ZASTOSOWANIE ALGORYTMU GENETYCZNEGO DO KONSTRUKCJI FUNKCJI OCENIAJĄCEJ W GRZE *OTHELLO*

Tomasz Dąbrowski, Halina Kwaśnicka, Maciej Piasecki

Wydziałowy Zakład Informatyki, Politechnika Wrocławska,
ul. Wybrzeże Wyspiańskiego 27, 50-370 Wrocław
e-mail: {kwasnicka,piasecki}@ci.pwr.wroc.pl

Typowym (choć nie jedynym) podejściem do projektowania gier logicznych jest tworzenie drzewa gry i szukanie optymalnego ruchu na podstawie oceny poszczególnych węzłów w drzewie. Do oceny węzłów (aktualnego stanu gry) stosuje się tzw. funkcje oceniające. Dobór tych funkcji ma duże znaczenie dla programu grającego, jako jedną z metod doboru próbuje się stosować algorytm genetyczny. W pracy przedstawiono sposób tworzenia programu grającego w grę Othello, gdzie algorytm genetyczny dobiera zestaw wag w funkcji oceniającej. Omówiono uzyskane wyniki.

1 Algorytmy ewolucyjne w grach komputerowych

Klasycznym wyzwaniem sztucznej inteligencji pozostaje od samego jej początku konstrukcja programów grających w gry logiczne. Mimo osiągniętych spektakularnych sukcesów (np. *Deep Blue*) zagadnienie ciągle jest na tyle trudne i ogólne, że może stanowić punkt odniesienia dla wypracowywanych rozwiązań. W niniejszej pracy przedstawiona została próba połączenia klasycznego algorytmu gier logicznych α-β obcinania z metodami ewolucyjnymi. Celem było sprawdzenie możliwości automatycznej konstrukcji funkcji oceniającej – jedynego czysto heurystycznego elementu algorytmu α-β, całkowicie zależnego od rozpatrywanej gry.

Gra jest to rywalizacja, prowadzona przez uczestników gry zgodnie z ustalonymi regułami, mająca na celu osiągnięcie wyznaczonego celu (lub celów). Są gry całkowicie losowe (np. ruletka), gry, w których jakiś element losowy wpływa na rozgrywkę (np. brydż) oraz gry całkowicie deterministyczne, jak np. warcaby czy szachy.

Możliwe podejścia

Gra rozpoczyna się od przyjęcia stanu początkowego, następnie gracze kolejno wykonują swoje posunięcia (podejmują decyzje), aż do osiągnięcia stanu końcowego. W każdym kroku rozgrywki gracze wykonują wybrane przez nich posunięcia spośród zbioru możliwych ruchów. Rozgrywkę można postrzegać jako poszukiwanie takich ruchów gracza, które zapewniają mu zwycięstwo. Taki proces poszukiwania można opisać za pomocą drzewa rozwiązań, którego korzeń jest stanem początkowym gry, a kolejne węzły są stanami, jakie będą aktualne po wykonaniu każdego ruchu spośród możliwych na danym etapie gry (Bolc, Cytowski, 1989). Potomkowie każdego z tych węzłów będą reprezentować stany gry po możliwych ruchach przeciwników. Liście takiego drzewa oznaczają stany końcowe gry: wygrane, remisowe lub przegrane dla gracza. Już dla dwóch graczy i stosunkowo prostej gry, drzewo takie jest bardzo duże: jest to reprezentacja wszystkich możliwych rozgrywek z danego stanu początkowego (dla warcabów jest to około 10^{40} węzłów, dla szachów – 10^{120}).

Strategia gry jest to określenie sekwencji posunięć gracza; *strategia wygrywająca*, to strategia doprowadzająca gracza do zwycięstwa, niezależnie od posunięć przeciwników. Znalezienie strategii wygrywającej wymagałoby zbudowania i przeszukania całego drzewa, co nie jest możliwe, z wyjątkiem bardzo prostych gier. Dlatego też stosuje się różne heurystyki do oceny aktualnego stanu gry i wyboru kolejnych posunięć, bez konieczności budowy całego drzewa. Jakość poszczególnych węzłów (stanów gry) szacuje się na podstawie wybranych cech charakteryzujących pozycje gry. Definiuje się *funkcję wartościującą* (oceniającą) poszczególne węzły. Do oszacowania węzłów trzeba wygenerować pewne poddrzewo gry, w którym za pomocą funkcji oceniającej szacujemy węzły na najgłębszym poziomie wygenerowanego drzewa. Pozostaje pytanie, jak głęboko w dół należy przeglądać drzewo. Z jednej strony, im głębiej program przegląda drzewo, tym większe szanse, że lepiej zostaną oszacowane węzły. Z drugiej strony, wydłuża to czas obliczeń, a gra powinna toczyć się w czasie rzeczywistym: komputer nie powinien „myśleć" zbyt długo.

