目 次

1　序論

1.1　目的

本研究では、人工知能研究の分野の１つである「ボードゲーム」、中でも低年齢向けの将棋である「どうぶつしょうぎ」の人工知能について研究した.

この将棋は、2008 年に発表されたものであり、人工知能はそれ程研究されておらず、どのような方法であれば強いものが作成できるのか不明である.

したがって、本研究ではどうぶつしょうぎの人工知能を作成するにあたり、

- 強い人工知能の作成.
- 盤面評価の方法の指針を示す.

の 2 つを目的として、研究を行う.

1.2　人工知能の概要

人工知能 (artificial intelligence) とは、大辞泉 [1] によると「コンピューターで、記憶・推論・判断・学習など、人間の知的機能を代行できるようにモデル化されたソフトウエア・システム.」と定義されている通り、人間の知能を模倣しようとする技術だけではなく、人間の代わりに解決させたい問題を解くための技術のことも含めて人工知能という.

人工知能は大別すると 2 種類あり、人間のように本当に知能のある人工知能を「強い人工知能」、知能があるようにみせる「弱い人工知能」がある．現在行われている人工知能研究のほとんどは弱い人工知能の研究である [2].

チェスや将棋の人工知能は弱い人工知能の一種であり、本研究も弱い人工知能についての研究である.

2　遺伝的アルゴリズムを用いた人工知能の理解

人工知能がどのようなものであるかを理解するために、まず人工知能の一種である「遺伝的アルゴリズム」のプログラムを作成した.

2.1　遺伝的アルゴリズムの概要

遺伝的アルゴリズム (genetic algorithm) とは、親の特徴が子へと混ざり合って遺伝されていく原理を利用した方法で、巡回セールスマン問題や機械学習などに適用されている．また、生物集団の進化過程や生体内の活動をシミュレーションする人工生命の研究にも使用されている.

遺伝的アルゴリズムでは、個体の遺伝子の集まりである染色体を文字列で表現する．染色体内部の表現方法を「遺伝子型」と呼ぶ．遺伝的アルゴリズムを利用する際には遺伝子型をどのように表現していくかを決めてやる必要がある．また、個体それぞれが環境への適応度を計算する方法も決めてやる必要がある．本研究では、規則的に動く青丸の動きに最初はランダムに動いている赤丸が青丸の動きに近付いていくというプログラムを作成した．赤丸の動きを決定していく際に遺伝的アルゴリズムを適用した．このプログラムでは、赤丸の軌跡を 20 分割し、それぞれの地点での x 座標を遺伝子とし、20 分割された座標値を格納した配列を染色体とした.

遺伝子型や適応度の計算方法を決定し、プログラムに落とし込んでいく際、遺伝的アルゴリズムの流れは 4 段階ある [3].

1. 第 1 世代の誕生（初期化）
2. 適性評価
3. 選択
4. 進化

まず、1 つ目の段階では初期集団を発生させる．次に適性評価のプロセスでは、各個体が環境に適応しているかどうかの適応度を計算し、ランク付けを行う．選択プロセスでは、適性評価でのランク付けを元に次世代へと残す個体を選択する．進化プロセスでは、選択された個体間の染色体情報を交叉させる．交叉することによって、次世代の個体が最適解に近い個体を得られる可能性が高くなる.

適性評価から進化のプロセスを何世代も繰り返していくことによって、より最適解に近い解を求めていくことが遺伝的アルゴリズムはできると期待される.

2.2 実験

遺伝的アルゴリズムがどのようなものかを理解するため、図 1 のような遺伝的アルゴリズムを用いたプログラムを作成した. 画面上では青丸と赤丸の 2 つが左右に動いている. 青丸は x と t を定義し、$x = \sin t$ と規則的に動く. 赤丸はランダムに作成された個体群の中から、現世代で最も環境に適応している個体の動きをする. このプログラムでは、青丸の動きにどれだけ赤丸が近付いていくか、何世代で誤差の縮まり方が収束するかを検証した.

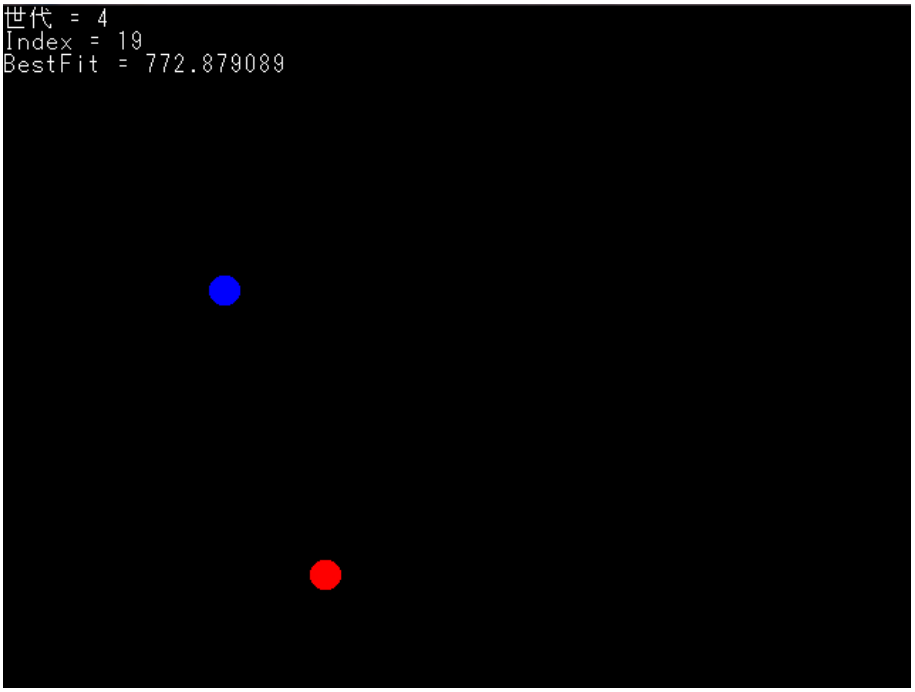

図 1: 遺伝的アルゴリズムのプログラム実行画面

2.2.1 方法

このプログラムでは、第一世代を誕生させる際に個体を 20 体作り、2 秒経過すると次世代へと変わっていく．丸の軌跡を 20 分割し、分割したそれぞれの区間での x 座標の値を格納した一次元配列を染色体として保持している．

青丸は $x = \sin t$ に $t = 0.1, 0.2, \cdots\cdots 1.9, 2.0$ と 0.1 刻みでその時点での x 座標の値を配列に格納している．赤丸は [-20,20] の区間の値が発生する乱数によって生成した値 20 個を染色体として持っている．

