# 基于ADP与MCTS算法的五子棋

唐振韬，赵冬斌，邵坤，吕乐

中国科学院自动化研究所复杂系统管理与控制国家重点实验室。北京100190

[tangzhentao2016@ia.ac.cn，dongbin.zhao@ia.ac.cn，shaokun2014@ia.ac.cn，iamlvle@126.com](mailto:tangzhentao2016@ia.ac.cn，dongbin.zhao@ia.ac.cn，shaokun2014@ia.ac.cn，iamlvle@126.com)

摘要

受AlphaGo核心思想的启发，我们将自适应动态规划（ADP）方法训练的神经网络与运用于五子棋的蒙特卡罗树搜索（MCTS）算法相结合。MCTS算法基于蒙特卡罗模拟方法，经过大量模拟并生成游戏搜索树。 我们展开该树并搜索其中叶节点的结果。结果，我们获得了MCTS获胜率。ADP和MCTS方法分别用于估算获胜率。我们对这两个中奖率进行加权，以选择最大一个的作用位置。实验结果表明，该方法可以有效地消除神经网络评价函数的“短视”缺陷。使用我们提出的方法，游戏的最终预测结果更准确，并且它优于基于ADP算法的五子棋。

关键词：自适应动态规划; 蒙特卡洛树搜索;五子棋

一，导言

电脑棋盘游戏长期以来一直是人工智能研究的重点。五子棋是一款受欢迎的双玩家战略棋盘游戏。传统上，我们在15x15交叉的棋盘上用棋子（黑色和白色的石头）玩。获胜者是首先水平，垂直或对角地获得连续五颗棋子的玩家。为了解决这类游戏，一些典型的方法被提了出来，如证明号搜索[1]，基于依赖关系搜索[2]和线程空间搜索[3]。玩五子棋的最经典的算法之一是游戏树搜索，它基于结合了叶子盘面情况的盘面评估函数的min-max树。然而，正如William所说[4]，对n次走子深度的完整搜索需要p！/（p-n）！次盘面评估，其中p是当前合法走子的数量。因此，完全完成一次搜索是一项不可能完成的任务。幸运的是，结合alpha-beta搜索的历史启发式算法已被用于加速游戏树搜索[5]。虽然我们都知道一个算法在游戏树中搜索得越深，它就越有效。这些方法有一个明显的缺陷：时间和空间复杂度随着搜索深度呈指数级增长。换句话说，搜索深度总是成为瓶颈。

为了解决这个问题，我们提出了一种将浅层神经网络与蒙特卡罗模拟相结合的五子棋新算法。使用ADP来训练神经网络并与自己对抗可以为五子棋产生专业的玩家。在训练之后，神经网络可以获得任何可能的棋盘状况的胜利概率。事实上，我们使用神经网络来评估盘面情况，并获得合理数量的候选走子以备选择。然后，我们将这些候选走子作为MCTS的根节点，并尝试将我们的走子预测网络与MCTS进行整合。因此，我们分别从神经网络和MCTS获得两个获胜概率结果。预测的最终获胜概率是加权神经网络和MCTS结果中的最大总和。

剩余论文的组织安排如下：在第二部分，我们讨论了一些使用神经网络或强化学习研究五子棋的相关工作。第三部分简要介绍了MCTS。第四部分详细介绍了MCTS的使用和ADP的实施。第五部分给出了显示ADP与MCTS算法的性能的实验结果以及比较结果。最后，我们提出讨论并总结论文，同时指出未来研究的方向。

二，相关工作

早在20世纪90年代，Freisleben提出了一个神经网络，它有能力学习下五子棋[6]。它的本质是脱胎于一种特殊的强化学习算法，通过奖励或惩罚来训练一个网络，这个算法被称之为比较训练[7]。强化学习是一种新颖的机器学习方法，它涉及软件应该如何在环境中采取行动，以最大化累积奖励的一些概念。强化学习的最具竞争优势在于它不需要关于马尔可夫决策过程（MDP）的知识，并且可以在确切方法变得极其复杂时针对大型MDP运用，如德州扑克[8]和围棋 [9]。此外，强化学习已被用作解释大脑行为选择机制的模型[10]。时序差分（TD）学习主要用于强化学习问题，这是一种基于预测的机器学习方法。Mo[11]和Gong[12]将TD学习算法应用于五子棋。尽管如此，实验结果表明，这种方法适用于五子棋不如TD-Gammon[13]那样有效。

