Data dostarczenia: 14.06.2005
Data prezentacji: 09.06.2005

Algorytmy Genetyczne

Wydział Informatyki i Zarządzania
Politechnika Wrocławska

Zastosowanie algorytmów ewolucyjnych w grach logicznych

Autor dokumentu: *Paweł Stawarz*
Indeks: *125939*
Grupa: *czwartek / 11:15*

Prowadzący: prof. dr hab. Halina Kwaśnicka

1. Gry logiczne

Gry logiczne towarzyszą człowiekowi już od czasów prehistorycznych. Dla wielu z nas stanowią coraz częściej bardzo przyjemną formę rozrywki intelektualnej. Na pewno nieraz zdarzyło Ci się zagrać w jakieś gry planszowe (np. szachy, warcaby) lub gry karciane (brydż, poker). Intuicyjnie wiemy, że w każdej grze dążymy do osiągnięcia określonego celu, postępując zgodnie z jasno sprecyzowanymi zasadami. W trakcie gry każdy układa swój własny plan działania – swoją własną strategię postępowania. Okazuje się, że wszystkie te intuicyjne pojęcia znajdują swe odzwierciedlenie w mocno sformalizowanej teorii gier, która opisuje gry za pomocą aparatu matematycznego.

1.1 Definicje elementarnych pojęć

Korzystając z intuicyjnego podejścia do gier, zrozumienie poniższych definicji nie powinno budzić żadnych wątpliwości:

Gra
Rozgrywka prowadzona przez gracza (lub graczy) zgodnie z ustalonymi zasadami, w której należy osiągnąć ściśle określony cel.

Strategia gry
Kompletny zbiór zasad, które determinują posunięcie gracza, wybierane zależnie od sytuacji powstających podczas gry.

Formalna definicja gry wg von Neumanna i Morgensterna
Gra składa się z zestawu reguł określających możliwości wyboru postępowania jednostek znajdujących się w sytuacji określanej mianem konfliktu interesów, w której każda z jednostek stara się maksymalizować swój własny zysk i jednocześnie zminimalizować zysk pozostałych jednostek (graczy). Reguły gry określają też ilość informacji dostępną każdemu z graczy oraz wysokość wygranych i przegranych.

1.2 Podział gier

Gry możemy podzielić ze względu na:

- **Liczbę graczy:**
  - gry bezosobowe – np. gra w życie,
  - gry jednoosobowe – np. puzzle, pasjans, mahjong,
  - gry dwuosobowe – np. szachy, warcaby, go, othello,
  - gry wieloosobowe – np. poker, brydż, scrabble, szachy (dla trzech osób),
- **Wynik gry:**
  - gry o sumie zerowej, gdzie wygrana jednego gracza jest przegraną drugiego gracza,
  - gry o sumie niezerowej – np. dylemat więźnia, o którym będzie jeszcze mowa,
- **Współpracę graczy:**
  - gry kooperacyjne – np. szachy dla trzech osób, brydż,
  - gry niekooperacyjne – np. szachy, warcaby,
- **Losowość gry:**
  - gry całkowicie losowe – np. ruletka, chińczyk,
  - gry częściowo losowe – np. brydż, poker,
  - gry całkowicie deterministyczne – np. szachy, warcaby, go,
- **Ilość informacji dostępnej graczom:**
  - gry o pełnej informacji o stanie gry – np. szachy,
  - gry o częściowej informacji o stanie gry – np. statki.

1.3 Związek ze sztuczną inteligencją

Gry logiczne są nierozerwalnie związane ze sztuczną inteligencją, już od początku jej istnienia. Stanowią one swego rodzaju świetny poligon doświadczalny zarówno dla fanatyków chcących ujarzmić tak złożone gry jak szachy lub go, jak i naukowców szukających nowych idei w sztucznej inteligencji. Dlaczego jest to świetne pole do popisu? Otóż, dlatego, że gry logiczne tworzą swój jasno sprecyzowany świat rządzący się własnymi regułami. Wypracowane techniki w wyniku badań zjawisk ewolucyjnych w grach mogą być następnie przenoszone w dużo poważniejsze dziedziny życia, bliższe naszemu fizycznemu światu rzeczywistemu, w których często nie da się przewidzieć wszystkich czynników zewnętrznych wpływających na badany obiekt.

Gry logiczne utożsamiane są też bardzo często (a na pewno były) z umiejętnością myślenia. Zatem stworzenie programu komputerowego zdolnego do wygrywania w danej grze z człowiekiem wiążę się z odważnym stwierdzeniem, że maszyna może myśleć. W szczególności szachy, złożona i niemal „nieskończona" gra, której nie jesteśmy w stanie ogarnąć w całości umysłem, a której najszybsza maszyna na świecie nie jest w stanie, w przyswajalnym dla wielu pokoleń czasie, wyznaczyć wszystkich możliwych wariantów, były uważane za wyznacznik zdolności maszyny do myślenia. Już w 1950 r. znany i mający duży wkład w sztuczną inteligencję naukowiec Shannon zaproponował pojedynek szachowy jako odpowiedź na pytanie Turinga: czy maszyna może myśleć? Niemal pół wieku później w 1997r. cała ludzkość dostaje odpowiedź ze strony komputera szachowego DEEP BLUE firmy IBM, który wygrywa mecz przeciwko ówczesnemu mistrzowi świata, najlepszemu szachiście wszechczasów, Garriemu Kasparowowi, wynikiem 3,5:2,5 (był to rewanż za przegraną z poprzedniego roku). Jednak nawet najwięksi entuzjaści sztucznej inteligencji są dalecy od stwierdzenia, że DEEP BLUE rzeczywiście myśli. Komputer ten został specjalnie wyszkolony pod swojego przeciwnika (Kasparowa). Zawierał on obszerną wiedzę książkową na temat gry w szachy jak i obszerny zbiór partii Kasparowa, z których w odróżnieniu od samego Kasparowa mógł korzystać. Sam DEEP BLUE wykazał się na pewno miażdżącą przewagą obliczeniową oraz psychologiczną! Projektanci zapewniali, że DEEP BLUE jest w stanie przeanalizować 200 milionów pozycji na sekundę, gdzie człowiek w tym samym czasie w najlepszym wypadku może przeanalizować około 3 pozycji. Stres wynikający z gry oraz presja milionów a może nawet miliardów ludzi wpatrzonych w historyczne wydarzenie na pewno były obce „inteligentnej" maszynie.