適性評価の方法は、式 (1) のように n 秒の時点での青丸の x 座標の値と赤丸の x 座標の値の差の総和を求めることで評価している．

$$S = \sum_{k=0}^{19} (a_k - b_k) \tag{1}$$

(但し、a は青丸の x 座標、b は赤丸の x 座標、k は添え字である)

それぞれの個体について式 (1) を用いて評価をしたら、次にその評価を元に次世代へと残す個体を選択し、その個体を親として子供を作る．

子供は親同士の遺伝子を一点で交叉させることによって作られ、また遺伝子に突然変異を起こさせることによって染色体の一部をランダムに入れ替えることで生成している．淘汰された遺伝子は新しく作られた子供によって補われるので個体数は常に 20 体となる．

1. 適性評価
2. 選択
3. 交叉
4. 突然変異

上記の過程を 1 世代とし、何世代も繰り返すことで青丸の動きに赤丸の動きが近付いていく．

2.2.2　結果

まず、選択プロセスの際に次世代へと残す個体数を 1 体と 2 体の場合を考える．また、それぞれの場合で突然変異を起こす確率を 1 %、3 %、5 %と変化させた．世代数は 1000 世代までとした．それぞれの結果を図 2、図 3 に示した．

グラフの縦軸はその世代で最も環境に適応した個体と青丸との差の総和、横軸は世代数を表している．

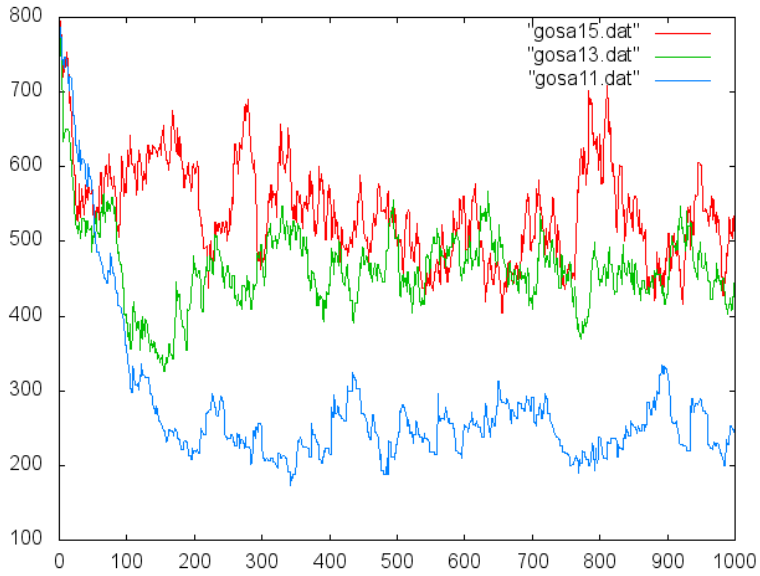

図 2: 次世代へ残す個体が 1 体の場合（突然変異率は青：1 %緑：3 %、赤：5 %、縦軸：誤差の総和、横軸：世代数）

2 つのグラフを見ると両方とも突然変異率が 1 %のときのものが最も差の総和が小さくなっている．これは、突然変異率が高いとそれまでの世代で蓄積されてきた優秀な個体が、突然変異によって異なったものになってしまう．その為、ランダムに個体を生成している様な状態になるので 1 %のものに比べて、3 %、5 %と突然変異率が高くなるに従い差の総和の値が大きくなっていってしまっていると考えられる．

次に 2 つのグラフを比較すると、選択プロセスの際に次世代へと残す個体の数が 1 体のものと比べて、2 体の方が 1000 世代のときに差の総和が小さくなっている．

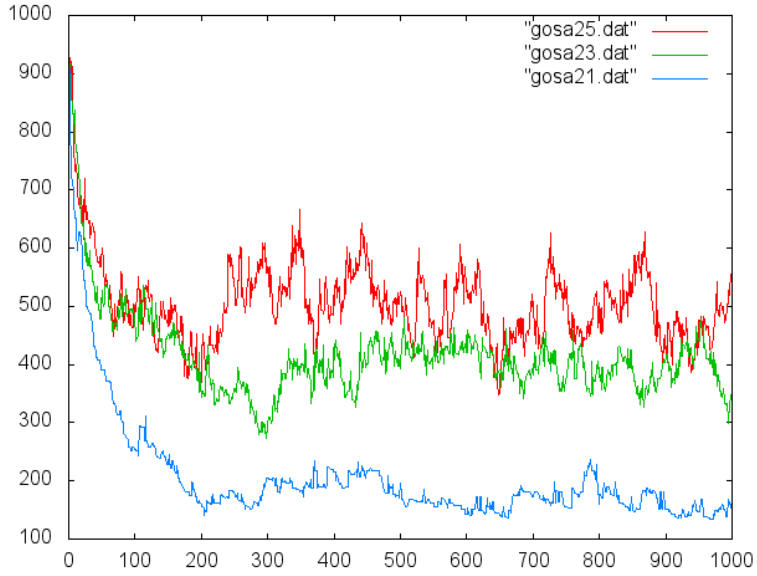

図 3: 次世代へ残す個体が 2 体の場合（突然変異率は青：1 %緑：3 %、赤：5 %、縦軸：誤差の総和、横軸：世代数）

このプログラムでは、選択の際に「エリート選択」と呼ばれる適応度が高い個体を上位 1 個体もしくは 2 個体を選択する手法を利用している．しかし、エリート選択では染色体の多様性が失われてしまうという欠点を持っている．その為、選択する個体を上位 1 体とした際に、他の個体よりも圧倒的に適応度が高い個体が現れた場合にはその個体からの変化が乏しくなってしまう．

結果、上位 2 体を次世代へと残していく場合に比べて、グラフの収束が早い段階で終わってしまい、それ以降は差が縮まりにくくなってしまったために、上位 1 体を残していく方が差が大きくなってしまったと考えられる．

また、次世代へと残す個体の数が 1 体の場合も 2 体の場合も、両方とも赤丸の動きは青丸の動きにある一定までは近付いたものの、完全に同一の動きにはならなかった．これは、最適解として得られたものが必ずしも厳密解ではないことが分かる．

