成　绩	
评卷人	

研究生	冯朗
学　号	2015363065

武漢紡織大學

研 究 生 课 程 论 文

论文题目	博弈树搜索技术
完成时间	2020. 10. 31
课程名称	人工智能
专　　业	电子信息
年　　级	2020

武汉纺织大学研究生处制

注：（1）正文宋体小四 1.5 倍行距；（2）不要列出源程序；（3）页数不少于 8 页；

一、 引言

关于搜索理论的研究是人工智能的核心领域，在人工智能领域，搜索即对某种问题的求解过程，而搜索方式就是求解该问题的方式或者模型。从提出问题（即初始状态）到解决问题（即目标状态），整个的求解过程， 事实上就是一个状态空间搜索[1]的过程。博弈树搜索在人工智能领域颇为重要，计算机博弈，也称机器博弈，是博弈论和计算机技术结合的产物。

20 世纪 50 年代开始，许多世界上著名的学者都曾经涉足计算机博弈领域的研究工作，为机器博弈的研究与开发奠定了良好的基础。阿兰·图灵(Alan Turing) 先生最早写下了能够让机器下棋的指令，计算机之父冯·诺依曼 (John von Neumann) 提出了用于博弈的极大极小定理，信息论创始人科劳德·香农(Claude E.Shannon)首次提出了国际象棋的解决方案，人工智能的创始人麦卡锡(John McCarthy)首次提出“人工智能”(artificial intelligence)这一概念。1958 年阿伯恩斯坦 (Alex Bernstein)等在 IBM704 机上开发了第 1 个成熟的达到孩童博弈水平的国际象棋程序。1959 年，人工智能的创始人之一塞缪(A.L.Samuel) 编了一个能够战胜设计者本人的西洋跳棋程序，1962 年该程序击败了美国的一个州冠军。

20 世纪 80 年代末，随着计算机硬件和软件技术不断发展，计算机博弈理论日趋完善，电脑能否战胜人脑这个话题产生了浓厚的兴趣，并提出了以棋类对弈的方式，向人类发起挑战。1989 年 IBM 公司研制的“深思”在与世界棋王卡斯帕罗夫进行的“人机大战”中，以 0：2 败北。1995 年 IBM 更新了“深蓝”程序，在 1996 年与卡斯帕罗夫的挑战赛中以 2：4 败北。1997 年“超级深蓝”融入了更深的开发，以 3.5：2.5 击败了卡斯帕罗夫，这场胜利引起了世界范围内的轰动。

由于最近几年，基于人工神经网络取得突破性的进展，计算机博弈领域中的许多问题得到解决。2012 年 6 月，谷歌公司的 Google Brain 项目用并行计算平台训练一种称为“深度神经网络” (deep neural networks, DNN)

[1] 状态空间搜索就是将问题求解过程表现为从初始状态到目标状态寻找这个路径的过程。由于求解问题的过程中分枝有很多，主要是求解过程中求解条件的不确定性，不完备性造成的，使得求解的路径很多这就构成了一个图，我们说这个图就是状态空间。问题的求解实际上就是在这个图中找到一条路径可以从开始到结果。这个寻找的过程就是状态空间搜索。

的机器学习模型，在 2015 年 10 月 5：0 击败了欧洲围棋冠军樊麾后，2016 年 1 月，谷歌 DeepMind 团队在自然杂志 (Nature) 上发表封面论文称，他们研发出基于神经网络进行深度学习的人工智能围棋程序 AlphaGo，能够在极其复杂的围棋游戏中战胜专家级人类选手，2016 年 3 月，AlphaGo 又以 4：1 战胜世界围棋冠军李世石，在学术界产生了空前的影响。这标志着计算机博弈技术取得重大成功。

各种博弈问题，例如扑克，麻将，象棋以及其他各种棋类的博弈问题都翼以转化为博弈树的搜索问题，搜索算法的优劣极大的影响人工智能的博弈水平。在传统的搜索算法研究上，根据搜索的策略，可以分为启发式搜索（有信息的搜索）和随机搜索（无信息的搜索）；根据搜索范围，可以分为剪枝搜索和完全搜索。在博弈树的搜索算法研究中，研究重点一般是随机搜索和剪枝式搜索。