2. Algorytmy genetyczne w grach logicznych

W zastosowaniu algorytmów genetycznych w grach logicznych wyróżnia się dwa główne podejścia:

- algorytm genetyczny szuka optymalnej gry, która następnie jest stosowana przez program grający – sam program nie zawiera algorytmu genetycznego;
- algorytm genetyczny szuka optymalnej decyzji w każdym kroku gry – algorytm wbudowany w program grający jest aktywny przez całą grę.

Oba podejścia zostaną zaprezentowane na przykładzie dwóch gier. Pierwsze podejście zostanie przedstawione w dylemacie więźnia, w którym naukowiec Axelrod za pomocą algorytmu genetycznego odnalazł o wiele lepszą strategię gry od ówcześnie uważanej za najlepszą. Drugie podejście zostanie przedstawione w grze MasterMind powstałej na Uniwersytecie w Grenadzie w Hiszpanii.

3. Dylemat więźnia – AG szuka optymalnej strategii

3.1. Zasady gry

Dylemat więźnia jest klasycznym przykładem problemu konfliktu i kooperacji. Jest to gra dwuosobowa o sumie niezerowej. W najprostszym sformułowaniu każdy gracz może „Współpracować" lub „Zdradzić". Nazwa gry stanowi abstrakcyjny odpowiednik sytuacji dwóch więźniów, którzy mogą współpracować ze sobą lub też nawzajem zdradzić się. Każdy z nich jest niezależnie namawiany do zdrady drugiego. Jeżeli jeden więzień zdradzi, będzie nagrodzony, a drugi więzień zostanie ukarany. Jeżeli obaj zdradzą, obaj będą uwięzieni i torturowani. Jeżeli żaden nie zdradzi, obaj otrzymają umiarkowaną nagrodę. Zazwyczaj jednak samolubny wybór zdrady jest zawsze bardziej nagradzany niż współpraca – niezależnie od decyzji drugiego więźnia – jeżeli jednak obaj zdradzą wyjdą na tym gorzej, niż gdyby współpracowali. Na tym też polega dylemat więźnia.

W formalny sposób gra jest reprezentowana przez macierz wypłat, która określa wypłatę (nagrodę) dla każdego gracza:

$$\begin{array}{cc} D & \begin{array}{cc} W & Z \end{array} \\ \begin{array}{c} W \\ Z \end{array} & \begin{bmatrix} (6,6) & (0,10) \\ (10,0) & (2,2) \end{bmatrix} \end{array}$$

gdzie D – decyzja graczy, W – współpraca, Z – zdrada.
Decyzja pierwszego gracza sprowadza się do wyboru wiersza tej macierzy, natomiast decyzja drugiego gracza sprowadza się do wyboru jej kolumny. Element macierzy wyznaczony przez przecięcie wybranego wiersza i kolumny określa wynik gry:

- jeżeli obaj gracze współpracują, to każdy z nich otrzymuje wypłatę 6,
- jeżeli obaj gracze zdradzą, to każdy z nich otrzymuje wypłatę 2,
- jeżeli jeden zdradzi, a drugi będzie współpracował, wówczas ten, który zdradził otrzyma wypłatę 10, natomiast ten, który współpracował nie otrzyma nic.

Problem robi się ciekawszy, gdy grę będziemy wielokrotnie powtarzać z udziałem tego samego gracza lub graczy, dzięki czemu przy podejmowaniu decyzji o współpracy bądź zdradzie będą mogli kierować się oni historią przeszłych zachowań. Ten tak zwany iterowany dylemat więźnia przyciągał od wielu lat uwagę badaczy zajmujących się teorią gier. Zorganizowano nawet dwa turnieje programów komputerów, w których spośród 76 konkurentów najlepszym okazał się ten stosujący bardzo prostą strategię, znaną pod nazwą „wet-za-wet". Strategia ta polega na współpracy w pierwszym ruchu i naśladowaniu przeciwnika w każdym następnym.

3.2. Algorytm genetyczny w dylemacie więźnia

Naukowiec Axelrod, który w latach 80. studiował zaciekle problem dylematu więźnia nie mógł uwierzyć, iż tak prosta strategia jak „wet-za-wet" okazała się być w ogólności najlepsza. Postanowił więc wykorzystać algorytm genetyczny do odnalezienia bardziej optymalnej strategii, która w ogólności mogłaby być lepsza od wspomnianej strategii „wet-za-wet". Punktem wyjścia do stworzenia algorytmu genetycznego było założenie, że reguły decyzyjne szukanej strategii mogą opierać się na ostatnich trzech posunięciach obu stron.

Osobnik

Aby zrozumieć budowę każdego osobnika w populacji, reprezentującego strategię gry, musimy najpierw przeanalizować sposób myślenia Axelrod`a. Jak już wcześniej wspomniałem wyszedł on od analizy trzech ostatnich ruchów obu graczy. W każdym ruchu istnieją cztery możliwości zachowania się graczy, które możemy odpowiednio zakodować:

- obaj gracze współpracują – WW – kodujemy jako 0,
- pierwszy gracz współpracuje, a drugi zdradza – WZ – kodujemy jako 1,
- pierwszy gracz zdradzam a drugi współpracuje – ZW – kodujemy jako 2,
- obaj gracze zdradzają – ZZ – kodujemy jako 3.

Korzystając z tak zakodowanych zachowań graczy w danym ruchu, możemy również zakodować zachowanie graczy w trzech ostatnich ruchach, np.:

- 000 – reprezentuje sytuację, w której gracze cały czas współpracowali,
- 113 – reprezentuje sytuację, w której pierwszy z graczy grał naiwniaka w pierwszych dwóch ruchach i współpracował, a następnie zdradził, natomiast drugi z graczy wciąż zdradzał.

Każdy kod reprezentujący zachowanie graczy w trzech ostatnich ruchach można interpretować jako trzycyfrową liczbę w systemie czwórkowym, której odpowiednikiem jest liczba dziesiętna z zakresu od 0 do 63.

