人工知能A

Topic 3: ゲーム木とゲーム理論の入門

Topic 3: Introduction of games and game theory

3

ゲーム木とゲーム理論入門

- 講義の内容
  - ゲーム木
  - ミニマックス法とアルファベータ枝刈り（ゲーム理論の基礎でもある）
  - その他の探索木：AND/OR木
  - 多様な対戦・協調・調整・競争、多人数（多エージェント）の場合
- 目標：以下を理解する
  - ミニマックス法とアルファベータ枝刈り
  - 問題構造や文構造を表すAND/OR木
  - ゲーム理論の入門

ゲームに関する研究

- AIではゲームの研究が初期から開始された
  - 初期はチェス、バックギャモンなど、その後将棋や囲碁など。
  - ゲームは比較的単純なルールと動作で構成される。ゲームの状態を正確に表現できる。
  - 敵対エージェントが味方を妨害するという状況の良い例である（このため、最近はより広く「Adversarial search」とも言われる）。
  - さらに、ゲーム（たとえばチェスや将棋）ができ、ある程度強い計算機ができれば、そこに知能が存在することを広く示せる・納得させられると考えた。

ゲームに関する研究

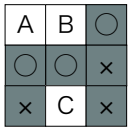
- ゲームの難しさ
  - 相手が何をするか分からないという不確実性を含む。
  - ゲームは実は複雑。たとえばチェスでは
    - チェスの平均分岐数は35と言われる。ゲームの平均手数は100したがって、35^{100}の探索木が必要。しかしチェスの可能な状態は10^{40}である。
  - 時間制限がある。効率が悪いのは致命的。
  - 勝敗が明確。少しの差で勝負という極端な結果となる。
- ゲームに関するテーマ
  - 探索木を制限（枝刈り、pruning)。悪手は考えない。
  - 相手の手を読む（推測する）。

ゲームを探索問題と捉える

- ゲームを以下のように考えると探索問題
  - 初期状態＝駒の最初の配置など
  - オペレーター
    - プレーヤが行えるルールに従った指し手
  - 終端テスト
    - ゲームが終了した状態を判定
  - 効用関数（utility function）または利得関数（payoff function）
    - ゲームの結果を数値としてとらえる関数。たとえば、勝ちなら1、負けなら-1、引き分けなら0など。
  - ゲーム木とよぶ

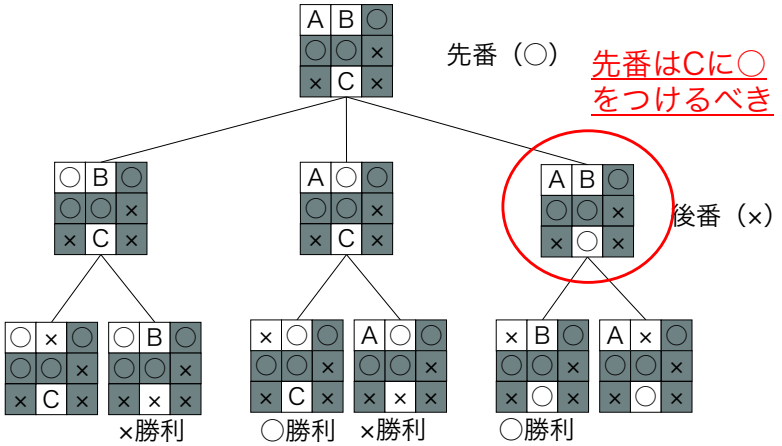
ゲーム木探索

- チェス、将棋などの２人が交互に手を指しながら進めるゲームを扱う。したがって木は、自分と相手の交互の手を表す探索木という特徴がある。
- 結果は、勝ち、負け、引き分けの３種類。各プレイヤー（エージェント）は勝ちを目指して手を決める。相手の手を直接妨ぐことはできない。
- 例から：　三目並べ （Tic-Tac-Toe）



