骰子下的命运游戏——随机过程在大富翁游戏中的概率分析与策略应用

姓　　名: 何啻
学　　号: 37720222204921
院　　系: 数学科学学院
专　　业: 统计学
年　　级: 2022 级
课程教师: 方明
日期: 2024 年 6 月 1 日

引言与摘要

大富翁作为曾经家喻户晓、全球最受欢迎的桌游之一，同时也是一个充满着运气和不确定性的“随机过程”，在一定条件的简化下可以很好地作为载体来分析和应用不少随机中的定理与结论。本文基于最早“Monopoly”版本的规则，对 Richard Durrett《应用随机过程》[1] 中的部分例题进行详细地补充和推导，并结合部分 Ross 同名著作中的定理，进一步对这个游戏中其他有趣的性质提出问题并尝试分析与讨论。笔者在能力有限下，希望能从这个游戏不同的角度，将本门课程的不少理论串在一条线上综合地实践运用。

对于**问题 1**，为了计算“超过起点步数”的极限分布，本文基于马尔科夫链建立一个概率转移矩阵，从直观的条件概率和稍微抽象的更新过程角度来分析，用两种不同的视角求得一致的结果。对于前者，本文利用**多步转移递推公式**，计算越过起点前可能位置的到达概率分布，然后利用条件概率的一步转移计算越过起点后的分布；后者我们引入**更新过程**中的“年龄”和“剩余寿命”概念，证明存在并计算了“剩余寿命”的平稳分布并等价地推导出“超过起点步数”的极限分布。

此后，本文将“先到达指定格子”的问题转化为计算**离出概率**、**离出时间**的一般性问题，利用矩阵乘法的形式推导出该类计算的通解，然后在具体的该问题中求得结果，并在最后采用蒙特卡洛模拟的方法计算了“访问过所有格子”的平均轮数。

对于**问题 2**，为了获得离散的轮数视角下，长远轮数时玩家位置的**极限分布**，我们先证明了**离散时间马氏链**中**平稳分布**的存在性和唯一性，然后通过极限逼近和解方程两种方法详细推导并交叉验证了得到平稳分布的一致性，并在对其进行排序后得出了最高访问概率格子编号，在分析后给出了游戏中购买地产的策略建议。

为了进一步获得连续的时间视角下，长远时刻三个玩家位置的极限分布，本文引入**封闭排队网络**，将“格子停留时间”映射为服务台“服务时间”，推导了三个玩家在地图上的联合概率分布，并在最后说明了利用**吉布斯抽样**的简化运算技巧。

对于**问题 3**，本文在简化游戏为“三人赌博”的财富转移模型后，在“相对公平”条件下构造了一个两阶段的**鞅**，然后在证明充分条件后利用**可选停时定理**，对“出现第一个玩家破产时间 τ”、“游戏结束时间 T”进行了上下界估计并通过随机模拟来验证估计的效果。最后，本文在更特殊的“绝对公平”条件下证明了先前的鞅此时在两阶段下的连贯性，并再次利用可选停时定理对“停时 T”进行直接估计。

我们说明，在本文中前半部分是对书上的例题详细的补充、推导，并通过可视化分析来辅助“直觉”的建立，后半部分利用了更特殊的定理来解决新问题和分析更有趣性质。为了涵盖了不少本学期学到的理论，在有限能力下难免有“先射箭后画靶”的牵强与不自然，只希望能在稍微符合逻辑的情况下，浅薄地锻炼“应用随机过程”的能力，望老师包涵文中出现的错误与纰漏！

关键词：更新过程　离出分布　极限/平稳分布　封闭网络　吉布斯抽样　鞅　可选停时定理

目录

1 游戏背景和规则假设

1.1 “大富翁”游戏背景

“大富翁”游戏于 1935 年由美国的查尔斯·道罗创造，最初名为“土地之主”（The Landlord's Game）。道罗设计这个游戏的初衷是为了展示地产投机的危险和对资本主义的讽刺。随后，经过一系列的发展和改进，该游戏于 1935 年正式发布，改名为“大富翁”（Monopoly）。自此之后，大富翁便成为了全球最受欢迎的桌上游戏之一，受到了广泛的认可和喜爱。

在大富翁游戏中，玩家扮演地产投资者的角色，目标是通过购买、交易和发展地产，最终使其他玩家破产，从而成为唯一的胜利者。游戏的核心机制包括购买地产、收取租金、建立房屋和酒店以增加租金收入、交易地产和支付各种费用（如税金和罚款）。此外，游戏还包括一些特殊的机会和命运卡，它们会为玩家带来意想不到的好处或挑战，增加了游戏的变数和趣味性。

在该游戏过程中，由于玩家掷骰子造成的移动到达的不同特殊地点会带来对应的特殊效果，某些机会和命运卡也会带来多种多样复杂的效果，为了便于分析，我们选取大富翁游戏的初始版本“Monopoly”作为蓝本，通过一些简化的规则或特定关注某个部分时增加、修改的规则，借此讨论和分析不同情况下随机过程的直观效果。

1.2 游戏规则介绍

为了便于讨论一般化的情景，我们采用一套简化后的规则，在下文中不额外说明补充的前提下，大部分情景都一致采用如下规则 [2]：

- **游戏流程：**

  1. **游戏开始** 三位玩家每人有 1600 元作为资产。
  2. **掷骰** 每位玩家每轮开始时，同时掷出两颗骰子，并向前行走骰子点数之和。
  3. **购买地产** 玩家到达无人拥有的地产（格子），默认将在资金允许的情况下购买地产。
  4. **过路费** 当某一玩家回合结束的落点在其他玩家的地产上，需要向对方缴纳过路费 200 元，相应也会在每一回合收到由其他玩家触发并缴纳的过路费。
  5. **破产** 当某轮中，玩家需要缴纳过路费但资金不足时，将缴纳剩余所有的资金并宣告破产退出游戏。

- **地图设置：**

  如图 (1) 所示，地图为边长 11 个格子的正方形轮廓，总共 40 个格子并从起点 GO 开始依次映射为 $0 \sim 39$ 的数字便于后续讨论。其中四个角落之外的、没有颜色（即非 CC 或 CH）的格子为普通的地产，我们事先将他们视作一般性的相同作用的格子，接下来着重介绍四个角落的特殊格子和带颜色的 CC（获得宝箱卡）、CH（获得机会卡）。

  - **GO**，玩家出发的起点，左上角
  - **G2J**，监狱传送门，当玩家该回合落在此格子，将会被传送至 JAIL。
  - **JAIL**，监狱，当玩家该回合结束落在此格子（包括从 G2J 传送来的情况），将会被关进监狱，接下来两轮将无法行动。

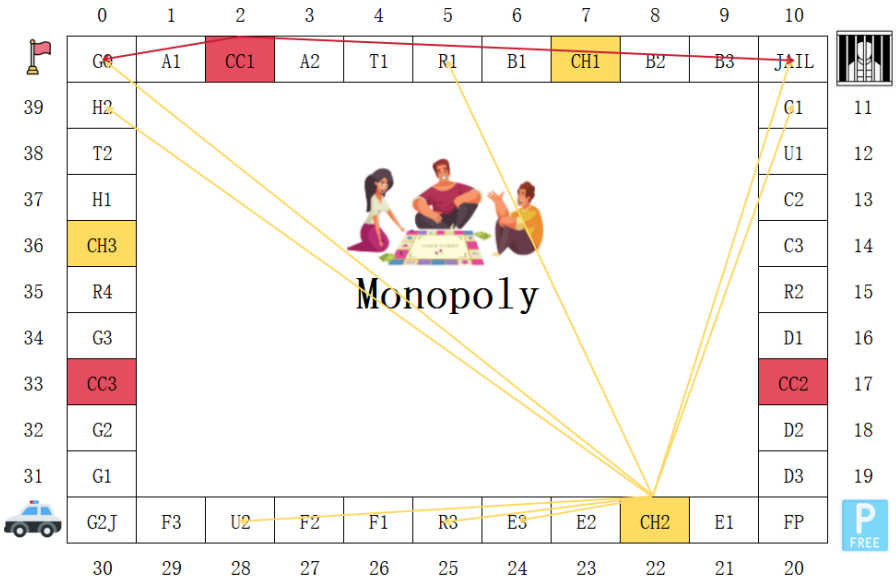


图 1: 游戏地图

– **FP**，FreePark，免费停泊地，当玩家该回合结束落在此格子，不会发生任何事情。

对于到达 **CC 格子**和 **CH 格子**将分别获得的宝箱卡和机会卡，我们暂时只关注会带来位移作用的卡片效果 **(宝箱卡 2/16 张, 机会卡 10/16 张)**，例如“移动到监狱 JAIL”. 并忽略其他类型卡片的效果。我们在上图中以 CC1 和 CH2 为例，一并展示了所有可能的位移效果，其余的具体效果和概率在后续转移矩阵的构造和表（1）中给出。

1.3 游戏模型假设

- 每名玩家行动相互独立，即每轮前进步数仅由掷骰子的点数决定。
- 正常游戏中获得机会卡和宝箱卡需要从牌堆中抽卡，为了实现马尔科夫性的构造，我们视作每次抽卡为有放回和重新洗牌的抽取。

2 游戏的问题提出

在大富翁游戏中，我们关心并提出以下问题：

问题 1 对于玩家来说，他们希望估计自己在若干轮后所处位置的概率，从而对自己的资产进行合理的规划，有如下情况：

- 考虑一个简化的规则，对于所处某位置 i 的玩家 A，如果只通过掷骰子移动，在相当多轮后他在超过上一轮位置 i 时，超过步数的数值分布是多少?
- 如果玩家位于格子 i，在到达格子 10（监狱）前到达格子 0（起点）的概率是多少? 大概要过几轮?
- 假设游戏刚刚开始，到所有格子都被三位玩家访问并购买时，大概过去了多久?

问题 2 长远来说，当游戏进行到相当多轮的时候玩家会出现在哪个位置？我们可以从不同的视角建立模型来考虑这个问题：

- 从离散的轮数角度来看，在某轮中玩家停留在每个格子的概率分布如何？如果当前位于格子 i，大概需要多少轮才能重新回到这里？
- 从连续的时间角度来看，在相当长时间后，三个玩家停留在地图上的位置有怎样的概率分布？

根据这一结果，玩家应该制定怎样的购买地产策略来使得自己尽可能获利？

问题 3 我们开始考虑玩家资金转移的过程，我们希望知道对于初始资金均为 K_0 的三位玩家，给定地图上某一种地产的分布情况，当有一位玩家因为破产退出游戏需要多少时间？当游戏结束，即出现两位玩家破产时，又需要多少时间？

3 随机过程模型的建立和求解

3.1 问题一

3.1.1 构建马尔科夫转移矩阵

在这个游戏中，我们将第 n 轮所处的位置记为状态 $X_n(n = 0, 1, 2, ...)$，状态空间恰好为地图上 40 个格子，从起点 GO 开始依次映射为 $0 \sim 39$ 的数字，即 $S = \{0, 1, 2, \ldots, 39\}$, 对任意一组状态序列 $j, i, i_{n-1}, \ldots, i_0$，我们将

$$P(X_{n+1} = j \mid X_n = i, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0)$$

称作在给定 $X_{n-1}, \ldots, X_0$ 状态下，从第 n 轮位置 $X_n = i$ 在下一轮转移到 $X_{n+1} = j$ 的概率，我们容易知道在这个游戏中马尔科夫链的条件独立性和时间齐次性是可以满足的。我们通过考虑游戏中通过掷骰子、机会宝箱卡和特殊格子，使得当前位置转移到新位置的各种情况：

(1) **掷骰子**

玩家每轮通过两个点数为 1 6 的独立骰子投出的点数之和来移动，对于两个独立的离散均匀分布，分别设为随机变量 X 和 Y，我们采用卷积公式计算它们的和 $T = X + Y$ ：

$$P(T = t) = \sum_{x=1}^{6} P(X = x)P(Y = t - x)$$

其中，$P(X = x)$ 和 $P(Y = y)$ 是均匀分布的骰子的概率质量函数，以 X 为例：

$$P(X = i) = \begin{cases} \frac{1}{6} & if: i \in \{1, 2, 3, 4, 5, 6\} \\ 0 & otherwise \end{cases}$$