Korzystając z powyższych zależności wiemy już, że dana strategia musi mieć opracowaną odpowiedź na każdą z 64 możliwości, a więc do zakodowania osobnika wystarczą 64 bity, i w ten sposób i-ty bit będzie odpowiedzią na i-tą sytuację: zdrada albo współpraca. Jednakże to nie wszystko, ponieważ musimy mieć opracowaną strategię również na pierwsze dwa ruchy w grze. Początkowo Axelrod założył, że należy współpracować z przeciwnikiem, ale ostatecznie doszedł do wniosku, że warunki początkowe też mogą mieć znaczenie na dalszy przebieg gry i postanowił wprowadzić do kodu osobnika dodatkowych 6 bitów reprezentujących, tzw. przesłanki gry, czyli hipotetyczne 3 ruchy, które miały miejsce jeszcze przed grą. Na podstawie tych 6 bitów, osobnik postępując zgodnie z poprzednio określonymi regułami mógł rozpocząć grę wybierając odpowiednią strategię.

Ewolucja populacji

Po określeniu kodowania osobnika, Axelrod przystąpił do poszukiwania lepszych strategii gry. W tym celu utworzył grupę składającą się ze znanych 8 najlepszych strategii i zrealizował swego rodzaju turniej między osobnikami populacji, a każdą strategią z tej grupy. Populacja liczyła 20 osobników, z której każdy rozgrywał pojedynek złożony ze 151 ruchów z każdym z ośmiu wspomnianych przeciwników. Funkcją przystosowania osobników była średnia punktów zdobyta we wszystkich pojedynkach. Populacja startowała od losowo wygenerowanych osobników. Do ewolucji populacji wykorzystano dwa podstawowe operatory genetyczne: krzyżowanie i mutację. Metoda selekcji preferowała oczywiście osobników, którzy radzili sobie najlepiej z grupą ośmiu wspaniałych.

Wyniki badań

Przeprowadzone badania mile zaskoczyły Axelrod`a. Okazało się, że algorytm genetyczny z losowo utworzonej populacji wyewoluował osobników reprezentujących strategie w ogólnej klasyfikacji lepsze od najlepszej strategii „wet-za-wet”. Najciekawsze jest to, że niektóre wzory zachowań rozwinęły się u zdecydowanej większości osobników, np.:

- nie zatapiaj łodzi – przedłużaj współpracę po trzech kolejnych współpracach – W po WW,WW,WW
- bądź podatny na prowokację – zdradzaj, jeżeli inny gracz cię zdradził po dłuższej współpracy – Z po WW,WW,WZ
- akceptuj przeprosiny – kontynuuj współpracę, gdy partner ją wznowił – W po WZ, ZW, WW
- zapominaj – współpracuj, gdy współpraca została nawiązana po jej naruszeniu – W po ZW, WW, WW
- trzymaj się rozwiązania rutynowego – zdradzaj po trzech kolejnych zdradach – Z po ZZ, ZZ, ZZ

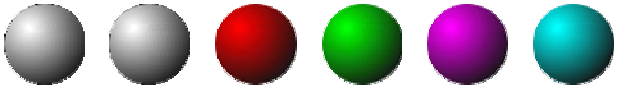
4. MasterMind – AG szuka optymalnej decyzji w grze

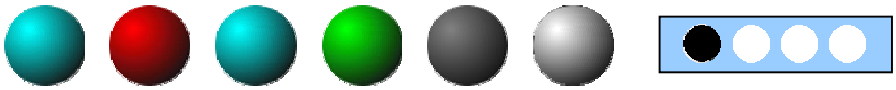
4.1. Zasady gry

MasterMind jest grą dwuosobową, w której jeden z graczy jest kodotwórcą (*codemaker*), a drugi kodołamaczem (*codebreaker*). Kodotwórca określa k-elementową kombinację (może być z powtórzeniami) elementów z jasno sprecyzowanego zbioru n-elementowego – mogą to być liczby, kolory, obrazki, przedmioty codziennego użytku. Kodołamacz z kolei musi odgadnąć zakodowaną kombinację poprzez wybieranie własnych kombinacji. Dla każdej zagranej kombinacji, kodotwórca odpowiada kodołamaczowi ile elementów znajduje się na właściwych pozycjach, a ile elementów jest poprawnych, ale nie znajduje się na właściwych pozycjach. Zadaniem kodołamacza jest odszyfrowanie kodu kodotwórcy w najmniejszej liczbie prób.

W przypadku analizowanej gry stajemy w roli kodotwórcy zadającemu komputerowi (kodołamaczowi) tajemniczy kod do odszyfrowania [3]. Do dyspozycji mamy zbiór 9 kolorów, z których możemy utworzyć kombinację maksymalnie 8-elementową.

Przykład (6-elementowa kombinacja elementów ze zbioru 9-elementowego):

tajemnicza kombinacja 

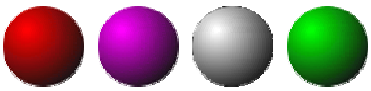
próba odgadnięcia 

Kodołamacz dowiaduje się, że w tej próbie:

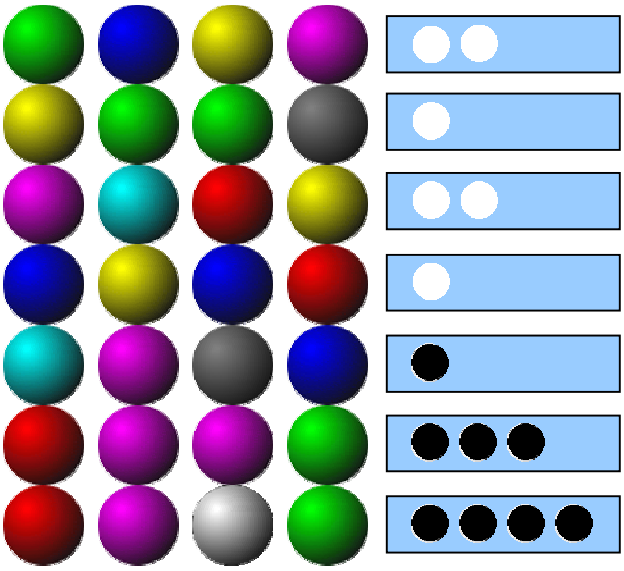
- 1 kolor znajduje się na swoim miejscu (1 czarny kołek),
- 3 kolory są poprawne, ale nie znajdują się na swoich miejscach (3 białe kołki).

Oczywiście kodołamacz nie wie, które to są kolory.

