第3章 对抗搜索与博弈论

南京信息工程大学
计算机学院

应龙

2024年秋季

主要内容

1. 对抗搜索 (Adversarial Search)

2. 博弈论相关概念 (Game Theory)

3. Alpha-Beta 剪枝算法 (Alpha-Beta Pruning)

4. 蒙特卡洛树搜索 (Monte Carlo Tree Search)

Bibliography:

[1] 吴飞 编著, “人工智能导论：模型与算法”, 高等教育出版社, 2020. Ch 3.3, Ch 3.4, Ch 8.1

[2] Stuart J. Russel, Peter Norvig, “Artificial Intelligence: A Modern Approach (4th Ed.)”, Pearson, 2020; 中译版 “人工智能 现代方法 (第4版) ”, 人民邮电出版社, 2022. Ch 6, Ch 17.1.2, Ch 17.2, Ch 17.3

主要内容

1. 对抗搜索 (Adversarial Search)

2. 博弈论相关概念 (Game Theory)

3. Alpha-Beta 剪枝算法 (Alpha-Beta Pruning)

4. 蒙特卡洛树搜索 (Monte Carlo Tree Search)

Bibliography:

[1] 吴飞 编著, “人工智能导论：模型与算法”, 高等教育出版社, 2020. Ch 3.3.1

[2] Stuart J. Russel, Peter Norvig, “Artificial Intelligence: A Modern Approach (4th Ed.)”, Pearson, 2020; 中译版 “人工智能 现代方法 (第4版) ”, 人民邮电出版社, 2022. Ch 6.1, Ch 6.2.1, Ch 6.2.2

博弈论 (Game Theory)

There are at least three stances we can take towards multi-agent environments.

- When there are a very large number of agents, is to consider them in the aggregate as economy, allowing us to do things like predict that increasing demand will cause prices to rise, without having to predict the action of any individual agent.
- Consider adversarial agents as just a part of environment that makes the environment nondeterministic, missing the idea that our adversaries are actively trying to defeat us.
- Explicitly model the adversarial agents with the techniques of adversarial game-tree search.

博弈论 (Game Theory)

竞争环境（competitive environments）中两个或多个 agents 的目标之间有冲突(conflicting goals)，产生了对抗搜索（adversarial search）问题。智能体(agents) 之间通过竞争实现各自相互冲突的利益。

➢ 古代博弈思想

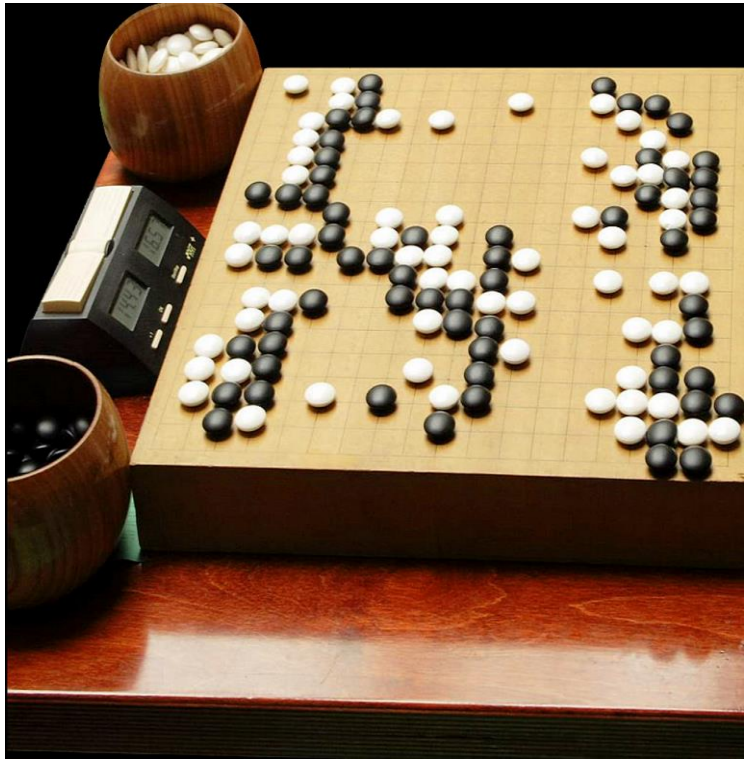
- 子曰：饱食终日，无所用心，难矣哉！不有博弈者乎？为之，犹贤乎已。

——《论语•阳货》

- 朱熹集注曰：“博，局戏；弈，围棋也。”；颜师古注：“博，六博；弈，围碁也。”
- 古语博弈所指下围棋，围棋之道蕴含古人谋划策略的智慧。
- 略观围棋，法于用兵，怯者无功，贪者先亡。

——《围棋赋》

- 《孙子兵法》等讲述兵书战法的古代典籍更是凸显了古人对策略的重视。



博弈论 (Game Theory)

➢ 田忌赛马

……齐将田忌善而客待之。忌数与齐诸公子驰逐重射。孙子见其马足不甚相远，马有上、中、下辈。于是孙子谓田忌曰："君弟重射，臣能令君胜。"田忌信然之，与王及诸公子逐射千金。及临质，孙子曰："今以君之下驷与彼上驷，取君上驷与彼中驷，取君中驷与彼下驷。"既驰三辈毕，而田忌一不胜而再胜，卒得王千金。于是忌进孙子于威王。威王问兵法，遂以为师。

——《史记•孙子吴起列传》

对局	齐王马	田忌马	结果	对局	齐王马	田忌马	结果
1	A +	A–	齐王胜	1	A +	C–	齐王胜
2	B +	B–	齐王胜	2	B +	A–	田忌胜
3	C +	C–	齐王胜	3	C +	B–	田忌胜

以己之长 攻彼之短

表8.1 齐威王与田忌进行赛马中所采取的不同对局

对于人工智能研究者来说，棋牌类游戏的简化性质是一个优点：游戏的状态很容易表示，而且 Agents 通常被限制在少数行动上，这些行动的效果是由精确的规则定义。专注于讨论博弈，如国际象棋、围棋和扑克，而不是处理真实世界中的混乱冲突。

体育博弈 (Physical games) 如台球 (croquet) 和冰球 (ice)， 则有复杂得多的描述， 有更大范围的可能行动， 并且有相当不精确的规则来定义行动的合法性。所以除了机器人足球，体育游戏目前并没有吸引人工智能界的很大兴趣。

对抗搜索 (Adversarial Search)

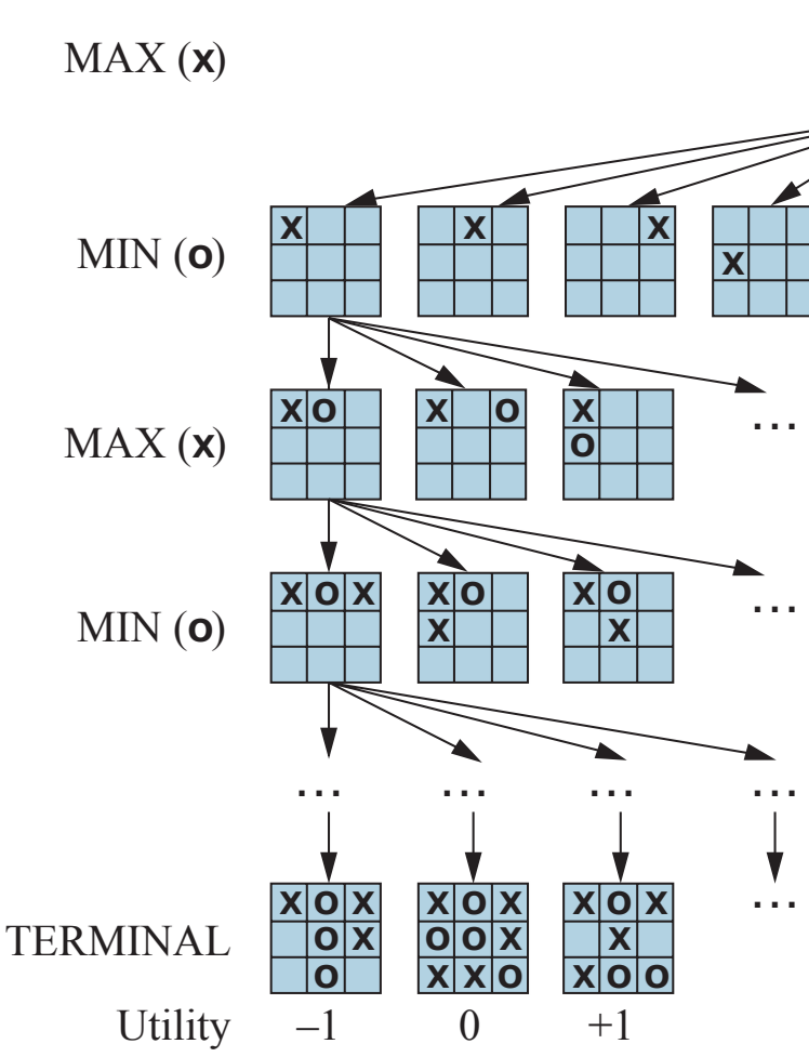
确定环境、全局可观测的两人轮流行动零和 (zero-sum) 博弈 (MAX and MIN, MAX先行) 可如下形式化描述，从而将其转换为对抗搜索 (Adversarial Search) 问题：

状态 s	状态s包括当前的游戏局面和当前行动的智能体。初始状态s_0包括初始游戏局面和初始行动的智能体。函数$player(s)$给出状态s下行动的智能体。
动作 $actions(s)$	给定状态s，动作指的是player(s)在当前局面下可以采取的操作a，记动作集合为actions(s)。
状态转移 $result(s,a)$	给定状态s和动作$a \in$ actions(s)，状态转移函数 result(s,a) 决定了在 s 状态采取 a 动作后所得后继状态
终局状态检测 $terminal\_test(s)$	终止状态检测函数terminal_test(s)用于测试游戏是否在状态s结束。
终局得分 $utility(s)$	终局得分 utility(s,p) 表示在终局状态 s 时玩家 p 的得分

注：所谓零和博弈是博弈论的一个概念，属非合作博弈。指参与博弈的各方，在严格竞争下，一方的收益必然意味着另一方的损失，博弈各方的收益和损失相加总和永远为“零”，双方不存在合作的可能。与“零和”对应，“合作博弈”可以“利己”不“损人”，通过谈判、合作达到皆大欢喜的结果。

对抗搜索 (Adversarial Search)

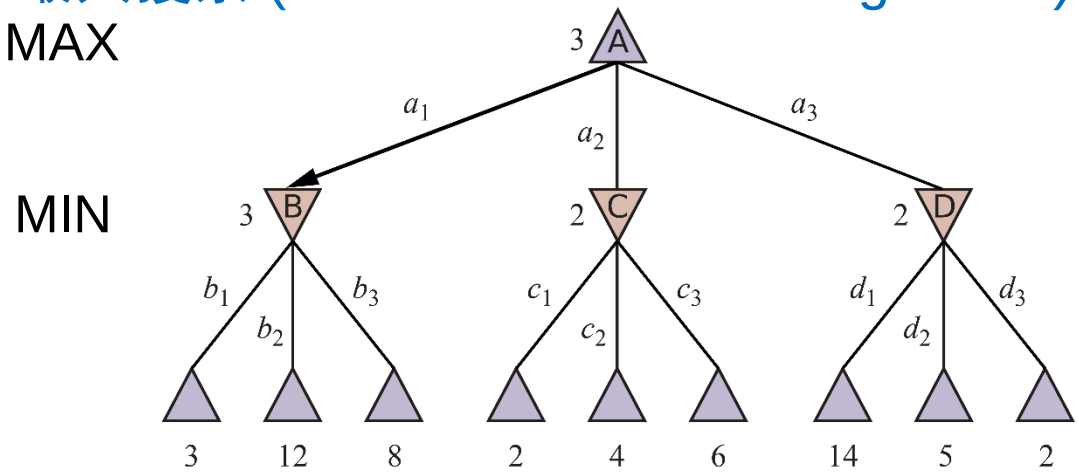
A (partial) game tree for the game of tic-tac-toe



- The initial state, ACTIONS function, and RESULT function define the state space graph
- Superimpose a search tree over part of that graph to determine what move to make
- Define the complete game tree as a search tree that follows every sequence of moves all the way to a terminal state

对抗搜索 (Adversarial Search)

- 最小最大搜索 (the minimax search algorithm)



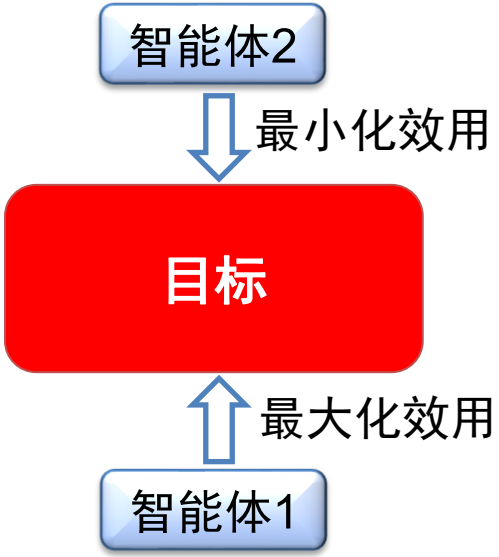


Figure 6.2 A two-ply game tree. The △ nodes are "MAX nodes," in which it is MAX's turn to move, and the ▽ nodes are "MIN nodes." The terminal nodes show the utility values for MAX; the other nodes are labeled with their minimax values. MAX's best move at the root is a_1, because it leads to the state with the highest minimax value, and MIN's best reply is b_1, because it leads to the state with the lowest minimax value.

假定对手拥有完全理性的决策能力

minimax(s) ← 博弈树当前局面 (状态) 优劣的评估值

递归结构

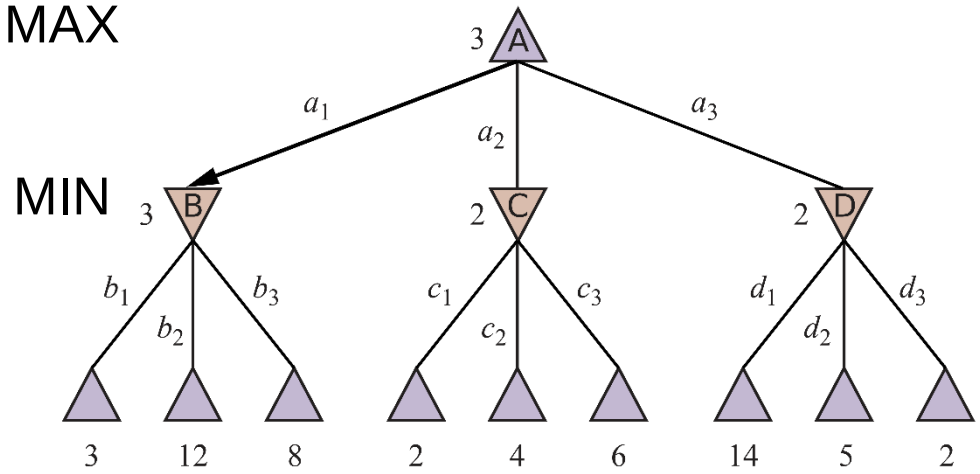
当前局面(状态)s采取行动a后导致的结果状态

$$= \begin{cases} \text{utility}(s), & \text{if terminal\_test(s)} \\ \max_{a \in \text{actions}(s)} \text{minimax}(\text{result}(s,a)), & \text{if player}(s) = MAX \\ \min_{a \in \text{actions}(s)} \text{minimax}(\text{result}(s,a)), & \text{if player}(s) = MIN \end{cases}$$

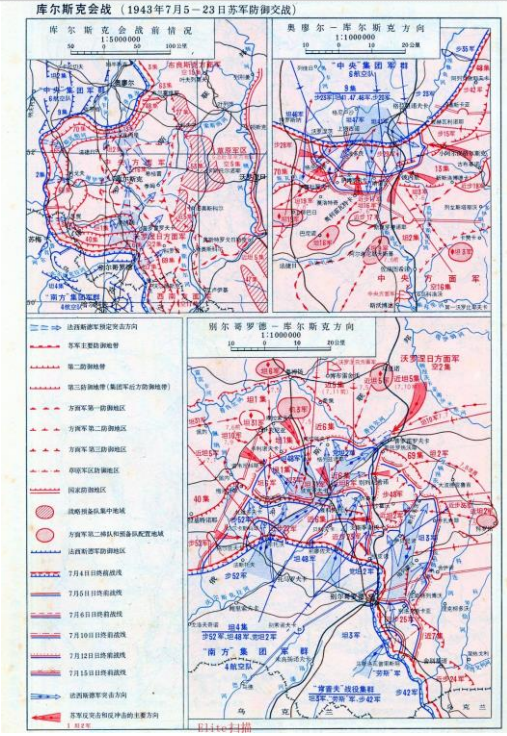
转移到其它局面(状态) 的评估值

对抗搜索 (Adversarial Search)

- 最小最大搜索 (the minimax search algorithm)



不要做敌人希望你做的事情，原因很简单，因为敌人希望你这样做。
— Napoleon Bonaparte



Minimax 算法是 pessimistic的

假定对手拥有完全理性的决策能力，每次决策时，希望找到对手让我方陷入最坏情况下各种策略中的最好策略，尽量降低损失。不是去找理论上的最优可能。
minmax 得名的由来 最小（坏）最大（好） 料敌从宽

广义化的与或搜索 (And-or search): For the first player to win a game, there must exist a winning move for all moves of the second player. This is represented in the and-or tree by using disjunction to represent the first player's alternative moves and using conjunction to represent all of the second player's moves.