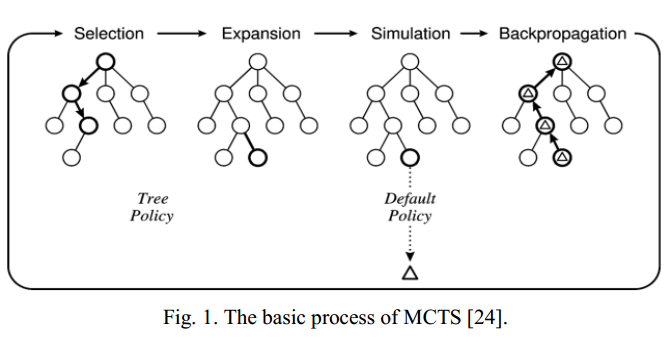
尽管如此，我们认为在TD学习中，行为决策或价值函数也可以用连续的形式来描述，通过神经网络中的非线性函数线来近似。这是自适应动态编程（ADP）的核心思想[14-16]。通过将它与三层完全连接的神经网络进行配对，以提供适应性和自学习行为，已经改进了应用于五子棋的ADP性能。但是，神经网络的输入是按照预定的模式设计的。因此，网络只对那些具有专业知识的游戏有效。而且，它对神经网络评估函数有一个短视的缺陷。

蒙特卡罗树搜索（MCTS）是一种通过在决策空间中进行随机模拟并根据结果构建搜索树来寻找给定域中的最优决策的方法。此外，它在数值算法中有着悠久的历史，并且在诸如Scrabble[17]，Bridge[18]，特别是Go[19]等各种AI游戏中取得了重大成功，如MoGo[20]，ManGO[21]。虽然MCTS也是被推荐运用于五子棋，但它没有像预期的那样取得很好的效果。这主要是因为MCTS需要额外的复杂领域知识才能在较高水平上工作。此外，MCTS必须花费大量的时间进行模拟才能获得满意的结果。

计算机Go计划由DeepMind创建的AlphaGo在与Lee Sedol的五场比赛中赢得4：1，是世界上最好的围棋玩家。根据DeepMind的论文[22]，AlphaGo采用了一种将深度神经网络与蒙特卡洛模拟相结合的新方法来评估电路板状况并选择最佳移动。受其启发，我们将蒙特卡罗树搜索应用到五子棋中，并结合我们以前的工作[23]。因此，我们实际上获得了ADP和MCTS算法的最终胜率。

三，蒙特卡罗树搜索

蒙特卡洛树搜索（MCTS）[24]需要大量的仿真，并根据结果建立一个大型的搜索树。 MCTS的一个重要特征是随着仿真时间和节点访问量的增加，估计值会变得越来越精确。 MCTS的基本过程如图1所示。它由四个主要阶段组成：选择，扩展，模拟和反向传播。



MCTS的基本过程从第一阶段选择阶段开始。在这个阶段，它从根节点开始，递归地应用子选择策略（也称为树策略），通过树下降直到到达最紧急的可扩展节点。然后在扩展阶段，可以根据可用操作添加一个或多个子节点来扩展树。在第三阶段模拟阶段，它可以根据已解决的策略从叶节点运行模拟（或称为默认策略）来产生结果。最后，在后向传播阶段，它可以通过选定的节点将仿真结果传回来更新它们的状态值。

在本文中，我们提出了两种MCTS算法。一个被称为启发式蒙特卡洛树搜索（HMCTS），另一个被称为树的上置信区间（UCT）。基于HMCTS的五子棋算法在算法1中给出。

**算法1：基于HMCTS的五子棋算法**

输入：原始状态s0;

输出：对应于MCTS的动作a的最高值;

添加启发式知识;

从状态s0获得可能的走子M;

For可能走子M中的每一步走子：

奖赏 rtotal←0;

当 （模拟时间 < 分配时间）：

奖励r ← 模拟(s(m));

rtotal ← rtotal + r;

模拟时间加1;

结束当循环

将（m，rtotal）加入数据;

结束for循环

返回行动Best(data)

仿真（状态st）

如果（状态st是赢且st是最后一子）则 返回1.0;

否则 返回0.0;

如果（st满足启发式知识）

然后 获得强制行动af;

新状态st+1←f(st,af);

否则 选择随机行动ar ∈ 未尝试的动作;

新状态st+1←f(st,ar);

返回 模拟（st+1）

Best(data)

返回 行动a //从数据中获取m中的最大值rtotal

请注意，这里f是一个函数，用于根据上一个盘面状态和动作生成新的盘面状态。对五子棋玩家来说是常识的启发式知识可以比随机抽样节省更多的模拟时间。因此，它有助于结果比以前更早地收敛。规则解释如下：

