## Streszczenie

Niniejsza paca zawiera ogólną prezentację wybranych projektów szachowych bazujących na różnego rodzaju podejściach wykorzystujących techniki sztucznej inteligencji. W wyniku dokonanego przeglądu do dalszych badań wybrano metodę uczenia sieci neuronowej na podstawie różnic czasowych, której zastosowanie w programie TD-GAMMON zakończone było obiecującymi wynikami w dziedzinie gry w backgammon. W pracy przedstawiony został autorski program *Neural Chess* uczący się gry w szachy, w którym wykorzystano sieć neuronową trenowaną na funkcję oceniającą pozycje szachowe. Zaprezentowano przy tym szczegółowy opis założeń poczynionych w trakcie tworzenia autorskiego systemu. Główny wysiłek został położony na zaproponowanie właściwego zestawu cech gry w szachy kodującego ważne koncepcje szachowe. Przeprowadzone eksperymenty potwierdziły, iż zastosowanie metod różnic czasowych w grze w szachy daje również bardzo interesujące wyniki. Pomimo tego istnieje jeszcze wiele innych możliwości na dalszy rozwój autorskiego systemu, które przedyskutowano na zakończenie pracy.



## Abstract

This thesis contains an overview of artificial intelligence methods that can be used in the game of chess. It includes general presentation of some chess projects based on various approaches. As a result of this survey author has chosen the method of temporal difference learning in neural network framework for further examination. This method has been used in *TD-GAMMON* with very promising results in the domain of backgammon. This thesis presents author's game-learning program called *Neural Chess* in which neural network has been trained to be an evaluation function for the game of chess. The thesis includes detailed descriptions of assumptions that have been made during *Neural Chess* design. Main effort has been done to propose proper set of chess features encoding important chess concepts. Experiments have confirmed that temporal difference method application is very promising in the game of chess. Although there are ample opportunities for further improvement of author's chess playing system which are discussed at the end of this thesis.

# Wstęp

Gry logiczne już od początku swego istnienia budziły wielkie zainteresowanie ludzkości. Jeszcze większym zaciekawieniem zaczęły się cieszyć w dobie automatyzacji wielu czynności, w której między innymi chciano konstruować automatycznych graczy. Zapewne wiele satysfakcji przynosiło autorom automatyzowanie nawet tak prostych gier jak kółko i krzyżyk. Jednak z biegiem czasu chciano osiągać więcej. Zaczęto interesować się grami coraz bardziej skomplikowanymi o rozbudowanych zasadach gry. Znalazły one szybko swoje miejsce w rozwijającej się do dzisiaj dziedzinie sztucznej inteligencji i stanowią swoisty poligon doświadczalny dla wielu prac naukowych. Wyniki prac nad grami logicznymi odnajdują później bardzo często swoje zastosowanie w zadaniach bardziej praktycznych.

Szczególnym zainteresowaniem zaczęła się cieszyć gra w szachy, w której sztuczna inteligencja miała dorównać ludzkiej. Już od początku XX. wieku pojawiały się prace nad automatyzacją gry w szachy. Jednak największe sukcesy zaczęto osiągać w drugiej połowie XX. wieku. Wtedy też pojawiło się wiele różnych podejść wykorzystujących techniki sztucznej inteligencji w celu usprawniania automatycznej gry w szachy, takie jak indukcja klasyfikatorów końcówek, uczenie się na podstawie wyjaśnień (*explanation based learning*), wnioskowanie na podstawie przypadków (*case based reasoning*) czy dostrajanie funkcji oceniającej pozycje szachowe. Wyniki zastosowania zwłaszcza tych ostatnich technik pod koniec XX. wieku okazały się być bardzo obiecujące i stąd między innymi wzięło się zainteresowanie autora niniejszej pracy możliwością zastosowania ich w praktyce w dziedzinie gry w szachy, którą darzy wielką sympatią.

Dokonanie przeglądu inteligentnych metod wykorzystywanych w usprawnianiu programów szachowych przyczyniło się do wyboru konkretnej metody w autorskim systemie *Neural Chess*. Realizacja systemu pozwoliła osiągnąć główny cel pracy, jakim było stworzenie programu grającego w szachy przy wykorzystaniu sieci neuronowej uczonej metodą różnic czasowych (*temporal difference learning*). Autor niniejszej pracy zachęcony spektakularnymi sukcesami Gerarda Tesauro, który zastosował metodę różnic czasowych w swoim programie *TD-GAMMON* (praca [34]) grającym na bardzo wysokim poziomie w grę *backgammon*, postanowił zbadać efektywność tej techniki w przypadku gry w szachy. Chociaż podobne próby zostały już podjęte kilka lat temu przez trzech naukowców (Baxter, Tridgell, Weaver w pracy [2]) w programie *KnightCap*, który osiągnął dosyć wysoki poziom gry, to jednak postanowiono wypróbować tę technikę przy nieco innych założeniach. Główny nacisk położono tutaj na zgłębienie wiedzy szachowej i zaproponowanie autorskich cech gry, które miałyby znaczący wpływ na ocenę pozycji szachowych. Wydobycie odpowiednich cech gry odbyło się przy starannej analizie jednej z najsłynniejszych książek szachowych [16] należącej do klasyki gatunku, a napisanej przez wybitnego teoretyka szachowego Arona Nimzowitcha. Wyniki przeprowadzonych w ostatniej części pracy eksperymentów wykazały ukierunkowany wzrost umiejętności automatycznego gracza. W podsumowaniu niniejszej pracy wskazano ewentualne możliwości dalszego rozwoju zaprojektowanego systemu *Neural Chess*.

Praca składa się z sześciu rozdziałów oraz siedmiu dodatków.

Rozdział 1. poświęcony jest historycznemu wprowadzeniu w tematykę automatyzacji gry w szachy, w której zaprezentowano interesujące ciekawostki XX. wieku oraz zarysowano tendencje czynionych prac, począwszy od idei stworzenia automatycznego gracza, poprzez pierwsze sprzętowe rozwiązania do ostatecznego zwycięstwa maszyny nad człowiekiem.

Rozdział 2. jest wprowadzeniem w podstawowe tajniki gry w szachy, w którym zdefiniowano planszę gry oraz figury na niej występujące, opisano sposób poruszania się

figur, przedstawiono zasady gry w szachy, a także zdefiniowano kilka pomocniczych pojęć szachowych, które wykorzystywano w dalszej części pracy.

Rozdział 3. zawiera przegląd technik sztucznej inteligencji stosowanych na przestrzeni ostatnich 20-30 lat w usprawnianiu automatycznej gry w szachy. Zaprezentowano wyniki wielu badań naukowych prowadzonych od początku lat osiemdziesiątych XX. wieku, wyróżniając przy tym w ogólnym zarysie cztery główne nurty badań: indukcje klasyfikatorów końcówek, uczenie się na podstawie wyjaśnień, wnioskowanie na podstawie przypadków oraz dostrajanie funkcji oceniających.

Rozdział 4. jest miejscem, w którym zebrano wszystkie założenia dotyczące budowanego systemu. W pierwszej najważniejszej części przedstawiono propozycję autorskich cech gry, na które powinien zwracać uwagę automatyczny gracz. Wydzielono cztery grupy cech: ogólne, związane z figurami, z pionkami oraz królami. Ich wybór został dokładnie umotywowany. Wytłumaczony został sposób ich wyznaczania, uwzględniając fakt, iż stanowią one informację wejściową dla sieci neuronowej. W dalszej części zaprezentowano wizję zaimplementowanego systemu *Neural Chess* służącego między innymi do przeprowadzania eksperymentów uczenia sieci neuronowej metodą *temporal difference learning*. Na koniec przedstawiono budowę sieci neuronowej i przeanalizowano sposób działania wybranego podejścia

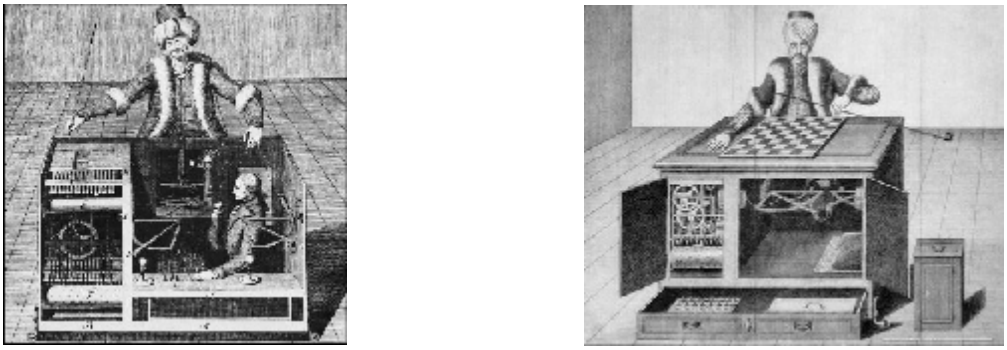
Rozdział 5. zawiera przebieg przeprowadzonych badań eksperymentalnych wraz z ich wynikami oraz komentarzami. Badając efektywność stosowanego podejścia starano się jednocześnie dostroić szereg modyfikowalnych parametrów w celu wybrania najodpowiedniejszego zestawu pozwalającego odnaleźć w procesie uczenia najlepszą strategię grającą w szachy. Zbadano między innymi wpływ stanu początkowego sieci neuronowej na szybkość jej uczenia, wpływ wartości parametrów uczenia na jego efektywność, wpływ głębokości rozwijania drzewa gry na efektywność uczenia oraz wpływ sposobu trenowania sieci neuronowej na jakość gry strategii przez nią reprezentowaną. Na koniec wybrano najlepszą uzyskaną strategię gry i skonfrontowano ją w grze z innymi programami grającymi w szachy.

Rozdział 6. stanowi podsumowanie poczynionych prac, w którym ostatecznie oceniono uzyskane wyniki eksperymentów oraz założenia przyjęte przez autora. Przedstawiono problemy, które pojawiły się w procesie przeprowadzania eksperymentów, starając się jednocześnie wskazać dalsze kierunki działań mających na celu ich wyeliminowanie oraz poprawę otrzymanych wyników.

Dodatki A-D zawierają krótkie opisy obsługi zaimplementowanych przez autora aplikacji *Neural Chess – Wartości Cech Gry* służącej do testowania wartości autorskich cech gry, *Neural Chess – Eksperymenty* służącej do przeprowadzania eksperymentów uczenia sieci neuronowej, *Neural Chess – Turniej* służącej do przeprowadzania turniejów dla dowolnej liczby zaimplementowanych w systemie strategii gry oraz *Neural Chess* służącą do gry użytkownika z najlepszą wyłonioną z eksperymentów strategią gry. Dodatek E zawiera zestawienie neuronów warstwy wejściowej sieci neuronowej otrzymujących informacje o wartościach odpowiednich cech gry. Dodatek F obejmuje zestawienie wszystkich wyników jednego z przeprowadzonych w piątym rozdziale eksperymentów – ze względu na dużą liczbę wykresów przedstawiających te wyniki postanowiono zamieścić je w odrębnym miejscu. W dodatku G natomiast zamieszczono zapisy wszystkich partii rozegranych przez najlepszą wyłonioną w trakcie eksperymentów strategią gry z innymi programami grającymi w szachy.

# 1. Historia automatyzacji gry w szachy

Pierwsze i najbardziej zadziwiające próby stworzenia „automatycznego" szachisty sięgają już XVIII wieku, kiedy to w 1769 roku węgierski inżynier Baron Wolfgang von Kempelen skonstruował dla rozrywki austriackiej królowej Marii Teresy maszynę świetnie grającą w szachy. Była ona zbudowana z gumowego Turka siedzącego przy szafce, na której leżała szachownica z rozstawionymi na niej figurami. W owych czasach niezwykle popularne było tworzenie wszelkich konstrukcji mechanicznych. Kempelen zaskoczył jednak swym wynalazkiem ówczesny świat, głównie ze względu na jego umiejętność gry w szachy, co było rzeczą jak najbardziej niesłychaną i zdumiewającą. Zaczęto się zastanawiać nad tym, w jaki sposób Kempelen tchnął życie w martwą maszynę i zadawano sobie pytania czy to rzeczywiście możliwe, aby owa maszyna potrafiła tak inteligentnie grać w szachy, pokonując wielu silnych amatorów, a także sławnych ludzi, takich jak: Napoleon Bonaparte czy Beniamin Franklin. Wynalazek Kempelena stał się jednocześnie jednym z najsłynniejszych automatów w historii, będącym również głównym tematem wielu ówczesnych opowieści, a nawet współczesnych wydań książkowych. Tajemnica niewiarygodnie silnej gry tej maszyny kryła się w jej wnętrzu… w którym mógł się skryć prawdziwy szachista. Osoba siedząca wewnątrz tego urządzenia miała do dyspozycji własne szachy, na których mogła analizować przebieg partii, świeczkę oświetlającą wnętrze oraz różnego rodzaju korbki i uchwyty, dzięki którym mogła panować nad ruchami wykonywanymi przez gumowego Turka (Rys. 1.1.). Dzięki dobrze przemyślanej konstrukcji, szachista siedzący w środku widział nad swoją głową aktualny stan szachownicy znajdującej się na szafie.



Rys. 1.1. Konstrukcja gumowego Turka grającego w szachy (źródło: [38])

## 1.1. Pierwszy elektroniczny mechanizm szachowy

Pierwsze dobrze udokumentowane prace nad prawdziwym automatem szachowym pojawiają się pod koniec XIX wieku, w momencie gdy świat zaczyna być zalewany różnorodnymi urządzeniami elektrycznymi. Hiszpański inżynier, matematyk oraz wynalazca Leonardo Torres y Quevedo (1852-1936), uważany za jednego z twórców podstaw automatyki, od 1890 roku konstruował pierwszy elektroniczny mechanizm wielkości szafy, nazwany przez niego 'El Ajedrecista' (z hiszpańskiego 'szachista'), którego celem było rozgrywanie końcówek szachowych (król+wieża przeciwko królowi). W 1912 roku Torres y Quevedo zakończył budowę pierwszej wersji tej maszyny, która wykonywała ruchy mechaniczną ręką i rozpoznawała ruchy przeciwnika poprzez odpowiednie sensory elektryczne umieszczone na szachownicy. Zgodnie z założeniem, niezależnie od pozycji początkowej, maszyna potrafiła zamatować samotnego króla przy wykorzystaniu króla i wieży. Chociaż sam manewr matowania nie był właściwie optymalny, tzn. liczba ruchów potrzebna do osiągnięcia celu niekoniecznie była minimalna, to jednak sam fakt, że maszyna

potrafiła całkowicie samodzielnie (automatycznie) dobrnąć w grze do mata budziło niewyobrażalne emocje. W dzisiejszych czasach, problem, który rozwiązywała maszyna Torres`a, wydaje się być trywialny, jednak na początku XX wieku było to wielkie wyzwanie. Zapoczątkowanie przez Torres`a elektronicznej realizacji gry w szachy, jak i inne prace (przegląd najciekawszych z nich można znaleźć w [23]), do dzisiaj uważane są za prekursorskie w dziedzinie sztucznej inteligencji i współczesnych obliczeń. Nad kolejnymi udoskonaleniami mechanizmu pracował jeszcze wraz z synem Gonzalesem do 1920 roku, w którym przedstawił drugą wersję tej maszyny ulepszoną m.in. o możliwość poruszania się figur na szachownicy dzięki magnesom umieszczonym pod nią. Obie wciąż działające wersje maszyny znajdują się w muzeum Politechniki Madryckiej (*Colegio de Ingenieros de Caminos, Canales y Puertos de Madrid*).

## 1.2. Inteligencja automatów szachowych – test Turinga

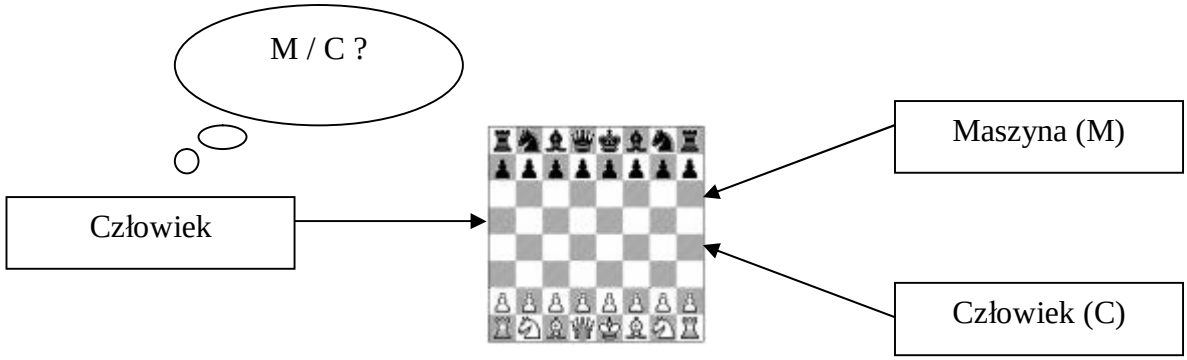
Wraz z rozwojem techniki (głównie za sprawą rozwoju elektroniki) zaczęto się zastanawiać nad „inteligentnymi" możliwościami maszyn. W połowie XX wieku, świetny matematyk, pionier informatyki, Alan Turing w odpowiedzi na pytanie „czy maszyna potrafi myśleć?" zaproponował w pracy [37] swój słynny test, który będzie mógł być zaliczony tylko i wyłącznie przez maszyny najlepiej naśladujące możliwości konwersacyjne człowieka. Turing uważał, że jego test prędzej czy później zostanie zaliczony. Chociaż w dzisiejszych czasach nie wszyscy uważają aby test Turinga był wystarczającym wyznacznikiem inteligencji, to jednak nikomu dotychczas nie udało się go zaliczyć. Okazuje się, iż uczynienie automatu na tyle „inteligentnym", aby mógł przeprowadzić „ludzką" rozmowę nie jest rzeczą łatwą. Co ciekawe, od 1990 roku zaczęto rozgrywać coroczne zawody o nagrodę Loebnera[1], które wyłaniają program najlepiej naśladujący zdolności konwersacyjne człowieka. Celem zawodów jest niezaprzeczalne przejście testu Turinga. Osoba, której program jako pierwszy przejdzie ogólny test Turinga wygra 100 tysięcy dolarów, a nagroda Loebnera zostanie rozwiązana.

Aby nieco uprościć zadanie testowanej maszyny, stosuje się często ograniczone testy Turinga. Jednym z takich testów jest szachowy test Turinga, który opisuje następującą sytuację: osoba grająca w szachy z pewnym przeciwnikiem na podstawie wykonywanych ruchów musi ustalić czy gra z człowiekiem czy też z maszyną (Rys. 1.2.). Zmodyfikowana wersja tego testu zakłada dodatkowo, że osoba nie gra bezpośrednio w szachy, tylko otrzymuje zapis pewnej partii, z której musi określić kim byli przeciwnicy: czy grały dwie osoby, czy człowiek z maszyną, czy może dwie maszyny. Celem tej modyfikacji było wzmocnienie uproszczonego testu poprzez dodatkową trudność – grając w szachy bezpośrednio z przeciwnikiem, którego „tożsamość" należy ustalić, można specjalnie wykonywać pewne sztuczne ruchy, czy też z premedytacją popełniać drobne błędy, aby wiarygodnie odpowiedzieć na pytanie. W tak zmodyfikowanym szachowym teście Turinga, podobnie jak w oryginalnym ogólnym teście Turinga, im „mądrzejsza" będzie maszyna, tym trudniej będzie ustalić „tożsamość" przeciwników.

Pod koniec XX wieku w nieformalnie przeprowadzonym przez Frederica Friedela eksperymencie[2], bazującym na zmodyfikowanym szachowym teście Turinga, ówczesny szachowy Mistrz Świata Garry Kasparov, na podstawie przedstawionych zapisów różnych partii nie był w stanie wiarygodnie określać „tożsamości" szachistów. Będąc bardzo doświadczonym szachistą udzielił kilku błędnych odpowiedzi. Łatwo sobie w związku z tym wyobrazić, iż wielu mniej doświadczonych graczy miałoby z tym poważne problemy. Zmodyfikowany szachowy test Turinga został już w pewnym stopniu zaliczony.

[1] Oficjalna strona nagrody Loebnera: http://www.loebner.net/Prizef/loebner-prize.html.
[2] Wzmiankę o tym zdarzeniu można znaleźć w pracy [Campbell1997].

Natomiast warto zastanowić się na czym tak naprawdę opiera się ta „sztuczna inteligencja" programów szachowych potrafiących współcześnie grać na bardzo wysokim poziomie.