对抗搜索 (Adversarial Search)

```
function MINIMAX-SEARCH(game, state) returns an action
    player ← game.TO-MOVE(state)
    value, move ← MAX-VALUE(game, state)
    return move

function MAX-VALUE(game, state) returns a (utility, move) pair
    if game.IS-TERMINAL(state) then return game.UTILITY(state, player), null
    v ← −∞
    for each a in game.ACTIONS(state) do
        v2, a2 ← MIN-VALUE(game, game.RESULT(state, a))
        if v2 > v then
            v, move ← v2, a
    return v, move

function MIN-VALUE(game, state) returns a (utility, move) pair
    if game.IS-TERMINAL(state) then return game.UTILITY(state, player), null
    v ← +∞
    for each a in game.ACTIONS(state) do
        v2, a2 ← MAX-VALUE(game, game.RESULT(state, a))
        if v2 < v then
            v, move ← v2, a
    return v, move
```

递归结构

计算导致最大效用的最终状态的行动，假设对手最小化效用。函数 MAX-VALUE 和 MIN-VALUE 经历整颗博弈树 (game tree) 一直到叶子节点，确定一个状态的倒推值 (backed-up value) 以及达到这个状态的行动。

最小最大算法的时间复杂度是$O(b^m)$，空间复杂度为$O(bm)$。

Optimal decisions in multiplayer games

许多流行的游戏都允许多个参与者。如何把最小最大思想推广到多人博弈。

- 首先需要用向量值替换每个结点上的单一值。例如，若博弈有3个参与者 A, B 和 C。则每个结点都与一个向量$< v_A, v_B, v_C >$相关联。对于终止状态，这个向量给出从每个人角度出发得到的状态效用值（在两人的零和博弈中，由于效用值总是正好相反，所以二维向量可以简化为单一值）。

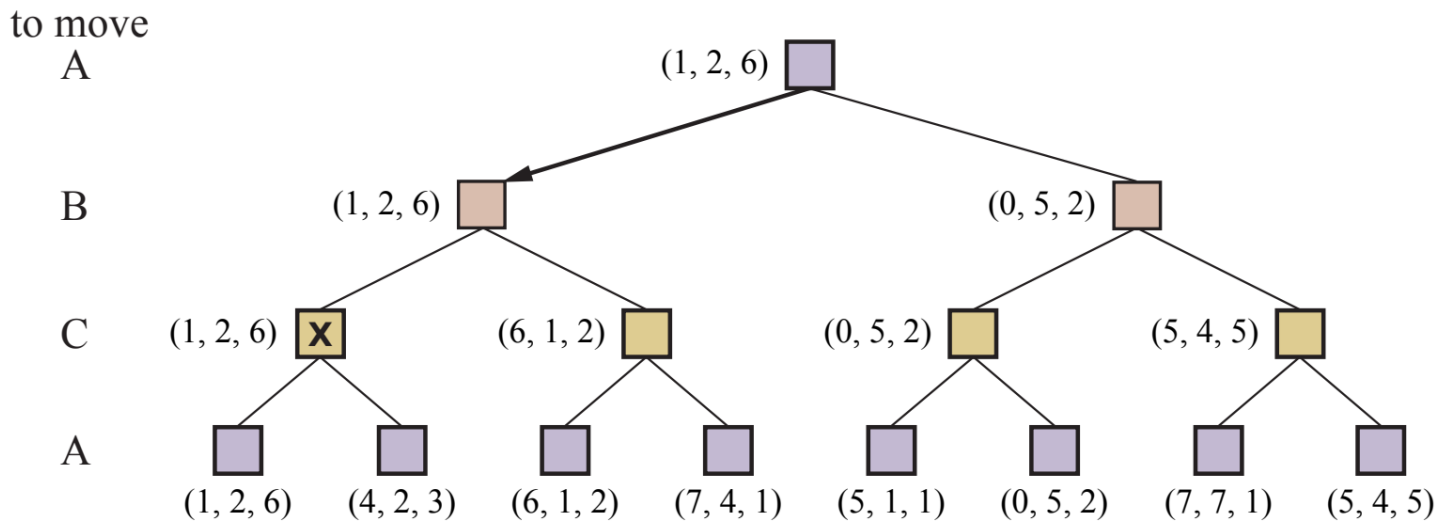


Figure 5.4 The first three ply of a game tree with three players (A, B, C). Each node is labeled with values from the viewpoint of each player. The best move is marked at the root.

- The backed-up value of a node n is always the utility vector of the successor state with the highest value for the player choosing at n.

Optimal decisions in multiplayer games

多人博弈通常会涉及在博弈参与者之间出现正式或者非正式的联盟 (formal or informal coalitions) 。随着博弈的进行联盟 (coalitions) 被建立或者破坏。

联盟是各参与者最优策略的自然结果。某些情况下，明确的联盟仅仅是使本将要发生的事情具体化。另外一些情况下，破坏联盟会损害社会声誉。参与者必须平衡破坏联盟带来的即刻有利因素和被认为不可信的长期不利因素。

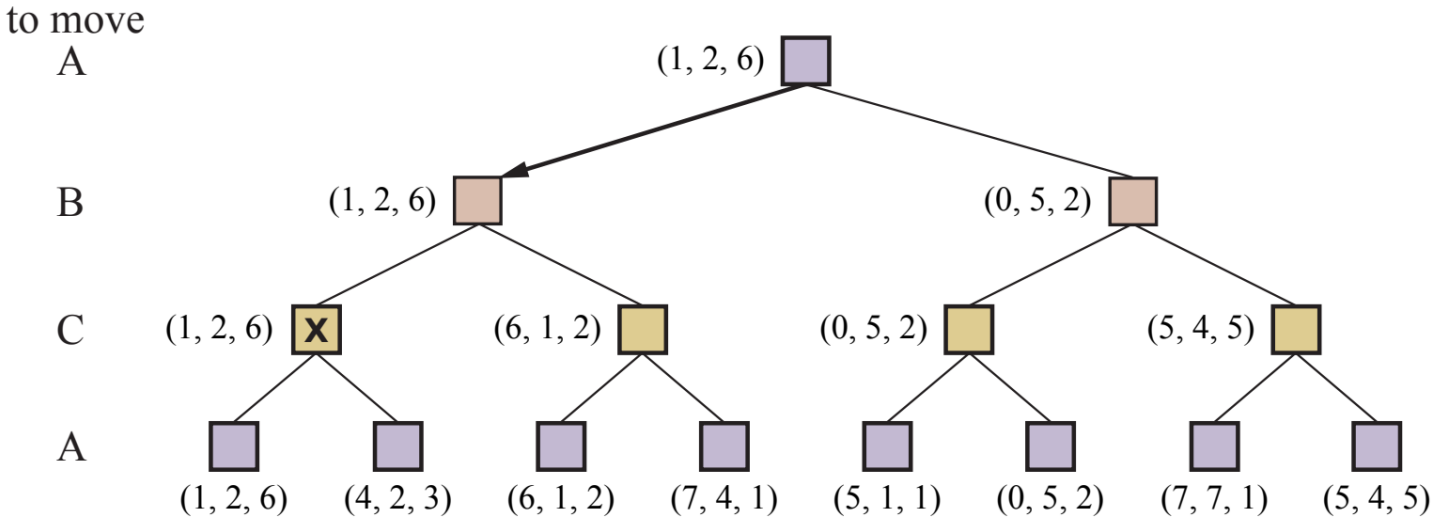


Figure 5.4 The first three ply of a game tree with three players (*A*, *B*, *C*). Each node is labeled with values from the viewpoint of each player. The best move is marked at the root.

如果博弈不是零和的 (zero-sum)，合作 (collaboration) 也可能发生在两个参与者的博弈中。双方会合作 (cooperate) 来达到共同的期望目标。

主要内容

1. 对抗搜索 (Adversarial Search)

2. 博弈论相关概念 (Game Theory)

3. Alpha-Beta 剪枝算法 (Alpha-Beta Pruning)

4. 蒙特卡洛树搜索 (Monte Carlo Tree Search)

Bibliography:

[1] 吴飞 编著, “人工智能导论：模型与算法”, 高等教育出版社, 2020. Ch 8.1

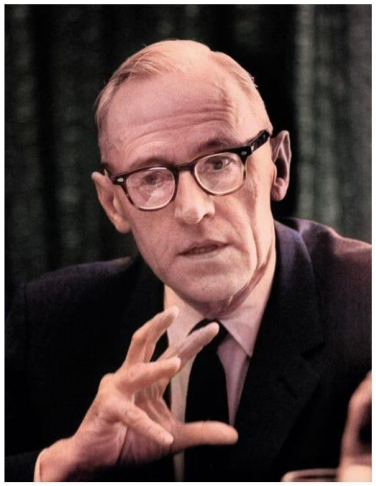
[2] Stuart J. Russel, Peter Norvig, “Artificial Intelligence: A Modern Approach (4th Ed.)”, Pearson, 2020; 中译版 “人工智能 现代方法 (第4版) ”, 人民邮电出版社, 2022. Ch 17.1.2, Ch 17.2, Ch 17.3

博弈论相关概念

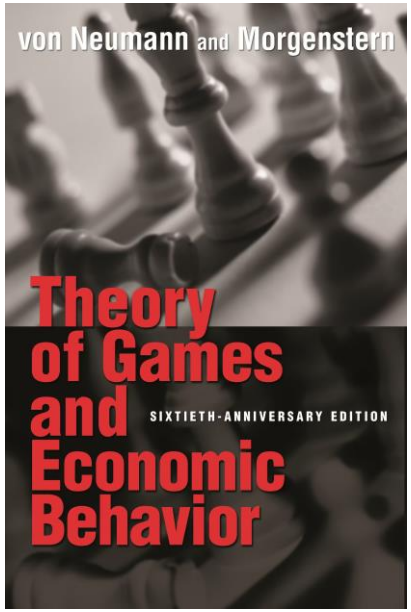
- 现代博弈论的建立

1944年冯·诺伊曼 (John von Neumann) 与奥斯卡·摩根斯特恩 (Oskar Morgenstern) 合著《博弈论与经济行为》，以数学形式来阐述博弈论及其应用，标志着现代系统博弈理论的初步形成，冯·诺伊曼被称为现代博弈论之父。

博弈论 (game theory)，又称对策论。



- 博弈行为：带有相互竞争性质的主体，为了达到各自目标和利益，采取的带有对抗性质的行为。
- 博弈论主要研究博弈行为中最优的对抗策略及其稳定局势，协助人们在一定规则范围内寻求最合理的行为方式。



✓ John von Neumann (1903-1957), Oskar Morgenstern (1902-1977), *Theory of Games and Economic Behavior*, Princeton University Press, 1944

博弈论相关概念

- 博弈的要素
  - 参与者(或称玩家) (player)：参与博弈的决策主体
  - 策略 (strategy)：参与者可以采取的行动方案，是一整套在采取行动之前就已经准备好的完整方案。
    - 某个参与者可采用策略的全体组合形成了策略集 (strategy set)
    - 所有参与者各自采取行动后形成的状态被称为局势 (outcome)
    - 如果参与者可以通过一定概率分布来选择若干个不同的策略，这样的策略称为混合策略 (mixed strategy)。 若参与者每次行动都选择某个确定的策略，这样的策略称为纯策略 (pure strategy)
  - 收益 (payoff)：各个参与者在不同局势下得到的利益

    混合策略意义下的收益应为期望收益 (expected payoff)
  - 规则 (rule)：对参与者行动的先后顺序、参与者获得信息多少等内容的规定

博弈论相关概念

囚徒困境 (prisoner's dilemma)

1950年，兰德公司的梅里尔·弗勒德和梅尔文·德雷希尔拟定了相关困境理论，后来美国普林斯顿大学数学家阿尔伯特·塔克以“囚徒方式”阐述：

- 警方逮捕了共同犯罪的甲、乙两人，由于警方没有掌握充分的证据，所以将两人分开审讯
- 若一人认罪并指证对方，而另一方保持沉默，则此人会被当即释放，沉默者会被判监禁10年
- 若两人都保持沉默，则根据已有的犯罪事实（无充分证据）两人各判半年
- 若两人都认罪并相互指证，则两人各判5年

- 参与者：甲、乙
- 规则：甲、乙两人分别决策，无法得知对方的选择
- 策略集：认罪、沉默（纯策略）
- 局势及对应收益（年）

甲沉默：-0.5	乙沉默：-0.5
甲沉默：-10	乙认罪：0
甲认罪：0	乙沉默：-10
甲认罪：-5	乙认罪：-5

	乙沉默 (合作)	乙认罪 (背叛)
甲沉默 (合作)	0.5年, 0.5年	10年, 0年
甲认罪 (背叛)	0年, 10年	5年, 5年

Normal-form Game 正规形式博弈

在囚徒困境中，群体最优解为两人同时沉默，但是两人实际倾向于选择同时认罪 均衡解 稳定解

占优策略 (Dominant Strategy)

个体理性有时会导致群体的非理性。

博弈论相关概念

- 研究范式

建模者对博弈参与者 (player) 定义可采取的策略集 (strategy sets) 和不同状况取得的收益 (博弈机制建模)，观察当参与者选择若干策略以最大化其收益时会产生什么结果。

完全理性：两害相权取其轻，两利相权取其重
向前展望，倒后推理 (backward induction)

- 应用

微观经济学, 计量经济学,（多主体环境）决策, 经济行为
It has applications in all fields of social science (经济学, 金融学, 国际关系, 政治学,军事战略), as well as in logic, systems science and computer science.

博弈论相关概念

博弈论经典教材（部分）：

[1] Robert Gibbons, “A Primer in Game Theory” (1992), Pearson Academic. 中译版“博弈论基础”. “Game Theory for Applied Economics” (1992), Princeton University Press.

[2] Osbourne and Rubinstein, “A Course in Game Theory” (1994), The MIT Press. 中译版“博弈论教程”.

[3] 张维迎编著, “博弈论与信息经济学” (2004), 上海人民出版社. 包含 Robert Gibbons 书中相当部分的内容.

[4] Roger B. Myerson, “Game Theory: Analysis of Conflict” (1997), Harvard University Press. 中译版“博弈论: 矛盾冲突分析” . 诺贝尔经济学奖获得者

进阶教材：

[5] Tim Roughgarden, “Twenty Lectures on Algorithmic Game Theory” (2016), Cambridge University Press. 中译版 “斯坦福算法博弈论二十讲”.

博弈论通俗读物：

[6] Avinash K. Dixit and Barry J. Nalebuff, “The Art of Strategy: A Game Theorist's Guide to Success in Business and Life” (2010). 中译版“妙趣横生博弈论”. “Thinking Strategically: The Competitive Edge in Business, Politics, and Everyday Life” (1993). “策略思维: 商界、政界及日常生活中的策略竞争”.

[7] Roger A. McCain, “A Nontechnical Introduction to the Analysis of Strategy” (3rd Ed, 2014). 中译版“博弈论: 策略分析入门” .

博弈论相关概念

- 博弈的分类

合作博弈与非合作博弈

- 非合作博弈 (non-cooperative game): 参与者在决策中都彼此独立。There is either no possibility to forge alliances or all agreements need to be self-enforcing {所有协议需要自我强化}, e.g. through credible threats.
- 合作博弈 (cooperative game): is a game with competition between groups of players ("coalitions") due to the possibility of external enforcement of cooperative behavior. A mandatory binding contract that can be reached by all parties on the basis of information exchange {在信息交换的基础上，各方可以达成的具有强制性约束力的合同}. Cooperative game provides a simplified approach that allows the analysis of the game at large without having to make any assumption about bargaining powers.

静态博弈与动态博弈

- 静态博弈 (static game): 所有参与者同时决策，或参与者互相不知道对方的决策。
- 动态博弈 (dynamic game): 参与者所采取行为的先后顺序由规则决定，且后行动者能够观测先行动者所采取的行为。　分阶段博弈，重复博弈

Normal-form 正规形式, 矩阵或张量; Extensive-form 扩展形式, 博弈树

博弈论相关概念

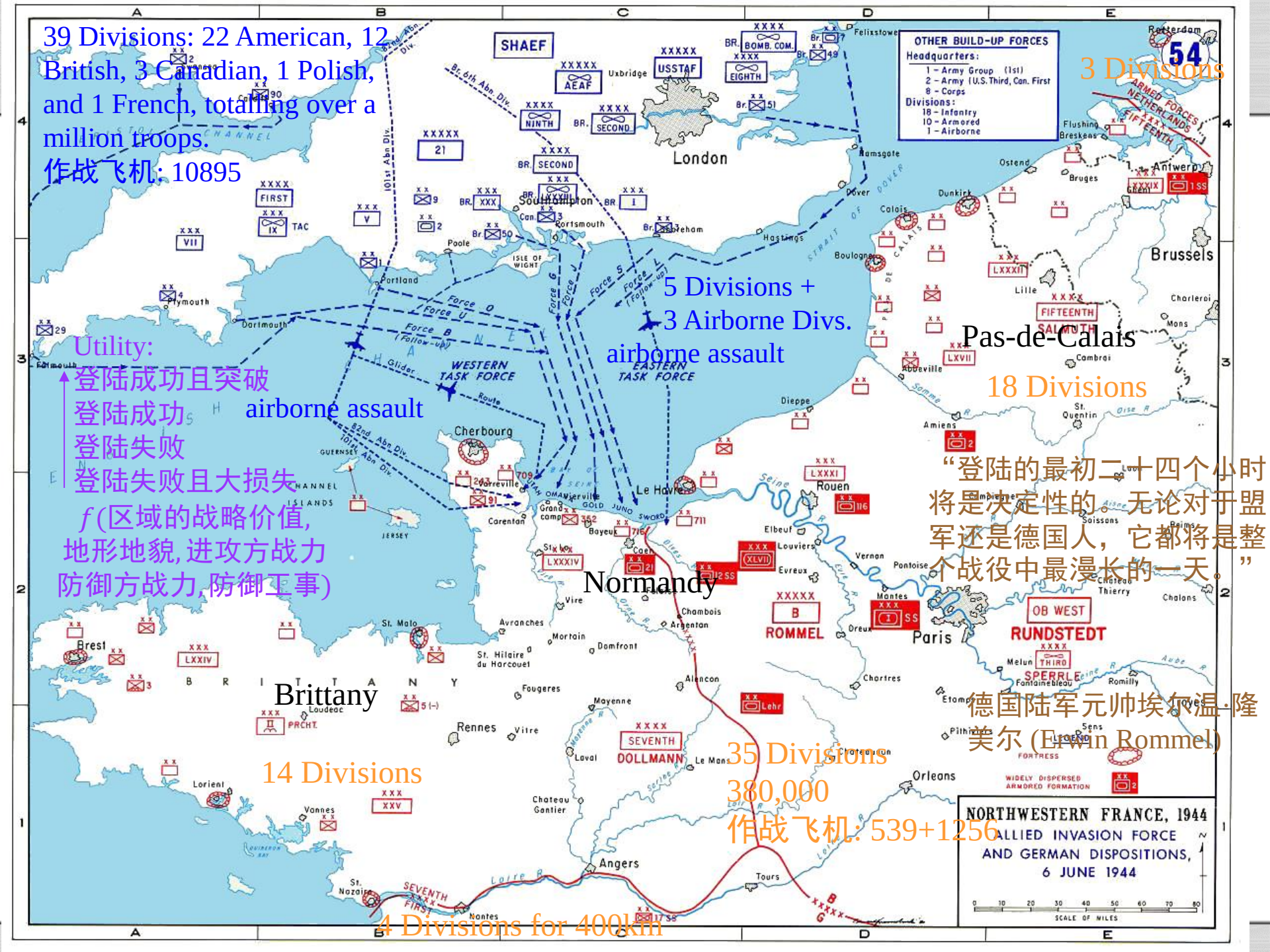
- 博弈的分类　　囚徒困境是一种非合作、完全不完美信息的静态博弈

根据博弈者掌握的信息

两个维度: complete (完全) vs. perfect (完美)

- 完全信息博弈 (complete information game)：所有参与者均可以获得关于其他参与者的知识，如：效用函数 (utility function), payoffs (收益), strategies (策略集) 和类型 (types) 等。　博弈的框架
- 不完全信息博弈 (incomplete information game)：参与者并不均掌握关于其它参与者的全部信息。一些参与者拥有私密信息。其他人在形成对这些玩家行为方式的期望时应该考虑这一事实。

  一个典型的例子是拍卖：每个参与者都知道自己的效用函数（对物品的估价），但不知道其他参与者的效用函数。
- 完美信息博弈 (perfect information game)：if each player, when making any decision, is perfectly informed of all the events that have previously occurred, including the “initialization event” of the game.　博弈的事件
- 不完美信息博弈 (imperfect information game)
- 非对称信息博弈 (asymmetric information game): Information asymmetry creates an imbalance of power in transactions, which can sometimes cause the transactions to be inefficient, causing market failure in the worst case, e.g. adverse selection, moral hazard, and monopolies of knowledge.