3　ボードゲームにおける人工知能

3.1　概要

将棋やチェスの様な偶然に左右されることのないゲームのことを「二人零和有限確定完全情報ゲーム（以下、完全情報ゲーム）」という [4]．それぞれの用語の意味は

二人
　ゲームを行うプレイヤーが 2 人．

零和
　ゲームをプレイしているプレイヤー全ての利得の総和がゼロになる．

有限
　そのゲームで各プレイヤーの考えられる手の組み合わせが有限である．

確定
　プレイヤーの打つ手以外にゲームに影響を与える要素が無い．

完全情報
　各プレイヤーが自分の手番の際に盤面を見ることでそれまでのゲームの推移を知ることができるゲーム．

である．

ボードゲームの人工知能の基本として、局面が自分にとって有利であるか不利であるかを評価することである．局面の評価を適切に行い、点数として表すことで、打てる手の中から自分にとって有利な手を選択すればよい．しかし、盤面の評価だけでは適切な局面評価はできない．そこで利用されるのが、完全情報ゲームの人工知能のアルゴリズムを用いた「先読み」である．

主だったものとして、「ミニマックス法」、それを応用した「アルファベータ法」がある．どちらのアルゴリズムも現在の盤面の状態からプレイヤーの打った手によって、どのような局面が現れるかを場合分けした「ゲーム木」を作成し、今後どのような局面が現れるかを想定し、それを考慮しつつ、次に打つ手を決定する．

3.2　評価関数

評価関数とは、ゲームの局面を評価する際に用いられる関数である．評価関数によって、局面の自分にとって有利不利を判断し数値化する．評価関数の値が正の値であれば自分にとって優勢、負の値であれば劣勢であることを表している．評価関数の値を元に指し手を決定する．

3.3　ゲーム木

図 4 のように、ゲームの盤面を木構造の節点とし、プレイヤーの指し手を辺で表したものである．

また「完全ゲーム木」は、あるゲームにおいて、想定される全て盤面を含んだゲーム木である．完全ゲーム木があれば、そのゲームを解くことができる．解に従って指していくことで、負けないことを保証できる．

ゲーム木は人工知能にとって重要なものであり、ゲーム木を探索していくことで最善手を得ることができる．三目並べなどの盤面の小さなゲームであれば考えられる場面数は少なく、ゲーム木も小さなものとなり探索が容易であるが、チェスや将棋などの盤面の大きなゲームであれば、大きなゲーム木となり、探索が難しくなる．

そのような場合に完全ゲーム木の代わりに用いられるものは「部分ゲーム木」である．部分ゲーム木は現在の盤面から指せる手を含んだ木である．

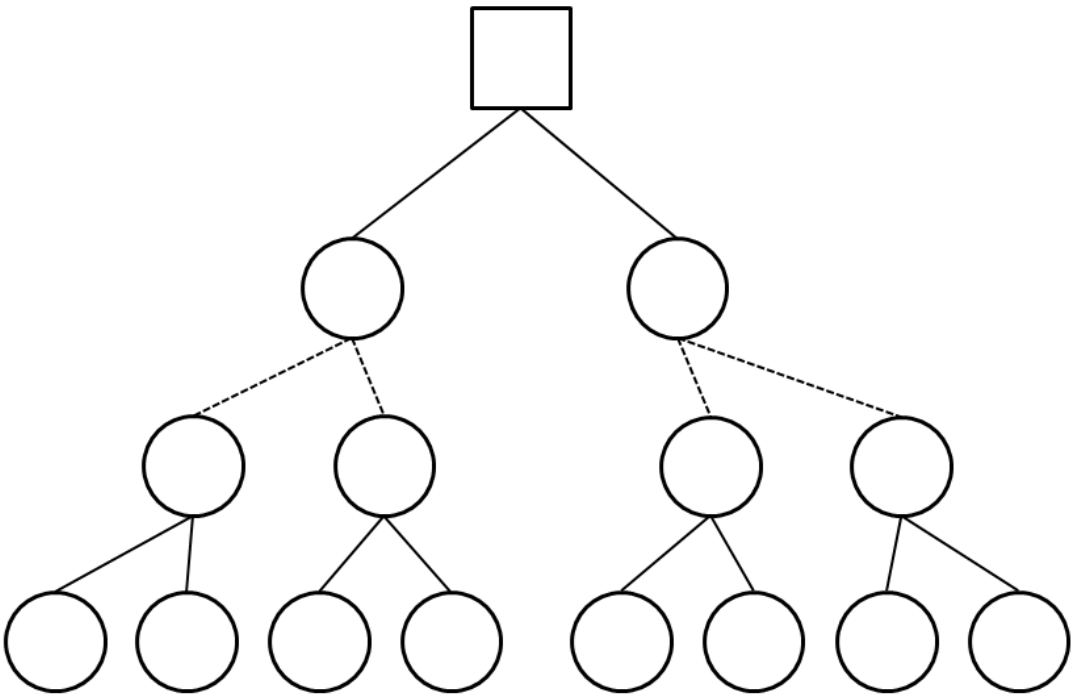

図 4: ゲーム木の例

3.4　ミニマックス法

「ミニマックス法」とは、相手から受ける被害を最小にしつつ、最善の手を打つアルゴリズムである.

考え方の基本として、

- 自分が手を打つ時には自分にとって最も有利な手を打つ (Max)
- 相手が手を打つ時には自分にとって最も不利な手を打ってくる (Min)

と仮定し、それらを交互に繰り返すことで、自身が次に打つ手を思考するものである.

図 5 のように 3 手先読みする場合について考えていく. ただし、図中の評価値は例である. 図の四角を現在の局面とし、実線の辺を自分の指し手、破線を相手の指し手、丸を手を指した後の局面とする.

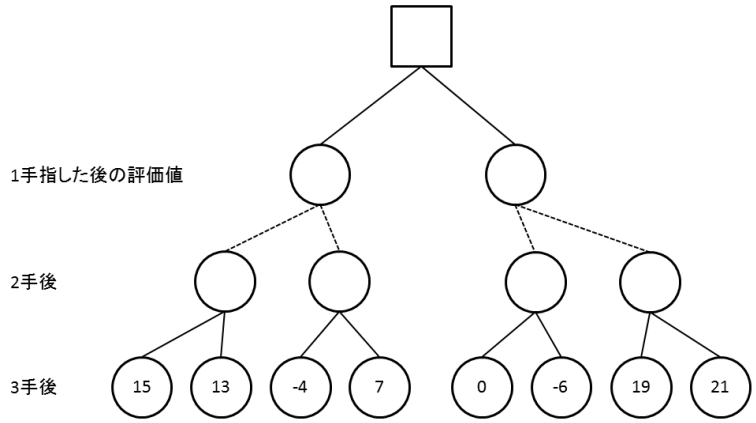

図 5: ミニマックス法のゲーム木の例 (3 手先読みする場合、評価値は例である)

まず、3 手指した後の局面を評価し、その局面での評価値を出す. 3 手目の手番は自分であるので、自分にとって最も有利な手＝評価値の値が高いものを選択し、それを 2 手目の評価値とする. すると、図 6 のようになる.