- 空白をA、B、Cなどと名前をつける。

ゲーム木の例（三目並べ）

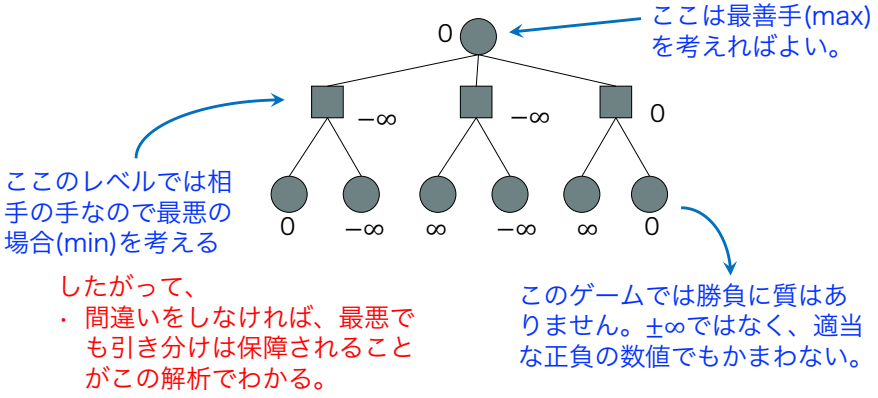


ミニマックス法 (minimax, MM)

- ゲームでは、何手か先を読み、それぞれの評価値により最善（と思われる）手を選択する。
  - 勝ちは+∞（勝ちの質に応じて評価をしてもよい。たとえば、囲碁なら何目で勝つかなど）、負けは−∞、引き分け0
  - 途中は勝ちにつながる可能性により、数値を割り当てる。
- ゲーム木では、自分と相手が交互になるので、自分は最善の手を打つとして評価値の最大(max)を、相手も最善（自分にとって最悪、つまり最小(min)）を取るとし、各ノードを評価する。【相手の最良は自分の最悪という関係を仮定していることに注意】
- これを繰り返し、ゲーム木の下側から評価し、手を決定する方法をミニマックス法という。

ミニマックス法 (minimax)

- 前の例を抽象化し、先番を○、後番を□で表してみる。



ミニマックス法の概要

1. すべての筋道で終端状態に至る完全な探索木を生成する。
2. それぞれの終端状態に効用関数を適応し評価する。
   - 単に勝負だけなら±1。バックギャモンや囲碁のように勝ちの度合いがあるならその数値など。
3. 終端状態の効用から一つ上のノードの効用を計算
   - 自分か相手の番によりminかmaxを取る。
4. 上記を繰り返し、初期状態（探索木の頂点）に達するまで計算する。
5. 最大の効用となる手を選択する

ミニマックス法の効果と問題点

- ミニマックス法では、最悪の場合の回避を中心に評価できる。
- 問題点
  - 最終的に読みきりができなければ、評価はできない。
  - 三目並べならたかだか９階層（９マスしかないから）だから、完全なる先読みは容易だが、一般にはできない。
  - 複雑になれば、計算量は急激に増大する。

ミニマックス法の問題点の対策

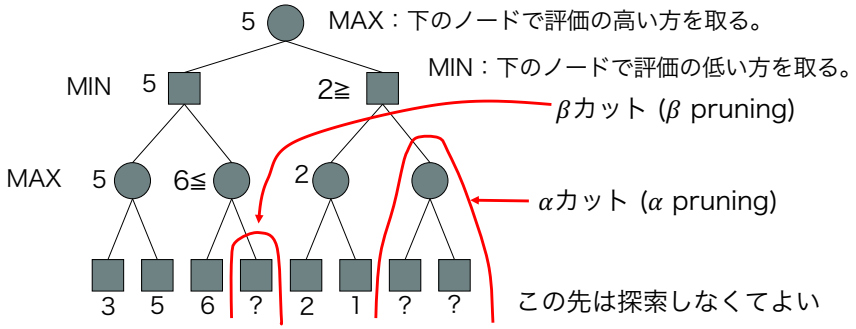
- 終端状態以前に探索を打ち切る（たとえば深さ*N*で）
- （終端状態でない）各状態を評価関数で評価する
  - そんなことできるか？ ➔ チェスでは実際に評価関数がある
- チェスの場合
  - 場面にある駒に種類に応じた点数がある。またそれらの配置に関しても点数がある。それらをまとめて点数の高いほうが有利と判定される。完全に正しくないが、かなりの指針になる（らしい）。
  - ただし次にクイーンが取られるとか、ポーンが成るとか打った次に劇的に評価関数の値が変わるかもしれない。深さ（つまり何手先まで読むか）を決めることは難しい。
- 一般にどちらが有利かを（人間なら判断できても）関数で表すことは難しい。

