评委一评分，签名及备注	队号： 10466	评委三评分，签名及备注
评委二评分，签名及备注	选题： A	评委四评分，签名及备注

题目：“2048”游戏的数学基础及其取胜策略研究

摘要

近期，有一款名为“2048”的策略益智游戏风靡网络。其简单的规则使游戏易于上手，却又蕴含不少数学知识使人难以获胜。本文从无计算机辅助分析和有计算机辅助分析两个角度，建立了两个求解“2048”问题的模型，并通过数学方法求得了 2048 所能玩到的最大方格数，且将结论推广到 $N\times N$ 的情况。

针对问题一，我们首先通过逻辑分析建立了无计算机辅助的初等策略模型，之后又在此基础上运用博弈论的方法建立了有计算机辅助的博弈策略模型。在初等策略模型中，我们得到了“2048”游戏中合成 2^k 需要的步数及至少需要的空间，进而证明了游戏中将方格按 S 形递减排列的稳定性，从而提出了玩家在游戏中必须采用的取胜策略。在博弈策略模型中，我们将“2048”游戏以博弈论的视角进行解读，建立了描述游戏过程的博弈论模型，并参考极小极大化算法和 alpha-beta 剪枝的思路，通过建立静态估价函数，借助计算机辅助试验，分析玩家的最优决策。该方法的成功率达到了 72%，平均成功操作步数为 895 步。

针对问题二，我们首先分析得到了一个重要的结论：游戏所能达到的最大方格数值仅与游戏本身有关。之后，通过反证法和数学归纳法得出，在 4×4 的情况下，当系统最后出现 4 时得到最大数值，为 131072，在 $N\times N$ 的情况下，当系统最后出现 4 时得到最大数值，为 $2^{N\times N+1}$。

关键字：策略益智游戏；博弈论；静态估价函数；历史启发

“2048”游戏的数学基础及其取胜策略研究

1. 问题的重述

近期，有一款名为“2048”的策略益智游戏风靡网络。其简单的规则使游戏易于上手，却又蕴含不少数学知识使人难以获胜，因而具备一定的研究价值。该游戏局面有4×4共16个方块，初始局面为2个数值为2的位置随机的方块。玩家可控制所有方块同时往上下左右方向移动，相邻相同数值的方块相撞则合并为一个方块，且该方块数值为前一个方块数值的两倍。每次操作后空白处会随机出现一个数值为2或4的方块。最终局面得到数值为2048的方块即为获胜。若16个格子被填满且无法移动即为游戏失败。

我们将首先建立一个模型，探讨能够使得玩家在“2048”游戏中尽量获胜的策略。由于游戏每次移动后出现方块的位置和数值随机，故该模型将从最终获胜的概率和达到目标所要移动方块的次数来衡量取胜策略的有效性。

其次，通过实际操作发现，当游戏达到2048后，一般还能进行移动格子的操作。所以我们将通过建立相关的的模型来分析该游戏中方格所能达到的最大数值。进一步的，我们将探讨在$N\times N$个方格的情况下，游戏所能达到的最大数值。

2. 模型假设

（1）“2048”游戏的版本为问题重述里描述的版本，其中 2 以 80%的概率出现；
（2）玩家在游戏过程中不允许采用“undo”等反悔策略；
（3）玩家在游戏获胜，即达到“2048”后，可以选择继续操作以达到更大数值的方块；
（4）玩家允许使用计算机等设备进行辅助分析。

3. 符号说明

G_k：第k轮游戏；

S_k：第k轮游戏开始时的局面；

s_{ij}：将S_k视为矩阵时S_k的元素；

X_k：第k轮游戏玩家做出的决策；

Y_k：第k轮游戏系统做出的决策；

M_k：第 k 轮游戏玩家做出决策后的局面；

$\overline{S_k}$：第 k 轮游戏开始时的所有可行局面集合；

$\overline{X}_k$：第 k 轮游中玩家的允许决策集合；

$\overline{Y}_k$：第 k 轮游戏系统的允许决策集合；

$\overline{M_k}$：第 k 轮游戏玩家做出决策后的可行局面集合；

f_k：玩家在第 k 轮游中的决策函数；

g_k：系统在第 k 轮游中的决策函数；

$V(.)$：静态估价函数；

n_k：S_k 的空白方格数 n_k；

$\max_k$：S_k 中最大值；

stg_k：S_k 中方格数值的标准差；

$d(.)$：位置赋分函数；

K：$\mathrm{K}=\{\kappa_i\}_{i=1}^{16}$ 为 S_k 对应的序列；

$\mathrm{K}^{(\mathrm{d})}$：$\mathrm{K}^{(\mathrm{d})}=\{\kappa_{(i)}\}_{i=1}^{16}$ 是将 $\mathrm{K}=\{\kappa_i\}_{i=1}^{16}$ 中元素按从大到小排列后的更序序列。