39 Divisions: 22 American, 12 British, 3 Canadian, 1 Polish, and 1 French, totalling over a million troops.
作战飞机: 10895
3 Divisions
5 Divisions +
3 Airborne Divs.
airborne assault
Pas-de-Calais
18 Divisions
Utility:
登陆成功且突破
登陆成功
登陆失败
登陆失败且大损失
airborne assault
f(区域的战略价值, 地形地貌, 进攻方战力 防御方战力, 防御工事)
“登陆的最初二十四个小时将是决定性的。无论对于盟军还是德国人，它都将是整个战役中最漫长的一天。”
Normandy
德国陆军元帅埃尔温·隆美尔 (Erwin Rommel)
Brittany
14 Divisions
35 Divisions
380,000
作战飞机: 539+1256
4 Divisions for 400km
SHAEF
London
Paris
Cherbourg
Le Havre
Rouen
Brussels
Antwerp
ROMMEL
RUNDSTEDT
DOLLMANN
SEVENTH
FIFTEENTH
OTHER BUILD-UP FORCES
Headquarters:
1 – Army Group (1st)
2 – Army (U.S. Third, Can. First
8 – Corps
Divisions:
18 – Infantry
10 – Armored
1 – Airborne
NORTHWESTERN FRANCE, 1944
ALLIED INVASION FORCE
AND GERMAN DISPOSITIONS,
6 JUNE 1944
SCALE OF MILES

博弈论相关概念

- 博弈的稳定局势即为纳什均衡 (Nash equilibrium)：指的是参与者所作出的这样一种策略组合，在该策略组合上，任何参与者单独改变策略都不会得到好处。即，如果在一个策略组合上，当所有其他人都不改变策略时，没有人会改变自己的策略，则该策略组合就是一个纳什均衡。

策略组 $\sigma^* = \{\sigma_1^*, \sigma_2^*, \cdots, \sigma_N^*\}$，对任意参与者 $i = 1, \cdots, N$:

$$u_i(\sigma^*) \geq \max_{\sigma_i' \in \Sigma_i} u_i(\sigma_1^*, \sigma_2^*, \cdots \sigma_i', \cdots, \sigma_N^*)$$

$$u_i(\sigma^*, \sigma_{-i}^*) \geq u_i(\sigma_i, \sigma_{-i}^*), \text{ for all } \sigma_i \in \Sigma_i$$

纳什均衡的本质: 不后悔

- Nash存在性定理 (Nash's existence theorem): If mixed strategies are allowed, then every game with a finite number of players in which each player can choose from finitely many pure strategies has at least one Nash equilibrium, which might be a pure strategy for each player or might be a probability distribution over strategies for each player.

John Nash, Non-Cooperative Games. *The Annals of Mathematics*. 54, 2 (1951), 286.

NON-COOPERATIVE GAMES

JOHN NASH

(Received October 11, 1950)

Introduction

Von Neumann and Morgenstern have developed a very fruitful theory of two-person zero-sum games in their book *Theory of Games and Economic Behavior*. This book also contains a theory of n-person games of a type which we would call cooperative. This theory is based on an analysis of the interrelationships of the various coalitions which can be formed by the players of the game.

Our theory, in contradistinction, is based on the *absence* of coalitions in that it is assumed that each participant acts independently, without collaboration or communication with any of the others.

The notion of an *equilibrium point* is the basic ingredient in our theory. This notion yields a generalization of the concept of the solution of a two-person zero-sum game. It turns out that the set of equilibrium points of a two-person zero-sum game is simply the set of all pairs of opposing "good strategies."

In the immediately following sections we shall define equilibrium points and prove that a finite non-cooperative game always has at least one equilibrium point. We shall also introduce the notions of solvability and strong solvability of a non-cooperative game and prove a theorem on the geometrical structure of the set of equilibrium points of a solvable game.

As an example of the application of our theory we include a solution of a simplified three person poker game.

Formal Definitions and Terminology

In this section we define the basic concepts of this paper and set up standard terminology and notation. Important definitions will be preceded by a subtitle indicating the concept defined. The non-cooperative idea will be implicit, rather than explicit, below.

Finite Game:

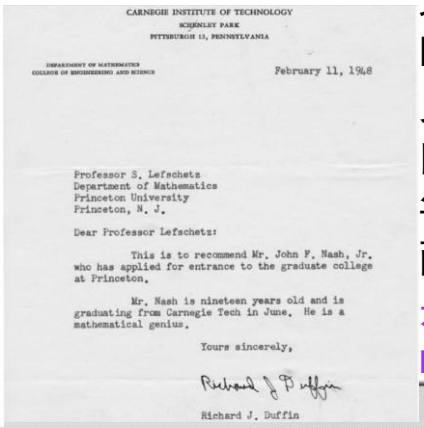
For us an *n-person game* will be a set of n *players*, or *positions*, each with an associated finite set of *pure strategies*; and corresponding to each player, i, a *payoff function*, p_i, which maps the set of all n-tuples of pure strategies into the real numbers. When we use the term *n-tuple* we shall always mean a set of n items, with each item associated with a different player.

Mixed Strategy, s_i:

A *mixed strategy* of player i will be a collection of non-negative numbers which have unit sum and are in one to one correspondence with his pure strategies. We write $s_i = \sum_{\alpha=1}^{m_i} c_{i\alpha} s_{i\alpha}$ with $c_{i\alpha} \geq 0$ and $\sum_{\alpha=1}^{m_i} c_{i\alpha} = 1$ to represent such a mixed strategy, where the $s_{i\alpha}$'s are the pure strategies of player i. We regard the s_i's as points in a simplex whose vertices are the $s_{i\alpha}$'s. This simplex may be re-

296

This content downloaded from 39.174.145.187 on Tue, 18 Dec 2018 09:46:31 UTC
All use subject to https://about.jstor.org/terms



CARNEGIE INSTITUTE OF TECHNOLOGY
SCHENLEY PARK
PITTSBURGH 13, PENNSYLVANIA

DEPARTMENT OF MATHEMATICS
COLLEGE OF ENGINEERING AND SCIENCE

February 11, 1948

Professor S. Lefschetz
Department of Mathematics
Princeton University
Princeton, N. J.

Dear Professor Lefschetz:

This is to recommend Mr. John F. Nash, Jr. who has applied for entrance to the graduate college at Princeton.

Mr. Nash is nineteen years old and is graduating from Carnegie Tech in June. He is a mathematical genius.

Yours sincerely,

Richard J. Duffin

Richard J. Duffin

理查德·达芬 (Richard Duffin) 教授给纳什写了史上短短数行推荐信，内容如下：纳什先生今年19岁，6月即将从卡耐基技术学院毕业。他是个数学天才 (he is a mathematical genius)

博弈论相关概念

完全信息静态博弈 (complete information static game)	纳什均衡 (Nash equilibrium)
博弈规则的扩展　拍卖、投票、信号传递、谈判、联盟…	
完全信息动态博弈 (complete information dynamic game)	子博弈精炼纳什均衡 (subgame perfect Nash equilibrium)
非完全信息静态博弈 (incomplete information static game)	贝叶斯纳什均衡 (Bayesian Nash equilibrium)
非完全信息动态博弈 (incomplete information dynamic game)	精炼贝叶斯均衡 (Perfect Bayesian equilibrium)

博弈论相关概念

➢ 混合策略下纳什均衡的例子

例子：公司的雇主是否检查工作与雇员是否偷懒

V是雇员的贡献，W是雇员的工资，H是雇员的付出，C是检查的成本，F是雇主发现雇员偷懒对雇员的惩罚（没收抵押金）。

假定 $H < W < V, W > C$

表8.4 雇主-雇员每次采取对应行动后的收益

		雇员	
		偷懒	不偷懒
雇主	检查	$-C+F, -F$	$V-W-C, W-H$
	不检查	$-W, W$	$V-W, W-H$

- 参与者：

  雇员、雇主

- 规则：

  雇员与雇主两人分别决策，事先无法得知对方的选择

- 混合策略集：

  雇员：偷懒、不偷懒

  雇主：检查、不检查

- 局势及对应收益：

  雇主采取检查策略时雇员工作与偷懒对应的结果

  雇主采取不检查策略时雇员工作与偷懒对应的结果

博弈论相关概念

V是雇员的贡献，W是雇员的工资，H是雇员的付出，C是检查的成本，F是雇主发现雇员偷懒对雇员的惩罚（没收抵押金）。

假定$H < W < V, W > C$

表8.4 雇主-雇员每次采取对应行动后的收益

		雇员	
		偷懒	不偷懒
雇主	检查	$-C+F, -F$	$V-W-C, W-H$
	不检查	$-W, W$	$V-W, W-H$

若雇主检查的概率为 α,
雇员偷懒的概率为 β.

表8.5 雇主-雇员之间博弈的期望收益

参与者	采取策略	期望收益
雇主	检查	$T_1 = \beta(-C+F)+(1-\beta)(V-W-C)$
	不检查	$T_2 = -\beta W+(1-\beta)(V-W)$
雇员	偷懒	$T_3 = -\alpha F+(1-\alpha)W$
	不偷懒	$T_4 = \alpha(W-H)+(1-\alpha)(W-H)=W-H$

博弈论相关概念

参与者	采取策略	期望收益
雇主	检查	$T_1=\beta(-C+F)+(1-\beta)(V-W-C)$
	不检查	$T_2=-\beta W+(1-\beta)(V-W)$
雇员	偷懒	$T_3=-\alpha F+(1-\alpha)W$
	不偷懒	$T_4=\alpha(W-H)+(1-\alpha)(W-H)=W-H$

V是雇员的贡献，W是雇员的工资，H是雇员的付出，C是检查的成本，F是雇主发现雇员偷懒对雇员的惩罚（没收抵押金）。

混合策略纳什均衡：博弈过程中，参与者通过概率形式随机从可选策略中选择一个策略而达到的纳什均衡被称为混合策略纳什均衡。

纳什均衡：其他参与者策略不变的情况下，某个参与者单独采取其他策略都不会使得收益增加⇔各参与者收益函数的极值点

于是有：

$$\nabla_\alpha(\alpha T_1+(1-\alpha)T_2)=0\Rightarrow T_1=T_2$$
$$\nabla_\beta(\beta T_3+(1-\beta)T_4)=0\Rightarrow T_3=T_4$$

- 性质：无论雇主如何检查，雇主的收益都一样；无论雇员如何偷懒，雇员的收益都一样。

博弈论相关概念

参与者	采取策略	期望收益
雇主	检查	$T_1=\beta(-C+F)+(1-\beta)(V-W-C)$
	不检查	$T_2=-\beta W+(1-\beta)(V-W)$
雇员	偷懒	$T_3=-\alpha F+(1-\alpha)W$
	不偷懒	$T_4=\alpha(W-H)+(1-\alpha)(W-H)=W-H$

V是雇员的贡献，W是雇员的工资，H是雇员的付出，C是检查的成本，F是雇主发现雇员偷懒对雇员的惩罚（没收抵押金）。

- 在纳什均衡下，由于$T_3 = T_4$，可知雇主采取检查策略的概率（雇主趋向于用这个概率去检查）：

$$\alpha = \frac{H}{W+F}$$

- 在纳什均衡下，由于$T_1 = T_2$，可知雇员采取偷懒策略的概率（雇员趋向于用这个概率去偷懒）：

$$\beta = \frac{C}{W+F}$$

- 在检查概率为α之下，雇主的收益：

$$T_1 = T_2 = V - W - \frac{CV}{W+F}$$

- 对上式中W求导，则当$W = \sqrt{CV} - F$时，雇主的收益最大，其值为：

$$T_{max} = V - 2\sqrt{CV} + F$$

博弈论相关概念

- 策梅洛定理 (Zermelo’s theorem)：对于任意一个有限的两名参与者完美信息 (perfect information) 动态零和博弈，若其中没有随机成分，一定存在先手必胜策略或后手必胜策略或双方保平策略。必为三种情况之一。

  strictly dominant strategy

策梅洛定理仅对两人博弈有效，如果博弈者超过两人，如对于三人博弈，策梅洛定理无法保证三方中一定有一方获胜、其他两方必败或者三方和局的策略。

博弈论相关概念

策梅洛定理 (Zermelo's theorem) 的简要证明：

假设双人博弈过程结果只有胜、负、和三种结果，博弈过程最长持续N步，下面通过数学归纳法来证明策梅洛定理：

1) 当$N = 1$时，先手玩家必选择自己收益最大的策略，若存在获胜的行动，则存在先手必胜策略。若不存在使先手玩家获胜的行动，则先手玩家会选择平局的行动，于是存在双方保平策略。若仅存在后手玩家获胜的行动，则存在后手必胜策略。
2) 假设当$N = 1,2 \ldots, k$时，均存在先手必胜策略、后手必胜策略或双方保平策略。
3) 当$N = k + 1$时，将先手玩家行动一次后的局势视为开启了一个新的博弈。在这开启的新博弈中，先手玩家为原博弈的后手玩家，新博弈中的后手玩家为原博弈的先手，应用2)中的假设，新开启的博弈必存在先手必胜策略或后手必胜策略或双方保平策略，则原博弈亦必存在后手必胜策略或先手必胜策略或双方保平策略。
4) 综上所述，对于任意一个有限步的双人完全信息零和动态博弈，一定存在先手必胜策略或后手必胜策略或双方保平策略。

人工智能与博弈论

- 人工智能与博弈论：博弈策略求解
  - 动机
    - 博弈论提供了许多问题的数学模型
    - 纳什存在性定理确定了博弈过程问题存在解
    - 人工智能的方法可用来求解均衡局面或者最优策略
  - 主要问题

    如何高效求解博弈参与者的策略以及博弈的均衡局势？
  - 应用领域
    - 大规模搜索空间的问题求解：围棋
    - 非完美信息博弈问题求解：德州扑克
    - 网络对战游戏智能：Dota、星球大战
    - 动态博弈的均衡解：厂家竞争、信息安全

[1] Matej Moravcik, Martin Schmid, Karel Ha, Milan Hladikand Stephen J. Gaukrodger. "*Refining Subgames in Large Imperfect Information Games.*" AAAI Conference on Artificial Intelligence (2016).

[2] Noam Brown, and Tuomas Sandholm. "*Safe and nested subgame solving for imperfect-information games.*" Advances in neural information processing systems 30 (2017).

博弈论与计算机科学

博弈论与计算机科学的交叉领域非常多

- 理论计算机科学：算法博弈论
- 人工智能：多智能体系统、AI游戏参与者、人机交互、机器学习、广告推荐
- 互联网：互联网经济、共享经济
- 分布式系统：区块链

人工智能与博弈论相互结合，形成了两个主要研究方向

- 博弈策略的求解
- 博弈规则的设计

Breaking the Prisoner's Dilemma

囚徒困境所反映出的深刻问题是，个体理性有时会导致群体的非理性。个体会被迫做出一些决策，无法达成自己在其中也可以收获更多的更好的整体利益。
内卷的实质就是囚徒困境 (prisoner's dilemma)。　　黑暗森林

解决囚徒困境的本质是要改变参与者的 payoff。重点是信任问题：参与者承诺，并彼此信任对方会信守承诺。手段：奖励，报复　　让阳光照进黑暗森林

- 在信息交换的基础上，达成具有强制力的契约、合同等。　合作博弈
- 把共有知识 (mutual knowledge) (各自知道) 变成共同知识 (common knowledge)。 (每个人知道, 每个人知道每个人知道, …)　教育
- 教育可以改变参与者的效用函数。
- 不确定期限的重复博弈　存在合作的纳什均衡解。

注: 确定期限的重复博弈与单次静态博弈结果一样

好心 (Nice): 不会在对手之前背叛;

报复 (Retaliating): "以德报怨，何以报德？" "以牙还牙"

原谅 (Forgiving): 尽管参与者会报复，但如果对方不继续背叛，他们将再次合作。

不嫉妒 (Non-envious): 不全力追求比其它人得分更多

--by Robert Axelrod, The Evolution of Cooperation (1984)

- 个体有限理性 (Bounded rationality)　　囚徒困境，个体理性的局限

主要内容

1. 对抗搜索 (Adversarial Search)

2. 博弈论相关概念 (Game Theory)

3. Alpha-Beta 剪枝算法 (Alpha-Beta Pruning)

4. 蒙特卡洛树搜索 (Monte Carlo Tree Search)