(2) **机会卡和宝箱卡** 这一部分见表（1）我们在这里略去该表对应具体的格子编号。

(3) **特殊格子**格子 30：G2J，将直接到达格子 10：JAIL

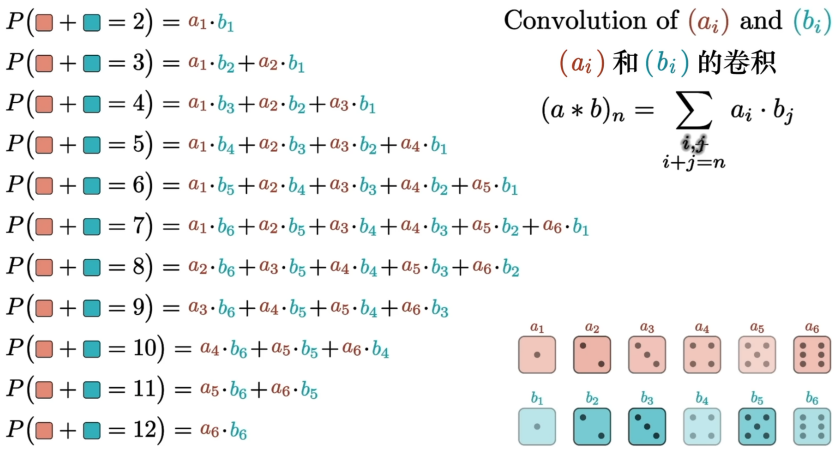


图 2: 两骰子点数和的分布

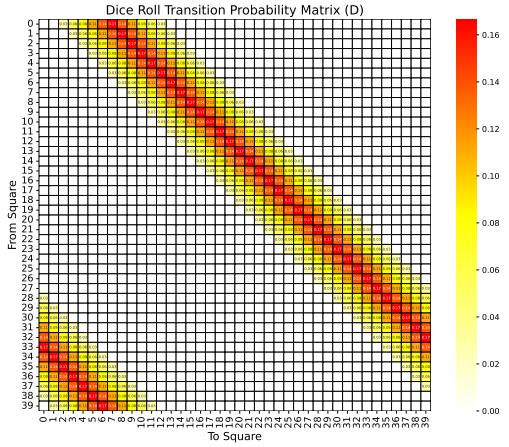
类型	卡片	概率
2* 宝箱卡 (CC)	回到起点 GO	$\frac{1}{8}=12.5\%$
	进入监狱 JAIL	$\frac{1}{8}=12.5\%$
10* 机会卡 (CH)	回到起点 GO	$\frac{1}{16}=6.25\%$
	进入监狱 JAIL	$\frac{1}{16}=6.25\%$
	移动到 C1	$\frac{1}{16}=6.25\%$
	移动到 E3	$\frac{1}{16}=6.25\%$
	移动到 H2	$\frac{1}{16}=6.25\%$
	移动到 R1	$\frac{1}{16}=6.25\%$
	移动到下一个 R	$\frac{1}{8}=12.5\%$
	移动到下一个 U	$\frac{1}{16}=6.25\%$
	后退三步	$\frac{1}{16}=6.25\%$

表 1: 卡牌的转移概率表

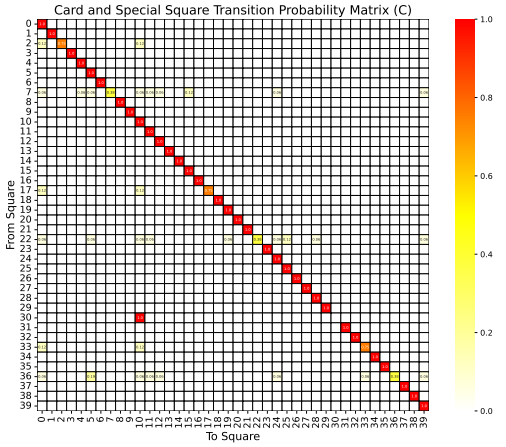
我们由此计算出格子 i 转移到 j 的概率 $p(i,j)$，并由此构建出 40*40 的转移概率矩阵 P，为了方便直观看到转移的情况，我们将其拆成“掷骰子转移矩阵 D”和“抽卡转移矩阵 C”，对于一轮后最终结果的转移矩阵 P，显然有：

$$p_{ij}=\sum_{k=0}^{39} d_{ik}\cdot c_{kj} \tag{1}$$

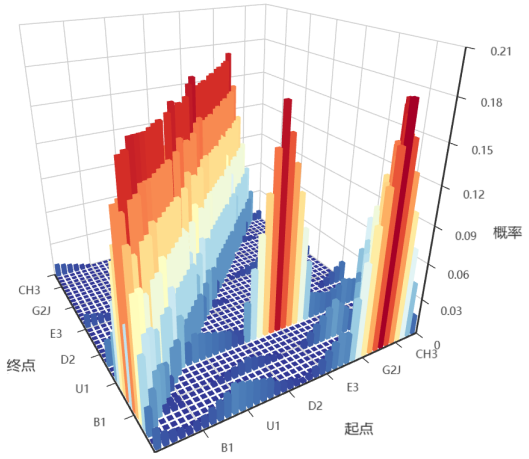
其中 d_{ij} 为掷骰子从 i 移动 j 的概率，c_{ij} 为因为抽卡或特殊格子从 i 移动 j 的概率，则转移矩阵之间相应形式 $P=D\cdot C$。在此特别说明，由于格子 30(G2J) 由于立刻转出不会被到达，理论上我们可以直接删去该状态，但为了后续作图和解释的连贯性我们仍然保留，不过之后我们将提到的“不可约性”是不考虑该格子而言的。我们画出每个转移矩阵相应反应概率值大小的热图（图3），我们可以对每次转移的情况做一个预览：



(a) Dice Roll Transition Probability Matrix (D)



(b) Dice Roll Transition Probability Matrix (C)



(c) 3D figure of Dice Roll Transition Probability Matrix (P)

图 3: Comparison of Dice Roll Transition Probability Matrices

3.1.2 问题 1.1: 计算超过固定位置的步数分布——递推与更新过程

这个问题下我们只考虑简化的规则，即只关注所有格子属性等价的一般情况，我们不妨将“某个位置”设定为“起点”，由普遍性可将这个概率分布推广到任意的格子。

- **解法 1：n 步转移递推**

  由于骰子的点数和在 2 到 12 之间，故超出起点的步数取值范围为 $0, 1, 2, \ldots, 11$，这些位置也仅可能由起点前 12 步内的位置一步转移而来，为了先获得这前 12 步位置的到达概率，我们定义向量：

$$\pi_n = \begin{pmatrix} P_{n-11} & P_{n-10} & P_{n-9} & \ldots & P_{n-1} & P_n \end{pmatrix}^T \quad \|\vec{\pi_n}\| = 12$$

  为包括位置 n 和后 12 步的到达概率分布,P_n 定义为恰好走到位置 n 的概率，相应设定初始到达概率分布：

$$\pi_0 = \begin{pmatrix} P_{-11} & P_{-10} & P_{-9} & \ldots & P_{-1} & P_0 \end{pmatrix}^T$$

  这里的初值 $P_{-11} = P_{-10} = \ldots = P_{-1} = 0$ $P_0 = 1$ 是设定的位置 0 前 11 个位置到达概率，

由于我们定义了行走的正方向，显然这 11 个位置的到达概率都为 0，而因为从起点出发，已经访问了起点，故而 $P_0 = 1$。

大富翁地图是一个 40 格子的循环，我们可以将每一圈拉直成一条线，超过起点后的位置编号将变成 40、41、42、... 不再重置，那么这个问题等价为：从位置 0 出发：

- 短步数内，位置 $40n-1$ 前 12 步的概率分布为 π_{40n-2}。
- 长远来看，n 充分大时，这 12 步的极限分布 π。

我们可以断定这个极限分布存在，并 $\lim_{n\to\infty} \pi_{40n-2} = \pi$ 收敛得到。

我们先来计算 π_{40n-2}，根据两枚骰子点数和 T 的概率分布，我们可以写出递推方程和递推转移矩阵：

$$\begin{aligned} P_n = &\frac{1}{36}P_{n-12} + \frac{2}{36}P_{n-11} + \frac{3}{36}P_{n-10} + \frac{4}{36}P_{n-9} + \frac{5}{36}P_{n-8} + \frac{6}{36}P_{n-7} + \frac{5}{36}P_{n-6} + \\ &\frac{4}{36}P_{n-5} + \frac{3}{36}P_{n-4} + \frac{2}{36}P_{n-3} + \frac{1}{36}P_{n-2} \end{aligned} \tag{2}$$

递推转移矩阵 A 的元素 a_{ij} 的取值分类情况如下：

$$a_{ij} = \begin{cases} 1 & \text{if } j = i+1 \text{ and } i \leq 11 \\ P(T = 13-j) & \text{if } i = 12 \\ 0 & \text{otherwise} \end{cases}$$

于是矩阵形式的递推方程和超过起点的分布：

$$\pi_{40n-2} = A^{40n-2} \cdot \pi_0$$

计算可得在 $n = 117$ 时大致收敛为同一个概率 $p = 0.14285714$，近似为两骰子点数和 T 的期望倒数 $1/E(T) = 1/7$，这可以解释为在轮次足够大时恰好落到任一格子的概率均匀分散为移动步数长度期望分之一，我们在稍后更新过程的解法中同样会提到这一点。于是对于位置 $40n-1$，记 $A_n = i$ 为前第 i 个格子,$Z_n = j$ 为后第 j 个格子, 则恰好越过 $40n-1$ 访问到 $Z_n = i$ 的概率为：

$$\begin{aligned} P(Z_n = j, A_n = i) &= \sum_{i=1}^{12} P(Z_n = j, A_n = i \mid \text{“}reach\ i\text{”}) \cdot P_{40n-1-i} \\ &= \sum_{i=1}^{12} P(T = i+j) \cdot p \\ &= P(T > j) \cdot p \end{aligned} \tag{3}$$

这与**解法 2** 的最终公式是等价的，我们在最后来展示这个分布。

- **解法 2：更新过程-年龄与剩余寿命**

我们令 $t_1, t_2, \cdots$ 表示独立同分布的每轮前进步数（两骰子点数和），$T_n = t_1 + \cdots + t_n$ 表示第 n 轮时前进的总步数, $N(t) = \max\{n : T_n \leqslant t\}$ 为到位置 t 之前发生的总轮数. 设

$$A(t) = t - T_{N(t)} \quad \text{且} \quad Z(t) = T_{N(t)+1} - t$$

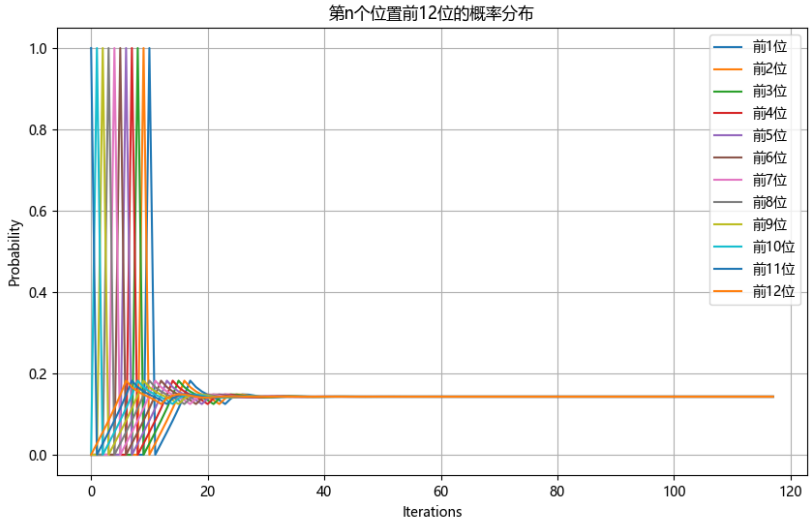


图 4: 位置 n 前 12 个位置概率分布

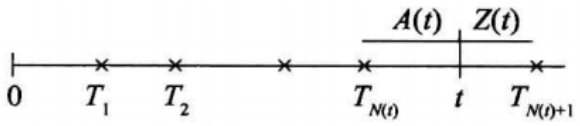


图 5: 年龄与剩余寿命

$A(t)$ 表示位置 t 离第 $N(t)$ 轮停留位置的距离（年龄），而 $Z(t)$ 表示位置 t 离第 $N(t)+1$ 轮停留位置的距离 (剩余寿命), 从这个具体的例子可以看出, 远离第 $N(t)$ 轮停留位置的步数在剩余寿命链中为 $j, j-1, \cdots, 1$, 而在年龄链中是 $1, 2, \cdots, j$, 因此有

$$\frac{1}{12}\sum_{t=1}^{n} P\left(A_t = i\right) = \frac{1}{12}\sum_{t=1}^{n} P\left(Z_t = i\right)$$

