Łukasz Kowalski
IZ, Inf, IO, 3 rok
Nr indeksu 96958

Data seminarium: 30.05.2001

ALGORYTMY EWOLUCYJNE W GRACH LOGICZNYCH

(część seminarium ‘Algorytmy ewolucyjne w wybranych dziedzinach’)

Wrocław 2001

Spis treści:

1. Wstęp

Gry logiczne są jednym z najstarszych i najbardziej szczegółowo analizowanych obszarów zastosowań sztucznej inteligencji. Zasady gier są precyzyjnie określone, a rezultat rozgrywki jest łatwo mierzalny. Z tego powodu są chętnie wykorzystywane w pracach naukowych i już dawno udowodniły, że są ważną dziedziną w badaniach nad sztuczną inteligencją.

1.1. Podstawowe pojęcia teorii gier

Gra - rozgrywka prowadzona przez gracza/graczy zgodnie z ustalonymi zasadami, w której należy osiągnąć ściśle określony cel.

Strategia gry - kompletny zbiór zasad, które determinują posunięcie gracza, wybierane dla sytuacji powstających podczas gry.

Drzewo gry - w każdym kroku rozgrywki gracze wykonują wybrane przez nich posunięcia spośród zbioru możliwych ruchów. Grę można postrzegać jako poszukiwanie takich ruchów gracza, które zapewniają mu zwycięstwo. Taki proces poszukiwania można opisać za pomocą drzewa, którego korzeń jest stanem początkowym gry, a kolejne węzły są stanami, jakie będą aktualne po wykonaniu każdego ruchu spośród możliwych na danym etapie gry. Nawet dla prostych gier drzewa są bardzo skomplikowane, dlatego stosuje się różne metody do oceny aktualnego stanu gry i wyboru kolejnych posunięć, bez konieczności budowy całego drzewa. Jakość poszczególnych węzłów czyli stanów gry definiuje się poprzez funkcję oceniającą poszczególne węzły.

Gry można podzielić ze względu na wiele aspektów:

1) liczba graczy:
   - gry bezosobowe (zero-player game), np. Życie (Conway's live)
   - gry jednoosobowe (one-player game), np. puzzle, pasjans
   - gry dwuosobowe (two-player game), np. szachy, warcaby, Otello, Go
   - gry wieloosobowe (multi player), np. Poker, Brydż
2) wygrania i przegrana:
   - wygrana jednego gracza jest przegraną drugiego (zero sum games), np. szachy, warcaby, Otello, Go
   - nie "zero - jedynkowe" (non zero sum), np. dylemat więźnia
3) współpraca graczy:
   - kooperacyjne (cooperative) – gracze współpracują ze sobą
   - nie kooperacyjne (non cooperative) – gracze nie współpracują ze sobą
4) ze względu na występujący w nich element losowości:
   - całkowicie losowe, np. ruletka
   - częściowo losowe, np. brydż i inne gry karciane
   - całkowicie deterministyczne, np. warcaby, szachy, Otello, Go

1.2. Zastosowania algorytmów ewolucyjnych w grach

Można wyróżnić dwa główne rodzaje zastosowań algorytmów ewolucyjnych (EA) w grach:

1) zastosowanie EA jako wbudowanego modułu do programu grającego, który na każdym etapie gry odpowiada za podjęcie optymalnej decyzji przez program grający. To zastosowanie będzie pokazane na przykładzie systemu Genetic Mastermind,

2) jako narzędzie wspomagające we wcześniejszym szukaniu optymalnej strategii gry. Program grający w trakcie gry nie wykorzystuje algorytmu ewolucyjnego, lecz ma zaimplementowaną znalezioną przez niego strategię (znany przykład dla gry Dylemat więźnia: Goldberg [1], Michalewicz [2]). Podczas szukania strategii osobniki-strategie z populacji mogą prowadzić rozgrywki między sobą lub można wykorzystać "zewnętrznych trenerów", którymi może być dobrany odpowiednio zestaw programów grających lub/i ludzie. Inne zastosowanie to znalezienie przez EA odpowiedniej funkcji oceniającej stan gry w celu wybrania najkorzystniejszego ruchu. W dokumencie omówione będzie zastosowanie algorytmu genetycznego do konstrukcji funkcji oceniającej w grze Othello.

2. Przegląd zastosowań algorytmów ewolucyjnych w grach logicznych

2.1. Genetic Mastermind

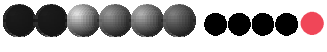
2.1.1. Zasady gry Mastermind

Mastermind jest grą logiczną dla dwóch graczy. Jeden z nich (*codemaker*) ustawia sekretną kombinację (ważna jest kolejność tych elementów w ciągu) pewnych elementów np. kolorów lub liczb. Drugi z graczy (*codebreaker*) ma za zadanie odgadnąć tą kombinację w jak najmniejszej liczbie prób. Odgadujący swoje próby pokazuje pierwszemu graczowi, który w pewien sposób ocenia je podając ile elementów jest na takich samych miejscach jak w ciągu do odgadnięcia, a ile elementów jest w ciągu ale nie na swoich miejscach.

Problem odgadnięcia ciągu kolorów w grze Mastermind możemy sformułować także jako poszukiwanie kodu przy pomocy podpowiedzi otrzymywanych z ‘czarnej skrzynki’. Nieznany kod oraz próby jego odgadnięcia są tworzone z alfabetu o liczności N a każda próba ma długość L. W takim przypadku rozmiar przestrzeni rozwiązań to N^L. Sprzedawana gra ma dwa warianty: ‘Klasyczna’: z N=6 i L=4 oraz ‘Super Mastermind’ z N=8 i L=5. W tych przypadkach rozmiary przestrzeni wyników to 1296 oraz 32768. Każda ‘podpowiedź’ od gracza ogranicza przestrzeń rozwiązań, wykluczając pewne kombinacje z tej przestrzeni. Jednak ponieważ przeszukiwana przestrzeń jest wielowymiarowa, nie jest prosto wyznaczyć podprzestrzeń która została wykluczona poprzez uzyskaną podpowiedź.