本文旨在介绍博弈树搜索技术中极大极小算法和剪枝搜索的基本思想，以及对其存在的一些问题进行讨论，并对此提出一些想法与思考，以供日后的学习和参考。

本文主要有以下几个部分组成，首先会介绍博弈树搜索的基本思想，以及对算法的问题及其中的关键技术做一个讨论，然后用一个实例来进行验证和分析，接着讨论一些相关的应用和发展，最后做一个总结展望。

二、 相关工作

1. 博弈树搜索

博弈论具有竞争或对抗性质的行为称为博弈行为。在这类行为中，参加斗争或竞争的各方各自具有不同的目标或利益。为了达到各自的目标和利益，各方必须考虑对手的各种可能的行动方案，并力图选取对自己最为有利或最为合理的方案。比如日常生活中的下棋，打牌等。博弈论就是研究博弈行为中斗争各方是否存在着最合理的行为方案，以及如何找到这个合理的行为方案的数学理论和方法。

在博弈过程中，任何一方都希望自己取得胜利。因此,当某一方当前有多个行动方案可供选择时，他总是挑选对自己最为有利而对对方最为不利

的那个行动方案。 假设在双人游戏中，先行的一方为 MAX，后行的一方为 MIN。此时,如果我们站在 MAX 方的立场上,主动权操在 MAX 方手里,他可以选择这个行动方案，或者选择另一个行动方案，完全由 MAX 方自己决定。当 MAX 方选取任一方案走了一步后,MIN 方也有若干个可供选择的行动方案，此时这些行动方案对 MAX 方来说则是不利的,因为这时主动权操在 MIN 方手里,这些可供选择的行动方案中的任何一个都可能被 MIN 方选中，MAX 方必须应付每一种情况的发生。

这样,如果站在某一方(如 MAX 方,即在 MAX 要取胜的意义下)，把上述博弈过程用图表示出来，则得到的是一棵“与或树”。 描述博弈过程的与或树称为博弈树,它有如下特点:

(1) 博弈的初始格局是初始节点。

(2) 在博弈树中，博弈双方的节点是逐层交替出现的。自己一方扩展的节点在此层,则下一层为对方的扩展节点。双方轮流地扩展节点。

(3) 所有自己一方获胜的终局都是搜索终点，相应的节点是可解节点；所有使对方获胜的终局都是不可解节点。

2. 极大极小算法（Minmax）

极大极小算法是博弈树理论中的经典算法，它由科劳德·香农(Claude E.Shannon)1950 年首次提出，核心的思想就是基于对方的最优选择来最小化对手的收益，通常使用递归方法来实现。在一般的搜索问题中，搜索的目标是找到一条从起始节点到目标节点的最优路径。搜索完毕，智能体只需沿着最优路径一直走下去，最终必然会到达目标节点。但是，而在博弈树搜索中，双方要根据当前的状态，选择 “‘最优” 的动作执行，期望能够到达自己获胜的目标状态 (节点)。在理想情况下，可以把获胜的棋局状态看作是目标状态。这意味着，在搜索时，必须执行一次完整的搜索，即从当前状态出发，考虑对手的所有可能应对，然后针对每一个可能的应对，生成新的状态。上述过程循环往复，一直进行到棋局终局时为止。最后，智能体将根据所有可能的终局状态，依照某种算法 (例如 Minimax 算法) 做出在当前状态下对自己最有利的落子选择。

但是，对于某些游戏，例如井字棋（TiaTacToc）这种简单的双人棋游戏，可以执行完全的搜索，然而，对于绝大多数棋类游戏来说，如国际象棋，中国象棋，围棋等，执行完全的搜索是不可能的。例如，国际象棋博弈树的平均分支因子大约是 35，每盘棋双方的平均行棋步数各为 50 步，那么树的节点个数大约为 35100 或者 10154 个（虽然不同的节点个数只有大约 1040 个)。下面给出几种棋种的博弈树复杂度，如图 2-1 所示。节点数目随搜索深度呈指数级增长，这对任何算法而言是一个巨大的灾难！