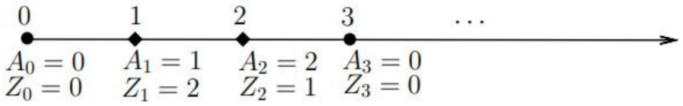


图 6: 年龄与剩余寿命的对应性

由这一对应性，我们只需要关注 Z_t 即可。根据定义，当 $Z_t = 0$ 时, 刚进行过一轮投骰子. 如果到下次投骰子得到的点数和为 k, 那么对于 $Z_t = 0$ 来说，在下一步 $t+1$ 时的“剩余寿命” Z_{t+1} 忽然以概率 $P(t_1 = k)$（即下一轮要走的点数为 k 的概率，由于投骰子之间独立同分布，不妨取第一个过程 t_1）获得 $k-1$，即 $Z_{t+1} = k-1$，然后逐步递减直到再次归零时，即下一轮投掷完成移动。

我们可以看到，引入 Z_t 本质上是将一次投骰子跳跃 k 步的过程，具象化成一步一步的细致过程：用一步获得 $k-1$ “剩余寿命”，用 $k-1$ 步不断减少的更新 Markov 链，Z_t 在一

个有限状态空间 $S = \{0, 1, 2, \cdots, 11\}$ 内，并对应转移概率:

$$\begin{aligned} &p(0, j) = P(t_1 = j + 1) && 1 \leqslant j \leqslant 11 \\ &p(i, i - 1) = 1 && i \geqslant 1 \\ &p(i, j) = 0 && \text{其他情形} \end{aligned}$$

在这个链中我们本质上还是要求它的平稳分布，我们先给出 Richard Durrett《应用随机过程》书中平稳分布存在的充分条件，然后一一验证他们:

- **定理 1.20** 若该链不可约、常返，则该链存在一个平稳测度 $\mu(i) > 0, \forall i$。
- 补充：若 $\sum_i \mu(i) < \infty$，则可归一化为一个平稳分布。这个条件可以等价为该链有正常返的性质，又由于不可约性，只需要验证其中一个状态正常返即可。

验证：

step a: S 是有限不可约的，因为从 0 可以一步转移到任何状态（除了 1，但 1 可由 $0 \to 2 \to 1$ 到达），从任何数也一定逐步返回到 0。常返在不可约的假设下，找到一个点是常返的，由可达性其余状态也相应常返。在这个链里 0 显然常返，这就完成了证明前半部分。

step b: 对于补充部分，在这个例子中恰好状态 0 的返回代表了一次过程的更新，即掷一次骰子，所以

$$E(T_0) = E(t_1) = \sum_{k=2}^{12} k \cdot P(t_1 = k) = 7 < \infty \tag{4}$$

$E(T_0)$ 是 $Z_t = 0$ 返回的期望时间，$E(t_1)$ 是一次投掷的点数和 t_1 的期望步数，故 0 正常返成立，平稳分布存在性得证。

计算：

下面我们来求具体的数值分布，我们将使用“循环技巧”，由

$$\mu(i) = E_0(\sum_{t=0}^{T_0 - 1} I(Z_t = i)) \tag{5}$$

定义了一个平稳测度, 其中 I 是一个示性函数，T_0 是 $Z_t = 0$ 返回需要花费的期望步数，也就是一般更新过程中的平均返回时间，这里额外说明是因为在投骰子这个过程中，原来“时间”的概念都变成了“步数”，而 $E(T_0)$ 代表的花费的平均步数和“剩余寿命” Z_t 取值的步数在定义上稍微有点不一样。在这个式子中为什么累加的上限是 $T_0 - 1$，是因为最后一步必须得用于回到 0。由于我们只关注一次掷骰子的小片段，所以我们不妨将这个片段看作第一次掷骰子，这是为什么 t 是从 0 开始的，这个式子表示的就是一次从 0 开始的更新/循环在步数 $0, \ldots, T_x - 1$ 中访问 i 的期望次数。

在这个例子中，由于 Z_t 从 0 一步跳跃出去，在返回到 0 之前至多访问某个状态 i 一次, 这个事件发生当且仅当第一步 Z_1 转移到达某个状态 $\geqslant i$, 即 $t_1 > i$。因此方程 (5) 定义的平

稳测度为：

$$\begin{aligned}\mu(i) &= P(Z_1 \geqslant i) \\ &= \sum_{j=i}^{12} p(0,j) \\ &= \sum_{j=i}^{12} P(t_1 = j+1) \\ &= P(t_1 > i)\end{aligned}$$

由于验证的平稳分布存在性，我们设平稳分布 $\pi(i) = c \cdot \mu(i)$，其中 c 为归一化常数，并通过概率和为 1 的约束求出 c：

$$\begin{aligned}\sum_{i=0}^{11} \pi(i) &= 1 \\ c \cdot \sum_{i=0}^{11} \mu(i) &= 1 \\ c \cdot \sum_{i=0}^{11} P(t_1 > i) &= 1 \\ c \cdot \sum_{i=0}^{11} \sum_{k=i+1}^{12} P(t_1 = k) &= 1 \\ c \cdot \sum_{k=1}^{12} \sum_{i=0}^{k-1} P(t_1 = k) &= 1 \\ c \cdot \sum_{k=1}^{12} k \cdot P(t_1 = k) &= 1 \\ c \cdot E(t_1) &= 1 \\ c &= 1/E(t_1)\end{aligned}$$

于是平稳分布最终可以写成：

$$\pi(i) = P(t_1 > i)/E(t_1) \tag{6}$$

回到大富翁游戏本身，掷骰子的部分我们已经将其定义为了 t_i，那么“起点”在长远上对应的其实就是一个总步数 t，相应的在 t 前后最近的一次掷骰子移动前位置 $T_{N(t)}$ 和移动后位置 $T_{N(t)+1}$，就可以去构造对应的“年龄”和“剩余寿命”，后者就是在 $T_{N(t)}$ 掷骰子后超过起点的步数。值得注意的是，此时我们将原来“固定”的起点当做在过程 t_i 中一个随机的位置，而移动前、后位置 $T_{N(t)}$ 和 $T_{N(t)+1}$ 反而看起来是固定的，这一点主要是在足够多轮下的极限下才能成立，这也是为什么“剩余寿命”的平稳分布对应着超过步数的极限分布。

解法 2 详细说明定义和证明花费了不少时间，尽管最后应用起来是非常自然而简单的。回顾**解法 1** 中的结果（3）：

$$P(Z_n = j, A_n = i) = P(T > j) \cdot p$$

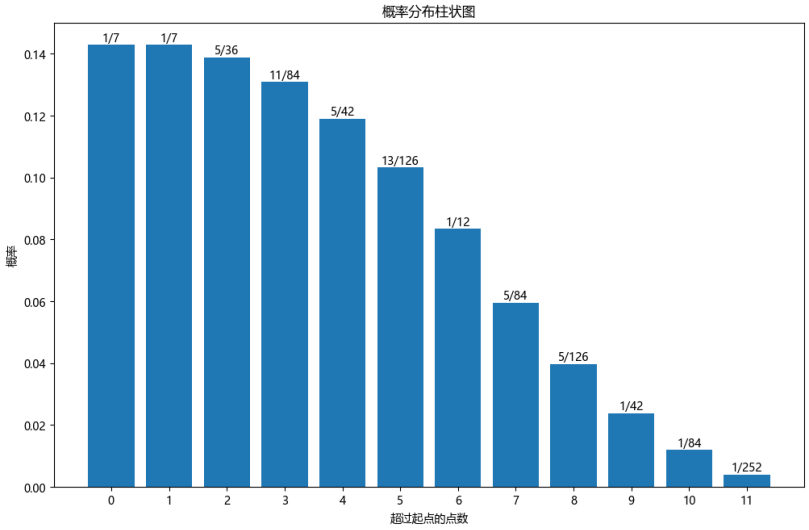


图 7: 超出起点步数的概率分布

我们之前已经提到 p 是 $1/E(t_1)$ 的近似，T 是两骰子点数和的随机变量，尽管在两个解法中我们对 Z_n 的定义有所不同，但最终他们所表示的“超出起点的数值分布”在结果上是一致的，这个结果也惊奇的让人发现，这个分布与当前位置无关，与一圈有多少个格子也无关，完全取决于“骰子点数”这么一个随机变量。

我们重新回过头来思考，为什么在关注一次投骰子的过程中，要引入拆分成“剩余寿命”和对应更新链，而不是直接注意最直观的两骰子点数和 T 的概率分布?

这是因为我们实际将无数个投骰子后移动的更新过程拆成了小段的“投骰子过程”，并只通过这个小段，利用离散马尔科夫链的性质来导出“投出两骰子的点数和”“获得 $k-1$ 的剩余寿命””超出起点位置的步数”这三个互相推导的定义，进而利用前俩个在离散马尔科夫链中可以计算得到的平稳分布，来得到第三个定义也就是目标的概率分布。

对比**解法 1** 和**解法 2**，前者更关注固定起点这个概念，通过起点前 12 个位置的极限分布，用掷骰子的一步转移概率来计算超过起点的数值分布；后者更关注投骰子这个过程本身，起点所代表的 t 反而成为在这个过程中随机的一个位置。综合来看，**解法 1** 更符合直观的逻辑，构造简单，**解法 2** 则更为精妙，较为深刻地展现了随机过程通过一个局部片段来研究长远性质的特点，在熟悉对应概念的情况下可以快速推广到类似的情形。

3.1.3 问题 1.2 先到达“起点”的概率与平均轮数——离出分布

这个问题下我们回到考虑监狱、特殊卡片的更真实的大富翁，我们先一般化要到达的目标格子为 $R=\{r_1,r_2,\cdots,r_m\}$，讨论最先到达其中某个目标格子的概率。不难发现，如果把这些目标格子作为吸收态，那么这个问题其实本质上就是一个朴实的计算离出概率、离出时间的步骤，我们先回顾

状态空间分解定理 若状态空间 S 是有限的，则 S 可以写为互不相交集合的并 $T\cup R_1\cup\cdots\cup R_k$，其中 T 是非常返状态组成的集合，$R_i(1\leqslant i\leqslant k)$ 为常返状态组成的不可约闭集。

我们将目标格子 $R=\{r_1,r_2,\cdots,r_m\}$ 对应的第 $r_1,r_2,\cdots,r_m$ 行的转移概率清空，并使

$p(r_i, r_i) = 1, i = 1, \cdots, m$，于是这里 R 显然就是常返态，剩余的格子就是非常返态 T。然后我们将转移矩阵 P 通过初等变换行列顺序：

$$P \to \begin{pmatrix} r_1 & 0 & \cdots & 0 & 0 \\ 0 & r_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & r_m & 0 \\ C_1 & C_2 & \cdots & C_m & T^* \end{pmatrix}，\text{记吸收态部分} I_{m\times m} = \begin{pmatrix} r_1 & 0 & \cdots & 0 \\ 0 & r_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_m \end{pmatrix}$$

由于每个吸收态只对应一个格子，所以这里的 $r_i = 1, i = 1, \cdots, m$。另外部分中，$T^*_{t\times t}$ 为非常返态 T 之间的转移概率矩阵，即目标格子之外的格子之间的转移概率矩阵；记 $C_{t\times m} = (C_1, \cdots, C_m)$，即非常返态进入吸收态的转移概率矩阵 $(t \times m)$，维数之间有 $t + m = K = 40$。