Rys. 1.2. Szachowy test Turinga

## 1.3. Pierwsze programy szachowe

Właściwie zanim zostały wynalezione pierwsze programowalne komputery, Alan Turing zdążył już zaproponować swój pierwszy program szachowy[3], który został zrealizowany za pomocą maszyny stanów. Pomimo wielkiego talentu matematycznego, Turing wcale nie był dobrym szachistą, jednak ośmielił się napisać pewien ciąg instrukcji umożliwiający maszynie grę w szachy i to na całkiem przyzwoitym poziomie. W celu przetestowania swojego „papierowego" wynalazku w 1952 roku w Manchesterze rozegrał partię szachową ze swoim kolegą[4] (Alick Glennie). Ponieważ nie istniała wówczas żadna maszyna, która byłaby w stanie zrealizować zaproponowany przez Turinga program, na każdy ruch potrzebował on osobiście średnio pół godziny, aby ręcznie wykonywać należyte instrukcje. Niestety w ostatecznym rozrachunku maszyna Turinga przegrała z Alickiem, ale najważniejszym było postawienie kolejnego kroku w stronę zautomatyzowania gry w szachy.

Niemalże w tym samym czasie co Turing, kolejny geniusz matematyki, Claude Shannon zastanawiał się nad możliwością uczenia maszyn gry w szachy. W swojej pracy[5] [28] rozpatruje szachy jako grę skończoną, dla której można wygenerować drzewo, którego węzłami są możliwe do uzyskania pozycje figur na szachownicy po wykonaniu odpowiednich posunięć będących jego łukami. Liściom tego drzewa, a więc pozycjom końcowym, można przyporządkować ostateczny wynik gry. Co prawda istnieją takie sytuacje na szachownicy, że w skończonej liczbie ruchów nie da się doprowadzić do końca gry (np. obie strony cyklicznie wykonują ten sam zestaw ruchów), ale dodatkowo można założyć, że jeżeli na przykład w ciągu 50 ruchów od ostatniego bicia, żadna ze stron nie doprowadzi do wygranej, to przyjmuje się wynik nierozstrzygnięty, czyli remis. Oczywiście cały problem ogarnięcia szachów tkwi w niemożności określenia całego drzewa gry – na szczęście, bo straciłyby one swój niepowtarzalny urok. Jeżeli na przykład przyjmiemy, że każda ze stron wykonująca ruch ma do dyspozycji średnio 30 możliwości, to dla przeanalizowania jednego ruchu mamy już 30*30 = 900 możliwości, a przy średniej liczbie 50 ruchów do rozstrzygnięcia gry liczba ta wzrasta do ok. $10^{150}$, co jest już liczbą niewyobrażalną. Dlatego też przeszukiwanie drzewa gry ograniczone zostało do określonej jego głębokości i dla tak ustalonego drzewa Shannon rozróżnił dwie strategie:

[3] Program Turinga o nazwie TurboChamp nie był już później dopracowywany.
[4] Partię papierowej maszyny Turinga z Alickiem Glennie obejrzeć można pod adresem http://www.chessgames.com/perl/chessgame?gid=1356927.
[5] Praca Shannona jest najprawdopodobniej pierwszą opublikowaną pracą na temat komputerów szachowych.

- strategia A – oceniająca wszystkie liście drzewa,
- strategia B – oceniająca liście wybranych gałęzi drzewa stanowiących tzw. pozycje stabilne (uwzględnia głębsze przeszukiwanie drzewa dla forsownych wariantów: wymiany figur, szachy, itp.).

Oceną liści drzewa zajmuje się wówczas odpowiednio zaproponowana funkcja oceniająca będąca najczęściej sumą ważoną wyznaczanych cech gry. Wagi takiej funkcji dostrajane miałyby być ręcznie. Obie strategie, w nieco zmodyfikowanej formie, znajdują swoje odzwierciedlenie we współczesnych programach szachowych, gdzie strategię A utożsamia się z brutalnym przeszukiwaniem (*brute force*) oraz strategię B z przeszukiwaniem selektywnym (*selective*).

Pomimo znacznego ograniczenia ocenianych możliwości, tzn. podcięcia drzewa do pewnej głębokości, nadal przejście węzłów tego drzewa przy wykorzystaniu dowolnej strategii wiązało się z ogromnym procesem obliczeniowym, którego ówczesne maszyny nie były w stanie podjąć w rozsądnym czasie. Wczesne maszyny szachowe potrafiły oceniać około 500 pozycji na sekundę, co dawało około 90 tysięcy pozycji na 3 minuty (np. w turniejach, w których obowiązywała zasada trzech minut na ruch). Okazało się, że przy takiej mocy obliczeniowej maszyna potrafiła przeanalizować co najwyżej półtora ruchu naprzód, co oczywiście nie umożliwiało jej przeprowadzenia gry na wysokim poziomie. Dlatego też możliwość zawładnięcia szachami przez maszyny nadal stała pod wielkim znakiem zapytania.

Na szczęście już w 1958 roku trzech naukowców A. Newell, J. C. Shaw i H. A. Simon z uniwersytetu w Pittsburghu odkryło bardzo ciekawą zależność, która pozwalała odcinać duże gałęzie przeszukiwanego drzewa bez wpływu na wynik końcowy. Zaproponowany w pracy [15] algorytm nazwali cięciami *Alpha-Beta*. Było to czysto matematyczne podejście i działało bez wykorzystywania dodatkowej wiedzy szachowej. Algorytm opierał się w przybliżeniu na następującym działaniu: po przeanalizowaniu pierwszego ruchu i jego ocenieniu maszyna przechodziła do analizy drugiego i jeśli podczas rozwijania poddrzewa dla tego ruchu napotykała na ocenę gorszą względem ruchu pierwszego, to rezygnowała z dalszego jego przeszukiwania. Zastosowanie tego algorytmu w praktyce pozwoliło maszynom zwiększyć głębokość przeszukiwań drzewa gry nawet do 3.5 ruchu w rozsądnym czasie, osiągając dzięki temu całkiem przyzwoity poziom gry. Jednak wykładniczy wzrost możliwości pojawiających się wraz z głębszym przeszukiwaniem drzewa, po raz kolejny umożliwił osiągnięcie tylko pewnej rozsądnej granicy, po przekroczeniu której maszyny nie były w stanie sobie poradzić.

Jako ciekawostkę warto dodać, że w tym samym roku program szachowy *NSS* wygrał w szachy z osobą, która pierwszy raz w życiu w ciągu godziny uczyła się grać w szachy. Oczywiście nie jest to wielkie osiągnięcie, ale jest na tyle ciekawe, że na pewno na stałe wpisało się w karty historii szachów.

## 1.4. Wielki wyścig sztucznego szachisty z człowiekiem

Od początku lat 60. XX wieku trwały intensywne prace nad oprogramowywaniem komputerów szachowych. Już w 1962 roku Alan Kotok wraz z John`em McCarthy`m napisali program szachowy grający regularnie na bardzo dobrym poziomie. Kilka lat później grupa Georgiej`a Adelson`a-Velskiy`ego w Moskwie stworzyła program, który wygrał dziewięciomiesięczny mecz korespondencyjny z programem Kotoka. W drugiej połowie lat 60. po raz pierwszy w historii program szachowy, *MAC HACK VI* Richard`a Greenblatt`a, uczestniczył w wielu turniejach z prawdziwymi szachistami, w których osiągał wyniki średnio-zaawansowanego szachisty. Program ten również jako pierwszy miał zaprogramowaną szachową książkę otwarć. W 1968 roku szachowy mistrz międzynarodowy David Levy założył się o 3000 dolarów z badaczem sztucznej inteligencji John`em

McCarthy`m, że żaden komputer szachowy nie pokona go w przeciągu 10 lat. Jak się później okazało został on wygrany przez David`a Levy`ego.

W latach 70. XX wieku, ze względu na duże zainteresowanie oprogramowywaniem komputerów szachowych, zaczęto organizować międzynarodowe turnieje. W 1970 roku w Nowym Jorku zorganizowano pierwsze *Północnoamerykańskie Mistrzostwa ACM*[6] *Komputerów Szachowych*, w których zwyciężył program *CHESS 3.0* (*CDC 6400*) napisany przez trójkę naukowców Slate, Atkin oraz Gorlen z Uniwersytetu Northwestern. W latach 1971-1973 kolejne wersje programu szachowego *CHESS* (3.5, 3.6, 4.0) zwyciężały kolejne *Północnoamerykańskie Mistrzostwa ACM*, odbywające się kolejno w Chicago, Bostonie oraz w Atlancie. W 1974 roku w Sztokholmie przeprowadzono *I Mistrzostwa Świata Komputerów Szachowych*, w których zwyciężył program *KAISSA*, stworzony przez Donskoy`a oraz Arlazarov`a z Moskwy, tuż przed programem *CHESS 4.0*. Do końca lat siedemdziesiątych prym najlepszych programów szachowych wiódł jednak wciąż udoskonalany program szachowy *CHESS*. W 1977 w Toronto program *CHESS 4.6* wygrał *II Mistrzostwa Świata Komputerów Szachowych*, a następnie w 1978 roku, w wersji 4.7 po raz pierwszy zremisował z szachowym mistrzem międzynarodowym. Od tego momentu optymiści przewidywali, że w ciągu 10 lat komputer zostanie mistrzem świata.

W 1980 roku Edward Fredkin zorganizował Nagrodę Fredkin`a w wysokości 100 tysięcy dolarów dla pierwszego komputera szachowego, który pokona panującego mistrza świata. W latach 1980-1983 prym najlepszego komputera na świecie wiodła maszyna *Belle* (przeznaczona tylko i wyłącznie do gry w szachy) skonstruowana w laboratoriach Belle`a przez naukowca Ken`a Thomnpson`a. Było to rozwiązanie typowo sprzętowe, które potrafiło przeszukiwać około trzydziestokrotnie więcej pozycji szachowych w tym samym czasie co najszybszy ówczesny superkomputer. Znaczące zwiększenie szybkości pozwoliło zwiększyć głębokość przeszukiwanego drzewa gry, a tym samym osiągnąć poziom gry niemalże mistrzowski. Dopiero w 1984 roku pojawiło się nieco lepsze rozwiązanie *Cray X-MP*, które zwyciężyło *IV Mistrzostwa Świata Komputerów Szachowych*, a którego skonstruowanie kosztowało kilka tysięcy razy więcej dolarów niż *Belle*. W połowie lat osiemdziesiątych profesor Hans Berliner, będący korespondencyjnym szachowym mistrzem świata, wraz z jego byłym studentem Carl`em Ebeling`iem, stworzyli sprzętowe rozwiązanie generatora ruchów szachowych (*HiTech*), które jako pierwsze w historii uzyskało ranking ELO[7] powyżej 2500. Niestety w 1986 roku w *V Mistrzostwach Świata Komputerów Szachowych*, *HiTech* przegrywając jedyną partię z *Cray*`em zdobył jednak za nim dopiero drugie miejsce. Zaraz po wynalazku Berliner`a, jego studenci Feng-hsiung Hsu, Murray Campbell i inni pracowali nad zwiększeniem szybkości przeszukiwań we własnej maszynie *ChipTest*, którą w ostatecznej wersji nazwali *Deep Thought*. Hsu oraz Campbell przyczynili się później do stworzenia potężnej maszyny szachowej *Deep Blue*. W 1988 roku w *Otwartych Szachowych Mistrzostwach Stanów Zjednoczonych*, *Deep Thought* podzielił pierwsze miejsce wraz z arcymistrzem Tony`m Miles`em, uzyskując jednocześnie turniejowy ranking 2745. W 1989 roku w Edmonton, *Deep Thought* wygrał *VI Mistrzostwa Świata Komputerów Szachowych*, a następnie wygrał zorganizowany z arcymistrzem Robert`em Byrne`m mecz. W rozegranym w marcu 1989 roku meczu, ówczesny mistrz świata Garry Kasparov pokonuje jednak maszynę *Deep Thought* w dwóch grach, ale w zorganizowanym meczu z mistrzem międzynarodowym David`em Levy`m *Deep Thought* zwycięża, wygrywając 4 partie.

Lata 90. XX wieku to okres największych osiągnięć programów szachowych, które rywalizowały już z najsilniejszymi arcymistrzami szachowymi na równym poziomie.

[6] ACM - *Association for Computing Machinery*.
[7] Nazwa ELO zapożyczona została od nazwiska węgierskiego profesora fizyki, twórcy tego systemu, Árpád Élő.

Z największych wydarzeń warto zwrócić uwagę na zwycięstwa programu *MEPHISTO* w 1990 roku z arcymistrzami Robertem Hubener`em, David`em Bronstein`em oraz z byłym mistrzem świata Anatoly`m Karpov`em, wygranie po raz pierwszy w historii szybkiej partii (po pięć minut na gracza) przez program *Fritz 2* z ówczesnym mistrzem świata Garry`m Kasparov`em, późniejsze wygrane programu *Fritz 3* w szybkich partiach z najsilniejszymi arcymistrzami na świecie: Kasparov`em, Anand`em, Short`em, Gelfand`em i Kramnik`iem, oraz zapierające dech w piersiach dwa długie pojedynki najlepszych rozwiązań IBM na poważnych zasadach turniejowych z panującym ówcześnie mistrzem świata Garry`m Kasparov`em. Do pierwszego spotkania Kasparov`a z *Deep Blue* doszło w lutym 1996 roku w Filadelfii. Pierwszą partię zwyciężył *Deep Blue*, stając się tym samym pierwszym komputerem w historii, który pokonał panującego mistrza świata w długiej (turniejowej) partii. Cały pojedynek ukończył się jednak zwycięstwem Kasparov`a 4:2. W maju 1997 roku w Nowym Jorku doszło do rewanżowego spotkania pomiędzy mistrzem świata Garry`m Kasparov`em a nową wersją *Deep Blue* firmy IBM, w którym po raz pierwszy w historii cały mecz wygrała maszyna[8]. Z tego też względu ustanowiona w 1980 roku nagroda Fredkina (w wysokości 100 tysięcy dolarów) trafiła w końcu do twórców komputera Deep Blue: Feng Hsu, Murray Campbell, Joseph Hoane. Deep Blue był komputerem szachowym skonstruowanym tylko i wyłącznie z myślą o grze w szachy. Składał się on z ponad 200 specjalnych chipów przeznaczonych do wykonywania szybkich obliczeń, z których każdy umożliwiał przeanalizowanie od dwóch do trzech milionów pozycji na sekundę. Wykorzystanie ponad 200 chipów pozwoliło osiągnąć ekstremalną liczbę 200 milionów analizowanych ruchów na sekundę.

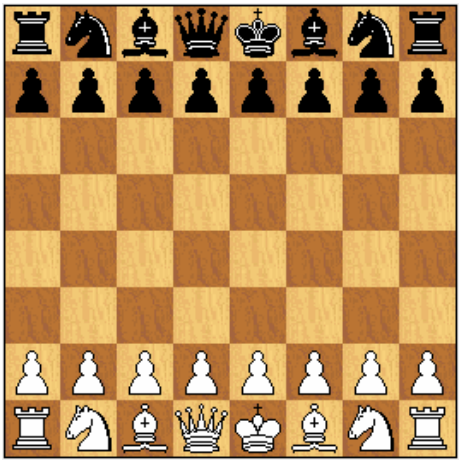
Na koniec warto jeszcze porównać przybliżoną siłę gry komputerów szachowych z aktualnymi najlepszymi szachistami na świecie. Od wielu lat siłę gry szachistów odzwierciedla międzynarodowy ranking ELO systematycznie aktualizowany przez ogólnoświatową federację szachową FIDE[9]. System obliczeń rankingowych wymyślony był początkowo z myślą o królewskiej grze, jednak obecnie wykorzystywany jest również w wielu innych popularnych grach. Najwyższym rankingiem ELO w historii może pochwalić się były mistrz świata Garry Kasparov, który uzyskał wynik 2851 na przełomie lat 1999-2000. Obecnie szachistą z najwyższym rankingiem ELO 2792 jest hindus Anand Viswanathan[10], natomiast najlepszym programem szachowym jest *Rybka* – zwycięzca *XV Mistrzostw Świata Komputerów Szachowych*[11], z rankingiem szacowanym na 2928 przez *Computer Schach und Spiele*[12] w organizowanych turniejach dla programów szachowych rozgrywających dziesięciominutowe partie. Ranking ten należy jednak traktować z lekkim przymrużeniem oka, ponieważ siła gry komputerów szachowych ze względów organizacyjnych nie jest mierzona na podstawie gier z prawdziwymi szachistami. Niemniej jednak można odczuć niewyobrażalną siłę współcześnie grających programów szachowych.

Na zakończenie warto jeszcze raz podkreślić historyczne zwycięstwo programu *Deep Blue* stworzonego przez firmę IBM nad ówczesnym mistrzem świata. Była to maszyna, która zakończyła pewną erę. Wreszcie doprowadzono do rozwiązania, które zwyciężyło z prawdziwym szachowym Mistrzem Świata Garry`m Kasparov`em.

[8] Oficjalna strona rozegranego meczu: http://www.research.ibm.com/deepblue/.
[9] Fédération Internationale des Échecs – http://www.fide.com.
[10] Zgodnie z listą rankingową FIDE na lipiec 2007.
[11] Mistrzostwa rozegrano w Amsterdamie w dniach 11-18.06.2007 – wyniki można obejrzeć na stronie: http://www.grappa.univ-lille3.fr/icga/tournament.php?id=173.
[12] Lista rankingowa programów szachowych z 18.03.2007 dostępna na stronie *Computer Schach und Spiele*: http://www.computerschach.de/index.php?option=com_wrapper&Itemid=238.

## 2. Zasady gry w szachy

W klasycznym wydaniu partia szachów rozgrywana jest na szachownicy przez dwóch przeciwników. *Szachownica* jest kwadratem złożonym z 64 (8x8) pól z występującymi na przemian polami ciemnymi (czarnymi) i jasnymi (białymi), przy czym jest ona ustawiona wobec graczy tak, aby u każdego z nich w lewym dolnym rogu znajdowało się pole ciemne. W chwili początkowej na szachownicy ustawiana jest tzw. *pozycja wyjściowa* (Rys. 2.1.), w której każdy z zawodników ma do dyspozycji 16 figur (Tab. 2.1.) ciemnych (czarnych) albo jasnych (białych) ustawionych w następujący sposób: cztery wieże w czterech rogach, obok nich cztery skoczki, dalej cztery gońce i w środku król obok hetmana stojącego na polu jego koloru. Bezpośrednio przed figurami ustawianych jest po 8 pionków.



Rys. 2.1. Pozycja wyjściowa na szachownicy

| Figura | Biała | Czarna | Polskie oznaczenie | Angielskie oznaczenie |
|---|---|---|---|---|
| Król | | | K | K |
| Hetman | | | H | Q |
| Wieża | | | W | R |
| Goniec | | | G | B |
| Skoczek | | | S | N |
| Pionek | | | (P) | (P) |

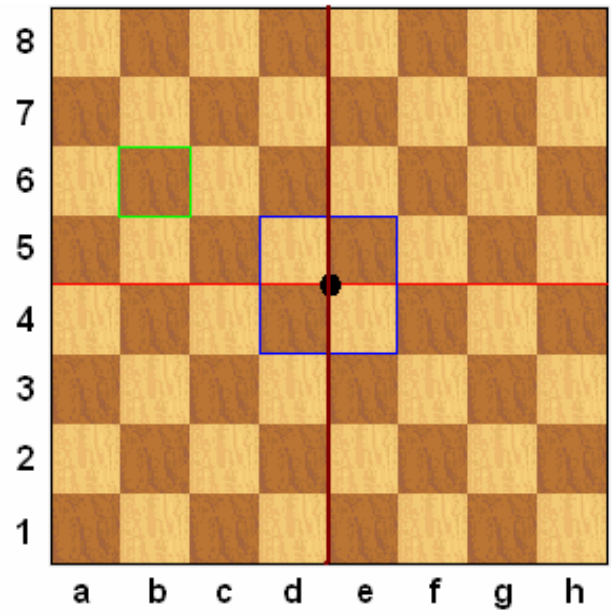
Tab. 2.1. Figury szachowe – wygląd i oznaczenia (oznaczenie pionka jest opcjonalne i zazwyczaj nie jest wykorzystywane w tzw. *algebraicznym zapisie posunięć szachowych*)

Grę rozpoczyna gracz grający figurami białymi poprzez wykonanie pierwszego poprawnego posunięcia. Następne możliwe posunięcia wykonywane są naprzemiennie przez obu graczy. Celem gry jest tzw. *zamatowanie* króla przeciwnika, które polega na takim jego zaszachowaniu, przed którym nie ma on żadnej obrony. *Szach* natomiast jest bezpośrednim atakiem na króla przeciwnika przez dowolną figurę z wyłączeniem króla. Partia szachowa

może zatem zakończyć się wygraną jednej ze stron po postawieniu mata bądź rezygnacji drugiej strony z dalszej gry, ale również remisem:

- po ustawieniu tzw. *pata*, w którym gracz będący na ruchu nie jest szachowany, ale nie może wykonać żadnego ruchu, który nie prowadziłby to bezpośredniego ataku na jego króla,
- po doprowadzeniu do tzw. *pozycji remisowej*, z której niezależnie od wykonywanych ruchów obu graczy, żaden z nich nie ma szans na zwycięstwo – są to w najprostszym przypadku pozycje trzyfigurowe (dwa króle + dodatkowa figura), w której jedna ze stron jest w posiadaniu skoczka lub gońca,
- po trzykrotnym powtórzeniu pozycji (to samo rozmieszczenie figur),
- po wykonaniu 50 ruchów bez rozstrzygnięcia od ostatniego bicia mającego miejsce na szachownicy.

