值在样本数量为 10 的时候就已经为 0。

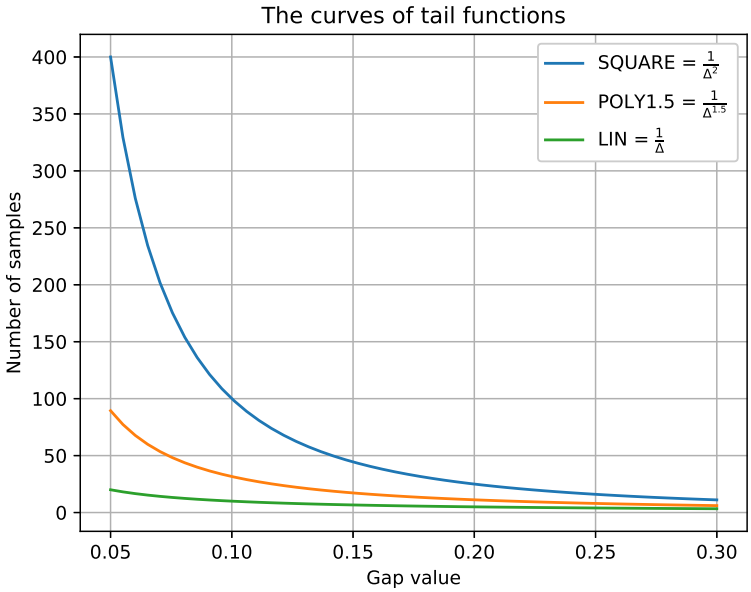


Figure 4.4　尾方程曲线图。

图 4.4展示了在模拟实验中用到的尾方程曲线，即 $\frac{1}{\Delta^x}$ 对于 $x \in [1,2]$。我们会通过实验来找到合适的 x。

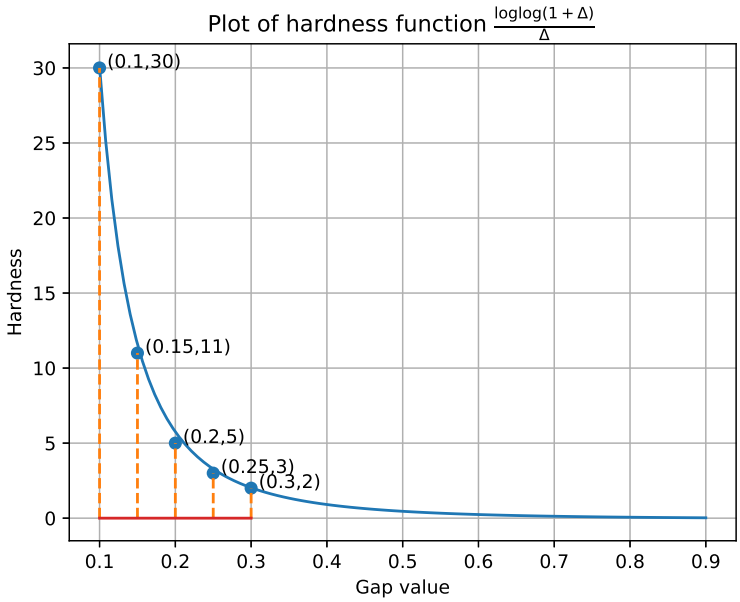
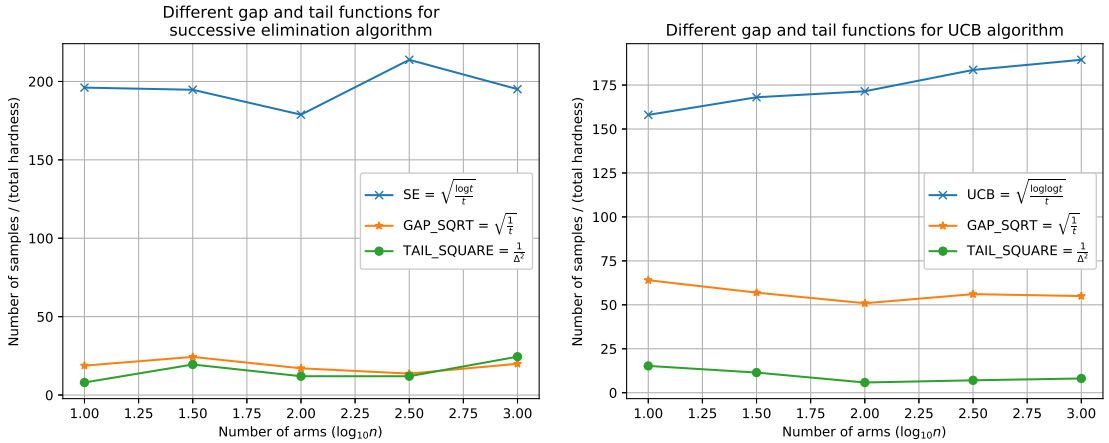


Figure 4.5

图 4.5展示了难度方程的曲线。值的注意的是，我们使用 $\log\log(1+\Delta)$ 而不是 $\log\log\Delta$，因为后者生成的难度值会有负数。当 $H = 4771$，$\Delta = 0.01$；当 $H = 30$，$\Delta = 0.1$；当 $H = 2$，$\Delta = 0.3$。