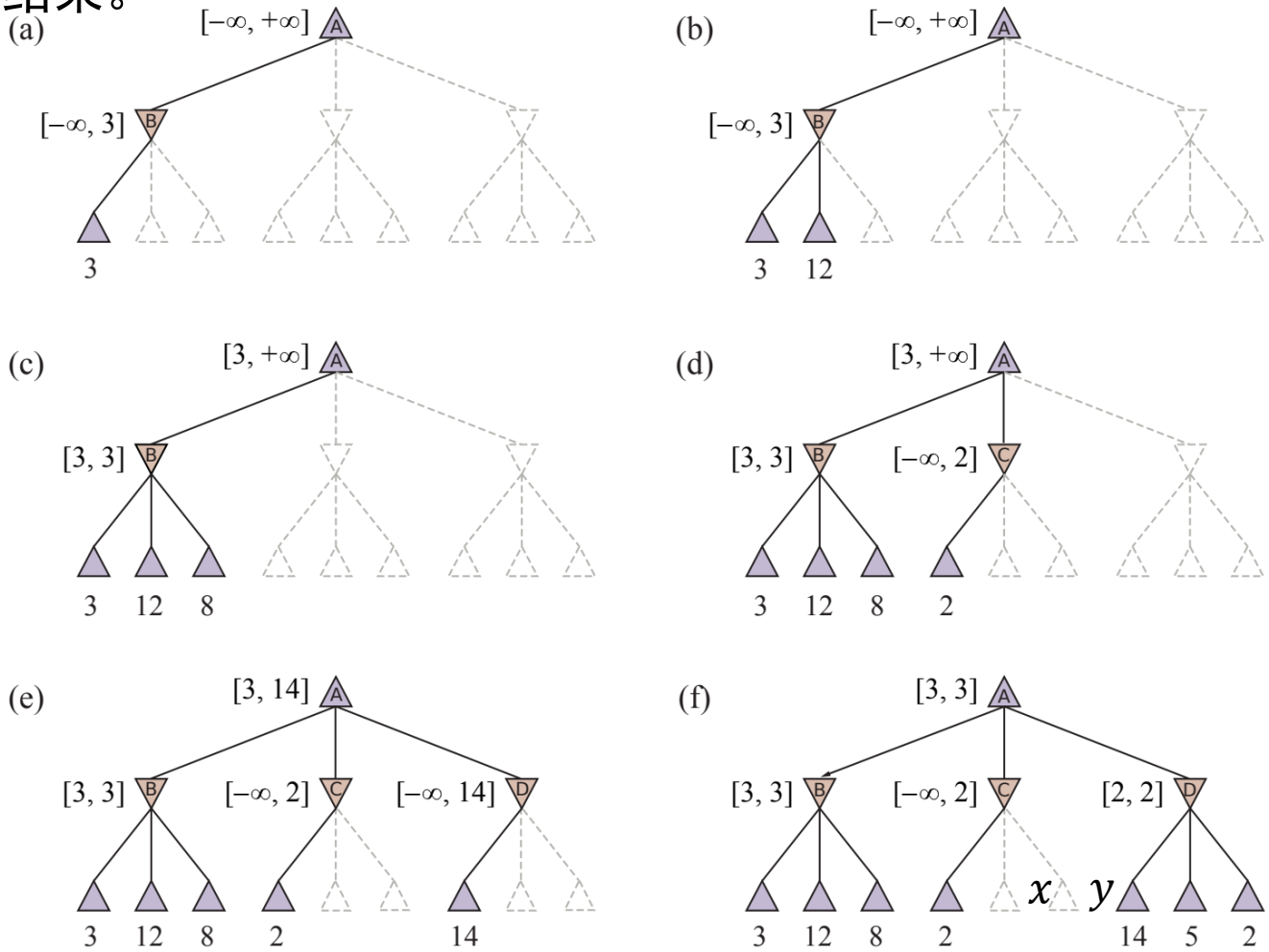
Bibliography:

[1] 吴飞 编著, “人工智能导论：模型与算法”, 高等教育出版社, 2020. Ch 3.3

[2] Stuart J. Russel, Peter Norvig, “Artificial Intelligence: A Modern Approach (4th Ed.)”, Pearson, 2020; 中译版 “人工智能 现代方法 (第4版) ”, 人民邮电出版社, 2022. Ch 6

Alpha-Beta 剪枝算法

博弈状态的数量随着树的深度指数增长。minimax 搜索是否存在不影响最终结果的搜索分枝，可以减少被搜索的节点数目，得到与原 minimax 决策同样的搜索结果。



图中MIN选手所在的节点C下属分支4和6与根节点最终优化决策的取值无关，可不被访问。

$$\text{minimax}(C) = \max(\min(3,12,8), \min(2,x,y), \min(14,5,2)) = \max(3, \min(2,x,y), 1)$$

Alpha-Beta 剪枝算法

最小最大搜索是深度优先的，一个时刻只需考虑树中一条路径上的节点。

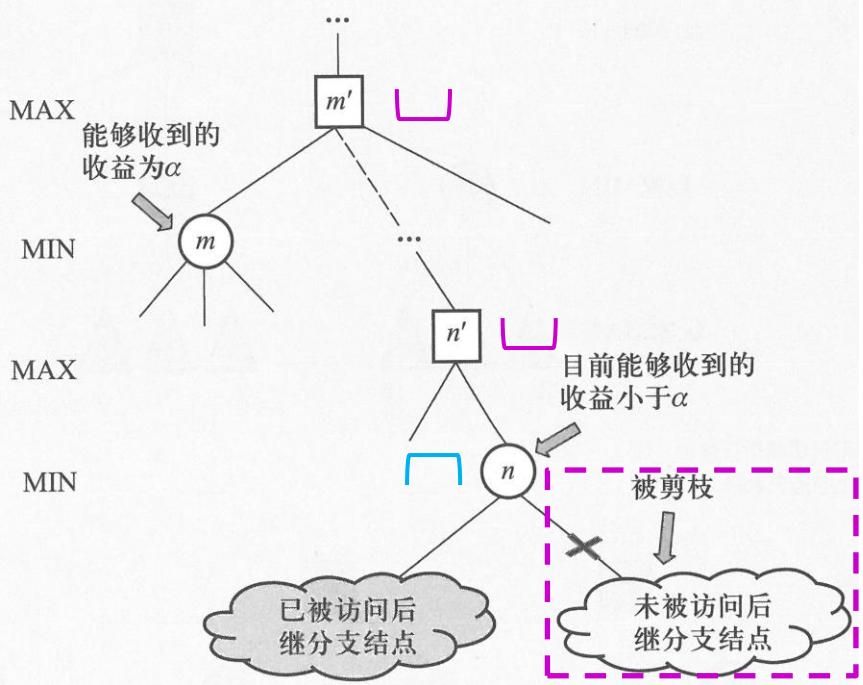
α = 当前为止路径上已经找到的 MAX 选择点最佳的回传值 (backed-up value)

β = 目前为止路径上已经找到的 MIN 选择点最佳的回传值 (backed-up value)

MAX节点当前最佳回传值

基于MIN 节点反馈收益进行剪枝

(Alpha 剪枝)



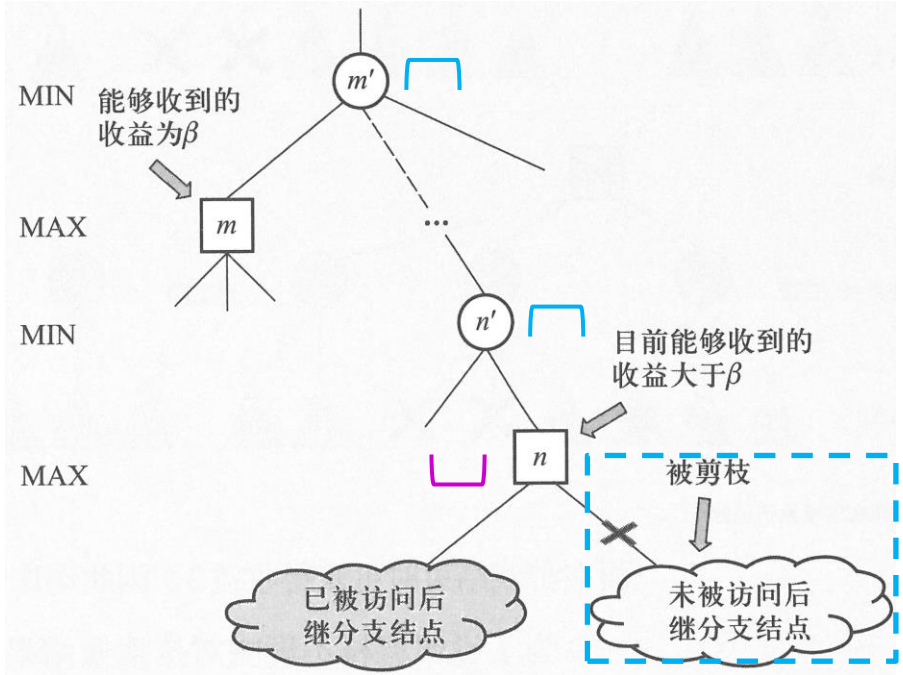
Alpha 剪枝

影响兄弟(sibling)节点路径的MIN节点

MIN节点当前最佳回传值

基于MAX 节点反馈收益进行剪枝

(Beta 剪枝)



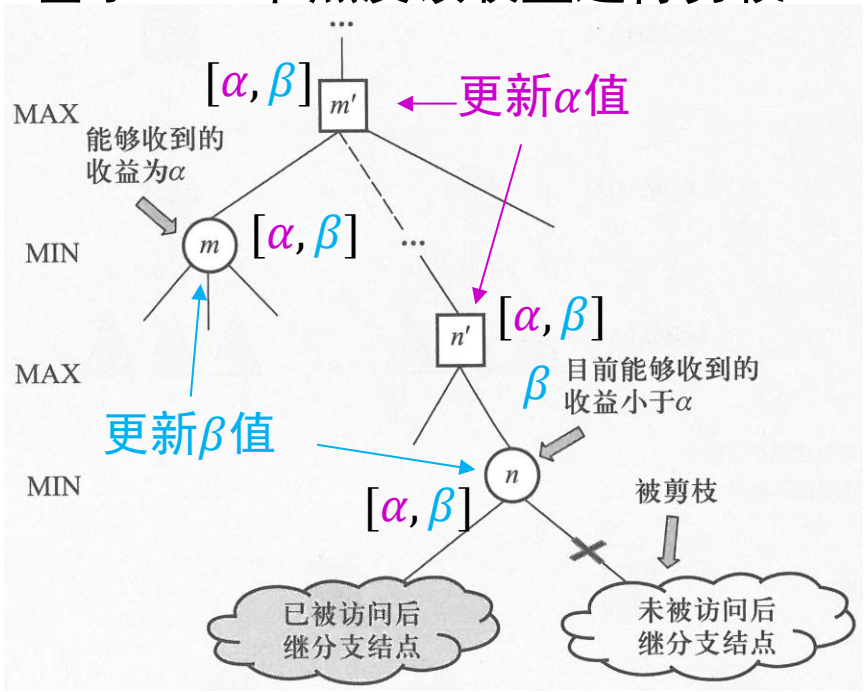
Beta 剪枝

影响兄弟(sibling)节点路径的MAX节点

Alpha-Beta 剪枝算法

MAX节点当前最佳回传值

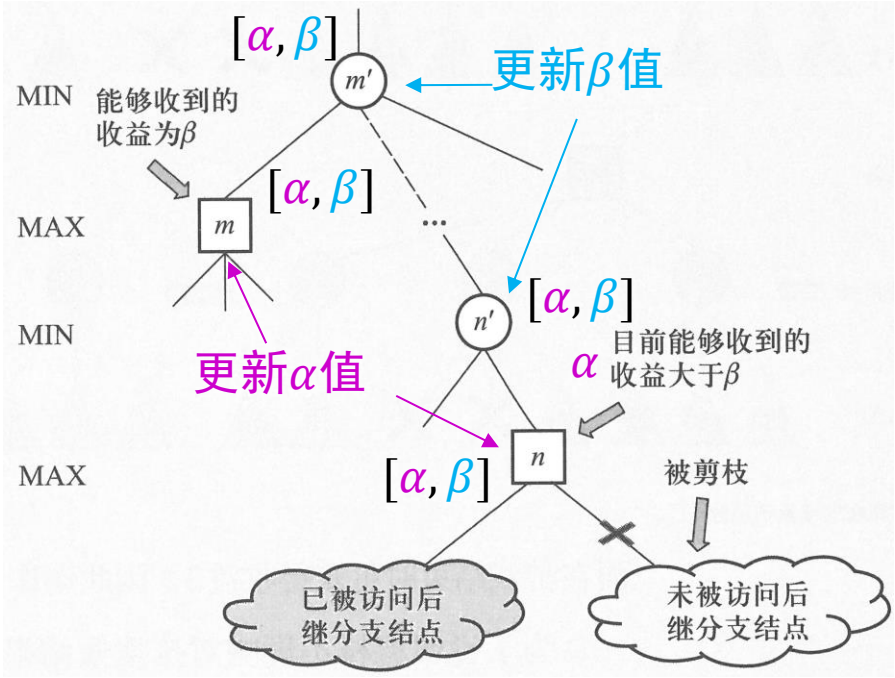
基于MIN 节点反馈收益进行剪枝



影响兄弟(sibling)节点路径的MIN节点

MIN节点当前最佳回传值

基于MAX 节点反馈收益进行剪枝



影响兄弟(sibling)节点路径的MAX节点

- 结点 m 通过其父母结点为MIN 层结点 n 提供了一个下界α，对于 n' 的MAX 层结点，这个下界仍然是有意义的。无论一个结点位于MIN 层还是MAX 层，都可以找到一个下界和一个上界，即可以为每个结点设置一个α值和一个β值，来判断该结点及其后继结点是否可被剪枝。
- 继承父结点的 α 值和 β 值，再按照一定的规则进行更新。

Alpha-Beta 剪枝算法

Alpha-Beta 剪枝的一般性原理如下：考虑树上某层的一个节点n，博弈者通过选择一组行动到达该节点。如果在同一层有一个更好的选择，或者在树上任何更高的位置有更好的选择，博弈者就不会采取到达结点n的行动。因此，一旦通过检查节点 n 的一些后代得到足够的信息可以得出上述的结论，就可以将该节点剪掉。算法可以被应用于任意深度的树，通常更有可能剪掉整棵子树，而不仅是叶子。

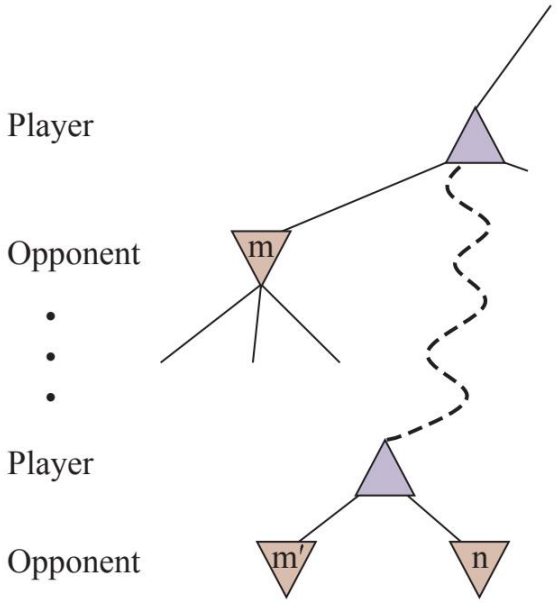


Figure 5.6 The general case for alpha-beta pruning. If m or m' is better than n for Player, we will never get to n in play.

最小最大搜索是深度优先的，一个时刻只需考虑树中一条路径上的节点。

α = 当前为止路径上已经找到的 MAX 选择点最佳的回传值 (backed-up value)

β = 目前为止路径上已经找到的 MIN 选择点最佳的回传值 (backed-up value)

Alpha-Beta 剪枝算法

函数：AlphaBetaDecision( )

输入：当前的盘面状态s
输出：玩家MAX行动下，当前最优动作a^*

$v, a^* \leftarrow \text{MaxValue}(s, -\infty, +\infty)$

函数：MaxValue( )

输入：当前的盘面状态s，当前结点的下界α和上界β
输出：玩家MAX行动下，当前状态的得分$v = \text{minimax}(s, \text{MAX})$和最优动作$a^*$
前置条件：$\alpha \leqslant \beta$

```
1  if terminal_test(s) then return utility(s), null
2  v ← −∞
3  a* ← null
4  foreach a ∈ action(s) do
5      v′, a′ ← MinValue(result(s, a), α, β)
6      if v′>v then
7          v ← v′
8          a* ← a
9      end
10     α ← max(α, v)        更新α值
11     if α ⩾ β then return v, a*    Alpha-Beta 剪枝
12 end
```

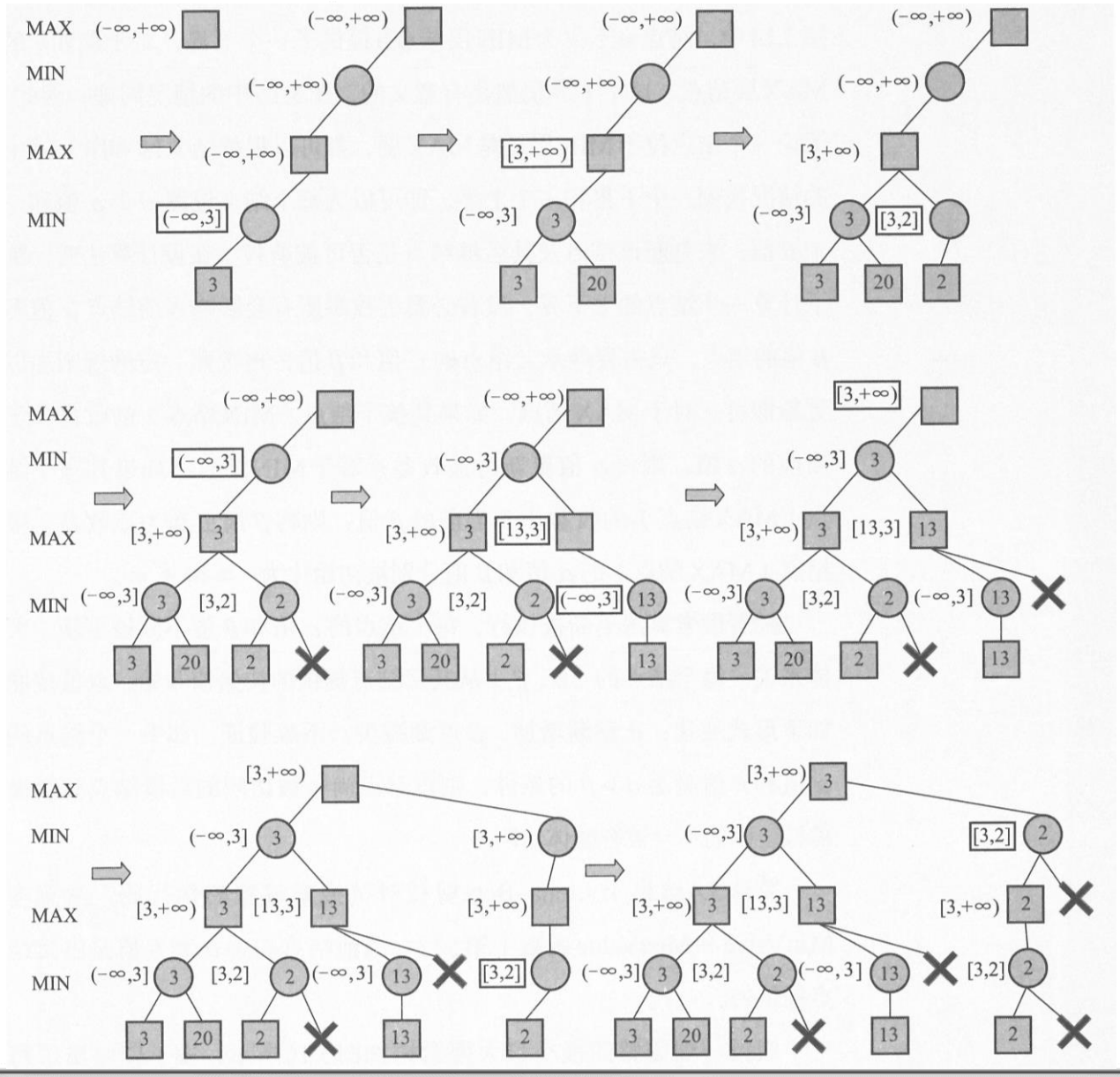
函数：MinValue( )

