清 华 大 学

综 合 论 文 训 练

题目：纯探寻组合多选择赌博机的近似算法与不可近似性

系　　别：交叉信息研究院

专　　业：计算机科学与技术

姓　　名：谭子涵

指导教师：李建助理教授

2015年 6月

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：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签　名：_____ 导　师　签　名：_____ 日　　期：_____

中文摘要

随着大数据时代的到来，信息处理的速度与量级需求都有本质的提高，因此在线优化方面涌现出很多有代表性并且颇具难度的问题。多选择赌博机便是其中的重要问题，求解这个问题代表着玩家要在学习未知的分布的同时操纵最优的赌博机选择（为了提升自己的收益），在两者之间达到一个平衡。这一里程碑性质的问题引来了大量研究与变形，应用甚为广泛。

组合多选择赌博机问题，是将组合问题融入赌博机问题的重要框架。玩家每一次不是简单地尝试一个选择而是要尝试一个选择组合，使得这个选择在满足某种组合条件的情况下，根据他已有的知识最大化他的收益。也就是说，玩家每做一次选择，就是解一个组合问题的过程，然而参数事先给定但玩家并不了解，而是要在不断选择并得到新的知识后增加对参数的了解从而最优化后面的选择。

本文中，我们研究组合多选择赌博机的“纯探寻”问题，即我们并不要求优化总收益，而是把目标定在输出最优的选择组合上。这个目标离不开对于相应的组合优化问题的求解，同时对于不同难度的优化问题我们能够输出选择组合的最优性也有着不同的要求。我们研究输出理论可能的最优组合的近似算法和不可近似性。应用的技巧是概率论的基本知识以及算法设计与分析。

关键词：赌博机；在线优化；近似算法；

ABSTRACT

With the rising of big data, our requirement for speed and volume of data processing has increased extensively. Online learning, as a representative class of problems, has captured much attention in theoretical analysis. Multi-armed bandit is an important problem in online learning and has been well-studied in recent years. It needs the player to maintain a balance between exploration and exploitation.

Combinatorial multi-armed bandit is a combination of combinatorial optimization and multi-armed bandit, in which the player tries a combination of arms (called a super arm) rather than one base arm as usual, so that his reward is optimized based on his previous knowledge on arms. In each round the player is indeed solving a combinatorial optimization problem, where the parameter is not given in the beginning, but needs to be learned by the player from previous rounds.

In this thesis, we studied the pure exploration problem, whose goal is not maximizing the accumulative reward or minimizing the regret, but maximizing the probability that the algorithm outputs the ground-truth optimal solution. For different combinatorial problems we can have different objectives towards them. We focused on designing approximation algorithms and proving hardness results of the problem. Techniques including basic probability theory and algorithm design are utilized.

Key words: Multi-armed bandit； online optimization； approximation algorithm

目 录

第 1 章　引言

随着大数据时代的到来，当今人们对于数据传输与处理的通量与速度的要求也在日渐提高。大数据目前的“3V”描述就生动形象的刻画了这一特点：velocity, volume, variety。速度，容量与多样化让当今的生活呈现出不同的形态。当我们把目光放在速度之上，就会在当今的算法及组合优化问题的研究中发现，越来越多的理论研究正在倾向于数据流算法，在线算法，对数空间算法或者是次线性复杂度算法。即我们不能够再像以前一样，面对一个问题设计算法把输入来回来去的用来计算与衡量，我们的输入量很大，大到我们的内存存储不下因而我们只能够有一次读取它们的机会。而这，便是“在线优化”的本质特征：我们需要在得到数据之后立即做出选择，而不是在所有数据都得到之后全面统筹做出选择。在这种情况下我们势必会损失一些收益，但我们可以从之前观察到的输入数据找出规律做出分析，从而帮助我们后面更好的做出选择。这便是微软高级研究员陈卫老师在[1] 中提到的经典的探寻-使用平衡（exploration and exploitation）。

1.1　多选择赌博机问题

在线优化方面有很多近年来兴起的问题，多选择赌博机是其中一个极具代表性的问题。这个问题在[3] 中被首先提出，具体来说，问题的设定如下：

我们面前拥有多个游戏臂（arm，在下文中我们一般称其为“选择”，即多选择赌博机中的选择）的一台赌博机(bandit),每一个选择的背后有一个固定的概率分布（在大多数的的研究论文中，我们可以正则化这个分布到[0, 1]上），但是玩家一开始并不知道这个分布的具体信息。玩家每一轮可以做出一个选择，然后得到一个从所选的选择背后的概率分布取样出来的值作为玩家这一轮的收益。玩家看得到每一轮他获得了多少收益，然后基于他之前做出的选择和获得的收益，在下一轮中做出一个新的选择，获得一个新的收益，依此类推。游戏总共进行T轮(这个T是事先设定好的，并告知玩家)，玩家的目标就是尽可能最大化他的收益值。或者从另一个角度来说，玩家在尽可能减小他的损失值。

为准确的叙述问题，我们引入下面的符号。设赌博机总共有n种选择(即n个

游戏臂)，第i种选择背后的概率分布用ϕ_i来表示，这里我们不妨设概率分布是定义在$[0,1]$上的。我们用μ_i代表概率分布ϕ_i的期望值，并用μ^*代表其中的最大值。游戏总共进行T轮，我们用$t_i(t)$代表第i个选择在前t轮中总共被选择的次数。游戏的进行规则如前一段所述。于是，一个策略便是根据前面的轮次观察到的信息，在新的一轮中做出选择。它的目标函数可以被定义为下面的“损失函数”regret:

$$\text{regret} = T\mu^* - \sum_{j=1}^{n} \mu_j \mathbb{E}[t_j(T)]$$

式中的数学期望项$\mathbb{E}[t_j(T)]$，其随机性来自每一轮取样的随机性以及策略中自带的随机性。

文章[3]中同时也提出了第一个解决此类问题的算法，现在被人们称为UCB算法(Upper Confidence Bound)。这个算法简单并且效果很好，实现容易并且分析简单，也引来后续对于多选择赌博机问题的变式的研究者们的竞相模仿。UCB算法事实上也是一个index算法，即给予之前的选择和收益值对于每一个选择计算一个特征数值，新的一轮选择就直接根据这些特征数值来做出（事实上是选取特征数值中的最大者），然后根据观察结果再来更新一些选择的特定数值，然后接着做出后续轮次的选择。index算法的特征就是简单容易施行，而一般来说，这些特征数值也会被赋予一些含义，使得算法变得更加直观而容易理解。在这个具体问题中，Upper Confidence Index被如下定义：

$$\text{UCI}(j,t) = \bar{x}_j + \sqrt{\frac{2\log t}{t_j(t)}}$$

其中$\bar{x}_j$代表之前的$t_j(t)$个取样值的平均值，而$\text{UCI}(j,t)$就是当游戏进行了t轮之后第j个选择的index值。

根据这个算法，经过正常的概率不等式分析，文章[3]证明了，损失函数的期望值有如下的上限：

$$\mathbb{E}[\text{regret}] \le O\left(\log T \sum_{\{i:\mu_i<\mu^*\}} \frac{1}{(\mu^* - \mu_i)^2}\right)$$

我们可以看到，损失函数是和$\sum_{\{i:\mu_i<\mu^*\}} \frac{1}{(\mu^*-\mu_i)^2}$成正比的。我们通过观察可以看出：如果我们提升其中一个μ_i一点点但不使它达到μ^*，那么这个$\frac{1}{(\mu^*-\mu_i)^2}$会增大，这貌似有些不符合直觉，因为看似我们将一个选择变得更好但是却最终

让regret值变大了。事实上却并不是这样，一方面，我们并不真正的知道算法的regret值在这一变动之后是增加了还是减小了，我们知道的只是一个上界变大了。另一方面，我们也可以思考为什么会发生上界变大的情况，这是因为μ_i增大了，那么算法通过取样，将其与μ^*区分开来就需要做更多轮数的取样，因此更多轮数的非最优选择就会带来更多的损失。

另外我们需要强调的一点便是，**这个损失值上界是problem-dependent的，因为很明显它取决于问题的真实参数(μ_i)。而这种问题参数依赖性事实上是不可避免的**，我们会在后文我们做出的不可近似性证明中看出，不存在复杂性与问题参数独立的算法。但从另一个方面来说，问题参数依赖性也可以稍定性的来分析依赖严重性。我们希望设计出依赖程度较轻的算法而不是依赖程度重的算法。

1.2 组合多选择赌博机问题

对于多选择赌博机的大量研究也催生出了这个问题的多个变型，其中一个重要的变型，也可以被称为新的问题范式便是陈卫在[2] 中提出的组合多项式赌博机问题。这个问题中，玩家每一轮并不是简单地选择一个游戏臂，而是可以选择一组游戏臂，获得的收益也是从这些游戏臂背后的分布中取样出来的收益值之和。但是需要满足的一个条件是，玩家选择的游戏臂组合必须满足事先规定的某种组合约束，这个约束可以被刻画成为$\mathcal{M} \subseteq 2^{[n]}$，即集合$[n] = \{1, 2, \cdots, n\}$的一个子集族。如所叙述的规则，玩家在做出这个组合选择之后，可以看到这个组合中的所有选择根据他们背后的概率分布取样出来的值，而玩家的目标依然是优化事先规定的T轮之后的总收益。

