

基于手牌拆分的“斗地主”蒙特卡洛树搜索

彭啟文,王以松,于小民,刘满义,徐方婧

(贵州大学计算机科学与技术学院,贵州 贵阳 550025)

[摘要] “斗地主”是典型的多人合作非完全信息博弈,蒙特卡洛树搜索是求解博弈(围棋、国际象棋等)问题的重要工具. 本文首先提出基于“斗地主”规则的手牌拆分算法,通过选择较小拆分以解决其动作空间较大问题;其次,通过蒙特卡洛抽样法,对“斗地主”非完全合作博弈进行不断抽样模拟,在满足一定预设条件后,选择收益最佳的节点作为本次最佳决策. 实验结果表明,基于手牌拆分的“斗地主”蒙特卡洛树搜索能较好地实现“斗地主”自动博弈.

[关键词] 斗地主,计算机博弈,强化学习,蒙特卡洛树搜索

[中图分类号] TP311 [文献标志码] A [文章编号] 1001-4616(2019)03-0107-08

Monte Carlo Tree Search for “Dou Di Zhu” Based on Splitting

Peng Qiwen, Wang Yisong, Yu Xiaoming, Liu Manyi, Xu Fangjing

(School of Computer Science and Technology, Guizhou University, Guiyang 550025, China)

Abstract: “Dou Di Zhu” is a typical multiplayer cooperative game with incomplete information. Monte Carlo tree search is an important tool to solve game problems (Go, chess, etc.). Firstly, this paper proposes a hand splitting algorithm based on the rules of “Dou Di Zhu”, which solves the problem of large action space by choosing smaller splitting. Secondly, this paper adopts Monte Carlo sampling method to simulate the incomplete cooperative game of “Dou Di Zhu”. After satisfying certain preset conditions, this paper chooses the node with the best income as the best decision. The experimental results show that the Monte Carlo tree search based on hand splitting can realize the automatic game of “Dou Di Zhu” in a smart way.

Key words: Dou Di Zhu, computer game, reinforcement learning, Monte Carlo tree search

博弈论(game theory)是研究当决策主体的行为发生直接相互作用的时候,决策主体间的决策以及这些决策的均衡问题^[1]. 根据博弈过程中博弈信息对于其他决策主体是否完全可知,可将博弈分为完全信息博弈和非完全信息博弈. 博弈研究的很多问题和社会中的商业、军事以及政治等方面的情况十分类似,因此博弈的研究对于建立生活中相应的决策支持系统具有很大的作用^[2]. 在计算机问世以后,让计算机像人类进行思考、判断和推理,能够做出理性的决策就成为了很多学者的研究工作,并最终形成机器博弈. 其后研究者在对机器博弈的研究中取得了很多成果. 其中计算机之父冯·诺伊曼就针对博弈问题提出了极大极小定理^[3],信息论的研究者香农首次提出国际象棋的解决方案^[4]. 20 世纪末期,相关学者分别提出了极大极小算法^[5]、 α - β 剪枝^[6]、上限置信区间^[7]、MCTS^[8]等算法.

近年来,随着机器学习的发展,该方法也在完全信息博弈方面取得了显著的成果. 其中具有里程碑意义的是:2016 年 3 月 15 日,google 公司使用深度学习和强化学习等方法,开发的 AlphaGo^[9] 智能 Agent 在围棋领域打败了世界围棋冠军李世石,其标志机器在围棋领域已经具有了智能^[10],但是这种方法严重依赖训练样本的质量及数量. 虽然该公司发布的 AlphaZero^[11]、AlphaGo Zero^[12] 智能 Agent 能在没有训练样本的情况下,也取得傲人的成果,但是这种方法需要大量的硬件资源作为支撑. 在不完全信息博弈方面,2017 年 1 月,卡内基梅隆大学开发的 Libratus 智能 Agent 在与人类顶级选手进行的德州扑克比赛中,取得

收稿日期:2019-07-05.

基金项目:国家自然科学基金联合基金重点项目(U1836205).

通讯联系人:王以松,博士,教授,博士生导师,研究方向:知识表示与推理、人工智能、机器学习. E-mail:yswang@gzu.edu.cn

了最终的胜利^[13],但是该方法使用了大量博弈论中的思想,所以导致针对特定游戏的算法难以设计,此外该方法目前只适用于两人博弈,针对多人博弈暂无解决方案.