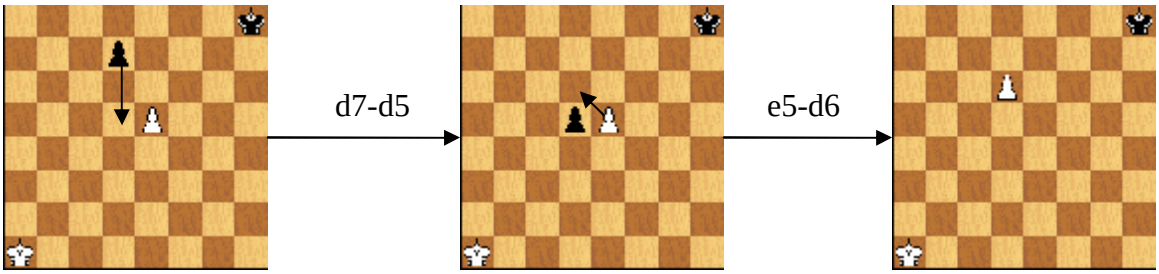
W niniejszej pracy stosowany jest tzw. *algebraiczny zapis posunięć szachowych*, w którym wykorzystywane są oznaczenia figur przedstawione w Tab. 2.1. oraz oznaczenia pól szachownicy – z Rys. 2.2. Każde pole szachownicy znajduje się na przecięciu wiersza (rzędu poziomego) oraz kolumny (rzędu pionowego), które zostały oznaczone odpowiednio cyframi od 1 do 8 (z dołu do góry z perspektywy koloru białego) oraz literami od a do h (z lewej do prawej z perspektywy koloru białego). Jedynie przy omawianiu w dalszej części zaproponowanych cech gry, w celu ułatwienia pewnych obliczeń, oznaczenia te są czasami zmieniane na 0-7 w wierszach i kolumnach w jednakowym porządku. Posunięcie szachowe zapisywane jest jako ciąg następujących informacji: oznaczenie figury, pole, na którym się znajdowała przed posunięciem oraz pole, na którym się znajduje po posunięciu. Na przykład, jeśli wieża wykonuje posunięcie z pola b6 na pole h6, to jest ono zapisane jako Wb6-h6 w zapisie polskim bądź Rb6-h6 w zapisie angielskim. Jeśli natomiast posunięcie wykonuje pionek z pola e4 na pole e5, to wystarczy je zapisać jako e4-e5, co w domyśle jednoznacznie wyznacza posunięcie pionka.




Rys. 2.2. Oznaczenia wierszy oraz kolumn szachownicy wraz z zaznaczonym przykładowym polem b6 (kwadrat zielony), centrum szachownicy (pola d4, e4, d5, e5), poziomą linią środkową (między wierszem 4 i 5) oraz pionową linią środkową (między kolumną d i e)

Każda figura ma ściśle określony sposób poruszania:

- hetman może poruszać się na dowolną odległość w kierunku poziomym, pionowym bądź dowolnym ukośnym, jednakże nie dalej niż ostatnie wolne pole przed figurą tego samego koloru bądź pole, na którym stoi figura przeciwnika,
- wieża może poruszać się na dowolną odległość w kierunku poziomym bądź pionowym, jednakże podobnie jak hetman nie dalej niż ostatnie wolne pole przed figurą tego samego koloru bądź pole, na którym stoi figura przeciwnika,
- goniec może poruszać się na dowolną odległość w dowolnym kierunku ukośnym, jednakże podobnie jak wieża i hetman nie dalej niż ostatnie wolne pole przed figurą tego samego koloru bądź pole, na którym stoi figura przeciwnika,
- skoczek porusza się o dokładnie trzy pola – dwa w dowolnym kierunku poziomym bądź pionowym i jeden w dowolny bok i nie ma przy tym znaczenia czy pola, ponad którymi skoczek przeskakuje są wolne czy też zajęte – ważne jest jedynie, aby docelowe pole nie było zajęte przez figurę tego samego koloru,
- król porusza się tylko i wyłącznie o jedno pole w dowolnym kierunku poziomym, pionowym bądź ukośnym – tzn. na dowolne sąsiednie pole. Jedynym wyjątkiem jest tzw. *roszada*, którą można wykonać co najwyżej raz w przeciągu całej partii, a polega na łącznym przemieszczeniu króla o dwa pola w lewo bądź w prawo wzdłuż wiersza wyjściowego w stronę wieży, która następnie przeskakuje króla i staje obok niego. Roszada może być wykonana pod warunkiem, że pomiędzy królem i wieżą nie stoją żadne figury, król i wieża nie wykonywali dotychczas żadnego ruchu oraz żadne pole przez które przechodzi król nie jest atakowane przez figury przeciwnika,
- pionek porusza się zasadniczo o jedno pole do przodu wzdłuż swojej kolumny w stronę obozu przeciwnika. Jedynie z pozycji wyjściowej pionek może poruszyć się o dwa pola naprzód o ile na drodze przemieszczenia nie stoi żadna figura. Pionek może zbić figurę przeciwnika poruszając się jedynie o jedno pole na ukos naprzód. Jedynym wyjątkiem od tej reguły jest tzw. *bicie w przelocie*, w którym uczestniczą dwa pionki przeciwnego koloru – pionek może zbić w przelocie pionka przeciwnika, tylko i wyłącznie bezpośrednio po jego poruszeniu o dwa pola naprzód – wówczas bicie pionka przebiega tak jakby poruszył się on tylko o jedno pole (przykład na Rys. 2.3.) Ostatnim szczególnym posunięciem pionka jest jego doprowadzenie do ostatniego rzędu, po którym następuje jego przemiana na dowolną figurę: skoczka, gońca, wieżę lub hetmana – pola, na których pion zostaje przemieniony w figurę nazywane są *polami przemiany*.




Rys. 2.3. Przykład bicia w przelocie

Początkującym szachistom wprowadza się zwykle podstawowe wiadomości dotyczące wartościowania elementarnych cech gry występujących na szachownicy. Najważniejszą z nich jest uwrażliwienie szachistów na tzw. *wartość materialną figur* pozwalającą porównać

siły bojowe jakimi dysponują obaj gracze. Wartość ta ma za zadanie oddać siłę figury zależną od jej zasięgu (jak daleko może się przesunąć w jednym posunięciu), zdolności do jednoczesnego atakowania wielu pól, itp. Najsilniejszą figurą w szachach jest hetman, który łączy w sobie możliwości poruszania się wieży i gońca (aż w ośmiu kierunkach). Praktyka wielu szachistów wykazała, że wartość hetmana powinna w przybliżeniu odpowiadać wartości dwóch wież bądź trzech lekkich figur. Przyjęło się uważać, iż następną w kolejności najsilniejszą figurą jest wieża operująca na całej szachownicy w przeciwieństwie np. do gońca, który zdany jest do prowadzenia działalności na polach jednego koloru (pół szachownicy), a także szybko przerzucająca się z jednego krańca szachownicy na drugi w przeciwieństwie np. do skoczka, który ma ściśle ograniczony zakres posunięcia. Przyjmuje się, iż następne w kolejności są jednakowo silne skoczek oraz goniec, a na samym końcu najsłabszy jest pionek. Aby ułatwić początkującym szachistom kalkulowanie swoich sił wprowadza się przybliżone wartości materialne figur, przy założeniu, że dla pionka stanowi ona 1 punkt (Tab. 2.2.). Ze względu na wartość materialną figur dokonuje się często podziału na tzw. *figury ciężkie* (hetman, wieża) oraz *figury lekkie* (skoczek, goniec).

| Typ figury | Nazwa figury | Wartość materialna |
|---|---|---|
| | Pionek | 1 |
| lekka | Skoczek | 3 |
| lekka | Goniec | 3 |
| ciężka | Wieża | 4.5-5 |
| ciężka | Hetman | 9-10 |

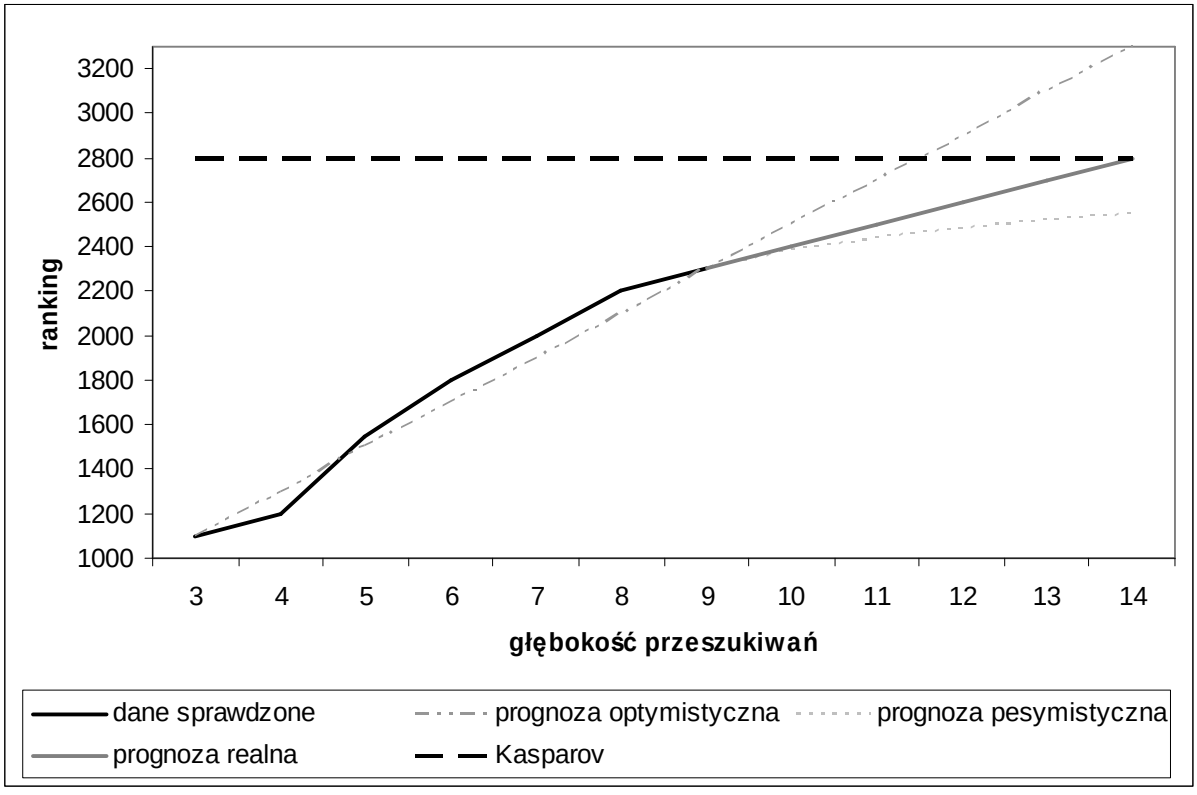
Tab. 2.2. Wartości materialne figur

Oczywiście najcenniejszą figurą jest król, którego nie da się wymienić z żadną inną figurą, gdyż to o niego właśnie toczy się walka. Dlatego też nie mówi się wprost o wartości materialnej króla.

Drugą ważną sprawą, na którą uczula się początkujących szachistów jest wartość pozycyjna figur, która ściśle zależy od miejsca na szachownicy, w którym znajduje się dana figura i wynikających z niej zdolności do realizowania pewnych zadań, np. zamatowania przeciwnika, obrony własnego króla, zbicia figury przeciwnika, obrony własnej figury, wzmocnienia pionka w drodze do pola przemiany, swobody ruchów, ograniczenia ruchów figur przeciwnika, itp. Stąd też ta sama figura może mieć różną wartość pozycyjną w różnych sytuacjach na szachownicy i nie ma nawet mowy o jej przybliżeniu tak jak to miało miejsce w przypadku wartości materialnej.

# 3. Przegląd inteligentnych metod wykorzystywanych w programach szachowych

Na przestrzeni ostatnich kilkudziesięciu lat można zaobserwować, iż główną tendencją gwarantującą wysoki poziom gry komputerów szachowych była coraz większa szybkość przeszukiwania drzewa gry od ustalonej na szachownicy pozycji. Za każdym razem osiągano to głównie dzięki rozwiązaniom sprzętowym pojawiającym się wraz z rozwojem elektroniki. Oczywiście niemniejszym zainteresowaniem cieszyły się rozwiązania programowe poszukujące coraz to lepszych możliwości algorytmicznych z wykorzystaniem programowania niskiego poziomu, np. przy wykorzystaniu assemblera. Ciekawe wyniki badań przedstawił w latach 80-tych Ken Thompson (Rys. 3.1.) – twórca komputera szachowego „Belle". W przeprowadzanych eksperymentach badał zależność siły gry komputera szachowego od głębokości przeszukiwanego przez niego drzewa. Wyniki jego badań wykazały, że średnio co pół ruchu głębszego przeszukiwania drzewa, siła gry komputera szachowego wzrasta o 200 punktów rankingowych. Przy głębokości przeszukiwania 4.5 ruchów, siłę gry komputera szachowego określił na 2328 punktów rankingowych. Po wykonanych eksperymentach określił hipotetyczny dalszy wzrost siły gry.




Rys. 3.1. Wyniki badań Kena Thompsona nad zależnością siły gry komputera od głębokości przeszukiwania drzewa gry (rysunek sporządzony na podstawie [6])

Warto w związku z tym zadać sobie pytanie czy maszyna świetnie grająca w szachy faktycznie jest inteligenta? Zestawiając obecne możliwości obliczeniowe komputera i człowieka oczywiste jest, że komputer miażdżąco przewyższa umiejętności zwykłego człowieka. W tym tkwi głównie moc komputera grającego w szachy, który w każdej pozycji szachowej jest w stanie przeanalizować nawet kilkanaście ruchów naprzód. Dla porównania najlepsi szachiści prawdopodobnie są w stanie przeanalizować kilkanaście ruchów, ale tylko w kilku interesujących ich wariantach, często forsownie przebiegających.

Człowiek nie posiada takich możliwości, aby w każdej sytuacji móc przejrzeć każdy możliwy wariant do głębokości kilkunastu ruchów. Dlatego jedyną bronią człowieka staje się niezastąpiona intuicja podparta teorią, której brakuje komputerowi.

Kolejną przewagą komputera nad człowiekiem jest jego zdolność do gromadzenia ustalonej wiedzy teoretycznej. Pamięć masowa komputerów wciąż rośnie i w związku z tym nie ma problemów z jej oszczędnym użyciem. Ważnym czynnikiem wykorzystującym możliwości pamiętania informacji jest oczywiście szybkość dostępu do przechowywanych informacji, jak również szybkość wyszukania właściwiej informacji z ogromnego ich zbioru. Pamięć masowa komputerów służy na przykład do przechowywania bazy wiedzy gry początkowej – wszystkich otwarć gry, które ewoluowały na przestrzeni wielu lat i które doczekały się również swojego miejsca w potężnych encyklopediach debiutów szachowych. Baza takiej wiedzy dostarcza komputerowi informacji o najlepszych ruchach w początkowej fazie gry, statystyki wykonywanych ruchów pod względem wygranych, remisów, przegranych, a także siły graczy wykonujących dany ruch. Przy wykorzystaniu tej wiedzy pierwsze obliczenia komputerowe związane z samą grą pojawić się mogą nawet po dwudziestym ruchu. Pamięć masowa wykorzystywana jest również do przechowywania bazy końcówek. Ciekawe eksperymenty poczynił w latach 80-tych Ken Thompson, który stworzył bazę wszystkich możliwych pozycji końcowych zawierających cztery oraz pięć figur szachowych, dla których wygenerował jedynie słuszny sposób gry. Większość współczesnych programów szachowych zaopatrzona jest w tego typu bazę. Nietrudno sobie więc wyobrazić, iż komputer dochodząc w grze do cztero- bądź pięcio-figurowej końcówki, rozegra ją perfekcyjnie nie dając żadnych szans człowiekowi. Końcówki wygrane zawsze wygra, końcówki remisowe co najmniej zremisuje, a w końcówkach przegranych zawsze będzie się bronić jedynym optymalnym sposobem. Obecnie znane są już wyniki badań nad wszystkimi sześciofigurowymi końcówkami. Na stronie komputera szachowego Shredder umieszczona jest internetowa baza danych wszystkich końcówek do sześciu figur włącznie[13] – dostępna tylko w wersji interaktywnej dla szachistów chcących przeanalizować określone pozycje. Rozpracowanie tego typu kilkufigurowych końcówek przyczyniło się tym samym, nie tylko do zwiększenia siły gry komputerów szachowych w końcówce, ale również do zwiększenia wiedzy szachistów o nich, którzy w wielu sytuacjach nie zdawali sobie sprawy z możliwości wygrania, wydawałoby się remisowych pozycji, bądź zremisowania, wydawałoby się przegranych pozycji. Świetnym przykładem wykorzystania stworzonych przez Thompsona baz końcówek są prace Johna Nunna, zdolnego matematyka-szachisty, dzięki którym powstały dwie książki dotyczące końcówek wieżowych [17] i pionowych [18].

W pierwszej części pracy pokazano jedynie historię związaną z automatyzacją gry w szachy, nie zwracając jednak większej uwagi na stosowane w praktyce techniki sztucznej inteligencji. Dlatego też w niniejszych podrozdziałach zostaną bardzo ogólnie zarysowane podejścia z zastosowaniem tych metod. Świetną pracą podsumowującą wyniki wykorzystania sztucznej inteligencji w szachach na świecie w latach 80.-90. jest praca Johannesa Fürnkranza [8], na której oparte będą przedstawiane rozważania. Obserwując przeprowadzane badania można wyróżnić cztery główne nurty, w których stosowane są metody sztucznej inteligencji mające zastosowanie w rozgrywkach szachowych:

- indukcja klasyfikatorów końcówek (*induction of endgame classifiers*),
- uczenie się na podstawie wyjaśnień (*explanation-based learning*),
- wnioskowanie na podstawie przypadków (*case-based learning*),
- algorytmy uczące dostrajające funkcję oceniającą (*evaluation function learning*).

[13] http://www.shredderchess.com/online-chess/online-databases/endgame-database.html

## 3.1. Indukcja klasyfikatorów końcówek

Wydobywanie koncepcji szachowych z końcówek, już od kilkudziesięciu lat, budziło niezwykłe zainteresowanie zarówno w zakresie zwiększania wiedzy szachowej szachistów jak również w zakresie poszukiwania nowych sposobów maszynowego uczenia. Kilka rodzajów końcówek szachowych doczekało się nawet swojego miejsca w repozytorium danych poświęconym testowaniu algorytmów maszynowego uczenia[14].

Początkowo zajmowano się pewnymi rodzajami końcówek, których pozycje były już sklasyfikowane jako wygrane bądź nie-wygrane. Stosowane algorytmy indukcyjnego uczenia wykorzystywały te pozycje do wydobycia reguł klasyfikacji, które byłyby w stanie w 100% poprawnie pokrywać oryginalną klasyfikację. Głównym celem tego procesu miało być uzyskanie korzyści zarówno pod względem mniejszego wykorzystania zasobów komputera, jak również pod względem lepszego zrozumienia samej gry przez człowieka.