从这个问题的定义可以看出，多选择赌博机已经不再仅仅是一个单独的问题，而是可以与其他问题相联系，产生新问题的一个框架。具体来说，如果我们考虑另外一个问题，我们便可以考虑它的“多选择赌博机累积优化”版本，即，问题原本作为输入的参数现在并不作为输入，而是满足一个未知的分布，我们需要不断地学习这个分布，并且在多轮优化中尽量最大化总收益。多选择赌博机已然为其他很多问题提出了一种新的范式。

将组合问题融入多选择赌博机，实际上是一种大成化研究方式。我们对于组合问题的了解有可能是：

（1）输入具体的参数，我们可以有效的(多项式时间内)精确求解这个组合

问题

（2）输入具体的参数，我们可以有效的(多项式时间内)近似求解这个组合问题

对于第二种情况，对于一个玩家他的目标函数也不能够仅仅设定为之前的regret函数，因为最优的有效算法也没有办法找到最优解。考虑到这一点，文章[2]中提出了一种新的regret函数的定义，即(α,β)-regret，粗略来说，这个定义指的是当一个组合问题的确定性版本能够被一个算法以大于$(1-\beta)$的概率输出α-近似解的时候，在多选择赌博机版本中，我们只能将累计收益和理论最优的累计收益的$\alpha\beta$倍相比较。

文章[2]中也提出了一种类似于UCB的算法，称为CUCB（Combinatorial Multi-Armed Bandit）算法，这个算法是基于一个线下解组合问题的黑箱，每一步通过对黑箱输入参数，考虑得到其输出解来做下一轮的选择。而输入参数的表达式也和UCB算法的表达式十分相近，如下：

$$\mathrm{CUCI}(i,t)=\bar{x}_i+\sqrt{\frac{3\log t}{2t_i(t)}}$$

其中$\bar{x}_j$代表之前的$t_i(t)$个取样值的平均值，而$\mathrm{CUCI}(i,t)$就是当游戏进行了t轮之后第i个选择的index值。

通过类似的理论分析，最终文章[2]证明了CUCB算法的期望(α,β)-regret值的一个上界如下：

$$\mathbb{E}[\mathrm{regret}]\leq O\left(\log T\sum_{i=1}^{n}\frac{1}{\Delta_i^2}\right)$$

其中Δ_i代表类似于某种”第i种选择和最优选择的差值”的定义。

1.3 纯探寻式组合多选择赌博机问题

对于多选择赌博机问题，我们可以对于目标函数做一些改变而使得问题的性质发生改变。比如，我们可以不再要求玩家优化最终的累积收益，而是要求玩家在最后一轮结束之后输出一个他认为最优的选择，玩家要做的就是之前调整取样策略，使得最终他输出的选择确实是最优选择的概率最大化。这种问题我们可以把它称为“纯探寻式”问题。

陈卫等研究者在文章[1]中提出了组合多选择赌博机的问题的纯探寻式版本(Combinatorial Pure Exploration of Multi-Armed Bandit)。具体来说，每一轮玩家只被允许尝试一种选择而不再是一个组合选择，之后他会观察到这个新选择背后的概率分布中的一个取样。他根据已经观察到的取样值去做下一轮选择，这中间可以使用一个线下的黑箱用以帮助决策,在游戏停止的时候，玩家需要输出一个满足某种组合约束的选择组合$S \in \mathcal{M} \subset 2^{[n]}$。在文章[1]中，作者假设我们拥有一个可以线下对应于输入的参数能够找出组合问题最优解的黑箱，从而设计出一个纯探寻式算法CLUCB(Combinatorial Lower-Upper Confidence Bound)算法。这个算法首先将当前观察到的各种选择的平均值输入到黑箱中，得到该参数下的最优解，然后根据这个输出调整参数再次输入到黑箱之中得到另外一个最优解，将这两个最优解进行比对，如果相同则停机输出，如果不同则继续取样直到相同为止。经过理论分析，作者证明了：CLUCB的期望停机时间有如下的上界：

$$\mathbb{E}[T] \leq O\left(\text{width}^2(\mathcal{M}) \sum_{i=1}^{n} \frac{1}{\Delta_i^2}\right)$$

其中Δ_i代表类似于某种“第i种选择和最优选择的差值”的定义,而width($\mathcal{M}$)是组合约束$\mathcal{M}$的某个本质性质参数，它的一个自然上界是n。

1.4 例子

在引言的最后，我们用一个例子来说明之前介绍的三种形式的问题的区别。

假设我们有一个无向图$G = (V, E)$，其中V代表顶点集合而E代表边的集合，我们下面分别叙述多种不同的问题如下。

•组合问题：寻找最大权的无圈图

图的每条边被赋予一个权值w_e,我们希望找到一个边集$E' \subset E$,使得这个边集中不含圈，同时$\sum_{e \in E'} w_e$最大化。

•多选择赌博机问题：选边优化累计收益

图的每条边背后有一个定义在[0, 1]上的收益的概率分布ϕ_e。玩家需要在每一轮选择一条边，得到这条边背后的概率分布的一个取样值作为此轮的收益，

游戏总共进行T轮，玩家希望设计算法最大化T轮之后的总收益，或者成为损失函数，定义如下

$$\text{regret} = T\mu^* - \sum_{e\in E}\mu_e\mathbb{E}[t_e(T)]$$

其中μ_e代表边e的背后的概率分布的期望，μ^*代表其中的最大值，$t_e(t)$代表边e在前t轮中总共被选择的次数，式中的数学期望项$\mathbb{E}[t_e(T)]$，其随机性来自每一轮取样的随机性以及策略中自带的随机性。

•组合多选择赌博机问题：选无圈图优化累计收益

图的每条边背后有一个定义在$[0,1]$上的收益的概率分布ϕ_e。玩家需要在每一轮选择一个物权的边集，得到这个边集的所有变边背后的概率分布的一个取样值，并以他们的和作为此轮的收益游戏总共进行T轮，玩家希望设计算法最大化T轮之后的总收益，或者称为损失函数，定义类似于前面的表达式。

•纯探寻式组合多选择赌博机问题：纯探寻优化最终输出最优无圈图的概率

图的每条边背后有一个定义在$[0,1]$上的收益的概率分布ϕ_e。玩家需要在每一轮选择一条边，得到这条边背后的概率分布的一个取样值，游戏总共进行T轮，玩家在游戏结束时需要输出一个无圈图(边集)，希望输出的边集就是让$\sum_{e\in E'}\mu_e$最大化的边集,即玩家希望最大化。

$$\Pr\left[\text{output} = \arg\max_{E'\subset E;E'\text{无圈}}\{\sum_{e\in E'}\mu_e\}\right]$$

其中上式的随机性来自每一轮取样的随机性以及策略中自带的随机性。

我们可以看出，上面几个问题是同一个本原问题的不同版本，所关心的优化目标也各有不同。对于多选择赌博机问题，它对于图中不同的边没有加以区分，也就是说图中的所有边在玩家的眼中是完全平等的，图结构并没有体现出它的特征。然而当我们把组合问题加入这个多选择赌博机问题之中之后，图的特性就产生了作用。玩家每一轮不能随意的选择任意游戏臂组合，这些组合必须要满足一定的组合约束。事实上，在我们所举的例子之中，组合问题是可以

有效的求最优解的。因此对于这个问题，我们是可以使用[1]中的转化方法，设计出相应的CLUCB算法来解决其纯探寻组合多选择赌博机问题。

1.5 本文研究的主要问题

在本文中，我们继续文章[1]的脚步，主要研究纯探寻组合多选择赌博机问题。在文章[1]中作者研究了当组合问题可以被有效地求出最优解的时候，纯探寻问题的算法应该如何的设计。然而事实上，在实际情况中我们遇到的组合问题大多数是NP-hard的，一般被认为没有办法在多项式时间内精确求解。退而求其次，这些问题可能会有多项式时间内的近似解法。于是对于这类问题，我们研究他们的纯探寻问题，目标便是设计出近似算法使得最终玩家输出的选择组合以大概率也是近似最优的。