棋种	状态空间复杂度（10 为底数）	博弈树复杂度（10 为底数）
国际跳棋(100 格)	30	54
海克斯 (11×11)	57	98
国际象棋	46	123
中国象棋	48	150
亚马逊(10×10)	40	212
将棋	71	226
六子棋	172	140
19 路围棋	172	360

图 2-1

而且，还有一个重要因素，即博弈树搜索并不是一个单纯的目标搜索问题——必须考虑对手的策略。因此，“最优解” 并非真正的最优解，而是考虑双方策略时的最优解，它是双方妥协的结果。因此，在博弈树搜索中，双方将始终依据 “己方利益最大化，对方利益最小化” 这一原则，选择对己方最有利的动作来执行。由此，从当前状态出发，一直到棋局终局时，双方都将选择最有利己方的动作来执行，这将得到一个最优解。我们可以把在这种情况下得到的最优解称为 “对抗最优解” 或 “博弈最优解”。

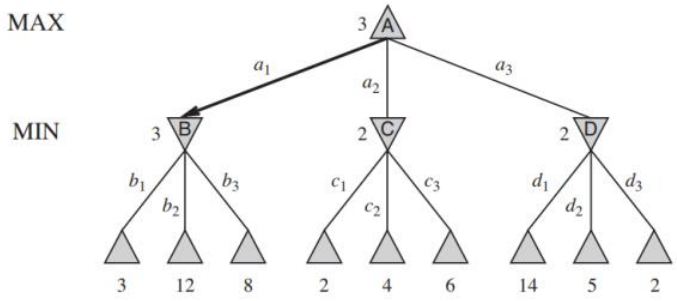

图 2-2

以图 2-2 为例，我们来解释一下简单的 2 层博弈树。

在该图中，根节点 MAX 的可能落子有 3 种选择： a1、 a2 和 a3。在此例中，博弈树的搜索目标是，确定 1 个最优落子。首先，从整体上来理解 MAX 方和 MIN 方如何按照最优策略行棋。在该图中，第 3 层都是叶子节点，并且每个叶子节点都有一个反映其价值的效用值 (这些值是从 MAX 方的角度定义的，值越大，对 MAX 方越有利)。例如，从左到右，第 1 个叶子节点的效用值为 3，第 2 个叶子节点的效用值为 12。显然， MAX 方更喜欢第 2 个叶子节点的棋局，而 MIN 方更喜欢第 1 个叶子节点的棋局。因此，MAX 方偏爱效用值大的节点， MIN 方偏爱效用值小的节点。

对于节点 B 而言，它的值将由 MIN 方确定： MIN 方将从 b1、 b2 和 b3 中选择 b1，并将 b1 的效用值 3 作为节点 B 的值。

对于节点 C 和节点 D，采用同样的方式来确定它们的值，得到的结果都是 2。

对于节点 A 而言，它的值将由 MAX 方确定： MAX 方将从 a1、 a2 和 a3 中选择 a1，并将 a1 的效用值 3 作为节点 A 的值。

在理解了算法的基本原理之后，我们来具体分析一下 Minimax 算法。它使用了简单的深度递归搜索技术来确定每个节点的值：它从根节点开始，一直前进到叶子节点，然后在递归回溯时，将叶子节点的效用值往上回传——对于 MAX 方，计算最大值，对于 MIN 方，计算最小值。在图 1-2 中，算法从节点 A 开始，一路深度递归到最左边的叶子节点，随后将值 3 回传给节点 B,然后又深度递归到 2 个叶子节点,值 12 被回传并与值 3 比较，不占优势， B 节点的值维持 3 不变,接着又深度递归到第 3 个叶子节点，值 8 被回传并与值 3 比较，也不占优势， B 节点的值仍然维持 3 不变。对于其余节点的分析，可依此类推。

