CN 43-1258/TP
ISSN 1007-130X

计算机工程与科学
Computer Engineering & Science

第 44 卷第 12 期 2022 年 12 月
Vol. 44, No. 12, Dec. 2022

文章编号:1007-130X(2022)12-2128-06

面向多智能体博弈的并行蒙特卡洛树搜索算法研究*

管延霞[1],刘逊韵[2],刘运韬[1],谢　旻[1],徐新海[2]
(1.国防科技大学计算机学院,湖南 长沙 410073;2.军事科学院战争研究院,北京 100091)

摘　要:蒙特卡洛树搜索算法是一种常用的强化学习算法,博弈过程中动态空间的指数级增长是制约该算法学习效率的因素。基于并行方法对蒙特卡洛树搜索算法进行优化,提出基于胜率估值传递的并行蒙特卡洛树搜索算法。改进后的并行博弈搜索策略框架包含一个主进程和多个子进程,其中子进程用于探索,主进程根据子进程传递的胜率估值数据进行决策。结合多智能体博弈平台 Pommerman 进行实验验证,与传统的蒙特卡罗树搜索算法相比,并行蒙特卡罗树搜索算法有效提高了资源利用率、博弈胜率及决策效率。

关键词:多智能体博弈;Pommerman;多进程;并行蒙特卡洛树搜索
中图分类号:TP391.41　　**文献标志码**:A
doi:10.3969/j.issn.1007-130X.2022.12.005

A parallel Monte Carlo tree search algorithm for multi-agent game

GUAN Yan-xia[1], LIU Xun-yun[2], LIU Yun-tao[1], Xie Min[1], XU Xin-hai[2]
(1. College of Computer Science and Technology, National University of Defense Technology, Changsha 410073;
2. War Research Institute, Academy of Military Sciences, Beijing 100091, China)

Abstract: Monte Carlo tree search algorithm is a commonly used reinforcement learning algorithm, and the exponential growth of the dynamic space of the algorithm in the game process has become a factor that restricts the improvement of the algorithm learning efficiency. Based on the parallel approach to optimize the Monte Carlo tree search algorithm, a parallel Monte Carlo tree search algorithm based on the transfer of winning rate estimate is proposed. The improved parallel game search strategy framework consists of one main process and several sub-processes, in which the sub-processes are used for exploration, and the main process makes decisions according to the winning rate estimate data transmitted by the sub-processes. Combined with the multi-agent game platform Pommerman for experimental validation, the parallel Monte Carlo tree search algorithm can enhance the resource utilization rate, game-winning rate, and decision-making efficiency over the traditional Monte Carlo tree search algorithm.

Key words: multi-agent game; Pommerman; multi-process; parallel Monte Carlo tree search

1　引言

计算机博弈是衡量人工智能发展水平的重要测试平台之一[1],已在象棋、围棋等棋类博弈的决策问题上取得了优异的成果[2,3]。计算机博弈目前的研究重点在于多智能体博弈。

多智能体会导致博弈过程中的动态空间和状态空间呈指数级增长,使得决策的搜索和选择过程需要消耗一定的时间和计算资源[4]。计算机资源的并行协作可以有效降低多智能体博弈中的维度灾难等问题。但是,该方法的主要难点在于如何有

* 收稿日期:2021-04-02;修回日期:2021-09-24
通信作者:徐新海(xuxinhai@nudt.edu.cn)
通信地址:100091 北京市海淀区厢红旗东门外 1 号院
Address: 1 Courtyard, East Gate, Xianghongqi, Haidian District, Beijing 100091, P. R. China

效利用计算资源并行加速搜索过程,以及如何实现信息的有效传递。

本文选取 Pommerman[5] 作为研究平台,针对多智能体博弈问题中的蒙特卡洛树搜索算法展开研究。Pommerman 是一种通过放置炸弹来淘汰敌人的多智能体博弈环境。博弈过程中的搜索算法优化是博弈性能提升的关键,针对原有蒙特卡洛树搜索算法搜索耗时过长的问题,本文借鉴蒙特卡洛树搜索常用的并行方法,对博弈中的搜索阶段进行并行优化,旨在充分利用有限的计算机资源,缩短搜索算法收敛到近似最优决策策略的学习时间。本文具体工作如下所示:

(1)建立了适合并行搜索策略的 Pommerman 博弈算法——基于胜率估值传递的并行蒙特卡洛树搜索算法,实现了搜索策略框架并行化,构建了多个子进程的并行机制,有效缩短了决策时间,提高了决策效率。

(2)将构建的并行算法应用于 Pommerman 的不同游戏模式,并分别与对应的原始搜索算法进行了性能对比,定量分析了本文算法的优势。

2 蒙特卡洛树搜索算法并行优化现状

Pommerman 博弈平台采用蒙特卡洛树搜索 MCTS(Monte Carlo Tree Search)算法[6-9]做出博弈策略,该搜索算法将蒙特卡洛算法应用于博弈树搜索过程中。围棋中的 AlphaGo[2,3] 即是以 MCTS 算法为基础进行研发的。

MCTS 算法是在搜索空间巨大的情况下仍然有效的算法。与 MCTS 算法出众的表现相对应,庞大的搜索空间使得 MCTS 算法在决策时耗费了更长的时间[10]。因此,研究蒙特卡洛树搜索算法的并行优化十分必要[11-13]。文献[14]表明,蒙特卡洛树搜索算法可以使用如图 1 所示的叶并行化(Leaf Parallelization)、树并行化(Tree Parallelization)和根并行化(Root Parallelization)3 种不同的方法进行并行化。3 种并行方法的优劣不同,叶并行化实现简单但缺乏节点探索性;树并行化可以最小化通信开销但会导致节点的过度开发;根并行化虽然减少了进程通信但同时也减少了每个进程的博弈模拟次数,也在一定程度上降低了决策的准确性。

在对 MCTS 进行并行优化设计时,不仅要考虑如何跟踪得到正确的统计数据,还要尽量减少进程之间的通信开销。

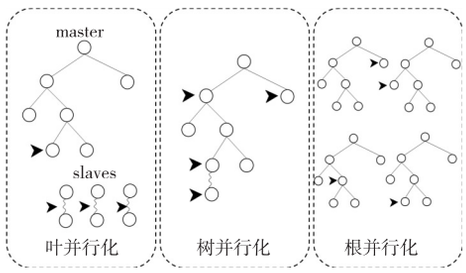

Figure 1 Mainstream parallelization methods of Monte Carlo tree search algorithm

图 1 蒙特卡洛树搜索算法的主流并行化方法

MCTS 算法依赖当前游戏状态之前的采样信息[15]来保持未知领域的探索和已有经验的利用之间的平衡。当使用多个工作程序并行化蒙特卡洛树搜索算法部署时,要确保每个工作程序都能获得最新的统计信息。在并行化过程中,本文想要尽可能实现如图 2a 所示的理想并行化,减少如图 2b 所示真实并行化可能会出现的进程冲突等问题。

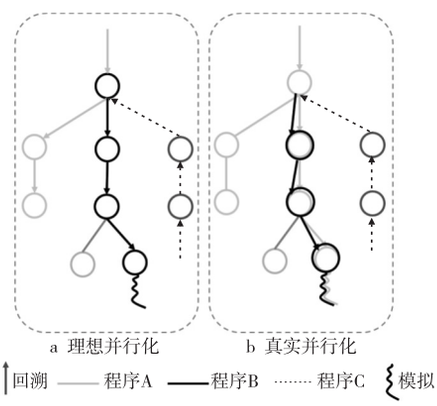

Figure 2 Parallelization of Monte Carlo tree search algorithm

图 2 蒙特卡洛树搜索算法并行化

3 基于胜率估值传递的并行蒙特卡洛树搜索算法

3.1 算法设计

基于多个进程根节点回报值汇总的思想,本文提出了一种基于胜率估值传递的并行蒙特卡洛树搜索算法。算法借鉴 Root Parallelization[7] 的思路,如图 3 所示,在进行 MCTS 搜索之前主进程将当前的游戏状态及游戏策略传递给各子进程,并将探索任务平均分配到各子进程中,使多个子进程基于同一个根节点状态独立并行执行搜索过程。每个子进程完成探索任务后,将回报信息传递给主进程,主进程依据最终的汇总信息进行决策并更新搜索策略。主进程将更新后的信息再次传递给子进