第 2 章　纯探寻组合多选择赌博机问题的近似算法

在本节中，我们主要论述关于纯探寻组合多选择赌博机的近似算法及其复杂性分析。首先，我们先定义一些符号如下。令n为选择的个数，对于每一个选择，其背后的收益概率分布的期望是μ_i，而第t轮完成过后，第i个选择总共被取样了$t_i(t)$次，于是我们定义这$t_i(t)$个值的平均值为$\bar{\mu}_i^t$，也称为经验平均值。组合约束我们用$\mathcal{M} \subset 2^{[n]}$来表示，即，玩家的输出一定只能是$\mathcal{M}$中的一个元素。一般的$\mathcal{M}$可以是：图中的所有无圈子图，或者是二分图中的一个匹配，等等。除此之外，我们定义向量$\mu = (\mu_i)_{i=1}^n$ 和$\bar{\mu}^t = (\bar{\mu}_i^t)_{i=1}^n$，并且不加区分的使用$M$ 来代表一个$[n]$ 的子集或者是这个子集的指示向量。具体来说，如果$M = \{1,3,4,6\} \subset [n]$, 那么向量$M = \langle 1,0,1,1,0,1,0,\cdots,0\rangle$。于是，我们用向量的乘积来表示一个集合中选择的收益之和的期望，例如：玩家的最终目标就是以尽可能大的概率输出M使得

$$M = \arg\max_{M\in\mathcal{M}} = \langle \mu, M\rangle$$

一般来说，对于优化问题的近似算法分加性近似和乘性近似两种。于是我们在下面两节分别讨论组合问题有加性近似算法和乘性近似算法这两种情况。

2.1　加性近似算法

在本节中，我们研究组合多选择赌博机的加性近似算法，即我们不像[1]中一样要求组合问题有多项式时间求解出最优解的黑箱，而是要求组合问题有多项式时间ϵ-加性近似求解问题的黑箱，即如果组合问题是$g : \mathbb{R}^n \to \mathcal{M}$（在输入参数$\mu$之下，最佳的输出集合是$g(\mu)$），那么存在一个多项式时间可计算的函数$f : \mathbb{R}^n \to \mathcal{M}$ 使得$\forall \mu, \langle \mu, f(\mu)\rangle \geq \langle \mu, g(\mu)\rangle - \epsilon$。

首先我们给出一个平凡的算法，用于跟后面我们提出的新算法进行比对。

这个算法的想法也基本平凡，我们即简单对于每一个选择进行足够多的取样，计算其经验平均并最终放入组合问题求解黑箱，以其输出作为整个算法的输出。

对于这个算法，我们证明下面的定理，我们将证明放在附录中。

Algorithm 1 加性Benchmark算法

Require: 概率参数：δ，近似参数ϵ, 组合优化问题的算法黑箱$f(\cdot):\mathbb{R}^n \to \mathcal{M}$;

Ensure: 选择集合 $M \in \mathcal{M}$;

T依定理2.1设定；

对每一个选择取样T次，并计算得到的平均值$\bar{\mu}_i$;

$M \leftarrow f(\bar{\mu}_i)$;

return M;

定理 2.1： 对于加性Benchmark算法，如我们设定$T = \frac{n^3R^2}{\epsilon^2}\log\frac{n}{\epsilon\delta}$，那么以大于等于$1-\delta$的概率，算法输出的选择集合$S$ 是一个ϵ-加性近似最优解。算法的复杂性是$O\left(\frac{n^3R^2}{\epsilon^2}\log\frac{n}{\epsilon\delta}\right)$,其中$R$是概率分布的高斯参数。

我们需要再做一点注记，即在这个算法中我们可以调整T的取值使得最终的2ϵ变成任意$(1+c)\epsilon$，自然相应的T要依赖于c，从而变得更大。所以这个定理的主要意思在于，ϵ-黑箱算法不能够转换成ϵ-纯探寻多选择赌博机问题的在线算法，但是可以转换成任意近似度量稍大一点的近似算法。

下面的算法的原思想来自[1]中的CLUCB算法，我们设计ϵ-CLUCB算法，对其做出一些改进，使其能够完成新的任务。

我们简单叙述如下，首先算法对于每一个选择取样一次作为初始化，之后的每一轮算法将当前各个选择的收益经验平均值$\bar{\mu}^t$ 送入组合问题求解黑箱，得到一个集合M_t，然后根据M_t对于经验平均值$\bar{\mu}^t$进行调整，对于属于M_t的指标，$\tilde{\mu}_i^t = \bar{\mu}_i^t - \mathrm{rad}_t(i)$, 对于不属于$M_t$的指标，$\tilde{\mu}_i^t = \bar{\mu}_i^t + \mathrm{rad}_t(i)$ (这里$\mathrm{rad}_t(i)$代表置信区间，即我们通过证明会看到，以大概率真正的期望值μ_i落在以经验平均值$\bar{\mu}_i$为中心，$\mathrm{rad}_t(i)$为半径的区间之中。这个证明可以由一些基本的概率不等式完成)。然后我们将调整的经验平均值$\tilde{\mu}^t$ 送入组合问题求解黑箱，得到另外一个集合$\tilde{M}_t$，如果这两个集合在调整下的经验平均值中权重相近（不超过ϵ），那么我们就停机并输出。如果这两个集合在调整下的经验平均值中权重不相近（相差超过ϵ），那么我们就继续取样，进入下一个循环。

这个算法的思想在于，如果我们仅仅将取样得到的经验平均值输入组合问题的算法黑箱，那么得到的解会有两层误差，一方面就是取样的随机性带来的误差，另一方面便是算法黑箱本身求解出的集合也带有一定的误差。对于Benchmark算法，我们简简单单将这两部分叠加，但在这个改进CLUCB算法

Algorithm 2 ϵ-CLUCB算法

Require: 概率参数：δ，近似参数ϵ, 组合优化问题的算法黑箱$f(\cdot):\mathbb{R}^n \to \mathcal{M}$;

Ensure: 选择集合 $M \in \mathcal{M}$;

初始化：对于每一个选择做一次取样，对于每一个$i \in [n]$，设定$t_i(n) = 1$并初始化$\bar{\mu}_i^n$;

对于 $t = n+1, n+2, \cdots$ 做

$M_t \leftarrow f(\bar{\mu}_t)$;

计算每一个$\mathrm{rad}_t(i)$ (依定理2.2设定);

对于 $i = 1, 2 \cdots, n$ 做

如果$i \in M_t$则设定$\tilde{\mu}_i^t = \bar{\mu}_i^t - \mathrm{rad}_t(i)$;

否则$i \notin M_t$则设定$\tilde{\mu}_i^t = \bar{\mu}_i^t + \mathrm{rad}_t(i)$;

$\tilde{M}_t \leftarrow f(\tilde{\mu}^t)$;

如果$|\langle\tilde{\mu}^t, \tilde{M}_t\rangle - \langle\tilde{\mu}^t, M_t\rangle| < \epsilon$，停机并输出$M_t$;

否则$p_t \leftarrow \arg\max_{i \in M_t \Delta \tilde{M}_t}\{\mathrm{rad}_t(i)\}$;

选择p_t进行取样;

对于每一个i，更新$t_i(t)$和计算$\bar{\mu}_i^{t+1}$;

中，我们将两部分误差进行了某种程度上的有机融合。具体来说，我们先用经验平均得出一个表现优秀的选择集合，接下来的工作就是对于我们得出的这个集合进行二次验证，事实上，我们降低了集合中选择的经验平均而提升了集合外选择的经验平均，这倾向于鼓励算法找到非我们已经找到集合的集合。如果在这个新参数下我们的集合仍然表现出众，那么就可以说明之前选出的集合“经受住了考验”，从而我们最终选择将它输出。

我们需要附注一下，停机条件$|\langle\tilde{\mu}^t, \tilde{M}_t\rangle - \langle\tilde{\mu}^t, M_t\rangle| < \epsilon$是不能够该换成为文章[1]中的停机条件$\langle\tilde{\mu}^t, \tilde{M}^t\rangle = \langle\tilde{\mu}^t, M_t\rangle$的，因为如果换成严格等于得表达式的话，由于黑箱算法不是每次能够输出最优解，ϵ-CLUCB算法就可能永远没有办法停机，这也是为什么我们需要将停机条件改成相差在ϵ范围之内的原因，同时这一改动也自动的导致ϵ-加性近似的算法黑箱变成了ϵ-加性近似的近似算法算法。

对于这个算法，我们给出下面的定理说明它的性能。

定理 2.2： 对于ϵ-CLUCB算法，如我们设定$\mathrm{rad}_t(i) = R\sqrt{\frac{2\log\frac{4nT^3}{\delta}}{\frac{T}{n}}}$，那么以大于等于$1-\delta$的概率，算法输出的选择集合$S$是一个$\epsilon$-加性近似最优

解。算法的复杂性是$O\left(R^2\text{width}\mathcal{M}^2\frac{n}{\epsilon^2}\log\frac{n}{\epsilon\delta}\right)$,即以大于等于$1-\gamma$的概率，$T \leq \left(R^2\text{width}^2(\mathcal{M})\frac{n}{\epsilon^2}\log\frac{n}{\epsilon\delta}\right)$.