Przykładowy ciąg do odgadnięcia:

Próba odgadnięcia i jej ocena:



Oznaczenia:

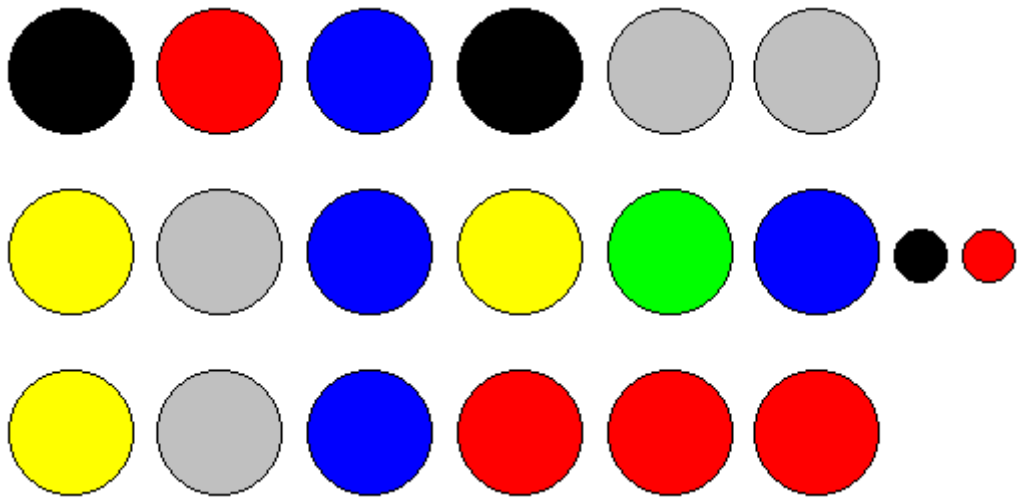
kółko czarne: odgadnięty kolor i pozycja;
kółko czerwone: odgadnięty tylko kolor, pozycja niewłaściwa

2.1.2. Przykład zastosowania algorytmu ewolucyjnego

W Internecie na stronie http://geneura.ugr.es/~jmerelo/GenMM/GenMM.shtml można znaleźć jedna z prób zastosowania algorytmu ewolucyjnego w grze Mastermind oraz zagrać z komputerem w tą grę układając dla niego do zgadnięcia ciąg do 8 cyfr z puli liczb 1-9 które są później przedstawione jako kolory. Osobnik populacji to ciąg kolorów-cyfr, który jest próbą rozwiązania łamigłówki.

2.1.2.1. Ocena osobników

Na początek rozważmy problem oceny osobników w populacji. Z oczywistych względów nie możemy dawać każdego osobnika populacji do oceny przez gracza. W tym przypadku musimy sobie poradzić zupełnie inaczej. Zademonstruję to na przykładzie (rysunek poniżej). Górny rząd to sekretna kombinacja do odgadnięcia, środkowy rząd to jedna z kombinacji która została już przedstawiona do oceny gracza (który ułożył kombinację do odgadnięcia) wraz z odpowiedzią. Najniższy rząd to jedna z kombinacji (osobnik z pokolenia) która musi zostać oceniona. W zagranej i ocenionej kombinacji (rząd drugi) tylko jedna pozycja się zgadza i jeden kolor z ukrytą kombinacją, natomiast kombinacja do oceny zgadza się z zagraną kombinacją tylko w trzech miejscach. Jednakże tylko dwa kolory wystarczy zmienić aby kombinacja trzecia uzyskała taką samą ocenę w stosunku do drugiej jak druga w stosunku do pierwszej. W naszym przypadku wystarczyłoby zmienić trzeci rząd na następujący: kolor czerwony, żółty, niebieski i 3 razy czerwony – zmieniliśmy tylko dwa kolory. W związku z powyższym dwa jest tutaj odległością trzeciej kombinacji do drugiej i osobnik uzyskuje ocenę –2.



Każdy osobnik populacji można więc ocenić względem każdej przedstawionej już (zagranej) kombinacji. W ten sposób odpowiemy sobie na pytanie z iloma odpowiedziami drugiego gracza (trafieniami pozycji i koloru lub tylko koloru) zgadza się osobnik lub jak ‘daleko’ jest do ich spełnienia. Z powyższego wynika, że osobnik-kombinacja która spełnia po kolei wszystkie odpowiedzi uzyskane do tej pory uzyska ocenę maksymalna równą 0.

2.1.2.2. Szczegóły zastosowanego algorytmu genetycznego

Algorytm genetyczny w analizowanym systemie stara się minimalizować odległość (distance) do optymalnej kombinacji to jest takiej która spełnia wszystkie odpowiedzi. Zastosowano bezpośrednie przedstawienie osobnika jako ciąg liczb odpowiadających konkretnym kolorom. Każda kombinacja to ciąg poprawnych kolorów o tej samej długości i wszystkie operatory genetyczne zawsze wygenerują poprawne ciągi.

Zostały zaimplementowane trzy następujące operatory:

Krzyżowanie – tradycyjne krzyżowanie jednopunktowe dwóch osobników, prawdopodobieństwo operatora: 40 %

Mutacja – zastępuje jeden z kolorów następnym odpowiadającym liczbie większej o 1 (lub kolorem odpowiadającym liczbie 1 po przekroczeniu maksymalnej ilości kolorów), prawdopodobieństwo operatora: 40 %

Transpozycja – wprowadza permutacje czyli przemieszczenie kolorów, prawdopodobieństwo operatora: 20 %

Zamiana – operator stosowany opcjonalnie, zamienia kolor wartością losową, nie używany ze względu na to iż operator mutacji działa bardziej systematycznie w odróżnieniu od losowości tego operatora.

Metoda selekcji
W każdym pokoleniu 50% najgorszych osobników jest eliminowanych i są one zastępowane przez potomstwo pozostałej części.

Ocena osobników
Sposób oceny przedstawiony wcześniej został użyty jako funkcja fitnes.