程,循环反复直至游戏状态满足结束条件。

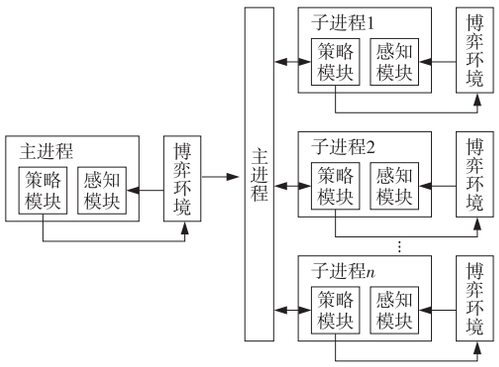

Figure 3 Parallelization of search process

图 3 搜索过程并行化

只在进程的开始和结束时进行算法进程之间的信息交互,以最小化子进程之间的信息交互。相比于采用树并行化实现的 Ary[16] 来说减少了进程交互;相比于叶并行化的实现,避免了节点频繁加解锁;相较于根并行化的实现,本文设计的算法各子进程的搜索耗时相当,减少了子进程信息收集的等待时差。本文设计的算法能够有效利用计算资源,缩短搜索时间,提高 MCTS 的搜索效率。

图 4 给出了实现并行化的搜索过程,主要分为 4 步:

(1) 游戏开局,参数初始化,主进程给子进程分配任务,并行执行子进程;

(2) 各子进程根据 MCTS 算法完成规定次数的迭代,产生根节点下所有合法行动的胜率估值,得到该子进程下的胜率估值向量,并将其发送给主进程;

(3) 主进程汇总各子进程的胜率估值向量,选择获胜概率最大的行动,并以此为依据,做出决策行动;

(4) 待对方智能体决策执行后,再重复进行步骤(2)操作,直到博弈终止。

本文提出的蒙特卡洛树并行搜索算法的典型特征是当各子进程得到各自根状态下的所有行动的胜率估值,并将该子进程的胜率估值向量传递给主进程后,主进程采用式(1)汇总所有子进程的胜率估值向量。

$$\boldsymbol{X}=\sum_{i=1}^{n} w_i \boldsymbol{x}_i \tag{1}$$

其中,$\boldsymbol{X}$ 表示汇总的胜率估值向量;n 表示划分的子进程数量;w_i 表示第 i 个子进程所占比重,在本文中每个子进程分配的迭代次数相同,所以假设每个子进程对应的贡献度相同,即 $w_i=1/n$;$\boldsymbol{x}_i$ 表示第 i 个子进程计算所得的胜率估值向量。智能体依据汇总的胜率估值向量 $\boldsymbol{X}$ 选取对当前局面最有利的决策行动。

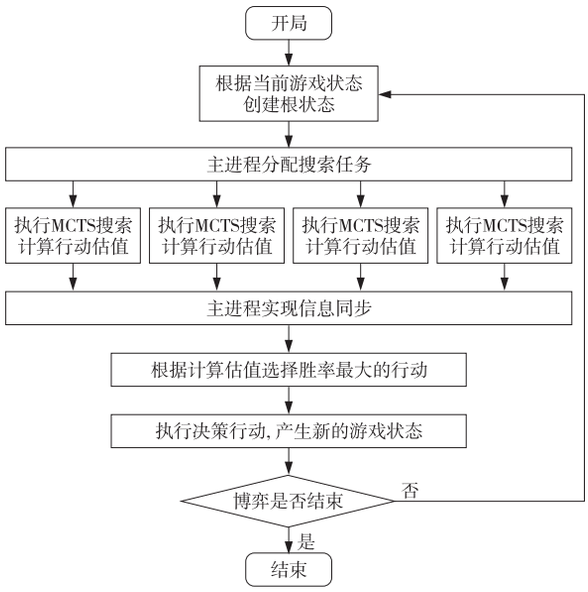

Figure 4 Flow chart of multi-process game search

图 4 多进程博弈搜索流程图

3.2 并行蒙特卡洛树搜索算法的实现

本文提出的蒙特卡洛树并行搜索算法流程如算法 1 所示。

算法 1 并行 MCTS 算法

输入:当前游戏状态(非游戏终态)。

输出:基于当前游戏状态做出的决策行动。

步骤 1 构建多个子进程进行搜索:

每个进程均以当前游戏状态作为根节点,独立并行执行步骤 2~步骤 7。

步骤 2 判断子进程博弈模拟局数是否达到要求:

if 满足条件

执行步骤 8,输出决策行动;

else

执行步骤 3;

步骤 3 判断当前游戏状态是否符合终止状态:

if 符合

执行步骤 7;

else

执行步骤 4;

步骤 4 选择节点:

if 当前状态在博弈树中不存在

执行步骤 5,拓展当前游戏状态;

else

执行步骤 6,选择最佳的子节点;

记录节点状态、action 和环境 reward;

步骤 5 拓展节点:

将当前状态添加到博弈树中,同时初始化当前状态下的所有动作节点所对应的游戏状

态,执行步骤 4。

步骤 6 状态更新:

按照式(2)计算每个节点的 *value* 值,选择值最大的节点的状态并将其更新为当前游戏状态:

$$value=\arg\max_{j\in[1,m]}\frac{y_j}{c_j}+\frac{\sqrt{\sum_{k=1}^{m}c_k}}{1+c_j} \tag{2}$$

假设智能体进行决策时共有 m 种行动可供选择,即当前节点有 m 个子节点,第 j 个节点具有参数 y_j,c_j($j\in[1,m]$),c_j 表示到该次模拟为止第 j 个节点被选中的次数,y_j 表示第 j 个节点在 c_j 次的模拟中模拟获胜的次数。y_j/c_j 表示第 j 个节点的胜率。

步骤 7 奖励回溯:

根据步骤 3 记录的节点、action 和环境 reward,反向更新沿途节点 reward 和访问次数。完善博弈树构建;

游戏状态回到当前状态,执行步骤 2。

步骤 8 胜率估值向量合并:

根据式(1)合并各子进程根节点下的胜率估值向量,根据式(2)计算当前状态根节点下适合的决策行动。

步骤 9 输出基于当前游戏状态作出的决策行动。

将待决策的当前游戏局面作为根节点,子进程并行进行蒙特卡洛树搜索,得到子进程胜率估值,由主进程进行合并得到最终的胜率估值向量,算法根据胜率估值向量,选取决策行动。

4 模型测试与数据分析

4.1 非零和多智能体博弈平台-Pommerman 简介

Pommerman 游戏界面如图 5 所示,每局游戏有 4 个智能体,每个智能体以网格 4 角为起始位置。每个智能体的目标就是存活到最后,通过炸弹的放置以及躲避炸弹等博弈策略来淘汰敌人,保全自己。在 Pommerman 竞赛中,智能体只能做出如表 1 所示的行动之一,不能做出其他多余动作。

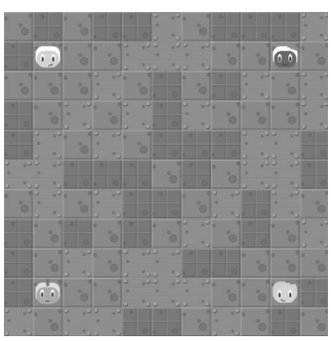

Figure 5 Game interface of Pommerman

图 5 Pommerman 游戏界面

Table 1 Legal actions of agent in Pommerman

表 1 Pommerman 中智能体合法动作

动作指令	动作含义
Stop	智能体静止不动
Up(↑)	智能体向上移动
Down(↓)	智能体向下移动
Left(←)	智能体向左移动
Right(→)	智能体向右移动
Bomb	智能体在当前位置放置炸弹