2.2 乘性近似算法

在本节中，我们研究组合多选择赌博机的加性近似算法，即我们不像[1]中一样要求组合问题有多项式时间求解出最优解的黑箱，而是要求组合问题有多项式时间α-乘性近似求解问题的黑箱，即如果组合问题是$g:\mathbb{R}^n \to \mathcal{M}$（在输入参数$\mu$之下，最佳的输出集合是$g(\mu)$），那么存在一个多项式时间可计算的函数$f:\mathbb{R}^n \to \mathcal{M}$使得$\forall\mu, \langle\mu, f(\mu)\rangle \geq \alpha\langle\mu, g(\mu)\rangle$。

首先我们给出一个平凡的算法，用于跟后面我们提出的新算法进行比对。

Algorithm 3 乘性Benchmark算法

Require: 概率参数：δ，近似参数ϵ, 组合优化问题的算法黑箱$f(\cdot):\mathbb{R}^n \to \mathcal{M}$;

Ensure: 选择集合 $M \in \mathcal{M}$;

T依定理2.3设定；

对每一个选择取样T次，并计算得到的平均值$\bar{\mu}_i$;

$M \leftarrow f(\bar{\mu}_i)$;

return M;

这个算法的想法也基本平凡，我们即简单对于每一个选择进行足够多的取样，计算其经验平均并最终放入组合问题求解黑箱，以其输出作为整个算法的输出。

对于这个算法，我们证明下面的定理，我们将证明放在附录中。

定理 2.3： 对于乘性Benchmark算法，如我们设定$T = (\frac{1}{\alpha}-1)^2\frac{3n}{p}\log\frac{n}{\delta}$，那么以大于等于$1-\delta$的概率，算法输出的选择集合$S$是一个$(\alpha-\epsilon)$-乘性近似最优解。算法复杂性是$O\left((\frac{1}{\alpha}-1)^2\frac{n}{p}\log\frac{n}{\delta}\right)$.其中$p$是我们事先知道的一个所有背后概率分布的期望的非零下界。

下面的算法的原思想仍然来自[1]中的CLUCB算法，我们设计(α,ϵ)-CLUCB算法，对其做出一些改进，使其能够完成新的任务

Algorithm 4 (α, ϵ)-CLUCB算法

Require: 概率参数：δ，近似参数ϵ, 组合优化问题的算法黑箱$f(\cdot) : \mathbb{R}^n \to \mathcal{M}$;

Ensure: 选择集合 $M \in \mathcal{M}$;

初始化：对于每一个选择做一次取样，对于每一个$i \in [n]$，设定$t_i(n) = 1$并初始化$\bar{\mu}_i^n$;

对于 $t = n+1, n+2, \cdots$ 做

$M_t \leftarrow f(\bar{\mu}_t)$;

计算每一个$\mathrm{rad}_t(i)$ (依定理2.2设定);

对于 $i = 1, 2 \cdots, n$ 做

如果$i \in M_t$则设定$\tilde{\mu}_i^t = \bar{\mu}_i^t - \mathrm{rad}_t(i)$;

否则$i \notin M_t$则设定$\tilde{\mu}_i^t = \bar{\mu}_i^t + \mathrm{rad}_t(i)$;

$\tilde{M}_t \leftarrow f(\tilde{\mu}^t)$;

如果$\langle \tilde{\mu}^t, \tilde{M}_t \rangle \geq (\alpha - \epsilon)\langle \tilde{\mu}^t, M_t \rangle$，停机并输出$M_t$;

否则$p_t \leftarrow \arg\max_{i \in M_t \Delta \tilde{M}_t} \{\mathrm{rad}_t(i)\}$;

选择p_t进行取样;

对于每一个i，更新$t_i(t)$和计算$\bar{\mu}_i^{t+1}$;

定理 2.4： 对于任意给定的$\alpha > \frac{1}{2}$以及$\epsilon > 0$,在(α, ϵ)-CLUCB算法中如果我们设定$\mathrm{rad}_t(i) = R\sqrt{\frac{2\log\frac{4nT^3}{\delta}}{\frac{T}{n}}}$，会以大于等于$1 - \delta$ 的概率，算法输出的选择集合S是一个$(2\alpha - 1 - \epsilon)$-乘性近似最优解。

我们简单叙述如下，首先算法对于每一个选择取样一次作为初始化，之后的每一轮算法将当前各个选择的收益经验平均值$\bar{\mu}^t$ 送入组合问题求解黑箱，得到一个集合M_t，然后根据M_t对于经验平均值$\bar{\mu}^t$进行调整，对于属于M_t的指标，$\tilde{\mu}_i^t = \bar{\mu}_i^t - \mathrm{rad}_t(i)$ (这里$\mathrm{rad}_t(i)$代表置信区间，即我们通过证明会看到，以大概率真正的期望值μ_i落在以经验平均值$\bar{\mu}_i$为中心，$\mathrm{rad}_t(i)$为半径的区间之中。这个证明可以由一些基本的概率不等式完成)。然后我们将调整的经验平均值$\tilde{\mu}^t$ 送入组合问题求解黑箱，得到另外一个集合M_t，如果这两个集合在调整下的经验平均值中权重相近（不超过$(\alpha - \epsilon)$倍），那么我们就停机并输出。如果这两个集合在调整下的经验平均值中权重不相近（相差超过$(\alpha - \epsilon)$倍），那么我们就继续取样，进入下一个循环。

我们需要附注一下，停机条件$\langle \tilde{\mu}^t, \tilde{M}_t \rangle \geq (\alpha - \epsilon)\langle \tilde{\mu}^t, M_t \rangle$是不能够再加强的，

因为如果再加强的话，由于黑箱算法不是每次能够输出最优解，(α, ϵ)-CLUCB算法就可能永远没有办法停机，这也是为什么我们需要将停机条件改成相差在$(\alpha - \epsilon)$倍范围之内的原因，同时这一改动也自动的导致α-乘性近似的算法黑箱变成了$(2\alpha - 1 - \epsilon)$-乘性近似的近似算法算法。

除此之外，我们还需要做出的一点说明就是，乘性的(α, ϵ)-CLUCB和加性的ϵ-CLUCB算法看起来形式一样，都是将算法黑盒的误差增大了一倍，但是实际上并不能说是一样。一方面加性近似算法的误差值是固定的，ϵ不会随着输入以及问题的大小而改变，但是乘性的近似算法则不然，这会引起误差的大小不能简简单单的由双倍这一特点来控制。另一方面，两种算法各有不同的分析方法，特别是对于乘性算法，如果我们利用加性误差来分析，那么就在近似比中永远摆脱不掉这个ϵ，因此在某种意义上我们可以说，问题的转化中，加性误差比乘性误差更加本质。

本节的所有定理的严格证明都将放到附录中。

第 3 章　纯探寻组合多选择赌博机问题的不可近似性结论

在本节中我们介绍若干纯探寻组合多选择赌博机问题的不可近似性结论。我们首先做下述讨论。

上一节中，我们设计了几个算法，和CLUCB算法的形式一样，他们都是要求有一个组合问题的黑箱算法，能够求出某种程度上的线下近似解（即，ϵ-加性近似算法黑箱能够提供和最优解相差不超过ϵ的次优解，α-乘性近似算法黑箱能够提供不低于最优解α倍的次优解），然而并没有对于这个算法黑箱做出额外的假设，比如输出稳定性等性质，而这一性质是众多近似算法真真切切具有的。所以，在并未做出这样的假设的情况下，我们对于算法黑箱的了解就非常之少，仅仅在于他的输出的最优性。因此，如果想证明形如“在我们拥有某组合优化问题的某种近似算法黑箱之下，不可能存在转化算法使得我们可以拥有这个组合问题的纯探寻多选择赌博机的相同近似程度的近似算法”便是极其困难的。一方面，我们对于黑箱没有额外信息，另一方面，转化算法的范围也极其广泛，我们没有一个范式加以限制。放眼理论计算机领域证明不可能性的方法，在计算理论方面，我们一开始看到的是对角化方法，即从逻辑矛盾的角度进行推理。在计算复杂型方面，证明一个问题没有多项式时间算法的一般方法是找到一个NP-complete 问题进行归约，尽管即便如此我们还并没有完全证明它没有多项式时间算法因为P-NP问题目前还没有得到最终解决，另外一种方法便是构造某一个反例或者反例模型，通过调整其中的参数来证明任意一个算法，不能够在参数任意变动的时候对于每一种情况都达到期望的效果，这在包含文章[5]在内的大量理论文章中被应用。

对于这些方法。我们通过观察可以得到，前两种方法，即对角线方法和归约方法，所使用的问题的描述都比较简单，即，问题的设定并不复杂。通常为“不存在一个图灵机能够判定停机问题”，或者“顶点覆盖问题是难解的”之类。而后面的一种方法并没有这么高的要求，尽管它也需求问题的设定尽可能的简单为好，它可以在各种情形下被灵活应用，证明一些不可能的结论。这种方法也是我们在这一节中希望采取的方法，其关键在于通过深度思考对于问题拥有深刻的洞察力，并依此构造出一个抓住关键性质的反例。

拥有了这种方法，但是我们所期望的命题形式仍然需要被更改，我们之前

的期望模式为：我们拥有了黑箱近似算法A，不存在这样的转化B，使得$B \circ A$能够解决纯探寻多选择赌博机问题。希望证明这样的B不存在，经过前面的分析我们事实上不妨就去证明这样的$B \circ A$整体不存在，也即，我们摆脱已经有线下近似算法黑箱这样的限制，去证明某个组合问题的纯探寻组合多选择赌博机问题没有办法得到某种近似要求的解。这毫无疑问是一个更强的结论，因此我们也被迫要求要拥有更强的假设。

在开始具体讨论之前，我们需要一些定义。

定义 3.1： 我们称组合优化问题的一个随机算法是(ϵ, δ)-精确的，如果它对于任意的输入，以大于$(1-\delta)$的概率输出ϵ-加性近似解。

例如，文章[5]中提出的Median Elimination算法便是一个(ϵ, δ)-精确的算法，其复杂性为$O\left(\frac{n}{\epsilon^2}\log\frac{1}{\delta}\right)$。

对于一个随机算法，我们也需要做如下分类。

定义 3.2： 对于一个随机算法，如果

(1)存在一个时间函数$T(\cdot): \mathbb{N} \to \mathbb{N}$,使得对于任意输入大小$n$，无论随机算法采取的随机性如何，算法都可以在$T(n)$步之内停机。我们称之为“确定时间停机算法”。