4.2　测试：差值方程

4.2.1　快速差值方程的效果



(a) 逐次删除算法　　(b) UCB 算法

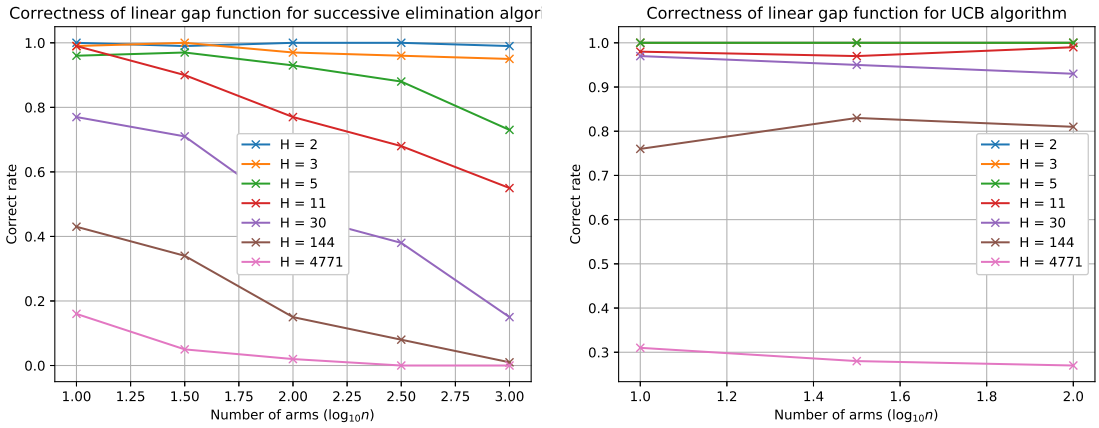
Figure 4.6　差值方程和尾方程的效果：使用平方根差值方程（黄色）和平方尾方程（绿色）都能降低取样次数。（蓝色代表原始的算法）

在第3.2.2中我们讨论了差值方程。粗略的来说，一个算法倾向使用比较“快速”的差值方程，即下降的越快越好，因为快速的差值方程能够降低取样的需求。图 4.6显示了快速差值方程的作用：使用$g_{sqrt}(t)$ 所需求的取样次数比$g_{se}(t)$ 和$g_{ucb}(t)$ 差值方程要少很多（如图所示的，橙色的线条要比蓝色的线条低很多）。

4.2.2　差的差值方程：线性差值方程

一个好的差值方程需要保证算法的正确率，因此它不能下降的太快。从实验可以看出，线性差值方程 $\frac{1}{t}$ 比$g_{sqrt}(t)$ 和$g_{ucb}(t)$ 快，但是使用的结果的正确率很低。图 4.7展示了使用线性差值方程和指数差值方程 $\frac{1}{2^t}$ 的实验的正确率：

- 对于逐次删除算法 (图 4.7(a))，线性差值方程值对难度非常小的数据集有小，对于稍微难一点（$H > 11$）的数据集则完全失效。
- 对于 lil'UCB 算法 (图 4.7(b))，线性差值方程对于大部分的数据都能给出正确结果，只有特别难的数据（$H > 4771$）会是算法失效。
- lil'UCB 算法对线性差值方程的鲁棒性要好于逐次删除算法，因为导致前者失败的数据的难度要高于后者。