アルファベータ法

- 探索の深さを限定するのではなく、無駄な探索を打ち切り、その先の探索をやめることで効率を上げる。
- アイデア：　最終的に取りうる最大値(α)と最小値(β)を推定する変数を導入する。
- 例から

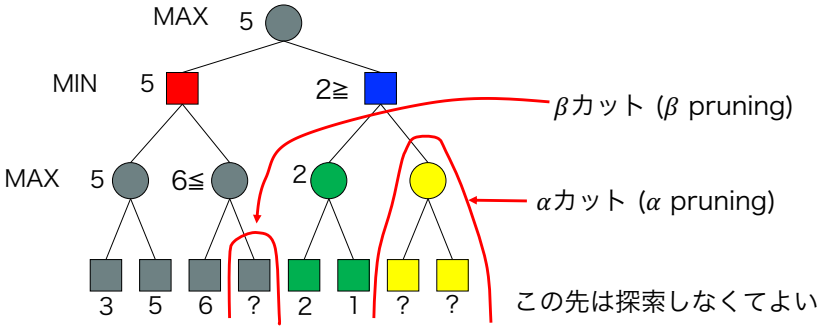
アルファベータ枝刈り

- 部分的に終端にたどり着くか、途中で評価関数を利用して不要な木の枝を刈る。このような枝刈りをする方法をアルファベータ枝刈り（alpha-beta pruning）という。アルファベータ枝狩りで探索範囲を省略したミニマックス法をアルファベータ法という。例から。



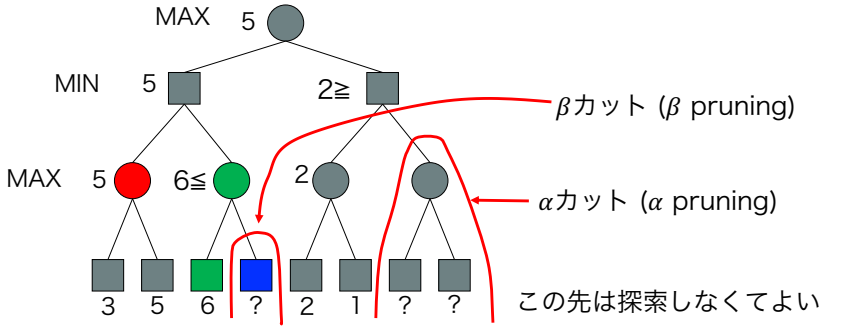
アルファベータ枝刈り

- アルファカット
  - 赤の下から5を得られる。緑の部分が２なので、相手が最善の手を打てば２を越えることはなく、黄色の部分は調べなくて良い。



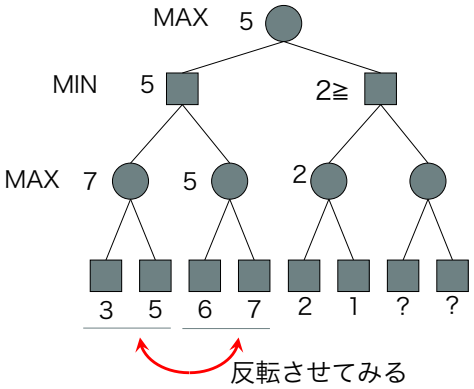
アルファベータ枝刈り

- ベータカット
  - 赤の部分が5。相手はこの値を小さくする。
  - 緑の部分は自分側で最善の手を打つので6以上。相手は不利なので緑の丸は選ぶはずはないので、青は調べなくて良い。