(2)对于任意的输入，算法以概率1停机，其中随机性包括算法内自带的随机性以及算法从输入概率分布中取样的随机性，我们称之为“ 概率1停机算法”。

事实上，对于不属于上述两种情况的算法，我们是并不感兴趣的。因为算法的定义即是，一串无歧义可停止的指令，如果有非零概率一个指令串无法停机，从定义上来说它也不应该被称为一个算法。在上面两种定义中，显然第二种是包含第一种的。因此第二种定义包含的算法范围更大，形成的结论也更强。

例如，算法CLUCB就是属于这一类的算法。因为它不断地从概率分布中取样，每一轮维护一个取样平均值和置信区间，当这个置信区间最终达到某种“分离特征”的时候（算法在某一轮两次使用线下黑箱算法并且得到一样的最优解，这说明低期望的置信区间和高期望的置信区间发生了某种分离）算法才能平安停机并且输出。而依概率中的弱大数定律或者Chernoff不等式的结论，取样平

均值以大概率落在真是数学期望的周围很小去见范围内，速度由Chernoff不等式给出，这个概率在n趋向于无穷大的时候是等于1的。所以说CLUCB 算法以概率1停机。

我们可以用另外一个例子来更清楚地诠释概率1停机，这个例子也是我们证明后面的不可能性结论的关键问题，也为上文中提到的“分离”给出进一步的解释。

我们考虑两个硬币，其中一个抛出正面的概率为p_1，另外一个抛出正面的概率为p_2，我们已知$p_1 \neq p_2$但我们并不知道这两个具体数值。我们需要不停地取样并且更新获得的知识，从而最终输出哪一个概率更高。

由大数定律来讲，在对于每一个硬币做足够多次取样之后，我们就会很精确的了解这个硬币抛出正面的概率值到底如何，具体来说，我们可以为每一个硬币定义一个“置信区间”，比如，当我们抛硬币a次得到b次正面的时候，我们可以定义这个置信区间为$[a-\frac{1}{b}, a+\frac{1}{b}]$，也就是说我们相信以大概率硬币抛出正面的真正概率值p就落在这个区间之内。对于区分两枚硬币的算法来说，我们可以设计为：当两个置信区间没有交集的时候，就选择置信区间高的那个作为输出。

这个算法便是又一个典型的以概率1停机的算法，而并不知道一个具体的表现很好的算法T使得算法在T步之后一定停止（我们会在下文证明）。事实上，如果两个正面概率值p_1,p_2非常的接近,在一定的之内步数之内有一定概率两者的输出完全相同，而这种情况我们没有看出两个硬币的任何区别，所以不能够做出良好的判断。因此任意T步之后停机的算法，其输出的准确度总是难以达到很好。

于是，在本小节的末尾，我们正式叙述一个问题，后续的不可能性定理都围绕这个问题来展开。

问题 3.1： 输入两枚硬币，抛出后得到正面的概率分别是p_1和p_2，我们可以对每一个硬币进行取样，获得其背后0 – 1概率分布的一个取样值。设计算法判定哪一个硬币的正面概率更大。如果$p_1 > p_2$那么输出硬币1则为正确，如果$p_2 > p_1$那么输出硬币2则为正确，如果$p_1 = p_2$那么输出硬币1或者硬币2均为正确。

3.1 最优解不可能性定理

在本节中，我们对于上面定义的问题，证明下面的不可能性定理：

定理 3.1： （最优解不可能性定理）

（1）对于任意$\delta < \frac{1}{2}$，不存在$(0,\delta)$-精确的确定时间停机算法

（2）对于任意$\delta < \frac{1}{2}$，如果没有事先确切知道$p_1 \neq p_2$（最优解的唯一性），不存在$(0,\delta)$-精确的概率1停机算法

我们需要简单做一点附注。事实上，如果事先确切知道$p_1 \neq p_2$，那么我们上述的“置信区间”算法（其实也是CLUCB算法）就是一种可行的$(0,\delta)$-精确的概率1停机算法。除了以概率1停机之外，对于CLUCB算法我们还知道，以大于等于$1-\delta$的概率，算法的停机时间满足

$$T \leq O\left(\text{width}^2(\mathcal{M}) \sum_{i=1}^{n} \frac{1}{\Delta_i^2}\right)$$

因此这个定理事实上把算法的存在与否这一问题转移到了是否事先拥有最优解唯一性的知识的问题。

对于这个命题，我们使用的便是之前陈述的构造反例调参数证明方法。我们将全部证明放在附录之中。

3.2 近似最优解不可能性定理

在本节中，我们分析乘性近似算法的不可能性，首先类似于(ϵ,δ)-精确的概念，我们定义乘性中类似的概念如下。因为在PAC-learning中经常使用之前定义的(ϵ,δ)-精确，因此我们并不把它称为(ϵ,δ)-精确，但由于下面这个定义是我们自己提出的，我们需要把它称为(α,δ)-乘性精确。

定义 3.3： 我们称组合优化问题的一个随机算法是(α,δ)-乘性精确的，如果它对于任意的输入，以大于$(1-\delta)$的概率输出α- 乘性近似解。

类似于上一小节的结论，本小节中，我们证明下面的不可能性定理：

定理 3.2： （近似解不可能性定理）

（1）对于任意$\delta < \frac{1}{2}$以及任意的$0 < \alpha < 1$，不存在(α,δ)-精确的确定时间停机算法

（2）对于任意$\delta < \frac{1}{2}$，如果没有事先确切知道一个$p > 0$,满足$p_1 \geq p, p_2 \geq p$（正面概率的已知下界），不存在(α, δ)- 精确的概率1停机算法

我们需要简单做一点附注。事实上，如果没有事先确切知道一个$p > 0$,满足$p_1 \geq p, p_2 \geq p$，那么我们上述的“置信区间”算法，稍加改进便可以做到(α, δ)-乘性精确，而算法的计算复杂性与p有直接的关联。

因此这个定理事实上把算法的存在与否这一问题转移到了是否事先了解概率下界问题。

对于这个命题，我们使用的便是之前陈述的构造反例调参数证明方法。我们将全部证明放在附录之中。

第 4 章　定理的证明

本节中，我们证明之前的小节中陈述的所有定理，我们沿用我们叙述算法那一节中所定义的符号。类似于文章[1]中的做法，我们先证明一些“大概率事件”的概率界。之后的证明都是建立在这些大概率事件成立的基础之上。

由于我们设定的置信区间和文章[1]中CLUCB算法相同，从而，我们下述定理采取类似的概率不等式和证明方法。

引理 4.1：　假设对于每一个选择，其背后的收益概率分布都是R-次高斯尾分布的，那么假设我们设定的置信区间的定义是

$$\mathrm{rad}_t(i) = \sqrt{\frac{2\log\left(\frac{4nT^3}{\delta}\right)}{t_i(T)}} \cdot R$$

那么，以大于等于$1-\delta$的概率，对于任意的$t \le T$以及任意的$i \in [n]$，下面的不等式成立：

$$|\mu_i - \mu_i^t| \le \mathrm{rad}_t(i)$$

在给出所有定理的证明之前，我们先证明下面的引理。

引理 4.2：　ϵ-CLUCB算法之中，对于任意的集合M，以及任意的常数$0 < \beta < 1$，有

$$\beta\langle\tilde{\mu}^t, M\rangle - \langle\tilde{\mu}^t, M_t\rangle \ge \beta\langle\mu, M\rangle - \langle\mu, M_t\rangle$$

证明　首先，我们假设：$M_1 = M \cap M_t, M_2 = M - M_1, M_3 = M_t - M_1$。

同时，我们令$\mathrm{rad}_t = (\mathrm{rad}_t(i))_{i\in[n]}$为代表第$t$轮置信区间的向量，其与集合指

示向量的点乘运算和之前定义的一样。

$$\begin{aligned}\beta\langle\tilde{\mu}^t, M\rangle - \langle\tilde{\mu}^t, M_t\rangle =&\beta\langle\tilde{\mu}^t, M\rangle - \langle\bar{\mu}^t, M_t\rangle + \langle \mathrm{rad}_t, M_t\rangle \\ =&\beta\langle\bar{\mu}^t, M_1\rangle + \beta\langle \mathrm{rad}_t, M_1\rangle + \beta\langle\bar{\mu}^t, M_2\rangle + \beta\langle \mathrm{rad}_t, M_2\rangle - \langle\bar{\mu}^t, M_t\rangle + \langle \mathrm{rad}_t, M_t\rangle \\ =&\left(\beta\langle\bar{\mu}^t, M_1\rangle + \beta\langle \mathrm{rad}_t, M_1\rangle\right) + (-1-\beta)\left(\langle\bar{\mu}^t, M_2\rangle - \beta\langle \mathrm{rad}_t, M_2\rangle\right) \\ &-\left(\langle\bar{\mu}^t, M_3\rangle + \langle \mathrm{rad}_t, M_3\rangle\right) \\ \geq&\beta\langle\mu, M_1\rangle + (-1-\beta)\langle\mu, M_2\rangle - \langle\mu, M_3\rangle \\ =&\beta\langle\mu, M\rangle - \langle\mu, M_t\rangle\end{aligned} \tag{4-1}$$