Rozmiar populacji
Standardowy rozmiar populacji to 400 osobników, widać tu wyraźny postęp w stosunku do poprzednich eksperymentów tych samych autorów gdzie wykorzystywano nawet 10000 osobników. Zmniejszenie liczy osobników było możliwe dzięki powyższej funkcji oceny, wcześniej w tym projekcie była używana inna funkcja.
Jeżeli do 15 pokoleń nie zostanie odnalezione rozwiązanie cała populacja jest unicestwiana i losowana jest nowa. Jednak takie przypadki zdarzają się bardzo rzadko. Głównym problemem tego algorytmu jest utrzymanie różnorodności populacji. Może być to rozwiązane przez zwiększenie współczynnika mutacji lecz doprowadziłoby to do losowego przeszukiwania przestrzeni rozwiązań czego autorzy próbowali uniknąć.

Pierwsza kombinacja będąca próbą zgadnięcia kombinacji
Dwie możliwości:
- wszystkie możliwe kolory po kolei od pierwszego, przykład: 1234 (odpowiedź co najmniej powie nam liczbę kolorów użytych w kombinacji do odgadnięcia),
- kolory po kolei podobnie jak powyżej ale korzystamy z połowy wszystkich dostępnych kolorów, przykład: 1122 (zaproponowane przez Knutha, analiza trudniejsza do przeprowadzenia i bardziej teoretyczna).

2.1.2.3. Wyniki

Autorzy porównali wyniki swojego systemu z systemami innych autorów i pracami teoretycznymi.

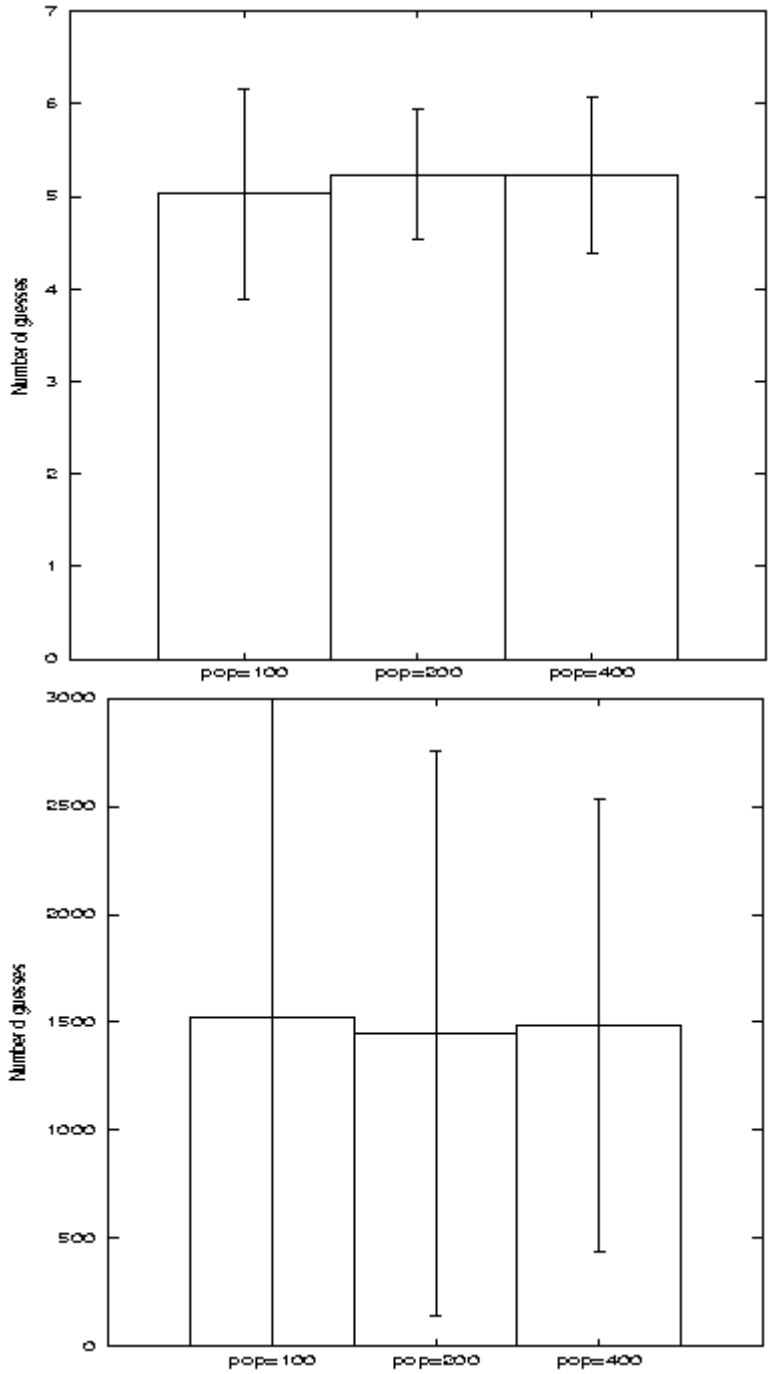
Wyniki dla 6 kolorów i 4 pozycji. Najlepsze rezultaty Genetic Mastermind są bliskie najlepszym znalezionym do tej pory rozwiązaniom lecz średnia GM jest mniejsza od rozwiązań otrzymanych przez Koyama'e i Knuth'a. Algorytm uzyskał lepsze rezultaty przy użyciu pierwszej kombinacji typu 1122 niż 1234. GM został uruchomiony 1000 razy z populacją równą 100 osobników.

Autor	Średnia liczba prób	Średnia liczba ocenionych przez AG kombinacji
Knuth	4.478	-
Koyama	4.340	-
Bestavros	3.8 ± 0.5	-
Rosu	4.66	-
Genetic Mastermind (1234)	4.312 ± 0.989	320 ± 268
Genetic Mastermind (1122)	4.132 ± 1.00	279 ± 188

Wyniki dla trudniejszego zadania: 8 kolorów i 5 pozycji. Wyniki GM są porównywalne z najlepszymi znalezionym przez Rosu. W porównaniu z pracą Bento algorytm znajdował rozwiązanie w mniejszej liczbie kroków. GM został uruchomiony 500 razy z populacją równą 400 osobników.