Poniżej znajduje się przykładowa gra z omawianym dalej programem Genetic MasterMind:
(4-elementowa kombinacja ze zbioru 9-elementowego)



Obliczono 1400 kombinacji ze wszystkich 6561 możliwych.



4.2. Problem do rozwiązania w MasterMind

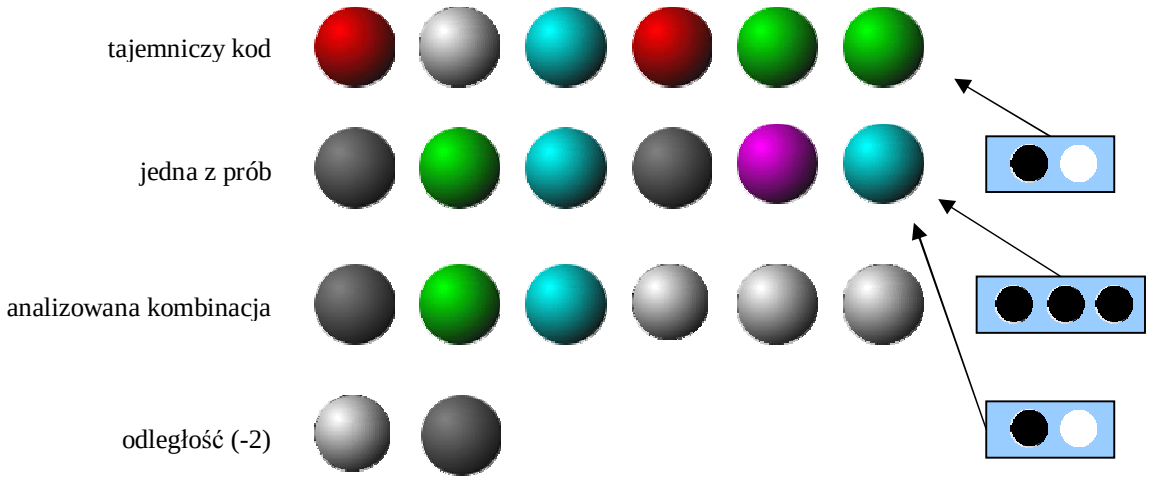
Uogólniony problem mastermind`a można przedstawić jako poszukiwanie ukrytego kodu przy wykorzystaniu podpowiedzi dostarczanych z czarnej skrzynki. W szczególności, tajemniczy kod oraz próby zgadywania tworzone są z elementów zbioru o liczbie kardynalnej N. Ukryty kod oraz każda próba odgadnięcia jest kombinacją K-elementową z powtórzeniami. Oznacza to, że cała przestrzeń poszukiwań zawiera N^K kombinacji.

Gra MasterMind sprzedawana jest w dwóch wersjach:

- klasycznej – gdzie N=6 i K=4, co łącznie daje 1296 kombinacji,
- „super MasterMind" – gdzie N=8 i K=5, co łącznie daje 32768 kombinacji.

Każda podpowiedź w formie białych (reprezentujących poprawny element w nieodpowiednim miejscu) i czarnych (reprezentujących poprawny element w odpowiednim miejscu) kołków dostarczona przez kodotwórcę ogranicza przestrzeń poszukiwań, wykluczając z niej część możliwych kombinacji. Jednakże z powodu wielowymiarowości tej przestrzeni wyznaczenie wykluczonej podprzestrzeni nie jest wcale takie proste.

Zaproponowano tym samym pewien sposób oceniania kolejnych kombinacji w stosunku do zrealizowanych w trakcie gry prób. Sposób ten korzysta z podpowiedzi skojarzonych z odpowiednimi próbami i stara się ocenić czy dana kombinacja może znajdować się w ograniczonej przestrzeni poszukiwań. Obliczana jest tzw. odległość danej kombinacji do zgodności z odpowiednią próbą. Zgodność jest tutaj rozumiana jako zgodność podpowiedzi do próby względem łamanego kodu z podpowiedzą do danej kombinacji względem tej próby. Odległość natomiast jest tutaj rozumiana jako najmniejsza liczba zmian kolorów w kombinacji potrzebna na uzyskanie zgodności z próbą.. Prześledźmy ten „zagmatwany" mechanizm oceniania kombinacji na poniższym przykładzie:



Pierwszy rząd kolorów reprezentuje tajemniczy kod stworzony przez kodotwórcę. Drugi rząd reprezentuje jedną ze zrealizowanych prób kodołamacza. Jak widać kodotwórca odpowiedział wówczas, iż jedna kula jest na swoim miejscu, a jedna nie. Trzeci rząd reprezentuje jedną z analizowanych przez kodołamacza kombinacji. Kombinację tę ocenia sam kodołamacz względem odpowiedniej próby (w tym przypadku wiersz drugi). Okazuje się, że w tej kombinacji trzy kule są na swoim miejscu (czarna, zielona oraz niebieska). Jak widać, podpowiedź kodotwórcy nie jest zgodna z oceną/podpowiedzią utworzoną przez kodołamacza. Aby osiągnąć omawianą zgodność należy np. zamienić pierwszą kulę czarną na kulę szarą oraz drugą kulę zieloną na kulę czarną. Jest to najmniejsza liczba zmian kul potrzebna do uzyskania zgodności. Wynika stąd, że odległość analizowanej kombinacji wynosi -2.

Sformułowany w ten sposób problem poszukiwania ukrytej kombinacji jest problemem optymalizacji kombinatorycznej: każda kombinacja posiada swoją ocenę, która jest maksymalizowana (do maksimum, którym jest 0) względem każdej próby.