针对上述情况,本文选取“斗地主”游戏作为研究载体,针对多人博弈展开研究. 该游戏作为在中国比较流行的纸牌游戏,深受大众的喜欢. 在 2018 年腾讯公司年度“斗地主”锦标赛中,参与人数多达 8 000 万. 与之相反的是,现在对于“斗地主”的研究较少,主要因为其较难且对其重视度不够,其中研究难主要体现在两方面:(1)在游戏过程中扑克信息是部分隐藏的^[14]; (2)该游戏是多人博弈问题^[15]. 上述方面都会导致决策主体行为空间过大. 此外玩家间的合作以及敌对关系无疑又增加了游戏难度. 针对行为空间大和玩家间关系导致的问题,本文使用蒙特卡洛树搜索方法,对行为空间进行抽样模拟,以期通过不断模拟后,能求解出最优解. 在进行蒙特卡洛树搜索之前,为了进一步减少对不必要的节点进行抽样模拟,从而将更多的资源用于可能含有最优解的节点中,本文提出了较小拆分算法.

1 技术介绍

对于普通的博弈问题,一般的方法就是使用博弈树搜索,其主要思想是:博弈玩家通过对手玩家以往的动作,预测对手玩家接下来可能采取的动作,通过对玩家可能采取动作后的博弈局面走势进行分析,从而回溯出当前博弈玩家应当采取的最优策略. 但是博弈树搜索有很大的局限性,即博弈树搜索中,树深度不宜过大,因为随着博弈树深度的增加,博弈树需要处理的节点数呈指数速度增加,这就会导致处理所有节点的时间开销也呈指数性增加.

针对规模大的问题,研究者们将蒙特卡洛方法和博弈树搜索相结合提出了蒙特卡洛树搜索(Monte Carlo tree search, MCTS)方法,该方法在保证降低问题规模的同时,能保持所求解的近似最优性^[2]. 其中,蒙特卡洛方法是利用经验平均来代替随机变量的期望. 比如在游戏中状态 s 时,期望值为 $v_{\pi}(s)$,难以通过计算求出该值,但是可以通过蒙特卡洛方法获取一系列收益 $G_1(s), G_2(s), \dots, G_n(s)$. 根据大数定律,当 n 趋于无穷大时,抽样收益的均值趋近于期望值. 定义 $v(s)$ 为系列收益的平均值,即:

$$v(s) = \frac{G_1(s) + G_2(s) + \dots + G_n(s)}{n} \tag{1}$$

$$v(s) \rightarrow v_{\pi}(s) \text{ as } n \rightarrow \infty. \tag{2}$$

MCTS 算法以初始化搜索树开始,一般抽象当前的游戏状态作为一个单独的博弈树根节点,一旦博弈树初始化完成后,就会在规定的时间内重复如图 1 所示的搜索过程^[16],可以分为 4 个步骤:

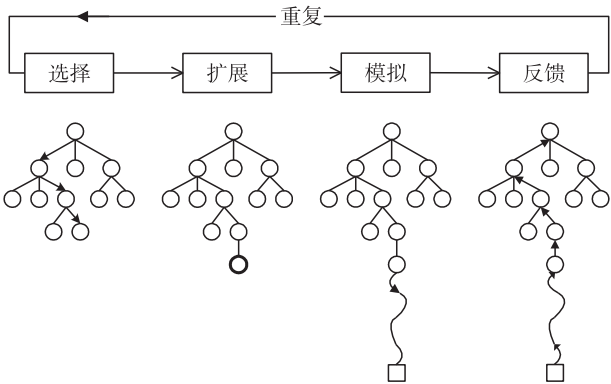


图 1 MCTS 博弈树搜索算法
Fig. 1 MCTS game tree search algorithm

(1)扩展节点的选择:递归地应用节点选择函数,从所有待选择的节点中,选择一个节点作为本次扩展的根节点,从该节点开始,对该节点表示的博弈局面进行一次模拟.

(2)扩展步骤:将一个或多个节点增加到 MCTS 搜索树中. 普通的策略是每次迭代,只向博弈树中增加一个新节点.

(3)模拟:模拟实际玩家博弈过程,进行从博弈树的叶子节点到终止状态的一次博弈过程.

(4)反馈:模拟的结果会从博弈树的叶子节点开始,通过逐层反馈给父节点的方式,最终将模拟结果

返回给根节点.

MCTS 算法会在满足最大抽样次数或者达到时间耗尽等设置后,根据第一层节点中每个节点的估值,从中选择一个决策作为本次 MCTS 算法的最佳决策. 针对选择决策的标准, Schadd 在其他研究者的研究基础上,总结出了 4 种标准^[17],分别为:

(1) 待选节点中,收益最佳的节点.

(2) 最可信节点,即在整个搜索过程中,被选中进行搜索次数最多的节点.

(3) 收益最佳且最可信节点,即同时拥有标准 1 和 2 的节点. 如果在整个搜索过程结束后,还不存在满足此条件的节点,则继续抽样模拟,直到满足条件为止^[18].