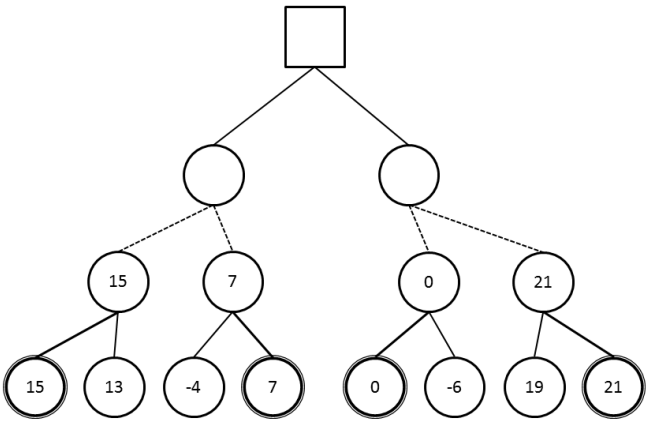

図 6: 2 手指した後の局面を評価したゲーム木（3 手目の評価値から max を選択）

次に、先程出した評価値を元に 1 手目の評価値を求める．2 手目の手番は相手であるので、今度は自分にとって最も不利な手＝評価値の低いものを選択し、1 手目の評価値とすると図 6 のようになる．

続いて、同様に 1 手目の評価値を元に、指し手を決定する．1 手目の手番は自分であるので評価値の高いものを指し手とすると、1 手目の評価値が「7」となるものが選択された．

ミニマックス法は、先読みの手数によってゲーム木の探索を打ち切ることができるが、先読みの手数を増やしていくことで探索しなければならない局面数が爆発的に増えていく．そのため、先読みの手数をある程度までにしないと、実用で使用することはできない．

3.4.1 擬似コード

先手番を人工知能の手番とした際の擬似コードを示す．また、ここでの eval 関数は評価関数である．

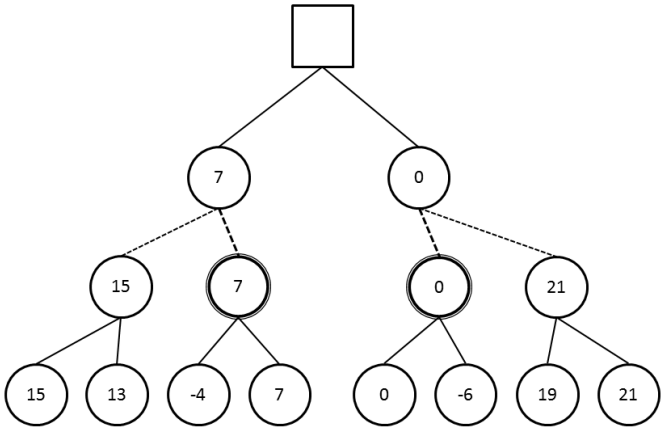

図 7: 1 手指した後の局面を評価したゲーム木（2 手目の評価値から min を選択）

```
int Minmax(局面 node, 手番 turn, 先読みの数 depth){
    if(depth == 0) eval(node);
    if(turn が先手番の時){
        int w, max = -∞;
        for(すべての子局面 n_i に対して){
            w = Minmax(n_i, 次の手番, depth-1)
            if(w > max) max = w;
        }
        return max;
     } else {
        int w, min = ∞;
        for(すべての子局面 n_i に対して){
            w = Minmax(n_i, 次の手番, depth-1)
            if(w < min) min = w;
        }
        return min;
     }
}
```

3.5　アルファベータ法

アルファベータ法は、前述のミニマックス法と同じ結果が得られるアルゴリズムであるが、ミニマックス法の短所である「盤面をしらみ潰しに探索していく」点を改善したものであり、計算しなくても同じ結果となる部分を枝刈りするものである．

例えば図 8 の場合を考えていく．図のゲーム木で「?」となっている部分は、まだ計算されていない局面だとする．

図のゲーム木では「自分→相手→自分」と手番がきているので、評価値の高いものを木の 2 層目の値とする．左のゲーム木を見ると 3 手先読みした局面で一部計算されていない局面がある．「3 と「4 と？の大きい方」の小さい方」が 2 層目の値となる．つまり、

$\min(3, \max(4, ?))$

である．これは「?」の部分がどのような値であっても「3」となる．なぜならば、$\max(4, ?)$ の部分は 4 以上であることが分かっている．そのため、

$\min(3, 4\leq)$

となり、「3」が選択されるからである．

同様に右側の木も

$\min(0, \max(1, ?)) \rightarrow \min(0, 1\leq) \rightarrow 0$

となり、最終的にこのゲーム木では

$\max(3, 0) \rightarrow 3$

となり、左側の指し手を指すこととなる．

アルファベータ法は順調にいくと、探索量が元の平方根に近い値となり、ミニマックス法に比べて探索が高速であることが知られている [5]．

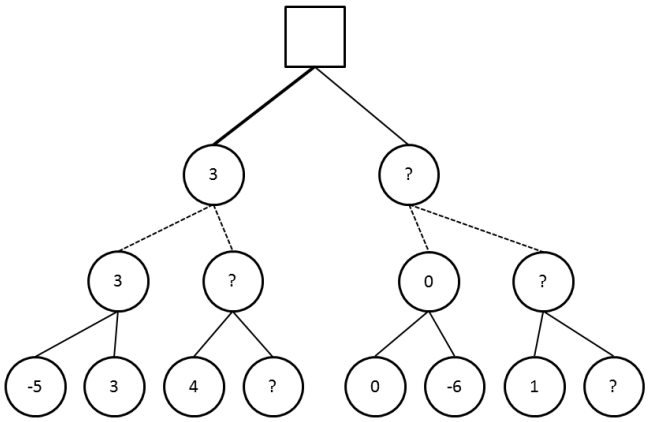

図 8: アルファベータ法のゲーム木の例

3.5.1 擬似コード

擬似コード中のαとβはそれぞれ各節点の下限と上限をであり、(α, β)を範囲として、範囲外の節点を先読みの対象から外す.

```
int Alphabeta(局　面 node, 手　番 turn, 先　読　み　の
数 depth, α, β){
    if(depth == 0) eval(node);
    if(turnが先手番の時){
        for(すべての子局面n_iに対して){
            α = max(α, Alphabeta(n_i, 次　の　手
番, depth-1, α, β));
            if(α >= β) return β;
        }
        return α;
    } else {
        for(すべての子局面n_iに対して){
            β = min(β, Alphabeta(n_i, 次　の　手
番, depth-1, α, β));
            if(α >= β) return α;
        }
        return β;
    }
}
```

最初、各節点の範囲は$(-\infty, \infty)$である．図8の例でいうと、「$\min(3, \max(4, ?))$」の節点の範囲は上限＝βが3になるので、範囲は$(-\infty, 3)$となる．この範囲外となる節点は先読みの対象から外す．このように節点の上限(β)を越えたため、先読みの対象から外すことを「βカット」といい、逆に節点の下限よりも下回り、先読みの対象から外された場合は「αカット」という.