1. 如果我方产生四子相连，棋手将会将其棋子下在可能产生我方五子相连的位置
2. 如果对方产生四子相连，棋手将被迫将其棋子下在可封锁对面五子相连的位置
3. 如果我方产生三子相连，棋手将会将其棋子下在可能产生我方四子相连的位置
4. 如果对方产生三子相连，棋手将被迫将其棋子下在可封锁对面四子相连的位置

五子棋虽然是像Go一样的零和游戏，但在五子棋中很少会出现平局。事实上，最后的结果通常是赢或者输。因此，当最终结果为赢时，我们将奖励设为1，否则当最终结果为损失或平局时，奖励为0。那么行为的Q值可以代表该行为的预期回报。

(1)

其中N(s,a)是从状态s中选择动作a的次数，N(s)是从状态s中游戏对局的次数，zi是第i次模拟的结果。如果动作a在第i次对局中被选中，那么li(s,a)等于1，否则为0。

另一种广泛使用的MCTS算法是UCT [24]，它基于上置信区间（UCB）。 UCB被称为有能力解决多臂赌徒问题。UCB最明显的优点在于，它有助于平衡勘探和开发之间的冲突，并在早期找出最终结果。其最简单的形式是：

(2)

其中，是第j次模拟的平均奖励，nj是节点j被访问的次数，n是到目前为止的总次数。 奖励鼓励利用更高的奖励选择，但是右边的公式鼓励探索较少访问的选择。

UCT起源于HMCTS，但与HMCTS的区别在于UCB可以比原始算法更早找出合适的叶节点，因此UCT比原始版本节省更多的时间。

基于UCT的五子棋算法在算法2中给出。

**算法2：基于UCT的五子棋算法**

输入：创建状态s0的根节点v0;

输出：对应于UCT的最高值动作a;

当 在计算预算内

vl←树策略(v0);

策略←启发式知识;

奖励r←策略(s(v1));

返回更新(vl,r);

结束 当

返回行动a(Best Child(v0))

树策略（节点v）

当 v不处于最终状态

如果 v未完全展开，则返回Expand(v);

否则 v←Best Child(v,);

结束当

返回v //这是最好的子节点

展开（节点v）

从A（s（v））中选择未尝试的随机动作a;

添加一个新的子v'到v

S(v’) ← f(s(v),a)和a(v') ← a;

返回v' //这是扩展节点

Best Child(节点v，参数c)

返回



策略（状态s）

当 s不是最终状态

如果 s满足启发式知识 那么获得强制行动a;

否则 选择随机行为a∈统一A(s);

s←f（s，a）;

结束当

返回 状态s的奖励

返回更新（节点v，奖励r）

当 v不是空的

N(v) ← N(v) + 1;

Q(v) ← Q(v)+ r;

v←v的父项;

结束当

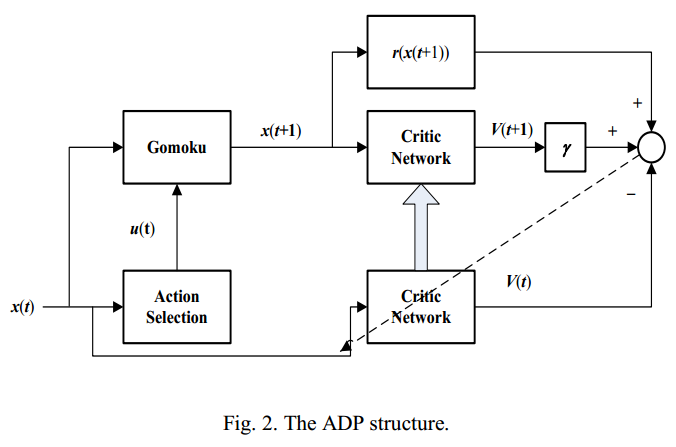
这里，v表示具有四个数据的节点：状态s(v)，下一个动作a(v)，总模拟奖励Q(v)，被访问计数N(v)。 v0是状态s0对应的根节点，v1是到达游戏模拟结束的最后一个节点，r是走棋策略达到最终状态的奖励，总体搜索结果为a(Best\_Child(v0))是导致根节点v0的最佳子节点的动作。

请注意，MCTS需要重复执行足够多的时间以确保预测的准确性。MCTS中最耗时的问题是MCTS必须花费大量时间搜索一些不必要的可行操作（不必要的操作意味着它以低赢率获胜）。