(a) 在逐次删除算法中使用线性差值方程的正确率。

(b) 在 UCB 算法中线性差值方程的正确率。可以看出对于容易的数据，线性方程的正确率尚可，对于难的数据则失效。

Figure 4.7　使用线性差值方程的正确率

4.3　提前停止条件

4.3.1　正确率：提前停止条件

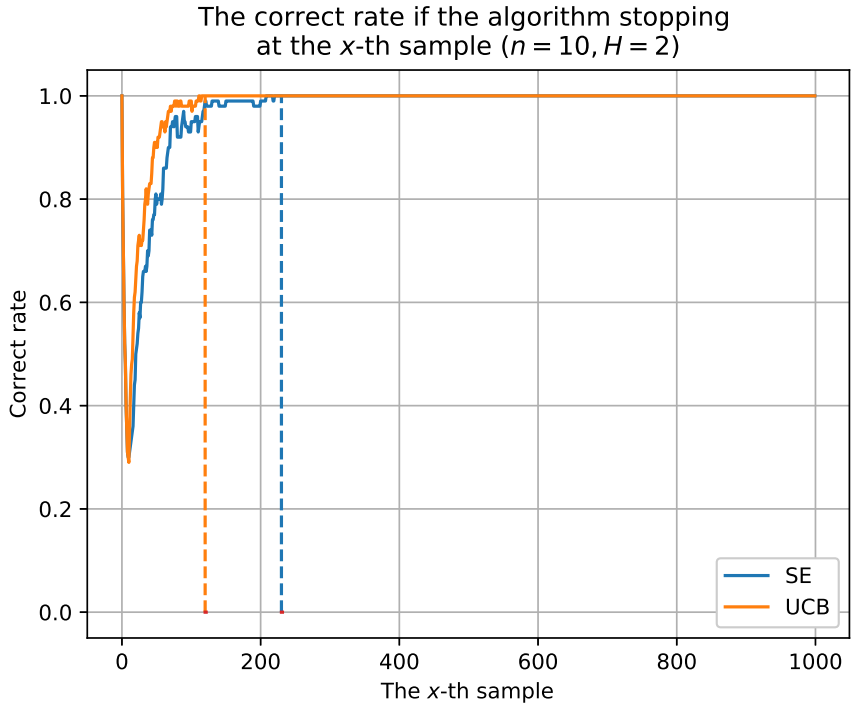


Figure 4.8　当强制算法在第 x 个取样的时候强制停止时的正确率。

图 4.8展示了如果算法在第 x 个取样之后直接停止的正确率。图中可以看出，算法在取样开始就已经能够输出正确的结果，后面的取样次数从这种意义上来说

时没有必要，这个现象促使我们寻找快速的停止条件。图 4.8同时还显示，lil'UCB 算法比逐次删除算法更快进入正确状态，从这个侧面来说前者时更好的算法。

4.3.2　算法的稳定阶段

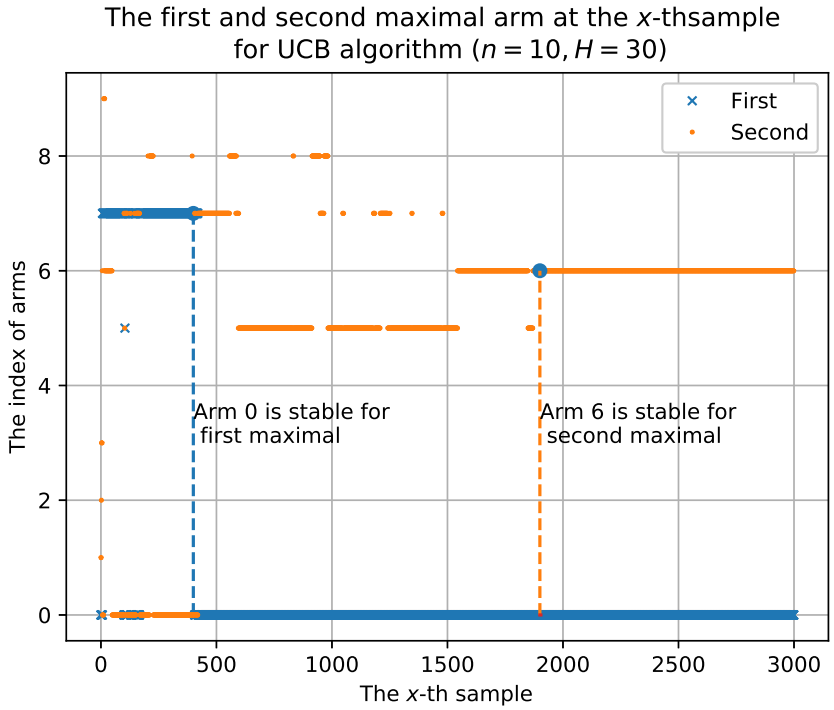


Figure 4.9　当进行第 x 次取样的时候最大和第二大手臂。图中看出，当进行第 400 次取样的时欧，最大的手臂基本保持不变了。当进行第 1900 次取样后，第二大手臂基本保持不变了，这个时候算法进入稳定阶段。

图 4.9展示了算法的稳定阶段。在这个模拟实验中第 1900 次取样年后，最大的两个手臂就保持不变了。如果算法直接停止，那将节省大约一半的样本次数（正常结束需要的样本次数时 3360）。我们用尾方程来估算算法进入稳定阶段的时刻。

图 4.10显示了 UCB 算法取样的详细过程，从某个时刻开始（本实验中是第 700 次取样开始），算法开始只对最大的手臂进行取样，这种情况下，我们可以认为算法已经进入稳定阶段了。