综上所述，命题得证。

□

4.1 定理2.1的证明

证明 根据停机条件，我们有：

$$\langle\bar{\mu}^t, M_t\rangle \geq \langle\bar{\mu}^t, M_t^*\rangle - \epsilon$$

其中M_t^*是参数为$\bar{\mu}^t$时的最优解。

所以，当T如定理中设定的情况时，我们有

$$\langle\mu, M_t\rangle \geq \langle\bar{\mu}^t, M_t\rangle - \frac{\epsilon}{2}$$

以及

$$\langle\mu, M^*\rangle \leq \langle\bar{\mu}^t, M^*\rangle + \frac{\epsilon}{2}$$

综合这几个不等式，我们得到：

$$\begin{aligned}\langle\mu, M_t\rangle &\geq \langle\bar{\mu}^t, M_t\rangle - \frac{\epsilon}{2} \\ &\geq \langle\bar{\mu}^t, M_t^*\rangle - \frac{3\epsilon}{2} \\ &\geq \langle\bar{\mu}^t, M^*\rangle - \frac{3\epsilon}{2} \\ &\geq \langle\mu, M^*\rangle - 2\epsilon\end{aligned} \tag{4-2}$$

故综上所述，命题得证。

□

4.2 定理2.2的证明

证明 根据停机条件我们知道：

$$\langle \tilde{\mu}^t, M_t \rangle \geq \langle \tilde{\mu}^t, \tilde{M}_t^* \rangle - \epsilon$$

其中$\tilde{M}_t^*$是参数为$\tilde{\mu}^t$的时候的最优解。

根据引理4.2的结论，我们有：

$$\begin{aligned} \langle \mu, M^* \rangle - \langle \mu, M_t \rangle &\leq \langle \tilde{\mu}^t, M^* \rangle - \langle \tilde{\mu}^t, M_t \rangle \\ &\leq \langle \tilde{\mu}^t, \tilde{M}_t^* \rangle - \langle \tilde{\mu}^t, M_t \rangle + \epsilon \\ &\leq 2\epsilon \end{aligned} \tag{4-3}$$

其中其中M^*是参数为真是期望值μ的时候的最优值。

综上所述，命题得证。

□

4.3 定理2.3的证明

证明 根据f的定义，当我们输入$\bar{\mu}^t$入黑箱的时候，对于得到的输出，有：

$$\langle \bar{\mu}^t, M_t \rangle \geq \alpha \langle \bar{\mu}^t, M_t^* \rangle$$

其中M_t^*是参数$\bar{\mu}^t$之下的最优解。

由于对于任意的$i \in [n]$，有：

$$|\mu_i^t - \mu_i| \leq \mathrm{rad}_i(t)$$

所以有：

$$\begin{aligned}\langle \mu, M_t \rangle &\geq \langle \bar{\mu}^t - \mathrm{rad}_t, M_t \rangle \\ &\geq \langle \bar{\mu}^t, M_t \rangle - \langle \mathrm{rad}_t, M_t \rangle \\ &\geq \alpha \langle \bar{\mu}^t, M_t^* \rangle - \langle \mathrm{rad}_t, M_t \rangle \\ &\geq \alpha \langle \bar{\mu}^t, M^* \rangle - \langle \mathrm{rad}_t, M_t \rangle \\ &\geq \langle (\alpha - \epsilon)\mu, M^* \rangle\end{aligned} \tag{4-4}$$

其中M^*是参数为真实的期望收益μ的时候的最优解。

从而综上所述，命题得证。

□

4.4 定理2.4的证明

在本节中，我们证明定理2.4，我们应用4.1和4.2的结论。

证明 首先，根据4.1的结论我们知道以大概率，所有的取样平均值都在真实的期望周围置信区间长度的范围之内。我们希望证明在这样的情况之下，定理的结论成立。

事实上，我们希望证明

$$\langle \mu, M_t - cM^* \rangle \geq 0$$

成立，其中c是一个常数（我们到最后会说明它是多少），而

$$M^* = \arg\max_{M \in \mathcal{M}} \{\langle \mu, M \rangle\}$$

是在真正的期望下的最优解。

假设结论不成立，则有

$$\langle \mu, cM^* - M_t \rangle > 0$$

根据引理4.2，假设算法在第t轮停机，我们有

$$\langle \tilde{\mu}^t, cM^* - M_t \rangle \geq \langle \mu, cM^* - M_t \rangle > 0$$

但是根据我们的算法，我们有：

$$\langle \tilde{\mu}^t, \alpha \tilde{M}_t^* - \tilde{M}_t \rangle \leq 0$$

其中$\tilde{M}^*$是在$\tilde{\mu}^t$下的最优解。

因此把上面的两个不等式相结合（第一个不等式反向与第二个不等式结合），我们得到

$$\langle \tilde{\mu}^t, \alpha \tilde{M}_t^* - \tilde{M}_t + M_t - cM^* \rangle < 0 \tag{4-5}$$

然而由于$\tilde{M}_t^*$的最优性，我们有：

$$\langle \tilde{\mu}^t, cM^* \rangle \leq \langle \tilde{\mu}^t, \tilde{M}_t^* \rangle$$

以及根据概率分析结论，我们有

$$\langle \tilde{\mu}^t, M_t \rangle \geq (\alpha - \epsilon) \langle \tilde{\mu}^t, \tilde{M}_t \rangle$$

以及

$$\langle \tilde{\mu}^t, \tilde{M}_t^* \rangle \leq \langle \tilde{\mu}^t, \tilde{M}_t \rangle$$

结合上面三个不等式，命$c = 2\alpha - 1 - \epsilon$，我们可以得到：

$$\langle \tilde{\mu}^t, \alpha \tilde{M}_t^* - \tilde{M}_t + M_t - cM^* \rangle \geq \langle \tilde{\mu}^t, \alpha \tilde{M}_t^* - (2\alpha - 1 - \epsilon) \tilde{M}_t^* - (1 - \alpha + \epsilon) \tilde{M}_t \rangle \geq 0$$

这与4-5相矛盾。

综上所述，命题得证。

□

4.5 定理3.1的证明

证明 我们用反证法.

文章[5]证明了下述结论(文章中的定理1)

引理 4.3： 存在正实数c_1, c_2, ϵ_0,和δ_0, 使得对于任意的$n \geq 2$,与$\epsilon \in (0, \epsilon_0)$,以及$\delta \in (0, \delta_0)$，以及对于任意的 (ϵ, δ)-精确的算法（概率1停机）, 存在某一个 $p \in [0,1]^2$ 使得:

$$\mathbb{E}_p[T] \geq c_1 \frac{n}{\epsilon^2} \log \frac{c_2}{\delta}$$

其中$\mathbb{E}_p[T]$是算法在p为参数的反例之下的期望运行时间。

如果命题叙述的算法A存在,我们就可以适当选取足够小的$\epsilon > 0$使得$c_1 \frac{n}{\epsilon^2} \log \frac{c_2}{\delta} > T$.因为算法$A$是$(0, \delta)$-精确的, 则它也是$(\epsilon, \delta)$-精确的. 但是根据引理的结论，它的期望运行时间超过T, 矛盾。

□

在证明第二小命题之前，我们需要先做一些概率论上的明确定义使得下面的数学证明是严谨的。

定义 4.1： (实例; 有限实例)

一对无穷长度的$0-1$序列$\tau = (a_1 a_2 \cdots a_n \cdots, b_1 b_2 \cdots b_n \cdots)$被称为一对硬币的一个实例,即当一个算法对于第1枚硬币做第i^{th}次取样的时候, 得到的结果是a_i, 当一个算法对于第2枚硬币做第j^{th}次取样的时候, 得到的结果是b_j

一对有限长度的$0-1$序列$\phi = (a_1 a_2 \cdots a_n \cdots, b_1 b_2 \cdots b_n \cdots)$被称为一对硬币的一个有限实例,即当一个算法对于第1 枚硬币做第i^{th}次取样的时候, 得到的结果是a_i, 当一个算法对于第2枚硬币做第j^{th}次取样的时候, 得到的结果是b_j。我们用$n = |\phi|$来表示这个有限实例的大小.

因为算法的输出仅仅取决于取样的带的结果，所以对于任意一个实例，算法要么在这个实例上不停机，要么在这个实例的一个有限部分（前n 位构成的有限实例）上停机。对于一个未知的概率结构p_1, p_2，我们可以如下定义一个σ-代数，同时在上面定义测度$\mathbb{P}_{p_1,p_2}(\cdot)$如下。

定义 4.2： 令Ω为所有实例的集合，那么对于一个有限实例ϕ，定义集合$S_\phi = \{\tau \mid \phi$是前缀于$\tau\}$，定义集合族$\mathcal{A} = \{S_\phi \mid \phi$ 是有限实例$\} \cup \{\emptyset\} \cup \{\Omega\}$，那么 $\mathcal{A}$是一个 Ω上的σ代数。

定义函数$\mathbb{P}_{p_1,p_2} : \mathcal{A} \to [0,1]$使得 $\mathbb{P}_{p_1,p_2}(\emptyset) = 0, \mathbb{P}_{p_1,p_2}(\Omega) = 1$ 以及

$$\mathbb{P}_{p_1,p_2}(\phi) = \prod_{1 \leq i \leq n} (1 - p_1 + a_i(2p_1 - 1))(1 - p_2 + b_i(2p_2 - 1))$$

不难验证 $\mathbb{P}_{p_1,p_2}$ 是一个 $(\Omega, \mathcal{A})$上的概率测度.