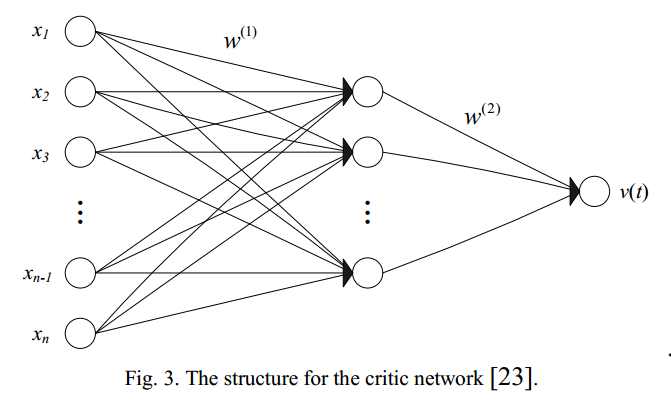
四，用蒙特卡洛树搜索进行自适应动态规划

五子棋中使用的自适应动态规划（ADP）通过时序差分学习（TDL）进行训练，这是一种广泛使用的强化学习算法。 ADP训练结构如图2所示。训练ADP的细节可参见[23]。为了解决我们之前提到的问题，我们试图通过ADP获得候选走子动作。每个从ADP获得的候选走子都应该是MCTS每个进程对应的根节点。换句话说，它不仅确保了搜索的准确性，而且还减少了搜索的宽度。与仅使用MCTS相比，应该节省大量时间以找出适合五子棋的动作。

当前棋盘状态x(t)被前馈给动作选择，它产生控制动作u(t)。在u(t)的作用下，我们得到下一步转移状态x(t+1)，它被前馈到产生奖励r(x(t+1))的效用函数r。批评网络被用于估计成本函数V.然后，利用奖励r(x(t+1))，估计值V(t)和估计值V(t+1)来更新批评网络的权重使成本函数V满足Bellman方程。



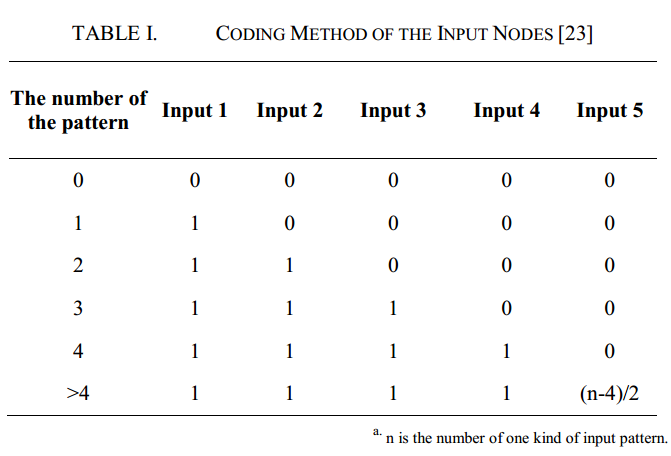
五子棋的ADP中的批评网络是一个前馈三层全连接神经网络。结构如图3所示



神经网络的最终输出v(t)是具有棋盘状态的玩家的获胜概率，推导如下。

其中是第j个输入节点和第i个隐藏节点之间的权重；xj是输入层的第j个输入；n是输入节点的总数；是第i个隐藏节点的输入；是第i个隐藏节点的输出；是隐藏节点和输出节点之间的权重；m是隐藏节点的总数； p(t)是输出节点的输入。

在批评网络中，输入层有274个节点，隐藏层有100个节点，输出层有1个节点。在输入层中，有五个输入节点，指示除五行之外的每个模式的数目。编码方法如表1所示。[23]中也提到了输入节点数量为274的原因。



作为[23]中所示的实验结果，我们选择与其余案例相比性能最好的程序ST-Gomoku[23]作为我们的神经网络评估函数。但不像我们以前的工作，我们不是通过神经网络评估函数只获得一次获胜概率最大的一次走子，而是获得五次获胜概率最高的五次候选走子。

受到AlphaGo思想的启发，我们尝试使用ADP与MCTS结合来训练浅层神经网络。首先，我们从ADP训练的神经网络中获得5个候选走子和他们的获胜概率。我们称他们为ADP获胜概率。其次，5个候选走子和他们的盘面情况被视为MCTS的根节点。然后，我们从MCTS方法分别获得5个获胜概率。我们称他们为MCTS获胜概率。为了更准确地预测获胜概率，我们计算ADP的和及其对应的MCTS的获胜概率的加权总和。它被定义为：