4.4　测试：尾方程

4.4.1　尾方程的作用

和差值方程不同，算法倾向与使用比较慢的尾方程，即尾方程不能随着差值趋近于 0 而增长的太快。慢的尾方程会使算法更有可能停止，因为在相同的差值Δ_i

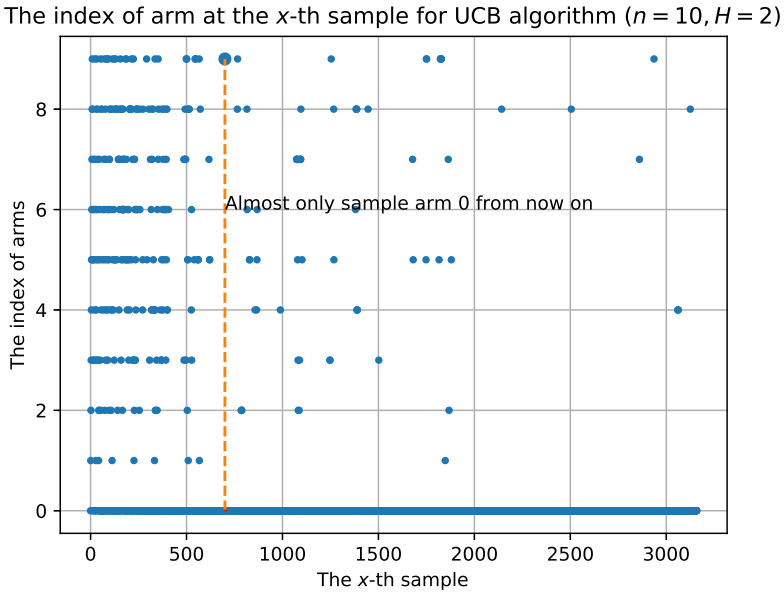
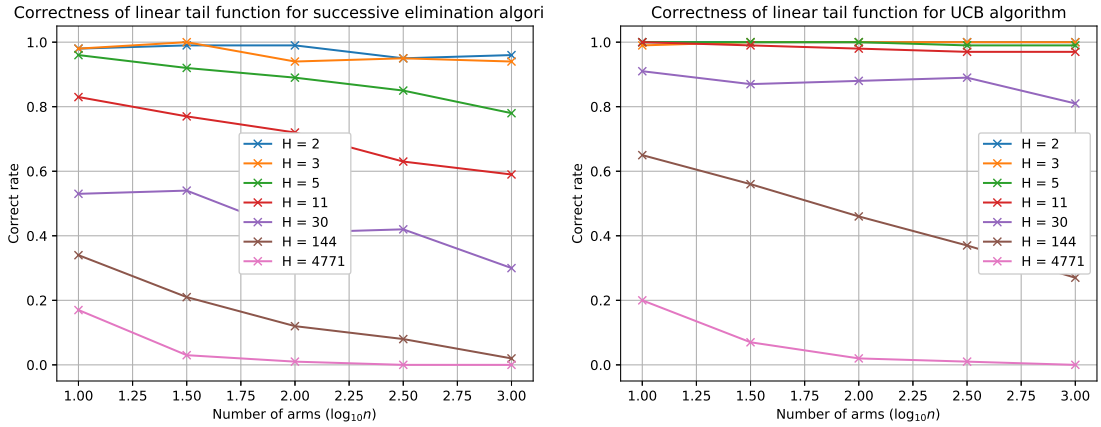


Figure 4.10　当第 x 次取样的时候手臂的下标。从 700 次取样开始，UCB 算法就只对最大的手臂进行取样了，这个时候算法则进入稳定阶段。

下，慢的尾方程的数值比较小，使得停止条件更容易满足。图 4.6展示了使用尾方程的效果：使用平方尾方程 $\frac{1}{\Delta^2}$ 可以减少样本需求（绿色的线比蓝色的线低）。

4.4.2　测试：线性的尾方程



(a) 逐次删除算法　　(b) UCB 算法

Figure 4.11　使用线性尾方程的算法的正确率

一个好的尾方程因该要保证算法的正确率，线性尾方程 $\frac{1}{\Delta}$ 则是一个非常差的尾方程，它使得算法过早的结束从而输出错误的结果。图 4.11展示了使用线性尾方程的正确率，逐次删除算法对于所有的数据都出错，lil'UCB 算法也只在难度很