但是作为一个静态的搜索算法，他的缺陷非常明显。如果博弈树的最大深度是 m，每个节点的合法落子有 b 个，那么 Minimax 算法的时间复杂度是 O(b^m)。该算法一次生成一条递归路径上每个节点的所有后继节点，它的空间复杂度是 O(bm)。在实际应用中，该算法的时间复杂度完全不实用。

3. Alpha-Beta 剪枝算法

Alpha-Beta 剪枝算法由极大极小算法改进而来，能够较好解决极大极小算法中的数据冗余问题，降低时间复杂度。该算法利用深度优先遍历的剪枝原理，使得博弈树不必完全展开，从而在改进极大极小算法的搜索效率。Alpha-Beta 算法涉及到 Alpha 和 Beta 两个参数，并将这两个参数作为极大极小值算法的参数值，Alpha 表示 MAX 方收益最低的情况，初始值社为负无穷。Beta 表示 MIN 方收益最低的情况，初始值设为正无穷。在 MAX 的局面中，如果 Alpha 值小于节点的值，则调整 Alpha 的值，使其逐渐递增；同理，在 MIN 的局面中，如果 Beta 值大于节点值，则调整 Beta 的值，使其逐渐递减。

随着递归的进行，Alpha 会越来越大，Beta 会越来越小，那么 Alpha-Beta 之间的范围就越来越小，则落在范围之外的节点越来越多，剪枝的效率就越来越高。下面给出剪枝过程如图 2-3 所示：

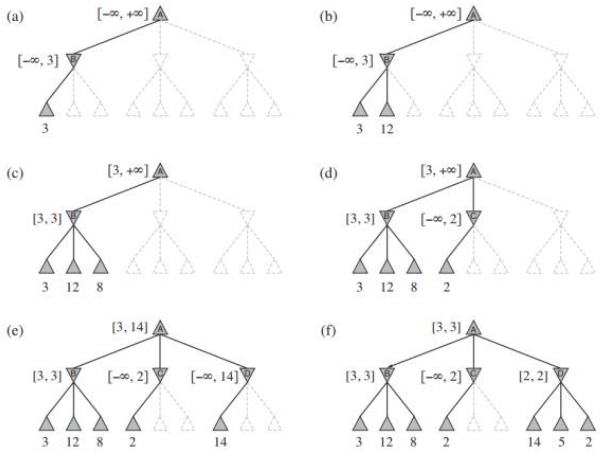

图 2-3：剪枝过程

在该图中，每一节点的左边标出了搜索进行时该节点的取值范围 4。在图 (a)中，A 为 MAX 方节点，下层还未有值回传，取值范围为初始范围 [-1; +1]， B 为 MIN 方节点，已得到一个回传值 3，取值范围为 [-1; 3]。在图 (b) 中， B 节点已得到另一个回传值 12，对 B 而言，效益比 3 要差，B 不会接受，故 B 节点的取值范围不会更新。在图 (c) 中， B 节点得到最后一个回传值 8，基于同样的原因， B 也不会接受，故 B 的最终取值为 3， B 将 3 回传给 A，于是 A 更新自己的取值范围为 [3; +1]。在图 (d)

中， C 为 MIN 方节点，已得到一个回传值 2，取值范围为 [-1; 2]。然而，A 的取值范围在 [3; +1]，而 C 将要回传给 A 的值只能在[-1; 2] 范围内，故 A 不会接受。在此情况下， C 节点的 2 个还未访问的子节点，已无扩展必要，这就出现了剪枝。对图 (e) 和图 (f) 可以做类似的分析。

4. Monte Carlo 树

基本的 Minimax 算法需要搜索一棵完整的博弈树，即树的叶子节点必须是终端棋局——具有明确结果 (胜负平) 的棋局。显然，对于那些具有高分支因子 (每步的可选落子数目较多) 的博弈树而言，这是非常不现实的——博弈树的节点数目随树的深度呈指数级增长，即满足 $O(b^m)$ 关系，其中 b 是分支因子， d 是树的深度。一般情况下，可以采取 2 种基本方法来减轻该问题。第 1 种方法是使用剪枝方法，将那些不满足最优解或目标解条件的分支去掉，例如 Alpha-Beta 算法。第 2 种方法是，使用启发函数 (Heuristic Function) 或评估函数 (Evaluation Function) 对非终局节点进行价值的估算。在这种情况下，算法只需要在博弈树上搜索到一定的深度即可——博弈树不需要完全扩展到终端棋局 (即叶子节点不是终端棋局)。如果评估函数能够准确地反映棋局的状态，那么这种方法将会非常有效。