4.3. Techniki gry w MasterMind

Pierwszy znany algorytm rozwiązujący MasterMind został opublikowany przez Donald`a Knuth`a w 1977r. Po nim zostało opublikowanych jeszcze wiele innych podejść. Wszystkie jednak sprowadzały się do jednej z następujących kategorii:

- **stepwise optimal** – każda próba odgadnięcia jest optymalna wtedy gdy jest zgodna ze wszystkimi wykonanymi do tej pory próbami; postępując zgodnie z tą techniką zapewnione jest osiągnięcie rozwiązania, ponieważ każda podpowiedź do próby wyklucza niepusty zbiór kombinacji; w każdym kroku liczba możliwych kombinacji jest redukowana i ostatecznie pozostaje tylko jedna możliwość,
- **strategically optimal** – w tym przypadku gramy kombinacjami, o których z góry wiemy, że są nieprawidłowe z zamierzeniem wydobycia informacji o kształcie ukrytej kombinacji lub dalszym zredukowaniu pozostałej przestrzeni poszukiwań; gra tego typu może wyglądać następująco: na początku gramy tylko jednym kolorem (kombinacja złożona z wszystkich elementów tego samego koloru), następnie kolejnym kolorem, itd. wydobywając w ten sposób informację o kolorach oraz ich liczbie w ukrytym kodzie; po zastosowaniu grupy „nieoptymalnych" kombinacji, prawidłowe rozwiązanie jest dedukowane analitycznie; zaletą tego podejścia jest gwarancja znalezienia odpowiedzi w stałej liczbie prób; wadą jednak jest fakt, że zwykle liczba ta jest większa od optymalnej; zatem w ogólności podejście to może być najgorsze; strategia ta została początkowo użyta w rozważaniach Knuth`a.

W przypadku strategii **stepwise optimal** istnieje kilka sposobów konsekwentnego wybierania kolejnego optymalnego ruchu:

- **gruntowne przeszukiwanie (exhaustive search)** – rozważa po kolei wszystkie kombinacje i wyklucza te, które nie są zgodne ze zbiorem wykonanych prób; istnieje wiele sposobów na przeszukiwanie w ten sposób przestrzeni w zależności od tego skąd startujemy i w którą stronę podążamy (np. jeśli zaczynamy od 1111, to następnie analizujemy 1112 aż do 1116, a następnie 1121 i tak dalej); sposób ten może być jeszcze dodatkowo ulepszony o pewne heurystyki do wykluczania kombinacji (np. jeśli próba nie miała żadnego poprawnego koloru, to natychmiast odrzucaj kombinacje zawierające którykolwiek z tych kolorów); zaletą tego sposobu jest jednorazowe przejście przestrzeni; natomiast wadą tego sposobu jest fakt przechowywania informacji o całej przestrzeni poszukiwań, która docelowo będzie zajmować w pamięci tyle bajtów ile jest kombinacji; oczywiście nie stanowi to problemu dla gry 4x6 (4-elementowe kombinacje zbioru 6-elementowego), która zawiera 1296 kombinacji, ale stanowi niemożliwą do przekroczenia barierę dla uogólnionej gry, np. 10x8, która zajęłaby około 1 TB pamięci. Podejście to zostało zrealizowane m.in. przez Koyamę.
- **losowe przeszukiwanie (random search)** – losowo generuje kombinacje i gra pierwszą zgodną ze zbiorem wykonywanych prób; wadą tego sposobu jest możliwość powtarzania generowanych kombinacji, a więc w pewnym sensie wielokrotnego błądzenia po całej przestrzeni poszukiwań; dużą zaletą na pewno jest niewielkie zapotrzebowanie na pamięć, ponieważ w danej chwili należy zapamiętać tylko jedną kombinację oraz zbiór zrealizowanych prób.

Knuth dowiódł, że do rozwiązania klasycznego MasterMind`a wystarczy co najwyżej 5 prób, uzyskując jednocześnie średnio 4.478 prób na grę. Wynik ten został następnie poprawiony przez Irving`a oraz Neuwirth`a na 4.364 prób na grę. Jednak nie dowiedziono optymalności żadnej z tych strategii, więc pole do popisu było wciąż otwarte i rzeczywiście Koyama oraz Lai poprawili średnią na 4.34 opierając się na gruntownym przeszukiwaniu (*exhaustive search*). Następnie Bestavros oraz Belal do rozwiązania gry wykorzystali teorię informacji uzyskując jednocześnie najlepszy do tej pory średni wynik 3.8 ± 0.6 prób na grę.

4.4. Algorytm genetyczny w MasterMind

Twórcy omawianego programu MasterMind próbowali rozwiązać uogólniony problem mastermind`a przy pomocy algorytmów genetycznych wykorzystujących dodatkowo omawianą wcześniej strategię *stepwise optimal*.

Algorytm genetyczny wykorzystany do rozwiązania MasterMind`a zasadniczo korzysta z algorytmu *stepwise optimal* i w każdym kroku stara się minimalizować dystans do optymalnej kombinacji, tzn. takiej, która jest zgodna ze wszystkimi dotychczasowymi próbami.

Osobnik

Genetic MasterMind nie wykorzystuje specjalnych reprezentacji danej kombinacji, która jest osobnikiem każdej populacji. Wszystkie kombinacje są przetwarzane bezpośrednio na łańcuchach określających odpowiednie kolory, a także wszystkie operatory genetyczne umożliwiają bezpośrednie działanie na tych kombinacjach. W rzeczywistości, Genetic MasterMind wykorzystuje EO (Evolving|Evolutionary objects), które umożliwia przetwarzanie (ewoluowanie) dowolnych obiektów.

Operatory genetyczne

W Genetic MasterMind zostało wykorzystanych kilka operatorów genetycznych zmniejszających oraz zwiększających różnorodność osobników:

- **transpozycja (transpose)** – wykonuje permutacje kolorów z prawdopodob. 40%,
- **mutacja (mutation)** – zamienia jeden z kolorów na kolor następny (cyklicznie) z prawdopodobieństwem 20%,
- **krzyżowanie (crossover)** – tradycyjne krzyżowanie jednopunktowe pomiędzy dwoma kombinacjami z prawdopodobieństwem 40%,
- **zamiana (zapping)** – opcjonalny operator, który zamienia jeden z kolorów na inny losowy (tradycyjna mutacja) – operator ten jednak nie został wykorzystany ze względu na cykliczną mutację, która działa w bardziej deterministyczny sposób.