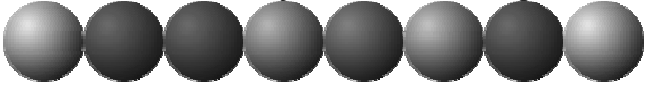
Autor	Średnia liczba prób	Średnia liczba ocenionych przez AG kombinacji
Rosu	5.88	-
Bento	6.866	1029.9
Genetic Mastermind	5.904 ± 0.975	2171 ± 1268

Pierwszy z wykresów pokazuje średnią liczbę prób przy różnej liczności populacji, drugi z wykresów przedstawia średnią liczbę wyewoluowanych kombinacji przy różnej liczności populacji. Wydaje się, że średnie te mało zależą od liczności populacji, jednak warto zauważyć, że dla większych populacji zmniejsza się odchylenie standardowe. Dlatego też większe populacje są preferowane. Wyniki dla 6 kolorów i 4 pozycji.

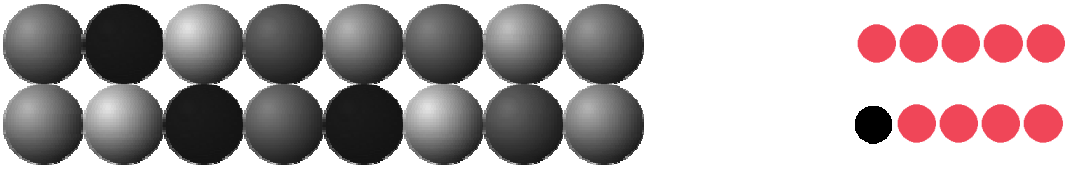


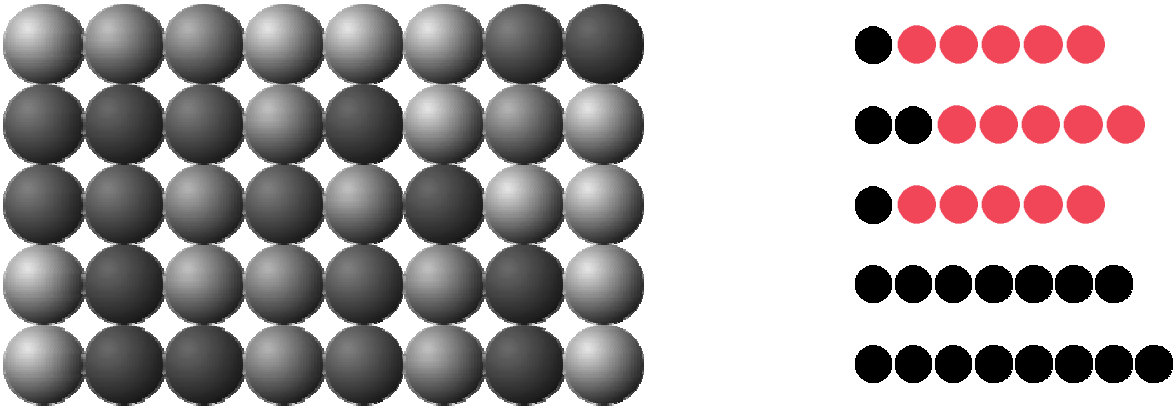
2.1.2.4. Przykładowa gra analizowanego systemu

Zadałem kombinację 34456743 co odpowiada kolorom:



Gra systemu:





Zostało ocenionych 3600 osobników (kombinacji) z 5.7648e+06 wszystkich możliwych.

2.2. Othello

2.2.1. Zasady gry

Otello (Othello) to bardzo starą gra wywodząca się z Japonii zwaną także Reversi. Dwóch graczy toczy grę na planszy 8x8 za pomocą zbioru dwukolorowych pionów (na przeciwnych stronach pionów są różne kolory przypisane konkretnym graczom). Celem gry jest zdobycie na planszy jak największej liczby pionów swego koloru. Gra rozpoczyna się zawsze od pozycji początkowej przedstawionej na Rysunku 1.

	a	b	c	d	e	f	g	h
1	*D*	*C*	*A*	*B*	*B*	*A*	*C*	*D*
2	*C*	*X*					*X*	*C*
3	*A*							*A*
4	*B*			○	●			*B*
5	*B*			●	○			*B*
6	*A*							*A*
7	*C*	*X*					*X*	*C*
8	*D*	*C*	*A*	*B*	*B*	*A*	*C*	*D*

Rysunek 1. Plansza do gry w Othello: początkowa sytuacja i charakterystyczne pola

Grę rozpoczynają piony czarne i kolejne ruchu wykonywane są przez graczy na przemian. Ruch polega na ustawieniu na wolnej pozycji na planszy pionka swoim kolorem do góry w ten sposób aby otoczyć z dwu stron swoimi pionami co najmniej jeden z pionków przeciwnika (lecz musi to być ciągła sekwencja pionów przeciwnika). Otaczać piony można w pionie, poziomie i na ukos. W ten sposób piony przeciwnika znajdujące się pomiędzy

wcześniej postawionym pionem gracza, a pionkiem obecnie postawionym zmieniają kolor na kolor gracza. Gra kończy się w momencie zapełnienia pionami całej planszy lub wtedy gdy jeden z graczy nie będzie już mógł wykonać legalnego ruchu, to znaczy takiego w wyniku którego przejmie choć jeden pion przeciwnika. Wygrywa gracz, który w momencie ukończenia rozgrywki dysponuję na planszy większą liczbą pionów swojego koloru. Dzięki tym zasadom gra jest bardzo dynamiczna i każdy ruch może diametralnie zmienić sytuacje na planszy. Duże znaczenie mają punkty znajdujące się na krawędziach planszy, np. punkty narożne (D na rysunku 1) są nie do przejęcia przez przeciwnika z tego prostego powodu, że nie da się ich otoczyć.
Analizując pobieżnie zasady gry można wymyślić trzy podstawowe strategie:

1. Strategia zdobywania za wszelką cenę maksymalnej liczby swoich pionów na planszy- prowadzi najczęściej do blokady ruchów gracza,
2. Strategia ważonych pól – uwzględnia tzw. stabilność pola (związana z jego położeniem na planszy – pola A, B, C, D) oraz negatywny lub neutralny wpływ zajętości pól na rozgrywkę (np. pola X ułatwiają przeciwnikowi postawienie swego pionka na polach D),
3. Strategia minimalnej liczby pionów – dąży się do minimalizacji ilości własnych pionków na planszy po to aby zwiększyć możliwości własnego ruchu – jednak strategia ta ślepo stosowana prowadzi do przegranej.