我们不加证明的说明，记 $V_{r_i} = \min(n \geqslant 0, X_n = r_i)$，有

- $h^{r_i}(x) = P_x(V_{r_i} = \min(V_{r_j}, j = 1, \cdots, m)$ 为从 x 最先到达 R 中 r_i 的离出概率
- $g(x) = \mathbb{E}_x[V_R]$ 为从 x 到达 R 中的离出时间

对应一步转移的递推式：

$$\begin{cases} h^{r_i}(x) = \sum\limits_y p(x, y) h^{r_i}(y) \\ g(x) = 1 + \sum\limits_y p(x, y) g(y) \end{cases} \tag{7}$$

将两式都分离出常返和非常返的部分分别到等号左右，并对 $\forall x \in T, \forall i \in (1, \cdots, m)$ 扩充成 $t \times m$ 的矩阵 $H(\vec{x}) = (h^{r_1}(\vec{x}), \cdots, h^{r_m}(\vec{x}))$ 和 $t \times 1$ 的列向量 $G(\vec{x})$，则有方程（7）矩阵形式（注：E 是 $t \times t$ 单位阵，η 是 $t \times 1$ 元素全为 1 的列向量）：

$$\begin{cases} (E - T)_{t\times t} \cdot H(\vec{x})_{t\times m} = C_{t\times m} \\ (E - T)_{t\times t} \cdot G(\vec{x})_{t\times 1} = \eta_{t\times 1} \end{cases} \to \begin{cases} H(\vec{x})_{t\times m} = (E - T)^{-1} C \\ G(\vec{x})_{t\times 1} = (E - T)^{-1} \eta \end{cases}$$

于是 $H(\vec{x})$ 和 $G(\vec{x})$ 相应位置的数值就是我们想要的关于方程 (7) 的解，其中

- $H(\vec{x})$ 的第 i 行 j 列元素 $H_{ij} = h^{r_j}(i)$，即从非常返态 T 中第 i 格子出发，进入吸收态 R 中 r_j 的离出概率。
- $G(\vec{x})$ 的第 i 行元素 $G_i = g(i)$，即从非常返态 T 中第 i 个格子出发，进入吸收态 R 中的离出时间。

有了这个结论后我们回到大富翁游戏，此时目标格子 $R = \{0, 10\}$，我们可以直接计算出从任意非目标格子到达起点的离出概率和离出时间，即 $H(\vec{x})$ 的第 1 列和 $G(\vec{x})$，直接作图（8）

3.1.4 问题 1.3 访问所有格子的平均时间——覆盖/集卡问题

我们来看问题 1 最后一个小问，求从起点开始到所有格子都被访问过需要的平均轮数有点像集卡问题，尽管它貌似和问题 1.2 求得的离出时间有一点关系，但由于大富翁里循环的设置使这个覆盖问题要复杂的多，我们直接采用蒙特卡洛统计模拟的方法去求一个平均时间，图（9）是模拟的情况，由于这不是本门课的主要内容，下仅快速给出结果：

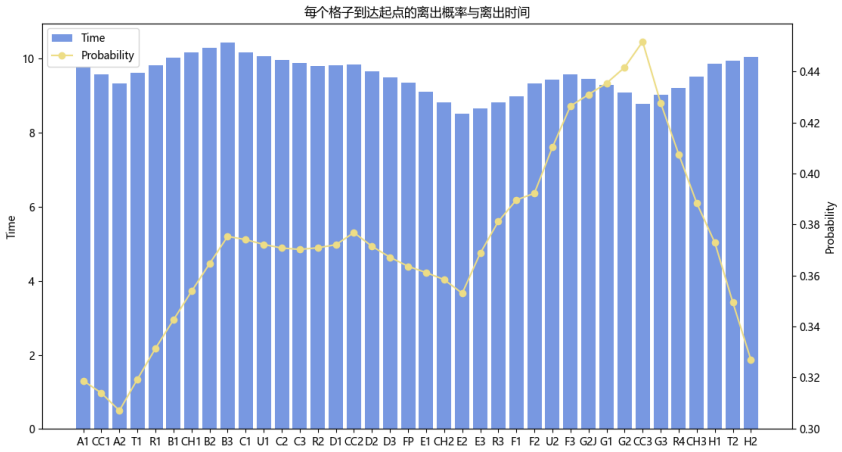


图 8: 每个格子到达起点的离出概率与离出时间

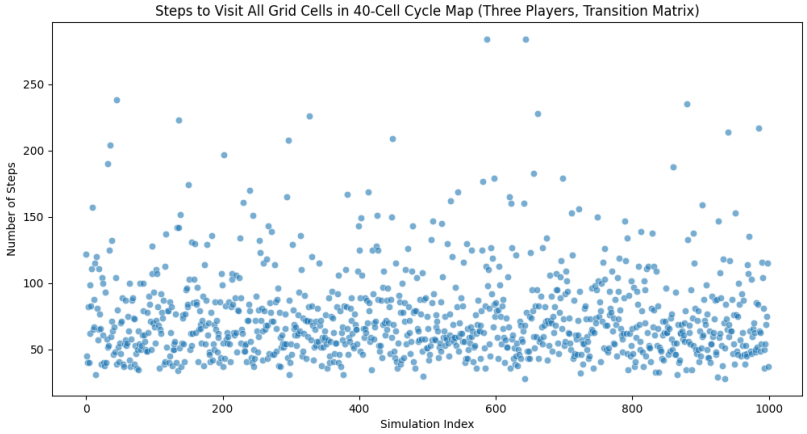


图 9: 三个玩家一起出发到访问所有格子的平均掷骰子轮数

如果考虑访问所有格子，那么 10000 次模拟求得的均值是 76.2871 轮；如果考虑是访问所有可以购买地产的格子，那么 10000 次模拟求得的均值是 56.3895 轮。

3.2 问题二

3.2.1 平稳分布、极限分布的引入

第一小问的背景其实是一个离散时间马氏链，我们给出 Richard Durrett《应用随机过程》书中的几个定理，并根据它们来推导我们最终想要的目标。

定理 1.19（收敛定理）假设该链满足不可约、非周期、存在平稳分布，则当 $n \to \infty$ 时，$p^n(x, y) \to \pi(y)$

定理 1.22（平均返回时间）若该链不可约且存在平稳分布，则平稳分布是唯一的：

$$\pi(y) = 1/E_y T_y$$

在大富翁游戏中，显然有限状态的位置之间互通且常返，不可约和存在平稳分布自然成立。

我们可以通过

(1) 转移矩阵 P 的 n 次乘积收敛后的第一行

(2) 将 $P-E(E$ 是单位阵) 的最后一列全部替换成 1 并记为 A，A^{-1} 的最后一行

来计算出平稳分布 π。方法（1）是根据收敛定理来近似的，方法（2）是直接从平稳分布的定义

$$\begin{cases}\pi \cdot P = \pi \\ \sum_i \pi_i = 1\end{cases} \tag{8}$$

出发，将第一个式子两边同减 π，然后将第二个约束条件代入，替换 $P-E$ 最后一列和等式的右边，替换后的方程等价为

$$(\pi_1, \pi_2, \cdots, \pi_n) \cdot A = (0, 0, \cdots, 0, 1)$$

等式同时右乘 A^{-1} 即得到了平稳分布 π。

我们解释为什么要做、可以做这一步替换，即书上“最后一个方程是多余”的原因：

这是因为，对单纯由平稳分布定义的等价概念：π 是矩阵 P 的一个特征值为 1 的左特征向量，$\pi \cdot (P-E) = 0$ 这个方程中 $P-E$ 是不满秩的。这是一个高等代数中的结论，即若 P 的特征值为 λ_i，$P-E$ 的特征值为 $\lambda_i - 1$，那么我们已经说平稳分布的概念其实是 π 对应矩阵 P 特征值为 1，那么 $P-E$ 自然有特征值 0，进而在 $\det(P-E) = \prod_{i=1}^{n} \lambda_i = 0$（不满秩）的情况下，$\pi \cdot (P-E) = 0$ 的解不唯一。

至于为什么可以这样替换并求得解，这是因为在平稳分布唯一时，有等价的几个概念：

$$\begin{aligned} P\text{仅有一重特征值1} &\iff P-E\text{仅有一重特征值0，秩为}n-1 \\ &\iff \pi \cdot (P-E) = 0 \text{ 有一重根子空间} \end{aligned}$$

所以在原先已有的 $n-1$ 个约束上，添加上新的一个约束条件 $\sum_i \pi_i = 1$，则 $(\pi_1, \pi_2, \cdots, \pi_n) \cdot A = (0, 0, \cdots, 0, 1)$ 的解是唯一确定的。至此，我们已经知道根据 P 计算出特征向量并归一化后就得到平稳分布，那么其实将 $P-E$ 换成 A 的目的也就很显然了：省略归一化的步骤，直接一步解出平稳分布。

对于转移矩阵 P 和平稳分布的关系，我们在这里推广一个更一般的结论。由于本质上平稳分布是在求 P 特征值为 1 的左特征向量，当方程 $\pi \cdot (P-E) = 0$ 的基础解系张成的空间为 $k < n$ 维时，意味着该链的状态空间可以分解为 k 个不交的闭集。这是因为解空间总可以通过正交化生成 k 个正交基，其归一化后就是一组在 k 个不交的闭集上各自的平稳分布，简单来说就是每个平稳分布中状态都只在一个闭集中流转。

这一结论可以怎么应用呢？我们回顾刚刚提到的 **状态空间分解定理：** 有限状态空间 $S = \{T \cup R_1 \cup \cdots \cup R_k\}$，其中 T 是非常返状态组成的集合，$R_i (1 \leqslant i \leqslant k)$ 为常返状态组成的不可约闭集。

也就是说，通过 P 计算出特征值 1 的特征向量，正交化、归一化后得到的平稳分布，我们可以直接通过 $\pi_i > 0$ 的位置判断这个平稳分布对应的不可约闭集 R_j。如果存在 π_i 在所有的平稳分布中恒为 0，那么显然这个位置对应的状态是非常返的载流态 T。通过这一性质我们可以在判断大型转移矩阵时，不需要通过繁琐的画图去分解，直接利用特征向量来对应得到。

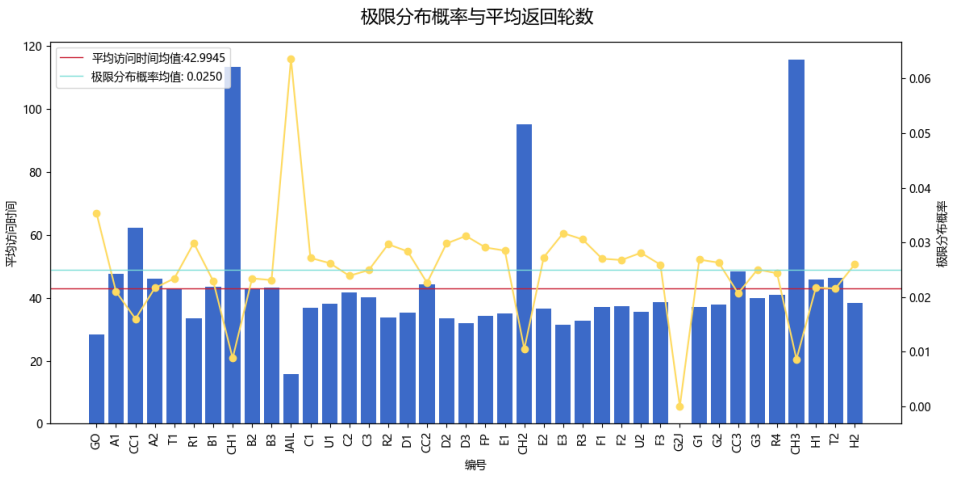


图 10: 极限分布概率和平均返回轮数

特别的，如果平稳分布不唯一时，对任意状态对应的位置 i 都存在一个平稳分布，在其中有 $\pi_i > 0$，也就是在这个链中不存在非常返的载流态 T，那么我们可以知道这个链任何时候在不可约闭集 R_j 之间是不可能互相转移的，那么就可以将它们分成若干个独立的马尔科夫链去研究各自的性质，从而简化分析。