Selekcja

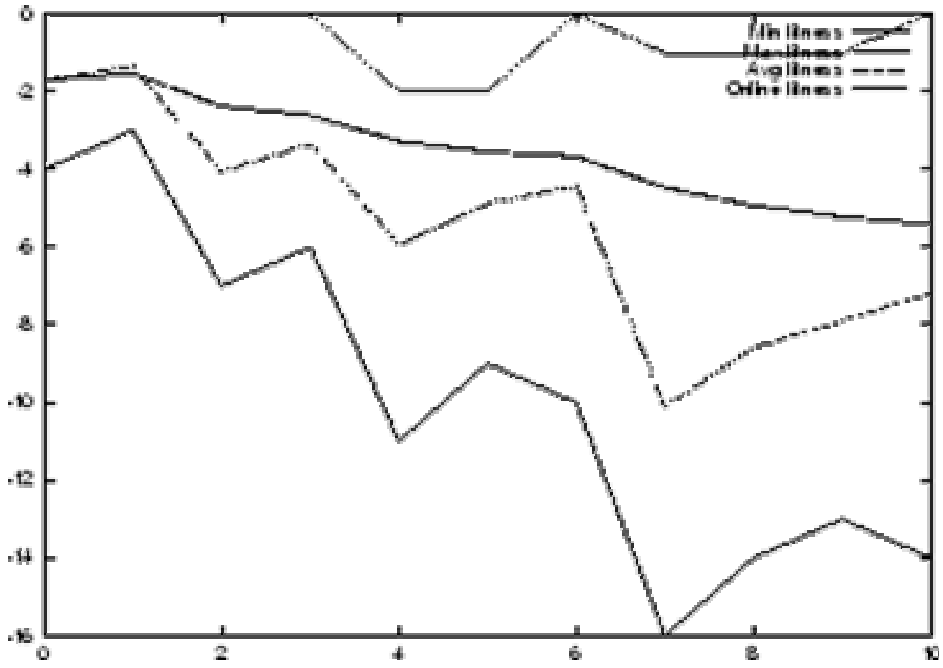
W Genetic MasterMind wykorzystano selekcję elitarną. W każdym pokoleniu eliminowanych jest 50% najgorszych kombinacji. W ich miejsce wchodzą potomkowie powstali w wyniku zastosowania operatorów genetycznych na pozostałych osobnikach. Funkcją przystosowania każdego osobnika jest suma odległości do wszystkich zrealizowanych prób. Jest ona jednocześnie bardzo dynamiczna, tzn. przystosowanie osobnika, którym zagraliśmy, ale nie odszyfrowaliśmy kodu, staje się liczbą ujemną wyliczoną jako liczba czarnych kołków pomniejszona o liczbę białych kołków oraz długość kombinacji. Okazuje się, iż również przystosowanie reszty osobników w populacji jest modyfikowane, ponieważ do ich przystosowania dochodzi odległość do ostatniej próby. Zwiększa się tym samym różnorodność osobników, ponieważ osobnik, który w poprzednim pokoleniu mógł mieć wysokie przystosowanie, teraz może mieć jedno z gorszych. W ten sposób reinicjalizacja populacji nie jest prawie w ogóle potrzebna.

Ewolucja populacji

Domyślnie w populacji znajduje się 400 osobników. Jeśli w przeciągu 15 pokoleń algorytm genetyczny nie wytworzy zgodnego osobnika (o przystosowaniu równym 0), to cała populacja jest niszczona i w jej miejsce powstaje nowa populacja z losowo wybranymi osobnikami. Jest to oczywiście drastyczny sposób traktowania populacji, ale jak się okazuje bardzo dobrze wpływa na zwiększenie różnorodności osobników. Poza tym sposób ten rzadko kiedy jest wykorzystywany – zazwyczaj algorytm genetyczny radzi sobie z wyewoluowaniem odpowiedniego osobnika.

Ewolucja funkcji przystosowania

Poniżej znajduje się wykres przedstawiający ewolucję funkcji przystosowania dla przypadku N=8 i K=5. Zestawione zostały wykresy maksymalnej, minimalnej oraz średniej wartości funkcji przystosowania dla danego pokolenia wraz z średnią wartością funkcji przystosowania dla wszystkich osobników ze wszystkich populacji (najbardziej płynny wykres).



Moment, w którym maksymalna wartość przystosowania osiąga 0, oznacza odnalezienie zgodnej kombinacji, którą program wykonuje ruch. Jak widać miało to miejsce w 0,1,2,3,6 oraz 10 populacji. Warto zauważyć, że po każdej próbie (a więc po wymienionych wcześniej pokoleniach) wszystkie wartości funkcji przystosowania maleją. Jest to efekt omawianej już wcześniej dynamicznej cechy tej funkcji, tzn. po każdej próbie, przystosowania osobników modyfikowane są o odległość do tej próby. Rozpiętość funkcji przystosowania wciąż rośnie (w sensie różnicy między maksymalną a minimalną wartością), co w oczywisty sposób wpływa na średnią wartość przystosowania.

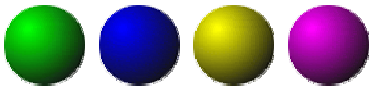
Dodatkowe spostrzeżenia

Głównym problemem omawianego algorytmu jest utrzymanie różnorodności osobników. Jeśli przez wiele pokoleń algorytm nie może znaleźć zgodnej kombinacji, to znaczy że populacja wchodzi w zastój i należy ją po prostu zresetować. Aby uniknąć problemu zastoju populacji można zwiększyć prawdopodobieństwo występowania mutacji, jednakże rozwiązanie takie może sprowadzić algorytm na zwyczajne przeszukiwanie losowe.

W przeprowadzonych doświadczeniach przeprowadzono testy dla dwóch rodzajów kombinacji inicjujących:

- każdy kolor występuje podwójnie (1) – proponowane przez Knutha jako teoretycznie lepsze, np.

- wszystkie elementy są różnych kolorów (2), np.



Wyniki badań

Testy przeprowadzono na komputerach AMD K6-2 400 MHz oraz Pentium 166 porównując zaprojektowany algorytm genetyczny z innymi znanymi podejściami rozwiązującymi MasterMind.

Wyniki testów dla klasycznego przypadku MasterMind, tzn. N=6 oraz K=4:

Autor	Średnia prób	Średnia kombinacji
Knuth	4.478	Wszystkie
Koyama	4.340	Wszystkie
Bestavros	3.8 ± 0.5	Wszystkie
Rosu	4.66	Przeszukiwanie losowe
Genetic MasterMind (1)	4.132 ± 1.00	279 ± 188
Genetic MasterMind (2)	4.312 ± 0.989	320 ± 268

Jak widać testy wypadły pomyślnie, ponieważ średnia liczba prób potrzebna na odszyfrowanie kodu jest w granicach najlepszych znanych wyników. Okazało się również, że teoretycznie lepsze rozpoczęcie gry (1) w ogólności dało lepszy rezultat. Genetic MasterMind uruchomiono 1000 razy z populacjami liczącymi 100 osobników.

