基于极小化极大的 2048 游戏算法设计

赵　胤，3170105124

黄亮彬，3160300097

（赵胤，17 级信息管理与信息系统（统计学交叉创新平台）；黄亮彬，16 级信息管理与信息系统）

摘　要:　“2048”是一款流行的数字游戏，由于其简单的规则和难以掌握的游戏玩法，近年来在大众间大受欢迎。本文介绍利用极小化极大算法（MiniMax）实现计算机自动运行“2048”游戏，并使用 α-β 剪枝算法（Alpha-Beta Pruning）、变邻域搜索算法（Variable Neighborhood Search, VNS）和模拟退火算法（Simulated annealing, SA）的思维对算法进行了改进，并用 python 语言对其进行实现，最终获得了相较于一般人类玩家较好的成绩。

关键词:　极小化极大；2048 游戏；对弈模型

1 引言

1.1 2048 游戏

2048 是一款简单而出色的游戏，其游戏逻辑的代码实现是较为容易的。游戏本身可以抽象为一个在离散的状态空间中通过玩家简单地输入（上，下，左，右）从而在不同的状态中变化的过程。

游戏的规则很简单，在一个可以控制的输入（上，下，左，右）后，所有的方块会先向同一个方向运动，直至所有方块都移动到底。此时若输入方向上存在相邻两个方块的值相同，则他们会在偏向输入的方向上的位置合并成为他们的和，而留出的空格被前面的方块移动填补。特别地，若有三个相同值的方块相邻，则优先合并较前的两个。如果某个输入不能造成任何一个方块发生变化，那么这个输入是无效的。每次玩家输入之后，系统会给出一个输入，这个输入在随机的空格处生成一个 2 或者 4。

在传统规则中，最终得到一个“2048”的方块就算作游戏胜利。但在游戏过程中，得到一个“2048”的方块对于一般游戏者和算法而言都是较为简单的，故在得到一个“2048”方块后，游戏仍可以继续进行，直到进行至某一步之后任何输入都是无效的，此时称为游戏结束，以游戏结束时最终得分的高低作为评判游戏者水平的依据。

2048 游戏的计分规则较为简单，初始分数为 0，在每一次输入后，任何通过合并新生成的方块都会给总分加上生成方块的值，直到游戏结束。

1.2 2048 游戏算法的发展

虽然 2048 游戏从逻辑上来看是简单的，但长期以来人们在进行 2048 游戏的过程中基本是基于启发式的想法，通过反复游戏获得的经验来修正自己的行动准则的。并且在一些程序员提出对 2048 算法进行代码化求解时，其基本思想仍然是启发式的，并在此基础上对游戏进行了一定的代码模型构建与实现。实证表明，2048 游戏的运算规模是庞大的，目前最好的算法是在使用 C 语言进行了效率优化并事先对游戏移动规则进行打表存储的前提下，大大缩减了深度优先搜索计算所需的时间，从而获得较好的直观结果的，在算法理论上的发展并未对现有的一些较有效率优势的算法进行应用。

2 2048 游戏数学模型的构建

2.1 符号表记

我们对 2048 游戏中出现的可视性内容进行符号化定义如下：

a_{ij}：第i行第j列位置的数字，为 2 的整数次方，特别地，将空格记为 0，其中$i,j \in \{1,2,3,4\}$。

S_i：记录了第i次输入后当前游戏内每一个位置的数字，简称为状态。

$E(S)$：对状态S进行评价的评价函数，$E(S)$愈大说明该状态愈好。

p_i：玩家的第i次输入，随机地有$p_i \in \{up, down, left, right\}$，使面板上的数字朝$p_i$方向移动与合并，若随机选取的$p_i$不能造成$S_{i-1}$的变化，则重新随机选择$p_i$，直至完成一次成功的输入。