4　「どうぶつしょうぎ」の予備知識

4.1　概要

「どうぶつしょうぎ」は、女流棋士の北尾まどか氏がルールを考案し、同じく女流棋士の藤田麻衣子氏がデザインを行い、2008 年に日本女子プロ将棋協会から発表された低年齢向けの将棋である.

低年齢層向けの将棋であるため、本将棋に比べ駒の種類も少なく、簡潔なルールとなっており、3 × 4 の盤面を使用する.

駒の種類は「らいおん」「きりん」「ぞう」「ひよこ」の計 4 種使用する.

図 9: どうぶつしょうぎ

4.2 道具

盤面は図 10 のような 3 × 4 の盤面を使用し、各列は横は左から A,B,C、縦は上から 1,2,3,4 と示され、各マスは [A4]、[C2] というように表される．1 行目と 4 行目はプレイヤー (先手, 後手) の陣地である．

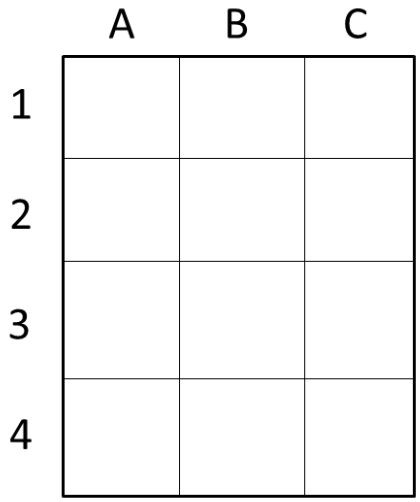

図 10: どうぶつしょうぎの盤面

駒は 4 種類を先手、後手にそれぞれ 1 つずつ、計 8 つ使用する．以下に使用する 4 種の駒を挙げる．

らいおん
隣接する 8 マスに移動できる.

きりん
上下左右 1 マスずつ移動できる.

ぞう
隣接する斜めのマスに移動することができる.

ひよこ
プレイヤーから見て前に 1 マスずつ進むことができる. 相手の陣地に入ると「にわとり」に成ることができる.
「にわとり」に成ると縦横 4 マス、斜め前の 2 マス、合計 6 マスに移動することができるようになる.

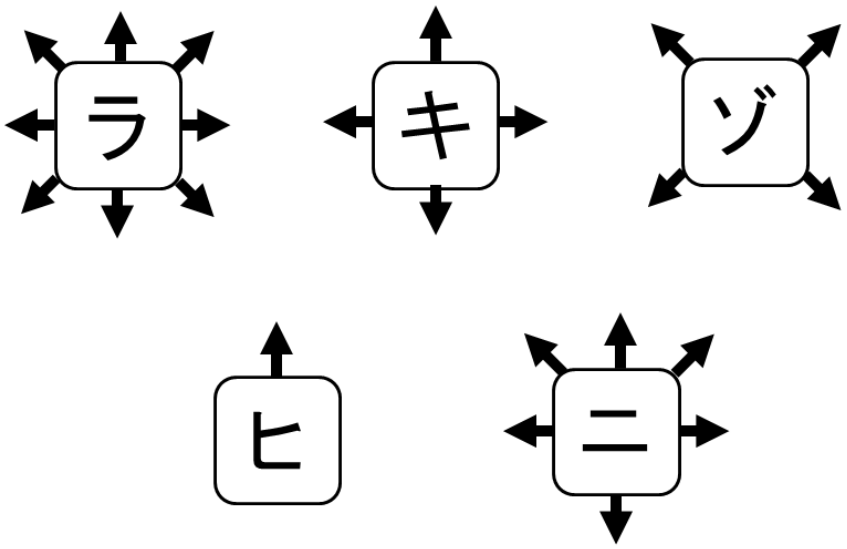

図 11: 駒動き方

4.3　ルール

最初、図 12 のような初期配置から互いに駒を打っていく．ただし、図 12 では、平仮名で表されている駒を自分の駒、片仮名で表されているものを相手の駒としている．

プレイヤーは将棋と同様に盤面に盤上の自駒を打つ、もしくは自身の持ち駒を打つといった行動を交互に行っていく．

また、駒を進める際に、進めたいマスに相手の駒があった場合には相手の駒を取り、自分の持ち駒にすることができる．

そして、相手のらいおんの駒を取る「キャッチ」、または相手の陣地の 1 番奥までらいおんを進める「トライ」をすることで勝利となる．

また、「ひよこ」は相手の陣地の 1 番奥まで進めることで「にわとり」と成る．

将棋のように禁じ手はなく、「二ひよこ」(将棋の二歩に相当) や「打ちひよこ詰め」(将棋の打ち歩詰めに相当)、敵陣の 1 番奥にひよこを打つこと (打った駒の利きがなく動けない駒) も禁じ手とならない．

また、「千日手」（手番が同じ状態が 3 回現れる）となると引き分けとなる．

	A	B	C
1	キ	ラ	ゾ
2		ヒ	
3		ひ	
4	き	ら	ぞ

図 12: 駒の初期配置

4.4 完全解析の結果

「どうぶつしょうぎ」は東京大学情報基盤センターの田中哲朗准教授によって、すべての局面の勝敗を求める完全解析が行われている [6]．発表された論文によると「どうぶつしょうぎ」は両者が最善手を打った場合、後手必勝であり、勝ちに要する手数は 78 手であると解明されている．

5　「どうぶつしょうぎ」の人工知能の作成

どうぶつしょうぎの人工知能のアルゴリズムとして「ミニマックス法」を使用した．現在主流であるミニマックス法を応用した「アルファベータ法」を使わなかった理由は、本将棋であれば1局面に平均80手程度の着手可能手が出てくるが、どうぶつしょうぎは1局面平均10手程度と少なく、6手先読みする程度ならばミニマックス法でも実用するには問題のない速度であったからである．