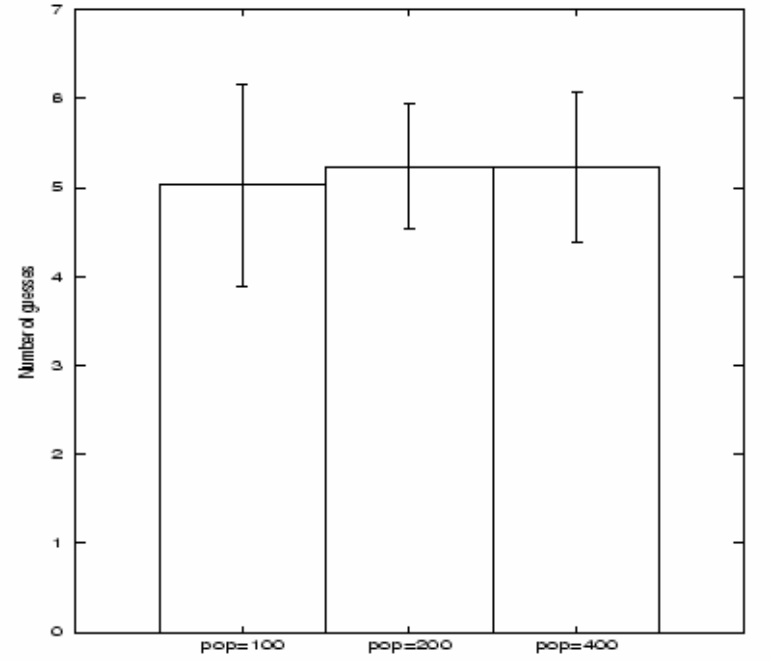
Wyniki testów dla przypadku „super MasterMind", tzn. N=8 oraz K=5:

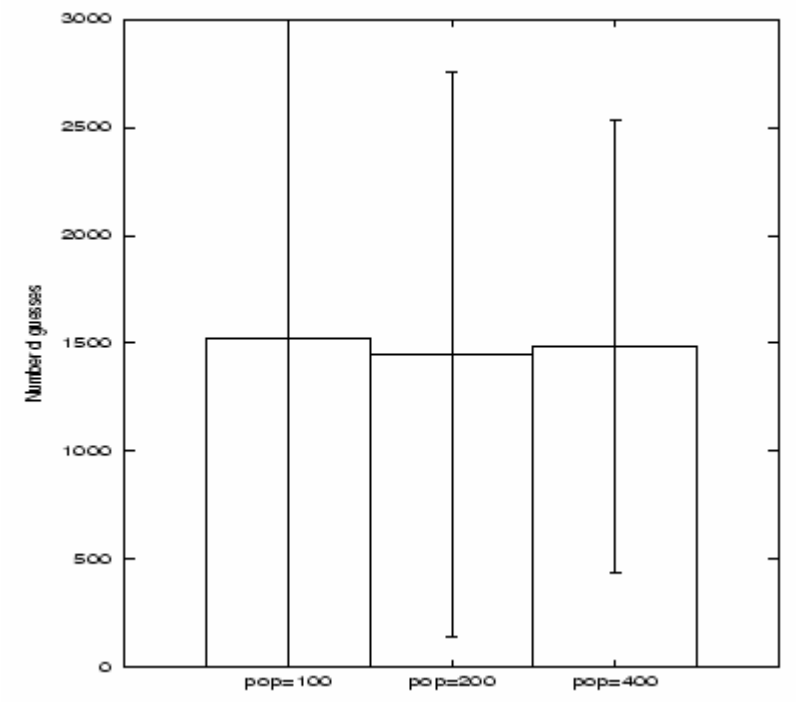
Autor	Średnia prób	Średnia kombinacji
Rosu	5.88	Przeszukiwanie losowe
Bento et al.	6.866	1029.9
Genetic MasterMind	5.904 ± 0.975	2171 ± 1268

Jak widać w tym przypadku testy również wypadły pomyślnie. Genetic MasterMind uruchomiono 500 razy z populacjami liczącymi 400 osobników.

Powyższe testy wykazały, że Genetic MasterMind uzyskiwał nieraz lepsze wyniki od innych znanych metod, uzyskując jednocześnie podobną średnią liczbę prób. Warto przy tym zwrócić uwagę, iż wyniki te zostały osiągnięte przy przeszukiwaniu tylko niewielkiej części całej przestrzeni poszukiwań.

Przeprowadzono również testy sprawdzające wpływ rozmiarów populacji na jakość rozwiązań dla przypadku N=6 oraz K=6.