4. 问题的分析

2.1.“2048”取胜策略问题的分析

2.1.1. “2048”取胜策略问题的实质

“2048”游戏的数学依据即为 2 的次方的累加。因此，取得游戏胜利的关键因素在于当前局面的最大值，方块可移动的空间以及当前局面各个方格中数值的关系。这就涉及到以下几个问题：

1. 如何提升当前局面中方块的最大数值。
2. 如何保持当前局面中剩余的空白方格数。
3. 如何缩小当前局面中相邻方块的差值。
4. 如何保持相邻方块的大小顺序关系。
5. 如何增加当前局面中所有方块数值的多样性。

因此，“2048”取胜策略问题的实质就是设计一套操作策略，使得能够在增加方块数值多样性、保持相邻两个方块的大小关系的同时，降低相邻两个方块的

差值，以期得到数值相同、位置相邻的方块，从而通过抵消增加局面中方块的最大值，并增加空白方格的数目。

2.1.2.“2048”取胜策略问题的解决思路

根据上一小节的分析，我们得到了“2048”问题的实质，因此，解决“2048”问题，也就是要解决上一小节提到的五个小问题。对此,我们首先将从定性分析的角度建立了玩家在无计算机辅助情况下的初等策略模型,其包含了一系列独立的策略，玩家依据这些策略进行游戏；之后,我们将结合博弈论、动态规划的一些思想，融合最小最大化算法和 alpha-beta 剪枝的方法，参考无计算机辅助的初等模型的思路，建立一个有计算机辅助的博弈策略模型，从而解决该问题。

2.2.“2048”最大值问题的分析

2.2.1.“2048”最大值问题的实质

从人机博弈的角度来看，“2048”游戏所能达到的最大值问题，实质是在假设玩家采取全局最优策略且系统时刻作出有利于玩家的决策的情况下，分析游戏所能达到的最终局面。

2.2.2.“2048”最大值问题的解决思路

由于在分析最大值问题的时候，已经假设玩家采取的是全局最优策略，且系统也采取有利于玩家的决策，故游戏的最终局面已经与游戏的过程没有关系，而仅与游戏本身的性质相关。所以我们可以单纯的通过分析游戏的性质（包括各种限制），用纯数学的方法得到该问题的解答。也正因为该问题仅与游戏本身的性质相关，故很容易将结论推广到 $N\times N$ 的情形。

5. 模型的建立与求解

5.1 问题一的模型及求解

5.1.1. 无计算机辅助的初等策略模型

（1）重要的结论

定理 5.1.1.在“2048”游戏中，若只出现 2，则合成 2^k 至少需要 $2^{k-1}-1$ 步；若只出现 4，则合成 2^k 至少需要 $2^{k-2}-1$.

证明：用数学归纳法，易得。

定理 5.1.2. 在“2048”游戏中，若只出现 2，则合成 2^k 至少需要 k 个格子；若只出现 4，则合成 2^k 至少需要 $k-1$ 个格子.

证明：用数学归纳法。以只出现 2 为例，显然，当 k=1 时成立；假设当 k=p 时成立，则当 k=p+1 时，为了合成 2^{p+1}，需要两个相邻的 2^p，我们首先合成一个 2^p，使用了 p 个格子；再合成一个 2^p，使用了 p 个格子。由于在合成第二个 2^p 时，第一个合成好的 2^p 占据了一个格子，所以合成 2^{p+1} 总

共至少需要 p+1 个格子。只出现 4 的情况类似，证毕。

定理 5.1.3.在“2048”游戏中，将已经合成好的方格按照从大到小的顺序按照 S 形排列是稳定的，即满足玩家的操作对已经合成的方格排列影响最小.