MCTS 的基本思想非常简单——使用随机模拟的方式完成对节点价值的评估，进而为节点的选取决策提供统计依据。它的核心过程是节点的选择与评估。Monte Carlo(MC) 方法也被称为统计模拟方法，属于一种基于概率统计的数

值计算方法。20 世纪 40 年代，由 John von Neumann、Stanislaw Ulam 和 Nicholas Metropolis 在 Los Alamos 科学实验室为核武器计划工作时，发明了 MC 方法。为象征性地表明该方法的概率统计特征，特借用摩纳哥的赌城 Monte Carlo 为该方法命名。实际上，在此之前， MC 方法就已经存在。 1777 年，法国数学家 Buffon 就提出使用投针实验来计算圆周率 π，这被认为是 MC 方法的起源。MC 方法是一类满足某种特征的随机算法的统称。这类方法的特点是，使用随机采样得到近似解，随着采样的增多，近似解越

来越接近真实解。

MC 方法通常遵循如下的求解步骤：

1. 定义与输入数据相关的空间；

2. 在该空间中，算法采用某种特定的概率分布，随机地产生一个输入数据；

3. 对该输入数据执行一次确定的计算；

4. 重复第 2-3 步，将生成一系列结果，最后执行统计分析，以得到最终的结果；

下面通过一个简单的例子来说明 MC 方法的基本思想。假设需要计算一个不规则图形的面积。易知，图形的不规则程度与分析性计算方法（例如积分方法）的复杂程度是成正比的。针对此问题，可以使用 MC 方法简单地求解。首先，在不规则图形的外围放置一个正方形，使正方形刚好围住不规则图形。然后，向正方形内随机地投掷 N 个小石子，并数出落在不规则图形内小石子的数目 M。最后，可以利用公式 S MN 计算出不规则图形面积的近似值，其中 S 为正方形的面积。注意，在使用 MC 方法求解问题时，随机数生成器与模拟次数都是非常重要的[2]。