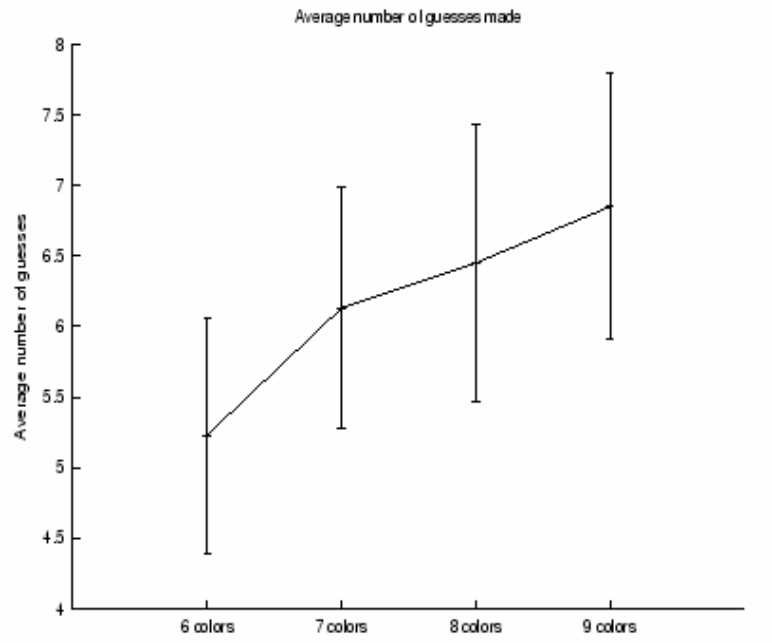


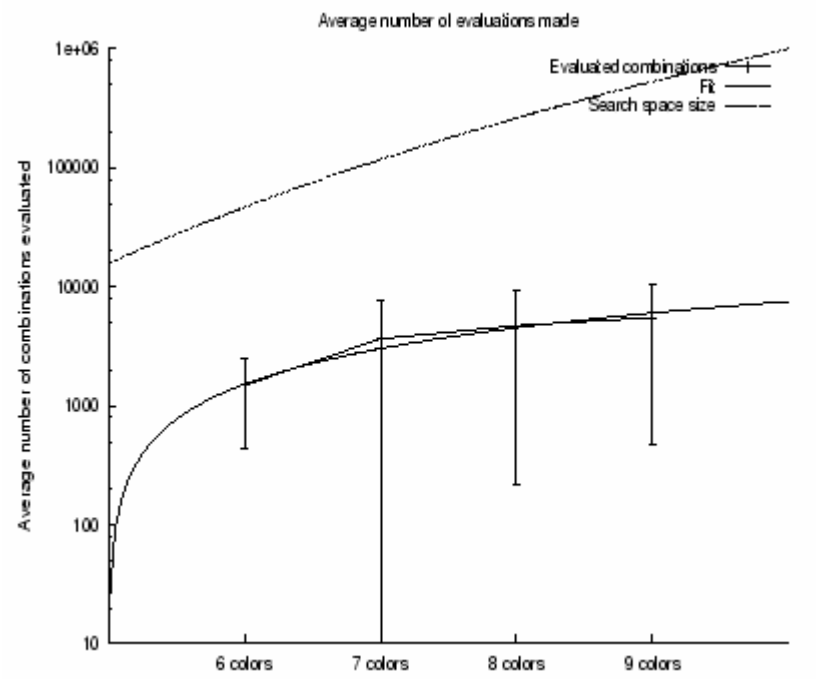


Pierwszy wykres przedstawia zależność średniej liczby prób od rozmiarów populacji. Natomiast drugi wykres przedstawia zależność liczby obliczonych kombinacji od rozmiarów populacji. Okazuje się, że średnia liczba prób oraz liczba analizowanych kombinacji nie zależą w znaczny sposób od rozmiarów populacji. Wzrost rozmiaru populacji wpływa jedynie na zmniejszenie odchylenia standardowego (praktycznie w obu przypadkach).

Powyższe testy wykazały, że wykorzystanie większych populacji nieznacznie poprawia wyniki rozwiązywania, a średnia liczba prób pozostaje praktycznie taka sama.

Ostatecznie wykonano jeszcze testy sprawdzające zaradność algorytmu dla problemów różnej skali, ponieważ z założenia program miał rozwiązywać uogólniony problem MasterMinda (tzn. dla dowolnego N oraz K). Testy przeprowadzono 100 razy dla problemu 6-elementowej kombinacji ze zbioru od 6- do 9- elementowego.





Pierwszy wykres przedstawia zależność średniej liczby prób od liczby dostępnych kolorów, natomiast drugi – zależność średniej liczby wyznaczonych kombinacji (z aproksymacją) oraz liczby wszystkich kombinacji od liczby dostępnych kolorów (drugi wykres jest w skali logarytmicznej). W obu przypadkach populacje liczyły 400 osobników.

Dla tak przeprowadzonych testów ciekawość może budzić drugi wykres, ponieważ wraz ze wzrostem liczby kolorów (elementów) średnia liczba wyznaczonych kombinacji rośnie wolniej od liczby wszystkich możliwych kombinacji. Zaspakajając naszą ciekawość autorzy programu wyznaczyli, że rozmiar przestrzeni poszukiwań rośnie ekspotencjalnie, natomiast liczba wyznaczanych kombinacji rośnie liniowo (wyznaczono $f(x) = ax + b$, *gdzie* $a = 1504.22 \pm 189.1$ *i* $b = 1519.26 \pm 143.8$).

Wnioski

Przeprowadzone badania udowodniły, że algorytm ewolucyjny jest skutecznym sposobem rozwiązywania problemów typu *oracle-type* takich jak Mastermind. Algorytm genetyczny wypadł na równi, a miejscami nawet lepiej, od najbardziej optymalnych algorytmów. Jego główną zaletą jest fakt, że może być stosowany do problemów o dowolnym rozmiarze (dowolne N i K).

5. Podsumowanie

W niniejszej pracy zostały przedstawione dwa podejścia stosujące algorytmy genetyczne w grach logicznych. W obu przypadkach zastosowanie algorytmów ewolucyjnych dało zaskakująco dobre rezultaty. Jednakże nie zawsze jest tak różowo. Wykorzystanie drugiego podejścia stosującego algorytm genetyczny na każdym etapie gry, ma swoją jedną zasadniczą wadę. Złożoność obliczeniowa potrzebna na ewoluowanie kolejnych populacji może być na tyle kosztowna, że aż nieopłacalna. Niemniej jednak podejście to stanowi pewną alternatywę na przyszłość, ponieważ zgodnie z prawem Moore`a moc obliczeniowa komputerów podwaja się średnio co półtora roku. Podejście to ma również szansę zaistnienia w problemach bardziej złożonych, dzięki swojej własności przeszukiwania tylko niewielkiej części ogromnej przestrzeni rozwiązań. W grach logicznych na pewno dużo częściej stosuje się pierwsze podejście wyszukujące optymalnej strategii gry. Chociaż sam etap doświadczeń może być również czasochłonny, to jednak odnaleziona strategia może bić wszystkie inne dotąd znane na głowę. Następnie zostaje ona na sztywno wbudowana w program grający, a sam algorytm genetyczny pozostaje dla nas jedynie miłym wspomnieniem.

6. Literatura

[1] Goldberg E. David: *Algorytmy genetyczne i ich zastosowania*, WNT, Warszawa 2003, s. 155-157.
[2] Michalewicz Zbigniew: *Algorytmy genetyczne + struktury danych = programy ewolucyjne*, WNT, Warszawa 1999, s. 49-51.
[3] Kwaśnicka Halina, Spirydowicz Artur: *UCZĄCY SIĘ KOMPUTER Programowanie gier logicznych*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2004.
[4] Genetic MasterMind - http://geneura.ugr.es/~jmerelo/GenMM/GenMM.shtml