输入：当前的盘面状态s，当前结点的下界α和上界β
输出：玩家MIN行动下，当前状态的得分$v = \text{minimax}(s, \text{MIN})$和最优动作$a^*$
前置条件：$\alpha \leqslant \beta$

```
1  if terminal_test(s) then return utility(s), null
2  v ← +∞
3  a* ← null
4  foreach a ∈ action(s) do
5      v′, a′ ← MaxValue(result(s, a), α, β)
6      if v′<v then
7          v ← v′
8          a* ← a
9      end
10     β ← min(β, v)        更新β值
11     if α ⩾ β then return v, a*    Alpha-Beta 剪枝
12 end
```

注：第6行和第11行中对分数相同情况的处理方法与正文中不同，此处只保留一个最优解，而放弃其它分数相同的解。教材原文如此。

Alpha-Beta 剪枝算法

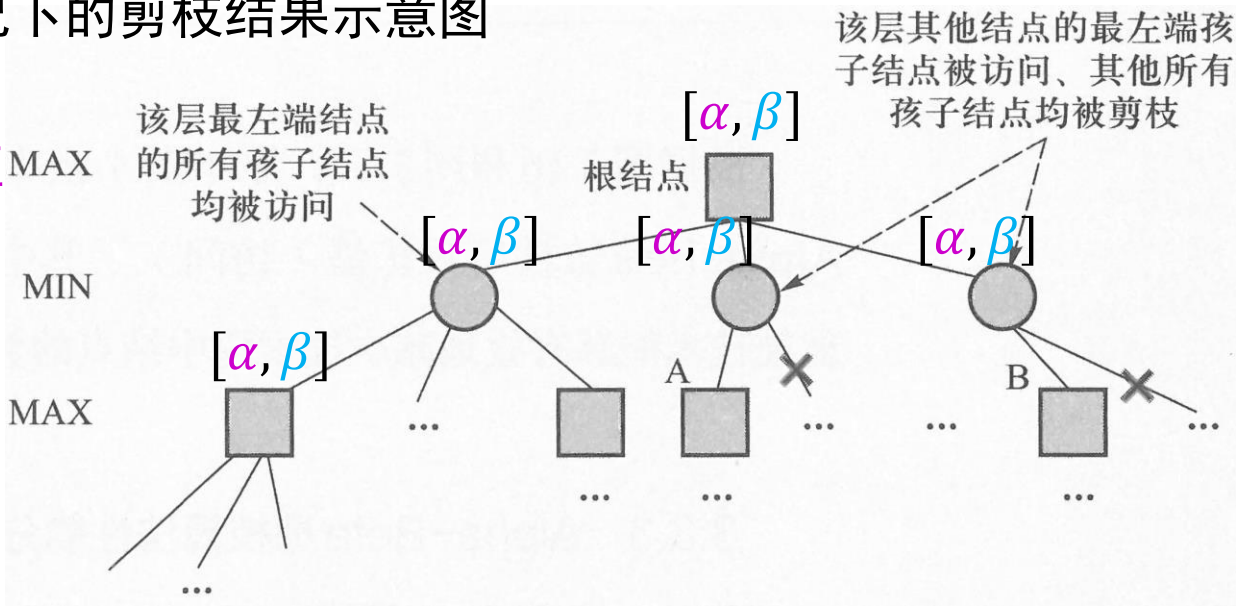


Alpha-beta 搜索进行时不断更新α和β的值， 并且当某个结点的值分别比目前的MAX的α或者MIN的β值更差的时候，剪裁此结点剩下的分支（即终止递归调用）。

Alpha-Beta 剪枝算法

- Alpha-Beta 剪枝算法性能分析

最好情况下的剪枝结果示意图



为了出现如图的剪枝效率最高情况，对于 MIN层 (更新β值) 的结点而言，它的第一个孩子结点的收益必须小于该结点的初始下界α; 对于 MAX层 (更新α值) 的结点而言，它的第一个孩子结点的收益必须大于该结点的初始上界β。

- 似乎除了每层最左端的结点，其余结点都可以只扩展一个孩子结点，算法在最优情况下达到线性(关于m) 的复杂度？当然这是不可能的。

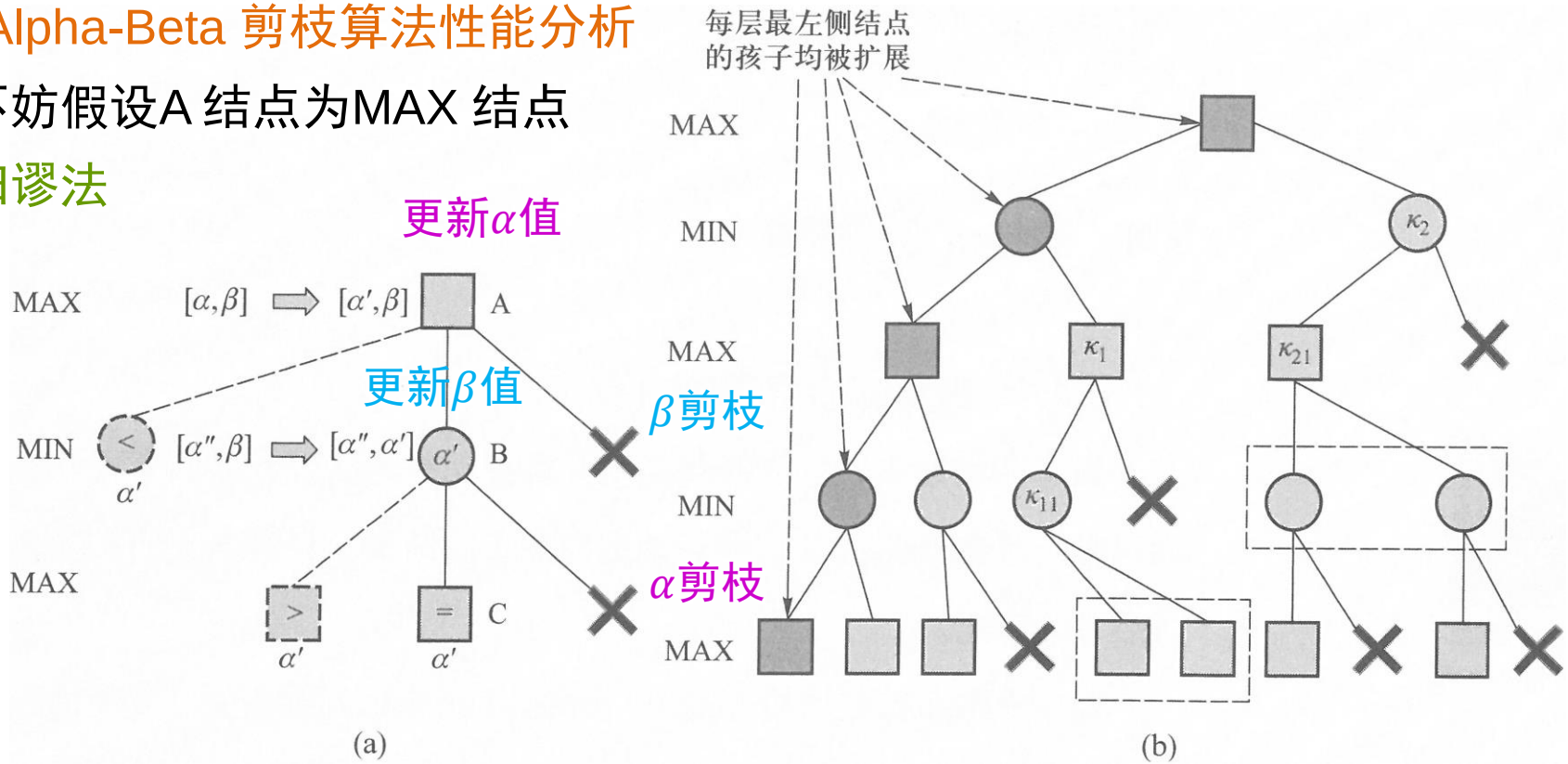
如果一个结点导致了其兄弟结点被剪枝，它所有的孩子结点必然被扩展。

Alpha-Beta 剪枝算法

- Alpha-Beta 剪枝算法性能分析

不妨假设A 结点为MAX 结点

归谬法



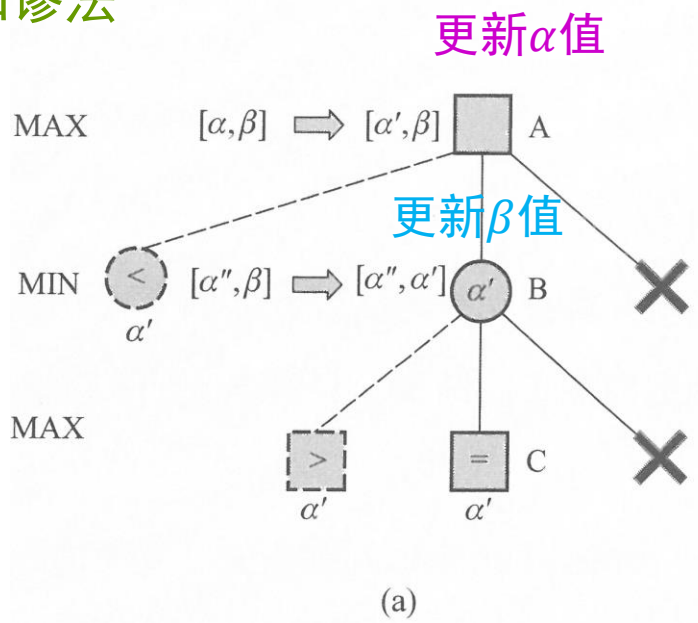
- ①假设结点A扩展孩子结点初始Alpha 和Beta值为$[\alpha,\beta]$，如果A的孩子结点B导致了它右端的兄弟结点被剪枝，结点B必然是A 已扩展孩子结点中收益分数最大的结点；
- ②同时根据剪枝的发生条件，可知B的收益分数α'必然满足$\alpha' > \beta$ 。
- ③假设扩展结点B时的上下界为$[\alpha'',\beta]$，其中α''来自结点B被扩展时结点A的下界。
- ④根据假设，B结点至少有一个孩子结点(C结点)被扩展，因此$\alpha'' \le \beta$ ，同时又根据②$\alpha' > \beta$，因此推出$\alpha'' < \alpha'$。

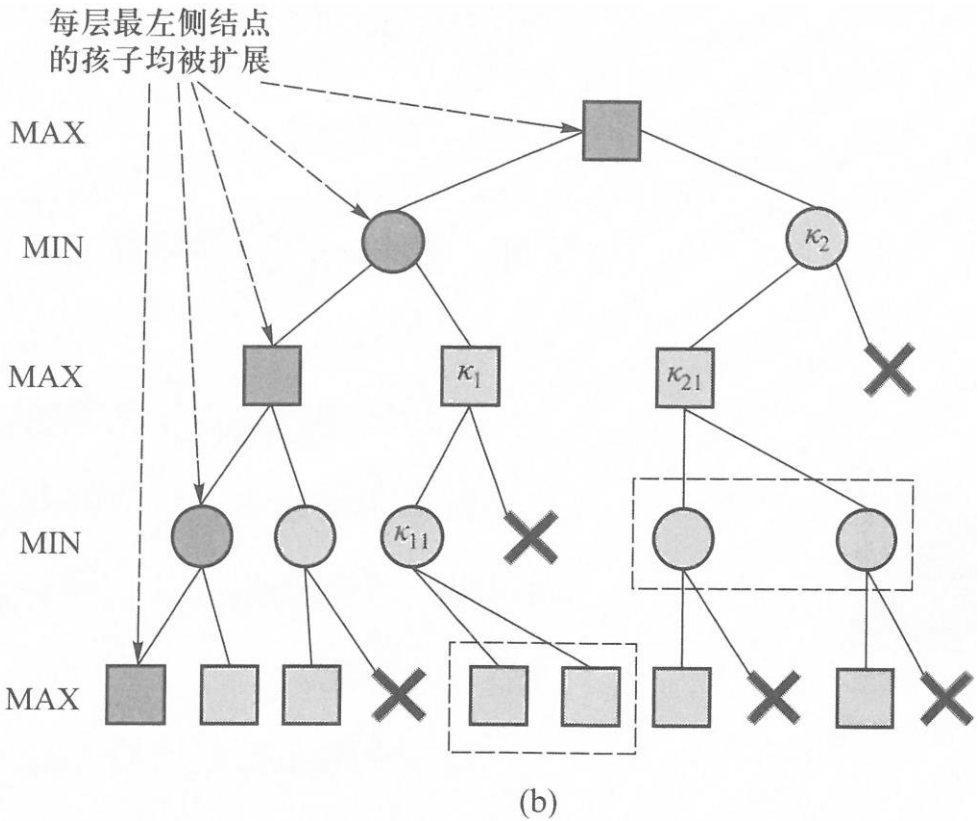
Alpha-Beta 剪枝算法

- Alpha-Beta 剪枝算法性能分析

不妨假设A 结点为MAX 结点

归谬法





- ⑤如果结点C导致B的其余子结点被剪枝，仿照对结点B的分析，可知结点C是结点B已扩展子结点中收益分数最小的一个。结点B 的收益分数必然是结点C 的收益分数，即 $\text{minimax}(C) = \text{minimax}(B) = \alpha'$。扩展结点C时结点B的界为$[\alpha'', \beta]$，扩展后$[\alpha'', \alpha']$
- ⑥同时根据剪枝的发生条件，C的收益分数α'必然满足 $\alpha'' > \alpha'$。矛盾

由④和⑥得：

如果某个结点导致了其右侧兄弟结点被剪枝，它所有孩子结点必然已被全部扩展。

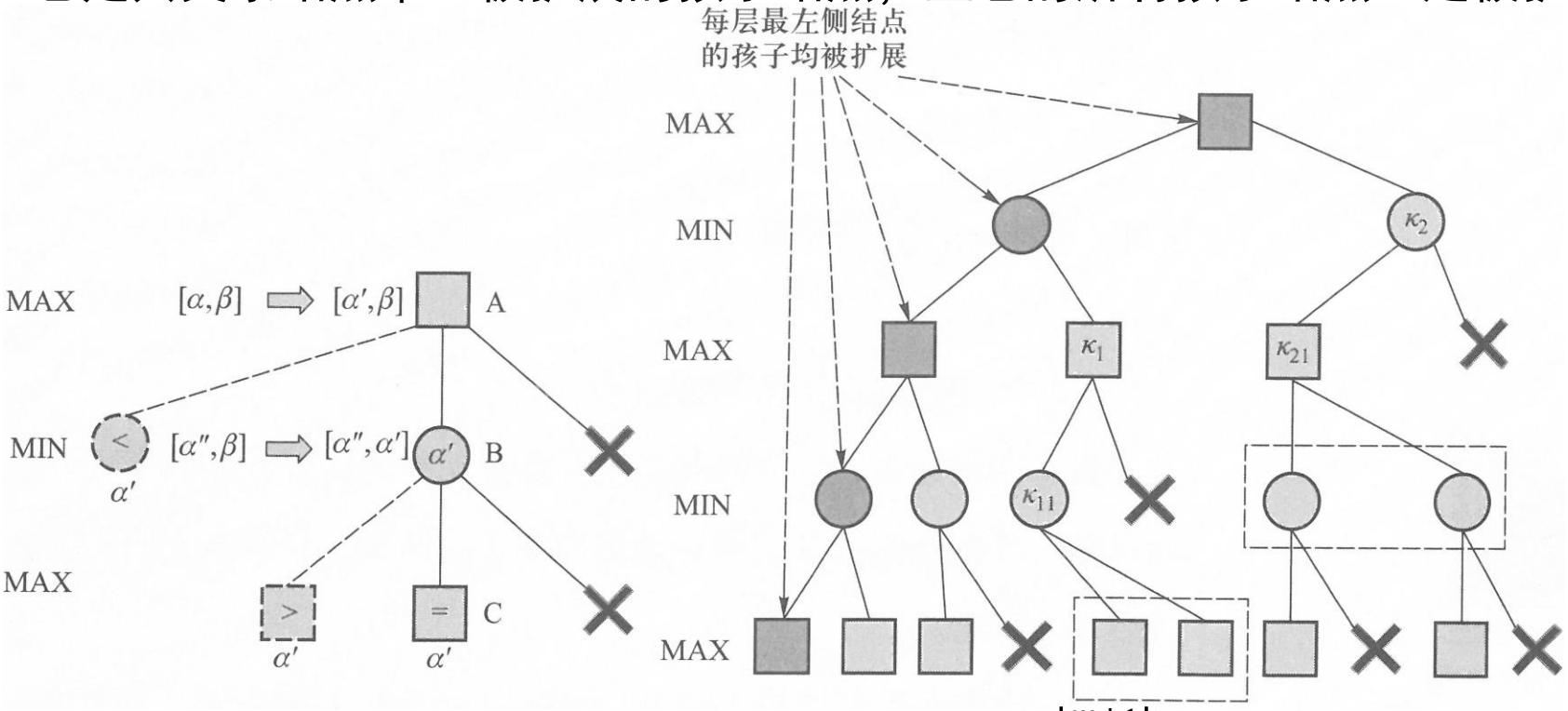
Alpha-Beta 剪枝算法

- Alpha-Beta 剪枝算法性能分析

搜索树每层最左端结点的孩子结点必然全部被扩展。其他结点的孩子结点被扩展情况分为如下两类：

① 只扩展其最左端孩子结点；

② 它是其父亲结点唯一被扩展的孩子结点，且它的所有孩子结点一定被扩展。



Alpha-Beta 剪枝在最优效率下扩展的结点数量为$O\left(b^{\left\lfloor\frac{m+1}{2}\right\rfloor}\right)$, m是树的最大深度, b为分枝因子。 比极小极大搜索的$O(b^m)$ 有显著提升。

Alpha-Beta 剪枝算法

- Move ordering

Alpha-beta 剪枝的效率高度依赖于状态被检查的顺序。

应该首先检查最有可能是最优的后继节点。

如果能够这样做，那么alpha-beta 算法只需检查$O\left(b^{\frac{m}{2}}\right)$个结点来做出决策（m是树的最大深度）， 而不是极小极大算法的$O(b^m)$。这意味着有效分支因子不是b而是$\sqrt{b}$。

如果后继状态采用随机顺序而不是最佳优先的顺序，那么要检查的总结点数大约是$O\left(b^{\frac{3m}{4}}\right)$。对于国际象棋，有一些相当简单的排序函数（如吃子优先，然后是威胁、前进、后退）可以使得检查的总结点数为 $O\left(b^{\frac{m}{2}}\right)$的两倍。

增加动态移动排序方案(dynamic move-ordering schemes)，如尝试首先使用先前走过的最好移动，可能让我们非常接近理论极限。

在博弈树搜索中，重复的状态频繁出现是因为换位 (transpositions) ——不同的移动序列导致相同的局面。第一次遇到某状态时把它的评估值存储在散列表里很有价值，这样当该局面后来再出现时不需要重新计算。

Heuristic Alpha-Beta Tree Search

Minimax won't work for games like chess and Go, because there are still too many states to explore in the time available.