W procesie indukcji reguł stosuje się najczęściej podział zbioru wszystkich możliwych przykładów na dwa rozłączne zbiory: uczący oraz testowy. Na podstawie zbioru uczącego indukowane są najpierw reguły klasyfikacji. Ich jakość testowana jest następnie na ustalonym zbiorze testowym. Testowanie polega zwykle na porównywaniu, dla każdego przypadku ze zbioru testowego, wyników klasyfikacji wnioskowanych na podstawie reguł z oryginalną klasyfikacją. Średnia dokładność testowanego zbioru reguł wyznacza z kolei ich jakość. Głównym problemem przy indukcji klasyfikatorów końcówek szachowych jest jednak odpowiednie przygotowanie reprezentacji przykładów (pozycji szachowych). Typowe algorytmy indukcyjnego uczenia wymagają, aby przykłady definiowane były przy pomocy sekwencji par typu atrybut-wartość, przy czym zbiór atrybutów jest stały i każdy z atrybutów charakteryzuje się dyskretną i skończoną dziedziną. Przy powyższych założeniach wyuczone reguły mają zwykle postać klauzul Hornowskich (bądź dowolnych implikacji), których działanie sprowadza się do sprawdzania wartości przyjmowanych przez predefiniowane atrybuty przykładu (przesłanki), na podstawie których odpowiednio go klasyfikuje (konkluzja). Reprezentacja pozycji jedynie na podstawie oczywistych atrybutów typu rozmieszczenie figur czy określenie strony, która jest na ruchu, na pewno jest niewystarczająca do użytecznych uogólnień. Zazwyczaj przykłady te powinny być dodatkowo zaopatrzone w atrybuty, które potencjalnie wydają się być użyteczne, np. odległości pomiędzy figurami, odległości od szczególnych punktów szachownicy (np. centrum, krawędź) czy wykrywanie wzorców szachowych będących szczególnym rozstawieniem figur (np. opozycja króli, figury w jednej linii bądź kolumnie). Określenie właściwego zbioru atrybutów pozwalających tworzyć dobrze uogólnione zestawy reguł wcale nie jest rzeczą prostą.

Jedne z pierwszych eksperymentów z tego zakresu opisywane były w 1983 roku przez Quinlana [19], który próbował wydobyć reguły klasyfikujące końcówki typu KRKN (*King-Rook-King-Knight,* tzn. król z wieżą przeciwko królowi ze skoczkiem). Quinlan wykorzystał swój algorytm tworzenia drzew decyzyjnych ID3 dla tych końcówek, które były klasyfikowane ze względu na przegrane w dwóch oraz trzech pół-ruchach. Ze zbioru składającego się z mniej niż 10% wszystkich pozycji typu KRKN, algorytm ID3 był w stanie zbudować drzewo, które podczas testów popełniło tylko 2 błędy na zbiorze testowym składającym się z 10000 losowo wybranych pozycji. Quinlan zauważył jednocześnie, iż sukces ten mógł być osiągnięty tylko i wyłącznie dzięki właściwemu doborowi dodatkowych atrybutów definiujących pozycje, który zajął autorowi trzy tygodnie w przypadku pozycji przegranych w dwóch pół-ruchach oraz trzy miesiące – w trzech pół-ruchach. Quinlan zamierzał zająć się również pozycjami KRKN przegranymi w czterech pół-ruchach, ale ostateczne wyniki dotyczące tych badań nie zostały przez niego

[14] *UCI Machine Learning Repository*: http://www.ics.uci.edu/~mlearn/databases/chess/

opublikowane. Wiadomo jedynie, że próbował dodatkowo eksperymentować metodami automatycznego odkrywania nowych potencjalnie użytecznych atrybutów. Chociaż powstające zestawy reguł charakteryzowały się 100% skutecznością oraz dużą szybkością odpowiedzi (szybciej niż dokładne przeszukiwanie drzewa gry) i z powodzeniem mogły być stosowane w komputerowej klasyfikacji pozycji, to jednak okazywały się całkowicie niezrozumiałe dla prawdziwego szachisty. Dlatego też dalsze prace w tym zakresie skupiały się nad możliwie największym uproszczeniem wydobywanych reguł, aby z wyników tych prac w sposób praktyczny mogli korzystać również szachiści. Proponowane były między innymi sposoby dzielenia analizowanego problemu na pod-problemy, dla których odrębnie wydobywane były następnie reguły klasyfikacji. Ideę indukcji klasyfikatorów wykorzystano również w późniejszych latach przy próbie wyuczenia definicji (klasyfikacji) ogólnych wzorców szachowych takich jak groźby, widełki, baterie, związania, szachy, itp. Jeszcze na początku lat 90. zaprezentowano sposoby wykorzystania indukcyjnego programowania logicznego do indukcji klasyfikatorów dla wzorców szachowych w pracy Moralesa [14] oraz dla końcówek typu KRK (*King-Rook-King*) składających się z króla i wieży przeciwko królowi przeciwnika prowadzących do wygranej w określonej liczbie pół-ruchów przy optymalnej grze obu stron w pracy doktorskiej Baina [1].

## 3.2. Uczenie się na podstawie wyjaśnień

Uczenie się na podstawie wyjaśnień, tzw. *EBL* (*explanation-based learning*), stworzono z myślą o metodach pozwalających uczyć się w problemach ze złożoną wiedzą dziedzinową. Główną ideą *EBL* jest konstruowanie wyjaśnienia danej obserwacji typu wejście/wyjście przy wykorzystaniu wiedzy apriorycznej, aby następnie mogło być one wykorzystane do uogólnienia definicji całej klasy podobnych przypadków. Technika *EBL* jest niejako rozwinięciem idei funkcji zapamiętujących napotkane rozwiązania, których głównym celem jest przyspieszenie pracy programów. Działanie tych funkcji opiera się na komunikacji z bazą, w której składowane są napotykane rozwiązania. Za każdym razem kiedy funkcja jest wywoływana w celu obliczenia rozwiązania, sprawdza ona najpierw czy już wcześniej wyniki tych obliczeń nie zostały zapisane w bazie – jeśli tak, to zwracany jest wynik z bazy, a jeśli nie, to wykonywany jest proces obliczeniowy, którego zwracane wyniki zostają złożone w bazie. Technika *EBL* bazuje na tej samej idei z tą różnicą, że nie są zapamiętywane wszystkie napotykane przypadki, tylko tworzone są ogólne reguły wyjaśniające (przy wykorzystaniu dostępnej wiedzy) ich klasyfikację, które pokrywają całą klasę podobnych przypadków. Głównym założeniem w tym podejściu jest fakt, że wiedza dziedzinowa może być wykorzystywana do analizowania poszczególnych przypadków i wyjaśniania, w jaki sposób spełniają one swoją klasyfikację. Na podstawie kolejnych przypadków tworzone są nowe reguły bądź uogólniane są istniejące, tak aby wszystkie dostępne reguły odpowiednio klasyfikowały wszystkie napotkane przykłady.

Szachy są problemem o dobrze zdefiniowanej i złożonej wiedzy dziedzinowej (znane są zasady gry). Dlatego też zastosowanie jeszcze pod koniec lat 70. metody *EBL* do szachów wydawało się już być całkiem uzasadnione. Jednakże niewiele osób starało się wykorzystać tę metodę w praktyce, ponieważ podejście *EBL* mogło być stosowane jedynie do wyuczenia prostych kombinacji wykorzystujących wyuczone wzorce szachowe. Najwcześniejsze próby wykorzystania *EBL* pojawiły się w 1976 roku w badaniach Jacquesa Pitrata [20] oraz [21], w ramach których utworzony przez niego program był w stanie nauczyć się definicji najprostszych wzorców szachowych, tj. związanie czy widełki. Co prawda program Pitrata odkrył również wiele innych użytecznych wzorców taktycznych, ale okazało się jednak, że rozmiar drzewa potrzebnych do rozpoznawania tych wzorców znacząco się

rozrastał. Ostatecznie Pitrat doszedł do wniosku, że moduł uczący programu działał bardzo dobrze, ale potrzebna byłaby bardziej wyszukana metoda radząca sobie z utrzymywaniem wielu wzorców odkrytych w trakcie nauki. W 1984 roku Minton w swoich badaniach [13] wykorzystał również *EBL* do uczenia się wzorców szachowych, ale zastosował technikę wstecznej analizy ruchów w momencie, gdy jeden z przeciwników osiągał swój cel, np. ustawił szacha bądź mata, zdobył przewagę materialną, itp. Na tej podstawie formułowane były pewne wzorce, które pozwalały rozpoznawać niebezpieczeństwo przed wykonaniem przez przeciwnika wygrywającego wariantu gry. Ostatecznie program Mintona potrafił wyuczyć się jedynie wzorców wykrywających forsowną sekwencję ruchów matujących. Wyniki tych prac wykorzystywał później Puget [22], który starał się uogólnić podejście Mintona poprzez założenie, że każdy wykonywany przez przeciwnika ruch jest w pewnym sensie optymalny, tzn. doprowadza go prędzej czy później do osiągnięcia celu. Wyniki prac Pugeta pozwoliły wydobyć wzorce, które były w stanie sugerować optymalne ruchy, nie gwarantujące jednak swojej skuteczności w każdej możliwej na szachownicy sytuacji. Jeszcze pod koniec lat 80. oraz na początku lat 90. próbowano rozszerzać proponowane wcześniej podejścia. W 1989 roku Tadepalli [32] zaproponował zastosowanie tzw. *Lazy Explanation-Based Learning*, w którym podczas uczenia nie uwzględniono ewentualnych trudności w realizacji celu. Program Tadepalliego podczas gry wykorzystywał pewien moduł planujący do wyznaczania ruchu na podstawie wyuczonych wcześniej różnych planów gry, odpowiednich dla bieżącej pozycji szachowej. Poszczególne plany dotyczyły pośredniej (ruchy przygotowawcze) albo bezpośredniej realizacji celu. Jeżeli w trakcie gry realizacja planów była odpierana przez przeciwnika, to program odpowiednio modyfikował swoje definicje uwzględniając zaistniałe przeszkody. Jeszcze w tym samym roku Flann [4] wykorzystał *EBL* do akwizycji reguł rozpoznających pewne sytuacje powstające na szachownicy, np. mój-król-w-szachu, dla których kojarzone były odpowiednie cele, np. usuń-mojego-króla-z-szachu. Program poszukiwał odpowiednich możliwości, które w zależności od sytuacji niszczyły, utrzymywały bądź osiągały cele, uogólniając na tej podstawie kolejne reguły.

## 3.3. Wnioskowanie na podstawie przypadków

Wnioskowanie na podstawie przypadków, tzw. *CBR* (*Case-Based Reasoning*) zostało wstępnie zaproponowane w 1989 przez Riesbecka i Schanka w pracy [25] jako kognitywny model wnioskowania. Główną ideą tego podejścia jest założenie, że do rozwiązania nowego problemu wykorzystuje się wiedzę zdobytą przy podobnych problemach, które wcześniej zostały już w jakiś sposób rozwiązane. Działanie metody *CBR* wygląda następująco: za każdym razem kiedy napotykany jest nowy problem do rozwiązania, z pamięci odzyskiwane są wszystkie doświadczenia (sposoby rozwiązań) podobnych problemów, aby następnie mogły być one w całości bądź w części wykorzystane w kontekście nowego problemu, a wybrany sposób rozwiązania zapamiętany w pamięci jako nowe doświadczenie. Zasadniczym problemem w przypadku zastosowania *CBR* w praktyce jest zaproponowanie odpowiedniej funkcji wyznaczającej podobieństwo problemów.

Najwcześniejsze próby zastosowania *CBR* w szachach znaleźć można w pracy [29] Spohrera w 1985 roku. Jego program *MAPLE* (*Mistakes As Plan Learning Experiences*) uczył się na podstawie doświadczeń prostych planów gry w szachy. Plany te były tworzone bądź modyfikowane za każdym razem, gdy podczas gry pojawiało się znaczne pogorszenie pozycji (np. strata materialna). Wówczas w pewien określony sposób odtwarzana była sytuacja na szachownicy w celu wyznaczenia przyczyn katastrofy i w odpowiedniej formie zaistniałe zjawisko było zapamiętywane. Ciekawym rozwiązaniem okazały się zaproponowane w 1990 roku przez Scherzera (w pracy [26]) techniki wykorzystujące tzw. tablice transpozycyjne, które pozwoliły programowi szachowemu *BeBe* unikać powtarzania

popełnianych błędów. *BeBe* przechowywał w tablicach transpozycyjnych wszystkie pozycje, które napotykał podczas gry i analizy drzewa gry zapamiętując wykonany ruch, głębokość przeszukiwań oraz ocenę pozycji. Za każdym razem kiedy podczas kolejnych gier znajdował się blisko przechowywanych pozycji, aktualizował odpowiednie dane wynikające z głębszego przeszukiwania drzewa gry dla najbardziej obiecujących wariantów. Eksperymentalnie dowiedziono, iż uczenie programu *BeBe* przy wykorzystaniu tej techniki zwiększało znacząco jego skuteczność w grze podczas 100-200 gier granych z tym samym przeciwnikiem. Podobne badania podjął w 1993 roku Krulwich w swojej pracy doktorskiej [10]. Stworzony przez niego program *CASTL*E uczył się za każdym razem, gdy napotykał na tzw. brak wyjaśnienia (*explanation failure*), np. w sytuacji gdy odpowiednie komponenty detekcji gróźb i obron nie wykryły groźby bądź nie zapobiegły groźbie. Oba wspomniane programy *MAPLE* i *CASTLE* są niejako rozwinięciem wcześniej wspomnianej metody *EBL*, na której nadal bazują. W 1995 roku Kerner w pracy [9] stworzył edukacyjny program szachowy, który potrafił analizować pod kątem strategicznym różne pozycje szachowe w pracy z użytkownikiem, sugerując jednocześnie możliwe do realizacji plany. Podstawowa wiedza systemu składała się z wybranych przez eksperta szachowego strategicznych pojęć szachowych.

## 3.4. Dostrajanie funkcji oceniającej

Optymalizacja funkcji oceniającej stan gry, stała się jednym ze sposobów zwiększania efektywności gry programu szachowego przy wykorzystaniu maszynowego uczenia. Stąd też wzięło się zainteresowanie tym podejściem przez autora niniejszej pracy. Zostanie ona w tej części nieco dokładniej omówiona. Dostrajanie takiej funkcji może przebiegać na wiele różnych sposobów: przy wykorzystaniu rozegranych partii arcymistrzów, na podstawie gry z przeciwnikiem (nauczycielem) bądź na podstawie tzw. samograja (grze z samym sobą). Konstrukcja takiej funkcji może odbywać się również na wiele sposobów. Dwoma najważniejszymi jej składnikami jest odpowiedni dobór cech gry mierzących siłę bądź słabość stanu gry oraz sposób powiązania tych cech, tak aby funkcja oceniająca w wyniku przekształcenia zwracała jedną wartość określającą ogólną ocenę stanu gry. W najprostszej postaci funkcja oceniająca jest ważoną kombinacją liniową cech gry, dla której wartości wag dostrajane są w procesie uczenia. Niezależnie jednak od postaci funkcji oceniającej modyfikacji może podlegać jeszcze wiele innych parametrów. Wykorzystanie metod maszynowego uczenia w celu dostrojenia kilkudziesięciu a może nawet kilkuset parametrów zdaje się być bardzo dobrym rozwiązaniem, tym bardziej, że ręczne ich dostrojenie wydaje się być praktycznie niemożliwe. Dlatego też pożądane są metody optymalizujące przeglądanie wielowymiarowej, niemalże nieskończonej przestrzeni wszystkich możliwych kombinacji. W ramach optymalizacji funkcji oceniających stosowane są i były różne metody, zarówno statystyczne, jak i z zakresu sztucznej inteligencji, z których najczęściej stosowane są sieci neuronowe i algorytmy genetyczne.

### 3.4.1. Algorytmy genetyczne

Algorytmy genetyczne były wykorzystane w automatyzacji gry w szachy na przykład w pracy [36], a także w pracy magisterskiej [12]. Algorytm genetyczny jest probabilistyczną metodą poszukiwania najlepszych rozwiązań. Nie jest to jednak zwykłe błądzenie losowe, lecz w pewien sposób ukierunkowane poszukiwanie coraz lepszych rozwiązań. Idea algorytmu genetycznego powstała w wyniku „podglądania przyrody". Poprzez analogię do natury (doboru naturalnego) wymyślono heurystykę bazującą na ewolucji populacji.

Populacja składa się z określonej liczby osobników. Natomiast każdy osobnik reprezentuje pewien punkt w przestrzeni wszystkich możliwych rozwiązań określonego zadania. Każdy osobnik posiada również swój własny kod genetyczny, który w klasycznym

podejściu stosowania algorytmów genetycznych reprezentowany jest jako ciąg zer i jedynek utożsamianych z ciągiem genów. W każdej populacji można wyróżnić osobniki mniej lub bardziej przystosowane, tzn. reprezentujące rozwiązania dalekie lub bliskie pewnemu rozwiązaniu optymalnemu. Osobniki najlepiej przystosowane, które potencjalnie posiadają najbardziej wartościowy materiał genetyczny, stają się podstawą utworzenia kolejnej populacji poprzez ich reprodukcję. Czynnik ten sprawia, iż działanie algorytmu genetycznego pomimo swej losowości związanej z doborem osobników do reprodukcji, a także z procesem samej reprodukcji jest jak najbardziej ukierunkowane na poszukiwanie optymalnego rozwiązania poprzez promowanie najlepiej przystosowanych osobników, które potencjalnie będą mogły wytworzyć grupę jeszcze lepiej przystosowanych osobników. W algorytmach genetycznych wyróżnia się wiele technik selekcji osobników (do reprodukcji), z których każda opiera się na promowaniu osobników lepiej przystosowanych. Osobniki dobrane zgodnie z odpowiednią techniką selekcji poddawane są następnie reprodukcji przy wykorzystaniu tzw. operatorów genetycznych, takich jak krzyżowanie polegające na wymianie materiału genetycznego pomiędzy dwoma osobnikami czy losowa mutacja określonego genu. Liczba operatorów genetycznych nie jest ograniczona – ważne jest tylko to, aby w wyniku ich zastosowania powstawały „poprawne" osobniki. Pod pojęciem „poprawnego" osobnika rozumiemy tutaj poprawny schemat materiału genetycznego oraz opcjonalnie poprawną postać osobnika spełniającą dodatkowe ograniczenia nałożone na rozwiązywane zadanie. Początkowe pokolenie składa się zazwyczaj z losowych osobników, jednak dopuszcza się możliwość ręcznego ustalenia ich materiałów genetycznych. Kolejne pokolenia powstają w wyniku omówionej wcześniej selekcji oraz reprodukcji. Algorytm genetyczny kończy swoje działanie w przypadku, gdy zostanie znalezione satysfakcjonujące rozwiązanie bądź liczba pokoleń/populacji osiągnie maksymalną dopuszczalną wartość.

Przy stosowaniu algorytmu genetycznego należy jednak pamiętać, iż znalezione przez niego rozwiązania niekoniecznie są najlepsze w skali całego problemu, lecz są jedynie najlepsze w skali rozwiązań do tej pory odkrytych. Algorytm genetyczny zapewnia odnalezienie ekstremum lokalnego rozwiązania, ale nie zapewnia odnalezienia ekstremum globalnego. Stosowanie tego algorytmu jest wskazane w problemach o dużej złożoności, w której przestrzeń rozwiązań jest ogromna, a znalezienie najlepszego rozwiązania w rozsądnym czasie nie jest zapewniane przez żaden inny deterministyczny algorytm. Największym problemem staje się jednak określenie funkcji przystosowania dla osobników populacji. Jak wiadomo funkcja ta ma w jednoznaczny sposób określić lepszość jednego z dwóch rozwiązań.