$o_i(x, y, r)$：电脑的第i次生成，随机选取$x, y \in \{1,2,3,4\}$,若$a_{xy} = 0$，则将a_{xy}随机替换为r，若$a_{xy} \neq 0$，重新随机选取x, y，直至成功完成一次替换。按传统 2048 规则[1]，r满足离散分布$\begin{pmatrix} r & 2 & 4 \\ p & 0.9 & 0.1 \end{pmatrix}$。

t_i：进行到状态S_i的总分数，$t_i = t_{i-1} + \sum(p_i$输入后合并得到的新方块值$)$。

n：若对于S_n，任何p_{n+1}均不能造成变化，则判定游戏结束，n为步数。

2.2 游戏模型构建

一局 2048 游戏可以抽象为如下的模型：

Step 1 机器进行两次$o_0(x, y, r)$，生成一个初始状态S_0，初始化$t_0 = 0$。

Step 2 对状态S_{i-1}，玩家进行一次输入p_i，更新得分t_i。

Step 3 对 Step 2 的结果，机器进行一次输入$o_i(x, y, r)$。

Step 4 若判定游戏未结束，反复进行 Step 2、3 两步操作；若$i = n$时游戏结束，记最终状态为S_n，最终得分为t_n。

END

对于该模型，我们的目标是最大化分数t_n，由于$t_i \geq t_{i-1}$恒成立，这个目标自然地几乎“完全”等价于最大化步数n，即尽量使游戏进行下去。在实际应用中，最大化步数n的目标相较于前者，更适合实现。

2.3 状态S_i的评价函数$E(S)$的构造

为了判断进行不同输入得到的结果状态的优度，我们构建了评价函数$E(S)$。$E(S)$由如下的指标构成，其选取要么是启发性的、要么是在大量样本的验证下显著有效的。

2.3.1 较大值e_1

基于最大化分数t_n的目标，我们希望生成数值较高的块以直接地获得分数，随着游戏的进行，合并出一个更大的块获得的收益是呈指数增长的。我们不希望在最大值过大时对分数整体造成影响，故而我们采用了对数函数对其线性化，即应用了以下式子

$$e_1 = log_2 a_1 + log_2 a_2 + log_2 a_3$$

其中a_1, a_2, a_3是从大到小排列的前三个a_{ij}值。

2.3.2 平滑性e_2

基于启发式的想法和大量样本验证，一个相邻方块值相差较小的状态是更好的，因为这有助于在之后的步骤中将相邻方块值的差缩小至 0，从而合并这两个方块。故而我们应用了

$$e_2 = \sum_{i=1}^{3}\sum_{j=1}^{4}\left|log_2 a_{ij} - log_2 a_{(i+1)j}\right| + \sum_{i=1}^{4}\sum_{j=1}^{3}\left|log_2 a_{ij} - log_2 a_{i(j+1)}\right|$$

2.3.3 空余度e_3

基于最大化步数n的目标，我们认为一个具有更多空格数k的状态是较好的，因为在游戏结束（此时$k = 0$）时步数不再能够增加，故而可以认为空格数k更大的状态具有发展更多后续步数的潜力。

基于大量样本的验证，可以发现，在k较大时，k值的变化对状态的实际优度的影响是较小的，而在k较小，趋近与 0 时，k值的变化对状态的实际优度的影响会急剧增大。由于$k \in [0, 15]$，故而我们应用了

[1] 实际上 2048 游戏对r的分布没有规范化的定义，且该分布在一定范围内对 2048 游戏算法的影响是不显著的。

$$e_3 = ln\frac{k+1}{32}$$

2.3.4 汇聚度e_4

基于启发式的想法和大量样本验证，当空格数始终为k时，我们认为这些空格较为汇聚的状态是较好的。故而我们应用了

$$e_4 = \sum_{i=1}^{3}\sum_{j=1}^{4}(a_{ij} = 0 \cup a_{(i+1)j} = 0) + \sum_{i=1}^{4}\sum_{j=1}^{3}(a_{ij} = 0 \cup a_{i(j+1)} = 0)$$