3.2.2 问题 2.1 格子的极限分布与平均返回轮数

回到大富翁游戏，如果没有入狱、宝箱卡和机会卡，由每个格子等价轮换的性质，在不考虑破产的情况下，即始终不停止游戏，不难得出在 n 充分大时，n 轮后玩家恰好停留在每个格子的概率为 $1/40 = 2.5\%$。

添加了稍微复杂的游戏规则后，我们前置一个假设：玩家到达监狱后会在第二轮支付罚金出狱，这个情况下监狱可以简化为和到达普通的一个格子一样的规则。

我们通过两种方式最终都可计算并获得一致的结果，给出了长时间下在大富翁游戏中通过投掷移动到达不同格子的极限频率分布和每个格子平均返回轮数 (图10):

我们可以确认，这与 Richard Durrett《应用随机过程》书中 P28 的计算机模拟结果是几乎一致的，不过显然根据理论上转移矩阵的计算方法要快速得多。

其中，访问频率最高的前 3 个格子是

- 格子 10：JAIL，概率为 6.3582%
- 格子 0：GO ，概率为 3.5337%
- 格子 24：E3 ，概率为 3.1707%

其余的概率数值在附录中列表给出。

我们延用书中的部分分析结果，可以得到如下结论：

1. 最大概率的几个格子显然是因为有机会卡或宝箱卡的转移从而提高的访问概率；除去格子 30 “送入监狱” 由于每次到达将以概率 1 转移到格子 10 “监狱”，于是长期访问的概率为 0（注：故而我们在图 (10) 中删去了其平均返回轮数 ∞），另外三个最小值发生在机会格子中，显然也是因为这些格子通过掷骰子到达后有概率转移出去从而减少了停留的概率。

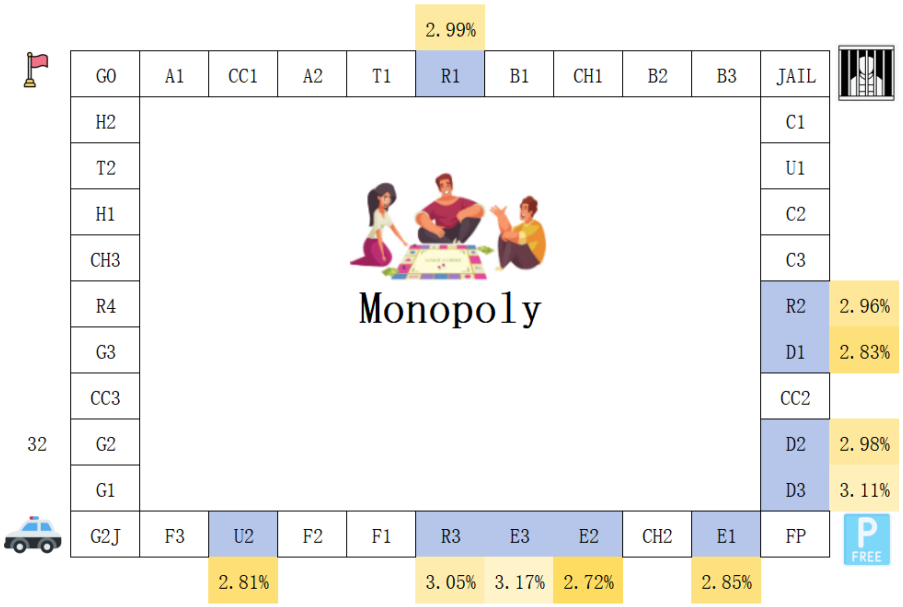


图 11: 优先购买的 10 个地产

2. 由于格子 30 “送入监狱” 到格子 10 “监狱” 的转移机制和特殊卡牌提供机会频繁地转移到监狱，格子 10 和格子 30 之间的极限概率相对于平均水平 2.5% 基本上都显著增加了，因为概率和为 1 的约束，相应减少了这个区域之外的访问概率。

那么问题二最后的策略也呼之欲出，显然我们应该购买最高频率被访问的那部分格子的地产（如果我们有的选的话），除去特殊格子 10 个，总共地图上有 30 个格子的地产可供购买，假设 3 位玩家之间购买的几率相等，一个玩家平均可以谋划购买 10 个格子的地产，我们将最高概率访问的格子展示如图（11），并可以清晰地看到在上述提到格子 10 和格子 30 之间的 “黄金地段” 的确是一个赚钱的风水宝地。

3.2.3 问题 2.2 三个玩家时间上的联合概率分布——封闭排队网络

在问题 2.2 中，我们不仅将离散的轮数角度转变为了连续的时间角度，同时还要进一步讨论三个玩家共同处于地图上的情况，从本质上来看，其实这就是一个顾客数 $N = 3$、服务器数 $K = 40$ 的封闭排队网络。

我们来详细介绍这一网络里的设定，这里和一般的排队网络最大的不同就是 “封闭” 这个概念，这意味着系统内有限的人数，既不能因为到来增加也不能因为离开减少。

其余的设定是在排队网络中类似的概念，在大富翁中我们设定 “顾客” 就是 “玩家”，“服务台” 就是 “格子”。长远来看玩家到达格子 j 的速率记为 r_j，由于没有来自外部玩家的到来，根据 r_j 的平衡性质，有

$$r_j = \sum_{i=1}^{K} r_i p(i,j) \tag{9}$$

其中 $p(i,j)$ 是在格子 i 完成 “服务” 后转到格子 j 的概率，其实就是我们在问题 1 中构造的转移概率。方程（9）矩阵形式下的

$$r = r \cdot P$$

揭示了 r 其实就是问题 2.1 中平稳分布的倍数 $R\cdot\pi$，其中 $R=\sum_j r_j$ 表示系统中服务完成的平均速率，也称作“吞吐率”。

相应地，这里“服务速率 u_i”的概念其实就是对玩家在格子 i 上平均停留时间 $1/u_i$ 的刻画。毕竟在更真实的大富翁中，玩家停留在不同格子上的时间，理应是不同概率分布的随机变量，例如在碰到地产时可能需要一定的决策时间，在进入监狱时我们可以将等待两轮看作更长的停留时间，在碰到特殊转移位置的效果卡片时，可能立刻就转移出去 $\cdots\cdots$ 这也是我们为什么要引入这个模型的原因，嵌入这么一个排队网络能从时间上更合理地探讨大富翁上玩家的位置情况。

于是我们同样对“服务速率”在这里的概念“离开格子的速率” u_i 作一些设置：

编号 i	格子	解释	近似停留时间 $1/u_i$
other	一般地产格子	考虑玩家购买、卖出、交易等决策	8 s
2,7,17,22,33,36	特殊卡 CC、CH 格子	抽卡时间较短	4 s
10	监狱格子（JAIL）	需要停留两轮	10 s
30	到达监狱格子（G2J）	立即转移到监狱	1 s
0,20	无效果格子（GO 和 FP）	什么都不需要做	3 s

表 2: 停留时间近似

引入这一封闭转移模型下的几条假定和规则：

(a) 游戏每轮三位玩家轮流投骰子前进，我们近似将其看作 3 个同时自由行动的顾客。

(b) 由于这个封闭网络中到达格子后实际上并不用“排队”, 所以每个格子的排队系统为 M/M/∞ 是合理的。

(c) 当站点 i 有 n_i 个顾客时，离开速率为 $\phi(n_i)$，其中 $\phi(0)=0$。在 M/M/∞ 下 $\phi(n_i)=n_i\cdot u_i$。

(d) 个体离开 i 后以概率 $p(i,j)$ 去往 j。

由于封闭性，此时各队列的长度之间理应存在相互依赖关系，也就是说当总顾客数固定时，各队列长度不可能是独立的。然而，我们在这里省略在书上已经完成的可行性证明，仍然在假设独立下构造具有乘积形式的平稳分布：

$$P_N(n_i\text{个玩家在格子}i)=P(n_0,...,n_{K-1})=C_N\prod_{i=0}^{K-1}\frac{\pi_i^{n_i}}{\psi_i(n_i)} \tag{10}$$

其中 $n_1+...+n_K=N=3,K=40,C_N$ 是归一化常数，这里 $\psi(n_i)=\prod_{m=1}^{n_i}\phi(m)$ ，$(\psi(0)=1)$ 是一种为了凑出平衡条件的特殊构造，在 M/M/∞ 下简化为：

$$P_N(n_0,...,n_{K-1})=C_N\prod_{i=0}^{K-1}\frac{\pi_i^{n_i}}{n_i!\cdot u_i^{n_i}}$$

为了计算归一化常数 C_N，我们需要对 $(n_0,...,n_{K-1})$ 排列组合的所有可能下的乘积 $\prod_{j=0}^{K-1}$ 进行求和，对于这个本质上的“可重排列”问题，我们已知 $N=3$ 具有的整数分拆为 $3=3,3=2+1,3=1+1+1$，对应排列数的和 $C_{40}^1+A_{40}^{39}+C_{40}^3=40+1560+9800=11480$ 个可能，我们可以计算出 $C_N=69.0786$，平稳分布（10）的其余参数均在上面给出，我们同样可以排序得到一些有趣的性质：

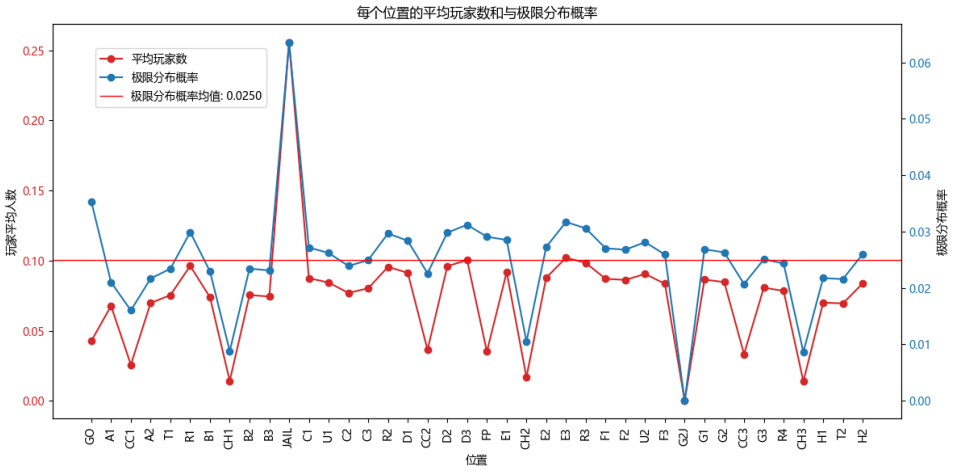


图 12: 极限分布概率和平均玩家数

- 所有组合中，最高概率的前三个状态（0.742%、0.730%、0.715%）都有两个玩家停留在格子 10 监狱，这看起来并不奇怪，毕竟在监狱呆的时间略久，又有很高的概率转移过去。
- 所有组合中，格子 10(Jail)，格子 24(E3)，格子 19（D3）依次降序拥有汇聚三个玩家的最大概率，至于为什么原先在离散的轮数极限分布中的格子 0（GO）没有上榜也不难理解，因为玩家几乎不在这个格子上花费什么时间来决策。
- 所有格子中哪些格子上最可能存在玩家呢? 我们计算每个格子的玩家人数均值，并将其与在问题 2.1 中的极限分布同时展示（图12），可以看到这里分布的情况和原来大致类似，除了少部分格子由于我们设置平均停留时间略短而有所改变。如果想要更准确的估计这一分布，也许我们应该从一些游戏数据中提取对每个格子平均停留时间 $1/u_i$ 的估计，然后再使用这个模型。

对于更庞大一点的模型，由于状态空间可能情况的排列数快速增长造成的复杂影响，我们一般只在 N 和 K 相对小时尝试如上的计算。所以我们采用另一些更巧妙的方法避开 C_N 的计算来得到这个分布并可以得到一些其他有趣的性质。对于封闭网络中一般的单一队列模型，Ross《应用随机过程》[3] 已给出了基于到达定理的 “平均值分析” 方法，来递推计算在格子 j 上停留的平均玩家数；对于我们这里考虑的 M/M/∞ 队列，我们可以选用**吉布斯抽样（Gibbs Sampling）**来生成具有这些平稳概率的多元马尔科夫链。