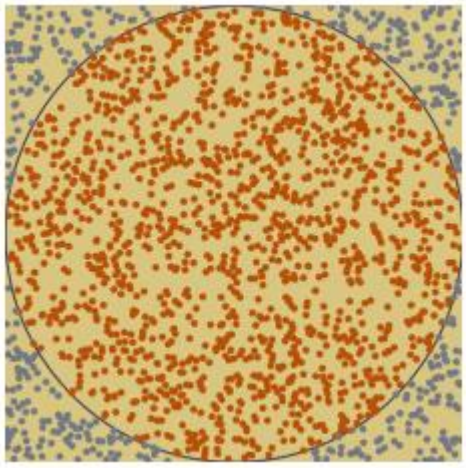

图 2-4

一个经典的例子是圆周率的计算，如图 2-4 所示，设圆的半径为 r,则正方形的边长为 2r,所以可以得出圆和正方形的面积之比为：

[2] 生成随机数越多，产生的结果越接近真实结果，模拟的效果更好。

$$\frac{S_{圆}}{S_{方}} = \frac{\pi * r^2}{(2r)^2} = \frac{\pi}{4}$$

MC 方法的应用实例非常多，其作用也相当明显。尤其是对一些采用传统方法难以解决的复杂问题， MC 方法能够很好地予以解决。

Monte Carlo 树总体上而言，就是在状态空间上随机采样，并动态的构建一棵搜索树，它具有以下的优点：

1.以随机采样的方式，对（非终端棋局）节点进行评估；

2.支持实时决策。但是，越多的运行时间，决策的效果越好；

3.无需或只需较少的领域知识；

4.可以有效地解决很难的决策问题，而这些问题不能被其它技术解决，例如计算机围棋；

5.易于实现，且方法具有普适性；

三、 算法描述（伪代码）和验证（结果与分析）

1．极大极小算法

```
def Minimax(node, depth, player):
    if depth == 0 or node is a terminal node:
        return the heuristic value of node
    if player == True:
        bestValue = -∞
        for each child of node:
            v = Minimax(child, depth-1, False)
            bestValue = max(bestValue, v)
        return bestValue
    else:
        bestValue = +∞
        for each child of node:
            v = Minimax(child, depth-1, True)
```

```
            bestValue = min(bestValue, v)
        return bestValue
```

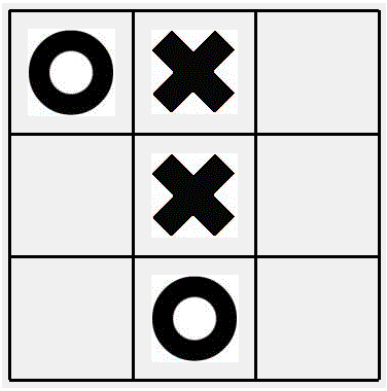

图 3-1

Minimax 算法的实现过程中，可以明显感觉到第一步的实现过程较长，因为他需要计算每一个节点的最佳值，遍历了所有的节点，所以花费了较长时间，但是当你下第二步棋时，很快他就得出了结果，因为他每一步运算出的值已经计算出来并且存储下来了，接下来每下一步，它直接就可以给出相应的方案。

2. Alpha-Beta 剪枝算法

```
def alpha-beta(node, depth, alpha, beta, player):
    if depth == 0 or node is a terminal node:
        return the heuristic value of node
    if player:
        v = -∞
        for each child of node:
            v = max(v, alpha-beta(child, depth-1, alpha, beta, False))
            alpha = max(alpha, v)
            if beta <= alpha:
                break #beta pruning
        return v
    else:
        v = +∞
```

```
        for each child of node:
            v = min(v, alpah-beta(child, depth-1, alpha, beta, True))
            beta = min(beta, v)
            if beta <= alpha:
                break #alpha pruning
        return v
```

该算法的执行过程与极大极小值算法非常相似，但是改进了极大极小值算法过程中遍历所有节点花费时间复杂度较多的缺点，对一些节点进行了剪枝操作，遍历的次数减少，提高了效率。

3. MCTS 算法

```
def MCTS(root):
    seed()
    decision_time = MAX_TIME
    for time in range(decision_time):
        path = [] #for backpropagation
        node = Select(root)
        simulation_node = Expand(node)
        simulation_result = Simulate(simulation_node)
        Backpropagate(simulation_result)
    retrun a child of root, with highest number of visits
    def Select(node):
        path.append(node)
        while node is nonterminal and node is fully expanded:
            node = a best UCT child of node
            path.append(node)
        return node
    def Expand(node):
        path.append(node)
        if node is nonterminal:
```

```
            child = a random unvisited child of node
            path.append(child)
            return child
        else:
            return node
    def Simulate(node):
        while node is nonterminal:
            node = a random child of node
        return result(node)
    def Backpropagate(result):
        for node in path:
            update node's statistics with result
```

MCTS 随机模拟下棋若干次，完成对每个棋子的评估，在非终局的棋局评估中有较大的优点，但是在井字棋实现中很难看出他的优点的表现，由于井字棋的棋路较少，没有明显体现出 MCTS 算法的优点，可能就是第一步棋计算的速度加快。