其中逻辑运算的真值取 1，假值取 0。

2.3.5 单调性e_5

基于启发式的想法和大量样本验证，我们认为一个具有显著良好单调性的状态是较好的，其中单调性指方块从左到右、从上到下均遵从递增或递减。一般来说，越单调的状态越好。故而我们应用了

$$e_5 = \sum_{i=1}^{3}\sum_{j=1}^{4}(a_{ij} > a_{(i+1)j}) + \sum_{i=1}^{4}\sum_{j=1}^{3}(a_{ij} > a_{i(j+1)})$$

其中逻辑运算的真值取 1，假值取-1。

2.3.6 总优度评价函数$E(S)$

在上述对评价指标量化的公式中，各个指标的值的相对大小差异较大，且其对状态实际优度的影响并不相同。故而通过对上述评价指标的加权求和，我们可以得到

$$E(S) = \sum_{i=1}^{5} c_i e_i$$

其中c_i是对应指标e_i的权重，经由启发式地设定初始值，再通过大量样本的计算结果进行调整，我们在实践时应用了权重矩阵

$$C = (2.72 \quad 10 \quad 72 \quad 1 \quad 1)$$

特别地，为了最大化步数n的目标，我们对状态S进行一次p_i输入判断，若任何p_i都不能造成S发生变化，即S是一个结束状态时，我们直接有

$$E(S) = -\infty$$

来避免在可能获得更大步数n时过早地结束了游戏。

3 2048 优化算法的实现

在实现过程中，我们为了兼顾效果与效率，主要运用了极小化极大算法（MiniMax），并用变邻域搜索算法（Variable Neighborhood Search, VNS）、模拟退火算法（Simulated annealing, SA）和$\alpha - \beta$剪枝（Alpha-Beta Pruning）的思想对其进行了改进。

3.1 使用极小化极大算法进行决策

极小化极大算法（MiniMax），是一种找出失败的最大可能性中的最小值的算法。极小化极大算法常用于棋类等由两方较量的游戏和程序，这类程序由两个游戏者轮流，每次执行一个步骤。我们众所周知的五子棋、象棋等都属于这类程序，所以说极小化极大算法是基于搜索的博弈算法的基础。该算法是一种零总和算法，即一方要在可选的选项中选择将其优势最大化的选择，而另一方则选择令对手优势最小化的方法。

极小化极大不找理论最优解，因为理论最优解往往依赖于对手是否足够愚蠢，极小化极大中我方完全掌握主动，如果对方每一步决策都是完美的，则我方可以达到预计的最小损失格局，如果对方没有走出完美决策，则我方可能达到比预计的最悲观情况更好的结局。

上述 2048 游戏模型是一个完全信息（Perfect Information）模型，玩家知道在输入p_i之前所有的步骤。在模型中，p_i是玩家可控制的输入，而$o_i(x,y,r)$是玩家不可控制的电脑输入，这两种输入即可对应于极小化极大算法中的博弈双方，故每一次状态的改变都相当于进行了极小化极大算法中的一次博弈过程，故而应用极小化极大算法，效果是显著的。

在 2048 游戏模型中应用极小化极大算法的基本步骤：

Step 1　输入（状态搜索深度$d \in N^*$，状态S_i）

Step 2　构建基于状态S_i，深度为$2d$的格局树（S_i为第 0 层），奇数层是 min 节点，以所有可能进行的操作p_{i+1}生成的状态S作为节点；偶数层是 max 节点，以所有可能进行的操作$o_{i+1}(x,y,r)$ 生成的状态S作为节点。

Step 3　在最大深度$2d$的子节点处，使用评分函数$E(S)$对所有S进行评价。

Step 4　从$2d-1$层的节点开始，自底向上为该层节点赋值，其中 max 节点取子节点最大值，min 节点取子节点最小值。

Step 5　输出第 1 层节点的最大值p_{mm}，作为玩家对S_i的实际输入p_{i+1}，再由电脑进行随机的输入$o_{i+1}(x,y,r)$，获得S_{i+1}。