(4) 最安全的节点,即在接近搜索结束的搜索过程中,收益波动最小的节点.

在扩展节点选择的时候,所使用的算法存在很大差异. 其中,现在广泛使用的是 2006 年, Kocsis 和 Szepesvari 在 UCB 算法基础上,针对 MCTS 算法中扩展节点选择问题,提出的 UCT 算法^[19]. 实践证明该算法非常高效.

2 算法设计

在介绍算法前,为了后续表示方便,先介绍斗地主游戏规则以及一些记法.

2.1 “斗地主”游戏出牌规则

令 $Pokers = \{0, \dots, 53\}$ 表示一副扑克. $eval(p)$ 表示一张扑克 $p \in Pokers$ 的值(忽略花色),如表 1 所示:

表 1 Poker 集合、eval 函数以及其值的映射关系

Table 1 Poker collection, eval function, and mapping of its value

a	0~3	4~7	8~11	12~15	16~19	20~23	24~27	28~31
$eval(a)$	3	4	5	6	7	8	9	10
真实扑克值	three	four	five	six	seven	eight	nine	ten
a	32~35	36~39	40~43	44~47	48~51	52	53	
$eval(a)$	11	12	13	14	15	16	17	
真实扑克值	Jack	Queen	King	Ace	two	black joker	red joker	

令 $X \subseteq Pokers$, 记 $eval(X) = \{eval(x) | x \in X\}$, $Max(X) = \max(eval(X))$, $Min(X) = \min(eval(X))$.

依据斗地主出牌规则,一手合理的出牌类型包括: pass(过), single(单牌), pair(一对), three(三张), three_plus_one(三带一张), three_plus_two(三带一对), sequence_one(单顺), sequence_two(双顺), sequence_three(三顺), sequence_three_one(飞机带翅膀,三顺+同数量的单牌), sequence_three_two(飞机带翅膀,三顺+同数量的对牌), bomb(炸弹), k_bomb(火箭,大小王). 令 T 是上述合理出牌类型及 undefined 的集合.

令 $X \subseteq Pokers$, 函数 $Type: Power(Pokers) \rightarrow T$ 定义如下:

- $Type(X) = pass$ 若 $X = \emptyset$; $Type(X) = single$ 若 $|X| = 1$;
- $Type(X) = pair$ 若 $|X| = 2$ 且 $|eval(X)| = 1$; $Type(X) = three$ 若 $|X| = 3$ 且 $|eval(X)| = 1$;
- $Type(X) = three_plus_one$ 若 $|X| = 4$ 且 $|eval(X)| = 2$, ($|\{x \in X | eval(x) = Max(X)\}| = 1$ 或 $|\{x \in X | eval(x) = Min(X)\}| = 1$);
- $Type(X) = three_plus_two$ 若 $|X| = 5$ 且 $|eval(X)| = 2$, ($|\{x \in X | eval(x) = Max(X)\}| = 2$ 或 $|\{x \in X | eval(x) = Min(X)\}| = 2$);
- $Type(X) = sequence_one$ 若 $|X| \geq 5$, $Min(X) \geq 3$, $Max(X) \leq 14$ 且 $Max(X) - Min(X) = |X| - 1 = |eval(X)|$;
- $Type(X) = sequence_two$ 若 $|X| \geq 3$, $Min(X) \geq 3$, $Max(X) \leq 14$, $|X| \% 2 = 0$, $Max(X) - Min(X) = |X| / 2 - 1$, 且 $\forall x \in X, \exists ! y \in X - \{x\}$ 使得 $eval(x) = eval(y)$;
- $Type(X) = sequence_three$ 若 $|X| \geq 3$, $Min(X) \geq 3$, $Max(X) \leq 14$, $|X| \% 3 = 0$, $Max(X) - Min(X) = |X| / 3 - 1$, 且 $\forall x \in X, \exists ! y \in X \exists ! z \in X$ 使得 $x \neq y, x \neq z, y \neq z, eval(x) = eval(y) = eval(z)$;
- $Type(X) = sequence_three_one$ 若 $\exists Y \subseteq X$ 使得 $Type(Y) = sequence_three$, 且 $|X - Y| = |eval(Y)|$;
- $Type(X) = sequence_three_two$ 若 $\exists Y \subseteq X$ 使得 $Type(Y) = sequence_three$, $|eval(X - Y)| = |eval(Y)|$, 且 $\forall x \in X - Y, \exists ! y \in X - Y - \{x\}$ 使得 $eval(x) = eval(y)$;

- $\text{Type}(X) = \text{bomb}$ 若 $|X| = 4$ 且 $|\text{eval}(X)| = 1$;
- $\text{Type}(X) = \text{k_bomb}$ 若 $\text{eval}(X) = \{16, 17\}$;
- $\text{Type}(X) = \text{undefined}$ 否则.