評価関数によって局面の評価値を出した際に、最終的に評価値が同じ手が複数あった場合にはランダムに選択した．

5.1　評価関数

局面の評価には、以下の指標を用いた．

- 駒得
- 王手、王手逃れ
- 利きの数

5.1.1 駒得

駒得とは、それぞれの駒に得点を付け、盤面上の駒、持ち駒をそれぞれ計算し、その総和を評価値とする．本研究ではそれぞれの駒の得点を以下のようにした．

盤上の駒

らいおん　50 点

きりん　5 点

ぞう　5 点

ひよこ　4 点

にわとり　8 点

持ち駒

きりん　3 点

ぞう　3 点

ひよこ　1 点

相手の駒の場合は上記の得点にマイナスを掛けたものを使用した．

5.1.2 王手、王手逃れ

王手する手を良い手とし、逆に王手される手を悪手とするものである．王手する手は 500 点、王手される手は-500 点として評価した．

5.1.3 利きの数

「利き」とは、ボードゲームにおいて駒が次に動けるマスのことである．盤面にある自身の駒のそれぞれの利きの総和を求める．また、相手の駒の利きの総和も求め、自身の利きの総和から相手の利きの総和を引いたものを評価値として使用した．

5.2　人工知能同士の対戦

5.2.1　「駒得」の評価関数

まず、駒得のみを使用した評価関数を使用した人工知能同士で対戦を50戦ずつ行わせた．駒得の各駒の得点は5.1.1節のものと同じものを使用した．

盤上の駒

らいおん　50点

きりん　5点

ぞう　5点

ひよこ　4点

にわとり　8点

持ち駒

きりん　3点

ぞう　3点

ひよこ　1点

相手の駒の場合は上記の得点にマイナスを掛けたものを使用した．

対戦結果は表1に示した．

表 1: 先手側の勝率（%、縦が先手の先読み数、横が後手の先読み数）

		先手			
		2手	3手	4手	5手
後手	2手	-	86	58	74
	3手	54	-	72	74
	4手	70	68	-	80
	5手	72	72	68	-

どの手数も先手側の方が勝率が高い理由として、仮に先手側がまず最初に [B2 ひよこ] という手を指したとする．(図 13)

すると、次に後手側はらいおんを先手側のひよこにキャッチされない位置に移動させなければならないが、「王手、王手逃れ」を考慮していないため、必ずしもらいおんをキャッチされないような位置に指すとは限らない．したがって、後手側がらいおんをキャッチされない位置に動かさずに別の駒を動かしてしまう場合が多くなり、先手側のひよこにらいおんをそのままキャッチされてしまい、後手側の負けとなるパターンが多くなる．そのため、先手側の方が先読みする手数に関わらず、勝率が高くなっていると考えられる．

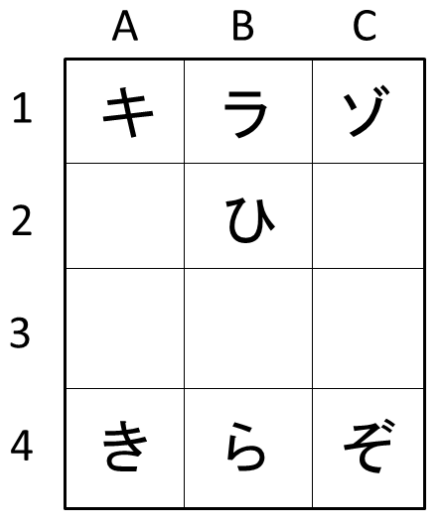

図 13: [B2 ひよこ] を指した後の盤面

5.2.2　様々な評価関数を搭載した人工知能同士の対戦

次に以下の組み合わせの評価関数を 3 種類作成し、先読み手数を 2 手先読みとし、それぞれを 100 局ずつ対戦させた．結果は表 2、表 3 に示した．

1. 駒得

2. 駒得+利き

3. 駒+利き+王手&トライ

表 2: 先手側の勝率 (1. 駒得、2. 駒得+利き、3. 駒+利き+王手&トライ)

		先手		
		1	2	3
後手	1	36 %	9 %	22 %
	2	6 %	42 %	7 %
	3	49 %	2 %	0 %

表 3: 後手側の勝率 (1. 駒得、2. 駒得+利き、3. 駒+利き+王手&トライ)

		先手		
		1	2	3
後手	1	49 %	30 %	63 %
	2	39 %	0 %	53 %
	3	43 %	52 %	100 %

表を見ると先手側よりも後手側のほうが勝率が高くなっている．この理由として、先手側が最初に打つ際にゲーム木で評価値が同じ値となる部分が多くなるためと考えられる．同じ値となるノードが複数あるために、どの手を打てば、その後の局面を有効に進めていくことが出来るかということが分からない為に、先手側は最終的にランダムに手を打つことになってしまい、有効な手を打つことができないために後手側の方が勝率が高くなったのではないかと考えられる．

また、棋譜を見てみると 1 対戦での棋譜は数パターン程度になっている．これはどうぶつしょうぎの盤面が小さいため、想定される局面数が少ないことと、序盤の局面の評価値が同じ値となる場合が多い為と考えられる．Minmax 法はランダムに手を打つのではなく、あるゲーム木である時、この手を選択するというのが決まっている．先手側が初手に打てるパターンは 4 パターンしかない．その 4 パターンの内から、どのような手を打つかによって、それ以降 Minmax 法によって選択される手がある程度決まってくると考えられる．そのために先手側は初手に打った手によって、勝つ棋譜パターン、もしくは負ける棋譜パターンに分かれるために初手をランダムに打ってしまうことの影響を大きく受けていると考えられる．

6　遺伝的アルゴリズムを適用させた「どうぶつしょうぎ」の人工知能

遺伝的アルゴリズムを用いることで「どうぶつしょうぎ」の人工知能は強くなるかを調べた.

6.1　方法

遺伝的アルゴリズムを適用させる対象として、「駒得」のみを評価関数として用い、2手先読みしていく人工知能を使用した．染色体として駒得の各駒の得点を保存した配列を使った．適用の流れは図14に示した.

まず第一世代として、駒得の各駒の得点が異なった人工知能を5つ作成する．作成した人工知能を先手・後手を交互に入れ替えながら総当り戦で対戦させ、その対戦結果で1位と2位となった人工知能の染色体を交配させる（適応度の評価・選択・交叉）.

そして、交配させてできた人工知能を5つ作成し、それぞれの染色体に突然変異を3%の確率で起こさせる．そのようにしてできた5つの人工知能で再び総当り戦を行う.