小的数据输出正确结果。

4.5　合并多种启发式方法

我们将使用 LUCB 算法的框架，每轮都对两个手臂进行取样。

4.5.1　寻找差值方程

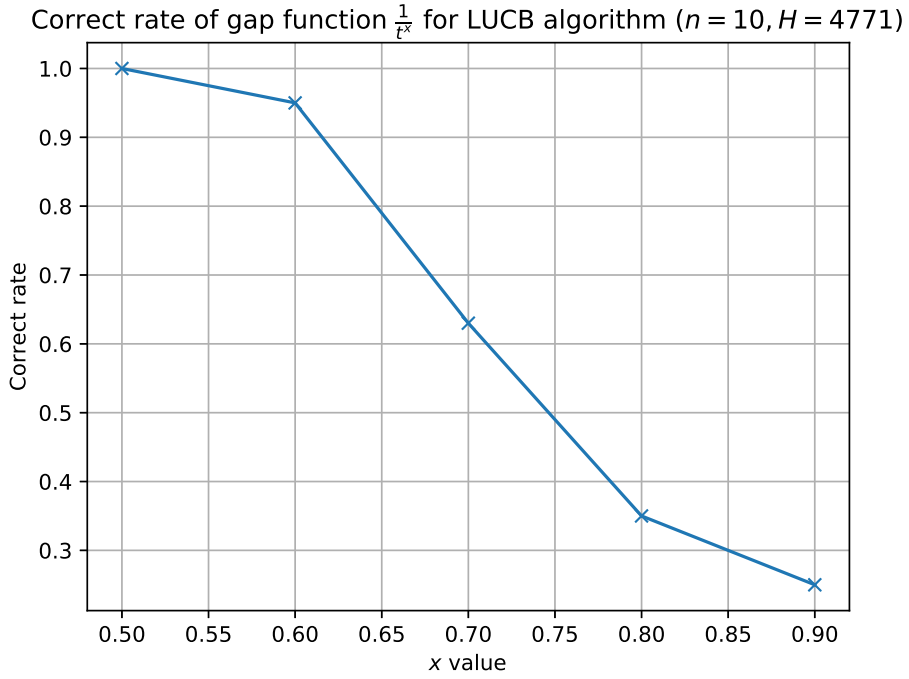


Figure 4.12　不同的多项式差值方程的正确率。

本次实验中我们将使用多项式差值方程 $\frac{1}{t^x}$，但时我们需要确定参数 x 的值。注意到当 $x = 0.5$ 的时候，这个差值方程就是平方根差值方程，这是一个比较好的差值方程；当 $x = 1.0$ 的时候，这个差值方程就是线性差值方程，这是一个很差的差值方程，所以我们可以推断 $x \in [0.5, 1.0]$。图 4.12显示了启发式算法使用不同的多项式差值方程的正确率：当 $x = 0.6$ 时，算法的正确率已经大于 0.9，已经满足要求，但是相对于$g_{sqrt}(t)$ 的 100% 正确率，这个结果不是特别满意，因为随着数据难度的增加，我们相信其正确率会下降。同时和$g_{sqrt}(t)$ 相比，其提升不是特别明显，所以我们最终选择$g_{sqrt}(t)$ 作为我们启发式算法的差值方程。

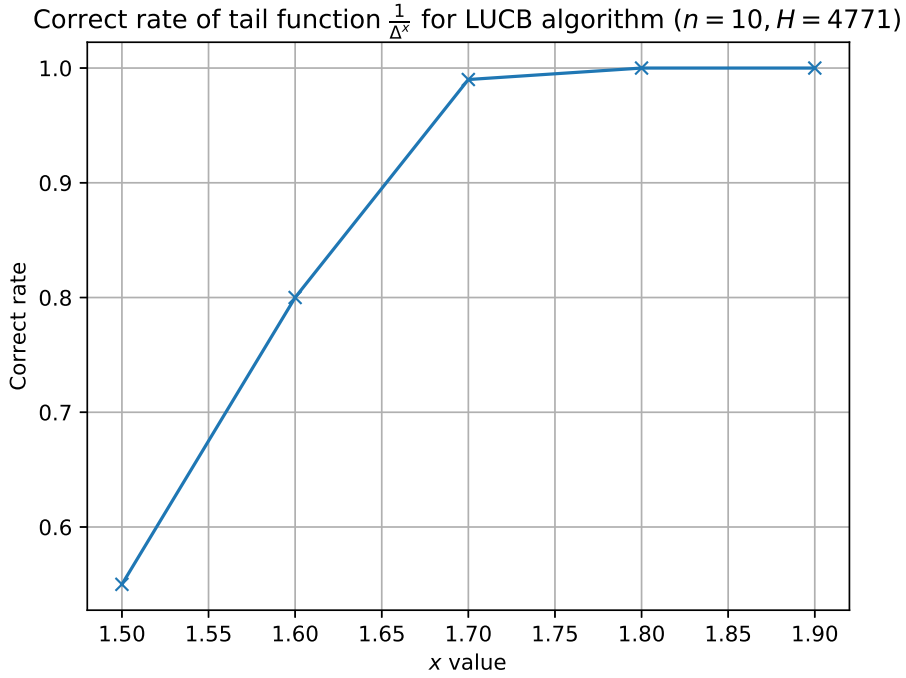


Figure 4.13　使用不同的多项式尾方程的正确率

4.5.2　寻找尾方程

本实验的启发式算法使用多项式尾方程$\frac{1}{\Delta^x}$，但我们必须确定 x 的值。注意到当 $x = 2.0$ 的时候，多项式尾方程就是平方尾方程，这个时一个好的尾方程；当 $x = 1.0$ 的时候，多项式尾方程就是线性尾方程，则是一个比较差的尾方程，所以 $x \in [1.5, 2.0]$。图 4.13展示了使用不同的多项式尾方程的正确率：当 $x = 1.8$ 的时候，算法已经不会输出错误的结果。但是算法是运行在一个很小的数据集上（$n = 10$），同时最难的难度时 $H = 4771$，对于更大更难的数据集需要很大的运算量，本次试验中比较难完成。尽管没有实际数据支持，我们还是相信对于 $x < 2.0$ 的多项式尾方程都不是好的尾方程，所以我们最终选择平方尾方程 t_{square} 作为最终的启发式算法的尾方程。