定义 4.3： (以概率1停机)

设A是一个算法, 令$I_A(\tau)$ 是停机的指示向量, 即$I_A(\tau) = 1$ 当且仅当 A 在 τ的前缀有限实例上停机。

我们说算法A以概率1停机，如果对于所有的(p_1, p_2), 有$\mathbb{E}_{\mathbb{P}_{p_1,p_2}}[I_A] = 1$成立。

于是我们来证明定理3.1的第二点。

证明 我们用反证法，假设这样的算法存在。首先不失一般性，我们假设算法在停机的时候总是对于两枚硬币做同样多次的取样（事实上，如果一个算法对于第一枚硬币做r次取样，而对于第一枚硬币做t次取样$(r > t)$，我们也可以令它对于两个硬币都做r次取样，但是仅仅用了硬币2取样的前t个结果）。

现在我们假设，当算法看到一个特定的有限实例的时候，它要么停机要么继续做下一次取样（我们原本允许算法包含随机性，即它可以随机选择停机或者继续去做下一个取样，但是由于下面证明的简便性与清晰性，我们先不考虑这种可能。尽管整个证明经过小小的调整就也可以同时适用于这种情况）。

定义 4.4： (停机的有限实例)

令ϕ是一个有限实例，那么对于算法A来说，它是一个停机有限实例，如果算法在看到他的一个前缀实例之后停机

根据定义，对于任意的 (p_1, p_2), 令

$$P_n(A) = \sum_{|\phi|=n;\phi \text{ 是对于算法A的停机有限实例}} \mathbb{P}_{p_1,p_2}(S_\phi)$$

我们有:

$$\lim_{n\to\infty} P_n(A) = 1$$

选取 $\epsilon > 0$ 使得 $\epsilon < \frac{1}{2} - \delta$ 成立.命 $p_1 = p_2 = \frac{1}{2}$, 则存在 $N(\frac{1}{2}) \in \mathbb{N}$ 使得对于任意的 $n \geq N(\frac{1}{2})$, 有$P_n(A) \geq 1 - \frac{1}{2}\epsilon$成立.

现在，我们取足够小的 $\epsilon_1 > 0$ 使得

$$\sum_{|\phi|=N(\frac{1}{2})} |\mathbb{P}_{\frac{1}{2},\frac{1}{2}}(S_\phi) - \mathbb{P}_{\frac{1}{2}+\epsilon_1,\frac{1}{2}-\epsilon_1}(S_\phi)| \leq \frac{1}{2}\epsilon$$

以及

$$\sum_{|\phi|=N(\frac{1}{2})} |\mathbb{P}_{\frac{1}{2},\frac{1}{2}}(S_\phi) - \mathbb{P}_{\frac{1}{2}+\epsilon_1,\frac{1}{2}-\epsilon_1}(S_\phi)| \leq \frac{1}{2}\epsilon$$

成立

这种选取是可以完成的，因为函数$g(x) = \sum_{|\phi|=N(\frac{1}{2})} |\mathbb{P}_{\frac{1}{2},\frac{1}{2}}(S_\phi) - \mathbb{P}_{\frac{1}{2}-x,\frac{1}{2}+x}(S_\phi)|$ 是连续的，而且$g(0) = 0$.

于是我们考虑下述两种构造。第一种中我们令 $p_1 = \frac{1}{2} + \epsilon_1, p_2 = \frac{1}{2} - \epsilon_1$, 在第二个构造中 $p_1 = \frac{1}{2} - \epsilon_1, p_2 = \frac{1}{2} + \epsilon_1$. 显然，在第一个结构中输出 1是正确的，而第二个结构中输出2是正确的。

我们考虑一个算法在停机有限实例上的输出，由于对于两种结构，算法在$N(\frac{1}{2})$步之内停机的概率大于等于$1 - \epsilon$。令s_1 是算法在第一种结构中输出1的概率，令s_2是算法在第一种结构中输出2的概率.令t_1是算法在第二种结构中输出1的概率，令t_2是算法在第而种结构中输出2的概率。因此，鉴于算法的输出仅仅取决于他观察到的实例，我们有

$$|s_1 - t_1| + |s_2 - t_2| \leq \epsilon$$

因此我们得到, $|s_1 - t_1| \leq \epsilon, |s_2 - t_2| \leq \epsilon$.所以$s_1 + s_2 \leq 1$, 由于算法输出正确概率至少是 $1-\delta$, 我们得到$s_1 \geq 1-\delta$, 所以 $s_2 \leq \delta$同时 $t_2 \leq \delta+\epsilon$，因此 $t_1 \geq 1-2\epsilon-\delta$, 我们得到，算法在第二种结构上输出正确的概率 $2\epsilon + \delta < 1 - \delta$, 矛盾。

综上所述，命题得证。

□

4.6 定理3.2的证明

证明 我们用反证法。

首先我们假设这样的算法存在，即对于任意的输入都以概率1停机。和上一个定理类似，我们不妨假设算法A对于两枚硬币做的取样次数是一样的。那么由于我们没有对于输入的下界的知识。我们先行考虑两枚硬币取正面概率都为0的情况。在这种情况之下，算法永远会看到背面。由于算法会以概率1停机，我们可以的得到概率序列$\{P_n^A\}_{n\in\mathbb{N}}$，使得

$$\lim_{n\to\infty} P_i^A = 1$$

所以对于任意$\epsilon > 0$，我们可以找到$N_\epsilon \in \mathbb{N}$，使得对于任意的$n > N_\epsilon$，我们有$P_n > 1 - \epsilon$成立。我们考虑如下的两种概率结构：

（1）$p_1 = \beta, p_2 = 0$

（2）$p_1 = 0, p_2 = \beta$

在这两种情况下，β被取到足够小以至于对于一个正面概率为δ的硬币，抛N_ϵ次之后得到的观察全是反面的概率大于$1 - \epsilon$。因此，以不小于$(1 - 3\epsilon)$的概率，算法在(1), (2)两种情况下都在N_ϵ步之内停机，并且在停机之前看到的取样全是反面。

由于两种情况下看到的取样结果都是一样的，因此算法不可能再这样的情况下以大于二分之一的概率做出正确的判断。所以，算法做出正确判断的概率就不大于$(1 - 3\epsilon) \cdot \frac{1}{2} + 3\epsilon = \frac{1}{2} + \frac{3}{2}\epsilon$。取$\epsilon < \frac{1}{2}(\delta - \frac{1}{2})$即可得到矛盾。

□

4.7 关于各定理中算法复杂性的分析

类似于文章[1],我们在复杂性分析中主要使用下面的两个概率密度集中引理：

引理 4.4： (Hoeffding Inequality)

假设$\{X_i\}_{1 \le i \le n}$是n个独立的R-次高斯分布随机变量，令$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$，那么我们有：

$$\Pr\left[|\bar{X} - \mathbb{E}[\bar{X}]| > \epsilon\right] \le 2\exp(-\frac{n\epsilon^2}{2R^2})$$

引理 4.5： (Chernoff Bound)

假设$\{X_i\}_{1 \le i \le n}$是n个独立同分布随机变量(取值范围是$[0, 1]$)，令$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$，那么我们有：

$$\Pr\left[\bar{X} > (1 + \epsilon)\mathbb{E}[\bar{X}]\right] \le 2\exp(-\frac{\epsilon^2\mathbb{E}[\bar{X}]}{3})$$

这两个定理帮助我们确定到底要取样多少次，从而可以达到应有的保证概率，对于算法复杂性的分析细节，我们在此略去。

第 5 章　结语

本文研究了组合多选择赌博机问题，在纯探寻方面设计了在线算法，分别达到了求加性近似解和乘型近似解的要求。这一结果推广了之前[1]文章的结论，将CLUCB算法进行改进使得原来只能够求解最优解的算法的能力得到了提升。另一方面，我们证明了在将近似黑箱算法转换成在线近似算法的时候，误差不可能得到完好的保存，即，近似比例或者近似系数都不可避免的有一定的增大。为此我们使用概率论方面的技巧，证明了两个不可能性定理，指出相应算法的存在与否转变为我们是否对于问题的输入系数有前提假设或者先知有关。本文提出的算法是基于CLUCB算法上做出一些调整，计算复杂性还有提高的空间，这也是未来这个问题的潜在研究方向。

公式索引

参考文献

[1] C.Wei et al. *Combinatorial Pure Exploration of Multi-Armed Bandits*. In *Proceedings of the 27th Annual Conference on Advances in Neural Information Processing Systems (NIPS)*, 2014.

[2] C.Wei et al. *Combinatorial multi-armed bandit: general framework, results and applications*. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.

[3] P.Auer et al. *Finite-time Analysis of the Multiarmed Bandit Problem*. In *Machine Learning*, 47, 235-256, 2002.

[4] S.M. Kakade et al. *Playing Games with Approximation Algorithms*. In *SIAM Journal of Computing*, Vol. 39, No. 3, pp. 1088-1106, 2009.