2.2.2. Przykład zastosowania algorytmu genetycznego

Jako przykład zastosowania algorytmu genetycznego do utworzenia funkcji oceniającej przedstawię pracę: Kwaśnicka H., Dąbrowski T., Piasecki M.: 'Zastosowanie algorytmu genetycznego do konstrukcji funkcji oceniającej w grze OTHELLO'. Autorzy wyróżnili osiem najważniejszych cech odpowiedzialnych za ocenę konkretnej sytuacji na planszy:
C_1 – liczba pionków obu graczy,
C_2 – liczba legalnych posunięć dla obu graczy,
C_3 – liczba pionków przeciwnika przyległych do pionków gracza,
C_4 – liczba pionków każdego z graczy przyległych do pola pustego (potencjalna ruchliwość),
C_5 – liczba pustych pól przyległych do pionków każdego z graczy,
C_6 – suma liczby pustych pól przyległych do pionków każdego z graczy (te same pola są liczone wielokrotnie dla poszczególnych pionków),
C_7 – liczba zajętych rogów przez pionki gracza,
C_8 – stabilność brzegowa (miara liczbowa wyliczana na podstawie pewnych algorytmów).

Cechy C_1-C_7 dają po dwie wartości – po jednej dla gracza. Pojedyncza wartość wynikowa jest tworzona w nieliniowy sposób według wzoru:

$$C_i = \frac{100 * (p - q)}{p + q + 2},$$

gdzie p jest wartością dla jednego gracza, a q dla drugiego.

Ostatecznie możemy zapisać poszukiwaną funkcję oceniającą E w postaci:

$$E = \sum_{i=1}^{8} w_i * C_i,$$

gdzie: C_i – miary liczbowe dla cech zidentyfikowanych jako istotne dla rozgrywki,
w_i – wagi określające udział każdej z cech w wynikowej wartości funkcji oceniającej.

Przy takich założeniach problem znalezienia jak najlepszej funkcji oceniającej sprowadza się do wyznaczenia wektora wag w_i, który można bardzo prosto przekształcić w chromosom:

- każda waga została zakodowana w kodzie binarnym na 8 bitach, przyjmując wartości z przedziału [-127, 128],
- chromosom powstał jako złączenie binarnych ciągów reprezentujących wszystkie cechy,
- porządek wag w chromosomie jest zgodny z wprowadzoną wcześniej numeracją cech.

2.2.2.1. Szczegóły zastosowanego algorytmu genetycznego

W zaimplementowanym systemie funkcja przystosowania jest wyznaczana w oparciu o wyniki osiągane poprzez poszczególne osobniki w trakcie rozgrywek z innymi programami trenującymi. Pojedynczy osobnik z populacji wykorzystuje funkcję oceniającą zdefiniowaną przez swój genotyp. Należy zauważyć, że wybranym podejściu jakość rezultatów zależy od liczby, różnorodności i jakości programów trenujących, ponieważ mniejsza liczba programów trenujących prowadzi do uczenia się przez osobniki tylko ich słabych stron. Na wartość funkcji przystosowania składają się dwa elementy:

- rezultat gry: zwycięstwo, porażka lub remis – wyrażony poprzez premię punktową w przypadku zwycięstwa lub remisu,
- różnica w końcowej liczbie pionków na planszy.

Okazało się, że konieczne jest ustalenie wysokiej premii za wygraną w stosunku do maksymalnej możliwej różnicy pionków, aby osobniki wygrywające więcej partii były dużo lepiej oceniane niż te które wygrywają rzadziej ale z większą przewagą swych pionów na planszy.

Rozgrywka i metoda selekcji

Każdy osobnik z populacji algorytmu genetycznego rozgrywał po dwie gry z każdym z nauczycieli, zaczynał raz jako pierwszy i raz jako drugi. Następnie po obliczeniu funkcji przystosowania, osobniki do reprodukcji były wybierane zgonie z metodą *wyboru wg reszt bez powtórzeń*. Klasyczna metoda ruletki została odrzucona jako dająca zbyt dużą wariancję.

Operatory genetyczne

Zastosowano dwa operatory genetyczne:

Krzyżowanie – prawdopodobieństwo operatora to 0,6,

Mutacja – prawdopodobieństwo mutacji miało różną wartość dla różnych eksperymentów.

Ze względu na stosunkowo małą wielkość populacji (z powodu dużej złożoności obliczeniowej) zastosowano mechanizm regulacji liczby kopii poprzez użycie *skalowania liniowego*:

$$f' = a * f + b,$$

gdzie f – przystosowanie pierwotne, a f ’ – przystosowanie po przeskalowaniu.

Współczynniki a i b zostały dobrane w taki sposób aby średnie przystosowanie nie uległo zmianie w wyniku skalowania oraz średnia liczba potomków o maksymalnym przystosowaniu była wielokrotnością średniego przystosowania $f'_{maks} = C_{zw} * f_{śr}$, gdzie wartość C_{zw} została ustalona na 2.