另外,竞赛中还存在不同的竞争模式,在团队模式(2V2)中,当一个团队的 2 个智能体都被摧毁时,游戏结束,获胜队伍为幸存智能体所在队伍;在 4 个智能体的单人模式(FFA)和 2 个智能体的单人模式(1V1)中,最多只有 1 个智能体存活时游戏结束,幸存者即为获胜者。

4.2 实验环境

本文的软件与硬件实验环境分别如表 2 和表 3 所示。本文基于表 2 和表 3 的实验环境,进行对比实验的设计与实现,所有的实验都是基于相同的运行时间(24 h)采用不同搜索算法的智能体进行博弈,测试博弈性能并进行数据分析。

Table 2 Experimental environment(software conditions)

表 2 实验环境-软件条件

配置名称	版本
Anaconda	4.2.0(64 bit)
Python	3.6.2
TensorFlow	1.14.0
Keras	2.3.1

Table 3 Experimental environment(hardware conditions)

表 3 实验环境-硬件条件

配置名称	配置型号	配置属性
CPU	Intel(R) Core(TM) i58250U CPU 8 核	1.6 GHz
操作系统	Windows10	64 bit

4.3 测试指标

在对蒙特卡洛树搜索算法进行测试之前,为了方便对比并分析并行版本与串行版本的蒙特卡洛树搜索算法的性能,本文使用以下 2 个测试指标进行性能分析:

(1) 对局耗时:即在相同的 MCTS 迭代次数之下,测试不同并行进程蒙特卡洛树搜索算法的平均每局耗时,即对同一个任务需要消耗的时间的对比。平均对局耗时可以反映不同进程的蒙特卡洛树搜索算法在搜索过程中的耗时情况。

(2) 对弈胜率:对弈胜率用于度量搜索算法的性能,即智能体使用不同版本的博弈算法进行对弈,记录使用本文设计的优化算法的智能体获胜的概率。在本文中不同版本的博弈算法对弈的设计如表 4 所示。将不同版本的博弈算法运用至不同的博弈模型(FFA,2V2 和 1V1),分析博弈效果。

Table 4 Comparative experiment design

表 4 对比实验设计

实验编号	实验对象设计
1(FFA)	1 个串行 MCTS agent vs 3 个基于规则的 agent
2(FFA)	1 个并行 MCTS agent vs 3 个基于规则的 agent
3(1V1)	1 个并行 MCTS agent vs 1 个串行 MCTS agent
4(2V2)	[1 个串行 MCTS agent & 1 个基于规则的 agent] vs 2 个基于规则的 agent
5(2V2)	[1 个并行 MCTS agent & 1 个基于规则的 agent] vs 2 个基于规则的 agent

4.4 性能分析

为了有效对比分析不同版本蒙特卡洛树搜索算法的性能,实验设置中串行版本和并行版本的 MCTS 迭代次数均设置为 400,并且为了使并行版本的 MCTS 迭代平均分布到不同子进程中,并行版本设置的子进程数量分别为 2,4 和 8。这样能够在充分利用有限计算机资源的同时,均衡实现子进程的 MCTS 搜索。

4.4.1 对局耗时

本次实验的实验对象是表 4 中的实验 1、实验 2 和实验 4、实验 5,串行 MCTS 与并行 MCTS 分别运行相同的时间(24 h)。记录实验 1、实验 2 和实验 4、实验 5 完成的对弈局数,FFA 模式和 2V2 模式的结果如图 6 和图 7 所示。

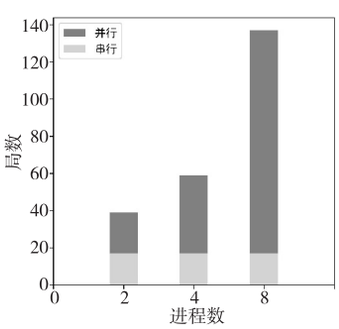

Figure 6 Number of games(FFA)

图 6 对弈局数(FFA)

通过图 6 和图 7 固定时间博弈局数的数据分析本文设计的算法的耗时。在 FFA 模式下,子进程数量为 2 的情况下,对局完成的时间约为串行完成时间的 40%。2V2 模式下,子进程数量为 2 的情况下,对局完成的时间约为串行完成时间的 25%。