4.5.3　需求取样

需求取样改进了 LUCB 算法的取样方法并且获得比较好的效果。图 4.14中显示了使用需求取样的效果（橙色的线条）。

4.5.4　动态短尾停止条件

动态短尾停止条件也对算法进行了改进。图 4.14显示了使用动态短尾停止条件的结果（绿色的线条）。

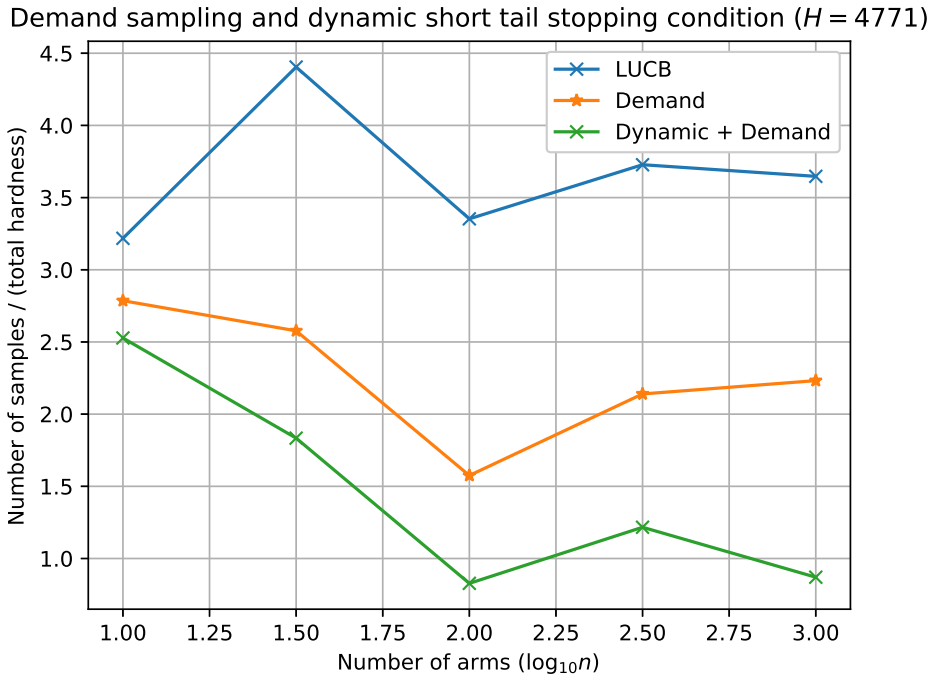


Figure 4.14　使用需求取样和动态短尾停止条件的算法的取样次数。在本次测试中，LUCB 算法使用了$g_{sqrt}(t)$ 差值方程和平方尾方程。

4.6　测试最终的启发式算法

4.6.1　最终的启发式算法

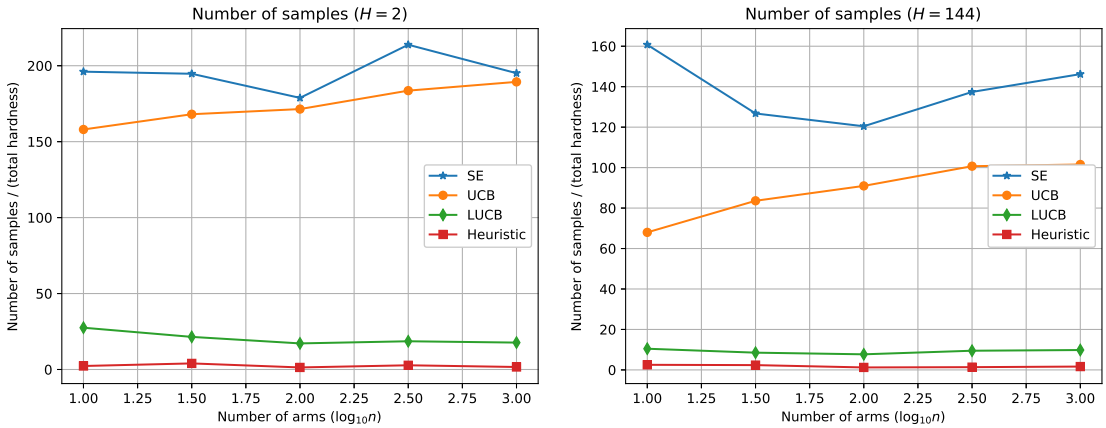
算法 6总结了最终的启发式算法，其中粗体字标出所采用的启发式方法。这个算法用在接下来的测试中。

4.6.2　和已知算法比较

本次时把最终的启发式算法和逐次删除算法，lil'UCB 算法和 LUCB 算法进行比较。图 4.15显示了在不同难度的数据集上的的模拟结果。图 4.16显示了在完全随机的数据集上的模拟实验结果。在两种不同的数据集中，我们的启发式算法都使用最少的取样次数。