- Type A strategy considers all possible moves to a certain depth in the search tree, and then uses a heuristic evaluation function to estimate the utility of states at that depth. It explores a wide but shallow portion of the tree.
- Type B strategy ignores moves that look bad, and follows promising lines "as far as possible." It explores a deep but narrow portion of the tree.

Claude Shannon in the very first paper on computer game-playing, *Programming a Computer for Playing Chess* (Shannon, 1950).

Heuristic Alpha-Beta Tree Search

在有限的计算时间内，将搜索较早截断，并在状态上使用启发式评估函数(evaluation function)，有效地把非终止结点转作为终止结点。
即：用估计棋局效用值的启发式评估函数EVAL取代效用函数，用决定什么时候运用EVAL的截断测试 (cutoff test) 取代终止测试 (terminal test)。因此得到如下的启发式极小极大值，s 为状态 d 为深度：

$$\text{H-MINIMAX}(s,d) = \begin{cases} \text{EVAL}(s,\text{MAX}) & \text{if IS-CUTOFF}(s,d) \\ \max_{a \in Actions(s)} \text{H-MINIMAX}(\text{RESULT}(s,a), d+1) & \text{if TO-MOVE}(s) = \text{MAX} \\ \min_{a \in Actions(s)} \text{H-MINIMAX}(\text{RESULT}(s,a), d+1) & \text{if TO-MOVE}(s) = \text{MIN}. \end{cases}$$

Evaluation functions：终止状态：$Eval(s,p) = Utility(s,p)$，
非终止状态：$Utility(loss,p) \leq Eval(s,p) \leq Utility(win,p)$

- 评估函数的计算本身不能花费太长时间（总观点是为了更快地搜索）。
- 对于非终止状态，评估函数应该和取胜几率密切相关。
- 大多数评估函数都要考虑状态的不同特征——例如在国际象棋中，包括白兵的数目、黑兵的数目、白后的数目、黑后的数目，等等。

加权线性函数：

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \cdots + w_n f_n(s) = \sum_{i=1}^{n} w_i f_i(s)$$

Heuristic Alpha-Beta Tree Search

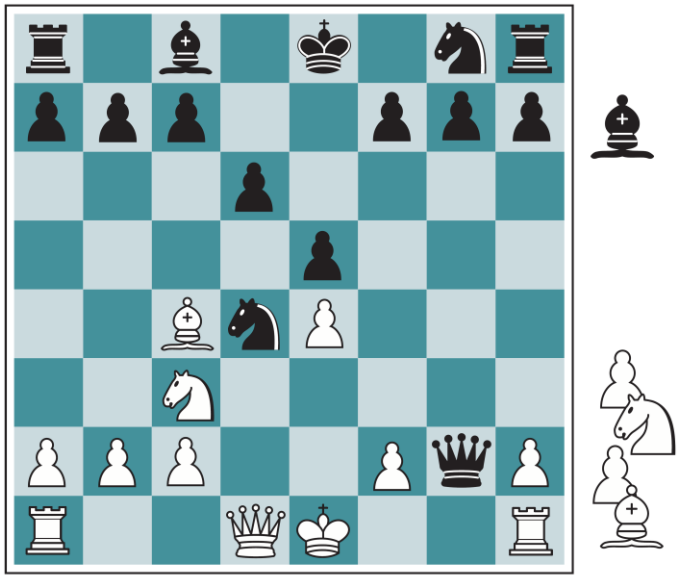
- Cutting off search: 当适合截断搜索时调用启发式函数EVAL。

if CUTOFF-TEST(state, depth) then return EVAL (state)

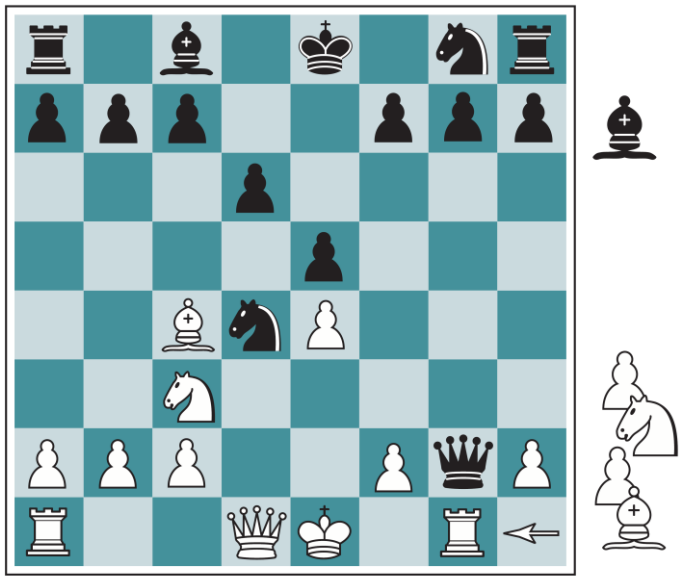
直接的控制搜索次数的方法是设置固定的深度限制，这样 CUTOFF-TEST (state, depth) 当depth大于固定深度d时返回true。

更好的方法是使用迭代加深搜索 (iterative deepening search)。当时间用光时，程序返回目前最深的完整搜索所选择的招数。而且迭代深入同样可以帮助行棋排序。

✓ 评估函数只是一种近似，可能导致误差



(a) White to move



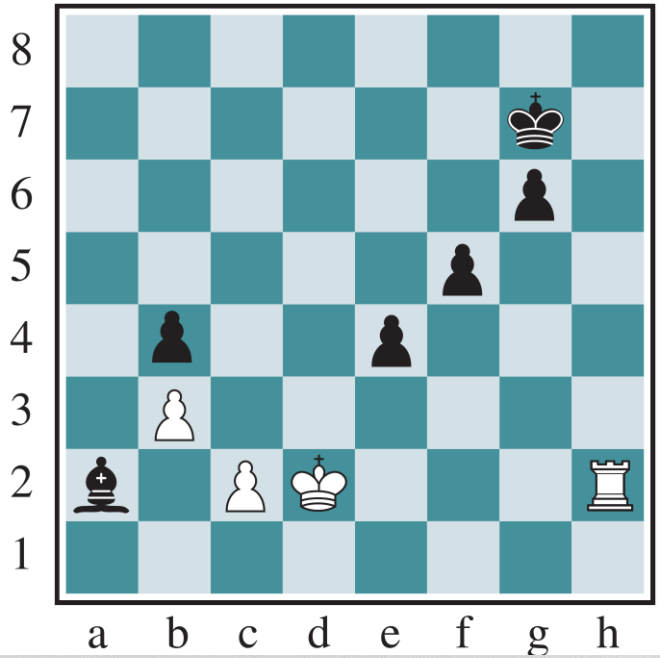
(b) White to move

这个局面实际上对白方有利，但这只有通过向前看才能知道。

Heuristic Alpha-Beta Tree Search

评估函数只适用于那些静态 (quiescent) 棋局，即没有搁置将会导致评估值很快出现剧烈变化的行动。在非静态棋局，搜索会继续进行一直达到静态棋局。

视野效应 (horizon effect) 更难消除。这是指对手招数导致我方严重损失并且从理论上基本无法避免时，采用延缓招数把不可避免的棋局推进了无法检测到的空间。单步延伸 (singular extensions) 是避免视野效应的一种策略，单步延伸指的是在给定棋局中一种棋招要“明显好于”其他棋招。一旦在搜索某处发现单步延伸，牢记它。当搜索到达指定深度界限，算法会检查单步延伸是否合法；如果是，算法允许考虑此棋招。



黑方移动后，黑象注定难逃厄运。但是黑方可以用兵来阻挡白方的王，引诱白王去吃掉兵。这会将不可避免的象的损失推到视野之外，因此搜索算法将牺牲兵的这一步看作“好招”

Heuristic Alpha-Beta Tree Search

- Forward pruning:

删除看起来不好的行动。节约了计算时间但是有出错的风险。

向前剪枝的一种方法是束搜索 (beam search): 在每一层只考虑最好的n步行棋可能，这称为 beam ，并不是考虑所有行棋招数。不幸的是，这种方法很危险，因为无法保证最佳的行棋不被剪枝。

PROBCUT算法 (概率截断算法) (Buro,1995)减除所有可能位于当前(α, β)窗口之外的节点。使用先前经验的统计信息减少最佳行棋被剪枝掉的概率。通过浅层搜索（shallow search）计算节点的 backed-up value，然后使用以前的经验估计深度 d 上的值 v 在(α, β) 范围外的可能性。

Heuristic Alpha-Beta Tree Search

- 搜索与查表 (search versus lookup): 初级机器学习——人工设计的模式

很多博弈程序在开局和残局都使用查表而不是搜索。

对于开局，计算机主要依靠人类的专业知识。还可以从以前游戏对局的数据库中收集统计数据，以判断哪种开局最容易取胜。
最开始几步可能的局面很少，大多数局面都能存储在表中。通常移动10-15步后会到达一个很少见的局面，程序必须从查表切换到搜索。

计算机可以通过生成一个策略 (policy) 将每一种可能状态映射到该状态最佳棋招，完全解决残局问题。通过 retrograde (逆向) minimax search 构建策略表。以KBNK (王象马对王)为例：start by considering all ways to place the KBNK pieces on the board. Some of the positions are wins for white; mark them as such. Then reverse the rules of chess to do reverse moves rather than moves. Any move by White that, no matter what move Black responds with, ends up in a position marked as a win, must also be a win. Continue this search until all possible positions are resolved as win, loss, or draw, and you have an infallible lookup table for all endgames with those pieces.

这种做法适用于所有棋子数不超过7的残局，这样的表格包含400万亿个状态。棋子数为8的表则包含40000万亿个状态。

主要内容

1. 对抗搜索 (Adversarial Search)

2. 博弈论相关概念 (Game Theory)

3. Alpha-Beta 剪枝算法 (Alpha-Beta Pruning)

4. 蒙特卡洛树搜索 (Monte Carlo Tree Search)

Bibliography:

[1] 吴飞 编著, “人工智能导论：模型与算法”, 高等教育出版社, 2020. Ch 3.4

[2] Stuart J. Russel, Peter Norvig, “Artificial Intelligence: A Modern Approach (4th Ed.)”, Pearson, 2020; 中译版 “人工智能 现代方法 (第4版) ”, 人民邮电出版社, 2022. Ch 6

Monte Carlo Tree Search (MCTS)

在问题特别复杂时，搜索树可能会变得十分巨大，导致搜索算法很难在短时间内探索整棵搜索树。为了解决这个问题，Heuristic 搜索利用领域特定的信息（知识）来找到高效的结点扩展顺序，Alpha-Beta 剪枝使用剪枝思想减少需扩展的结点数。

对搜索算法进行优化以提高搜索效率基本上是在解决如下两个问题：优先扩展哪些结点以及放弃扩展哪些结点，可以综合概括为如何高效地扩展搜索树。

- 分枝因子 (branching factor) 很大时，alpha-beta 搜索只能进行浅搜索
- 很难定义一个好的价值函数 (evaluation function)

如果将目标稍微降低， 改为求解一个近似最优解，则上述问题可以看成是如下探索性问题：算法从根结点开始， 每一步动作为选择（在非叶子结点）或扩展（在叶子结点）一个孩子结点。 可以用执行该动作后所收获的奖励来判断该动作优劣。奖励可以根据从当前结点出发到达目标路径的代价或博弈终局分数来定义。算法会倾向于扩展获得奖励较高的结点。evaluation function

算法事先不知道每个结点将会得到怎样的代价（或终局分数） 分布，只能通过采样式探索来得到计算奖励的样本。由于这个算法利用蒙特卡洛法通过采样来估计每个动作优劣，因此它被称为蒙特卡洛树搜索 (Monte-Carlo tree search) 算法。

Probability axioms

- σ-algebra

Let X be some set, and let $\mathcal{P}(X)$ represent its power set. Then a subset $\Sigma \subseteq \mathcal{P}(X)$ is called a σ-algebra if and only if it satisfies the following three properties:

(1) X is in Σ, and X is considered to be the universal set in the following context.

(2) Σ is closed under complementation: If $A \in \Sigma$, then so is its complement, $X \backslash A$.

(3) Σ is closed under countable unions.

Σ is also closed under countable intersections (by applying De Morgan's laws). The ordered pair (X, Σ) is called a measurable space.

X 上的集族 $\mathfrak{F}$ 未必是一个 σ-algebra，但一定有包含它的 σ-algebra。所有包含 $\mathfrak{F}$ 的 σ-algebra 的交还是 σ-algebra，称之为由 $\mathfrak{F}$ 生成的 σ-algebra。记作 $\mathcal{B}(\mathfrak{F})$.

如果 $\mathfrak{F}$ 取 X 上的所有 open set，那么它生成的 σ-algebra 称为 Borel σ-algebra。其中的元素称为 Borel set.

Probability axioms

- Probability axioms:

Given any set Ω (also called sample space) and a σ-algebra $\mathfrak{F}$ on it, a measure P defined on $\mathfrak{F}$ is called a probability measure if $P(\Omega) = 1$.
The probability of a set E in the σ-algebra $\mathfrak{F}$ is s defined as:

$$P(E) = \int_{\omega \in E} \mu_{\mathfrak{F}}(d\omega)$$

where the integration is with respect to the measure $\mu_{\mathfrak{F}}$ induced by $\mathfrak{F}$.
Any subset of the sample space that is not an element of the σ-algebra is not an event, and does not have a probability. All events of interest are elements of the σ-algebra.

定义2.1 [Kolmogorov] 称 $\mathfrak{F}$ 上集函数 P 为概率，如果 P 满足
(1) 非负性，即 $\forall A \in \mathfrak{F}, P(A) \geq 0$.
(2) 规范性，$\forall P(\Omega) = 1$.
(3) 可数可加性（σ可加性），设 $\{A_n\}_{n \geq 1} (\subset \mathfrak{F})$ 之间的交集为空集，则：

$$P\left(\sum_{n=1}^{+\infty} A_n\right) = \sum_{n=1}^{+\infty} P(A_n)$$

Stochastic Process

Random Variable: Let $(\Omega, \mathcal{F}, P)$ be a probability space and $(E, \mathcal{E})$ a measurable space. Then an $(E, \mathcal{E})$-valued random variable is a measurable function $X: \Omega \to E$, which means that, for every subset $B \in \mathcal{E}$, its preimage is $\mathcal{F}$-measurable; $X^{-1}(B) \in \mathcal{F}$, where $X^{-1}(B) = \{\omega: X(\omega) \in B\}$. The probability that X takes on a value in a measurable set is written as:

$$P(X \in S) = P(\{\omega \in \Omega \mid X(\omega) \in B\})$$

Stochastic Process: A collection of random variables defined on a common probability space $(\Omega, \mathcal{F}, P)$, where Ω is a sample space, $\mathcal{F}$ is a σ-algebra, and P is a probability measure; and the random variables, indexed by some set T, all take values in the same mathematical space S, which must be measurable with respect to some σ-algebra Σ.