アルファベータ枝刈り

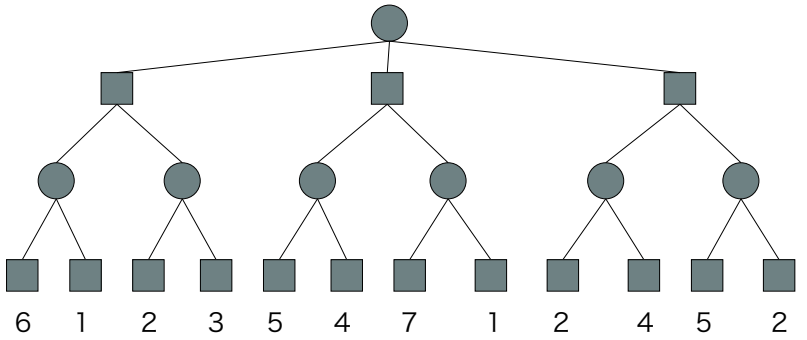
- アルファベータ枝刈りは子ノードの並びかたに依存する。



- βカットができなかった！

課題3-1

- このゲーム木をアルファベータ法を用いて探索し、選択された状態に○印、枝狩りされた部分に×をつけよ。



ゲーム木探索のまとめ

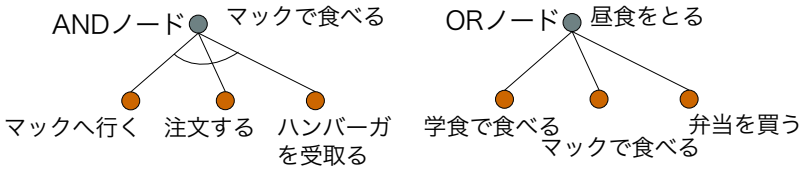
- 情報が完全にある場合にはミニマックス法により最善の手を決定することができる。
- アルファベータ枝刈りは、最終結果に関係ないゲーム木の枝の探索を省き、ミニマックス法を効率化する。
- ゲームでは最終の手まで読みきることは不可能なことも多いので、適当な状態で探索を打ち切り、評価関数とそれに基づくミニマックス法を利用する。

20 AND/OR木

別のタイプの探索木

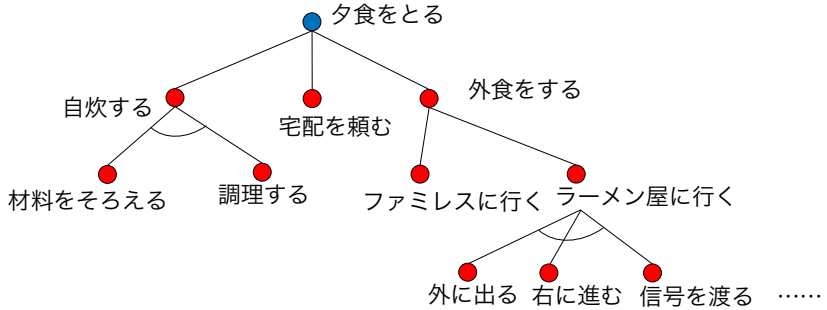
問題分割

- 問題をいくつかの小問題に分けて問解決することがある。もとの問題を親問題、分割した問題を子問題という。
- 二つのケース：
  - 子問題をすべて解決する必要がある場合⇒AND結合
  - 子問題の一つを解決すればよい場合⇒OR結合
- これを木構造でならべて問題を表現する。



AND/OR木

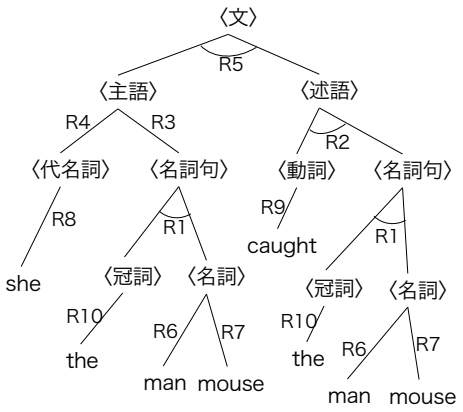
- AND/OR木とは、問題を小問題に分け、それらをANDノードとORノードの木構造で階層的に表したもの。



AND/OR木の例 ― 文章生成問題

- 以下の文法を導入：

R1: 〈名詞句〉→〈冠詞〉〈名詞〉
R2: 〈述語〉→〈動詞〉〈名詞句〉
R3: 〈主語〉→〈名詞句〉
R4: 〈主語〉→〈代名詞〉
R5: 〈文〉→〈主語〉〈述語〉
R6: 〈名詞〉→man
R7: 〈名詞〉→mouse
R8: 〈代名詞〉→she
R9: 〈動詞〉→caught
R10: 〈冠詞〉→the



AND/OR木の例 ― 経路探索

- たとえば高田馬場から大阪に行く
  - OR (新幹線で行く、高速バスで行く、飛行機で行く)
  - 新幹線で行くなら

    OR(高田馬場→東京、高田馬場→品川、高田馬場→新横浜)

    青の部分は、
    AND (OR (高田馬場→山手線→新宿→中央線→東京駅、
    高田馬場→山手線→池袋→丸の内線→東京駅、
    高田馬場→東西線→大手町→丸の内線→東京駅)
    OR (のぞみ1号、のぞみ3号、のぞみ5号…….))