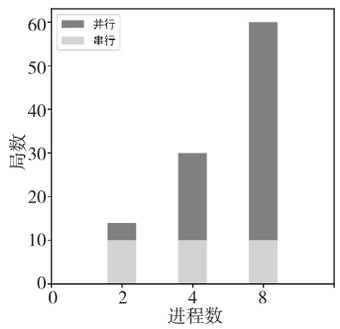

Figure 7 Number of games(2V2)

图 7 对弈局数(2V2)

这验证了本文设计的基于胜率估值传递的并行蒙特卡洛树搜索算法可以有效缩减博弈树的搜索时间。

4.4.2 对弈胜率

由 4.4.1 节的数据分析可以得到,在 FFA 模式下,本文设计的基于胜率估值传递的并行蒙特卡洛树搜索算法会将搜索时间大约缩短为原来的 $1/n$ (n 为并行的子进程数量),在大幅缩短搜索时间的同时也降低了胜率。为了探索胜率不变的情况下,搜索时间最多能够缩短多长时间,本节实验选取了迭代次数不同的串行智能体与并行智能体进行对抗。

本节按照表 4 中实验 3 的配置进行设计,在博弈局数相同的情况下,不同迭代次数的串行 MCTS 与不同子进程下并行 MCTS 智能体进行对弈,实验结果如图 8 所示。本文将子进程数为 2、迭代次数为 200 的并行 MCTS 与迭代次数为 150 的串行 MCTS 进行对弈。统计结果如表 5 所示,数据变化曲线如图 8 所示。通过图 8 可以得出,随着博弈局数的增加,并行 MCTS 胜率在数据 0.5 上下波动,并行智能体与串行智能体的博弈能力不相上下,两者对弈胜率相近。

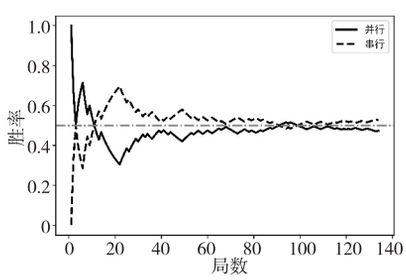

Figure 8 Winning rate of parallel MCTS agents vs serial MCTS agents when playing chess

图 8 并行 MCTS 智能体 vs 串行 MCTS 智能体对弈胜率

本文对表 5 和图 8 的数据分析,验证了本文设计的基于胜率估值传递的并行蒙特卡洛树搜索算法的有效性。基于表 4 中实验 3 的实验配置,本文设计的并行 MCTS 算法在有效缩短约 20%搜索时间的情况下,与串行MCTS智能体对抗获胜的胜

Table 5 Parallel MCTS vs serial MCTS

表 5 并行 MCTS vs 串行 MCTS

总局数	并行 MCTS 获胜局数	胜率
15	8	0.533
30	15	0.500
45	20	0.445
60	30	0.500

率约为 50%。进一步验证了在有效缩短搜索时间的情况下,本文设计的基于胜率估值传递的并行蒙特卡洛树搜索算法未对博弈性能造成影响。

本文通过对弈耗时与对弈胜率 2 个指标值的分析,验证了本文设计的基于胜率估值传递的并行蒙特卡洛树搜索算法是可行且有效的,提高了 MCTS 中博弈树的搜索效率。

5 结束语

基于 Pommerman 博弈平台,本文在蒙特卡洛树搜索算法的并行优化的理论基础上,设计并实现了基于胜率估值传递的并行蒙特卡洛树搜索算法,并给出了算法的详细步骤。本文通过一系列的对比实验来对优化的并行蒙特卡洛树搜索算法进行测试并评估其性能,实验结果表明,本文设计的基于胜率估值传递的并行蒙特卡洛树搜索算法在与原有博弈算法保持相同博弈胜率的情况下,能够至少缩短 20%的搜索时间。

未来的研究重点将集中于尝试将蒙特卡洛树搜索与深度强化学习策略相结合,例如叶子节点第 1 次展开时可以用策略网络获得先验评分;探索一种 CPU 与 GPU 异步工作的方案,在提高算法在博弈树中搜索效率的同时,更加充分地调用计算资源,避免 GPU 资源的浪费。

参考文献:

[1] Cooper S B, Leeuwen J V. Digital computers applied to games[M]//Alan Turing: His Work and Impact. Amsterdam: Elsevier, 2013: 623-650.