Step 6　若判定游戏未结束，则将S_i替换为S_{i+1}，d不变，从 Step 1 开始循环；若判定游戏结束，输出t_n。

END

3.2 使用$\alpha-\beta$剪枝进行的算法效率改进[2]

$\alpha-\beta$剪枝算法（Alpha-Beta Pruning）是基于以下思想和步骤的：

在每一个节点都由 alpha 和 beta 两个值来确定一个范围 [alpha, beta]，alpha 值代表的是下界，beta 代表的是上界。alpha 被初始化为$-\infty$，beta 被初始化为$+\infty$。自下而上地每搜索一个子节点，该节点的 alpha 和 beta 都会按规则进行更新。

更新规则为：max 节点继承父节点的 beta 值后，用其子节点的最大值替换其自身的 alpha 值；min 节点继承父节点的 alpha 值后，用其子节点的最小值替换其自身的 beta 值。

如果某个节点出现了 beta≤alpha 的情况，则可以证明，不用搜索该节点更多的子树了，未搜索的这部分子树将被忽略，这个操作就被称作$\alpha-\beta$剪枝。

上述的 2048 算法是一个标准的极小化极大算法，故可以直接应用$\alpha-\beta$剪枝，在一定程度上将规模降低，但直观地，其规模仍是指数级的，且在d较小时其效果不显著，故对应下一段中提到的d值，仅在$d=3$时应用$\alpha-\beta$剪枝算法改进效率。

3.3 使用变邻域搜索算法思想进行的算法效率改进

可以证明，在d不变时，格局树的规模大小m与状态S_i的空格数k具有正相关关系，且随着状态S_i的空格数$k(k \in [0,15]$且$k \in \mathrm{Z})$线性上升，格局树的规模大小程显著的指数上升趋势，其相关比大约为$m \propto O(4^{k^d})$。这使得在实际应用中空格数k较大（尤其是开局阶段）时，算法的效率是相当低下的。

同时地，我们通过大量样本验证发现，在k较大时，d的大小对单次循环中获得的p_{i+1}的影响是不显著的；而在k较小时，d的大小对单次循环中获得的p_{i+1}具有较为显著的影响。

基于以上两点，我们使用变邻域搜索算法（Variable Neighborhood Search, VNS）的思想，对原极小化极大算法进行了改进：

格局树的构建，实际上就是一个邻域搜索算法过程的前半部分，搜索深度d的大小对应邻域的大小。变邻域搜索算法的核心便是在搜索过程中，邻域大小是实时可变的。

---

[2] 参考 http://web.cs.ucla.edu/~rosen/161/notes/alphabeta.html

在k较大时，邻域的大小对结果的影响较低，此时我们采用较小的领域，即使用较小的搜索深度d。

在k较小时，邻域的大小对结果的影响较大，此时我们采用较大的领域，即使用较大的搜索深度d。

具体到原算法中，我们在 Step 6 中对k进行判断，若$k \in [5,15]$则使用$d = 1$；若$k \in [0,4]$则使用$d = 3$。

3.4 使用模拟退火算法思想进行的辅助决策改进[3]

极大化极小算法是一种悲观算法，即假设对手每一步都会将我方引入从当前看理论上价值最小的格局方向，即对手具有完美决策能力。但在 2048 游戏模型中对手（电脑生成）并非完美的，而是一个随机生成$o_i(x,y,r)$，这使得极大化极小算法在大部分时候并未在趋近实际最优解——即便最优解在决策后才是可知的。

在之前算法讨论中，我们在单次 2048 游戏模型中的目标是最大化分数t_n，但实际上，我们执行的是一个多次重复的 2048 游戏模型，此时我们的目标是最大化分数序列$\{t_n\}$。

在实际操作过程中，我们通过反复运行极大化极小算法，给出了一系列的结果t_n，但由于极大化极小算法并不趋向最优解而是趋向于悲观解，$\{t_n\}$的方差σ_t^2是相对较小的，其结果稳定于均值$\overline{t_n}$附近。