作为开始，我们在这里简单介绍一下吉布斯抽样方法。回顾 Richard Durrett《应用随机过程》书中在第一章马氏链中介绍过了 Metropolis-Hastings 算法，它是马尔可夫链蒙特卡罗方法 MCMC 中相当重要的一环，通过一定的概率接受/拒绝来更新状态的过程，一直持续到采样收敛为目标分布。而吉布斯采样就是这个算法的一个特例，适用于联合分布未明确知道或难以直接抽样，但每个变量的条件分布是已知的并且很容易（或者至少更容易）从中抽样的情况。

注意因为总是有 N 个顾客在系统中, 方程 (10) 可以等价地化简为在 $k-1$ 个格子 $0,\cdots,K-$

2 中顾客数的联合质量函数如下:

$$\begin{aligned} P_N\left(n_0,\cdots,n_{K-2}\right) &= C_N\left(\pi_K/\mu_K\right)^{N-\sum_{j=0}^{K-2} n_j} \prod_{j=0}^{K-2} \frac{\pi_j^{n_j}}{n_j!\cdot u_j^{n_j}} \\ &= M \prod_{j=0}^{K-2} \frac{1}{n_j!}\left(a_j\right)^{n_j}, \quad \sum_{j=0}^{K-2} n_j \leqslant N \end{aligned}$$

其中 $a_j=\frac{\pi_j\mu_K}{\pi_K\mu_j}, j=0,\cdots,K-2$。我们做的其实就是把关于格子 $K-1$ 作为已知的条件（由于其他 $K-1$ 格子已知下第 K 个格子自然就已知了），把方程 (10) 中有关的 $\left(\pi_K/\mu_K\right)^{N-\sum_{j=0}^{K-2} n_j}$ 拆成指数上的两部分，前一部分汇总到 C_N 里成为 M，后一部分分配到每个累乘号里凑成 a_j。然后为了方便起见, 将这 $K-1$ 个格子的分布情况写作有上述的联合分布的多元随机变量 $N=\left(N_0,\cdots,N_{K-2}\right)$, 那么我们就可以得到 N_i 的条件概率:

$$\begin{aligned} &\mathrm{P}\left\{N_i=n \mid N_0=n_0,\cdots,N_{i-1}=n_{i-1},N_{i+1}=n_{i+1},\cdots,N_{K-2}=n_{K-2}\right\} \\ =&\frac{P_N\left(n_0,\cdots,n_{i-1},n,n_{i+1},\cdots,n_{K-2}\right)}{\sum_r P_N\left(n_0,\cdots,n_{i-1},r,n_{i+1},\cdots,n_{K-2}\right)} \end{aligned}$$

分子为:

$$P_N\left(n_0,\cdots,n_{i-1},n,n_{i+1},\cdots,n_{K-2}\right)=M\prod_{j\neq i}\frac{1}{n_j!}\left(a_j\right)^{n_j}\cdot\frac{1}{n!}\left(a_i\right)^n$$

分母为:

$$\begin{aligned} \sum_r P_N\left(n_0,\cdots,n_{i-1},r,n_{i+1},\cdots,n_{K-2}\right) &= \sum_r M\prod_{j\neq i}\frac{1}{n_j!}\left(a_j\right)^{n_j}\cdot\frac{1}{r!}\left(a_i\right)^r \\ &= M\prod_{j\neq i}\frac{1}{n_j!}\left(a_j\right)^{n_j}\sum_r\frac{1}{r!}\left(a_i\right)^r) \end{aligned}$$

注意到 $1/\left[\sum_r \frac{1}{r!}\left(a_i\right)^r\right]$ 这部分可直接计算，不妨用常数 C 来代替，将分子和分母代入就得到最终的条件概率:

$$\begin{aligned} &\mathrm{P}\left\{N_i=n \mid N_0=n_0,\cdots,N_{i-1}=n_{i-1},N_{i+1}=n_{i+1},\cdots,N_{K-2}=n_{K-2}\right\} \\ =&\frac{M\prod_{j\neq i}\frac{1}{n_j!}\left(a_j\right)^{n_j}\cdot\frac{1}{n!}\left(a_i\right)^n}{M\prod_{j\neq i}\frac{1}{n_j!}\left(a_j\right)^{n_j}\cdot\sum_r\frac{1}{r!}\left(a_i\right)^r} \\ =&C\frac{1}{n!}\left(a_i\right)^n, \quad n\leqslant N-\sum_{j\neq i}n_j \end{aligned}$$

上式即可在以下吉布斯抽样法的具体步骤中使用,来生成具有极限概率质量函数 $P_N\left(n_0,\cdots,n_{k-2}\right)$ 的马尔可夫链了，步骤如下:

(1) 初始化变量空间，取 $\left(n_0,\cdots,n_{K-3}\right)$ 是满足 $\sum_{j=0}^{K-2} n_j \leqslant N$ 任意非负整数分拆的有序排列。

(2) 一个 epoch 开始，生成一个在 $0,\cdots,K-2$ 中等可能地取值的随机变量 I。

(3) 假设 $I=i$，生成此时在其他格子状态已知为 $n_{j\neq i}$ 条件下 n_i 的取值上限 $s=N-\sum_{j\neq i}n_j$，然后通过 s 计算出此时的 C，并生成一个具有概率质量函数为

$$\mathrm{P}\{X=n\}=C\frac{1}{n!}\left(a_i\right)^n,\quad n=0,\cdots,s$$

的随机变量 X ，并通过这个概率密度随机取一个值 x。

(4) 将这个值 x 赋给 n_i，然后回到第 2 步迭代。

我们忽略对“应该在多少 epoch 停止迭代的判定策略”的介绍，仅从理论上关注结果得到的状态向量 $\left(n_0,\cdots,n_{K-2},m-\sum_{j=1}^{K-2}n_j\right)$ 的相继值构成一个具有极限分布 P_N 的马尔可夫链的状态序列，这就是平稳分布（10）的近似估计了，我们可以利用它来观测任何有趣的量。例如第 j 个格子的平均玩家数, 第 j 个格子玩家数少于 r 的极限概率 $\cdots\cdots$ 总之，在游戏简化的规则下，也只能大致地模拟到这里了。

3.3 问题三：简化的财富流动模型——“三人赌博”

到了这里我们已经对大富翁这个游戏中，玩家的位置这个概念做了相当多的估计了，但是游戏中最精华的资金流动部分我们始终没有涉及，这主要是因为在这个庞大的位置转移过程中，复杂的地产分布很难给我们机会推导出一个漂亮简洁的式子来模拟玩家资金的情况。

但是我们还是说，就在这里停止显得略有遗憾，所以我们不妨最后考虑再稍微简化一点这个“三人赌博”模型，用在现代金融领域广泛运用的“鞅论”，来做一个简单的刻画，在这里我们主要关心的几个问题就是，一局游戏中当有一个玩家破产的平均时间是多少？当整局游戏结束，即有两人破产的时间又为多少?

3.3.1 转移规则介绍

对给定地图上地产分布，玩家走一圈后的收益显然服从某种分布的随机变量，我们把这种随机获得收益的过程简化成每轮中资金在三位玩家 X,Y,Z 之间的转移过程，那么就可以列出下表（3），我们说明

- 在这里特别地近似“一轮中由于踩到非地产格子（例如起点、监狱）而不发生金额转移”对应下表中最后一行的事件
- 这里的概率满足 $p_1+p_2=p_3+p_4=p_5+p_6=\frac{1-p_7}{6}$，这主要保证了“其中两个玩家交互”的事件之间发生概率均等

然后我们沿用游戏中初始金额均衡的设置，X,Y,Z 三位玩家的初始资金均为 $X_0=Y_0=Z_0=K_0$，第 n 轮结束的资金分别为 X_n,Y_n,Z_n，由于一阶统计量 $X_n+Y_n+Z_n=3K_0$ 已经固定，于是考虑二阶统计量 $S_n=X_nY_n+Y_nZ_n+X_nZ_n$，构造

$$M_n=\sum_{k=1}^{n}(S_k-\mathbb{E}[S_k|(X_0,Y_0,Z_0),...,(X_{k-1},Y_{k-1},Z_{k-1})])$$

则可以证明 M_n 在一定条件下是鞅。由于该游戏中有玩家破产时会导致规则发生一定的改变，所以我们在分情况讨论前规定基本的符号：

概率	付钱方	收钱方	转移金额
p_1	X	Y	C_1
p_2	Y	X	C_2
p_3	Y	Z	C_3
p_4	Z	Y	C_4
p_5	X	Z	C_5
p_6	Z	X	C_6
p_7	无	无	0

表 3: 转移金额概率表

- $\tau = \min\{n : X_n \leqslant 0 \ or \ Y_n \leqslant 0 \ or \ Z_n \leqslant 0\}$，即第一个玩家破产的时间
- T，游戏结束，即出现两个玩家破产的时间

3.3.2 验证 M_n 是鞅

验证 M_n 为鞅只需要验证 S_n 的性质，我们先讨论 $k \leqslant \tau$ 时，简记 $X_{k-1} = x, Y_{k-1} = y, Z_{k-1} = z$，则：

$$\begin{aligned}\mathbb{E}[S_k|(X_0,Y_0,Z_0),...,(X_{k-1},Y_{k-1},Z_{k-1})] = & \ xy + yz + xz \\ & + x(C_1p_1 - C_2p_2 + C_5p_5 - C_6p_6) \\ & + y(-C_1p_1 + C_2p_2 + C_3p_3 - C_4p_4) \\ & + z(-C_3p_3 + C_4p_4 - C_5p_5 - C_6p_6) \\ & - (\sum_{i=1}^{6} p_i \cdot C_i^2)\end{aligned} \tag{11}$$

这一计算可以通过列表来相对直观地展示：

概率	转移过程	转移金额	X_kY_k	Y_kZ_k	X_kZ_k
p_1	X → Y	C_1	$(x - C_1)(y + C_1)$	$(y + C_1)z$	$(x - C_1)z$
p_2	Y → X	C_2	$(x + C_2)(y - C_2)$	$(y - C_2)z$	$(x + C_2)z$
p_3	Y → Z	C_3	$x(y - C_3)$	$(y - C_3)(z + C_3)$	$x(z + C_3)$
p_4	Z → Y	C_4	$x(y + C_4)$	$(y + C_4)(z - C_4)$	$x(z - C_4)$
p_5	X → Z	C_5	$(x - C_5)y$	$y(z + C_5)$	$(x - C_5)(z + C_5)$
p_6	Z → X	C_6	$(x + C_6)y$	$y(z - C_6)$	$(x + C_6)(z - C_6)$
p_7	无	0	xy	yz	xz

表 4: 无人破产时的转移情况

特别说明当玩家出现需要支付的金额大于当前资产时，我们添加上一条规则，即该玩家 L 将会把所有地产抵押给银行，然后刚好获得一笔足以支付该环节金额的钱，并在支付后退出游戏。此后，剩下的两个玩家 R_1, R_2 将以下表（5）的规则继续玩下去，其中转移金额仍然不变，概率为原有三个概率的归一化来表示。总而言之，这主要是为了保证第一个停时出现后规则的衔接。

概率	转移过程	转移金额	$R_{1k}R_{2k}$
p_1^*	$R_1 \to R_2$	C_1^*	$(r_1 - C_1^*)(r_2 + C_1^*)$
p_2^*	$R_2 \to R_1$	C_2^*	$(r_1 + C_2^*)(r_2 - C_2^*)$
p_3^*	无	0	$r_1 r_2$

表 5: 有人破产后剩余玩家的转移情况

在出现玩家破产前，我们已知道 $p_i \cdot C_i$ 是一步转移给某一方的平均金额，如果记 $f_0 = \sum_{i=1}^{6} p_i \cdot C_i^2$，记 XY, YZ, XZ 之间平均转移金额差分别为 D_1, D_2, D_3，例如 $D_1 = C_{xy} \cdot p_{xy} - C_{yx} \cdot p_{yx} = C_1 p_1 - C_2 p_2$，那么上面的期望（公式 (11)）就可以简记为

$$\mathbb{E}[S_k|...] = S_{K-1} + x(D_1 + D_3) + y(-D_1 + D_2) + z(-D_2 - D_3) - f_0$$