Algorithm 6 启发式算法

设 $g(t) = \sqrt{\frac{1}{t}}$　(**平方根差值方程**) $t(\Delta) = \frac{1}{\Delta^2}$　(**平方尾方程**)
对每个手臂进行一次取样。
设 $\hat{\mu}_i$ 是手臂的经验均值。
设 $A_{first} = \arg\max_A \hat{\mu}_i$ 和 $A_{second} = \arg\max_{A/A_{first}} \hat{\mu}_i$　(**最大两个手臂**)
设 t_i 为手臂A_i 被取样的次数。
设 $c_{first} = 0, i_{first} = A_{first}, c_{second} = 0, i_{second} = A_{second}$
设 $\hat{\Delta}_i = \hat{\mu}_{first} - \hat{\mu}_i$　(**经验差值**)
设 $i^* = \arg\max_{A/A_{first}}(\hat{\mu}_i + g(t_i))$　(**最大的置信上界手臂**)
while ($t_{first} \leq t(\hat{\Delta}_{second})$ and $t_{second} \leq t(\hat{\Delta}_{second})$)　(**短尾停止条件**)
or ($c_{first} \leq t(\hat{\Delta}_{second})$ and $c_{second} \leq t(\hat{\Delta}_{second})$)　(**动态短尾停止条件**)
or $\hat{\mu}_{first} - g(t_{first}) > \hat{\mu}_{i^*} + g(t_{i^*})$　(**不相交圆停止条件**) **do**
　对手臂 i^* 进行一次取样　(**UCB 取样**)
　if $t_{first} \leq t(\hat{\Delta}_{second})$ **then**
　　对手臂A_{first} 进行一次取样 (**需求取样**)
　end if
　更新 $t_i, A_{first}, A_{second}, \hat{\Delta}_i, i^*$
　if $i_{first} == A_{first}$ **then**
　　$c_{first} = c_{first} + 1$　(**第 1 位连续长度**)
　else
　　$c_{frist} = 0, i_{first} = A_{first}$
　end if
　if $i_{second} == A_{second}$ **then**
　　$c_{second} = c_{second} + 1$　(**第 2 位连续长度**)
　else
　　$c_{second} = 0, i_{second} = A_{second}$
　end if
end while
输出 $\arg\max_A A_i$



(a) $H = 2$　　(b) $H = 144$

Figure 4.15　固定难度的数据下启发式算法所需的取样次数。

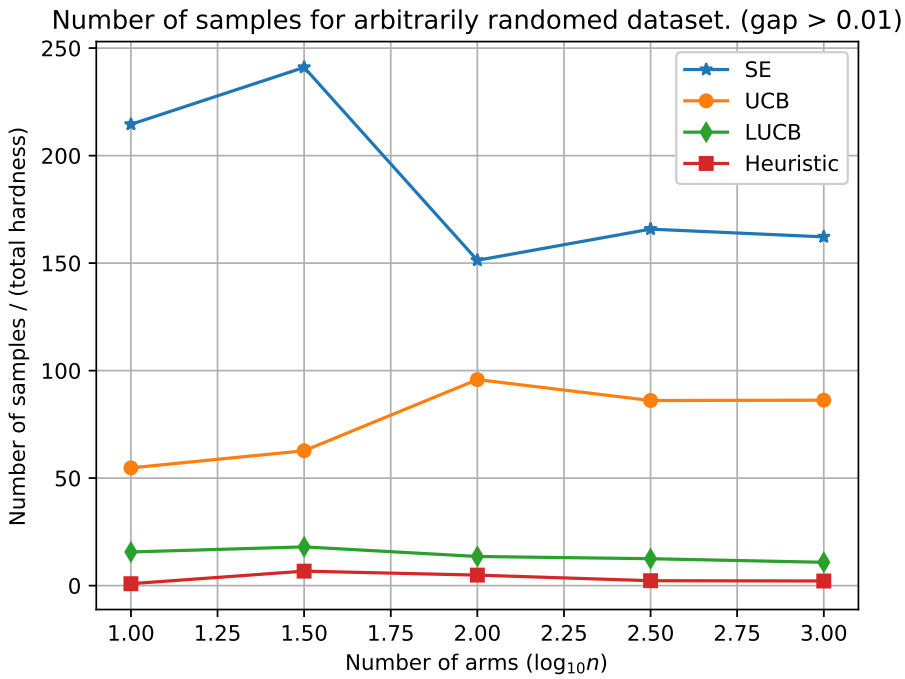


Figure 4.16　随机数据集中启发式算法所需的取样次数。

第 5 章　总结

本文研究了最有臂选择问题的经典算法逐次删除算法[13] 和 lil'UCB 算法[16] 并且提炼出算法的关键部分，并对这些部分进行启发式的改进：差值方程，短尾停止条件和需求取样。然后设计了一系列的模拟实验来测试这些启发式的改进的效果，最终选择平方根差值方程和平方尾方程。最终的实验结果显示，我们的启发式算法有最好的运行结果。

这里必须强调的时，所有的启发式步骤都是根据模拟实验结果的，但时受限于机器的运算能力，我们无法对极端的数据集进行测试，比如极小的差值和极多的手臂，在这些极端情况下，实际结果可能会和我们目前的实验结果不一样。

Bibliography

[1] Nicolò C B, Gábor L. Prediction, learning, and games[M]. [S.l.]: Cambridge university press, 2006

[2] Sébastien B, Nicolò C B. Regret analysis of stochastic and nonstochastic multi-armed bandit problems[J]. Foundations and Trends® in Machine Learning, 2012, 5(1): 1–122.

[3] Herbert R. Some aspects of the sequential design of experiments[J]. Bulletin of the American Mathematical Society, 1952, 58(5): 527–535.

[4] Jean-Yves A, Sébastien B. Best arm identification in multi-armed bandits[J]. 23th Conference on Learning Theory, 2010: 13 p.

[5] Deepayan C, Ravi K, Filip R, et al. Mortal multi-armed bandits[J]. Advances in Neural Information Processing Systems 21, 2009: 273–280.