Człowiek grający np. w szachy czy warcaby ocenia przyszły układ na planszy (stan gry) decydując się na kolejny ruch. W jaki sposób dokonuje tej oceny? Na ile korzysta tu z intuicji i doświadczenia, bo przecież nie sprawdza drzewa rozwiązań ani nie przeprowadza obliczeń funkcji oceniającej, nawet jej nie definiuje. Dla komputera potrzebne jest jednoznaczne kryterium, według którego może on ocenić poszczególne układy na planszy. W tym celu buduje się właśnie funkcje oceniające. Budując taką funkcję należy brać pod uwagę specyfikę gry, wybrać cechy ważne dla strategii gry, (w warcabach może to być liczba pionków gracza i przeciwnika, w grze w kółko i krzyżyk może to być liczba rzędów, w których można ustawić trójkę, itp.). Z reguły funkcje oceniające dany stan gry mają postać ważonej sumy wartości wybranych cech, uznanych za istotne dla danej gry:

$$E = w_1 \cdot C_1 + w_2 \cdot C_2 + ... + w_n \cdot C_n \ , \tag{1}$$

gdzie: E – funkcja oceniająca konfigurację na planszy,
C_i – i-ta cecha istotna dla strategii,
w_i – waga oznaczająca wpływ i-tej cechy na ostateczną ocenę pozycji.

Wyznaczając strategię gry dla programu zakłada się, że przeciwnik wykona najlepszy dla niego ruch. Aby nie przeglądać całego drzewa stosuje się różne algorytmy heurystyczne, np. algorytm cięć α-β (Bolc, 1989).

Ostatnio, wraz ze wzrostem popularności algorytmów ewolucyjnych (**EA**) coraz częściej podejmowane są próby ich wykorzystania do tworzenia programów grających. Wyróżnić tu można dwa główne podejścia:

- Zastosowanie **EA** jako narzędzia wspomagającego proces szukania dobrej strategii gry (Goldberg, 1998). Program grający nie wykorzystuje algorytmu ewolucyjnego, ma wbudowaną znalezioną z pomocą **EA** strategię, która jest strategią programu. Przystosowanie osobników powinno odzwierciedlać wyniki, jakie uzyskuje dana strategia, bądź jakie uzyskuje się z zastosowaniem danego zestawu wag w prowadzonych rozgrywkach. Osobniki z populacji mogą prowadzić rozgrywki między sobą, można wykorzystać „zewnętrznych trenerów", którymi może być dobrany odpowiednio zestaw programów grających lub/i ludzie.

- Nasuwającym się wykorzystaniem **EA** jest konstrukcja odpowiedniej funkcji oceniającej ustawienie na planszy (stan gry). Można tu optymalizować wagi dla wybranego zestawu cech istotnych w danej grze. Do tego zadania nadaje się klasyczny algorytm genetyczny, chromosomem może być wektor wag, przystosowaniem zaś wyniki w rozgrywkach uzyskiwanych z zastosowaniem danego wektora wag. Inna możliwość, to zastosowanie programowania genetycznego do konstrukcji postaci funkcji oceniającej układ na planszy. W obu podejściach, pierwotny zestaw istotnych dla danej gry cech (C_i) musi być podany, natomiast program grający wykorzystuje jedną z metod przeszukiwania drzewa rozwiązań, na przykład algorytm cięć α-β, a do oceny sytuacji na planszy (węzłów w drzewie) stosuje znalezioną przez **EA** funkcję oceniającą.
- Inne podejście, to nie definiowanie funkcji oceniającej lecz zdefiniowanie chromosomu osobników jako potencjalnej strategii gry (Kozieja, 1997). Znaleziona strategia jest implementowana w programie grającym, taki program nie wymaga budowy drzewa ani żadnych heurystyk do oceny sytuacji na planszy. Podejście to jest zwykle trudniejsze, ponieważ wymaga zakodowania w chromosomach możliwych strategii.