[2] Silver D, Huang A, Maddison C, et al. Mastering the game of Go with deep neural networks and tree search[J]. Nature, 2016, 529(7587): 484-489.

[3] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge[J]. Nature, 2017, 550(7676): 354-359.

[4] Yin Chang-sheng, Yang Ruo-peng, Zhu Wei, et al. A survey on multi-agent hierarchical reinforcement learning[J]. CAAI Transactions on Intelligent Systems, 2020, 15(4): 646-655. (in Chinese)

[5] Resnick C, Eldridge W, Ha D, et al. Pommerman: A multi-agent playground[J]. arXiv: 1809.07124, 2018.

[6] Coulom R. Efficient selectivity and backup operators in Monte-Carlo tree search[C]//Proc of the 5th International Conference on Computers and Games, 2006: 72-83.

[7] Kupferschmid S, Helmert M. A skatplayer based on Monte-Carlo simulation[C]//Proc of the 5th International Conference on Computers and Games, 2006: 135-147.

[8] Leckie W, Greenspan M. Monte-Carlo methods in pool strategy game trees[C]//Proc of the 5th International Conference on Computers and Games, 2006: 244-255.

[9] Chaslot G, Bakkes S, Szita I, et al. Monte-Carlo tree search: A new framework for game AI[C]//Proc of the 4th Artificial Intelligence & Interactive Digital Entertainment Conference, 2008: 216-217.

[10] Lu Meng-xuan. Key technologies research of Einstein Würfelt Nicht! computer game [D]. Hefei: Anhui University, 2019. (in Chinese)

[11] Lin Hua. Gobang intelligent gaming robot based on Self-Play[D]. Hangzhou: Zhejiang University, 2019. (in Chinese)

[12] Liu Zi-zheng, Lu Chao, Zhang Rui-you. Monte Carlo simulation and Z-test based parallel algorithm for the game of 2048[J]. Journal of Chinese Computer Systems, 2016, 37(3): 562-566. (in Chinese)

[13] Zhang Xiao-chuan, Li Qin, Nan Hai, et al. Application of improved UCT algorithm in EinStien Würfelt Nicht! computer game[J]. Computer Science, 2018, 45(12): 196-200. (in Chinese)

[14] Chaslot G M J B, Winands M H M, van den Herik H J. Parallel Monte-Carlo tree search[C]//Proc of the 6th International Conference on Computers and Games, 2008: 60-71.

[15] Liu A J, Chen J S, Yu M Z, et al. Watch the unobserved: A simple approach to parallelizing Monte Carlo tree search [J]. arXiv: 1810.11755, 2018.

[16] Méhat J, Cazenave T. Tree parallelization of Ary on a cluster[C]//Proc of International Joint Conference on Artificial Intelligence, 2011: 39-43.

附中文参考文献:

[4] 殷昌盛,杨若鹏,朱巍,等. 多智能体分层强化学习综述[J]. 智能系统学报,2020,15(4):646-655.

[10] 陆梦轩. 爱恩斯坦棋计算机博弈关键技术研究[D]. 合肥:安徽大学,2019.

[11] 林华. 基于 Self-Play 的五子棋智能博弈机器人[D]. 杭州:浙江大学,2019.

[12] 刘子正,卢超,张瑞友. 基于蒙特卡罗模拟和 Z 检验的"2048"游戏并行优化算法[J]. 小型微型计算机系统,2016,37(3):562-566.

[13] 张小川,李琴,南海,等. 改进 UCT 算法在爱恩斯坦棋中的应用[J]. 计算机科学,2018,45(12):196-200.

作者简介:

管延霞(1998 -),女,山东潍坊人,硕士生,研究方向为并行强化学习。**E-mail**: gyxouc@163.com

GUAN Yan-xia, born in 1998, MS candidate, her research interest includes parallel reinforcement learning.