Dla problemu gry w szachy, w którym przykładowo osobniki populacji definiują różne strategie gry w szachy, określenie funkcji przystosowania nie jest łatwe. Trudność w jej określeniu bierze się ze względnego pojęcia „lepszy", którego w analizowanym zagadnieniu nie da się jednoznacznie sprecyzować. Patrząc przez analogię do rzeczywistości można by zadać sobie pytanie, który z dowolnie wybranych dwóch szachistów jest lepszy? Oczywiście, aby to stwierdzić należałoby skonfrontować ich ze sobą. Niestety jeden pojedynek nie może być podstawą oceny wyższości jednego szachisty nad drugim. Wypadałoby rozegrać między nimi mecz złożony z większej liczby pojedynków i dopiero wówczas wysnuć odpowiedni osąd. Powstaje tutaj kolejny problem – czy możemy jednoznacznie stwierdzić, że jeżeli szachista $s_1$ jest lepszy od szachisty $s_2$, a szachista $s_2$ jest lepszy od szachisty $s_3$, to szachista $s_1$ jest lepszy od szachisty $s_3$? Oczywiście sprawa nie jest taka prosta. W przypadku, gdy różnica w poziomie gry pomiędzy kolejnymi szachistami jest znacząca, to powyższe stwierdzenie byłoby prawdziwe. Jednak w przypadku, gdy poziom gry jest prawie porównywalny, to na podstawie dwóch wymienionych przesłanek nie można wysnuć takich wniosków, gdyż może się okazać, że to właśnie szachista $s_3$ jest lepszy

od szachisty $s_1$. Dlatego też w pracy [12] autor doszedł do wniosku, że najlepszym rozwiązaniem będzie przeprowadzanie turniejów dla wszystkich osobników populacji (będących różnymi funkcjami oceniającymi – graczami) w systemie „każdy z każdym" i na podstawie takich wyników wyłaniany jest potencjalnie lepszy osobnik. W pracy [36] założono z kolei, że osobnicy populacji nie będą rozgrywali między sobą partii, tylko będą zgadywali poszczególne ruchy w 500 losowo wybranych pozycjach z 389 partii arcymistrzów (w tym przypadku lepszym osobnikiem okazywał się ten, który odgadywał prawidłowy ruch w większej liczbie pozycji). W obu przypadkach stwierdzono ukierunkowane zwiększanie możliwości szachowych osobników, jednakże nie osiągnięto żadnych zaskakujących wyników w porównaniu do ówcześnie grających programów szachowych.

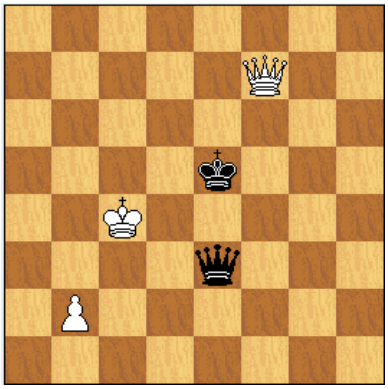
### 3.4.2. Temporal Difference Learning

Kamieniem milowym, jeśli chodzi o wykorzystywanie sieci neuronowych do optymalizowania funkcji oceniającej dowolnej gry, było sformułowanie w 1988 roku przez R. S. Suttona w pracy [30] metody uczenia *Temporal Difference Learning*. Jej zastosowanie przez Tesaura na początku lat 90. w grze backgammon zakończyło się wielkim sukcesem (szczegóły w [31]). Algorytm ten znany również pod nazwami *TD-Learning*, bądź *TD(λ)*, zaliczany jest do klasy przyrostowych procedur uczących ukierunkowanych na predykcję w nie do końca poznanych systemach, tzn. takich, w których nieznany jest rzeczywisty wynik końcowy dla określonego stanu, w którym może się znajdować. Procedury te wykorzystują swoje przeszłe doświadczenie do przewidywania przyszłego zachowania obserwowanego systemu.

Przykładem takiego systemu może być gra w szachy, w której przewidywany jest wynik końcowy dla określonej pozycji (Rys. 3.2.). Ogromna liczba możliwości jaką daje królewska gra nie pozwala nam z pewnością ocenić milionów podobnych sytuacji, ponieważ potrzebowalibyśmy w tym celu rozwinąć z każdej takiej pozycji całe drzewo gry i dopiero na jego podstawie moglibyśmy stwierdzić czy przy najlepszej grze obu stron możliwa jest czyjaś wygrana. W większości przypadków jednak rozwinięcie całego drzewa gry jest praktycznie niewykonalne ze względu na bardzo duży współczynnik rozgałęzienia. W przytoczonym dość prostym przykładzie (Rys. 3.2.) mamy do czynienia z pozycją, w której białe mają przewagę materialną (posiadają jednego pionka więcej). Okazuje się jednak, że stuprocentowa ocena tej pozycji nie jest do końca możliwa, ponieważ nie wiemy bynajmniej która strona ma w tej chwili wykonać posunięcie. Jeżeli założymy, że białe mają wykonać ruch, to w prosty sposób mogą one doprowadzić do sytuacji, która uważana jest przez szachistów za wygraną (grając hetmanem z pola f7 na pole e7 z szachem zmuszają czarne do wymiany hetmanów doprowadzając do sytuacji, w której białe bez przeszkód mogą wykorzystać przewagę wolnego pionka wędrując nim do pola przemiany znajdującego się na ósmej linii). Jeżeli jednak założymy, że czarne mają wykonać posunięcie, to sprawa nie jest już tak oczywista, ponieważ czarne mogą wykorzystać swojego hetmana w celu wiecznego szachowania białego króla i utrzymania np. remisu. Poniższy przykład jest o tyle prosty, iż korzystając ze wspomnianej wcześniej bazy końcówek szachowych[15] można łatwo sprawdzić jak powinna wyglądać optymalna gra każdej ze stron (w zależności od tego kto pierwszy ma wykonać ruch).

Algorytm *Temporal Difference Learning* w odróżnieniu od dotychczas znanych klasycznych metod uczenia sieci neuronowych nie porównuje bieżącej predykcji z właściwą odpowiedzią dla danego przykładu, tylko porównuje kolejne predykcje aż do momentu uzyskania właściwej odpowiedzi – stąd też jego nazwa *temporal-difference-learning* – co można dosłownie przetłumaczyć jako uczenie oparte na różnicach czasowych.

[15] http://www.shredderchess.com/online-chess/online-databases/endgame-database.html



Rys. 3.2. Przykład pozycji szachowej, w której trudno ocenić końcowy wynik gry

Zasadniczym założeniem w tej metodzie jest fakt, że właściwa odpowiedź systemu jest obserwowalna dopiero po określonej sekwencji danych wejściowych. Dlatego też pomiędzy kolejnymi wejściami w procesie uczenia wykorzystywane są tylko i wyłącznie predykcje uczącego się algorytmu. Ucząc system metodą *TD-Learning* nie musimy czekać do osiągnięcia końcowego stanu obserwowanego systemu, a co za tym idzie nie musimy składować w pamięci wszystkich obserwacji. Dzięki temu w systemach, w których właściwe wzmocnienie przychodzi po dość długim czasie, możemy zaoszczędzić sporo pamięci. Drugą zaletą jest sam fakt obliczeń – obliczenia wpływające na ewentualne zmiany systemu uczącego mogą być dokonywane w trakcie obserwacji kolejnych wejść (stanów obserwowanego systemu), a nie, tak jak w przypadku dotychczasowych metod, dopiero po otrzymaniu ostatecznej odpowiedzi. Wszystkie zalety podejścia *TD(λ)* uwidaczniają się tak naprawdę w tzw. problemach wielokrokowych (*multi-step prediction problems*), w których informacja o prawidłowości poszczególnych predykcji uzyskiwana jest po więcej niż jednym kroku (każdy krok, to kolejna obserwacja systemu). W każdym takim kroku uzyskiwana jest tylko częściowa informacja odpowiadająca prawidłowości predykcji, na którą składa się nowa obserwacja wejściowa wraz z nową dla niej predykcją. W tego typu problemach *TD(λ)* wprowadza szereg niespotykanych dotychczas możliwości uczenia wyrażanych różnymi wartościami parametru $\lambda$. Natomiast w problemach jednokrokowych (*single-step prediction problems*) działanie omawianego algorytmu sprowadza się właściwie do działania dotychczas znanych metod (np. *backpropagation*).

Wyniki eksperymentów przeprowadzonych w 1992 roku przez Tesaura przeszły jego nawet najśmielsze oczekiwania (szczegóły w pracy [34]). *TD-GAMMON* jest programem grającym w grę backgammon, w którą nauczył się grać na arcymistrzowskim poziomie dzięki metodzie uczenia *TD(λ)*. Algorytm ten został zastosowany w wersji *TD-GAMMON 0.0* do uczenia trójwarstwowej sieci neuronowej, w której warstwa wyjściowa składała się z 1-3 neuronów, warstwa ukryta z 40-80 oraz warstwa wejściowa ze 198. W uproszczeniu wyjście tej sieci miało oszacowywać ostateczny wynik gry dla określonych stanów gry (rozmieszczeń pionków na planszy). Liczba neuronów w warstwie ukrytej była dobierana eksperymentalnie, natomiast w warstwie wejściowej zależała ściśle od sposobu kodowania aktualnego stanu gry. W pierwszej wersji (0.0) Tesauro wykorzystał najprostszy i najbardziej surowy sposób reprezentacji stanu planszy nie wykorzystujący żadnej dodatkowej wiedzy na temat dodatkowych cech gry backgammon. Sieć neuronowa przyjmując odpowiednie wartości wejściowe obliczała swoje wyjście w klasyczny sposób. W swoim podejściu Tesauro zaproponował następujący sposób uczenia: sieć neuronowa grała jak najwięcej partii z samą sobą. W wersji 0.0 wybór ruchu ograniczał się do analizy tylko i wyłącznie pierwszego poziomu drzewa. Oczywiście wybierany był zawsze ten ruch, dla którego stan planszy po jego wykonaniu był najlepiej oceniany przez sieć. Każda gra traktowana była jako

oddzielne doświadczenie składające się z sekwencji obserwacji (poszczególnych pozycji) oraz ostatecznego wzmocnienia (wyniku gry). Tesauro skorzystał także z najbardziej przyrostowego schematu uczenia *TD(λ)*, tzn. wszystkie aktualizacje wag następowały po każdym ruchu. Stan początkowy sieci, tzn. wszystkie modyfikowalne wagi zostały wybrane w sposób losowy jako małe wartości rzeczywisto-liczbowe. Wynikiem tego była całkowicie losowa gra tego programu na początku. Jednakże już po kilkuset rozegranych partiach zaczął być zauważalny ciągły wzrost jego umiejętności. Ku wielkiemu zaskoczeniu, po rozegraniu 300 tysięcy gier z samym sobą, *TD-GAMMON 0.0* nauczył się grać równie mocno jak najlepszy znany do tej pory inny program grający w backgammon (wykorzystujący bardziej wyrafinowane cechy gry) – *Neurogammon*, program Tesaura oparty również na sieci neuronowej, ale nie uczony metodą *TD(λ)*. Oczywistym następstwem tego sukcesu była kontynuacja tych eksperymentów z wykorzystaniem wyspecjalizowanych cech gry backgammon w wersji *TD-GAMMON 1.0*, które jak się można spodziewać zakończyły się jeszcze większym sukcesem. Siła gry *TD-GAMMON 1.0* była już nieporównywalnie większa od innych znanych programów grających w backgammon i zaczęła poważnie zagrażać prawdziwym arcymistrzom tej gry. Kolejne wersje tego programu były ulepszane pod kątem drzewa przeszukiwań (głębokość, wybór kolejności analizowanych węzłów) oraz odpowiednim doborem liczby neuronów w warstwie ukrytej. Wykazywały one znaczny wzrost umiejętności gry. Program *TD-GAMMON* przyczynił się do zmiany sposobu dotychczasowego myślenia o grze początkowej - nauczył się grać pewne otwarcia całkowicie inaczej od ówczesnych mistrzów.

Ciekawe wyniki zastosowania łączonego podejścia uczenia sieci neuronowej metodą *TD-Learning* z uczeniem się na podstawie wyjaśnień (*explanation-based learning*) przeprowadzono w pracy [35] Thruna z programem *NeuroChess*. W procesie uczenia się tego programu wykorzystano dwie trójwarstwowe sieci neuronowe. Jedna z nich, będąca szczególną wersją podejścia *explanation-based learning*, była pomocna w reprezentacji szachowej wiedzy dziedzinowej (175 neuronów w warstwie wejściowej, 165 – w warstwie ukrytej, 175 – w warstwie wyjściowej). Była ona trenowana jako pierwsza na podstawie 120 tysięcy arcymistrzowskich partii do predykcji stanu szachownicy na dwa pół-ruchy naprzód (względem bieżącej pozycji). Druga sieć, uczona metodą *TD(0)* na podstawie własnych oszacowań, pełniła zadanie funkcji oceniającej pozycje prognozowane przez poprzednią sieć (175 neuronów w warstwie wejściowej, 0-80 – w warstwie ukrytej, 1 – w warstwie wyjściowej). Poziom programu *NeuroChess* mierzony był w regularnych odstępach czasu na podstawie bezpośredniej gry z ogólnodostępnym programem szachowym *GNUChess*, z którego został zaczerpnięty mechanizm przeszukiwania drzewa gry. W pierwszych 200 grach *NeuroChess* zdołał zdobyć zaledwie 2 punkty, ale w procesie uczenia stale polepszał swój wynik. Po rozegraniu ponad dwóch tysięcy partii okazało się, że zdołał zdobyć 100 punktów z ostatnich 400 gier. Autorzy zgodnie stwierdzili, iż *NeuroChess* w pewnym stopniu zdołał dostroić swoją funkcję oceniającą – zwracając uwagę na przewagę materialną, wymianę figur oraz ochronę króla. Niestety większość partii była bardzo źle rozgrywana w fazie początkowej, co w znacznej mierze decydowało o wielu remisach i przegranych *NeuroChess*.

Po nie do końca satysfakcjonujących wynikach programu *NeuroChess*, w 2001 roku pojawiły się interesujące wyniki zastosowania algorytmu uczenia *TD-Learning* w programie *KnightCap*, osiągnięte przez Jonathana Baxtera, Andrew Tridgella oraz Lexa Weavera w pracy [2]. Autorzy przedstawiają w niej pewną modyfikację omawianego algorytmu, który nazywają *TDLeaf(l)*, a w którym właściwie jedyną różnicą jest założenie, że wynikiem oceny nie jest ocena bieżących pozycji powstających podczas gry, tylko ocena najlepszego liścia drzewa gry wyprowadzanego dla bieżącej pozycji do określonej głębokości w minimaxowym podejściu. Największym sukcesem programu *KnightCap* było osiągnięcie

w jednym z eksperymentów znacznego wzrostu jego siły gry do poziomu mistrzowskiego. Autorzy podkreślają, iż wyniki te są w głównej mierze zasługą metody uczenia on-line, tzn. w grze z innymi szachistami na serwisach internetowych typu *FICS*[16] lub *ICC*[17]. Stwierdzają, że powyższy styl uczenia ma wiele zalet, głównie ze względu na różnorodność siły gry przeciwników. Na tego typu serwisach większość użytkowników preferuje grę z przeciwnikami na porównywalnym poziomie, tzn. mających podobny wynik rankingowy. Dzięki temu *KnightCap* zyskując wciąż na rankingu, miał możliwość gry z coraz silniejszymi szachistami. Najlepszy ranking uzyskany przez *KnightCap* na serwisie *FICS* wynosił nawet 2632, ale w tym wypadku nauczanie rozpoczęto od starannie dobranych wartości wszystkich modyfikowalnych parametrów (wag). Jednakże w wiodącym eksperymencie, w którym ustalono jedynie parametry dotyczące sił materialnych (resztę ustalając na zero) *KnightCap* w procesie nauki potrafił w przeciągu 1000 gier (rozegranych na serwisie *FICS*) polepszyć swoją grę z rankingu 1600 do 2200. Zaimplementowanie dodatkowo książki otwarć pozwoliło osiągnąć ranking rzędu 2500 na serwisie ICC. Autorzy zwracają jednak uwagę na fakt, iż szybkość działania zaimplementowanego programu pozostawia wiele do życzenia. Proces obliczeniowy działania sieci neuronowej związanej z wyznaczaniem wyjścia na podstawie wyszukanych cech gry był bardzo czasochłonny – porównując z *Deep Blue* działał on 6 tysięcy razy wolniej. *KnightCap* grając z programami przeszukującymi drzewa gry szybciej i głębiej stał zwykle na straconej pozycji. Autorzy ostatecznie podkreślają, iż w procesie uczenia bardzo ważne jest środowisko treningowe (w tym wypadku najlepszym jest on-line) oraz odpowiedni dobór wag przed rozpoczęciem nauki, aby przyspieszyć jego proces i wystartować jak najbliżej pewnego optymalnego rozwiązania, do którego uczony system będzie zdążał.

[16] *Free Internet Chess Server* - http://freechess.org/
[17] *The Internet Chess Club* - http://www.chessclub.com/

# 4. Opis własnego systemu NeuralChess wykorzystującego sztuczną sieć neuronową uczoną algorytmem Temporal Difference Learning w poszukiwaniu optymalnej strategii gry w szachy

Po dokonaniu przeglądu różnych inteligentnych metod wykorzystywanych na przestrzeni kilkudziesięciu ostatnich lat w usprawnianiu gry programów szachowych, przejdziemy do opisu rozwiązania zastosowanego w autorskim programie grającym w szachy (*NeuralChess*) stworzonym na użytek pracy. Autor niniejszej pracy zachęcony ciekawymi wynikami programu *KnightCap* z pracy [2] postanowił wykorzystać do oceny stanu gry sieć neuronową uczoną metodą *Temporal Difference Learning*. Proces uczenia polega na modyfikacji wag sieci (zgodnie z algorytmem *TDL*) każdorazowo po rozegraniu partii z przeciwnikiem bądź przejrzeniu partii rozegranych przez dwóch szachistów. W ocenie różnych konfiguracji plansz, sieć neuronowa wykorzystuje podawane jej na wejściu informacje o stanie planszy reprezentowane przez cechy gry. Podczas bezpośredniej gry z określoną strategią bądź obserwacji partii rozegranej przez dwóch szachistów, sieć korzystając z informacji wejściowych ocenia wszystkie możliwe ruchy, które można w danej chwili wykonać i na tej podstawie w grze wybiera dla siebie najlepsze możliwe posunięcie, a podczas obserwacji partii pozostaje przy ocenie posunięcia wykonanego przez szachistę. Uzyskany ostatecznie rezultat partii (zarówno rozegranej jak i przeanalizowanej) wykorzystywany jest jako wzmocnienie uczenia, po którym następuje aktualizacja stanu sieci.

W niniejszej pracy zdecydowano się wykorzystać tę samą ideę uczenia co w przypadku programu *KnightCap* z tą samą drobną modyfikacją *TDLeaf(λ)*, jednak przy całkowicie innych założeniach, tzn. wykorzystano trójwarstwową sieć neuronową (a nie dwuwarstwową), założono aktualizację stanu sieci po każdej partii (a nie po czterech) oraz zaproponowano własny zestaw cech gry podawanych na wejście sieci.

W dalszej części pracy zostaną omówione szczegóły zarysowanej tutaj idei. Rozpoczniemy od przedstawienia zaproponowanych przez autora cech gry, które będą podawane na wejście sieci neuronowej. Następnie opisana zostanie ogólna koncepcja stworzonego programu szachowego wraz ze szczegółami dotyczącymi samego procesu uczenia.

## 4.1. Cechy gry w szachy

Najważniejszym etapem w tworzeniu programu szachowego jest zaproponowanie odpowiednich cech gry w szachy, przydatnych w procesie oceny poszczególnych pozycji szachowych. Autor niniejszej pracy, będąc zagorzałym szachistą próbował samodzielnie określić cechy gry w szachy na podstawie jednej z najsłynniejszych książek szachowych [16] napisanej przez legendarnego szachistę Arona Nimzowitscha, wybitnego teoretyka szachowego, który wniósł bardzo duży wkład w rozwój teorii szachów. Jego książka „Mój system” utworzona w pierwszej połowie XX wieku do dzisiaj uważana jest za klasykę strategii szachowej, do której warto zaglądać. W wyniku opracowany został zestaw 45 cech gry pogrupowanych w sposób naturalny w cztery kategorie tematyczne: ogólne, związane z figurami, pionkami oraz królami. Definiując kolejne cechy starano się utrzymywać konwencję, w której dodatnia wartość cechy wyznacza jej występowanie na korzyść koloru białego, a w przeciwnym przypadku – koloru czarnego. Ponieważ wartości wszystkich cech gry podawane są na wejście uczącej się sieci neuronowej, dlatego zostały one znormalizowane do przedziału wartości [-1,1]. Każda zaproponowana przez autora cecha gry została dalej dokładnie omówiona.

*Ogólne cechy gry* zestawione w Tab. 4.1. związane są z całkowicie elementarnymi pojęciami szachowymi, takimi jak wspomniana wartość materialna figur, szachowanie i matowanie króla oraz kolor figur będących na ruchu. W sposób oczywisty z wartością materialną figur związane są takie cechy jak siła pionów, skoczków, gońców, wież oraz hetmanów. Reszta cech nie wymaga w tej chwili komentarza.