- Zastosowanie **EA** jako wbudowanego modułu do programu grającego, który na każdym etapie gry odpowiada za podjęcie optymalnej decyzji przez program grający (Kot 1997).

Zarówno w pierwszym podejściu, jak i w drugim, program może reprezentować kilku graczy, grających różnymi strategiami.

2 Gra OTHELLO

Gra *Othello* w swojej koncepcji jest pochodną gier planszowych typu Go. Wybór *Othello* do eksperymentu był podyktowany jej dość skomplikowaną rozgrywką i brakiem dobrego algorytmu grającego. Głębokość całkowitego drzewa gry jest olbrzymia, co powoduje, że jakość funkcji oceniającej staje się czynnikiem decydującym.

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

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

Głównym celem gry jest zdobycie jak największej liczby pionków. Gra jest rozgrywana na planszy 8×8 (Rysunek 1) za pomocą zbioru dwukolorowych pionków. Każdy z pionków jest czarny po jednej stronie i biały po drugiej. Gra rozpoczyna się w pozycji inicjującej przedstawionej na Rysunku 1. Grę rozpoczynają pionki czarne i jest ona kontynuowana tak długo, aż któryś z graczy nie może wykonać legalnego ruchu. Zwycięzcą jest gracz z większą liczbą pionków. Ruch jest wykonywany przez postawienie na planszy pionka kolorem gracza do góry. Ruch jest legalny, gdy pole przed postawieniem jest puste oraz postawienie pionka umożliwia przejęcie jakiś pionków przeciwnika. Przejęcie pionków następuje przez wzięcie ich ‘w nawias’ w liniach prostych lub po przekątnych. Wzięcie w nawias może być zastosowane

jedynie wobec ciągłej sekwencji (przynajmniej jeden) pionków przeciwnika. Przejmowane pionki zmieniają kolor – stają się pionkami gracza wykonującego ruch. Dzięki tym zasadom gra charakteryzuje się dużą dynamika i niestabilnością sytuacji – prawie do ostatniej chwili większość pionków może zmienić kolor co daje gwałtowną zmianę rezultatu rozgrywki. Specjalnego charakteru nabierają punkty brzegowe, które zajęte przez jednego z graczy dają mu oparcie np. nie do odebrania są punkty narożne.

Z pobieżnej analizy gry wyłaniają się trzy podstawowe strategie.

1. Strategia maksymalnej liczby punktów – prowadzi zwykle do blokady ruchów gracza.
2. Strategia ważonych pól – uwzględniająca tzw. *stabilność* pola (związaną z jego położeniem na planszy – pola *A, B, C* na Rysunku 1) oraz negatywny lub neutralny wpływ zajętości pól na rozgrywkę (pola *X*) .
3. Strategia minimalnej liczby pionków – gracz stosujący tą strategię dąży do minimalizacji ilości własnych pionków po to aby zwiększyć możliwości własnego ruchu – strategia ślepo stosowana prowadzi do przegranej.

Autorzy programów grających w Othello, uwzględniając zaprezentowane powyżej strategie podstawowe, rozszerzyli zbiór czynników mogących mieć wpływ na funkcję oceniającą. Lista zidentyfikowanych w literaturze czynników (np. Kai-Fu, 1990, Rosenbloom, 1982) jest długa jednak wykorzystanie niektórych z nich pociąga za sobą duży wzrost złożoności obliczeniowej. Ponadto niektóre czynniki w swoim ‘działaniu’ się pokrywają. Do rozwiązania prezentowanego tutaj zostały wybrane następujące czynniki:

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 wyliczona na podstawie zmodyfikowanych algorytmów (Kai-Fu, 1990, Rosenbloom, 1982).

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

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

gdzie p jest wartością dla pierwszego gracza a q dla drugiego.