证明：首先，S 形排列可以逐层将界面填满；其次，对于已经填好的一行，玩家可以在之多三个方向上移动其他方格而不改变已经排列好的方格，故 S 形排列是稳定的。

（2）取胜策略[1]

按照（1）中的三个定理，，结合 2.1 的分析，我们得到了如下的取胜策略。

a. 将最大数放在最角落（如图 5.1[1]）

由于最大数在移动的时候大多无法合并，所以将其固定在最角落保证其不影响其他数值的合并。

图 5.1

图 5.2

b. 以最大数角落为起点按数值从大到小依次横向排列

这样做当较大数附近出现可以合并的数的时候可以及时合并，可以减小最大数与相邻数的数值差距且留出更多空白格子(图 5.2)。

c. 维持最大数所在行始终满格

这样可以保证最大数位置稳定不会随其他格子移动而移动（图 5.3）。

d. 最大数在下方则不按向上方向键，在上方不按向下方向键

这样可以保证最大数始终处于角落，最大数一旦离开角落很难恢复原位（图 5.4）。

[1] 图片 5.1 到图片 5.4 取自 http://www.gezila.com/raiders/11551_pic.html#p=1

图 5.3

图 5.4

5. 在较大数（32 以上）旁边要有数值相近的数

若较大数旁边没有数值相近数，在移动过程中该数易被 2, 4 等小数值方块包围，则占据空白格子的同时妨碍其他方块合并。

5.1.2. 有计算机辅助的博弈策略模型

（1）博弈论模型的建立

由于“2048”游戏中，玩家和系统同时面对相同的局面，拥有完全的信息，且胜负标准唯一，收益之和为零，故可以将其视为完全信息下玩家和系统的零和博弈。

首先，我们假设游戏从玩家的决策开始，即初始局面视为独立于玩家和系统的一个随机的设置。玩家决策后由系统进行决策，之后再由玩家决策，形成交替决策的循环。玩家的第 k 次决策的开始标志着第 k 轮游戏的开始，系统的第 k 次决策的结束标志着第 k 轮游戏的结束。

记第 k 轮游戏为 G_k；第 k 轮游戏开始时的局面记为 S_k，第 k 轮游戏玩家做出的决策为 X_k，系统做出的决策为 Y_k，第 k 轮游戏玩家做出决策后的局面为 M_k。第 k 轮游戏开始时的所有可行局面集合为 $\overline{S_k}$，第 k 轮游中玩家的允许决策集合记为 $\overline{X}_k$，系统的允许决策集合记为 $\overline{Y}_k$，玩家做出决策后的可行局面集合为 $\overline{M_k}$；。玩家在第 k 轮游中的决策函数定记为 f_k，系统在第 k 轮游中的决策函数定记为 g_k。则：

$$f_k:\overline{S_k}\to\overline{X}_k$$

$$S_k\mapsto X_k \qquad (5.1)$$

$$g_k:\overline{M_k}\to\overline{Y}_k$$

$$M_k\mapsto Y_k \qquad (5.2)$$

显然，$\overline{X}_k$由S_k决定，且是集合{上，下，左，右}的一个子集，且当且仅当$\overline{X}_k$为空集时，玩家将无法决策，游戏结束，玩家输。所以，玩家每轮游戏至多能有四种决策可以选择，具体的决策由玩家面对的局面，通过决策函数决定。当玩家作出决策后，局面改变为M_k，并由系统决策。系统的允许决策集合$\overline{Y}_k$由M_k决定，$\overline{Y}_k$的基数取决于M_k的空白方格数、最大数和随机出现 2 和 4 的概率，且当且仅当最大数为 2048 时，$\overline{Y}_k$为空集，系统输。系统做出的决策由系统面对的局面和决策函数决定。当系统做出决策后，局面变更为S_{k+1}，进入下一轮。

（2）博弈论模型的分析

根据上面的博弈论模型，我们知道，由于游戏中系统决策的随机性可知，g_k是一个未知的函数，即我们不能通过分析M_k得出Y_k；其次，系统的决策是随机的，故系统不会考虑将自身利益最大化、将玩家利益最小化；此外，根据前文的分析，$\overline{X}_k$的基数由S_k决定，具体表现在，S_k的空白方格越多、与相邻方格数字相同的格子越多，$\overline{X}_k$越大，玩家才能尽可能的继续游戏；最后，为了让系统输，玩家必须选择能够使得更多方格抵消的策略。

如果从使得玩家赢得游戏这一目的出发，我们下面要做的就是对每一轮游戏G_k，做出一个在全局上有利于玩家的决策X_k，换言之，我们要决定f_k。

（3）静态估价函数的建立[2]