*Cechy gry związane z figurami* zestawione w Tab. 4.1. dotyczą wszelkich charakterystyk, które można utożsamiać ze wspomnianą poprzednio wartością pozycyjną figur, a więc ich zdolnością do realizacji pewnych zadań. Bliskość figur do poziomej bądź pionowej linii środkowej wyraża planowany rozwój figur w stronę środka szachownicy skąd możliwy jest ewentualny dalszy atak w stronę obozu przeciwnika. Aktywność figur związana jest stricte z fazą początkową gry, w której niezwykle ważne jest wyprowadzanie figur z pozycji wyjściowej. Agresywność figur wiąże się z zaangażowaniem własnych figur w walkę, dzięki której wiele figur przeciwnika znajduje się w niebezpieczeństwie. Przestrzeń figur oraz opanowanie szachownicy wyznacza zarówno swobodę ruchów własnych figur jak i jej ograniczenie dla figur przeciwnika. Cechy związane z otwartymi liniami oraz (niebezpiecznymi) wieżami i hetmanami z nich korzystających z jednoczesnym ich wzmocnieniem w kolumnie dotyczą wszystkich ważnych cech pozycyjnych, w których najlepiej czują się ciężkie figury. Cechy takie jak potencjalne przyczółki i przyczółki są niejako konsekwencją wykorzystywania otwartych linii. Związania figur, baterie czy końskie widełki to cechy wyznaczające pewne niebezpieczne sytuacje szachowe z udziałem figur, po których może dochodzić potencjalnie do diametralnie różnych ocen pozycji.

| **Cechy gry** | |
|---|---|
| ogólne | związane z figurami |
| 1. siła pionów<br>2. siła skoczków<br>3. siła gońców<br>4. siła wież<br>5. siła hetmanów<br>6. szach<br>7. szach z możliwą ucieczką króla<br>8. mat<br>9. kolor | 10. bliskość figur do poziomej linii środkowej<br>11. bliskość figur do pionowej linii środkowej<br>12. aktywność figur<br>13. agresywność figur<br>14. przestrzeń figur<br>15. otwarte linie<br>16. otwarte wieże/hetmany<br>17. niebezpieczne wieże<br>18. wzmocnienie ciężkich figur w kolumnie<br>19. potencjalne przyczółki<br>20. przyczółki<br>21. związanie figur z królem/hetmanem<br>22. baterie<br>23. gońce Gorwitza<br>24. końskie widełki<br>25. opanowanie szachownicy |

Tab. 4.1. Wykaz *ogólnych cech gry* oraz *cech gry związanych z figurami*

*Cechy gry związane z pionkami* zestawione w Tab. 4.2. dotyczą wszelkich możliwych sytuacji mogących powstać na szachownicy, które wzmacniają lub osłabiają zarówno wartość pozycyjną jak i wartość materialną pionków. Cechy te zyskują szczególne znaczenie w końcówkach, w których uczestniczy już niewiele figur, jednak nie mniej ważne bywają również w środkowej jak i początkowej fazie gry. Na przykład aktywność centralnych

i przycentralnych pionów oraz ewentualne wzmocnienie pionowego centrum są nieodłącznymi elementami początkowej i środkowej fazy gry. Mają one na celu uwolnić własne figury, tj. gońce, wieże czy hetmana, które w pozycji wyjściowej nie mają możliwości rozwoju. Zdobycie centrum pionowego ma również swoje elementarne znaczenie strategiczne, które realizuje planową grę w stronę środka szachownicy, co potrafi znacznie komplikować grę poprzez zacieśnienie pozycji. Strona, która oddaje centrum i pozwala rozwijać się przeciwnikowi z góry zakłada konieczność obrony, która ma również swoje znaczenie w sensie przeczekania środkowej fazy gry i wykorzystania swoich sił w końcówce. Z końcówkami najczęściej związane są pojęcia wolnych pionków, które mają otwartą drogę do pola przemiany oraz kandydatów na wolne pionki, które mają duże szanse stać się wolnymi pionkami. Wartość materialna takich pionków wzrasta gdyż mają one potencjalnie szansę osiągnąć pole przemiany, na którym przeobrażają się w dowolną figurę. Biorąc pod uwagę takie pionki można by wyróżnić szereg dodatkowych pozycyjnych cech, które również decydują o wartości pionków, np. ich zaawansowanie, czyli bliskość do pola przemiany, ich połączenie bądź duże oddalenie znacznie utrudniające przeciwnikowi zatrzymanie takich pionków, ich wzmocnienie przez inne pionki bądź figury, które pozwalają utrzymać ich wolność w dłuższej perspektywie. Innymi szczególnymi typami pionków są również pionki izolowane bądź zdwojone, które przeważnie tracą na wartości materialnej, gdyż są łatwiejszym punktem ataku przez przeciwnika. Są jednak szachiści, którzy preferują posiadanie takich pionków, gdyż powodują one znaczną koncentrację na nich sił obu stron - jedna chce go zdobyć, druga wybronić i wykorzystać w odpowiednim momencie w celu otwarcia gry.

| **Cechy gry** | |
|---|---|
| związane z pionkami | związane z królami |
| 26. przewaga pionowa na skrzydłach<br>27. aktywność centralnych pionów<br>28. aktywność przycentralnych pionów<br>29. wzmocnienie pionowego centrum<br>30. wolne pionki<br>31. zaawansowanie wolnych pionków<br>32. połączone wolne pionki<br>33. oddalone wolne pionki<br>34. wzmocnienie wolnych pionków<br>35. wzmocnienie wolnych pionków przez wieże<br>36. blokada wolnych pionków przeciwnika<br>37. kandydaci na wolne pionki<br>38. zaawansowanie kandydatów na wolne pionki<br>39. wzmocnienie kandydatów na wolne pionki<br>40. izolowane pionki<br>41. zdwojone pionki | 42. bliskość króli do centrum<br>43. bezpieczeństwo króli<br>44. królowie przewodnicy pionków<br>45. królewska blokada wolnego pionka |

Tab. 4.2. Wykaz *cech gry związanych z pionkami* oraz *cech gry związanych z królami*

*Cechy gry związane z królami* zestawione w Tab. 4.2. dotyczą kilku strategicznych motywów związanych z najważniejszymi figurami na szachownicy – królami. Trzy z nich: zbliżenie do centrum, przewodniczenie wolnemu pionkowi, blokowanie wolnego pionka

przeciwnika dotyczą tylko i wyłącznie końcowej fazy gry, w której bardzo często królowie muszą wkroczyć do akcji, aby pomóc swoim pionkom i innym nielicznym figurom w realizacji przewagi pozycyjnej. W końcówkach, ze względu na niewielką liczbę figur, wskazana jest ich współpraca z królami, którzy nie muszę obawiać się ewentualnego zamatowania. Natomiast cecha związana z bezpieczeństwem króla jest niesłychanie ważna w środkowej fazie gry, w której należy zapewnić królowi najbezpieczniejsze miejsce na szachownicy. Bezpieczeństwo to utożsamione jest z umiejscowieniem króla blisko rogu szachownicy za niewyprowadzonymi własnymi pionkami. Przerzucenie w takie miejsce króla wiąże się z planową grą do centrum obu stron, które wyprowadzają w jego stronę pionki i figury, co z kolei powoduje znaczne osłabienie króla z pozycji wyjściowej, który znajduje się w jednej z centralnych kolumn.

### 4.1.1. Ogólne cechy gry

W ogólnej grupie cech gry znajdują się cechy związane z elementarnymi pojęciami szachowymi, z których najważniejsza jest wartość materialna figur oraz świadomość szachowania oraz matowania króla przeciwnika. Poniżej znajdują się definicje wszystkich tych cech, po których umieszczono przykład pozycji z wyznaczonymi dla nich wartościami (Rys. 4.1.). Należy pamiętać, iż dodatnie wartości cech świadczą o ich występowaniu na korzyść koloru białego, a w przeciwnym razie – koloru czarnego.

**Siła pionów** określa przewagę jednej ze stron wynikającą z posiadania większej liczby pionków. Wyznaczana jest jako iloraz różnicy liczby pionów białych i liczby pionów czarnych do liczby pionów jednego koloru z pozycji początkowej – 8:

$$\langle sila\ pionow \rangle = \frac{n_B - n_C}{8}, \tag{4.1}$$

gdzie $n_B$ to liczba białych pionów, a $n_C$ – liczba czarnych pionów.

**Siła skoczków** określa przewagę jednej ze stron wynikającą z posiadania większej liczby skoczków. Wyznaczana jest jako iloraz różnicy liczby skoczków białych i liczby skoczków czarnych do liczby skoczków jednego koloru z pozycji początkowej – 2:

$$\langle sila\ skoczkow \rangle = \frac{n_B - n_C}{2}, \tag{4.2}$$

gdzie $n_B$ to liczba białych skoczków, a $n_C$ – liczba czarnych skoczków.

**Siła gońców** określa przewagę jednej ze stron wynikającą z posiadania większej liczby gońców. Wyznaczana jest jako iloraz różnicy liczby gońców białych i liczby gońców czarnych do liczby gońców jednego koloru z pozycji początkowej – 2:

$$\langle sila\ goncow \rangle = \frac{n_B - n_C}{2}, \tag{4.3}$$

gdzie $n_B$ to liczba białych gońców, a $n_C$ – liczba czarnych gońców.

**Siła wież** określa przewagę jednej ze stron wynikającą z posiadania większej liczby wież. Wyznaczana jest jako iloraz różnicy liczby wież białych i liczby wież czarnych do liczby wież jednego koloru z pozycji początkowej – 2:

$$\langle sila\ wiez \rangle = \frac{n_B - n_C}{2}, \tag{4.4}$$

gdzie $n_B$ to liczba białych wież, a $n_C$ – liczba czarnych wież.

**Siła hetmanów** określa przewagę jednej ze stron wynikającą z posiadania większej liczby hetmanów. Wyznaczana jest jako różnica liczby hetmanów białych i liczby hetmanów czarnych:

$$\langle sila\,hetmanow \rangle = n_B - n_C, \tag{4.5}$$

gdzie $n_B$ to liczba białych hetmanów, a $n_C$ – liczba czarnych hetmanów.

**Szach** określa sytuację, w której jedna ze stron jest zaszachowana – wyznaczany jest jako:

- 1.0 w przypadku gdy białe figury szachują czarnego króla,
- 0.0 w przypadku gdy żadna ze stron nie jest szachowana,
- -1.0 w przypadku gdy czarne figury szachują białego króla.

**Szach z możliwą jedynie ucieczką króla** określa sytuację, w której jedna ze stron jest zaszachowana w ten sposób, iż jedyną możliwą jej obroną przed szachem jest ucieczka króla. Wystąpienie takiego szacha na szachownicy może być o wiele groźniejsze od zwykłego szacha. Cecha ta wyznaczana jest jako:

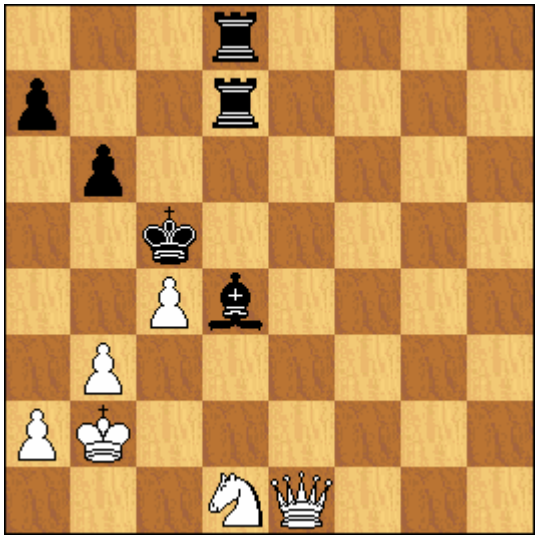
- 1.0 w przypadku gdy białe figury szachują czarnego króla w ten sposób, że jedyną jego obroną jest ucieczka,
- -1.0 w przypadku gdy czarne figury szachują białego króla w ten sposób, że jedyną jego obroną jest ucieczka,
- 0.0 w przypadku gdy nie występuje żadna z powyższych sytuacji.

**Mat** określa sytuację, w której jedna ze stron została zamatowana, a więc na szachownicy wystąpił szach, przed którym nie ma żadnej obrony. Cecha ta wyznaczana jest jako:

- 1.0 w przypadku gdy białe figury zamatowały czarnego króla,
- 0.0 w przypadku gdy żadna ze stron nie jest zamatowana,
- -1.0 w przypadku gdy czarne figury zamatowały białego króla.

**Kolor** określa gracza, który aktualnie ma wykonać posunięcie. Właściwie trudno dokładnie powiedzieć na ile istotna może stać się ta cecha w procesie oceny poszczególnych pozycji – niemniej jednak oczywiste jest, iż ocena pewnych pozycji szachowych może znacząco zależeć od strony, która ma wykonać posunięcie, chociaż ocena ta wynika przede wszystkim z analizy możliwych do wykonania posunięć, a nie samej informacji o kolorze. Warto jednak spróbować wykorzystać taką informację w ocenie pozycji. Cecha ta wyznaczana jest jako:

- 1.0 w przypadku gdy biały kolor znajduje się na ruchu,
- -1.0 w przypadku gdy czarny kolor znajduje się na ruchu.



$$\langle sila\ pionow \rangle = 0.125$$
$$\langle sila\,skoczkow \rangle = 0.5$$
$$\langle sila\,goncow \rangle = -0.5$$
$$\langle sila\,wiez \rangle = -1.0$$
$$\langle sila\,hetmanow \rangle = 1.0$$
$$\langle szach \rangle = -1.0$$
$$\langle szach\,z\,mozliwa\,ucieczka\,krola \rangle = 0.0$$
$$\langle mat \rangle = 0.0$$
$$\langle kolor\,na\,ruchu \rangle = 1.0$$

Rys. 4.1. Przykładowa pozycja z wyznaczonymi wartościami *ogólnych cech gry*

### 4.1.2. Cechy gry związane z figurami

W grupie cech gry związanych z figurami znajdują się cechy, które oddają wartość pozycyjną poszczególnych figur (nie będących pionkami), ale również zwracają uwagę na sytuacje, które mogą prowadzić do zwiększania tych wartości. Zaproponowane zostały cechy wyznaczające ogólną mobilność, aktywność i agresywność wszystkich figur, ale również cechy związane bezpośrednio z wybranymi figurami: skoczkami, gońcami, wieżami oraz hetmanami. Poniżej znajdują się definicje wszystkich tych cech. Na końcu przedstawiono trzy przykłady pozycji (Rys. 4.3-5.), dla których wyznaczono wartości prezentowanych cech. Konsekwentnie przyjęto założenie, iż dodatnia wartość cechy świadczy o jej występowaniu na korzyść koloru białego, a w przeciwnym razie – koloru czarnego.

**Bliskość figur do poziomej linii środkowej** wyraża przewagę jednej ze stron wynikającą z rozwinięcia własnych figur w obrębie środka (a właściwie poziomej linii środkowej) szachownicy. Cecha wyznaczana jest jako różnica średnich bliskości białych oraz czarnych figur (nie będących pionkami) do poziomej linii środkowej. Zakładając, że wiersze szachownicy numerowane są od 0 do 7, bliskość dowolnej figury do poziomej linii środkowej wyznaczana jest jako:

$$\langle bliskosc\ figury\ do\ poz.\rangle = \begin{cases} \dfrac{y+1}{4} & dla\ y < 4 \\ \dfrac{8-y}{4} & dla\ y \geq 4 \end{cases}, \tag{4.6}$$

gdzie $y$ jest wierszem szachownicy, w którym znajduje się określona figura. Zatem

$$\langle bliskosc\ figur\ do\ poziomej\ linii\ srodkowej\rangle = \frac{\sum_{i=1}^{n_B} \langle bliskosc\ figury\ do\ poz.\rangle_{B_i}}{n_B} - \frac{\sum_{i=1}^{n_C} \langle bliskosc\ figury\ do\ poz.\rangle_{C_i}}{n_C}, \tag{4.7}$$

gdzie $n_B$ to liczba białych figur, a $n_C$ – liczba czarnych figur.

**Bliskość figur do pionowej linii środkowej** wyraża przewagę jednej ze stron wynikającą z rozwinięcia własnych figur w strefie środkowej (a właściwie pionowej linii środkowej) szachownicy. Cecha wyznaczana jest jako różnica średnich bliskości białych oraz czarnych figur (nie będących pionkami) do pionowej linii środkowej. Zakładając, że kolumny szachownicy numerowane są od 0 do 7, bliskość dowolnej figury do pionowej linii środkowej wyznaczana jest jako:

$$\langle bliskosc\ figury\ do\ pion.\rangle = \begin{cases} \dfrac{x+1}{4} & dla\ x < 4 \\ \dfrac{8-x}{4} & dla\ x \geq 4 \end{cases}, \tag{4.8}$$

gdzie $x$ jest kolumną szachownicy, w której znajduje się określona figura. Zatem

$$\langle bliskosc\ figur\ do\ pionowej\ linii\ srodkowej\rangle =$$

$$\frac{\sum_{i=1}^{n_B}\langle bliskosc\ figury\ do\ pion.\rangle_{B_i}}{n_B} - \frac{\sum_{i=1}^{n_C}\langle bliskosc\ figury\ do\ pion.\rangle_{C_i}}{n_C}, \tag{4.9}$$

gdzie $n_B$ to liczba białych figur, a $n_C$ – liczba czarnych figur.

**Aktywność figur** określa przewagę jednej ze stron wynikającą z wyprowadzenia własnych figur z pozycji wyjściowej. Cecha wyznaczana jest jako różnica średnich aktywności białych oraz czarnych figur (nie będących pionkami). Aktywność dowolnej figury wyznaczana jest jako 1.0 w przypadku gdy znajduje się ona na polu szachownicy innym niż w pozycji wyjściowej, a 0.0 w przeciwnym przypadku. Zatem

$$\langle aktywnosc\ figur\rangle = \frac{\sum_{i=1}^{n_B}\langle aktywnosc\rangle_{B_i}}{n_B} - \frac{\sum_{i=1}^{n_C}\langle aktywnosc\rangle_{C_i}}{n_C}, \tag{4.10}$$

gdzie $n_B$ to liczba białych figur, a $n_C$ – liczba czarnych figur.

**Agresywność figur** wyraża przewagę jednej ze stron wynikającą z większej części atakowanych (obstrzeliwanych) figur przeciwnika. Cecha wyznaczana jest jako różnica części atakowanych figur czarnych oraz białych:

$$\langle agresywnosc\ figur\rangle = \frac{\langle l.zaatakowanych\rangle_C}{n_C} - \frac{\langle l.zaatakowanych\rangle_B}{n_B}, \tag{4.11}$$

gdzie $\langle l.\ zaatakowanych\rangle_B$ oraz $\langle l.\ zaatakowanych\rangle_C$ jest odpowiednio liczbą figur białych będących pod obstrzałem figur czarnych oraz liczbą figur czarnych będących pod obstrzałem figur białych, a $n_B$ oraz $n_C$ to odpowiednio liczba wszystkich białych oraz czarnych figur.

**Przestrzeń figur** określa przewagę jednej ze stron wynikającą z większej swobody ruchu figur. Cecha wyznaczana jest jako różnica średnich przestrzeni białych oraz czarnych figur (nie będących pionkami ani królami). Przestrzeń dowolnej figury wyznaczana jest jako iloraz liczby pól, na które figura może się dostać w jednym ruchu do maksymalnej liczby pól, na które figura mogłaby się dostać w najlepszym miejscu pustej szachownicy:

$$\langle prz.\ figury\rangle = \frac{\langle liczba\ dostepnych\ pol\rangle}{\langle max\ liczba\ pol\rangle}, \tag{4.12}$$

gdzie $\langle max\ liczba\ pol\rangle$ wynosi 8 – dla skoczka, 13 – dla gońca, 14 – dla wieży, 27 – dla hetmana. Zatem

$$\langle przestrzen\ figur\rangle = \frac{\sum_{i=1}^{n_B}\langle prz.\ figury\rangle_{B_i}}{n_B} - \frac{\sum_{i=1}^{n_C}\langle prz.\ figury\rangle_{C_i}}{n_C}, \tag{4.13}$$

gdzie $n_B$ to liczba białych figur, a $n_C$ – liczba czarnych figur, przy czym jeśli $n_B$ lub $n_C$ przyjmuje wartość 0, to związane z nim wyrażenie uzyskuje również wartość 0.