但作为一个以刷新纪录为目标的游戏，对于序列$\{t_n\}$，实际上我们更为关心的是其中的最大值t_{max}的大小而非其均值$\overline{t_n}$的大小。因此，我们能够接受并欢迎以一定的均值$\overline{t_n}$大小的代价换取$\{t_n\}$的方差σ_t^2增大，从而得到更大的t_{max}。

为此我们在$d = 1$引入模拟退火的思想（$d = 3$ 时使用了$\alpha - \beta$剪枝，剪枝后的信息可能不足以满足使用模拟退火）。在格局树收敛至第 1 层时，算法处于 max 节点，本应将选择得分最高的输出p_i。但应用改进后，此时计算其余输出p_j与p_i的差值，分别记为$\Delta t_{ij} = p_j - p_i$，显然有$\Delta t_{ij} < 0$，则以概率$e^{\frac{\Delta t_{ij}}{T}}$接受该输出$p_j$替代$p_i$作为实际输出。

模拟退火要求T是逐渐下降且有收敛极限的，在这里我们自然地取$T = k + 1$，其中k为空格数。

4 算法的代码实现与结果

4.1 算法的代码实现与优化

我们使用了 python 语言对算法进行了实现，并尝试优化其效率。代码的构建顺序为①游戏逻辑实现②评价函数实现③算法逻辑实现④算法优化实现⑤检测代码与可视化。在数据结构方面并没有进行较多地编写与设计，单纯地使用了二维 list 对状态数据进行储存，对每一次输入产生的状态变化也是实时计算的，故而效率较低。整体代码与文件见文档附件。

4.2 代码的不足之处与可优化空间

在格局树的构建过程中，格局树所选取的数据结构运行效率是显著低下的，这成为了拖累代码运行效率的主要原因。鉴于代码层面的算法优化程度仍然较低，若能对代码结构和数据结构进行好的优化，结合一些科学计算库，在可接受时间范围内有望将搜索深度拓展至$d \geq 8$，这对算法的结果提升将会是显著的。

4.3 部分结果展示

在这里，我们附上一些在算法研究过程中考虑到的有代表意义的算法的简介与算例结果，每种算法给出 100 个最终得分t，平均得分$\bar{t}$，最高得分t_{max}，运行 100 个算例总消耗时间$time$，以展示在数周的研究过程中，算法的发展情况。特别地，为了对比我们对后两种算法输出了他们的方差σ_t^2。

4.3.1 随机选取算法

随机选择可行的方向，记录了 2048 游戏在没有任何优化算法时的得分情况。

[3] 此处用到的思想来源于模拟退火算法，但并未完全应用其全过程。

1060	2140	2572	600	308	432	676	716	2508	1128
364	2376	464	1536	1204	1132	1260	2648	576	896
1268	1772	1384	1140	344	1044	1216	768	940	500
1104	1044	1356	1052	584	664	436	1140	1468	1184
584	1304	1280	1712	1392	704	1240	1060	2232	1088
960	2300	500	808	1448	1576	1028	1244	456	256
564	1392	1076	2680	1032	644	1324	984	1132	764
2292	1364	700	540	1020	1032	1028	656	1024	420
604	892	632	1144	780	1120	576	1492	700	1156
532	700	1296	1120	1284	1004	2336	1384	1264	1364

$\overline{t} = 1112.24 \ \ t_{max} = 2680 \ \ time = 0.839s$

4.3.2 启发式算法

较多人在游戏中使用的，将方块尽量向一个角落移动的启发式算法。

1668	3164	1316	2436	5700	1480	1268	3048	2568	2492
3052	3180	460	2728	1120	4728	2348	3184	3392	5672
2752	2628	4152	5712	2792	3152	4168	3240	1588	2788
1468	3276	3412	1464	3168	1824	1564	3624	3272	3056
4536	2920	4216	3184	1408	5448	6164	1424	7976	2580
3148	1568	2700	3204	3052	3484	2268	1560	3460	2536
4900	4012	3852	3504	2916	2280	3524	2880	6188	3308
3576	3040	1612	2956	3892	3948	3700	1864	2068	900
2268	6060	2472	1520	3056	5244	6976	3364	5492	1440
5428	908	3856	6420	5768	6124	1752	3444	5944	1840