为了衡量一个决策X_k的优劣程度，我们引入静态估价函数的概念。

设$S=\bigcup_{k=1}^{\infty}\overline{S_k}$，其中，当$\overline{X}_k=\varnothing$或$\overline{Y}_k=\varnothing$时，定义$\overline{S_l}=\varnothing,(l\geq k+1)$。定义静态估价函数$V(.)$如下：

$$V:S\to\mathbb{R}$$
$$S_k\mapsto r \qquad (5.3)$$

$V(.)$实质是一个评价体系，它可以对玩家面对的局面打分。$V(.)$是一个“增函数”，即局面S_k越好，$V(S_k)$越大。

由于$V(.)$是对局面S_k好坏的一个量化评定，故首先应明确S_k的好坏。根据前文对问题的分析和初等策略模型可知，满足下列条件的S_k是好的：

（a）S_k的空白方格多；

（b）S_k中最大值大；

（c）S_k中方格的数值离散程度大；

（d）S_k中方格依其数值的大小S型单调排列，且在排列中相邻两个方格的数值足够接近。

令S_k的空白方格数为n_k，S_k中最大值为$\max_k$，S_k中方格的数值离散程度用标准差表示为stg_k，则$V(.)$是n_k、$\max_k$、stg_k的增函数；对于（d），我们将用一个位置赋分函数$d(.)$将其量化。

（4）位置赋分函数及静态估价函数的形式

对于一个给定的局面S_k，其实质是一个4×4的矩阵，其元素为s_{ij}；定义一个序列$\mathrm{K}=\{\kappa_i\}_{i=1}^{16}$，其中

$$\kappa_i=s_{1i} \qquad (\mathrm{i}=1,2,3,4)$$
$$\kappa_i=s_{2,9-i} \qquad (\mathrm{i}=5,6,7,8)$$
$$\kappa_i=s_{3,i-8} \qquad (\mathrm{i}=9,10,11,12)$$
$$\kappa_i=s_{3,17-i} \qquad (\mathrm{i}=13,14,15,16) \qquad (5.4)$$

称$\mathrm{K}=\{\kappa_i\}_{i=1}^{16}$为$S_k$对应的序列。现将K的元素按照从大到小的顺序重排，得到更序数列$\mathrm{K}^{(\mathrm{d})}$，$\mathrm{K}^{(\mathrm{d})}=\{\kappa_{(i)}\}_{i=1}^{16}$。若$\mathrm{K}=\{\kappa_i\}_{i=1}^{16}$和$\mathrm{K}^{(\mathrm{d})}=\{\kappa_{(i)}\}_{i=1}^{16}$完全相同，则

称K是完全良序的；若两者前 i 个元素相同，则称K为 i-良序的，此时 i 称为K的良序数。显然 $d(.)$ 应当是 i 的增函数。

另一方面，考虑K中相邻两个元素的差值。由于K的元素都是 2 的幂，所以我们用 δ_t 表示相邻另个元素幂指数只差小于等于 t 的数对个数，即

$$\delta_t = |\{\kappa_i \in \mathrm{K} \mid i \le 15, |\log_2 \kappa_i - \log_2 \kappa_{i+1}| \le \mathrm{t}\}| \tag{5.5}$$

则对给定的 t， $d(.)$ 是 δ_t 的增函数。

故我们可以定义 $d(.)$ 的表达式为：

$$d = d(i, \delta_t) = A_d i + B_d \delta_t \tag{5.6}$$

其中 A_d 、 B_d 和 t 都是正未知参数。

综上，我们可以定义静态估价函数的表达式为：

$$V = V(n_k, \max_k, stg_k, d) = A_V n_k + B_V \max_k + C_V stg_k + D_V d \tag{5.7}$$

或

$$V = V(n_k, \max_k, stg_k, d) = A_V n_k + B_V \max_k + C_V stg_k + D_V (A_d i + B_d \delta_t) \tag{5.8}$$

其中 A_V 、 B_V 、 C_V 、 D_V 、 A_d 、 B_d 和 t 都是未知的正参数。

（5）模型的求解

由于系统的决策是随机的，通过分析给出玩家每一步的全局最优决策是不可能的，因而只能考虑一定深度范围内的最优解。传统的方法是使用最小最大化算法和 alpha-beta 剪枝，结合静态估价函数，分析一定深度内的所有解。但是，由于最小最大化算法的基础是博弈双方都采取有利于自己、不利于对方的策略，而“2048”游戏中，只有玩家采取这样的策略，系统在决策时只是根据局面的限制随机决策，不考虑利益最大化，因而如果单纯的采取传统的办法，虽然成功率很高，但是由于玩家的决策过于保守，会导致成功步长过长等结果[3]。

为此，我们根据实际，在牺牲一定的成功率的基础上，提出了一个较为均衡的求解算法。算法如下[4]