其中是预测的最终获胜概率，是ADP的获胜概率，是MCTS的获胜概率，λ是[0,1]之间的实常数。正如它表示的那样，当λ= 0时，获胜预测仅取决于MCTS。相反，λ= 1意味着获胜预测仅取决于ADP。

完整的ADP-MCTS在算法3中给出。

**算法3：ADP和MCTS结合**

输入：原始状态s0;

输出：对应于结合的ADP和MCTS动作a;

MADP，WADP ← ADP状态(s0);

WMCTS ← MCTS状态(MADP);

对 概率w1，w2在成对的(WADP,WMCTS)中

←;

将p添加到P中;

返回 动作a对应于P中的max p

ADP的状态(状态s)

从ADP获得前5个获胜概率WADP;

获得他们对应于WADP的走子MADP;

返回MADP，WADP

MCTS的状态（走子MADP）

对 MADP中的每一步走子都是这样做

创建m作为具有相应状态s的根节点

从MTCS(m,s)获得

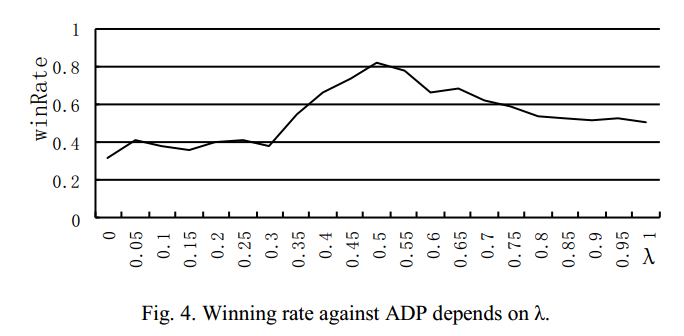
将添加到WMCTS中

返回WMCTS

这里s0表示原始状态。MADP是ADP走子的集合，MMCTS是MCTS走子的集合。 WADP是ADP获得前5名的概率的集合，WMCTS是MCTS获胜概率的集合。ADP和MCTS方法意味着分别由ADP或MCTS从可行的走子中找出他们的获胜概率。

如图4所示，当λ= 0.5时，它似乎是最好的结果。换句话说，只有当它对ADP和MCTS的依赖性平衡时，获胜概率预测才会更加精确。

应该指出的是，从ADP获得的候选走子使得MCTS的搜索空间比以前小。这就是为什么结合ADP和MCTS的方法比仅使用MCTS的方法节省更多时间。另外需要注意的是选择5作为候选走子数的原因。当候选走子的数量远远大于5时，它不会像预期的那样节省时间。相比之下，如果候选走子的数量小于5，那么最有可能与ADP结果相同。另外，5是基于实验的结果，其结果5是平衡五子棋耗时和游戏等级所得出的结果。



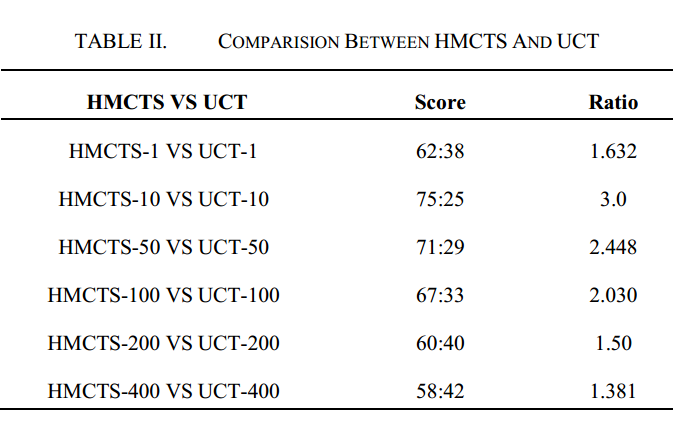
五，实验和分析

目前ADP与HMCTS或UCT相结合的方法已经实施。 我们的目标如下：首先，比较HMCTS和UCT的区别。然后，比较上面提到的四种不同的方法，并挑选出相互竞争时表现最好的那个。最后，选出的最好的一个将与一个名为五星级的五子棋[25]的商业程序进行比赛。

在这些实验中，测试系统基于AMD A10-5750M APU的硬件平台，采用Radeon(tm) HD Graphics 2.50GHz，软件平台为Windows 10.此外，开放式多处理（OpenMP）是一款应用程序编程接口,它支持C和C ++中的多平台共享内存多处理编程，因此我们可以使用它来帮助MCTS更早地收敛到最终结果。