记: $\Gamma(X) = \{YX | \text{Type}(Y) \neq \text{undefined}\}$. 一个集合 $S = \{S_1, \dots, S_n\}$ 是 Pokers 空子集 X 的一个拆分, 如果对于任何 $1 \leq i \leq n, S_i \in \Gamma(X), S_i \neq \emptyset$ 且 $X = \bigcup_{i=1}^n S_i$.

2.2 算法介绍

本文模型主要包括两个模块:较小拆分模块和蒙特卡洛树搜索模块. 在对一般博弈问题进行求解时,通常使用蒙特卡洛树搜索方法来进行解决. 但是该方法中,需要对当前状态进行大量的模拟后,最终求解出博弈问题的解;针对“斗地主”博弈问题而言,由于开始时根据手牌及游戏规则等求出的所有可能解种类较多,故而导致博弈树根节点的分支较多;此时如果对该博弈树进行蒙特卡洛抽样,会导致求解时间较长. 针对这种情况结合“斗地主”游戏及人类玩家,针对“斗地主”游戏中的规律进行分析,本文提出使用较小拆分算法对博弈树根节点进行剪枝,其后再根据剪枝后的博弈树进行蒙特卡洛抽样,进而提高求解效率. 实验过程如下:当游戏开始的时候,先使用较小拆分模块求解出待抽样的出牌集,然后通过蒙特卡洛树搜索模块,对待模拟抽样的所有出牌进行游戏模拟,通过不断的模拟,根据其模拟后的收益,选择平均收益最大的出牌作为本次的最佳出牌. 具体流程如图 2 所示.

2.2.1 基于“斗地主”规则实现手牌的较小拆分算法

“斗地主”游戏一般由 3 个玩家进行,其中一方为地主,其余两玩家组成另外一方. 在进行“斗地主”的过程中,三游戏玩家根据游戏规则交替出牌,先出完牌的一方获胜. 由于在游戏的进程中,由地主开始先出牌,其后 3 个玩家交替进行出牌,所以在一轮游戏中,玩家可出牌次数是相同的. 因此在相同手牌的情况下,将手牌按照“斗地主”游戏出牌规则进行拆分. 如果拆分后,出牌组合数比较少,也就表明该玩家如果按照该拆分结果进行游戏,那么该玩家在相同的出牌次数中,就能最先出完手牌,取得胜利. 因此,作者希望通过对玩家手牌进行拆分,拆分后选取组合数最小的拆分进行博弈. 但是仅仅只使用最小拆分的组合进行博弈时,当组合中的扑克在每次出牌后,其他玩家都不能大过时,这种最小拆分能达到很好的效果. 但是如果在出牌后丢失牌权,且其他玩家出牌后,该玩家不能再获得牌权,这种情况下,博弈结果不容乐观. 所以,不能仅仅根据最小拆分算法进行博弈,还需要将较小拆分的结果纳入待出牌集中.

通过对竞技世界(北京)网络技术有限公司,组织的高级比赛中人类玩家历史数据的分析,得出表 2 结果.

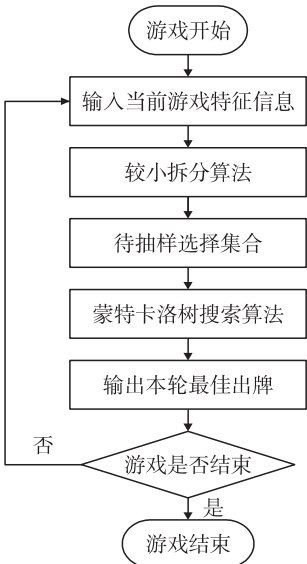


图 2 基于 MCTS 的“斗地主”流程图
Fig. 2 The MCTS based algorithm of “Dou Di Zhu”

表 2 出牌包含于对应长度的拆分结果中的比例

Table 2 Proportion of the card included in the split result of the corresponding length					
长度	L_{\max}	L_{\min}	$L_{\min} + 1$	$L_{\min} + 2$	$L_{\min} + 3$
次数	21 130	18 289	20 204	20 907	20 989
比例/%	100	85.55	95.62	98.94	99.33

表中 L_{\min} 、 L_{\max} 分别表示以当前手牌进行拆分后,拆分组合数的最小值和最大值. 从表 2 可知:当以当前手牌进行拆分后,如果只选择实际拆分数 $= L_{\min}$ 组合组成待出牌集合时,人类实际出牌包含于待出牌集合的可能性为 88.55%;随着实际拆分数的增加,当选择实际拆分数 $\leq L_{\min} + 3$ 时,人类实际出牌包含于该待出牌集合的可能性为 99.33%. 结果表明,选取所有拆分数 $\leq L_{\min} + 3$ 的较小拆分算法结果,基本能求出所有待出牌. 因此,在较小拆分算法中,就选取拆分数 $\leq L_{\min} + 3$ 的拆分牌组成带抽样集.