Step1. 游戏开始。

Step2. 试验评估。

Substep1. 玩家根据当前局面 S_k ，确定允许决策集合 $\overline{X}_k$ ；

Substep2. 对允许决策集合 $\overline{X}_k$ 中的每一个元素 X_k ，以其为第一步，与系统交替进行一定步长的随机决策，并对每一次决策前的局面进行静态估价函数评估，记录每次评估结果，并加总。

Substep3. 重复 Substep2. 的过程，达到预先设定的试验次数后，分别求所有试验结果中，以 X_k 为第一步的既定步长随机决策试验下，每次试验所得的赋权加总静态估价函数值的均值。

Step3. 比较 Step2.-Substep3. 中各个均值的大小，选择均值最大的那一个决策 X_k^* 作为第 k 轮玩家的决策。

Step4. 系统决策。

Step6. 重复 2-4 步，直到游戏结束。

通过上面的算法，经过试验，在确定相关参数的数值后，我们用计算机模拟了“2048”的求解过程。运行结果如下。其中，步长设定为 15，试验总次数为 200。由于定义中的静态估价函数不适合直接编程，也不适合参数调整，所以在编程中将静态估价函数做了形式上的分解。具体参数设置见附录一中的程序。

表 1. 计算机模拟求解结果

编号	是否达到 2048	运行结束后的最大数	成功步长
1	Y	2048	902
2	N	512	-
3	Y	2048	1009
4	Y	2048	987
5	Y	2048	916
6	N	1024	-
7	N	2048	956
8	Y	2048	959
9	Y	2048	909
10	Y	2048	870
11	Y	2048	899
12	Y	2048	926
13	N	1024	-
14	Y	2048	914
15	Y	2048	914
16	Y	2048	940
17	N	1024	-
18	Y	2048	909
19	Y	2048	883
20	Y	2048	924
21	N	1024	-
22	Y	2048	915
23	N	1024	-
24	Y	2048	978
25	Y	2048	910
26	N	1024	-
27	Y	2048	927

28	N	1024	-
29	Y	2048	870
30	Y	2048	918
31	N	1024	-
32	Y	2048	920
33	Y	2048	905
34	Y	2048	914
35	N	1024	-
36	N	1024	-
37	Y	2048	937
38	N	1024	-
39	Y	2048	881
40	Y	2048	897
41	Y	2048	904
42	N	1024	-
43	Y	2048	931
44	Y	2048	931
45	Y	2048	879
46	Y	2048	891
47	N	1024	-
48	Y	2048	924
49	Y	2048	955
50	Y	2048	907

由此可以看出，该组测试中，本算法的成功率约为 72%，平均成功步长约为 895 步。实际中，我们可以提高算法中的测试步长，加大测试次数来提高准确率，但相应的会提高运算分析时间。

特别的，我们还做了 5 次试验测试算法能得到的最大值，结果如下。

表 2. 无限制下的最大值

组数	1	2	3	4	5
最大值	1024	2048	2048	4096	1024

5. 2 问题二的模型及求解

首先我们证明，游戏所能达到的最大方格数值仅与游戏本身有关。

定理 5.2.1.“2048”游戏中，游戏所能达到的最大方格数值仅与游戏本身有关。

证明：①由定理 5.1.2 可知，由于游戏的方格数有限，故游戏不可能无限玩下去，否则最大数值无上限，玩到该数值所需的最小方格数也没有上界，与游戏的方格数有限矛盾。所以，游戏存在最大方格。

②我们再来说明游戏所能达到的最大方格数值仅与游戏本身有关。这里，由于我们考虑的是游戏所能达到的最大方格数，故前提条件是游戏能达到这个数。不妨设这个数为 a，则根据定理 5.1.3，玩家采取有利于自己的策略，游戏结束时的最终局面一定有且仅有一个 a，且没有比 a 大的数。

③另一方面，游戏一定是在系统决策后结束的，即因为玩家的允许决策集合为空集导致了游戏结束。若不然，若系统的允许决策集合为空，则玩家决策后局面没有空格，而玩家的决策不会减少空格数目，所以玩家面对的局面没有空格，这就表明，玩家的决策没有增加空格，即没有合并数字，又由于这种情况下，玩家面对的局面已经没有空格了，所以推得玩家的允许决策集合为空，游戏本应在玩家决策的时候结束，矛盾。

④假设游戏与系统的最后一步决策有关，而由③，此时系统面对的局面有且仅有一个空格，所以系统的允许决策集合只有 2 和 4 两个元素。不妨设系统做出 2 的决策导致游戏结束，那么假设系统做出的是 4 的决策，如果游戏仍然结束，则说明游戏与系统的最后决策无关；若游戏没有结束，则可以进行下去，这种进行的过程一定伴随着方格的合并和方格数值的增加，由①，这一过程一定会终止，即存在一轮游戏，系统无论如何做决策都不能继续游戏。故游戏的结束与系统的最后决策无关。