$\overline{t} = 3252.36 \ \ t_{max} = 7976 \ \ time = 45.9s$

4.3.2 贪婪算法

深度优先搜索可能取得最优的方向，不考虑随机生成带来的不良后果。

16444	30456	43992	16128	12140	5640	30884	16632	18612	36256
27616	36340	13532	27016	37156	27768	15584	26700	6360	26776
17936	10896	14788	21868	33628	33372	31644	7140	14468	3320
14128	16924	29588	39096	21792	8064	12516	27276	7192	31576
37888	16772	27824	18472	37892	12696	32708	14040	28064	16356
24744	27416	21936	36180	27272	11104	16132	36152	16348	16384
13536	26076	60744	23544	16096	16844	33248	37828	28228	8852
27580	28936	27340	16960	16412	25816	18148	14460	18572	29588
21400	36988	30132	35824	36336	32520	10236	33328	28252	27616
7144	15444	22180	32496	27616	16984	7428	16404	32944	31856

$\overline{t} = 23375.60 \ \ t_{max} = 60744 \ \ time = 1287s$

4.3.3 极小化极大算法

本文中主要介绍的极小化极大算法，使用了可变深度和$\alpha - \beta$剪枝。

15660	50140	47980	13700	22952	33532	53512	53008	28856	35388
50396	16692	21856	46532	20360	48560	20816	21360	63120	16816
27656	81140	16140	78628	36048	16272	18832	43432	29952	16576

22940	35692	48324	58072	24832	36196	21582	11956	43752	37156
45862	43278	25582	21964	19556	16240	10220	18372	48700	34692
32304	42320	22696	22420	30732	9504	44272	20956	42900	22492
41472	23084	34384	18384	35496	42992	6608	47144	26264	36644
47682	95382	42912	22272	47872	4116	9528	48604	36684	68392
48572	48276	47868	43820	45900	41948	36528	49108	21752	21668
21828	23620	36728	48682	21152	21176	75128	34208	95808	47700

$\bar{t} = 35288.88\ \ t_{max} = 95808\ \ \sigma_t^2 = 181380271.6\ \ time = 1278s$

4.4.3 使用模拟退火优化的极小化极大算法

上述算法用模拟退火结合优化后的结果。

39624	51404	62104	27360	20844	40644	39112	23392	27652	39036
39980	37958	21262	73424	46012	51524	10224	58072	20592	52828
13728	28284	44360	50460	23896	22452	50272	88060	27984	39300
46252	19832	47196	90592	50404	28132	17744	28716	102068	20528
23640	11216	34308	101324	21744	85932	23020	43312	21884	10576
45704	25668	23508	43304	5232	12540	23372	18192	53078	24744
3796	21920	74020	24156	51336	14880	49312	41516	8156	18300
86796	45496	23440	84872	83944	73552	22456	22936	18068	49896
51424	52932	55748	25348	33436	38688	30532	46252	23220	23384
20168	29732	20892	52868	10184	23520	51812	39964	86876	49180

$\bar{t} = 38606.24\ \ t_{max} = 102068\ \ \sigma_t^2 = 497148259.7\ \ time = 1469s$

可以见到使用模拟退火进行一定概率的随机选择对方差有显著提升，对最值有较小幅度的提升。

5 附件

文件夹中包含以下附件。

文件名	内容	制作
main.py	主程序	赵胤
game.py	2048 游戏逻辑实现	赵胤
test.py	状态评分函数	赵胤
ai.py	算法代码	赵胤
gui.py	可视化代码	赵胤
gif/…	可视化用图像文件	赵胤
m.json	可视化用临时文件	赵胤
2048.pptx	课堂展示 ppt	赵胤，黄亮彬
基于极小化极大的 2048 游戏算法设计.pdf	算法报告	赵胤

（完）