[6] Shouyuan C, Tian L, Irwin K, et al. Combinatorial pure exploration of multi-armed bandits[J]. Advances in Neural Information Processing Systems 27, 2014: 379–387.

[7] Yuan Z, Xi C, Jian L. Optimal PAC multiple arm identification with applications to crowdsourcing [J]. Proceedings of the 31st International Conference on Machine Learning, 2014, 32(2): 217–225.

[8] Wei C, Jian L, Yufei T, et al. On top-k selection in multi-armed bandits and hidden bipartite graphs[J]. Advances in Neural Information Processing Systems 28, 2015: 1036–1044.

[9] Weiwei S, Jun W, Yu-Gang J, et al. Portfolio choices with orthogonal bandit learning[J]. Proceedings of the 24th International Conference on Artificial Intelligence, 2015: 974–980.

[10] Robert E. B, Milton S. A single-sample multiple decision procedure for ranking variances of normal populations[J]. The Annals of Mathematical Statistics, 1954, 25(2): 273–289.

[11] Edward P. A sequential procedure for selecting the population with the largest mean from k normal populations[J]. The Annals of Mathematical Statistics, 1964, 35(1): 174–180.

[12] Robert E. B, Jack K, Milton S. Sequential identification and ranking procedures, with special reference to koopman-darmois populations[M]. [S.l.]: University of Chicago Press Chicago, 1968: xvii, 420 p.

[13] Shie M, John N. T. The sample complexity of exploration in the multi-armed bandit problem[J]. The Journal of Machine Learning Research, 2004, 5: 623–648.

[14] Eyal E D, Shie M, Yishay M. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems[J]. Journal of Machine Learning Research, 2006, 7: 1079–1105.

[15] Zohar K, Tomer K, Oren S. Almost optimal exploration in multi-armed bandits[J]. Proceedings of the 30th Internatinal Conference on International Conference on Machine Learning, 2013, 28: 1238–1246.

[16] Kevin J, Matthew M, Robert N, et al. lil' UCB: An optimal exploration algorithm for multi-armed bandits[J]. Proceedings of The 27th Conference on Learning Theory, 2014, 35: 423–439.

[17] Lijie C, Jian L. On the optimal sample complexity for best arm identification[J]. arXiv preprint arXiv:1511.03774, 2015.

[18] Alexandra C, Andrea L. Tight (lower) bounds for the fixed budget best arm identification bandit problem[J]. Conference of Learning Theory, 2016.

[19] Aurélien G, Emilie K. Optimal best arm identification with fixed confidence[J]. 29th Annual Conference on Learning Theory, 2016, 49.

[20] Lijie C, Jian L. Open problem: Best arm identification: Almost instance-wise optimality and the gap entropy conjecture[J]. 29th Annual Conference of Learning Theory, 2016.

[21] Lijie C, Anupam G, Jian L. Pure exploration of multi-armed bandit under matroid constraints[J]. 29th Annual Conference of Learning Theory, 2016.

[22] Lijie C, Jian L, Mingda Q. Towards instance optimal bounds for best arm identification[J]. Proceedings of the 2017 Conference on Learning Theory, 2017, 65: 535–592.

[23] Lijie C, Jian L, Mingda Q. Nearly instance optimal sample complexity bounds for top-k arm selection[J]. Artificial Intelligence and Statistics, 2017: 647–669.

[24] Shivaram K, Peter S. Efficient selection of multiple bandit arms: Theory and practice[J]. Proceedings of the 27th International Conference on Machine Learning, 2010.

[25] Victor G, Mohammad G, Alessandro L, et al. Multi-bandit best arm identification[J]. Advances in Neural Information Processing Systems 24, 2011: 2222–2230.

[26] Victor G, Mohammad G, Alessandro L. Best arm identification: A unified approach to fixed budget and fixed confidence[J]. Proceedings of the 25th International Conference on Neural Information Processing Systems, 2012, 2: 3212–3220.

[27] Shivaram K, Ambuj T, Peter A, et al. PAC subset selection in stochastic multi-armed bandits[J]. Proceedings of the 29th International Conference on Machine Learning, 2012: 655–662.

[28] Sébastien B, Tengyao W, Nitin V. Multiple identifications in multi-armed bandits[J]. Proceedings of the 30th International Conference on International Conference on Machine Learning, 2013, 28: I–258–I–265.

[29] Emilie K, Shivaram K. Information complexity in bandit subset selection[J]. Journal of Machine Learning Research, 2013, 30: 228–251.

[30] Emilie K, Olivier C, Aurélien G. On the complexity of best-arm identification in multi-armed bandit models[J]. The Journal of Machine Learning Research, 2016, 17(1): 1–42.

[31] Victor G, Alessandro L, Mohammad G, et al. Improved learning complexity in combinatorial pure exploration bandits[J]. Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, 2016, 51: 1004–1012.

[32] Roger H. F. Asymptotic behavior of expected sample size in certain one sided tests[J]. The Annals of Mathematical Statistics, 1964, 35(1): 36–72.

[33] Eyal E D, Shie M, Yishay M. PAC bounds for multi-armed bandit and markov decision processes [J]. Proceedings of the 15th Annual Conference on Computational Learning Theory, 2002: 255–270.