对于后面需要构造的 M_n，我们可以证明当且仅当 $D_1 = D_2 = D_3 = 0$ 时 x, y, z 的系数均为 0，这个条件就是任意两人之间平均转移金额差相同，意味着二者某种程度上公平，只是各自获得收益的方差可能不一样。于是在这个 “相对公平” 条件下，就可以验证 M_n 为鞅（当 $n \leqslant \tau$），此时

$$\begin{aligned}
M_n &= S_n - S_0 + nf_0 \\
\mathbb{E}[M_n|(X_0, Y_0, Z_0), ..., (X_{k-1}, Y_{k-1}, Z_{k-1})] &= E[S_n - S_0 + nf_0|(X_0, Y_0, Z_0), ..., (X_{k-1}, Y_{k-1}, Z_{k-1})] \\
&= E[S_n|(X_0, Y_0, Z_0), ..., (X_{k-1}, Y_{k-1}, Z_{k-1}) - S_0 + nf_0 \\
&= S_{n-1} - S_0 + (n-1)f_0 \\
&= M_{n-1}
\end{aligned}$$

3.3.3 可选停时定理充分条件证明与停时的上下界估计

我们希望根据**可选停时定理** $M_\tau = M_0$ 来获得一些特定的量，所以先验证该定理两个充分条件：

- $P(\tau < \infty) = 1$
  我们稍微往后一点，先估计 $P(T < \infty)$ 的概率。我们可以关注三个玩家各自资金的增减情况，例如对 X_n 来说，其实就是一个给定初始位置 X_0，每一轮都有概率向前后跳若干步的一维随机游走，吸收态是 “$a \leqslant 0, b \geqslant 3K_0$”。“$a \leqslant 0$” 不难理解就是对应着破产，“$b \geqslant 3K_0$” 其实就近似为获得游戏胜利，毕竟在此时该玩家已经汇聚了所有资金总 $3K_0$。为什么说这里是近似呢？因为我们考虑到，按照我们前面的规则，若玩家破产时 $X_\tau \leqslant 0$ 会在此时向银行抵押地产来补上表（3）中不足的支付部分，从而导致场上总资金稍微变多。但总之，判定游戏结束条件 $b \geqslant K$ 中 K 一定是一个有限的数。

所以，尽管这里稍微比一般的随机游走复杂一点，但是这个近似在直觉上是有效的，而由随机游走的结论可以知道这里的 $P(T<\infty)=1$ 是一定成立的，那么对于 $\tau<T$ 也一定有 $P(\tau<\infty)=1$

- $|M_{\tau\wedge n}|<M^*<\infty$

  这里很好验证，因为 $M_{\tau\wedge n}$ 中非常数项只有 $S_{\tau\wedge n}$，而 $X_{\tau\wedge n},Y_{\tau\wedge n},Z_{\tau\wedge n}$ 三项都有上界（因为资金总和就那么多），所以 $|M_{\tau\wedge n}|$ 有上界是自然成立的。

于是就有

$$\begin{cases}\mathbb{E}[M_\tau]=\mathbb{E}[M_0]=0\\ \mathbb{E}[M_\tau]=\mathbb{E}[S_\tau]-\mathbb{E}[S_0]+\mathbb{E}[\tau]\cdot f_0\end{cases}\text{解得 }\mathbb{E}[\tau]=\frac{\mathbb{E}[S_0]-\mathbb{E}[S_\tau]}{f_0}$$

我们说这里未知的只有 $\mathbb{E}[S_\tau]$，因为它实在有点复杂很难通过一般方法得到这个均值，所以我们不妨对它进行一个放缩，从而得到关于 $\mathbb{E}[\tau]$ 的一个范围估计。假设破产的玩家为 L，另外两位留存的玩家为 R_1,R_2，注意到

$$\begin{cases}0\geqslant L_\tau\geqslant -m_l=-\max\{C_i,i=1,\cdots,6\}\\ m_r\leqslant R_{i\tau}<K=3K_0+\max\{C_i,i=1,\cdots,6\}\text{，由于}R_i\text{不均为 0，}m_r\text{是}R_i\text{所能达到的最小非 0 值}\\ m_r(3K_0-m_r)<R_{1\tau}\cdot R_{2\tau}=(K-R_{2\tau})R_{1\tau}<\dfrac{1}{4}K^2\text{，在约束条件}0<R_{1\tau}+R_{2\tau}<K\text{下}\end{cases}$$

故而

$$m_r(3K_0-m_r)-2m_rm_l<\mathbb{E}[S_\tau]=\sum_{i=1}^{2}L_\tau R_{1\tau}+R_{1\tau}R_{2\tau}<\frac{1}{4}K^2$$

$$\frac{S_0-\frac{1}{4}K^2}{f_0}<\mathbb{E}[\tau]<\frac{S_0-[m_r(3K_0-m_r)-m_rm_l]}{f_0}$$

这就是我们得到的上下界估计，比如取下表（7）的转移规则：其中 $f_0=46428.57$，初始资金

概率	付钱方	收钱方	转移金额
$\frac{8}{70}$	X	Y	300
$\frac{12}{70}$	Y	X	200
$\frac{4}{70}$	Y	Z	400
$\frac{16}{70}$	Z	Y	100
$\frac{10}{70}$	X	Z	250
$\frac{10}{70}$	Z	X	250
$\frac{10}{70}$	无	无	0

表 6: 例子

$K_0=1600,S_0=7680000$，我们就有了第一个玩家结束的平均时间范围 $41.353<\mathbb{E}[\tau]<120.76$，python 模拟 10000 次的平均时间结果是 60.79，这验证了我们的结果。

对于整局游戏结束的时间 T，我们同样可以使用**可选停时定理**，其充分条件我们同理可证，直接给出 $n>\tau$ 时 M_n 的修正形式：

$$M_n{}^*=S_n^*-S_0+[\tau f_0+(n-\tau)f_0^*],n>\tau$$

在这里 $f_0^* = \sum_i^2 p_i^* \cdot C_i^{*2}$ 是和原来一样末尾的常数，$S_n^* = R_{1n}R_{2n}, n > \tau$ 表示的是在 τ 后将出局者资金 L_n 修正为 0 的二阶量，相应的有 $S_\tau^* \geqslant S_\tau$。由于此后两人转移金额的鞅仍然是和三人时类似的形式，用这个方法的前提下我们直接给出估计的结果：

$$\begin{cases} \mathbb{E}[{M_T}^*] = \mathbb{E}[{M_\tau}^*] = \mathbb{E}[{S_\tau}^*] \\ \mathbb{E}[{M_T}^*] = \mathbb{E}[S_T^*] - S_0 + \mathbb{E}[\tau](f_0 - f_0^*) + \mathbb{E}[T]f_0^* \end{cases}$$

$$\begin{aligned} \mathbb{E}[T] &= \frac{\mathbb{E}[{S_\tau}^*] + S_0 - \mathbb{E}[S_T^*] - \mathbb{E}[\tau](f_0 - f_0^*)}{{f_0}^*} \\ &= \frac{\mathbb{E}[{S_\tau}^*] + S_0 - \mathbb{E}[S_T^*]}{{f_0}^*} + \mathbb{E}[\tau](1 - \frac{f_0}{{f_0}^*}) \\ &\geqslant \frac{m_r(3K_0 - m_r) + S_0 + 0}{{f_0}^*} + \mathbb{E}[\tau](1 - \frac{f_0}{{f_0}^*}) \text{ 记留下玩家组合中最大/小的}{f_0}^*\text{为}{{f_0}^*}_{max}/{{f_0}^*}_{min} \\ &\geqslant \frac{m_r(3K_0 - m_r) + S_0}{{{f_0}^*}_{max}} + \frac{S_0 - \frac{1}{4}K^2}{f_0} \cdot (1 - \frac{f_0}{{{f_0}^*}_{min}}) \end{aligned}$$

取刚刚的例子表（7）的转移规则，此时所有留下的玩家组合中最大的 ${f_0}^* = 41666.6$，可计算出下界 $\mathbb{E}[T] > 190.87$，实际模拟结果均值为 225.03。

我们整理一下关于这个“分阶段”鞅 M_n 的形式：

$$M_n = \begin{cases} S_n - S_0 + nf_0 & \text{if } n \leqslant \tau \\ S_n^* - S_0 + [\tau f_0 + (n - \tau)f_0^*] & \text{if } n \geqslant \tau \end{cases}$$

在上面的分析中已知到，由于发生一位玩家破产的事件会带来规则的改变，所以 M_n 在 ${S_0}^*$ 修正、f_0 和 ${f_0}^*$ 差异的影响下，会导致前后不连贯，我们总是将它们分成两个过程进行讨论。于是我们在最后介绍一个在“相对公平下”更特殊的条件，来统一两个过程——“绝对公平”。

3.3.4 更特殊的例子：绝对公平条件与推广

“绝对公平”指的是，所有玩家等概率地转移相同的金额给另外一个玩家，即 $p_i = \frac{1-p_7}{6} = p_0, i = 1, \cdots, 6$, 我们仍然取初始资金 $K_0 = 1600$, 转移金额 $C_0 = 200$, 无转移事件概率 $p_7 = 1/7$, 在表中即有

概率	付钱方	收钱方	转移金额
$p_0 = 1/7$	X	Y	$C_0 = 200$
$p_0 = 1/7$	Y	X	$C_0 = 200$
$p_0 = 1/7$	Y	Z	$C_0 = 200$
$p_0 = 1/7$	Z	Y	$C_0 = 200$
$p_0 = 1/7$	X	Z	$C_0 = 200$
$p_0 = 1/7$	Z	X	$C_0 = 200$
$p_7 = 1/7$	无	无	0

表 7: 绝对公平下的转移

在此时，根据定义可以知道

$$\begin{cases} f_0 = (1-p_7)C_0 = {f_0}^* \\ S_\tau = R_{1n}R_{2n} = {S_\tau}^* \end{cases}$$

第二个条件是因为在此时 K_0 整除 C_0，故而当有人破产时，他的资金一定为 0 而不会减到负数，而我们之前修正，即“把出局玩家的资产归零”的步骤自然而然地省略了。那么此时 $M_n = S_n + S_0 + nf_0$ 的形式将完整贯穿整局游戏，我们验证在破产时刻的连贯性：

$$\begin{aligned} \mathbb{E}[M_{\tau+1}|(X_{\tau-1}, Y_{\tau-1}, Z_{\tau-1}), \cdots] &= E[S_{\tau+1}|(X_{\tau-1}, Y_{\tau-1}, Z_{\tau-1}), \cdots] - S_0 + \tau f_0 \\ &= \mathbb{E}[R_{1\tau+1}R_{2\tau+1}] - S_0 + \tau f_0 \\ &= R_{1\tau}R_{2\tau} - S_0 + (\tau-1)f_0 \\ &= S_\tau + S_0 + (\tau-1)f_0 \\ &= M_\tau \end{aligned}$$

于是根据可选停时定理，有

$$\begin{cases} \mathbb{E}[M_T] = \mathbb{E}[M_0] = 0 \\ \mathbb{E}[M_T] = \mathbb{E}[S_T] - S_0 + \mathbb{E}[T] \cdot f_0 \end{cases}$$

注意到 $\mathbb{E}[S_T]$ 时由于存在两个破产玩家的资产为 0，$S_T = X_TY_T + Y_TZ_T + X_TZ_T = 0$，解得

$$\mathbb{E}[T] = \frac{S_0}{f_0} = 224$$

也就是说，这局游戏结束的时间完全由初始资金和转移资金决定，考虑到“任意两个玩家发生财富转移”事件概率均等，所以游戏结束的轮数应该为 $224/3 = 74.6$ 轮。进一步，由我们以上的分析可以很容易将其推广到 m 个人 $(X^{(m)})$ 的情况，因为在 $\mathbb{E}[S_k]$ 展开式中二阶项就是 S_{k-1}，剩下就是系数均为 0 的一阶项和一个常数。即相同的规则下：

$$\mathbb{E}[T] = \frac{\sum_{i\neq j} X_0^{(i)} X_0^{(j)}}{(n-1)p_0C_0^2}$$

至此，我们完成了对大富翁游戏简化成的“三人赌博”财富转移模型的估计。

4 结语与致谢

我们回顾文中和本学期学习的大多结论，都能体悟到各种定理结论的神奇和其中推导构造的绝妙。我们往往可以发现，根据“随机过程”中一定的规则、确定的初始条件又或者是某个局部的特殊性质，就能直接得到关于整个随机过程中种种事件的概率、等待的平均时间 …… 等等有用的性质，本质上其实都揭示了，尽管对于一个“随机过程”其有着种种**不确定**的特性，但在长期、平均意义下却又可以得知的**确定性**的结果。

正如引言所说，为了涵盖了不少本学期学到的理论，学生在有限能力下对理论的应用总觉有“先射箭后画靶”的不自然，大概是“一看就是初学者写的”的水平。本文中对该游戏的多种近似也许并不精确和严谨，只希望能够提高知识的掌握程度，并锻炼一下实践能力，在最后还是希望老师海涵和指正！

虽然本文作为论文而言，格式确实有点不太严谨，但最后还是感谢本课程老师方明对文中多个部分的解答指导以及一学期的辛苦教学！另特别鸣谢 16 级学长刘理在知乎开设的随机过程专栏，在学习过程中帮助良多。

5 参考文献

[1] Richard Durrett. 应用随机过程. Trans. by 张景肖，李贞贞. 机械工业出版社, 2014. ISBN: 9787111447511.