以上の流れを何世代も繰り返し行うことで、どのような得点で各駒を評価するのがよいのかを遺伝的アルゴリズムを用いて調べた.

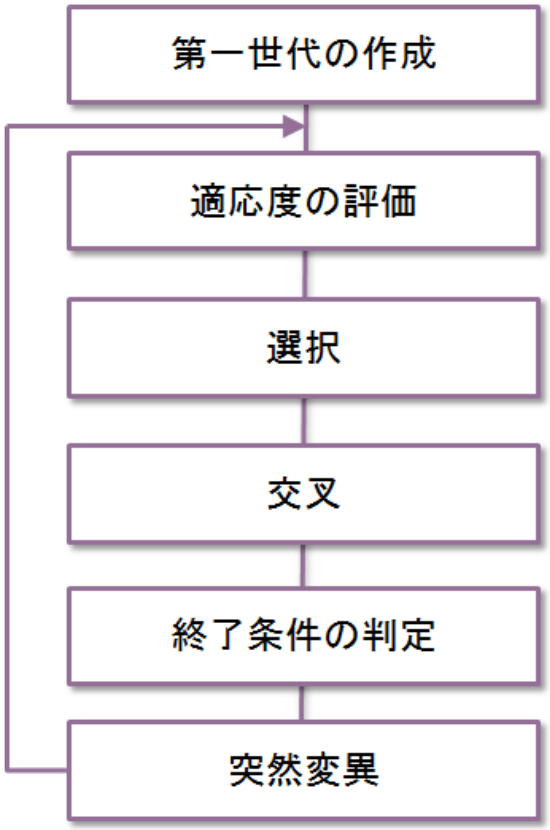

図 14: 遺伝的アルゴリズム適用の流れ

第一世代の人工知能を作成する際に、各駒の得点の基準として表 4 の点数を用いた.

表 4: 基準とした駒得の得点

	ヒヨコ	キリン	ゾウ	ライオン	ニワトリ
盤面上の駒	4	5	5	50	8
持ち駒	1	3	3	100	-

これらの点数に一様実乱数 [0,1] を掛けたものを第一世代の染色体とした.

また、評価値の収束条件として、100 世代ごとに過去 100 世代の評価値の標準偏差が 30 以下になったら収束したとし、それ以降は値を固定した.

6.2 結果

遺伝的アルゴリズムを適用させた結果、1100 世代で全ての値が収束した. その結果を表 5、表 6 に示す.

表 5: 遺伝的アルゴリズムによる評価関数値の変化

	ヒヨコ	キリン	ゾウ	ライオン	ニワトリ
盤面	$4 \rightarrow 286$	$5 \rightarrow 137$	$5 \rightarrow 202$	$50 \rightarrow 63$	$8 \rightarrow 189$
持ち駒	$1 \rightarrow 266$	$3 \rightarrow 293$	$3 \rightarrow 129$	$100 \rightarrow 16$	-

表 6: 評価関数値が固定された世代数

	ヒヨコ	キリン	ゾウ	ライオン	ニワトリ
盤面上の駒	900	200	800	800	900
持ち駒	900	900	1100	200	-

持ち駒のニワトリは、盤面上のニワトリが持ち駒となる際にはヒヨコとして持ち駒になるので、値は初期より 0 とした.

また、盤面のヒヨコの評価関数値の変動を見るとヒヨコの得点が固定された 900 世代付近で値が収束していっていることが分かる（図 15）.

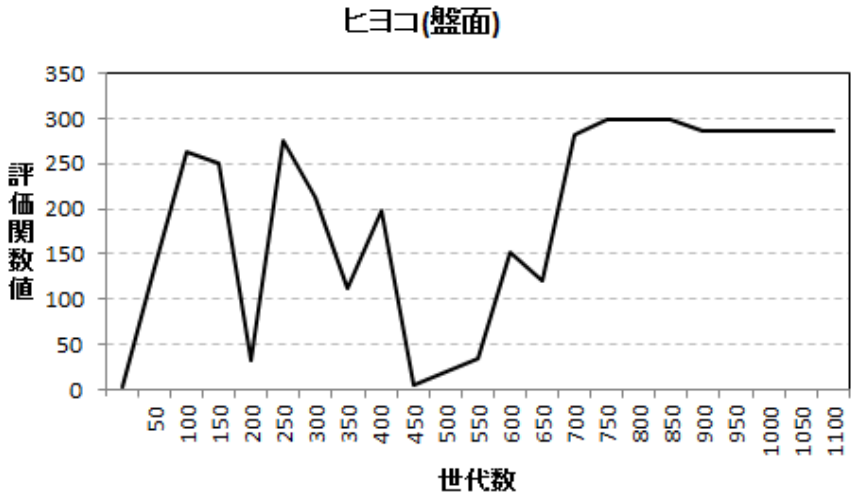

図 15: ヒヨコ（盤面）の評価値の変動

遺伝的アルゴリズムを適用した結果が表 5 のようにヒヨコが高得点となったのは、ヒヨコを活かした戦い方をしているためと考えられ、その為ヒヨコは盤面、持ち駒共に高い得点となっていると考えられる．また、ライオンが持ち駒では 16 点という低い得点になっているのは 3 手先読みとしているため、ライオンを持ち駒とする＝勝敗が決まるところまで読むことがあまり無いために、ライオンの得点が低くなっていると考えられる.

学習させた人工知能と学習させる前の人工知能を先手後手を入れ替えて 300 戦ずつ対戦させると、学習させた人工知能が先手側のときの勝率は 28 %、引き分けは 17 %となり、後手側のときは勝率 56 %、引き分け 12 %となった.

表 2、表 3 と比較すると、後手側の勝率は上がった.