算法事例:

比如现在玩家手牌为:34556789LB,

拆分算法求出所有拆分结果为:

$$S_1 = \{3, 4, 5, 5, 6, 7, 8, 9, L, B\}$$

$$S_2 = \{3, 4, 5, 5, 6, 7, 8, 9, LB\}$$

$$S_3 = \{3, 4, 55, 6, 7, 8, 9, L, B\}$$

$$S_4 = \{3, 4, 55, 6, 7, 8, 9, 2B\}$$

$$S_5 = \{3, 4, 5, LB, 56789\}$$

$$S_6 = \{3, 4, 5, L, B, 56789\}$$

$$S_7 = \{3, 5, 9, L, B, 456789\}$$

$$S_8 = \{3, 5, 9, LB, 456789\}$$

$$S_9 = \{3, 5, L, B, 456789\}$$

$$S_{10} = \{3, 5, LB, 456789\}$$

$$S_{11} = \{5, 8, 9, L, B, 34567\}$$

$$S_{12} = \{5, 8, 9, LB, 34567\}$$

$$S_{13} = \{5, 9, L, B, 345678\}$$

$$S_{14} = \{5, 9, LB, 345678\}$$

$$S_{15} = \{5, L, B, 3456789\}$$

$$S_{16} = \{5, LB, 3456789\}$$

$L_{\min} = 3$, 所以待出牌集合 $S = \{5, LB, 3456789, L, B, 9, 345678, 8, 34567, 456789, 3, 45678, 56789, 4\}$.

表 3 手牌拆分算法

Table 3 The hand cards splitting algorithm

算法 1: 手牌拆分算法 Split(F, X)
 输入: 当前拆分子集 F 和手牌 X
 输出: 手牌 X 的所有拆分子集 S
 if $X = \emptyset$ then return F
 else
 令 $F(X) = \{Y_1, Y_2, \dots, Y_n\} - \{\}$
 Return(Split($F+Y_1, X-Y_1$) $\cup \dots \cup$ Split($F+Y_n, X-Y_n$)),
 其中 $F+Y_i = \{z \cup Y_i \mid z \in F\}$

表 4 手牌较小拆分算法

Table 4 The smaller hand cards splitting algorithm

算法 2: 手牌较小拆分算法
 输入: 当前玩家手牌 X , 拆分上界 k
 输出: 当前玩家的较小拆分抽样集 MS
 $F = \{\}$, $MS = \emptyset$;
 $S = \text{Split}(F, X)$
 $L_{\min} = \text{Min}(\{ \text{len}(z) \mid z \in S \})$
 For each s in S :
 If $\text{len}(s) \leq (L_{\min} + k)$ then $MS = MS \cup s$

2.2.2 蒙特卡洛树搜索算法

很显然,在“斗地主”游戏中,其他玩家手牌信息对于当前玩家而言是隐藏的,这无疑会大大增加了博弈树中的节点数,导致博弈树空间进一步增大,增加解决“斗地主”游戏的难度. 所以普通的博弈树搜索算法,对于“斗地主”这类不完全信息博弈游戏不能提供很好的解决方法.

针对问题规模大的问题,研究者们提出了蒙特卡洛树搜索(Monte Carlo tree search, MCTS). 具体于“斗地主”游戏中,玩家根据自己手牌、两个玩家已出扑克以及两个玩家手牌张数等信息,通过对游戏未完成的博弈过程进行不断的模拟,以确定在当前情况下,玩家最佳的决策(出牌). 具体决策流程如下:

步骤 1 首先,根据当前玩家的手牌,以及所有玩家已出扑克等信息,基于“斗地主”的较小拆分算法,求解出待出牌集合;再将求解结果中的每个元素增加到博弈树中作为叶子节点.

步骤 2 选择过程:在博弈树中所有的叶节点中,选取一个叶节点作为本次模拟的根节点. 在选取的时候,由于需要权衡探索和利用的关系,所以在本文中使用了 UCT 算法. 将博弈树中的叶节点通过 UCT 算法计算该节点的选择评估值 γ , 选择评估值 γ 最大的叶节点,作为新一次模拟的根节点进行模拟.

UCT 算法为:

$$\gamma_i = \bar{V}_i + C \sqrt{\frac{2 \ln \sum_j^n n_j}{n_i}}. \quad (3)$$

式中, γ_i 表示节点 i 的选择评估值, \bar{V}_i 表示节点 i 的平均收益; C 是常数,其作用是为了平衡探索和利用;

n_i 是以第 i 个节点作为模拟搜索的根节点的次数.