四、 算法应用

博弈树搜索被广泛的应用于各个领域之中。不仅仅是各种棋类活动，在生活中应用也非常广泛，像医疗，交通，运输等都博弈树技术的相关应用

1.航班服务保障问题

随着民航业的持续飞速发展,旅客和货物运输的业务量激增带来了航班起降架次的增长,机场面临巨大运行压力。如何在现有机场资源配置条件下保障航班准点率、提升服务质量成为了保证机场安全高效运行的关键步骤。为了保障航班过站的安全和高效,提出了一种基于博弈树理论的航班地面保障过程动态控制方法。首先将航班地面保障过程抽象为一个多主体多目标的优化问题,并针对该问题建立了多重约束的优化数学模型。然后根据模型建立了三方博弈树的动态控制方法,并根据 Alpha-Beta 剪枝算法对结构简化。达到提高保障资源的服务效率。

2．经济风险预测

P2P 网贷的出现为大量无法获得银行借款的小额资金需求者提供了有效的借款渠道，也有利于社会闲散资金的有效配置，进而完善金融资源配置体系。我国 P2P 网贷经历了最初的爆炸式发展后,近年开始进入矫正调整阶段。相对于中心化的传统金融服务而言，P2P 网贷这种脱离实体金融中介的纯线上信用审核模式会引发信息不对称问题，并由此产生逆向选择和道德风险。网贷市场充斥着违约行为，潜藏着巨大的信用风险，P2P 网贷平台爆雷事件频发，信用风险大有演化为社会风险之势。对网贷市场中各参与主体的网贷行为进行博弈理论分析，建立博弈树来分析 P2P 网贷过程中信用风险，对借款人和投资人的得益函数进行博弈仿真，进一步探究信用风险生成的影响因素，为有效控制 P2P 网贷行业信用风险提供依据。

3.船舶自动规避

船舶在海上的航线以及运行有时会产生一些重复航线或者相交航线，需要一些合理的规避策略来避开其他的船舶。针对常规水面船舶自动避碰决策难以实现的问题,基于船舶《国际海上避碰规则》，将博弈理论引入到船舶动态避碰系统中,通过设计船舶避碰博弈扩展树,建立船舶动态避碰博弈模型，结合避碰规则的约束条件,利用相关软件建立船舶自动避碰决策系统,实现两船自动避碰的目标。自动避碰决策系统对于两船间的避让能够采取合理的避让措施,效果良好,为后续研究多物标避让提供参考。

五、 结论与展望

因为自己算法掌握得不是很熟练，在算法的理解及验证（读代码）方面花费了大量的时间，所以本文并没有做过多深入的拓展，更多的是根据讲义内容，简要了解了极大极小算法及 Alpha-Beta 算法和 MCTX 算法的相关问题，并结合相关资料做了基本实现及验证，同时，通过了解一些具体的应用，加深了对 博弈树相关算法的认识和学习，在学习的过程中，我也认识到，该算法不仅在传统领域应用广泛，在目前比较火热的机器学习，人工智能领域也有着不可获缺的作用。因此，本文在扎实自己算法功底的基础上，希望能给自己以后机器学习提供一些创新思路，为以后的学习做一些参考。

六、 参考文献

[1] Stuart J. Russell， Peter Norvig，殷建平等译。《人工智能：一种现代的方

法》，第 3 版，清华大学出版社。

[2]Wikipedia: Game Theory...Accessed on 3 October 2020

[3]Wikipedia:Game Tree...Accessed on 11 May 2020

[4] 王亚杰,邱虹坤,吴燕燕,李飞,杨周凤.计算机博弈的研究与发展[J].智能系统学报,2016,11(06):788-798.

[5] 彭啟文,王以松,于小民,刘满义,徐方婧.基于手牌拆分的“斗地主”蒙特卡洛树搜索[J].南京师大学报(自然科学版),2019,42(03):107-114.

[6] 邢志伟,李彪,马浩然,戴铮.航班地面保障过程动态控制方法研究[J].计算机仿真,2020,37(07):78-83+212.

[7] 谭中明,刘媛媛.P2P 网贷市场主体信用风险生成机理的博弈仿真研究[J].财会月刊,2019(22):129-133.

[8] 孔祥生,卜仁祥,刘勇.基于扩展博弈理论的船舶自动避碰决策系统[J].计算机仿真,2019,36(05):154-158+268.

[9] 杜小勤.《人工智能》 课程系列:博弈树搜索技术,2018/10/23