**Otwarte linie** określają przewagę jednej ze stron wynikającą z posiadania większej liczby otwartych linii. Linia (kolumna) jest otwarta dla określonego koloru wówczas, gdy nie znajduje się na niej żaden pion tego koloru. Otwarte linie mają szczególne znaczenie dla ciężkich figur, gdyż w środkowej fazie gry pozwalają w łatwy sposób atakować obóz przeciwnika z drugiego końca szachownicy (a więc z własnego obozu) i ewentualnie umożliwiają późniejsze szybkie wdarcie na teren przeciwnika. Na Rys. 4.3. białe dysponują siedmioma wolnymi liniami (kolumny b-h), a czarne sześcioma (kolumny a-f). Cecha wyznaczana jest jako różnica średniej liczby otwartych linii (kolumn) białych oraz czarnych. Stąd

$$\langle otwarte\ linie \rangle = \frac{\langle otwarte\ linie \rangle_B - \langle otwarte\ linie \rangle_C}{8}. \tag{4.14}$$

**Otwarte wieże/hetmany** wyrażają przewagę jednej ze stron wynikającą z wykorzystania większej części posiadanych ciężkich figur (wież i hetmanów) na otwartych liniach. Na Rys. 4.3. biała wieża znajduje się na otwartej linii, a czarna wieża – nie. Cecha wyznaczana jest jako różnica części ciężkich figur (wieże/hetmany) białych oraz czarnych stojących na otwartych liniach:

$$\langle otwarte\ wieze/hetmany \rangle = \frac{\langle otwarte\ w/h \rangle_B}{n_B} - \frac{\langle otwarte\ w/h \rangle_C}{n_C}, \tag{4.15}$$

gdzie $<otwarte\ w/h>_B$ oraz $<otwarte\ w/h>_C$ jest odpowiednio liczbą ciężkich figur białych oraz liczbą ciężkich figur czarnych znajdujących się na otwartych liniach, a $n_B$ oraz $n_C$ to odpowiednio liczba wszystkich białych oraz czarnych ciężkich figur, przy czym jeśli $n_B$ lub $n_C$ przyjmuje wartość 0, to związane z nim wyrażenie uzyskuje również wartość 0.

**Niebezpieczne wieże** określają przewagę jednej ze stron wynikającą z wdarcia większej części wież do obozu przeciwnika. Jest to niejako konsekwencja wykorzystywania w późnym stadium gry środkowej otwartych linii. Dobrze użyte wieże na tyłach przeciwnika potrafią być szczególnie niebezpieczne. Na Rys. 4.3. biała wieża jest „niebezpieczna", gdyż znajduje się w drugiej linii obozu przeciwnika. Cecha wyznaczana jest jako różnica części wież białych oraz czarnych znajdujących się w jednym z dwóch pierwszych wierszy obozu przeciwnika, tzn. jest to wiersz 6 lub 7 dla wież białych i 0 lub 1 dla wież czarnych (w numeracji wierszy od 0 do 7):

$$\langle niebezp.wieza \rangle = \begin{cases} 1 & (wieza_B \wedge y \in \{6,7\}) \vee (wieza_C \wedge y \in \{0,1\}) \\ 0 & wpp. \end{cases}, \tag{4.16}$$

gdzie $wieza_B$ i $wieza_C$ oznaczają odpowiednio, że wieża jest biała bądź czarna, a $y$ jest wierszem szachownicy, w którym znajduje się ta wieża. Stąd

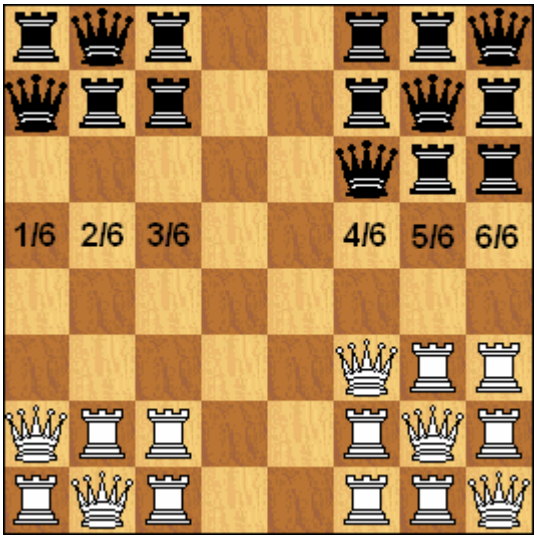
$$\langle niebezpieczne\ wieze \rangle = \frac{\sum_{i=1}^{n_B} \langle niebezp.wieza \rangle_{B_i}}{n_B} - \frac{\sum_{i=1}^{n_C} \langle niebezp.wieza \rangle_{C_i}}{n_C}, \tag{4.17}$$

gdzie $n_B$ to liczba białych wież, a $n_C$ – liczba czarnych wież, przy czym jeśli $n_B$ lub $n_C$ przyjmuje wartość 0, to związane z nim wyrażenie uzyskuje również wartość 0.

**Wzmocnienie ciężkich figur w kolumnie** wyraża przewagę jednej ze stron wynikającą z lepszego przegrupowania ciężkich figur współdziałających w kolumnie. Współdziałanie to zyskuje szczególnie na znaczeniu na otwartych liniach. Cecha wyznaczana jest jako różnica wzmocnień białych oraz czarnych ciężkich figur w kolumnie określanych na podstawie zajścia jednej z sześciu możliwych kombinacji ustawień dwóch wież oraz hetmana w kolumnie (Rys. 4.2.). Na Rys. 4.4. dwie białe wieże znajdują się w jednej z możliwych konfiguracji ciężkich figur w kolumnie. W danym ustawieniu ważna jest jedynie kolejność występowania figur w kolumnie, a nie ich dokładne umiejscowienie w wierszach. Jeśli w kolumnie znajdują się więcej niż dwie wieże bądź więcej niż jeden hetman, to wyznaczana jest najlepsza zachodząca dla danego koloru konfiguracja. Zatem

$$\langle wzmocnienie\,ciezkich\ figur\,w\,kolumnie\rangle = \sum_{i=1}^{8}\langle kombinacja\rangle_{B_i} - \sum_{i=1}^{8}\langle kombinacja\rangle_{C_i}, \tag{4.18}$$

gdzie $\langle kombinacja\rangle_{Bi}$ oraz $\langle kombinacja\rangle_{Ci}$ wyznaczają wartość ustawienia odpowiednio białych oraz czarnych ciężkich figur w *i*-tej kolumnie (zgodnie z Rys. 4.2.).




Rys. 4.2. Wartościowanie poszczególnych kombinacji ustawień ciężkich figur w kolumnie

**Potencjalne przyczółki** wyrażają przewagę jednej ze stron wynikającą z posiadania większej części otwartych linii, na których można potencjalnie ustawić przyczółek (zdefiniowany przy okazji omówienia następnej cechy). Potencjalnym przyczółkiem jest pole szachownicy bronione pionem i znajdujące się na otwartej linii w obozie przeciwnika. Na Rys. 4.4. potencjalnymi przyczółkami białych są pola b5 oraz e6 (na którym stoi biały skoczek), a czarne nie posiadają żadnego potencjalnego przyczółka. Cecha wyznaczana jest jako różnica średniej liczby potencjalnych przyczółków powstałych na otwartych liniach dla koloru białego oraz czarnego. Zatem

$$\langle potencjalne\ przyczolki\rangle = \frac{\langle l.pot.przycz.\rangle_B}{\langle l.otwartych\,linii\rangle_B} - \frac{\langle l.pot.przycz.\rangle_C}{\langle l.otwartych\,linii\rangle_C}, \tag{4.19}$$

gdzie $<l.\ pot.\ przycz.>_B$ oraz $<l.\ pot.\ przycz.>_C$ wyznaczają liczbę potencjalnych przyczółków odpowiednio dla koloru białego oraz czarnego, a $<l.\ otwartych\ linii>_B$ i $<l.\ otwartych\ linii>_C$ – liczbę otwartych linii odpowiednio koloru białego oraz czarnego, przy czym jeśli któraś z nich przyjmuje wartość 0, to związane z nim wyrażenie uzyskuje również wartość 0.

**Przyczółki** określają przewagę jednej ze stron wynikającą z posiadania większej części otwartych linii, na których ustawiono przyczółki (wykorzystano potencjalne przyczółki). Przyczółkiem jest figura stojąca na polu szachownicy znajdującym się w obozie przeciwnika na otwartej linii, które jednocześnie jest bronione pionem. Figura będąca przyczółkiem może stać się wyjątkowo groźną figurą, która broniona jednocześnie przez pionka i ewentualnie wieżę łatwo obstrzeliwuje tereny przeciwnika i pozwala również odkryć w najmniej oczekiwanym momencie atak wieży stojącej na otwartej linii. Na Rys. 4.4. przyczółkiem białych jest biały skoczek na polu e6. Cecha wyznaczana jest jako różnica średniej liczby wykorzystanych potencjalnych przyczółków białych oraz czarnych. Zatem

$$\langle przyczolki \rangle = \frac{\langle l.wyk.\ przycz. \rangle_B}{\langle l.\ pot.\ przycz. \rangle_B} - \frac{\langle l.wyk.\ przycz. \rangle_C}{\langle l.\ pot.\ przycz. \rangle_C}, \tag{4.20}$$

gdzie $<l.\ wyk.\ przycz.>_B$ oraz $<l.\ wyk.\ przycz.>_C$ wyznaczają liczbę wykorzystanych potencjalnych przyczółków odpowiednio dla koloru białego oraz czarnego, a $<l.\ pot.\ przycz.>_B$ i $<l.\ pot.\ przycz.>_C$ – liczbę potencjalnych przyczółków odpowiednio koloru białego oraz czarnego, przy czym jeśli któreś z nich przyjmuje wartość 0, to związane z nim wyrażenie uzyskuje również wartość 0.

**Związane figury z królem/hetmanem** wyrażają przewagę jednej ze stron wynikającą z mocniejszego uwiązania figur przeciwnika do obrony najcenniejszych figur (króla i hetmana). Uwiązane figury tracą w tym momencie na wartości pozycyjnej, gdyż nie mogą spełniać innych agresywnych zadań i muszą się zadowolić bierną postawą przysłaniającą obiekty ataku. W związaniu biorą udział trzy figury: związująca, związana oraz król lub hetman, będący przysłoniętym obiektem ataku. Związanie może mieć miejsce w jednej linii (wiersz lub kolumna) bądź diagonali. Na Rys. 4.4. związaniu uległa czarna wieża osłaniająca swojego króla przed atakiem białego gońca oraz biały goniec osłaniający swojego króla przed atakiem czarnego hetmana. Cecha wyznaczana jest jako iloraz różnicy zsumowanych wartości związań figur czarnych oraz białych do maksymalnej zsumowanej wartości tych związań. W zależności od figury związującej i związanej występującej na szachownicy przypisywana jest temu związaniu inna wartość (Tab. 4.3.)

| Figura związująca | Figura związana | Wartość związania |
|---|---|---|
| Goniec | Goniec | 1 |
| | Skoczek | 2 |
| | Wieża | 4 |
| | Hetman | 8 |
| Wieża | Wieża | 1 |
| | Skoczek lub Goniec | 2 |
| | Hetman | 6 |
| Hetman | Skoczek | 2 |
| | Goniec (związanie w linii) | 2 |
| | Wieża (związanie w diagonali) | 3 |

Tab. 4.3. Wykaz wartości związań w zależności od figury związującej i związanej

Wartości związań zostały dobrane przez autora dosyć arbitralnie, jednak miały one za zadanie oddać moc występującego związania, gdyż np. ze związania hetmana gońcem może wynikać jego strata przynajmniej w procesie wymiany, a jak wiadomo hetman jest silniejszą figurą niż goniec, co jednocześnie prowadzi do uzyskania przewagi materialnej. Natomiast związanie skoczka gońcem nie spowoduje już na pewno uzyskania przewagi materialnej, tylko co najwyżej przewagi pozycyjnej, w której skoczek uwiązany obroną swojej cennej figury nie może wykonywać żadnych innych zadań. Zatem

$$\langle zwiazane\ figury\ z\ krolem/hetmanem\rangle = \frac{\sum_{i=1}^{n_B}\langle zwiazanie\ figury\rangle_{B_i} - \sum_{i=1}^{n_C}\langle zwiazanie\ figury\rangle_{C_i}}{\max\left\{\sum_{i=1}^{n_B}\langle zwiazanie\ figury\rangle_{B_i}, \sum_{i=1}^{n_C}\langle zwiazanie\ figury\rangle_{C_i}\right\}}, \tag{4.21}$$

gdzie $\langle związanie\ figury\rangle_{Bi}$ oraz $\langle związanie\ figury\rangle_{Ci}$ wyznaczają wartość *i*-tego związania odpowiednio czarnej figury przez białą oraz białej figury przez czarną, a $n_B$ oraz $n_C$ określają liczbę wszystkich związań czarnych figur przez białe i białych figur przez czarne. Jeżeli na szachownicy nie występuje żadne związanie, to powyższe wyrażenie przyjmuje wartość 0.

**Baterie** określają przewagę jednej ze stron wynikającą z posiadania większej liczby baterii. Baterią jest dowolna figura, której odejście powoduje odsłonę szacha. Na Rys. 4.4. baterią jest biały skoczek stojący na polu e6. Prawie wszystkie baterie potrafią być niezwykle groźne ze względu na możliwość użycia ich do zdwojonego ataku. Wykonanie ruchu baterią oprócz odsłony szacha może powodować również niespodziewany atak na najsilniejsze figury przeciwnika. W takiej sytuacji przeciwnik musi myśleć o obronie własnego króla poprzez likwidację szacha i może nie mieć tym samym już możliwości wybronienia innych zaatakowanych figur. Cecha wyznaczana jest jako iloraz różnicy liczby baterii ustawionych przez białe oraz czarne do maksymalnej z tych liczb. Zatem

$$\langle baterie\rangle = \frac{\langle l.baterii\rangle_B - \langle l.baterii\rangle_C}{\max\left\{\langle l.baterii\rangle_B, \langle l.baterii\rangle_C\right\}}, \tag{4.22}$$

gdzie $\langle l.\ baterii\rangle_B$ oraz $\langle l.\ baterii\rangle_C$ wyznaczają liczbę odpowiednio białych oraz czarnych baterii. Jeżeli na szachownicy nie występuje żadna bateria, to powyższe wyrażenie przyjmuje wartość 0.

**Gońce Gorwitza** określają przewagę jednej ze stron wynikającą z posiadania większej liczby par gońców Gorwitza. Parą gońców Gorwitza są dwa gońce działające na sąsiednich diagonalach. Na Rys. 4.4. białe gońce tworzą parę gońców Gorwitza. Wartość pozycyjna gońców Gorwitza w większości sytuacji znacznie wzrasta, gdyż uzupełniają się one w swych działaniach obstrzeliwujących sąsiednie diagonale. Działania takie mogą stać się niezwykle kłopotliwe dla przeciwnika. Cecha wyznaczana jest jako iloraz różnicy liczby par gońców Gorwitza białych oraz czarnych do maksymalnej z tych liczb. Zatem

$$\langle gonce\,Gorwitza\rangle = \frac{\langle l.\,par\,GG\rangle_B - \langle l.\,par\,GG\rangle_C}{\max\left\{\langle l.\,par\,GG\rangle_B, \langle l.\,par\,GG\rangle_C\right\}}, \tag{4.23}$$

gdzie $\langle l.\ par\ GG\rangle_B$ oraz $\langle l.\ par\ GG\rangle_C$ wyznaczają liczbę par gońców Gorwitza odpowiednio białych oraz czarnych. Jeżeli na szachownicy nie występuje żadna para gońców Gorwitza, to powyższe wyrażenie przyjmuje wartość 0.

**Końskie widełki** określają przewagę jednej ze stron wynikającą z posiadania większej liczby skoczków nabijających figury przeciwnika na tzw. *widełki*. Skoczek tworzy widełki, jeśli atakuje przynajmniej dwie figury przeciwnika (z wyłączeniem pionków). Na Rys. 4.4. czarny skoczek tworzy widełki atakując dwie białe wieże. Stworzenie przez skoczka widełek szczególnie na ciężkich figurach wiąże się najczęściej ze zdobyciem przewagi materialnej. Cecha wyznaczana jest jako iloraz różnicy liczby skoczków białych oraz czarnych tworzących widełki do maksymalnej z tych liczb. Zatem

$$\langle konskie\,widelki\rangle = \frac{\langle l.widelek\rangle_B - \langle l.widelek\rangle_C}{\max\{\langle l.widelek\rangle_B, \langle l.widelek\rangle_C\}}, \tag{4.24}$$

gdzie $\langle l.\ widelek\rangle_B$ oraz $\langle l.\ widelek\rangle_C$ wyznaczają liczbę skoczków odpowiednio białych oraz czarnych tworzących widełki. Jeżeli na szachownicy żaden skoczek nie tworzy widełek, to powyższe wyrażenie przyjmuje wartość 0.

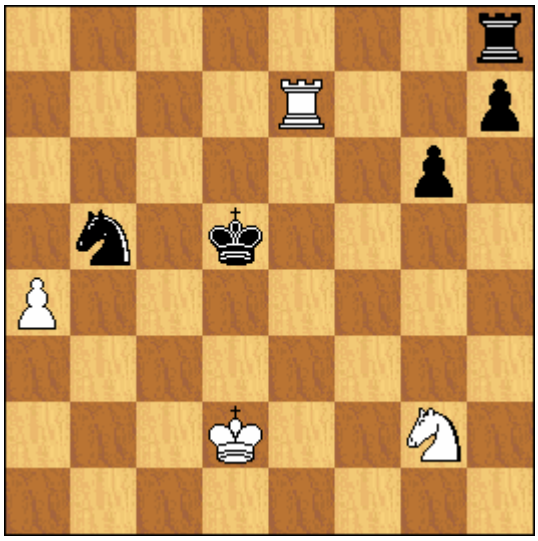
**Opanowanie szachownicy** wyraża przewagę jednej ze stron wynikającą z posiadania większej liczby opanowanych pól szachownicy. Pole jest opanowane, jeśli znajduje się na nim figura albo przynajmniej jedna figura może się na nim znaleźć w jednym posunięciu. Opanowanie szachownicy można utożsamić z przejmowaniem inicjatywy w grze – im większe jest opanowanie tym większa jest władza na szachownicy i tym większe możliwości działania figur. Cecha wyznaczana jest jako iloraz różnicy liczby pól, z których lub na które w jednym ruchu mogą zagrać białe oraz czarne figury do maksymalnej z tych liczb. Przykładowo na Rys. 4.5. białe opanowały 16 pól szachownicy, natomiast czarne – 8. Zatem

$$\langle opanowanie\ szachownicy\rangle = \frac{\langle l.\,pol\rangle_B - \langle l.\,pol\rangle_C}{\max\{\langle l.\,pol\rangle_B, \langle l.\,pol\rangle_C\}}, \tag{4.25}$$

gdzie $\langle l.\ pol\rangle_B$ oraz $\langle l.\ pol\rangle_C$ wyznaczają liczbę pól, z których lub na które w jednym ruchu mogą zagrać odpowiednio białe oraz czarne figury.

Na poniższych rysunkach 4.3.-5. przedstawiono trzy przykładowe pozycje szachowe, dla których wyznaczono wartości prezentowanych w tym rozdziale cech gry związanych z figurami. Na Rys. 4.3. mamy do czynienia z trudną dla obu stron końcówką. Chociaż bliskość figur do poziomej linii środkowej przemawia na korzyść czarnych, to jednak bliskość figur do pionowej linii środkowej przemawia na korzyść białych. Z punktu widzenia wartości tych cech oznacza to względnie podobne rozmieszczenie figur białych oraz czarnych na szachownicy. Łączna obserwacja tych dwóch cech nie pozwala stwierdzić, która ze stron ma w rzeczywistość lepiej rozstawione figury na szachownicy. Obserwując jednak wartości kilku następnych cech można by przypuszczać, iż mimo wszystko białe figury są w nieco lepszej sytuacji. Aktywność figur przemawia na korzyść białych ze względu na czarną wieżę schowaną w rogu szachownicy, która wygląda tak jakby się stamtąd nie ruszała przez całą partię. Białe figury są zdecydowanie bardziej agresywne, gdyż atakują dwie spośród pięciu figur przeciwnika, podczas gdy czarne nie atakują żadnej białej figury. Nieco lepsza przestrzeń białych figur wynika niewątpliwie z bardzo dobrze ustawionej białej wieży, której możliwości przemieszczenia są o wiele lepsze w porównaniu do wieży czarnej, a jednocześnie wykorzystuje ona jedną z otwartych linii, która pozwala na szybką komunikację między dwoma końcami szachownicy i spełnia dodatkowo rolę niebezpiecznej

wieży, która wdarła się do obozu przeciwnika i swoim działaniem na siódmej linii może sprawić czarnym jeszcze wiele przykrości. Podsumowując ostatecznie tę pozycję można rzec, iż przemawia ona nieznacznie na korzyść koloru białego, głównie za sprawą o wiele większej wartości pozycyjnej białej wieży, która rekompensuje w pewnym stopniu stratę materialną jednego pionka.