6.3　手番を固定して遺伝的アルゴリズムを適用した場合

6.1 節と同様の方法で、手番を変更せずに、先手に適した評価関数値を求める場合は先手の勝利した回数を求め、それをもとに人工知能の評価・選択を行うようにした．ただし、終了条件の判定はしていない．

手番を先手に固定し、1600 世代まで遺伝的アルゴリズムを適用した場合、評価関数値は表 7 のようになった．

表 7: 手番を固定して遺伝的アルゴリズムを適用した評価関数値（先手、1600 世代）

	ヒヨコ	キリン	ゾウ	ライオン	ニワトリ
盤面上の駒	116	172	378	406	384
持ち駒	268	293	275	83	-

この評価関数値を用いて、学習前の人工知能と 100 回対戦させると先手側の勝率が 55 %、引き分け 8 %となり、表 2、表 3 と比べて、勝率が上がっていることが分かる．

また、後手に手番を固定した場合は、表 8 のようになった．

表 8: 手番を固定して遺伝的アルゴリズムを適用した評価関数値（後手、1600 世代）

	ヒヨコ	キリン	ゾウ	ライオン	ニワトリ
盤面上の駒	160	155	59	138	138
持ち駒	0	420	222	468	-

この評価関数値を用いて、学習前の人工知能と 100 回対戦させると後手側の勝率が 59 %、引き分け 12 %となり、表 2、表 3 と比べて後手の場合も勝率が上がった．

この事から、手番を固定して遺伝的アルゴリズムを適用することでそれぞれの手番をに適した評価関数値を求めることができた．

7　「どうぶつしょうぎ」アプリケーション

「どうぶつしょうぎ」のアプリケーションを AndroidOS 向けに作成した．作成する際に「Cocos2d-x[7]」というオープンソースの 2D ゲームフレームワークを使用した．

図 16: どうぶつしょうぎアプリ

7.1 「Cocos2d-x」の概要

cocos2d-x は 2D ゲームフレームワークである．２ D ゲームフレームワークとは、画像の表示やアニメーションなど、コンピュータゲームの開発に必要な共通の機能が詰め込まれたソフトウェアの事である．

cocos2d-x は、マルチプラットフォーム開発に対応しており、ソースコードや画像などのリソースを Android や iOS など、OS 毎に作成する必要が無く、1 つのソースコードで複数の OS 向けにアプリケーションのビルドを行うことができる．

cocos2d-x を利用する利点として、

- 前述したマルチプラットフォーム開発ができること．
- 開発言語が C++、Javascript、Lua の 3 つの言語で開発できること．
- オープンソース (MIT ライセンス) であること．

以上の 3 点が挙げられる．

7.2 アプリケーションの操作方法

アプリケーションを起動すると図 17 のようなタイトル画面が表示される．

「Human VS Human」、「Human VS CPU」どちらかのボタンをクリックすることで対局画面に遷移する．

「Rule」ボタンをクリックすると、どうぶつしょうぎのルール説明画面に移動する．(図 18)

図 17: どうぶつしょうぎアプリ (タイトル画面)

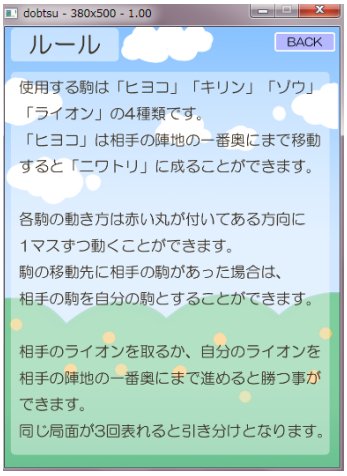

図 18: どうぶつしょうぎアプリ (ルール画面)

「Human VS Human」ボタンをクリックすれば人対人、「Human VS CPU」ボタンをクリックすれば人対人工知能でどうぶつしょうぎの対局をすることができる．また、「CPU VS CPU」ボタンをクリックすれば人工知能対人工知能の対戦を見ることができる．

対局画面に移動すると図 19 のような画面となる．画面左上に表示されている「○」で現在の手番が先手か後手かを表しており、青丸だと先手番、赤丸であれば後手番である．また、画面右上にある「menu」ボタンをクリックすることでタイトル画面に戻ることができる．

図 19: どうぶつしょうぎアプリ (対局画面)

自分の手番のときに自分の駒をクリックすると図のように選択した駒が拡大し、その駒が移動できるマスが表示される．移動できるマスの中からいずれかをクリックするとその位置に駒を移動させ、手番が交代となる．また、一回目のクリックの際に選択した駒とは別の駒を動かしたくなった場合は移動できるマス以外の部分をクリックすると、現在選択している駒の選択が解除されるので、再度移動させたい駒を選択することが出来る．

この操作を先手、後手、先手……と交互に繰り返し、「先手勝ち」「後手勝ち」「引き分け」のいずれかになった時には図 20、図 21、図 22 の画面に遷移し、対局が終了する．

図 20: どうぶつしょうぎアプリ (先手勝ち)

図 21: どうぶつしょうぎアプリ (後手勝ち)

図 22: どうぶつしょうぎアプリ (引き分け)

8　まとめ

本研究では、「強い人工知能の作成」「盤面の評価方法の指針を示す」の２つを目的として研究した.

盤面の評価方法として、各駒にそれぞれ得点をつける「駒得」、次に打てる手の数を評価値とする「利き」、王手されているか、王手しているかを基準とする「王手・王手逃れ」、次の手でトライできるか、トライされるかを評価する「トライ・トライ逃れ」の３つを盤面の評価方法として、使用した.

また、遺伝的アルゴリズムを適用し、自己対戦によって学習させていくことで人工知能が進化し、学習前の人工知能と対戦した結果、学習後の人工知能の方が勝率が上がったことから、遺伝的アルゴリズムによる学習は効果があることが分かり、学習前に比べて強い人工知能を作成することができた.

また、作成した人工知能を搭載したスマートフォンアプリケーションを作成した.

今後の課題として本研究で作成したものよりもより強い人工知能を作るためには、盤面を評価する要素を増やすこと. また、指し手を選択するアルゴリズムであるミニマックス法でゲーム木の探索をする際に、木の葉の部分の局面を評価していくだけでなく、葉に至るまでの局面も評価することでより良い手を指すことができるのではないかと考える.

参考文献

[1] 小学館国語辞典編集部編著,「大辞泉（第 2 版）」, 松村明監修, 小学館,2012 年

[2] (社) 人工知能学会「What's 人工知能」,http://www．人工知能-gakk 人工知能．or．jp/whats 人工知能/

[3] David M． Bourg・Glenn Seemann,「ゲーム開発者のための人工知能入門」, 株式会社クイープ翻訳, オライリージャパン,2005 年

[4] 瀧澤武信・他,「人間に勝つコンピュータ将棋の作り方―あから 2010 を生み出したアイデアの工夫と軌跡」, コンピュータ将棋協会監修, 技術評論社,2012 年

[5] 小谷善行,「コンピュータ将棋の頭脳―人間に追いつく日はいつ？」, サイエンス社,2007 年

[6] 田中哲朗,「　　「どうぶつしょうぎ」の完全解析」,2009 年,http://media.itc.u-tokyo.ac.jp/ktanaka/dobutsushogi/animal-private.pdf,2013 年 12 月 15 日アクセス

[7] 「Cocos2d-x」,http://www.cocos2d-x.org/,2013 年 12 月 15 日アクセス