**A. HMCTS和UCT之间的比较**

在表II中，第一列及其对应行中的数字表示使用UCT或HMCTS进行模拟的次数。考虑到相同的MCTS方法，采取的模拟越多，获胜的概率就越高。这证明了随着模拟次数的增加可以提高获胜概率的精度。请注意，MCTS预测的精度随对数形式增加，因此，当模拟数量超过1000时，它几乎不可察觉。



表II列出了HMCTS-n或UCT-n，其中n表示模拟棋盘游戏的次数。当n次数较少时，HMCTS的性能比UCT好。然而，随着仿真次数的增加，UCT的水平将更接近HMCTS。但总体而言，HMCTS的水平略高于UCT，而HMCTS的所用时间低于UCT。

值得注意的是，由于MCTS耗时过多，我们将模拟次数限制为400次。实际上，当模拟次数达到400次时，决定合适的移动可能需要2到3分钟，所以它显然难以用于五子棋。另外，如果仅增加一点模拟时间，它就不会取得重大进展。

关于MCTS的最糟糕的事情是为了获得更准确的预测而花费的时间。而大部分可行的举措都是毫无价值的。

**B.四种不同方法之间的比较**

请注意，我们只使用HMCTS作为MCTS成为竞争对手，因为已经证明HMCTS的性能优于UCT。表4显示了五子棋从高到低的水平是ADP-UCT，ADP-HMCTS，ADP，HMCTS。

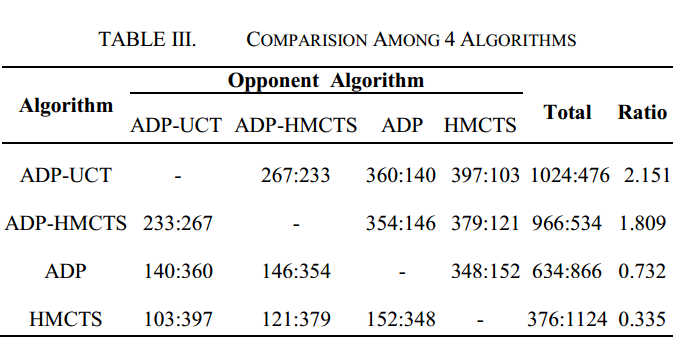


表3显示，ADP-MCTS的性能优于ADP，并且比MCTS好得多。虽然UCT的表现比HMCTS差，但ADP-UCT的表现略好于ADP-HMCTS。

在实验过程中，所提议的方法所耗费的时间主要在MCTS上。ADP的耗时仅为每次移动80毫秒左右。尽管如此，MCTS可以花费比ADP多得多的时间，每次移动大约需要5〜10分钟。造成这一时间消耗的主要原因是MCTS可能会花费大量时间进行大量可能移动的模拟。为了使其有效，我们使用ADP来选择少量的最终候选动作（例如5次走子.），这样可以在短时间内保证MCTS的模拟。结果表明，使用OpenMP的ADP-HMCTS成本从原来的10s降低到2〜3s，如果没有它的话。 但是，ADP-UCT仅花费了大约4〜5秒的时间而未使用OpenMP。显然，ADP-UCT不仅比ADP-HMCTS更强大，而且更快。

**C.对抗五星级的五子棋算法**

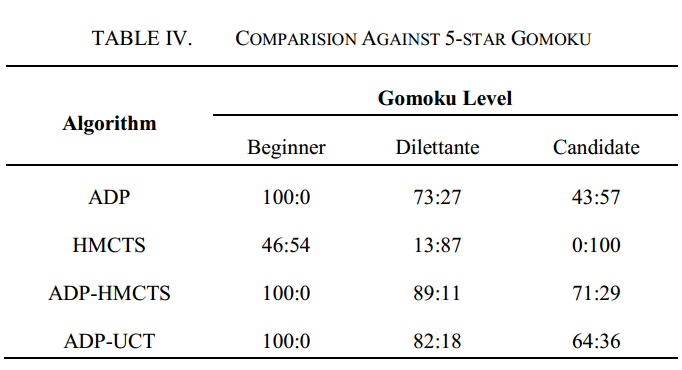
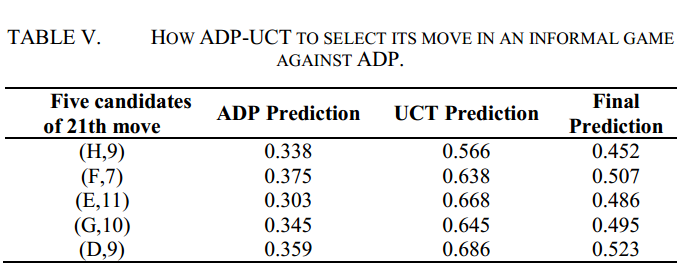
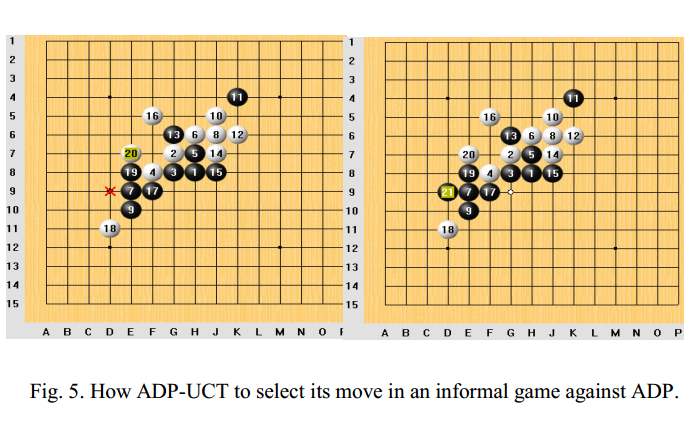