步骤 3 扩展过程:在“斗地主”的过程中,由于“斗地主”游戏属于不完全信息博弈的一种,在对其叶节点进行扩展的时候,选中的叶节点的子节点数量较大.因此,在本游戏中,扩展阶段只将本次抽样模拟的根节点加入博弈树中.

步骤 4 模拟过程:从抽样的博弈环境信息开始,3 个玩家根据手牌在满足游戏规则的前提下,随机进行交替出牌,直到到达博弈终止状态.

步骤 5 反馈过程:游戏到达终止状态后,将收益值返回到根节点,并沿途更新节点的统计估计值.在本文中,定义收益为:

当地主取得胜利时,地主收益为+2;两个农民的收益均为-1.

当地主输时,地主收益为-2;两个农民的收益均为+1.

实验过程中,在进行蒙特卡洛抽样过程中涉及到随机性问题,但其结果收敛性与实验环境无关,因为根据大数定律可知:当抽样次数足够多时,抽样收益的均值趋近于期望值.在“斗地主”游戏中的决策主体身份分为地主和农民,但博弈过程中,无论智能体身份为地主或农民,进行蒙特卡洛抽样时,对于其身份仅仅在最终反馈收益时加以区分,在抽样的过程中,所有决策主体无任何区别.

3 实验结果与分析

本章节将使用较小拆分算法以及蒙特卡洛树搜索方法实现的“斗地主”智能 Agent 与其他智能 Agent 进行比较.由于做“斗地主”游戏的研究人员较少,相应的研究成果更是少之又少,这无疑增加了选择其他智能 Agent 的难度.通过不断地分析,最终选择了两个智能 Agent:基于固定规则的 Agent 以及 7k7k 小游戏世界斗地主实现的智能 Agent.

3.1 与固定规则方法比较

游戏设定:在进行游戏的过程中,通过程序随机地将 51 张扑克平均地分配给每个玩家(发牌过程,每玩家分配到 17 张手牌),然后再通过随机数指定一个玩家作为地主,并将剩余的 3 张地主牌分配给该地主玩家.从地主玩家开始,按照“斗地主”游戏规则进行博弈.其中三玩家中:一个玩家为本文中实现的智能 Agent,其余两个玩家为固定规则方法的智能 Agent.其中固定规则是指,当玩家为主动出牌时,按照如下优先级 PRI 进行出牌:PRI(飞机带翅膀)>PRI(三带一张)>PRI(三带一对)>PRI(三张)>PRI(单牌)>PRI(双顺)>PRI(三顺)>PRI(对子)>PRI(单牌)>PRI(炸弹)>PRI(王炸),在相同优先级中,选择最小的进行出牌;当玩家为被动出牌时,选择最小能大过的扑克进行出牌,如无,则 Pass.

通过上述游戏设定,本文的 Agent 通过和固定规则实现的 Agent 进行 1582 次游戏博弈,其中赢了 1300 场,输了 282 场,胜率为 82.17%.详细胜率随着博弈次数的变化如图 3 所示(注:下列所有的图中横坐标表示博弈总次数,纵坐标表示本文中智能 Agent 的胜率).

从图中可得:当博弈刚开始的时候,由于分配扑克和指定地主的随机性,导致在第一次博弈中输掉比赛,所以胜率为 0.但是随着博弈次数的增加智能 Agent 胜率大幅度上升,并在博弈次数低于 200 局时,智能 Agent 博弈胜率呈现较大幅度的波动.当博弈局数大于 400 后,可以看出,虽然胜率出现小幅度波动,但是基本维持在 82%左右,其中作为地主、农民时的胜率分别如图 4、5 所示.

从图中可得:当博弈刚开始的时候,由于匹配扑克和指定地主的随机性,导致在第一次博弈中智能 Agent 赢得比赛,所以胜率为 1.但是随着博弈次数的增加智能 Agent 胜率大幅度下降,并在博弈次数低于 150 局时,智能 Agent 博弈胜率呈现较大幅度的波动.当博弈局数大于 200 后,可以明显看出,虽然胜率出现小幅度波动,但是基本维持在 88%左右.