3 Zastosowane metody i rozwiązania

Rezygnując w pierwszym podejściu z szukania całej strategii rozgrywki za pomocą algorytmu genetycznego, skoncentrowano się na zastosowaniu algorytmu genetycznego jedynie do znalezienia, jak najlepszej postaci funkcji oceniającej. Przyjmując powyższe osiem cech, funkcja oceniająca *E* wyraża się wzorem:

$$E = \sum_{i=1}^{8} w_i \cdot C_i \,, \tag{3}$$

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

Przyjmując to założenie, problem znalezienia jak najlepszej funkcji oceniającej sprowadza się do znalezienia wyznaczającego ją wektora wag w_i. Wektor ten w prosty sposób można przekształcić w chromosom:

- każda waga została zakodowana w kodzie binarnym na 8 bitach, przyjmuje wartości z przedziału [–127, 128],
- chromosom powstał przez konkatenację binarnych ciągów,
- porządek wag w chromosomie jest zgodny z wprowadzoną wcześniej numeracją cech.

W implementacji algorytmu genetycznego funkcja przystosowania została wyliczona w oparciu o wyniki osiągane poprzez poszczególne osobniki w rozgrywce z programami trenującymi, gdzie pojedynczym osobnikiem jest implementacja algorytmu α-β wykorzystująca funkcję oceniającą zdefiniowaną poprzez genotyp. Wybór schematu uczenia z nauczycielem został podyktowany dużą złożonością obliczeniową alternatywnego podejścia bazującego na rozgrywce pomiędzy wszystkimi osobnikami w populacji.

W wybranym podejściu jakość osiąganych rezultatów zależy od liczby i jakości programów trenujących (mniejsza liczba ‘trenerów’ prowadzi do uczenia się przez osobniki ich słabych stron). Nie koliduje to jednak z głównym celem eksperymentu, jakim było sprawdzenie możliwości zastosowania analizowanej metody uczenia.

Programy trenujące powinny zostać wybrane spośród najlepszych (co jednak nie warunkuje wyniku końcowego), zadecydowała niestety dostępności kodu poszczególnych programów. Tylko to umożliwiało automatyczne prowadzenie rozgrywek (których liczba sięgała setek dla każdej z populacji).

Na wartość funkcji przystosowania składają się dwa elementy:

- rezultat rozgrywki: zwycięstwo, porażka, remis – wyrażony poprzez premię punktową (w przypadku zwycięstwa i remisu),
- różnica w końcowej liczbie pionków.

Konieczne było ustalenie wartości premii za wygraną na odpowiednio dużym poziomie w stosunku do maksymalnej możliwej różnicy pionków: np. przyjmując wartość premii za zwycięstwo 500, osobnik *A* na 10 rozgrywek ma 9 zwycięstw z wynikiem punktowym 64 : 0 i jedną porażką 31 : 33, co daje mu 5714 punktów; natomiast osobnik *B* ma 10 zwycięstw przewagą 2 punktową, daje mu to 5660. Wyraźnie widać, że premia 500 jest za niska aby zawsze premiować osobniki zwyciężające wszystkie swoje partie. Ostatecznie przyjęto wartość 1000.

Każdy osobnik z populacji rozgrywał po dwa mecze z każdym z nauczycieli (zaczynając raz białymi a raz czarnymi pionkami). Następnie, po obliczeniu funkcji przystosowania, osobniki do reprodukcji były wybierane zgodnie z metodą *wyboru wg reszt bez powtórzeń*. Klasyczna metoda ruletki została odrzucona jako dająca zbyt dużą wariancję. Prawdopodobieństwo mutacji miało różną wartość dla różnych eksperymentów (patrz następny rozdział). Poza mutacją i krzyżowaniem, z prawdopodobieństwem 0,6 dla wszystkich eksperymentów, nie zastosowane żadnego dalszego operatora. Ze względu na

małą wielkość populacji (z powodu dużej złożoności obliczeniowej) zastosowano mechanizm regulacji liczby kopii poprzez użycie *skalowania liniowego* (Goldberg, 1989):

$$f' = a \cdot f + b \ , \tag{4}$$

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 pierwotnego $f'_{maks} = C_{zw} * f_{śr}$, gdzie wartość C_{zw} została ustalona na 2.

Wartość czynników składowych C_i funkcji oceniającej jest bardzo zróżnicowana: waha się od $(-67, 67)$ dla C_7 do $[-3440, 3440]$ dla C_8. Aby zapewnić równomierny wpływ poszczególnych wag w_i na funkcję oceniającą konieczna poddano normalizacji wartości czynników przed wyliczeniem wartości funkcji oceniającej. Dodatkowo, w końcowej fazie rozgrywki, 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. Niezbędne było zastosowanie odpowiedniego rzutowania. Głębokość przeszukiwania drzewa gry zmieniała się w zależności od liczby pustych pól na planszy: od 3 do 5. Ponieważ nie są to wartości zbyt duże, jakość testowanych programów (osobników) zależała w dużej mierze od jakości funkcji oceniającej.