表4表明，ADP-HMCTS和ADP-UCT均达到候选人水平，而ADP达到业余爱好者水平，HMCTS仅仅达到初学者水平。



表五显示每次走子后总是有五个候选走子。所选走子由最终预测值的最大值决定，相当于0.5×(ADP预测+UCT预测)。从表五和图五我们可以看到(D,9)是确保赢得的最终选定举措。在这个过程中，它主要取决于UCT。而ADP仍处于前两名的位置。事实表明，UCT可能是ADP的优秀补充，它真正提高了获胜概率预测的准确性。实际上，ADP和UCT提高了下五子棋的表现。

六，讨论与结论

在之前的五子棋研究中，作为一种常用的方法，α-β剪枝用于生成节点顺序，而访问节点的具体操作是计算静态评估函数的值。事实表明它在传统的五子棋算法中扮演了重要角色。而这种传统的方法也带来了3个严重的问题。首先，静态评估函数总是需要复杂的人工设计，并且需要大量的时间来考虑大量的情况。第二个是在下五子棋时不能学习任何东西。它只服从以前制定的规则，不能通过下棋来改进。最后是那个搜索深度总是一个瓶颈。时间和空间的复杂性将随着搜索深度呈指数级增长，这限制了基于游戏树的解算器的实时性能。

但是，我们可以训练一个能够学习使用ADP下五子棋的神经网络。事实证明，ADP的五子棋计划接近五星级五子棋的候选水平。同时，获得神经网络获胜概率的预测只需要一两毫秒的时间，并且它可以决定平均60ms到80ms的走子。因此，它比基于游戏树的求解器快得多。此外，我们可以通过自我对抗来训练神经网络，并通过学习盘面的情况来显示其提高五子棋水平的能力。也就是说，我们可以通过训练神经网络而不是用规则编程来提高水平。

总体而言，我们提出了一种通过结合ADP和MCTS算法来解决本文中的策略游戏的方法。在实验中，结合了APD和MCTS的算法已经战胜了达到候选人水准的五星五子棋。然而，它与五子棋最好的AI程序YiXin还有一定的差距。尽管为特定的棋盘游戏开发更强大的AI是终极目标，但自我游戏仍然是AlphaGo所证明的强大技术。我们将尝试采用深度神经网络来在下一阶段做一个更好的棋盘游戏。

致谢

我们特别感谢张震和戴钰桀的宝贵前期工作。 同时我们也感谢DeepMind为AlphaGo做的出色工作。

参考

[1] L. V. Allis，M. V. D. Meulen和H. J. V. V. Herik，“证明号码搜索”，Artificial Intelligence，vol。 66，第91-124页，1994。

[2] I. C. Wu，H. H. Kang，H. H. Lin，P. H. H. Lin，T. H. Wei和C. M. Chang，依赖依赖搜索Connect6：Springer International Publishing，2013。

[3] L.V. Allis，H.J.Van Den Herik和M. P.H.Huntjens，“Go-Moku and Threat-Space Search”，面试问题，1994。

[4] W.T.Katz和S.Pham在IEEE International Conference on Systems，Man，and Cyber​​netics，1991，pp.1405-1410第2卷，“使用Go-Moku的基于体验的学习实验”