从图中可得:当博弈刚开始的时候,由于匹配扑克和指定地主的随机性,导致在第一次博弈中智能 Agent 输了比赛,所以胜率为 0.但是随着博弈次数的增加智能 Agent 胜率大幅度上升,并在博弈次数低于

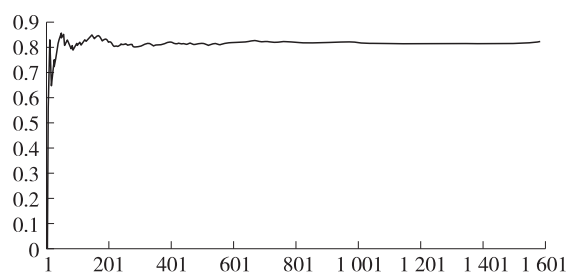


图 3 与基于规则博弈的胜率变化图

Fig. 3 The winning rate against a rule-based agent

150 局时,智能 Agent 博弈胜率呈现较大幅度的波动. 当博弈局数大于 200 后,可以看出,农民的胜率虽然出现小幅度波动,但是基本维持在 79%左右.

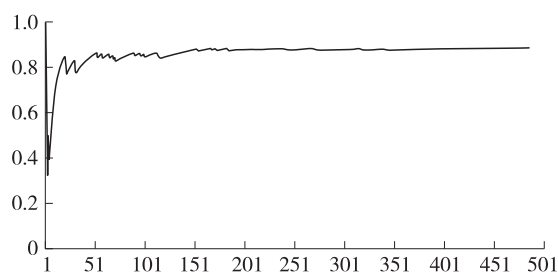


图 4 与基于规则博弈中,作为地主的胜率变化图

Fig. 4 The winning rate as a landlord against a rule-based agent

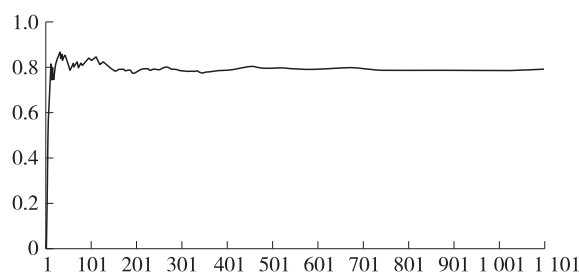


图 5 基于规则博弈中,作为农民的胜率变化图

Fig. 5 The winning rate as a farmer against a rule-based agent

3.2 与 7k7k 小游戏世界斗地主智能 Agent 比较

7K7K 是国内专业的休闲游戏网站之一,它搜集互联网广泛的小游戏(包括斗地主等)资源. 游戏设定:在进行游戏的过程中,由于该公司未开放游戏接口,所以只能通过浏览器的方式获得当前博弈信息. 在游戏开始的时候,随机选择一个玩家开始进行叫地主. 当确定地主后,就以上家出牌、上上家(即下家)出牌以及自己手牌等信息,通过本文中的模型运算后,根据模型输出结果,并人为地将相应的结果以动作的形式进行出牌,如此反复进行,直到分出胜负为止(即结束一局游戏). 其中三玩家中:一个玩家为本文中实现的智能 Agent,其余两个玩家为 7k7k 的智能 Agent.

通过上述游戏设定,本文的智能 Agent 通过和 7k7k 小游戏世界斗地主智能 Agent 进行 86 次游戏博弈,其中赢了 56 场,输了 30 场,胜率为 65.11%. 详细胜率随着博弈次数的变化如图 6 所示.

从图中可得出:当博弈刚开始的时候,由于匹配扑克和指定地主的随机性,导致在第一次博弈的时候,该智能 Agent 输了比赛,所以胜率为 0. 但是随着博弈次数的增加智能 Agent 胜率大幅度上升,并在博弈次数低于 70 局时,智能 Agent 博弈胜率呈现较大幅度的波动. 当博弈局数大于 70 后,分别可以看出,智能 Agent 的胜率虽然出现小幅度波动,但是基本维持在 65%左右.

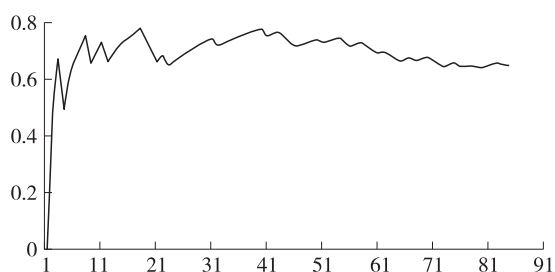


图 6 与 7k7k 小游戏世界“斗地主”智能 Agent 博弈的胜率变化

Fig. 6 The winning rate against the ‘Dou Di Zhu’ agent from 7k7k

3.3 合作问题

在对游戏结果进行分析的过程中,作者发现部分实例可以验证本文中使用的的方法,可以一定程度解决多人博弈中的合作问题,具体实例如下:

当前玩家手牌:334567QQKKA2B

当前玩家位置:1(其中 0 表示地主,1 表示农民一,2 表示农民二)

地主已出牌:339922789JQK666JJL

当前玩家已出牌:55TTB

农民二已出牌:77AA89TJQKA44488

本轮中地主出牌:L

游戏过程:0|33,1|55,2|77;0|99,1|TT,2|AA;0|22,1|pass,2|pass;0|789TJQK,1|pass,2|89TJQKA;0|pass,1|pass,2|44488;0|666JJ,1|pass,2|pass;0|L,1|B,2|pass;1|3,2|2.