4 Eksperymenty

Przeprowadzono 8 kolejnych eksperymentów z czego dwa ostanie dotyczyły uczenia na bazie rywalizacji wewnątrz populacji. Pierwszy zakończył się niepowodzeniem (brak zbieżności) z powodu oceny osobników tylko na podstawie końcowej różnicy liczby pionków oraz braku rzutowania wartości funkcji oceniającej dla liści.

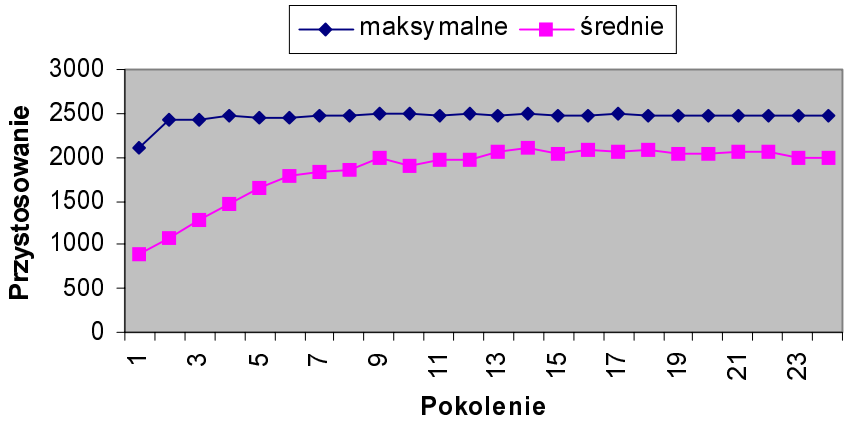
- Eksperyment drugi:

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

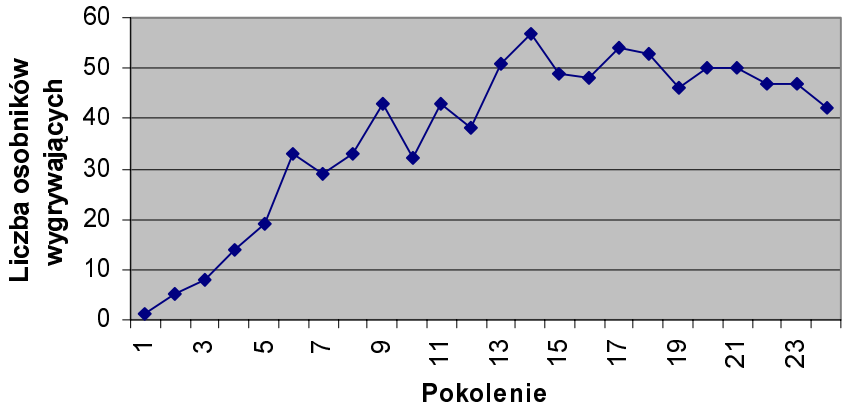
Po poprawieniu błędów uzyskano postęp w uczeniu się widoczny na wykresie (Rysunek 2). Jednocześnie zaobserwowane zostało silne zróżnicowanie pomiędzy najlepszymi osobnikami: wśród zwycięskich osobników trudno było wyznaczyć jakieś typy wag. Prawdopodobnie, ze względu na niewielką liczbę nauczycieli i ich niską jakość, poszczególne osobniki nauczyły się wykorzystywać różne słabe strony nauczycieli. Zaobserwowane wahania w liczbie osobników wygrywających ze wszystkimi nauczycielami były spowodowane przyjętym w eksperymencie drugim zbyt dużym prawdopodobieństwem mutacji.

W trzecim eksperymencie, startując z populacją osobników jak w eksperymencie drugim, zwiększono liczbę nauczycieli do 4 osiągając podobne rezultaty. W eksperymencie 4 została zastosowano inną metodę tworzenia nowego pokolenia: *metodę ścisku De Jonga* (Goldberg, 1989). Procent populacji podlegający reprodukcji został ustalony na 0,2, natomiast współczynnikowi ścisku nadano wartość 3. W rezultacie nastąpił 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 eksperymentach 2 i 3). Wartość

maksymalna funkcji przystosowania osiągnęła swoje maksimum (8949 – co przy czterech nauczycielach oznaczało wysokie wygrane z każdym w każdej partii) w 11 pokoleniu, a następnie utrzymywała się prawie na stałym poziomie (wartość średnia wykazywała stałą tendencję wzrostową).