⑤类似上一步的推理，容易得出，游戏的结束与所有决策无关。

综上，“2048”游戏中，游戏所能达到的最大方格数值仅与游戏本身有关！

由定理 5.2.1，结合定理 5.1.1、定理 5.1.2、定理 5.2.3，我们很容易知道下面推理的合理性

5.2.1. 方格为 4×4 时的最大值情况

由于数值“2”“4”出现的随机性，不妨作以下讨论：

1. 假设出现的数值均为“2”

最后出现在左上角的数值为“2”，倒数第二步以及倒数第三步出现的数值为“2”，则可知，在左上第二格的数值合并为“4”，根据数学归纳法以此类推可知最终局面为：

表 3. 当每次移动方格后只出现“2”的最终局面

2	4	8	16
256	128	64	32
512	1024	2048	4096
65536	32768	16384	8192

2. 假设出现的数值均为“4”

同于上述讨论，最后出现在左上角的数值为“4”，倒数第二步以及倒数第三步出现的数值为“4”，则可知，左上第二格的数值合并为“8”，根据数学归纳法你以此类推可知最终局面为：

表 4. 当每次移动方格后只出现“4”的最终局面

4	8	16	32
512	256	128	64
1024	2048	4096	8182
131072	65536	32768	16384

3. 假设出现的数值中仅一次为“4”，其余都为“2”

若是在下图所示局面时出现在左上角为“4”

表 5. 系统面对的最终局面

	4	8	16
256	128	64	32
512	1024	2048	4096
65536	32768	16384	8182

由上述讨论可知，此时新出现的“4”与右侧相邻的“4”合并为“8”，新生成的“8”再与右相邻的“8”合并为“16”……依次类推，最大值在左下角数值依然为 131072，最终局面与表 4 相同。

若是任意一次出现在任意位置“4”，即最后一次出现的数值为“2”，则同上述第一种讨论情况，即最终局面如表 3 所示。

4. 推广假设条件，任意出现数值“2”“4”。

结合前三种情况的讨论及数学归纳法可知，当局面如表 5 所示时，出现的数值为“2”则最终局面与表 3 相同。出现的数值为“4”时则最终局面与表 4 相同。

综上所述，“2048”在 16 个方格中理论可以达到的最大值为 131072，最终局面为

4	8	16	32
512	256	128	64
1024	2048	4096	8182
131072	65536	32768	16384

图 5.5

5.2.2. 方格为 $N\times N$ 时的最大值情况

1.若 N 为偶数，则根据构造的 S 型模型以及最后出现的数在左上角，可知最终局面的最大值在左下角。

结合 5.2.1 中的讨论，运用数学归纳法和递推法，可知，当最后出现的数值为“2”时则有最终局面为：

表 6.最后出现数值为“2”且 N 为偶数时的最终局面

2	4	8	16	…	2^{N}
2^{2N}	…	…	…	…	2^{N+1}
…					
…					
…					
$2^{N\times N}$					

当最终局面为“4”时则有最终局面为：

表 7.最后出现数值为“4”且 N 为偶数时的最终局面

4	8	16	32	…	2^{N+1}
2^{2N+1}	…	…	…	…	2^{N+2}
…					
…					
…					
$2^{N\times N+1}$					

2.若 N 为奇数，则根据构造的 S 型模型以及最后出现的数在左上角，可知最终局面的最大值在右下角。

结合 5.2.1 中的讨论，运用数学归纳法和递推法，可知，当最后出现的数值为“2”时则有最终局面为：

表 8.最后出现数值为“2”且 N 为奇数时的最终局面

2	4	8	16	…	2^{N}
2^{2N}	…	…	…	…	2^{N+1}
…					2^{3N}
…					…
…					…
…					$2^{N\times N}$

当最终局面为“4”时则有最终局面为：

表 9.最后出现数值为“2”且 N 为奇数时的最终局面

4	8	16	32	…	2^{N+1}
2^{2N+1}	…	…	…	…	2^{N+2}
…					2^{3N+1}
…					…
…					…
…					$2^{N\times N+1}$