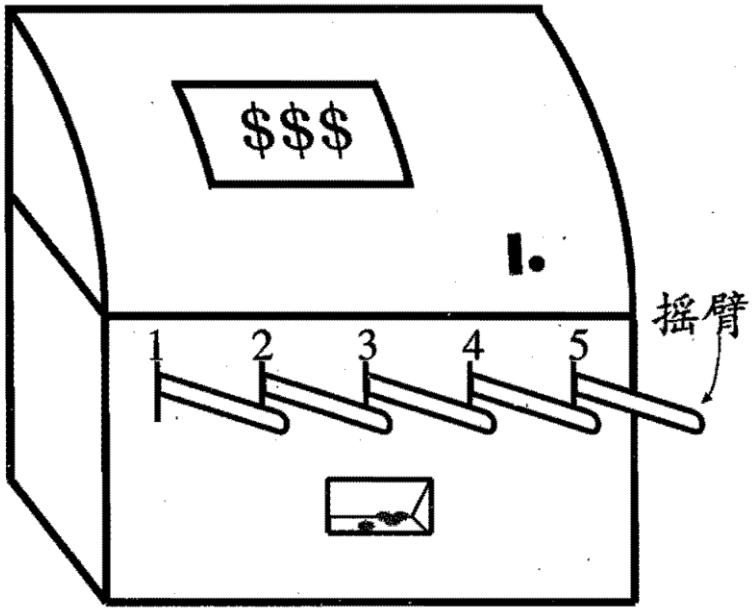
For a given probability space $(\Omega, \mathcal{F}, P)$ and a measurable space (S, Σ), a stochastic process is a collection of S-valued random variables, which can be written as:

$$\{X(t, \omega): t \in T\}$$

Monte Carlo Tree Search (MCTS)

- 探索 (exploration) 与利用 (exploitation) 的平衡

在有些问题中，Agents 对环境并没有完整的认识，它们事先并不知道每种行动的优劣以及这些行动造成的不同结果，只有当它们实际尝试过后才能确定。



多臂赌博机 (multi-armed bandit) 问题：摇臂赌博机有 K个摇臂，赌徒在投入一个硬币后可选择按下其中一个摇臂，赌博机则会以一定的概率吐出硬币，将所吐出的硬币的币值用收益分数来表示。现在假设给智能体 $\tau(\tau > K)$次转动摇臂的机会，那么智能体如何选择并转动τ次摇臂，能够获得更多的收益分数呢？

方法：让赌徒先把K个赌博机的臂膀依次转动一遍，观察在摇动每个赌博机臂膀时的收益分数，然后去转动那些收益分数高的赌博机臂膀。但是，由于从每个赌博机获得的收益分数是随机的，显然一个刚刚给赌徒带来可观收益分数的赌博机在下一次摇动其臂膀时就可能不会获得可观收益分数了，因此这一方法不可取。

Monte Carlo Tree Search (MCTS)

- Each arm M_i is a Markov reward process or MRP, that is, an MDP with only one possible action a_i. It has states S_i, transition model $P_i(s'|s, a_i)$, and reward $R_i(s, a_i, s')$. The arm defines a distribution over sequences of rewards, $R_{i,0}, R_{i,1}, R_{i,2}, \cdots$, where each $R_{i,t}$ is a random variable.
- The overall bandit problem is an MDP: the state space is given by the Cartesian product $S = S_1 \times \cdots \times S_n$; the actions are $a_1, \cdots, a_n$; the transition model updates the state of whichever arm M_i is selected, according to its specific transition model, leaving the other arms unchanged; and the discount factor is γ.
- **状态**。每个被摇动的臂膀即为一个状态，记K个状态分别为$\{s_1, s_2, \cdots, s_K\}$，没有摇动任何臂膀的初始状态记为$s_0$。
- **动作**。动作对应着摇动一个赌博机的臂膀，在多臂赌博机问题中，任意状态下的动作集合都为$\{a_1, a_2 \cdots, a_K\}$，分别对应摇动某个赌博机的臂膀。
- **状态转移**。选择动作$a_i(1 \leq i \leq K)$后，将相应的改变为s_i。
- **奖励 (reward)**。假设从第i个赌博机获得收益分数的分布为D_i，其均值为μ_i。如果智能体在第t次行动中选择转动了第l_t个赌博机臂膀，那么智能体在第t次行动中所得收益分数$\hat{r}_t$服从分布D_{l_t}，$\hat{r}_t$被称为第t次行动的奖励。进一步可假设 $\hat{r}_t \in [0,1]$。

Monte Carlo Tree Search (MCTS)

- **悔值 (regret) 函数**。根据智能体前T次动作，可以如下定义悔值函数：

$$\rho_T = T\mu^* - \sum_{t=1}^{T} \hat{r}_t \quad (3.2) \qquad \mu^* = \max_{i=1,\cdots,K} \mu_i$$

T次操作中最优策略的期望得分减去智能体的实际得分

为了尽量减少后悔，在每次操作时，智能体应该总是转动能够给出最大期望奖励的赌博机臂膀，但是这是不现实的，因为智能体并不知道哪个臂膀的奖励期望最大。

问题求解的目标为最小化悔值函数的期望，该悔值函数的取值取决于智能体所采取的策略。

Monte Carlo Tree Search (MCTS)

- 贪心算法策略：智能体记录下每次摇动的赌博机臂膀和获得的相应收益分数。给定第$i(1 \leq i \leq K)$个赌博机臂膀，记在过去$t-1$次摇动赌博机臂膀的行动中，一共摇动第i个赌博机臂膀的次数为第$T_{(i,t-1)}$。于是，可以计算得到第i个赌博机在过去$T_{(i,t-1)}$次被摇动过程中的收益分数平均值$\bar{x}_{i,T_{(i,t-1)}}$。这样，智能体在第t步，只选择$\bar{x}_{i,T_{(i,t-1)}}$值最大的赌博机臂膀进行摇动，这是贪心算法的思路。
  - 不足：忽略了其他从未摇动或很少摇动的赌博机，而失去了可能的机会。

上述困境体现了探索 (exploration) 和利用 (exploitation) 之间存在对立关系。

贪心算法基本上是利用从已有尝试结果中所得估计来指导后续动作，但问题是所得估计往往不能准确反映未被（大量）探索过的动作。因此，需要在贪心算法中增加一个能够改变其“惯性”的内在动力，以使得贪心算法能够访问那些尚未被（充分）访问过的空间。

Monte Carlo Tree Search (MCTS)

- ϵ-贪心算法：在探索与利用之间进行平衡的搜索算法。

在第t步，ϵ-贪心算法按照如下机制来选择摇动赌博机：

$$l_t = \begin{cases} \text{argmax}_i \, \bar{x}_{i,T(i,t-1)}, & \text{以} 1-\epsilon \text{的概率} \\ \text{随机的} i \in \{1,2,\cdots,K\}, & \text{以} \epsilon \text{的概率} \end{cases}$$

即以$1-\epsilon$的概率选择在过去$t-1$次摇动赌博机臂膀行动中所得平均收益分数最高的赌博机进行摇动；以ϵ的概率随机选择一个赌博机进行摇动。

不足：与被探索的次数无关。可能存在一个给出更好奖励期望的动作，但因为智能体对其探索次数少而认为其期望奖励小。因此，需要对那些探索次数少或几乎没有被探索过的动作赋予更高的优先级。

Monte Carlo Tree Search (MCTS)

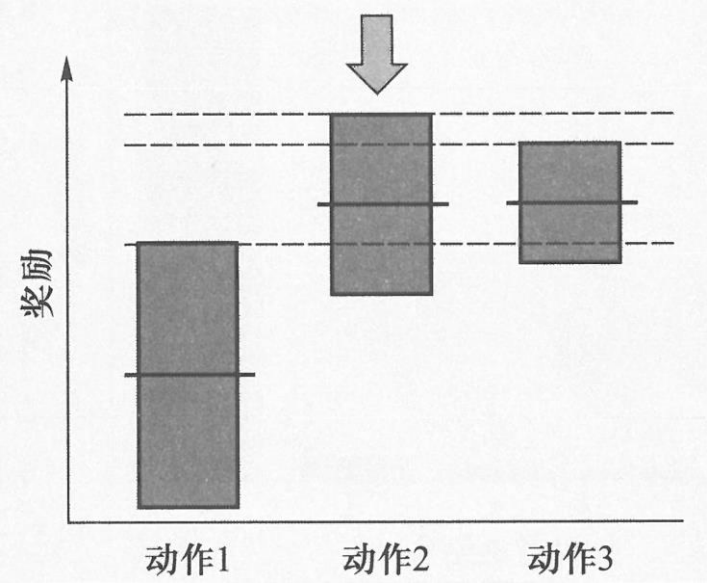
- 上限置信区间算法 (Upper Confidence Bounds, UCB1)

假设算法在第t次已经摇动了第i个赌博机的臂膀$T_{(i,t-1)}$次，执行动作a_i所收到收益分数的均值为$\bar{x}_{i,T_{(i,t-1)}}$。这$T_{(i,t-1)}$次动作可以看作$T_{(i,t-1)}$个取值范围在$[0,1]$的独立同分布随机变量的样本，根据霍夫丁不等式 (Hoeffding's inequality)：

$$P\left(\mu_i - \bar{x}_{i,T_{(i,t-1)}} > \delta\right) \leq e^{-2T_{(i,t-1)}\delta^2} \qquad (3.4)$$

只需找到一个δ，使得不等式（3.4）右侧足够小，即可认为 $\bar{x}_{i,T_{(i,t-1)}} + \delta$ 是μ_i的一个上界。

找一个随时间增长快速趋近于0的函数，令 $e^{-2T_{(i,t-1)}\delta^2} = t^{-4}$，则$\delta = \sqrt{\frac{2\ln t}{T_{(i,t-1)}}}$，因此$\mu_i$的上界为 $\bar{x}_{i,T_{(i,t-1)}} + \sqrt{\frac{2\ln t}{T_{(i,t-1)}}}$。如果换成关于$t$的其它幂函数，结果将会和当前$\delta$取值相差一个常数系数。奖励期望的上界也可以写成 $\bar{x}_{i,T_{(i,t-1)}} + C\sqrt{\frac{2\ln t}{T_{(i,t-1)}}}$，其中$C$是一个预先指定的超参数，可以理解为用来调节探索和利用两者权重的因子。



Monte Carlo Tree Search (MCTS)

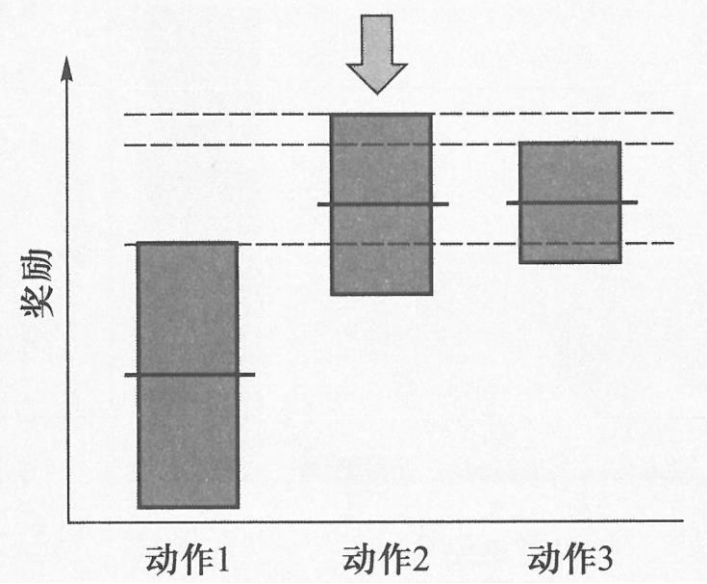
- 上限置信区间算法 (Upper Confidence Bounds, UCB1)

UCB1算法的策略是： 为每个动作的奖励期望计算一个估计范围，优先采用估计范围上限较高的动作。

可以描述为，在第t次时选择使得式（3.5）取值最大的动作a_{l_t}，其中l_t由如下式子计算得到：

$$l_t = \underset{i}{\operatorname{argmax}}\left(\bar{x}_{i,T_{(i,t-1)}} + C\sqrt{\frac{2\ln t}{T_{(i,t-1)}}}\right) \qquad (3.5)$$

在过去第t次已经对动作a_i探索了$T_{(i,t-1)}$次，在当前问题中对应摇动了第i个赌博机的臂膀$T_{(i,t-1)}$次，执行动作a_i所收到收益分数的均值为$\bar{x}_{i,T_{(i,t-1)}}$

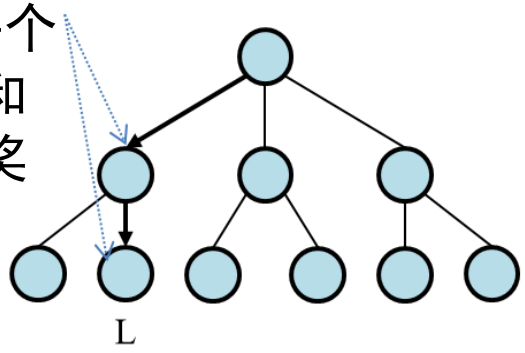


Monte Carlo Tree Search (MCTS)

(1) 选择 (selection)

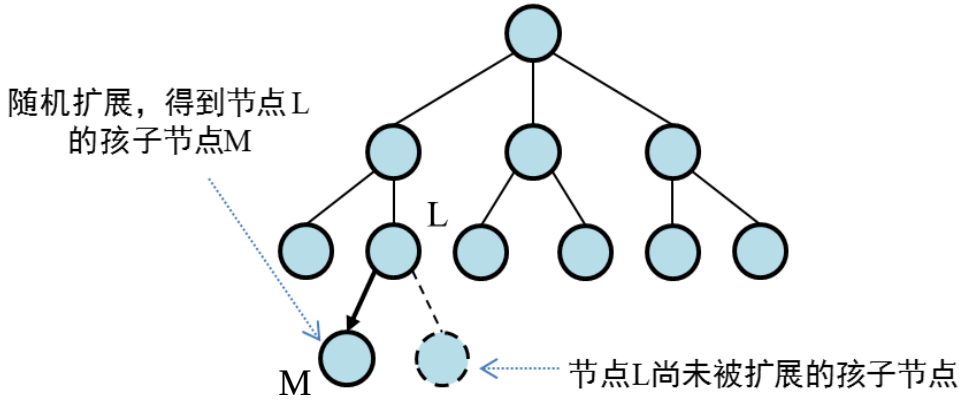
向下递归选择子节点，直至到达叶子结点或者到达具有还未被扩展过的子节点的结点L。记录每个结点被选择次数和每个结点得到的奖励均值。

selection policy
or tree policy



(2) 扩展 (expansion)

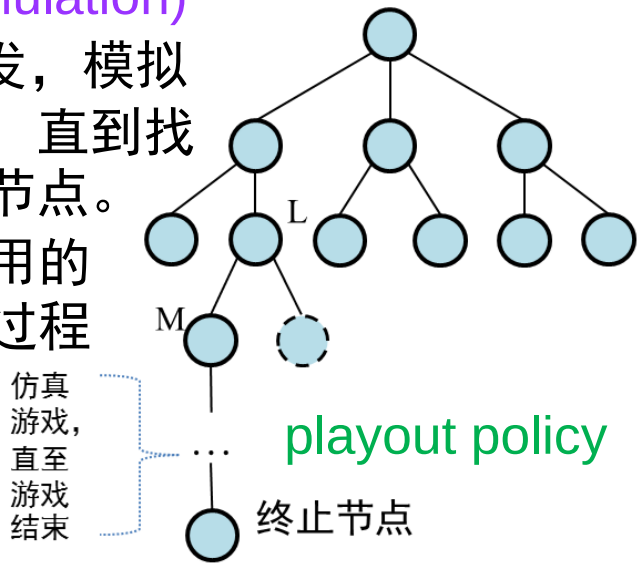
若L不是终止结点，随机扩展它的一个未被扩展过的后继边缘节点M。



(3) 模拟 (simulation)

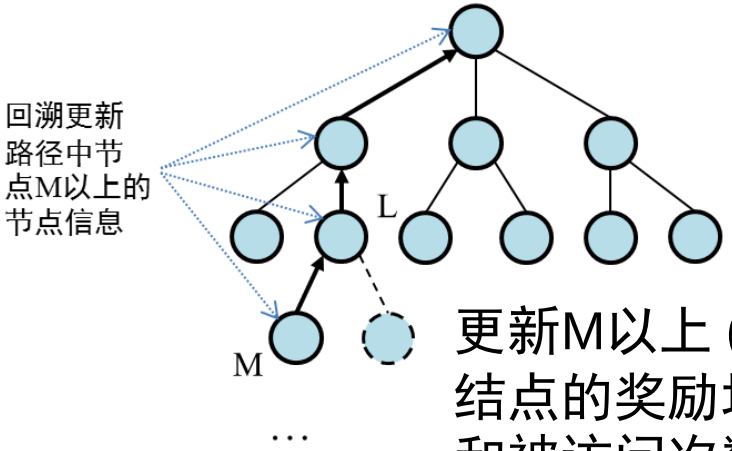
从节点M出发，模拟扩展搜索树，直到找到一个终止节点。

模拟过程使用的策略和选择过程并不相同。



playout policy

(4)反向传播 (Back Propagation)



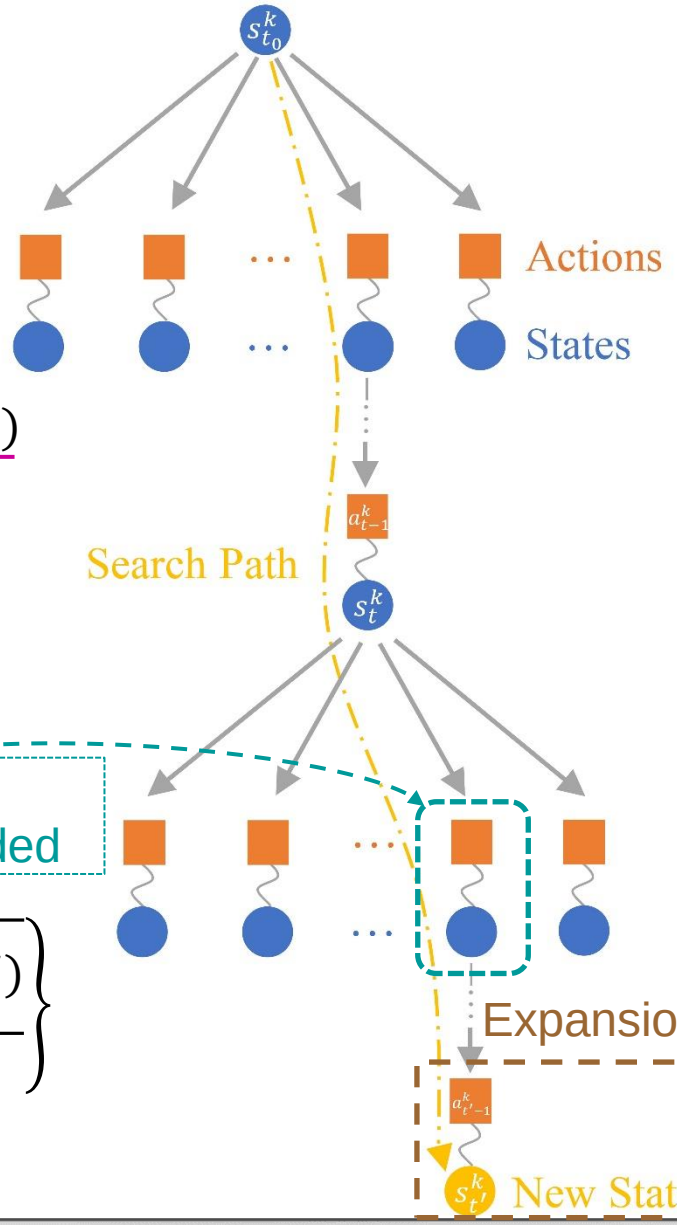
更新M以上 (含M) 结点的奖励均值和被访问次数

Monte Carlo Tree Search (MCTS)

- 选择 (selection) 和扩展 (expansion)

向下递归选择子节点，直至到达叶子节点或者到达具有还未被扩展过的子节点的节点L。记录每个结点被选择次数和每个结点得到的奖励均值。

selection policy or tree policy



动作-价值函数

$$\sum_{a'} N^{k-1}(s_t^k, a') = N^{k-1}(s_t^k)$$

UCT

$$a_t^k = \underset{a}{\operatorname{argmax}} \left\{ \bar{Q}^{k-1}(s_t^k, a) + \sqrt{\frac{2\ln \sum_{a'} N^{k-1}(s_t^k, a')}{N^{k-1}(s_t^k, a)}} \right\}$$

exploration　　exploitation

Partially expanded node
Node that is not fully expanded

AlphaGo Zero

$$a_t^k = \underset{a}{\operatorname{argmax}} \left\{ \bar{Q}^{k-1}(s_t^k, a) + c f_\pi(s_t^k, a) \frac{\sqrt{\sum_{a'} N^{k-1}(s_t^k, a')}}{1 + N^{k-1}(s_t^k, a)} \right\}$$