[5] J. Schaeffer，“实践中的历史启发式和alpha-beta搜索增强”，IEEE Transactions on Pattern Analysis and Machine Intelligence，vol。 11，第1203-1212页，1989。

[6] B. Freisleben，“学习五连奏的神经网络”，在新西兰国际双流会议人工神经网络和专家系统，1995年。会议记录，1995年，页。 87-87。

[7] G. Tesauro，“Connectionist learning of expert preferences by comparison training，”Advances in neural information processing systems 1,1989，pp.99-106。

[8] FA Dahl，“增强学习算法应用于简化的双人德克萨斯扑克游戏”，机器学习：Emcl 2001，欧洲机器学习会议，德国弗赖堡，2001年9月5日至7日，会议论文集， 2001年，第85-96页。

[9] D. Silver，R. Sutton和M.Müller，“强化学习局部形状在围棋中”，载于人工智能国际联合会议论文集，印度海德拉巴，2007年1月， 1053年至1058年。

[10] F. Ishida，T. Sasaki，Y. Sakaguchi和H. Shimai，“具有不同温度参数的增强学习代理解释了马尔可夫决策过程任务中人类行动选择行为的多样性”，Neurocomputing， 72，pp。1979-1984，2009。

[11] J. W. Mo，“关于GamePlaying的机器自学的研究和实践”，。第一卷。硕士论文：广西师范大学，2003。

[12] R. Gong Gong，“基于强化学习的计算机游戏策略的研究与实现”，。第一卷。硕士论文：沉阳理工大学，2011。

[13] G. Tesauro，“时间差异学习和TD-Gammon，”​​ACM的通信，第一卷。 38，第58-68页，1995。

[14] A. G. Barto，R. S. Sutton和C. W. Anderson，“Neuronlike adaptive elements that can solve difficult learning control problems，”IEEE Transactions on Systems Man＆Cyber​​netics，vol。 SMC-13，第834-846页，1983。

[15] J.W.保罗，“强化学习随着时间的推移设计菜单”神经网络的控制，麻省理工学院出版社，剑桥，麻省，1990年。

[16] D. Zhao，Z. Xia，D. Wang，“基于动作依赖启发式动态规划和收敛性分析的仿射非线性系统的无模型最优控制”，IEEE自动化与科学工程汇刊。第一卷。 12，没有。 4，pp。1461-1468，2015

[17] A.Ramírez，FGAcuña，AG Romero，R.Alquézar，E.Hernández，AR Aguilar和IG Olmedo，“基于概率在冠军水平上执行的拼字游戏启发式”，MICAI 2009：Advances in Artificial Intelligence，第5845卷计算机科学讲义，第112-123页，2009年。

[18] M. L. Ginsberg，“GIB：在计算挑战性游戏中的不完美信息”，Journal of Artificial Intelligence Research，第303-358页，2001。

[19] A. F. Smith和G. O. Roberts，“Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods，”Journal of the Royal Statistical Society。 B辑（方法论），第3-23页，1993年。

[20] S. Gelly和Y. Wang，“勘探开发中的应用：UC​​T for MonteCarlo go”，NIPS：神经信息处理系统会议2006年在线勘探开发工作坊。

[21] G. M. J. Chaslot，M. H. Winands，H. J. V. V. HERIK，J. W. Uiterwijk和B. Bouzy，“Progressive strategies for Monte-Carlo tree search”，New Mathematics and Natural Computation，vol。 4，pp.343-357,2008。

[22] D. Silver，A. Huang，CJ Maddison，A. Guez，L. Sifre，G. van den Driessche，J. Schrittwieser，I. Antonoglou，V. Panneershelvam，M. Lanctot，S. Dieleman，D. Grewe，J. Nham，N. Kalchbrenner，I. Sutskever，T. Lillicrap，M. Leach，K. Kavukcuoglu，T. Graepel和D. Hassabis，“用深度神经网络和树搜索掌握Go的游戏” Nature，vol。 529，no.7587，pp.484-489,2016。

[23] D.赵，Z.张和Y.戴，“自我教学自适应动态编程Gomoku，”Neurocomputing，第一卷。 78，第23-29页，2012。

[24] CB Browne，E. Powley，D. Whitehouse，SM Lucas，PI Cowling，P. Rohlfshagen，S. Tavener，D. Perez，S. Samothrakis和S. Colton，“蒙特卡罗树搜索方法综述，“IEEE Transactions on Computational Intelligence and AI in Games，vol。 4，第1-43页，2012。

[25] L.Atomax，“http://www.5-star-gomoku.com-about.com/”，2006年。