通过对所有手牌以及当前玩家手牌的分析,可知:在上述游戏过程中,当地主出 L 后,虽然不知道两个玩家的具体手牌,但是可以推理出这两玩家手牌只能是 2 或者 5. 所以,这时候果断出 B 管住后,再出 3,可帮助另一个农民取得胜利,这一动作体现了本文中的智能 Agent 能解决部分合作博弈问题.

4 结论

本文主要基于蒙特卡洛树搜索方法实现“斗地主”智能 Agent。文中通过蒙特卡洛抽样法,对“斗地主”这类非完全合作游戏进行不断抽样模拟,最终在满足设定条件后,选择收益最佳的节点作为本次最佳决策。此外针对游戏动作空间较大问题,本文提出较小拆分算法,实验证明:该算法在保证有效的情况下,能将游戏动作空间进行缩减,使得有限的资源能用在最佳决策的搜索上。

实验表明,蒙特卡洛树搜索及较小拆分算法能解决“斗地主”游戏中的多人博弈和合作博弈问题。但是,文中的方法在保证准确率的情况下,也存在消耗时间过长问题。此外对于其他玩家手牌的推理方面其能力和人类相比还较弱。这些都是作者后续研究的方向。

[参考文献]

- [1] 张维迎. 博弈论与信息经济学[M]. 上海:上海人民出版社,2004.
- [2] 张加佳. 非完全信息机器博弈中风险及对手模型的研究[D]. 哈尔滨:哈尔滨工业大学,2015.
- [3] VON N J, MORGENSTERN O. Theory of games and economic behavior[M]. Princeton:Princeton University Press,1994.
- [4] SHANNON C E. Programming a computer for playing chess[M]//Computer chess compendium. New York, USA:Springer, 1988:2-13.
- [5] ROIZEN I, PEARL J. A minimax algorithm better than alpha-beta? Yes and No[J]. Artificial intelligence, 1983, 21(1/2): 199-220.
- [6] FULLER S H, GASCHNIG J G, GILLOGLY J J. Analysis of the alpha-beta pruning algorithm[M]. USA: Carnegie-Mellon University, 1973.
- [7] GELLY S, SILVER D. Combining online and offline knowledge in UTC[C]//Proceedings of the 24th International Conference on Machine Learning. New York, USA:ACM, 2007:273-280.
- [8] CHASLOT G, BAKKES E, SZITA I. Monte-Carlo tree search: a new framework for game AI[J]. In Proceedings of AIIDE-08, 2008, 4(2): 216-217.
- [9] SILVER D, HUANG A, MADDISON C J, et al. Mastering the game of Go with deep neural networks and tree search[J]. Nature, 2016, 529(7587): 484-489.
- [10] 刘洋. 点格棋博弈中UCT算法的研究与实现[D]. 安徽:安徽大学, 2016.
- [11] SILVER D, HUBERT T, SCHRITTWIESER J, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play[J]. Science, 2018, 362(6419): 1140-1144.
- [12] SILVER D, SCHRITTWIESER J, SIMONYAN K, et al. Mastering the game of Go without human knowledge[J]. Nature, 2017, 550(7676): 354-359.
- [13] BROWN N, SANDHOLM T. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals[J]. Science, 2018, 359(6374): 418-424.
- [14] DARSE B, AARON D, JONATHAN S, et al. The challenge of poker[J]. Artificial intelligence, 2002, 134: 201-240.
- [15] STURTEVANT N. Current challenges in multi-player game search[C]//International Conference on Computers and Games. Berlin, Heidelberg: Springer, 2004: 285-300.
- [16] GELLY S, SILVER D. Monte-Carlo tree search and rapid action value estimation in computer Go[J]. Artificial intelligence, 2011, 175(11): 1856-1875.
- [17] SCHADD F C. Monte-Carlo search techniques in the modern board game thurn and taxis[D]. Netherlands: Maastricht University, 2009.
- [18] COULOM R. Efficient selectivity and backup operators in Monte-Carlo tree search[C]//5th Int Conf Comput and Games. Turin, Italy: Natural Comput, 2006: 72-83.
- [19] GELLY S, WANG Y. Exploration exploitation in Go: UCT for Monte-Carlo go[C]//NIPS: Neural Information Processing Systems Conference On-line trading of Exploration and Exploitation Workshop. Canada: fhal-00115330f, 2006.

[责任编辑:黄敏]