综上所述，当“2048”扩展到$N\times N$个方格的局面时，最大值为$2^{N\times N+1}$

6. 模型的评价

针对第一个问题，我们建立了两个模型。第一个模型是无计算机辅助的初等策略模型，该模型通过分析得出了能够指导玩家赢得游戏的相关策略，经过实际操作检验，这些策略是有效的。但是该模型只运用了逻辑分析建模，虽然给出的策略是有效的，但是在运用这些策略时，需要玩家仔细判别，依赖于玩家的经验，更重要的是，该模型没有针对玩家面对的局面给出切实可行的决策。第二个模型是有计算机辅助的博弈策略模型，该模型通过建立博弈论模型，详细分析了玩家和系统在游戏中的情况，并根据极小极大化算法和 alpha-beta 剪枝算法的思想，结合实际情况，通过建立静态估价函数评估评估一定深度内玩家不同决策的结果，给出了玩家每一步的局部最优决策，兼顾了成功率和成功步长，针对问题一给出了高效的解决方案。

针对第二个问题，我们通过纯数学分析，结合有计算机辅助的博弈策略模型，用反证法和数学归纳法给出了完整的解答，思路清晰，分析简单易懂。

7. 模型的改进与推广

7.1 模型的改进

1. 针对第一个问题，有计算机辅助的博弈策略模型虽然给出了高效的解决方案，但是在参数设置方面仍具有主观随意性。在分析各个因素的影响时，也没有做到定量分析，也忽略了其之间的相互联系。后续可以通过大量的计算机模拟试验，做一些统计分析的工作，以确定各个参数的最佳值，并明确各个因素的关系。

2.针对第一个问题，我们提出的模型需要借助计算机辅助分析。也即没有做到提出一种硬解的策略。事实上，由于“2048”游戏具有很多随机性，故想要提出一种类似魔方求解的确定性通解几乎是不可能的。后续可以从降低随机性出发，逐步脱离计算机辅助分析，提出一种简单易行的解决策略。

7.2 模型的推广

本模型的解决思路，尤其是有计算机辅助的博弈策略模型的解决思路，可以发展后应用于一类人机博弈游戏的开发和通解研究上。

参考文献

[1]. Heinrich, 2048 在理论上是否可以无限玩下去,http://m.zhihu.com/question/23492860,2014-5-25
[2]. Wikipedia, Evaluation function,http://en.wikipedia.org/wiki/Evaluation_function,2014-5-25
[3]. Ovolve, A simple AI for 2048,https://github.com/ov3y/2048-AI,2014-5-24
[4].goldengrape. solve 2048 game.https://github.com/goldengrape/2048.2014-5-24

附录一：有计算机辅助的博弈策略模型测试程序

1.主程序

```
clear all
close all
count=0;
for ex=1:50
    ex
    situation=2.^(reshape(randperm(16),4,4)<=2)
    originalsituation=situation;
    situation=log2(situation);
    for m=1:5000
        m;
        largestblock(m)=max(max(situation));
        for test=1:200
            pretesteststep=15;
            dir=floor(rand(1,pretesteststep).*4+1);
            firstmove=dir(1);
            tsituation=situation;
            testscore=0;
            for step=1:pretesteststep
                tsituation=move(tsituation,dir(step));
                testscore=testscore+situationcore(tsituation);
            end
            testresult(test,:)=[firstmove max(testscore)];
        end
        maxscore=0;
        for i=1:4
            meanscore(i)=(testresult(:,2)'*(testresult(:,1)==i))/(sum
            ((testresult(:,1)==i)));
            if meanscore(i)>=maxscore
               maxscore=meanscore(i);
               bestdir(m)=i;
            end
        end
        if maxscore==0
           break;
        end
```

```
        nsituation=move(situation,bestdir(m));
        situation=nsituation;
        showsituation=2.^situation;
    end
    situation=2.^situation;
    finalsituation=situation
end
```

2.移动方块的函数（move.m）

```
function nsituation=move(osituation,dir)
m=dir;
situation=rot90(osituation,m);
for k=3:-1:1
    for i=3:-1:1
        for j=1:4
            if situation(i+1,j)~=0
                if situation(i+1,j)==situation(i,j)
                    situation(i+1,j)=situation(i+1,j)+1;
                    situation(i,j)=0;
                end
            else
                situation(i+1,j)=situation(i,j);
                situation(i,j)=0;
            end

        end
    end
end
posofnewblock=randperm(4);
R=rand();
if R<0.2
   anumofnewblock=2;
else
   anumofnewblock=1;
end
for i=1:4
    if situation(1,posofnewblock(i))==0
        situation(1,posofnewblock(i))=anumofnewblock;
        break;
    end
end

nsituation=rot90(situation,-m);
```

```
end
```