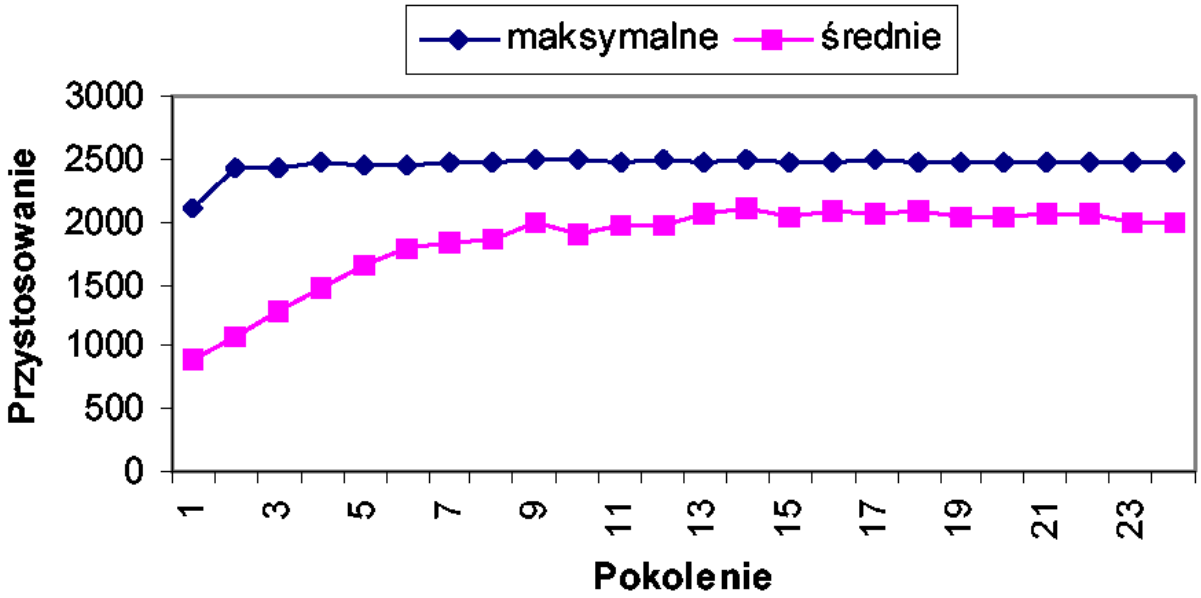
Warto też wspomnieć, że wartości czynników składowych C_i funkcji oceniającej są bardzo zróżnicowane (wahają się od (-67, 67) dla C_7 do [-3440, 3440] dla C_8) i aby zapewnić równomierny wpływ poszczególnych wag w_i na funkcję oceniającą poddano je normalizacji przed wyznaczeniem wartości funkcji oceniającej. Dodatkowo w końcowej fazie gry, w momencie osiągnięcia liścia drzewa gry jedyną miarą staje się różnica w ilości pionków [-64, 64] relatywnie mała w stosunku do rozpiętości przedziału wartości funkcji oceniającej. Dlatego też niezbędne stało się zastosowanie odpowiedniego rzutowania.

2.2.2.2. Eksperymenty

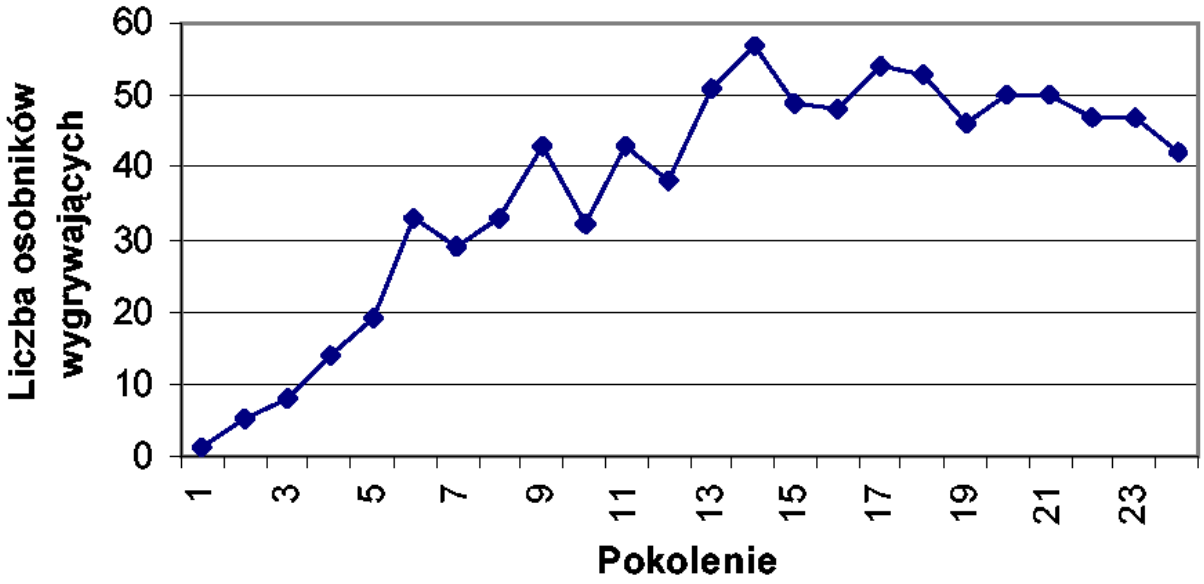
Autorzy przeprowadzili 8 eksperymentów z czego dwa ostatnie dotyczyły uczenia na podstawie rywalizacji wewnątrz populacji. Opiszę najciekawsze fragmenty eksperymentów.

- **Eksperyment drugi**
  Liczba osobników = 100
  Liczba pokoleń = 23
  Liczba nauczycieli = 2
  Prawdopodobieństwo mutacji = 0,01
  Maksymalna możliwa ocena = 2512

  Zaobserwowano silne zróżnicowanie pomiędzy najlepszymi osobnikami: wśród zwycięskich osobników trudno było wyznaczyć jakieś typy wag. Prawdopodobnie, ze względu na niewielką liczbę nauczycieli i ich niską jakość, poszczególne osobniki nauczyły się wykorzystywać słabe strony nauczycieli. Zaobserwowane wahania w liczbie osobników wygrywających ze wszystkimi nauczycielami były spowodowane przyjętym zbyt dużym prawdopodobieństwem mutacji.



Wykres funkcji przystosowania dla eksperymentu 2



Liczba osobników, które wygrały wszystkie rozgrywki z nauczycielami (eksperyment 2)

- **Eksperyment trzeci**
  Populacja osobników jak w eksperymencie drugim, ale liczba nauczycieli została zwiększona do czterech. Rezultaty podobne jak w poprzednim eksperymencie.

- **Eksperyment czwarty**
  Zastosowano inną metodę tworzenia nowego pokolenia: *metodę ścisku De Jonga* (Goldberg [1]). Procent populacji podlegający reprodukcji został ustalony na 0,2, natomiast współczynnikowi ścisku nadano wartość 3. W rezultacie otrzymano szybki wzrost liczby osobników wygrywających wszystkie partie (10 osobników w 2 pokoleniu). Dalszy wzrost tego wskaźnika był nieregularny i wolny, ale znacznie stabilniejszy w późniejszych pokoleniach (nie wykazujący takich wahnięć jak w eksperymencie 2 i 3). Wartość maksymalna funkcji przystosowania osiągnęła swoje maksimum w 11 pokoleniu, a następnie utrzymywała się prawie na stałym poziomie (wartość średnia wykazywała stałą tendencję wzrostową).