[5] S.Mannor et al. *The Sample Complexity of Exploration in the Multi-Armed Bandit Problem*. In *Journal of Machine Learning Research*, No. 5, pp. 623-648, 2004.

[6] A.Badanidiyuru et al. *Bandits with Knapsacks*. In *54th IEEE Symposium on Foundations of Computer Science (FOCS)*, 2013.

[7] T.M. Cover et al. *On Determining the Irrationality of the Mean of a Random Variable*. In *The Annals of Statistics*, Volume 1, No. 5, 862-871, 1973.

致　谢

我首先希望感谢本系的李建老师，为他在研究方向的选择方面给予的指导，在我完成毕业论文的过程中对于我的勉励，以及在研究问题的过程中给我的帮助。其次，我希望感谢微软的高级研究员，也是交叉信息院的客座教师陈卫老师，感谢他提出这个问题并且给予我耐心的指导，让我在深度阅读文献以及作报告方面的能力获得了一定的提高。除此之外我也希望感谢林添学长，在我做研究的过程中对于一些细节和我进行过深入的讨论并且想我讲述不同相关问题的之间的联系，与特殊概率技巧的使用，等等。

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____ 日　期：_____

附录 A　英文文献调研报告

Multi-armed bandit problem has been studied for years since the big-data time has come. Big-data means data with larger volume, velocity and variety. In big-data days there is a growing need for handling problems more efficiently and handling problems online. Multi-armed bandit is a representative problem for online learning. First issued in[3], and leads a discussion of thousands of theoretical and experimental papers.

In[3], P.Auer et al set up the framework for the multi-armed problem. We are faced with a bandit with n arms (or n bandits), and each arm is equipped with a latent distribution which is unknown to the player (the distribution could take values only on $[0, 1]$ after normalization). The multi-armed bandit game is played in rounds (the number of rounds T is specified to the player before the game). In every round, the player is allowed to play one arm, and then a reward, a sampled value from its latent distribution, is given to the player. The player need to simultaneously learn the information of the distribution (what is its expectation, or variance, etc) as well as maximize his payoff, which is the sum of all rewards in every round. To be specific, the objective function is called *regret function*, defined to be $T\mu^* - \sum_{i=1}^{T} \mu_{\tau(i)}$, where μ^* is the largest expectation over the expectations of all distributions, and $\mu_{\tau(i)}$ is the expectation of the distribution of the arm selected by policy τ in the i^{th} round.

The first algorithm called *upper confidence index* was proposed in[3] (also known as *upper confidence bound* algorithm (UCB)). This is an index algorithm, making the decision for the next round (pick which arm to push) by updating the index of each arm in certain ways. To be specific, for an arm that has been played n_j times in the first n rounds, the index for this arm is set to be $\bar{x} + \sqrt{\frac{2\log n}{n_j}}$. And the arm with largest index is selected to be the arm to be played in the next round.

It is proved that when the number of rounds is T, then the upper confidence bound algorithm can achieve a regret bound of $O(\log T \cdot \sum_{i=1}^{n} \frac{1}{\mu^* - \mu_i})$. It can be observed that if we replace a sub-optimal arm with another better sub-optimal arm (μ_i is replaced with μ_i' but still $\mu_i' < \mu^*$), then the regret bound may increase. This seems a little bit weird but actually makes sense. The intuition could be: if the arms are getting close, then we

need more bad samples to distinguish it from optimal one, and during which we would suffer more loss, i.e., more regret. And since the distinguishing-complexity is $O(\frac{1}{\epsilon})$ and the one-round regret is $O(\epsilon)$, then the total regret would be of the order $O(\frac{1}{\epsilon})$.

Various interesting generalizations are raised in the past 10 years, and make multi-armed bandit no longer a single problem but a "flavor" or a "style" of the problems. An important generalization towards the classic multi-armed bandit is Combinatorial Multi-Armed Bandit problem (CMAB), proposed by Wei Chen et al in[2]. It included combinatorial problem into multi-armed bandit problem. In other words, it set up the framework for multi-armed bandit version of combinatorial problem. As a brief introduction, every round a combination of arms (called a super arm) is played and the sum, as well as individual values of sampled reward of its base arms are revealed to the player. The objective for the player is still the regret function. But in CMAB setting some modification is necessary. For example, sometimes the essential combinatorial problem is NP-hard, and therefore cannot be solved efficiently. In this case a new notion called (α,β)-regret is introduced in[2], where α represents the approximation ratio and β represents the confidence probability.

A similar algorithm *Combinatorial Upper Confidence Bound* algorithm (CUAB) is proposed in[2] for this problem. This algorithm utilizes an offline oracle solving (approximately) the essential combinatorial problem deterministically in each round, and feed the oracle with an set of updated parameter (the updating is done in the similar way with UCB, and this is also why it is called CUCB). And it is proved that the regret it achieve is of the order $O(\log T)$, with some mild assumption of the distribution.

Another research direction of multi-armed bandit is to find algorithm that only outputs the index of the best arm, but not the accumulated regret. This problem is named the "Pure Exploration". Usually the problem is related with hypothesis testing, and some impossibility result and lower bound for sample complexity can be inherited here. Wei Chen et al studied the pure exploration problem of their previous research CMAB problem in[1], and called it Computational Pure Exploration of Multi-Armed Bandit problem (CPEMAB). The problem of combinatorial pure exploration of multi-armed bandit is the following. n base arms are given (each with its weight, or reward $w(e)$ following a certain underlying distribution ϕ_e which is required to be

R-sub-Gaussian) and a decision class $\mathcal{M} \in 2^{[n]}$, indicating the solution range that we could output from, in every round we are required to make a sample on one base arm (note that it is different from CMAB setting, where we are allowed to make sample on a super-arm in a round) and then output a $S \in \mathcal{M}$ such that the total expected reward $w(S) = \sum_{e \in S} w(e)$ is maximized with certain probability.

Two types of optimization criterion is discussed in[1]: Fixed budget setting and fixed confidence setting. In the fixed budget problem the number of rounds are specified before the player started making samples. In the fixed confidence setting, we need the algorithm to output correctly (output the index of the ground-truth optimal arm with certain probability). And the objective is also clear, in the fixed budget setting, the objective is to maximize the probability of outputting correctly. In the fixed confidence setting, the goal is to design algorithm that has low sample complexity.

In the paper[1], a further generalization of UCB algorithm is proposed to the fixed confidence setting of CPEMAB problem. It is called Combinatorial Lower-Upper Confidence Bound algorithm (CLUCB). It utilize an offline oracle like CMAB (two times every round), for selecting which base arm to probe next. The algorithm can be roughly described in the following box.

Algorithm 5 CLUCB Algorithm

Offline Oracle outputs $M = \text{argmax}_{S \in \mathcal{M}}\{w(S)\}$

For $t \geq 1$, maintain $\bar{w}_t$;

$\bar{w}_t \rightarrow Oracle \rightarrow M_t$;

Perturb $\bar{w}_t$ to get $\tilde{w}_t$ (according to M_t), then $\tilde{w}_t \rightarrow Oracle \rightarrow \tilde{M}_t$;

if $\tilde{w}(\tilde{M}_t) = \tilde{w}(M_t)$, **return** M_t; **otherwise** continue;

The key lemma for the correctness of CLUCB algorithm is called *interpolation lemma*, which makes the best use of the definition of Δ_e, the gap between the optimal set and the sub-optimal set with opposite choice of e, and eventually make concrete connection between the property of oracle and the sampling error. However, this what we called "transform" (transforming from an offline oracle to an online pure exploration algorithm) only works for those "exact oracles" (a oracle that outputs exactly the optimal solution for the underlying combinatorial problem), and we may encounter some problems when attempting to design a similar transform for combinatorial prob-

lems with only approximation oracles instead of exact ones. And the reasons are the following.

On one hand, the proof for "exact-CLUCB" is "additive", by which we mean that every inequality and the application of interpolation lemma is manipulating $\pm\epsilon$ terms with $\bar{w}$ and $\tilde{w}$, and the correctness of the output is proved by contradiction of former setting of optimality. However, for α-approximation algorithm this does not work that well, since we are given an approximation oracle and required to output an approximation solution. It is meaningless to compare the output of the approximation oracle with the optimal solution and thus the similar argument in the proof may collapse.

On the other hand, if we want to design a similar algorithm for approximation solution, the stopping criterion might not be satisfied since the output of the oracle can be any approximately good solution, while in the original CLUCB algorithm this problem could be overcome by an "exact" oracle. Probably we should modify the algorithm so that these difficulty is overcome, for example the approximation ratio can be lower down.

For other related research done in the pure exploration problem. Certain complexity lower bounds are studied in[5]. Mannor et al proved a lower bound of order $O(\frac{n}{\epsilon^2}\log\frac{1}{\delta})$ sample complexity of PAC-learning algorithm (PAC learning algorithm requires the algorithm to output an ϵ-additive solution with probability larger than $1-\delta$), where n is the number of arms. Besides, a weaker bound $O(\frac{1}{\epsilon^2}(n+\log\frac{1}{\delta}))$ is proved in the Bayesian setting. The technique comes from classical techniques probability theory and hypothesis testing.