3.静态估价函数（situationcore.m）

```
function y=situationcore(osituation)
osituation=(osituation);
largestblock = max(osituation(:));
remainblocks = sum(sum((osituation==0)));
situationstd = std2(osituation);
seq(1)=osituation(1,1);seq(2)=osituation(1,2);seq(3)=osituation(1,3);
seq(4)=osituation(1,4);seq(5)=osituation(2,4);seq(6)=osituation(2,3);
seq(7)=osituation(2,2);seq(8)=osituation(2,1);
seq(9)=osituation(3,1);seq(10)=osituation(3,2);
seq(11)=osituation(3,3);seq(12)=osituation(3,4);
seq(13)=osituation(4,4);seq(14)=osituation(4,3);
seq(15)=osituation(4,2);seq(16)=osituation(4,1);
positionscore=0;
cseq=sort(seq,'descend');
if(seq(1)==max(seq))&(seq(2)==cseq(2))&(seq(3)==cseq(3))&(seq(4)==cse
q(4))&(seq(5)==cseq(5))&(seq(6)==cseq(6))&(seq(7)==cseq(7))&(seq(8)==
cseq(8))
   positionscore=positionscore+800;
elseif
(seq(1)==max(seq))&(seq(2)==cseq(2))&(seq(3)==cseq(3))&(seq(4)==cseq(
4))&(seq(5)==cseq(5))&(seq(6)==cseq(6))&(seq(7)==cseq(7))
     positionscore=positionscore+750;
elseif
(seq(1)==max(seq))&(seq(2)==cseq(2))&(seq(3)==cseq(3))&(seq(4)==cseq(
4))&(seq(5)==cseq(5))&(seq(6)==cseq(6))
    positionscore=positionscore+700;
elseif
(seq(1)==max(seq))&(seq(2)==cseq(2))&(seq(3)==cseq(3))&(seq(4)==cseq(
4))&(seq(5)==cseq(5))
    positionscore=positionscore+650;
elseif
(seq(1)==max(seq))&(seq(2)==cseq(2))&(seq(3)==cseq(3))&(seq(4)==cseq(
4))
    positionscore=positionscore+600;
elseif (seq(1)==max(seq))&(seq(2)==cseq(2))&(seq(3)==cseq(3))
    positionscore=positionscore+500;
elseif (seq(1)==max(seq))&(seq(2)==cseq(2))
    positionscore=positionscore+300;
elseif (seq(1)==max(seq))
    positionscore=positionscore+200;
```

```
end
for i=5:8
    if (seq(i)>=seq(i+1))
        positionscore=positionscore+15;
        if (seq(i+1)>=seq(i+2))
           positionscore=positionscore+20;
           if (seq(i+2)>=seq(i+3))
               positionscore=positionscore+30;
               if (seq(i+3)>=seq(i+4))
                   positionscore=positionscore+40;
               end
           end
        end
    end
end
for i=1:7
    if (seq(i)-seq(i+1))<=1
       positionscore=positionscore+300;
    elseif (seq(i)-seq(i+1))<=2
       positionscore=positionscore+150;
    end
end
t=[1,10,3,30];
y=[largestblock remainblocks situationstd positionscore]*t';
if remainblocks==0
    y=0;
end

end
```

附录二：有计算机辅助的博弈策略模型求解程序

1.主程序

```
clear all
close all
situation=2.^(reshape(randperm(16),4,4)<=2)
originalsituation=situation;
situation=log2(situation);
for m=1:5000
    m;
    largestblock(m)=max(max(situation));
    for test=1:200
        preteststep=15;
        dir=floor(rand(1,preteststep).*4+1);
        firstmove=dir(1);
        tsituation=situation;
        testscore=0;
        for step=1:preteststep
            tsituation=move(tsituation,dir(step));
            testscore=testscore+situationcore(tsituation);
        end
        testresult(test,:)=[firstmove max(testscore)];
    end
    maxscore=0;
    for i=1:4
        meanscore(i)=(testresult(:,2)'*(testresult(:,1)==i))/(sum
        ((testresult(:,1)==i)));
        if meanscore(i)>=maxscore
           maxscore=meanscore(i);
           bestdir(m)=i;
        end
    end
    if maxscore==0
       break;
    end
    nsituation=move(situation,bestdir(m));
    situation=nsituation;
    showsituation=2.^situation
    if max(max(situation))==11
```

```
            break;
        end
    end
    situation=2.^situation;
    finalsituation=situation
```

2.移动方块的函数（move.m）

同附录一

3.静态估价函数（situationcore.m）

同附录一