[2] Project Euler. *https://pe-cn.github.io/84/*.

[3] S.M. Ross. 应用随机过程：概率模型导论：第 *9* 版. Trans. by 龚光鲁. 北京：人民邮电出版社, 2007. ISBN: 9787115167330.

6 附录

```python
#计算转移概率矩阵
import numpy as np
dice_probs = np.array([0, 0, 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1]) / 36

D = np.zeros((40, 40))
for i in range(40):
    for dice_sum in range(2, 13):
        D[i, (i + dice_sum) % 40] += dice_probs[dice_sum]

C = np.eye(40)
def find_next(current, targets):
    for i in range(1, 40):
        if (current + i) % 40 in targets:
            return (current + i) % 40
    return current

railroads = [5, 15, 25, 35]
utilities = [12, 28]
CC_squares = [2, 17, 33]
CC_transitions = [(0, 1/8), (10, 1/8)]
sum_prob = 0
for square in CC_squares:
    for target, prob in CC_transitions:
        C[square, target] += prob
        sum_prob += prob
    C[square, square] = 1-2/8
sum_prob = 0
CH_squares = [7, 22, 36]
CH_transitions = [(0, 1/16), (10, 1/16), (11, 1/16), (24, 1/16), (39, 1/16), (5,
    1/16), (-3, 1/16)]
for square in CH_squares:
    for target, prob in CH_transitions:
        if target == -3:
            C[square, (square - 3) % 40] += prob
            sum_prob += prob
        else:
            C[square, target] += prob
            sum_prob += prob

    next_railroad = find_next(square, railroads)
    C[square, next_railroad] += 1/8

    next_utility = find_next(square, utilities)
```

```
    C[square, next_utility] += 1/16
    C[square, square] = 1-10/16

C[30] = 0
C[30, 10] = 1
C[30, 30] = 0
P = np.dot(D, C)
```

```
#问题1.1，解法一：递推计算
dice_prob = np.array([0, 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1, 0, 0]) / 36
pi = np.zeros(12)
pi[11] = 1

A = np.zeros((12, 12))
for i in range(11):
    A[i, i+1] = 1
A[11, :] = dice_prob[1:13]

def next_pi(current_pi, A):
    return A @ current_pi

max_iterations = 10000
tolerance = 1e-10
pi_distributions = []

for iteration in range(max_iterations):
    pi_new = next_pi(pi, A)
    pi_distributions.append(pi_new)
    if np.linalg.norm(pi_new - pi, ord=1) < tolerance:
        convergence_iteration = iteration
        break
    pi = pi_new

convergence_pi = pi
```

```
#计算问题1.2离出分布
P0 = np.delete(P, [0, 10], axis=0)
R1 = P0[:, 0]
P1 = np.delete(P0, [0, 10], axis=1)

I = np.eye(P1.shape[0])
pro = np.dot(np.linalg.inv(I - P1), R1)

ones = np.ones(P1.shape[0])
```

```
T = np.dot(np.linalg.inv(I - P1), ones)
```

```
#计算问题2.1极限分布
import pandas as pd
def get_steady_vector(P, tol=1e-10):
    vec_i = np.array([1] + [0] * (P.shape[0] - 1))
    iterations = [vec_i]
    while True:
        vec_s = vec_i @ P
        iterations.append(vec_s)
        if np.linalg.norm(vec_s - vec_i) < tol:
            return vec_s, iterations
        vec_i = vec_s

steady_vector, iterations = get_steady_vector(P)
steady_df = pd.DataFrame({
    "编号": range(40),
    "名称": labels,
    "极限分布概率": steady_vector,
    "平均访回时间": [1 / p if p > 0 else 0 for p in steady_vector]
})
```

```
#计算问题2.2封闭排队网络平稳分布
u = np.ones(40) * 1/8
u[[2, 7, 17, 22, 33, 36]] = 1/4
u[10] = 1/10
u[30] = 1
u[[0, 20]] = 1/3

r = steady_vector
R = np.sum(r)
pi = r
K, N = 40, 3

def generate_combinations(K, N):
    results = []
    partitions = [[3], [2, 1], [1, 1, 1]]
    for partition in partitions:
        if partition == [3]:
            results.extend([(3 if i == j else 0) for j in range(K)] for i in range(K))
        elif partition == [2, 1]:
            results.extend(
                [(2 if j == i else 1 if j == k else 0) for j in range(K)]
                for i, k in itertools.permutations(range(K), 2)
```

```
            )
        elif partition == [1, 1, 1]:
            results.extend(
                [(1 if j in indices else 0) for j in range(K)]
                for indices in itertools.combinations(range(K), 3)
            )
    return [tuple(result) for result in results]

combos = generate_combinations(K, N)
C_N = sum(
    np.prod((pi[i] ** n_i) / (np.math.factorial(n_i) * (u[i] ** n_i)) for i, n_i in
        enumerate(combo))
    for combo in combos
)

def P_N(combo, C_N, pi, u):
    return np.prod((pi[i] ** n_i) / (np.math.factorial(n_i) * (u[i] ** n_i)) for i,
        n_i in enumerate(combo)) / C_N

probabilities = [(combo, P_N(combo, C_N, pi, u)) for combo in combos]
```

```
#问题3 鞅论
#条件1：相对公平
probabilities = [8/70, 12/70, 4/70, 16/70, 10/70, 10/70, 10/70]
p0 = [[0,8/70,10/70],
      [12/70,0,4/70],
      [10/70,16/70,0],]
a0 = [[0,300,250],
      [200,0,400],
      [250,100,0]]
transfers = [
    (0, 1, 300),
    (1, 0, 200),
    (1, 2, 400),
    (2, 1, 100),
    (0, 2, 250),
    (2, 0, 250),
    (-1, -1, 0) # No transfer
]

f0 = 0
for i in range(len(probabilities)):
    f0 += probabilities[i]*transfers[i][2]**2

```

```
24
25 #条件2：绝对公平
26 probabilities = [1/7,1/7,1/7,1/7,1/7,1/7,1/7]
27 transfers = [
28     (0, 1, 200),
29     (1, 0, 200),
30     (1, 2, 200),
31     (2, 1, 200),
32     (0, 2, 200),
33     (2, 0, 200),
34     (-1, -1, 0) # No transfer
35 ]
36 f0 = 6/7*200*200
37
38 def simulate_game():
39     money = [initial_money, initial_money, initial_money]
40     rounds = 0
41     rounds1 = 0
42     sump = 0
43     while all(m > 0 for m in money):
44         rounds += 1
45
46         transfer_idx = np.random.choice(len(probabilities), p=probabilities)
47         payer, receiver, amount = transfers[transfer_idx]
48         if payer != -1 and money[payer] > 0:
49
50             money[payer] -= amount
51             money[receiver] += amount#min(money[payer], amount)
52     remaining_money = [m for m in money if m > 0]
53     wealth_product = remaining_money[0] * remaining_money[1] if len(remaining_money)
           == 2 else 0
54     wealth_product = money[0]*money[1]+money[1]*money[2]+money[0]*money[2]
55     remain_player = [i for i, m in enumerate(money) if m > 0]
56     leave_player = [i for i, m in enumerate(money) if m <= 0]
57     sump =
           probabilities[6]+p0[remain_player[0]][remain_player[1]]+p0[remain_player[1]][remain_player[0]
58     p2 = [p0[remain_player[0]][remain_player[1]]/sump,
           p0[remain_player[1]][remain_player[0]]/sump, probabilities[6]/sump]
59     a2 = [a0[remain_player[0]][remain_player[1]],
           a0[remain_player[1]][remain_player[0]], 0]
60     f1 = p2[0]*a2[0]**2+p2[1]*a2[1]**2+p2[2]*0
61     tra2 = [
62     (remain_player[0], remain_player[1], a2[0]),
63     (remain_player[1], remain_player[0], a2[1]),
64     (-1, -1, 0) # No transfer
```

```
    ]
    remain_player.append(3)
    rounds1 = rounds
    remaining_money = money
    remaining_money[leave_player[0]] = 1
    while all(m>0 for m in remaining_money):
        rounds += 1
        transfer_idx = np.random.choice(len(p2), p=p2)
        payer, receiver, amount = tra2[transfer_idx]
        if payer != -1 and money[payer] > 0:
            remaining_money[payer] -= amount
            remaining_money[receiver] += amount#min(money[payer], amount
    return rounds1,rounds, wealth_product,f1

# 运行模拟
initial_money = 1600
num_simulations = 10000
results = [simulate_game() for _ in range(num_simulations)]

times1, times2,wealth_products,f1 = zip(*results)
mean_time = np.mean(times1)
over_time = np.mean(times2)
mean_wealth_product = np.mean(wealth_products)

print('Mean time until a player exits:', mean_time)
print('Mean time until a game over:', over_time)

#上下界估计
f1max = max(f1)
f1min = min(f1)
S0 = initial_money*initial_money*3
K = (3 * initial_money + np.max(a0))
#\tau的上下界
print((S0-initial_money*initial_money*9/4)/f0)
print(S0/f1+((S0-(3/2*initial_money+np.max(a0))**2)/f0))
#T的下界估计
print((S0+50*(3*initial_money-50))/f1max + (S0 - K**2/4) / f0*(1-f0/f1min))
```

概率排序	编号	名称	极限分布概率
1	10	JAIL	0.063582
2	0	GO	0.035337
3	24	E3	0.031707
4	19	D3	0.031198
5	25	R3	0.030566
6	5	R1	0.029904
7	18	D2	0.029819
8	15	R2	0.029674
9	20	FP	0.029086
10	21	E1	0.028508
11	16	D1	0.028366
12	28	U2	0.028118
13	23	E2	0.027284
14	11	C1	0.027175
15	26	F1	0.027037
16	31	G1	0.026863
17	27	F2	0.026788
18	32	G2	0.026312
19	12	U1	0.026211
20	39	H2	0.026019

概率排序	编号	名称	极限分布概率
21	29	F3	0.025932
22	34	G3	0.025076
23	14	C3	0.024940
24	35	R4	0.024354
25	13	C2	0.023903
26	8	B2	0.023434
27	4	T1	0.023418
28	9	B3	0.023113
29	6	B1	0.022955
30	17	CC2	0.022605
31	37	H1	0.021758
32	3	A2	0.021677
33	38	T2	0.021578
34	1	A1	0.021014
35	33	CC3	0.020698
36	2	CC1	0.016028
37	22	CH2	0.010501
38	7	CH1	0.008816
39	36	CH3	0.008647
40	30	G2J	0.000000

表 8: 极限分布概率排序表