    という感じ。中には受理できない（時刻の制限・希望などから）もある。これはあとで評価し削除。

AND/OR木の探索

- これまでの探索木は、ゴールノードとなる探索木の末端を探すことだったが、AND/OR木の場合は、次のような部分木Gを探し出すことにある。この部分木を解グラフ（solution graph）という。
  - A) Gは初期ノード（頂点）を含む
  - B) Gに含まれるANDノードのすべての子ノードもGに含まれる
  - C) Gに含まれるORノードの子ノードは一つだけ含まれる
  - D) Gの末端ノード（リーフ）は、はじめのAND/OR木の末端ノードであり、解の一部となりうるものである（このようなノードをAND/OR木の目標ノードと呼ぶ）。

AND/OR木の探索アルゴリズム

- アルゴリズム

Step1: 頂点だけのグラフを作り、このグラフをOPENリストに加える。

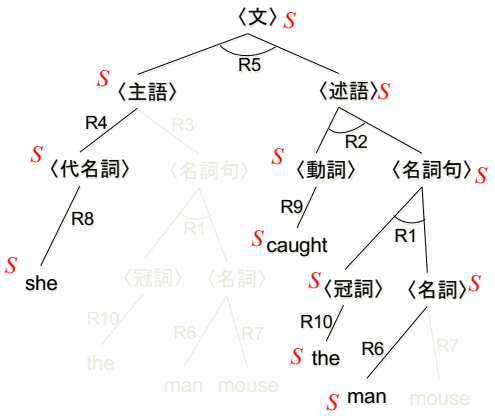
Step2: OPENリストが空なら失敗として終了。OPENリストからグラフを一つ選択しgとする。gが解グラフとなっていれば成功としgを出力する。

Step3: gを展開（次のスライド）しグラフを生成する。生成されたグラフを評価（次のスライド）し、残ったグラフをOPENリストに加え、Step2へ。

AND/OR木の探索アルゴリズム

- 評価法：
  - gの各リーフを調べ、当初のAND/OR木の目標ノードであればSのラベル、それ以上展開できないノードであればNのラベルをつける
  - gのリーフ以外のノードに関しては、もしそれがANDノードであれば、そのすべての子ノードにSのラベルがあればSのラベルをつけ、一つでもNがあればNのラベルをつける。ORノードの場合には、その子ノードと同じラベルをつける。
  - 頂点がSなら解グラフ、Nなら評価の結果としてこのグラフは排除される。なおNは受け入れられないノードや、探索の限界などで途中で切られたもの。
- 展開法：
  - gの各リーフについて、ANDノードであればその子ノードをすべて加える。ORノードであれば、その子ノードを一つづつ加えたグラフを子ノードの数だけ作る（つまりグラフの数が増える）

アルゴリズムの動作例



- 左のR4を展開するときに、同時にR3も展開され二つのグラフができる。ここでは後者のグラフは省略したが、実際にはアルゴリズムではOPENリストに加えられ、別の文もできる。

AND/OR木のまとめ

- 問題をいくつかの小問題に分割して問題を解決する。その際に、全体の問題はAND/OR木として表現できる。
- AND/OR木の探索は、問題を分割したものから全体を構成する必要があるので、単にある状態を探索するのではなく、AND/OR木から解グラフという部分木を発見する過程となる。

30 ゲーム理論入門

協力・調整・対立の構造

- 2エージェント（2人、2グループ）が独立にある戦略をとる。相手の戦略は自分の結果にも影響する。
- 各行動を適切に選択することも重要だが、そもそも何を目指して行動を取るべきかの判断は根本的な課題。
  - 単独なら明確。複数なら相手の行動に依存。
  - 相手の過去の行動にも依存（相手の行いを学習して決める）
  - これまでのゲームは敵対。実社会は多様。たとえば、
    - エージェント間のwin-win（協力、協調）の関係
    - 対立構造。しかし共に最善手を選ぶとエージェント間で対立するが、どちらかが2番目に良い手だと、両者がある程度利益を得られるかもしれない。この関係には今後のこともあるかもしれない。
    - 自分の行動コストをさほど下げること無しに、他のエージェントの利益を上げられるなら、協力するか？（調整）