- **Eksperyment piąty**
  W eksperymencie piątym radykalnie zmniejszono prawdopodobieństwo mutacji do 0,0001 (oraz rozmiar populacji na 30), co doprowadziło do szybkiej zbieżności oraz upodobnienia się osobników zwycięskich. Po 36 pokoleniach populacja zawierała prawie identyczne wektory wag. Mimo to najlepszy osobnik z 30 pokolenia wygrywając ze wszystkimi nauczycielami stracił jedynie średnio po 10,5 pionka w grze.

- **Eksperyment szósty**
  W tym eksperymencie w stosunku do poprzedniego zwiększono prawdopodobieństwo mutacji do 0,001 co doprowadziło znowu do większego zróżnicowania zwycięskich osobników.

- **Eksperyment siódmy i ósmy**
  W dwóch ostatnich eksperymentach spróbowano zrealizować schemat uczenia bez nauczyciela: osobniki rywalizowały wewnątrz populacji – każdy z każdym po dwie gry. Oto parametry symulacji:
  Liczba osobników = 30
  Liczba pokoleń = 11
  Prawdopodobieństwo mutacji = 0,0033
  Maksymalna możliwa ocena = 11848224

  Niestety w obu eksperymentach nie zaobserwowano postępu ani w wartości średniej, ani w maksymalnej. Przyczyn może być wiele, np. zbyt mała liczba pokoleń (ze względu na długi czas obliczeń) lub nieprzydatność stosowanej funkcji przystosowania do porównywania wyników osiąganych w kolejnych pokoleniach w zmieniającym się środowisku. Z całą pewnością trudniej jest znaleźć strategię wygrywającą z dwudziestoma dziewięcioma innymi, na dodatek zmieniającymi się w czasie strategiami niż ze stałymi czterema zewnętrznymi trenerami.

2.2.3. Inne projekty z grą Othello

2.2.3.1. Genetic Othello – Itamar Faybish: *Applying the genetic algorithm to the game of Othello*

Praca częściowo podobna do wcześniej omówionej w punkcie 2.2.2. Autor wyróżnił 7 parametrów dla wszystkich osobników:

1. dla każdego pustego pola tablicy, zobacz czy jest tam przynajmniej jeden pionek w danym kolorze, który jest w bezpośrednim sąsiedztwie (jeden kwadrat dalej). Jeżeli tak to dodaj jeden do zmiennej odniesionej do znalezionego koloru,
2. dla każdego pola danego koloru w centralnej tablicy 7x7, oblicz liczbę pustych pól otaczających je,
3. parametr liczony jak powyżej, ale dla tablicy 8x8,
4. dla każdego pola gdzie znajdują się pionki danego gracza zobacz czy jest tam przynajmniej jedno puste pole, wokół niego. Jeśli tak wtedy dodaj jeden do zmiennej,
5. dla każdego pola w danym kolorze dodaj liczbę pól w tych samych kolorach, które otaczają go,
6. dla każdego pola w danym kolorze zobacz czy jest tam przynajmniej jeden pionek w tym samym kolorze w bezpośrednim sąsiedztwie. Jeżeli jest dodaj jeden do wartości dla odpowiedniego koloru,
7. parametr nadający specjalne znaczenie pustym polom, które są całkowicie otoczone przez piony w różnym kolorze i gdzie tylko jeden gracz może grać w tym polu. Te regiony mają ważne znaczenie w Othello. Głównym powodem jest to, że ostatni gracz, który kładzie pionek na tablicy ma dużą przewagę. Jeśli komputer ma jeden lub wiele takich pól, to wzrasta jego szansa grania ostatniego ruchu.

2.2.3.2. Discovering Complex Othello Strategies Through Evolutionary Neural Networks

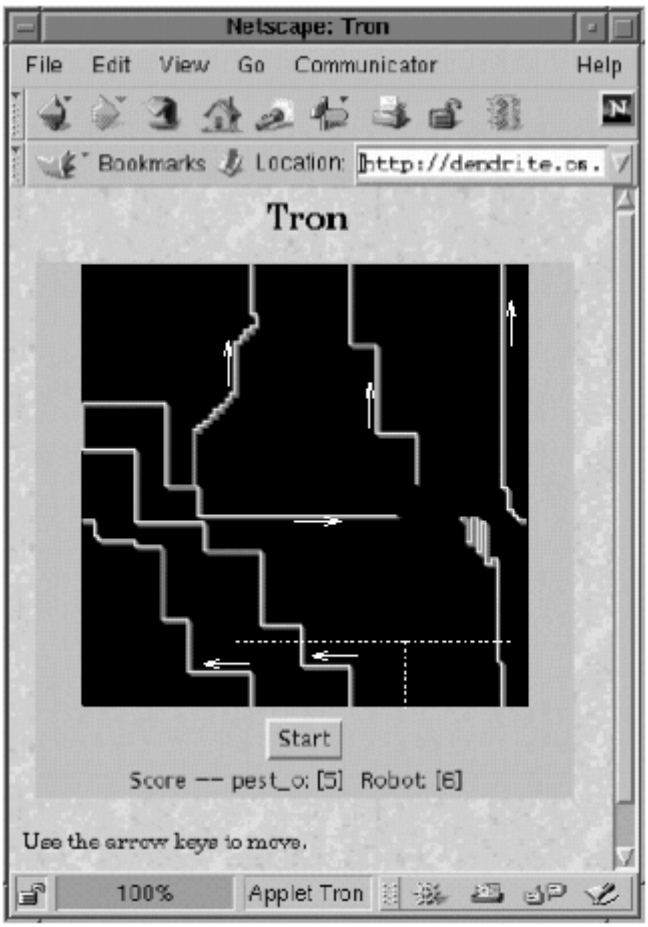
Praca w której autorzy wykorzystali połączenie algorytmów genetycznych z sieciami neuronowymi. W projekcie ewoluowano sieć neuronową w celu odnalezienia jak najlepszych strategii gry w Othello.