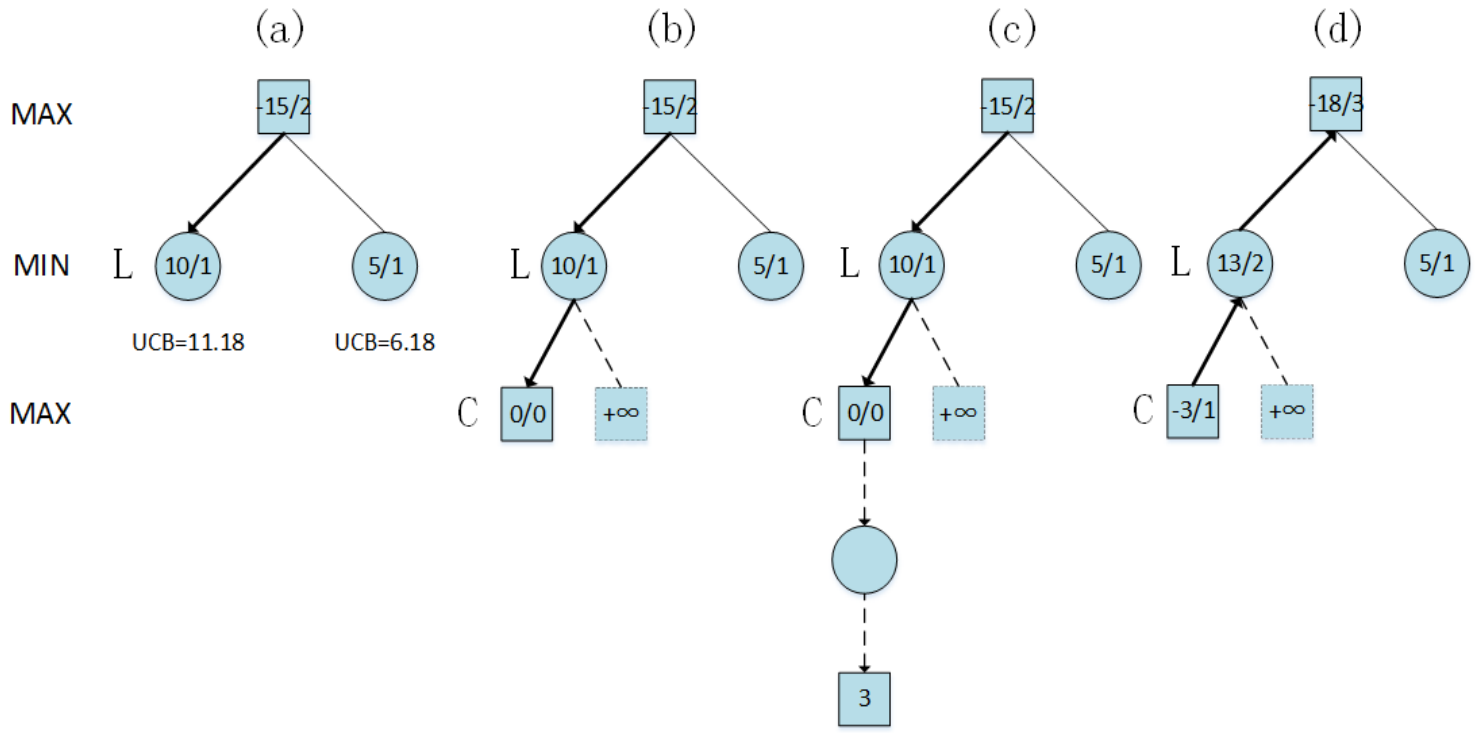
the policy network as prior probability

Monte Carlo Tree Search (MCTS)

- 选择 (selection)：从搜索树的根节点开始，向下递归选择子节点，直至到达叶子节点或者到达具有还未被扩展过的子节点的节点L。在递归选择过程中记录下每个节点被选择次数和每个节点得到的奖励均值。可由UCB1算法实现。
- 扩展 (expansion)：如果节点 L 不是一个终止节点（或对抗搜索的终局节点），则随机扩展它的一个未被扩展过的后继边缘节点M。
- 模拟 (simulation)：从节点M出发，模拟扩展搜索树，直到找到一个终止节点。模拟过程使用的策略和选择过程并不相同，前者通常会使用比较简单的策略，例如使用随机策略。
- 反向传播 (Back Propagation)：用模拟所得结果（终止节点的代价或游戏终局分数）回溯更新模拟路径中M以上（含M）节点的奖励均值和被访问次数。

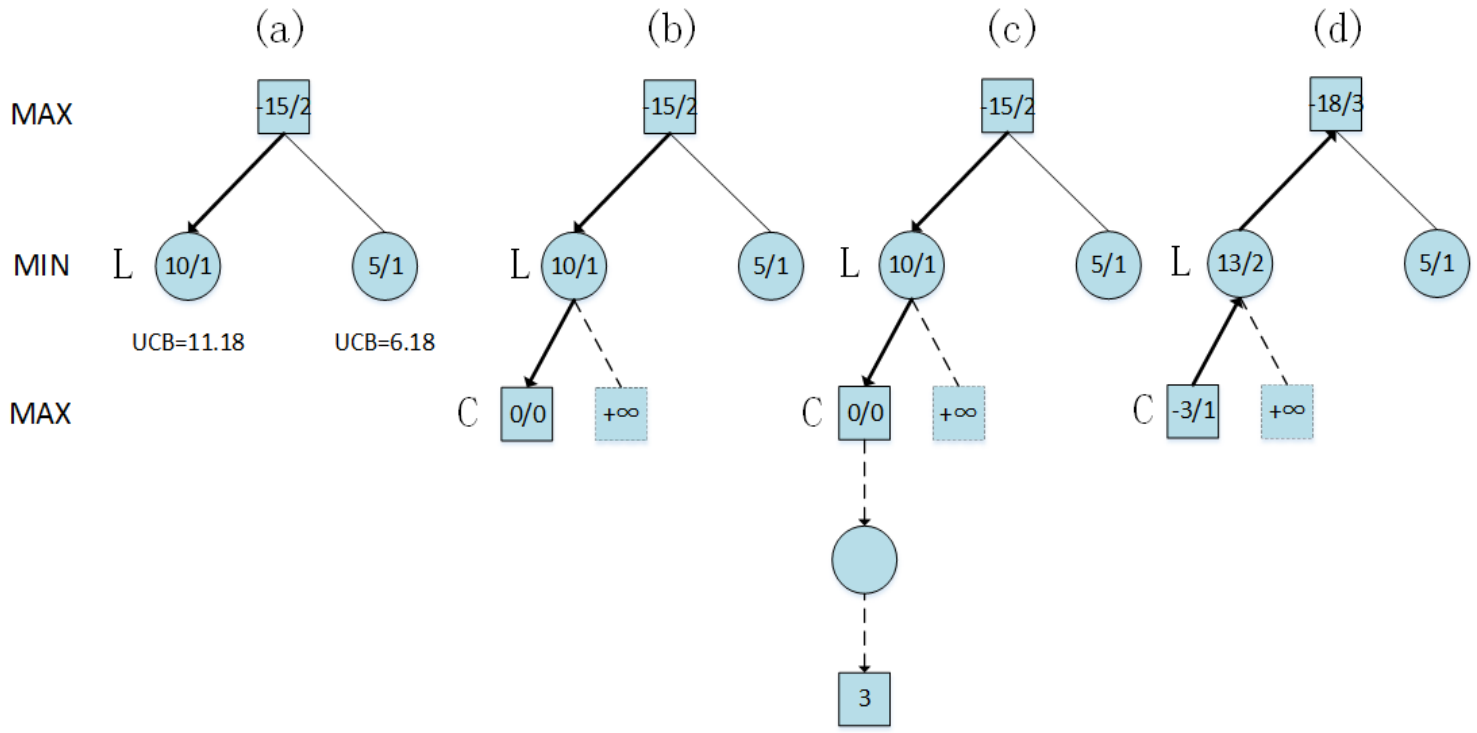
- early playout termination: in games that can last many movers, we stop a playout that is taking too many moves, and either evaluate it with a heuristic evaluation function or just declare it a draw.
- The selection and playout policies can make good use of hand-crafted expert knowledge, but good policies can be learned using neural networks trained by self-play alone.

Monte Carlo Tree Search (MCTS)



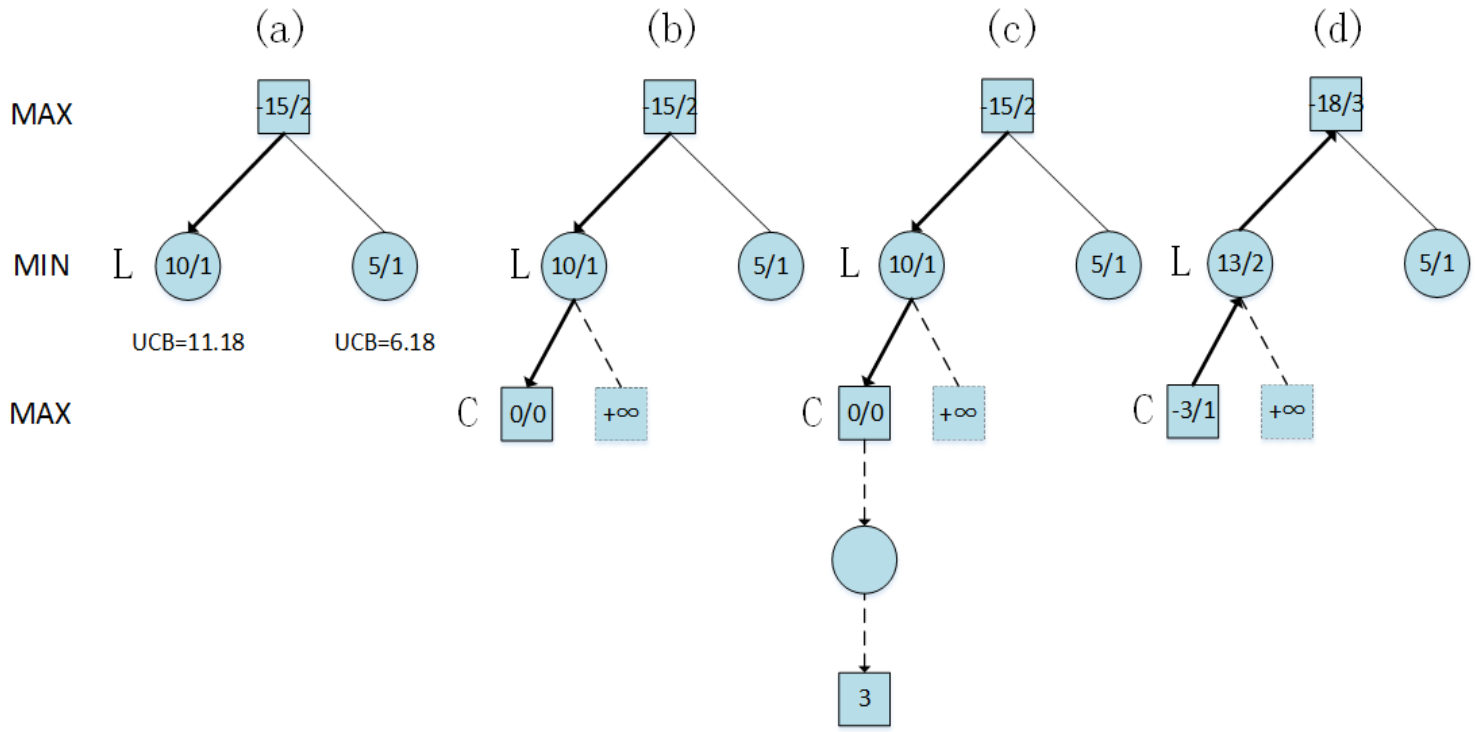
计算第二层节点的UCB值：左侧节点为$\frac{10}{1}+\sqrt{\frac{2\ln 2}{1}}=11.18$，右侧节点为$\frac{5}{1}+\sqrt{\frac{2\ln 2}{1}}=6.18$，因此算法选择第二层左侧的节点L，由于该节点有尚未扩展的子节点，因此选择阶段结束。

Monte Carlo Tree Search (MCTS)



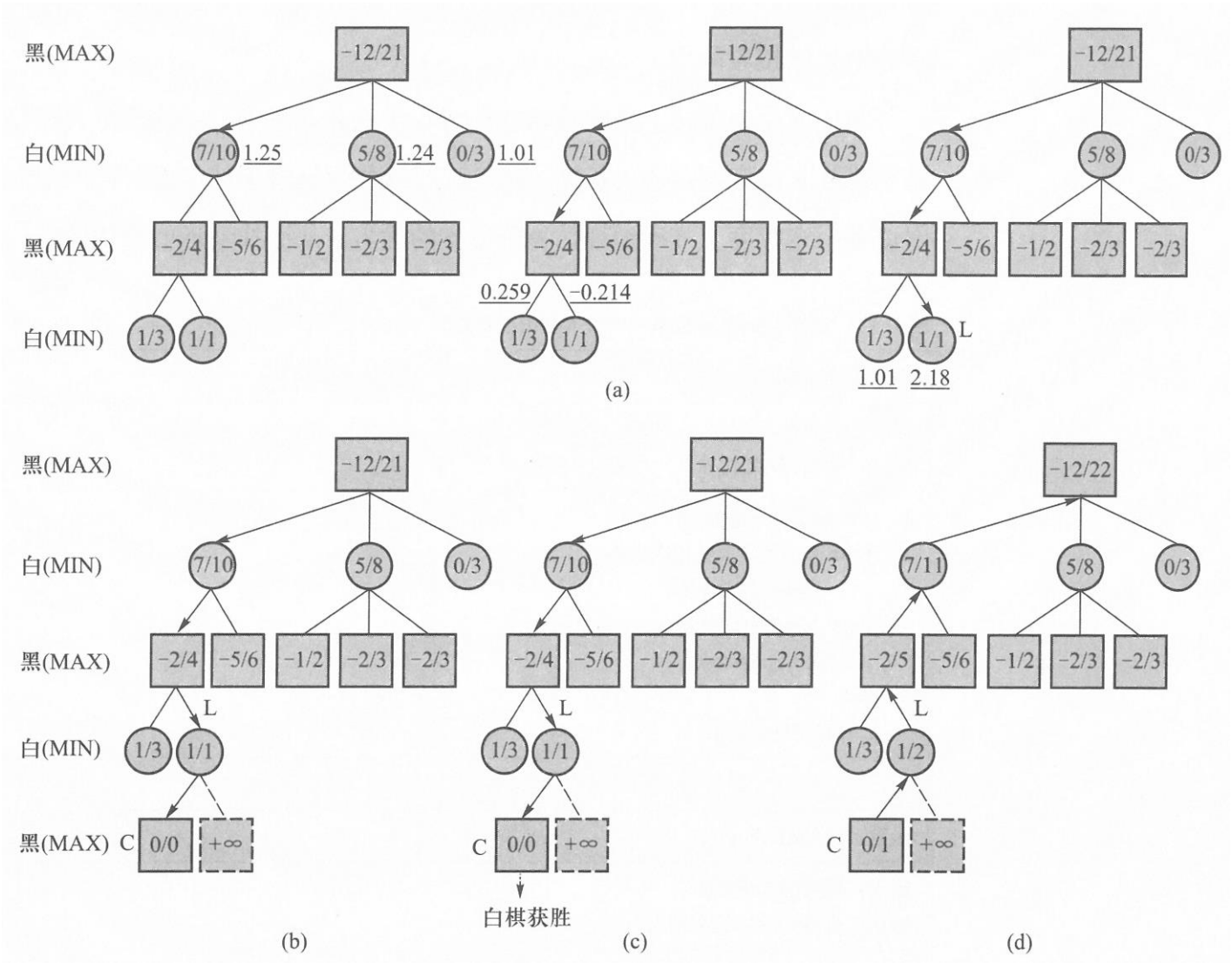
在图3.4.6(b)中，算法随机扩展了L的子节点C，将其总分数和被访问次数均初始化为0。注意，为了清晰地展示算法选择扩展的节点，图3.4.6(b)画出了L的其他未被扩展的子节点，并标记其UCB值为正无穷大，以表示算法下次访问到L时必然扩展这些未被扩展的节点。图3.4.6(c)中采用随机策略模拟游戏直至完成游戏。当游戏完成时，终局得分为3。

Monte Carlo Tree Search (MCTS)



在图3.4.6(d)中C节点的总分被更新为-3，被访问次数被更新为1；L节点的总分被更新为13，被访问次数被更新为2；根节点的总分被更新为-18，被访问次数被更新为3。在更新时，会将MIN层节点现有总分加上终局得分分数， MAX层节点现有总分减去终局得分分数。这是因为在对抗搜索中，玩家MIN总是期望最小化终局得分，因此在MIN层选择其子节点时，其目标并非选取奖励最大化的子节点，而是选择奖励最小化的节点，为了统一使用UCB1算法求解，算法将MIN层的子节点（即MAX层节点）的总分记为其相反数。

Monte Carlo Tree Search (MCTS)



本章小结

- 最小最大搜索 (Minimax Search): 最小最大搜索是在对抗搜索中最基本的一种让参与者计算最优策略的方法。
- 博弈论基本概念 (Game Theory): 研究博弈行为中最优的对抗策略及其稳定局势，协助人们在一定规则范围内寻求最合理的行为方式。
- Alpha-Beta剪枝搜索 (Pruning Search): 一种对最小最大搜索进行改进的算法，即在搜索过程中可剪除无需搜索的分支节点，且不影响搜索结果。
- 蒙特卡洛树搜索 (Monte-Carlo Tree Search): 通过采样而非穷举方法来实现搜索。

Bibliography:
[1] 吴飞 编著, “人工智能导论：模型与算法”, 高等教育出版社, 2020. Ch 3.3, Ch 3.4, Ch 8.1
[2] Stuart J. Russel, Peter Norvig, “Artificial Intelligence: A Modern Approach (4th Ed.)”, Pearson, 2020; 中译版 “人工智能 现代方法 (第4版) ”, 人民邮电出版社, 2022. Ch 6, Ch 17.1.2, Ch 17.2, Ch 17.3