ゲーム理論

- ゲーム理論とは、 ２組以上のプレイヤーの合理的・戦略的意思決定を記述する数学的枠組み。
  - ゲーム理論は、 フォン・ノイマンとモルゲンシュタインの「ゲームの理論と経済活動」（1944年）が始まり。フォン・ノイマンは、コンピュータの父とも言われている。
  - 経済学、社会学、生物学、防衛学などの理論的枠組みとしも使用
  - 計算機科学での研究も歴史はあるが、インターネットにより、多くのプログラムが相互に影響し合うようになり特に研究が活発化した。
    - たとえば、価格決定：航空券は、その売り上げ量に応じて価格を変動させるが、他方、ライバル会社の存在もある。その状況での価格決定は、エージェントによる自動的な判断による。電力も要求量に応じた価格変動制だが、やはりライバル会社との価格競争もあり、これらを総合的に自動判断する。Googleなどの検索時の広告位置と広告費の動的決定も自律エージェントとゲーム理論の応用。株式の自動取引など。

ゲーム理論

- 計算機科学では、2者、あるいはそれ以上のエージェント間のインタラクション構造（関係）を表現し、最適な戦略を決定する。
  - ここでエージェントは、組織、会社、グループ、個人などの代理、あるいは、ロボットや自律走行機械の制御ソフトウェアなど。
- 戦略とは
  - これまでゴールを目指して、その状態を実現するオペレーションの列を生成するために、そこに至る経路を探索した。
  - 何をゴールにするか？これは根本的問題。特に、相手がいる場合。
  - 協調・調整・敵対。それによりゴールも変わる。従ってここで戦略とは、オペレーションの列や次の一手ではなく、目指すゴール状態や相手への対応の基本方針を決定すること。
  - そのためゲーム理論では行動より戦略という言葉がよく使われる。

ゲームの表現

- プレイヤ（＝エージェント） P_1, P_2が戦略としてS_1, S_2を持つとき、戦略の組合せにより、P_i がa_i, b_i, c_i, d_iのいずれかの利得を得る。ここでa_iなどは実数（整数のことが多い）とする。
- 右のような行列を利得行列 (payoff matrix) と呼ぶ。

P_1 \ P_2	S_1	S_2
S_1	a_1, a_2	b_1, b_2
S_2	c_1, c_2	d_1, d_2

- 各プレイヤは一回だけ同時に戦略を選択する。そのとき利得行列で対応する利得を得る。大きい利得の方がうれしい。
  - ただし利得は広い意味であり、収入などの直接的な利益に限らず、「こうなって欲しい」という好ましさを表すもの。
- 各プレイヤは、自分の手と相手の手を予想して、自分の利得が増えるような戦略を考える。

ゲームの例

- 右のような利得行列。
  - 新聞社1 (P_1)と新聞社2 (P_2)がある。 80%人が一面に経済ニュースの新聞 (S_1) を買い、20%の人が一面にスポーツの新聞 (S_2) を買うとする。利得の値は10% = 1として表示。
  - 各プレイヤのの利得の成分を取り出した行列を、 M_1, M_2としたとき、
    $$M_1 = M_2^T$$
    なら対称ゲームと呼ぶ。それ以外の場合を非対称ゲームと呼ぶ。上記は対称ゲームの例。(Tは転置)
  - P_1からすると、相手がS_1とするとP_1はS_1，相手がS_2とするとP_1はやはりS_1の方がよい。どちらも同じ結果S_1となる。
  - これはMinimax法（と同じ考え方）
    - 注意：利得行列の数値はさほど重要ではない。相対的関係が重要。たとえば、上記の例で8を7としても戦略は同じ (6=60%にすると残りが4となり変わるかも)