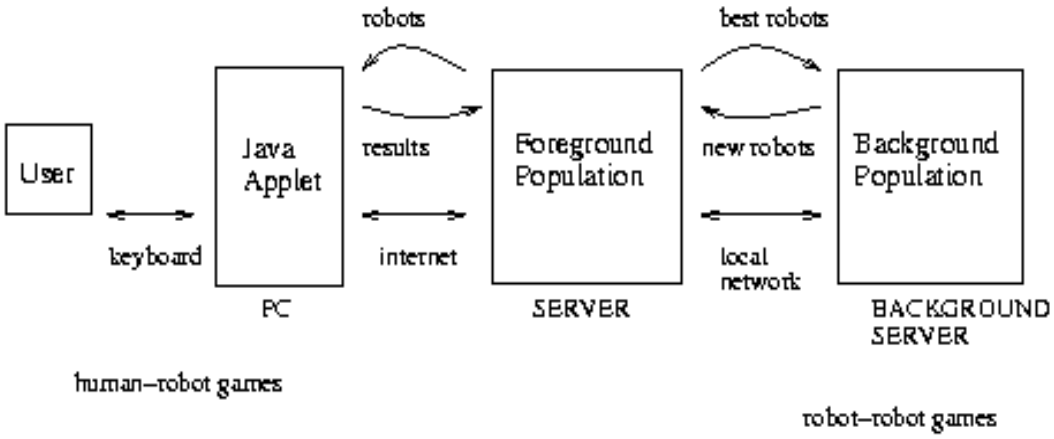
2.3. Algorytmy ewolucyjne w innych grach

- **Tron** – projekt ten został omówiony dokładnie przez poprzednią grupę seminaryjną jako przykład zastosowania koewolucji.

Bardzo ciekawy projekt łączący w sobie elementy koewolucji oraz współzawodniczenie ludzi z wyewoluowanymi agentami. Gra Tron nawiązuje do filmu pod tym samym tytułem. Na planszy dwóch graczy porusza się zostawiając za sobą ślad który staje się nieprzekraczalną barierą (ścianą). Celem gry jest zmuszenie przeciwnika do rozbicia się na ścianę.

Użytkownicy poprzez stronę http://www.demo.cs.brandeis.edu/tron/data/tron_login.cgi logują się do systemu Tron gdzie mogą toczyć gry z wyewoluowanymi sztucznymi graczami – agentami. Niejako w tle całego systemu toczy się walka między samymi wyewoluowanymi graczami.





- **Go**
  Gra troszkę podobna zasadami do Othello pochodząca z Chin: dwóch graczy, plansza 19 na 19 pól. Znalezione prace zajmujące się tą grą wykorzystywały najczęściej ewolucję sieci neuronowej poprzez algorytmy ewolucyjne.

- **Corewars** czyli **Wojny Rdzeniowe**
  Rywalizacja prostych programów komputerowych (podobnych do wirusów) w sztucznym środowisku pewnego komputera.

- **Pragnienie**
  W pracy dyplomowej Radosława Kozieji „Zastosowanie algorytmów genetycznych w modelowaniu ekonomicznym" podjęto próbę wykonania prostej gry menedżerskiej o nazwie PRAGNIENIE, w której konkurują między sobą producenci napojów. Jednym z uczestników gry mógłby być komputer, grałby on stosując strategię znalezioną przez algorytm genetyczny.

- **Szachy**

- **Senet**
  Senet to starożytna gra egipska. Zastosowano programowanie genetyczne do ewoluowania funkcji oceniających planszę.

3. Materiały źródłowe

[1] David E. Goldberg: G*enetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, Inc., 1989; wydane po polsku: *Algorytmy genetyczne i ich zastosowania*, WNT, Warszawa 1995.

[2] Zbigniew Michalewicz: *Algorytmy genetyczne + struktury danych = programy ewolucyjne*, WNT, Warszawa 1999.

[3] Halina Kwaśnicka, Tomasz Dąbrowski, Maciej Piasecki: *Zastosowanie algorytmu genetycznego do konstrukcji funkcji oceniającej w grze OTHELLO*, III Krajowa Konferencja Algorytmy Ewolucyjne i Optymalizacja Globalna KEAiOG'99, 25-28 maja 1999, Potok Złoty. Materiał dostępny w Internecie: http://www.ci.pwr.wroc.pl/~kwasnick/othello.zip.

[4] Zbigniew Kowalczyk, Ireneusz Matysiak, Paweł Myszkowski, Tomasz Skawiński: *Algorytmy genetyczne w grach*, seminarium z roku 2000.
[5] Radosław Kozieja: *Zastosowanie algorytmów genetycznych w modelowaniu ekonomicznym*, praca magisterska.

Materiały w Internecie:
[6] Genetic Mastermind
http://geneura.ugr.es/~jmerelo/GenMM/GenMM.shtml (gra online)
[7] Genetic Othello
http://www.geocities.com/ifaybish/othello.html
[8] Discovering Complex Othello Strategies Through Evolutionary Neural Networks
http://www.cs.utexas.edu/users/nn/pages/publications/abstracts.html#moriarty.discovering.ps.Z
[9] Strona projektu Tron
http://www.demo.cs.brandeis.edu/tron
http://www.demo.cs.brandeis.edu/tron/data/tron_login.cgi (gra online)
[10] Skarbnica odnośników do projektów związanych ze sztuczną inteligencją w grach
http://satirist.org/learn-game/ (strona główna)
http://satirist.org/learn-game/methods/ga/ (projekty wykorzystujące AG w grach)
http://satirist.org/learn-game/lists/papers.html (dokumenty)
http://satirist.org/learn-game/systems/othello/ (projekty związane z grą Othello)
[11] Dynamical & Evolutionary Machine Organization (DEMO) – szereg ciekawych projektów związanych ze Sztuczną Inteligencją
http://www.demo.cs.brandeis.edu/
http://www.demo.cs.brandeis.edu/pr/learning.html