Rysunek 2. Wykres funkcji przystosowania (eksperyment 2)



Rysunek 3. Liczba osobników, które wygrały wszystkie rozgrywki z nauczycielami (eksperyment 2)

W eksperymencie piątym radykalnie zmniejszono prawdopodobieństwo mutacji, do 0,0001 (w celu skrócenia czasu obliczeń ustalono 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. Ponowne zwiększenie mutacji w eksperymencie szóstym (do 0,001) doprowadziło znowu do większego zróżnicowania zwycięskich osobników.

W eksperymentach siódmym i ósmym podjęto próbę zrealizowania schematu uczenia bez nauczyciela: osobniki rywalizowały wewnątrz populacji – każdy z każdym po dwie gry. Przykładowe parametry symulacji to:

Liczba osobników = 30	Liczba pokoleń = 11
Prawdopodobieństwo mutacji = 0,0033	Maksymalna możliwa ocena = 11848224

Niestety nie zaobserwowano tutaj 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.

5 Podsumowanie

Przeprowadzone eksperymenty wskazują, że algorytmy genetyczne mogą być skuteczne w projektowaniu gier logicznych. Powyższe wyniki należy jednak uznać za etap wstępny, uzasadniający celowość kontynuacji prac. Zwłaszcza dotyczy to dwóch ostatnich eksperymentów. Wydaje się, że ewolucja populacji osobników konkurujących w rozgrywkach między sobą powinna produkować lepsze rozwiązania. Z całą pewnością liczba pokoleń jest tu zbyt mała, trudniej jest znaleźć strategię wygrywającą z dwudziestoma dziewięcioma innymi, na dodatek zmieniającymi się w czasie strategiami niż ze stałymi czterema zewnętrznymi trenerami. Korzystny stabilizujący efekt powinno tu dać założenie, że w każdym pokoleniu zmianie ulega jedynie część populacji.

Interesujące byłoby porównanie wyników uzyskanych przy zastosowaniu programowania genetycznego – zamiast współczynników liniowych produkowane byłyby dowolne funkcje wybranych cech. Podejście to dawałoby możliwość uwzględnienia większego zestawu cech i ewentualnej ich redukcji w trakcie ewolucji.

Algorytmy ewolucyjne mogą być również stosowane do szukania całej strategii gry (Goldberg, 1989, Kozieja, 1997, Kwaśnicka, 1999). Stosunkowo trudnym etapem w tym podejściu jest zakodowanie strategii w postaci chromosomu. W szukaniu strategii dla gry menedżerskiej dobre efekty uzyskano gdy część chromosomu stanowił wektor wskaźników – liczb rzeczywistych, a część drzewa definiujące funkcyjne zależności pomiędzy różnymi wielkościami. Tak zdefiniowane chromosomy wymagają zdefiniowania odpowiednich operatorów genetycznych.

LITERATURA

Rosenbloom P.S. (1982), *A World-Championship-Level Othello Programm*. Artificial Intelligence vol. 19.

Kai-Fu Lee, Mahajan Sanjoy (1990), *The Development of a World Class Othello Program*. Artificial Intelligence vol. 43.

Bolc L., Cytowski J. (1989), *Metody przeszukiwania heurystycznego* (tom I i II), PWN, Warszawa.

Goldberg D.E., (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, Inc.

Kozieja R. (1997), *Zastosowanie algorytmów genetycznych w modelowaniu ekonomicznym: poszukiwanie optymalnej strategii gry kierowniczej*, Praca magisterska, Wydziałowy Zakład Informatyki, PWr, Wrocław.

Kot W. (1997), *Konkurencja firm na rynku – gra symulacyjna*, Praca magisterska, Wydziałowy Zakład Informatyki, PWr, Wrocław.

Kwaśnicka H. (1999), *Obliczenia ewolucyjne w sztucznej inteligencji*, praca niepublikowana, zgłoszona do wydania Oficynie Wydawniczej Politechniki Wrocławskiej.