P_1 \ P_2	S_1	S_2
S_1	4, 4	2, 8
S_2	8, 2	1, 1

合理性と支配戦略均衡

- プレイヤの合理性 (rationality, rational agent)：
  - 各プレイヤは自分の利得を最大にするように努力をする (Self-interested agentも同じような意味合いで使われ、それぞれの観点から最大限の利得を得ようとすること。まれにselfish agentとも)
- 支配戦略 (dominant strategy)
  - 相手が如何なる手を打っても、他よりは利得が高くなる戦略。
  - 合理的プレイヤなら必ず取る手。
  - 新聞社の例では、 S_1が支配戦略となる。この例のようにS_1はS_2を取ったときより利得が常に大きい。この場合、強支配戦略と呼ぶ。等号が有る場合には弱支配戦略と呼ぶ。
- 支配戦略均衡 (dominant strategy equilibrium)
  - 対称ゲームで合理的プレイヤなら両者同じ戦略を選択するはず。一般に、両者が支配戦略を持てば、それをとり続ける。この組合せを支配戦略均衡という。新聞社の例では、(S_1, S_1) がこれに該当。

反復支配戦略（均衡）

- 反復支配戦略（均衡）
  - 大きい豚 (BP)と小さい豚 (SP) が檻にに入っている。どちらかの豚がボタンを押すと、少し離れたところにあるえさ箱にえさが出てくる。小さい豚がボタンを押すと、えさ箱に着く前に大きな豚がほとんど食べてしまう。大きな豚が押すと小さな豚も半分食べることができる。ただ、ボタンを押しえさ箱に戻るのに2のエネルギー (−2)をつかう。
  - 大きい豚に支配戦略はない。小豚は「待つ（待ち戦略）」が支配戦略となる。従って小豚は待ち続ける。大豚はしかたなく押しに行く。
  - 一方に支配戦略があり、その結果他方の戦略も決まる場合に反復支配戦略といい、その組合せを反復支配戦略均衡と呼ぶ。
  - 相手の合理性を仮定していることに注意

BP \ SP	押す	待つ
押す	5, 1	4, 4
待つ	9, -1	0, 0

ミニマックス戦略 (Minimax)

- 可能な戦略は２種類だけではないので、一般には$m \times n$の利得行列になる。
- $a_{ij} = -b_{ij}$となるときゼロサムゲームと呼ぶ。
  【この場合、利得行列は一方のプレイヤがわかればよいので、一つの行列のみ（たとえばa_{ij}のみ）示す】

P_1 \ P_2	S'_1	$\cdots$	S'_m
S_1	a_{11}, b_{11}	$\cdots$	a_{1m}, b_{1m}
$\vdots$	$\vdots$	$\cdots$	$\vdots$
S_n	a_{n1}, b_{n1}	$\cdots$	a_{nm}, b_{nm}

ミニマックス戦略 (Minimax)

- （反復）支配戦略均衡が存在しない場合はどのような戦略があるか？ゼロサムゲームでは、最悪の状態は避けるという戦略がある。(右下例)
  - P_1は、S_2だと最悪になる可能性がある。
  - この考えだとS_3がよい。最悪でも0 。
  - この戦略は相手が最善の戦略取った（こちらにとっては最小値）として、その値が最大になるようにする。
    $max_i\ min_j\ a_{ij}\ (=0)$
  - 相手からすれば、逆で、
    $min_j\ max_i\ a_{ij}\ (=1)$
  - (jの利得は−1) となり、 S'_2となる。
  - これをミニマックス戦略という。

P_1 \ P_2	S'_1	S'_2	S'_3
S_1	4	-1	-2
S_2	-3	0	3
S_3	0	1	2

均衡点 (Equilibrium point)

- 一般には $max_i\ min_j\ a_{ij} \leq min_j\ max_i\ a_{ij}$が成り立つ。特に
  $$max_i\ min_j\ a_{ij} = min_j\ max_i\ a_{ij}$$
  となる戦略の組があるとき、その戦略の組を（ゼロサムゲームの）均衡点 (equilibrium point) と呼ぶ。前ページの例では均衡点はない。 (S_3, S'_2)
- 右のゲームでは、(S_3, S'_1) が均衡点。
  - このときは戦略が安定する。たとえば、P_1は相手がS'_2を取る限り戦略を変えると損をする。 P_2側も同じ。
  - つまり均衡点は安定点。余計なこと（作為、悪意ある操作など）を考えるインセンティブがない。
- 前の例は、相手の視点からminimaxを求め先読みでき、安定しない。

P_1 \ P_2	S'_1	S'_2	S'_3
S_1	4	-1	-2
S_2	-3	0	3
S_3	3	1	2

囚人のジレンマ

- 前ページの新聞社の例では、均衡点は個々の最適ではないが、社会的に（やや）望ましい点に見える。
  - 右の例では、均衡点は (S_2, S_2)となり、望ましいとは考えにくい。(S_1, S_1)としたい。
  - 右のような状況を囚人のジレンマといい、S_1は協調 $(C, \text{cooperate})$、 S_2は裏切り (D, defect)と言われる。
    - 囚人のジレンマは、自己の利益（合理性）を追求する組織や人の間で、いかに協力生み出すかという社会科学の基本問題であり、政治学、経済、心理学、生物学 (たとえば蟻の協力行動) などで活発に研究されている。
    - 計算機科学でも、エージェントがインターネットにより相互に影響しあうこと、社会実装が進んだ結果として、類似のジレンマ状態が発生し、協調を生むアルゴリズム・学習法が重要となってきた。（自動運転、カーシェアリング、空港のゲート割当問題、TCPなどがこの例）

P_1 \ P_2	S_1	S_2
S_1	4, 4	5, 0
S_2	0, 5	1, 1

囚人のジレンマ（参考）

- 共同で犯罪を行ったと思われる囚人A、Bを自白させるため、検事は2人に次のような司法取引をもちかけた。
  - もし、お前らが2人とも黙秘したら、2人とも懲役2年だ。
  - お前らのうち1人だけが自白し、共犯を認めたらたらそいつはその場で釈放してやろう（懲役0年）この場合自白しなかった方は懲役10年だ。
  - ただし、お前らが2人とも自白したら、2人とも懲役5年だ。

P_1 \ P_2	S_1	S_2
S_1	-2, -2	0, -10
S_2	-10, 0	-5, -5

- この時、2人の囚人は共犯者と**協調**して黙秘すべきか、それとも共犯者を**裏切って**自白すべきか、というのが問題である。なお彼ら2人は別室に隔離されており、相談することはできない状況に置かれているとする。（以上Wikipediaより）

繰り返し囚人のジレンマ

- 一回だけなら裏切りが最適戦略となるが、同じ相手と数回繰り返すことが分かっていれば、協調が生まれる可能性がある。(Iterated Prisoner's dilemma)
  - 協調を生みやすいアルゴリズム「しっぺ返し行動戦略」など
  - 協調する相手とは協調するが、非協調的相手との協調は無駄という考え方。
- このほか、人間社会や生物間では、寛容性、互恵性などが協調を生む基本性質であることが指摘されている。たとえば、
  Nowak, M.: Five rules for the evolution of cooperation. Science 314(5805), 1560–1563 (2006)
- AIで人とエージェントあるいはエージェント同士のインタラクションは、繰り返しと思われるので、寛容性や互恵性をプログラムとして実装するか、過去の経験から学習する必要がある。これらの知見の活用した学習アルゴリズムが重要（でもまだよく分かっていない）。

最後に (1)

- 合理性のこと：
  - 合理的戦略は、それぞれの評価尺度（たとえば人間なら収入、社会的地位、モラル、評判、友情とか）に基づいてそれを最大化させようという行為であり、利己的な戦略ばかりではない。
  - 人を助けるのが最善というのであれば、そのような利得行列を与える（か、学習させる）。
  - 利他的な状況を利得と考えること（があるが、これ）を人間は如何に獲得するのか？後天的学習か、長い進化の過程で選択されたものか？間接互恵（将来のリターンを暗に考慮）か？これには実験や数理解析からの研究が多数ある。これは社会性という知的行動の発現に必要かもしれない。

最後に (2)

- 均衡点のこと
  - これは安定点。わざと戦略的に振舞うインセンティブが存在せず、その戦略を継続する。信頼性やセキュリティ面で重要な性質である（たとえば、嘘をつくインセンティブが無くなる）。
  - ただし、安定点がいつも望ましい行為とは限らない（ジレンマ）。
- エージェントと人間（社会性と個人的な合理性）
  - 前に 「ここでエージェントは、組織、会社、グループ、個人などの代理、あるいは、ロボットや自律走行機械の制御ソフトウェアなど」と書いたが、広く捉えれば人間もエージェントとしてモデル化。
  - プログラムが合理性のみを追求したときに、協調相手として目の前の人間が相応しいかを判断するときが来るかも。他方、人間は常に合理的というわけではない（いかなる人間も非合理的行動を取りたくなる場面があるように思われる＝＞例：最後通牒ゲーム）。
  - ではどうして合理的ではないことがあるのか？