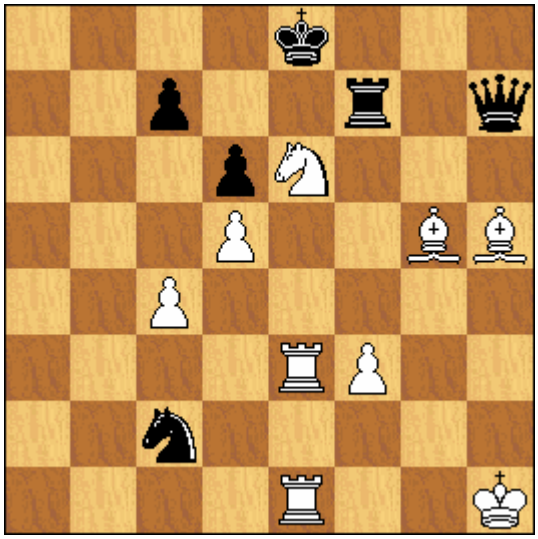


$$\langle bliskosc\ figur\ do\ poz.l.\ srodkowej \rangle = -0.25$$
$$\langle bliskosc\ figur\ do\ pion.l.srodkowej \rangle = 0.25$$
$$\langle aktywnosc\ figur \rangle = 0.333$$
$$\langle agresywnosc\ figur \rangle = 0.4$$
$$\langle przestrzen\ figur \rangle = 0.125$$
$$\langle otwarte\ linie \rangle = 0.125$$
$$\langle otwarte\ wieze\ /\ hetmany \rangle = 1.0$$
$$\langle niebezpieczne\ wieze \rangle = 1.0$$

Rys. 4.3. Przykładowa pozycja z pierwszą częścią wyznaczonych wartości cech gry związanych z figurami

Na Rys. 4.4. mamy z kolei do czynienia z bardzo ciekawą ostrą pozycją z nierównym materiałem, w której to białe rozgrywają partię bez hetmana kosztem wieży, dwóch gońców i pionka. Z punktu widzenia wartości materialnej można rzec, iż nieznaczną przewagę mają białe. Niemniej jednak bardzo istotne stają się w tej sytuacji wartości pozostałych cech pozycyjnych, które dodatkowo mogą wzmocnić lub osłabić przewagę materialną białych figur. Okazuje się, iż wyznaczone wartości cech dla tego przykładu przemawiają na korzyść białych. Ciężkie figury białych (dwie wieże) są wzmocnione w kolumnie, dzięki czemu współdziałając razem mogą łatwiej przedrzeć się do obozu przeciwnika. Białe dysponują dwoma potencjalnymi przyczółkami (pola b5 oraz e6) w przeciwieństwie do czarnych, które nie posiadają żadnego potencjalnego przyczółka. Z dwóch potencjalnych przyczółków białych jeden jest wykorzystany (skoczek na polu e6). W tym przypadku przyczółek tworzy jednocześnie szczególnie niebezpieczną baterię, która umożliwia odsłonę szacha i wprowadzenie białych wież po otwartej linii do obozu przeciwnika. Świetne ustawienie białych gońców (para gońców Gorwitza) również wzmacnia ich współdziałanie i w tym przypadku jednocześnie mocno zagraża czarnemu królowi, przez którego związana (unieruchomiona) staje się czarna wieża. Związanie białych jest nieco mocniejsze od czarnych ze względu na figury biorące udział w związaniu, tzn. biały goniec wiąże mocniejszą od siebie czarną wieżę, natomiast czarny hetman wiąże o wiele słabszego białego gońca. Jedyną cechą przemawiającą na korzyść czarnych są końskie widełki, które stawiają białe wieże w nieco kłopotliwej sytuacji, w której bardzo prawdopodobne staje się zdobycie jednej z nich. Najważniejsza jednak w tej pozycji okazuje się informacja o stronie, która ma wykonać ruch. W przypadku gdy ruch mają wykonać czarne, to w bardzo szybki sposób mogą one odzyskać przewagę materialną i jednocześnie zlikwidować występującą przewagę pozycyjną – bijąc hetmanem gońca z szachem, po którym następuje nieuchronne zbicie jednej z wież przez skoczka. W przypadku gdy jednak ruch mają wykonać białe, to stoją one przed szansą wykorzystania swojej przewagi pozycyjnej (pomimo groźnie

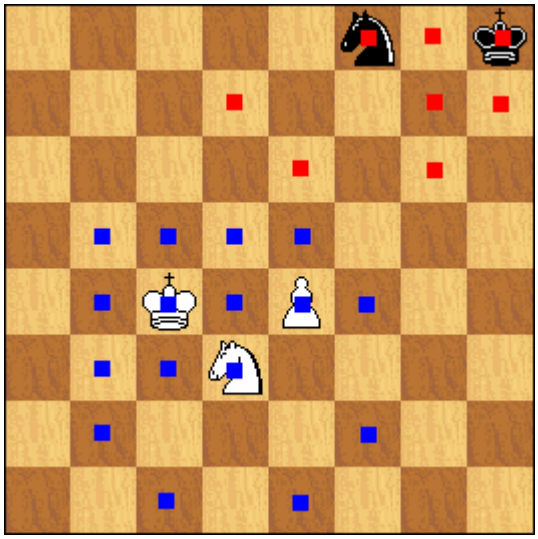
wyglądających końskich widełek) poprzez uruchomienie baterii z ewentualnym podwójnym szachem i wprowadzeniem wież na siódmą bądź ósmą linię.



$$\langle wzmocnienie\,ciezkich\ figur\,w\,kol.\rangle = 0.5$$
$$\langle potencjalne\ przyczolki\rangle = 0.4$$
$$\langle przyczolki\rangle = 0.5$$
$$\langle zwiazanie\ \ figur\,krolem/hetmanem\rangle = 0.5$$
$$\langle baterie\rangle = 1.0$$
$$\langle gonce\,Gorwitza\rangle = 1.0$$
$$\langle konskie\,widelki\rangle = -1.0$$

Rys. 4.4. Przykładowa pozycja z drugą częścią wyznaczonych wartości cech gry związanych z figurami

Ostatni przykład z Rys. 4.5. przedstawia jeszcze jedną cechę: opanowanie szachownicy. W wyniku lepszego rozstawienia białych figur opanowanie szachownicy przemawia na korzyść białych – w rzeczywistości białe kontrolują 16 pól szachownicy, a czarne tylko 8. W przypadku przykładowej końcówki cecha ta może mieć nieco mniejsze znaczenie, niemniej jednak daje ona obraz rozsądniejszego rozmieszczenia figur, które opanowują większą od przeciwnika część szachownicy.



$$\langle opanowanieszachownicy\rangle = 0.5$$

Rys. 4.5. Przykładowa pozycja z wyznaczoną wartością cechy opanowania szachownicy (niebieskie kwadraty wyznaczają pola opanowane przez białe, czerwone – przez czarne)

### 4.1.3. Cechy gry związane z pionkami

W grupie cech gry związanych z pionkami znajdują się cechy dotyczące tylko i wyłącznie wartości pozycyjnych pionków. Liczba zaproponowanych cech jest na tyle duża, iż postanowiono wydzielić je w osobnej kategorii. W większości przypadków zdefiniowane niżej cechy powinny zyskiwać na znaczeniu w końcowej fazie gry, w szczególności związane

z wolnymi pionkami oraz kandydatami na wolne pionki. Jednak część z nich powinna być też brana pod uwagę już od początku rozgrywania partii, np. związane z pionkami centralnymi oraz przycentralnymi, bądź pionkami zdwojonymi i izolowanymi. Po zdefiniowaniu wszystkich cech umieszczono dwa przykłady (Rys. 4.8.-9.) przedstawiające wartości wyznaczonych cech. Nadal świadomie korzystano z założenia, iż dodatnia wartość cech świadczy o ich występowaniu na korzyść koloru białego, a w przeciwnym przypadku – koloru czarnego.

**Przewaga pionowa na skrzydłach** wyraża przewagę jednej ze stron w liczbie pionów znajdujących się na jednym skrzydle. Szachownica podzielona jest na dwa skrzydła – hetmańskie (kolumny a-d) i królewskie (kolumny e-h) – względem pionowej linii środkowej. Przewaga pionowa na dowolnym skrzydle może się wiązać z ewentualnymi korzyściami płynącymi z przyszłego wyrobienia tzw. wolnego pionka (opisanego w jednej z dalszych cech), który może stać się szczególnie groźnym pionkiem w końcowej fazie gry. Cecha wyznaczana jest jako iloraz różnicy liczby przewag pionowych na skrzydłach dla białych oraz czarnych do liczby skrzydeł, tzn. 2. Przewagą pionową na danym skrzydle jest posiadanie na nim większej liczby pionków niż przeciwnik:

$$\langle przewaga\ pion.na\ skrz.\rangle = \begin{cases} 1 & \langle l.p.na\ skrz.\rangle > \langle l.p.\ p.na\ skrz.\rangle \\ 0 & wpp. \end{cases}, \qquad (4.26)$$

gdzie <*l. p. na skrz.*> wyznacza liczbę pionków danego koloru na rozpatrywanym skrzydle, a <*l. p. p. na skrz.*> – liczbę pionków przeciwnika (przeciwnego koloru) na tym samym skrzydle. Na Rys. 4.8. czarne mają przewagę pionową na skrzydle królewskim, a białe na skrzydle hetmańskim. Zatem

$$\langle przewaga\ pionowa\ na\ skrzydlach\rangle = \frac{\sum_{i=1}^{2}\langle przewaga\ pion.na\ skrz.\rangle_{B_i} - \sum_{i=1}^{2}\langle przewaga\ pion.na\ skrz.\rangle_{C_i}}{2}, \qquad (4.27)$$

gdzie <*przewaga pion. na skrz.*>$_{Bi}$ oraz <*przewaga pion. na skrz.*>$_{Ci}$ jest odpowiednio wartością przewagi pionowej białych oraz czarnych na *i*-tym skrzydle.

**Aktywność centralnych pionów** określa przewagę jednej ze stron związaną ze zdobyciem centrum szachownicy przez pionki. Zdobycie centrum jest jednym z elementów strategicznych początkowej i środkowej fazy gry, które umożliwia oswobodzenie własnych figur i które prowadzi do planowego rozwoju wydarzeń w środkowej części szachownicy. Cecha wyznaczana jest jako różnica aktywności centralnych pionów białych oraz czarnych (znajdujących się w kolumnach d i e) określanych na podstawie ich bliskości do centrum:

$$\langle akt.centralnego\ piona\rangle = \begin{cases} \frac{y-1}{4} & y < 4 \\ \frac{6-y}{4} & y \geq 4 \end{cases}, \qquad (4.28)$$

gdzie *y* jest wierszem szachownicy, w którym znajduje się centralny pion. Zatem

$$\langle aktywnosc\ centralnych\ pionow \rangle =$$

$$\sum_{i=1}^{n_B} \langle akt.centralnego\ piona \rangle_{B_i} - \sum_{i=1}^{n_C} \langle akt.centralnego\ piona \rangle_{C_i}, \quad (4.29)$$

gdzie $<akt.\ centralnego\ piona>_{Bi}$ oraz $<akt.\ centralnego\ piona>_{Ci}$ wyznacza aktywność *i*-tego centralnego piona odpowiednio białych jak i czarnych, a $n_B$ oraz $n_C$ to odpowiednio liczba wszystkich białych oraz czarnych pionków centralnych.

**Aktywność przycentralnych pionów** określa przewagę jednej ze stron wynikającą z włączenia do gry przycentralnych pionów pośrednio biorących udział w grze o centrum szachownicy. Cecha wyznaczana jest jako różnica aktywności przycentralnych pionów białych oraz czarnych (znajdujących się w kolumnach c i f) określanych na podstawie ich bliskości do poziomej linii środkowej:

$$\langle akt.\ przycentralnego\ piona \rangle = \begin{cases} \frac{y-1}{4} & dla\ y < 4 \\ \frac{6-y}{4} & dla\ y \geq 4 \end{cases}, \quad (4.30)$$

gdzie *y* jest wierszem szachownicy, w którym znajduje się przycentralny pion. Zatem

$$\langle aktywnosc\ przycentralnych\ pionow \rangle =$$

$$\sum_{i=1}^{n_B} \langle akt.\ przycentralnego\ piona \rangle_{B_i} - \sum_{i=1}^{n_C} \langle akt.\ przycentralnego\ piona \rangle_{C_i}, \quad (4.31)$$

gdzie $<akt.\ przycentralnego\ piona>_{Bi}$ oraz $<akt.\ przycentralnego\ piona>_{Ci}$ wyznacza aktywność *i*-tego przycentralnego piona odpowiednio białych jak i czarnych, a $n_B$ oraz $n_C$ to odpowiednio liczba wszystkich białych oraz czarnych pionków przycentralnych.

**Wzmocnienie pionowego centrum** wyraża przewagę jednej ze stron wynikającą ze wzmocnienia centralnych pionków przez pionki przycentralne. Obrona pionków centralnych przez inne pionki powoduje znaczne umocnienie zdobytego centrum. Cecha wyznaczana jest jako różnica liczby wzmocnień pionowego centrum białych oraz czarnych. Pionowe centrum jest wzmocnione jeśli pionek centralny (z kolumny d lub e) znajduje się w centrum szachownicy (z wiersza 4 lub 5) i jest broniony przez przycentralnego pionka (z kolumny c lub f). Zatem

$$\langle wzmocnienie\ pionowego\ centrum \rangle = \langle l.wzmocnien \rangle_B - \langle l.wzmocnien \rangle_C, \quad (4.32)$$

gdzie $<l.\ wzmocnien>_B$ oraz $<l.\ wzmocnien>_C$ wyznacza liczbę wzmocnień pionowego centrum białych jak i czarnych.

**Wolne pionki** określają przewagę jednej ze stron wynikającą z posiadania większej liczby wolnych pionków. Pionek jest wolny, jeśli przed nim w jego kolumnie oraz w dwóch sąsiadujących z nim kolumnach nie znajduje się żaden pionek przeciwnika. Na Rys. 4.9. wolnymi białymi pionkami są pionki b6 oraz c5, a czarnymi – a6. Posiadanie wolnych pionków szczególnie w końcówce może być niezwykle kłopotliwe dla przeciwnika, gdyż nie są one blokowane przez żadne jego pionki w drodze do pola przemiany. Przeciwnik zmuszony jest wówczas wykorzystywać swoje figury do ich biernego blokowania.

Cecha wyznaczana jest jako iloraz różnicy liczby wolnych pionków białych oraz czarnych do maksymalnej z tych liczb. Zatem

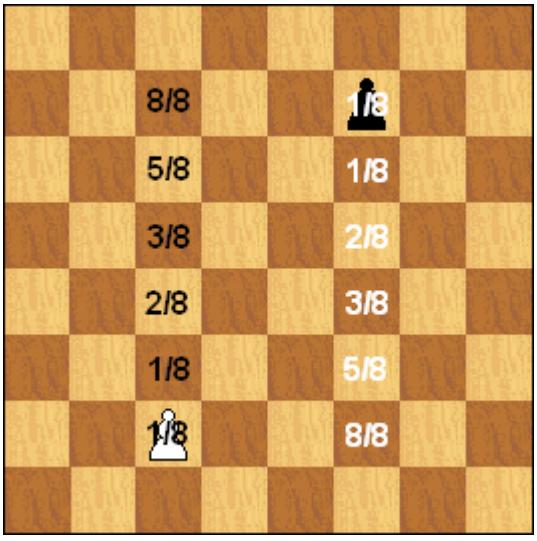
$$\langle wolne\ pionki \rangle = \frac{\langle l.wolnych\ pionkow \rangle_B - \langle l.wolnych\ pionkow \rangle_C}{\max\{\langle l.wolnych\ pionkow \rangle_B, \langle l.wolnych\ pionkow \rangle_C\}}, \qquad (4.35)$$

gdzie $<l.\ wolnych\ pionkow>_B$ oraz $<l.\ wolnych\ pionkow>_C$ wyznacza liczbę wolnych pionków odpowiednio białych oraz czarnych. Jeżeli na szachownicy nie znajduje się żaden wolny pionek, to powyższe wyrażenie przyjmuje wartość 0.

**Zaawansowanie wolnych pionków** wyraża przewagę jednej ze stron wynikającą z lepszego zbliżenia wolnych pionków do pól przemiany. Zaawansowanie wolnego pionka jest tym większe im znajduje się on bliżej pola przemiany – Rys. 4.6. Im bardziej zaawansowany staje się wolny pionek tym większa staje się jego wartość pozycyjna, gdyż nieuchronnie zbliża się on do pola, na którym przeobraża się w dowolną figurę szachową. Cecha wyznaczana jest jako różnica średniej zaawansowania wolnych pionków białych oraz czarnych. Zatem

$$\langle zaawansowanie\ wolnych\ pionkow \rangle = \frac{\sum_{i=1}^{n_B} \langle zaawansowanie\ w.p. \rangle_{B_i}}{n_B} - \frac{\sum_{i=1}^{n_C} \langle zaawansowanie\ w.p. \rangle_{C_i}}{n_C}, \qquad (4.36)$$

gdzie $<zaawansowanie\ w.p.>_{Bi}$ oraz $<zaawansowanie\ w.p.>_{Ci}$ wyznacza zaawansowanie *i*-tego wolnego pionka odpowiednio białych oraz czarnych, a $n_B$ oraz $n_C$ to odpowiednio liczba wszystkich białych oraz czarnych wolnych pionków, przy czym jeśli któraś z nich przyjmuje wartość 0, to związane z nim wyrażenie uzyskuje również wartość 0.




Rys. 4.6. Wartościowanie zaawansowania wolnego pionka białego oraz czarnego

**Połączone wolne pionki** określają przewagę jednej ze stron wynikającą z posiadania większej części wolnych pionków znajdujących się w sąsiednich kolumnach. Wolne pionki są połączone, jeśli znajdują się w sąsiednich kolumnach. Na Rys. 4.8. połączonymi wolnymi pionkami są białe pionki b6 oraz c5. Połączenie wolnych pionków jeszcze bardziej zwiększa ich wartość pozycyjną, gdyż jeszcze trudniej jest powstrzymać je przed przemarszem do pola

przemiany. Cecha wyznaczana jest jako różnica średniej liczby połączonych wolnych pionków białych oraz czarnych. Zatem

$$\langle polaczone\,wolne\,pionki \rangle = \frac{\langle l.pol.w.p. \rangle_B}{\langle l.w.p. \rangle_B} - \frac{\langle l.pol.w.p. \rangle_C}{\langle l.w.p. \rangle_C}, \tag{4.37}$$

gdzie $<l.\ pol.\ w.\ p.>_B$ oraz $<l.\ pol.\ w.\ p.>_C$ wyznacza liczbę połączonych wolnych pionków odpowiednio białych oraz czarnych, a $<l.\ w.\ p.>_B$ oraz $<l.\ w.\ p.>_C$ – liczbę wszystkich wolnych pionków odpowiednio białych oraz czarnych, przy czym jeśli któraś z nich przyjmuje wartość 0, to związane z nim wyrażenie uzyskuje również wartość 0.

**Oddalone wolne pionki** wyrażają przewagę jednej ze stron wynikającą z posiadania bardziej oddalonych wolnych pionków. Przeważnie w końcówkach z małą liczbą figur na szachownicy, oddalenie wolnych pionków (np. na dwóch skrajnych kolumnach szachownicy) może się przyczyniać do całkowitego braku obrony przeciwnika wobec przemarszu jednego z nich do pola przemiany. Utrudniona staje się jednoczesna obrona na obu zupełnie skrajnych częściach szachownicy. Cecha wyznaczana jest jako różnica średniej odległości wolnych pionków białych oraz czarnych od pionowej linii środkowej. Odległość wolnego pionka określana jest jako:

$$\langle odleglosc\,wolnego\,pionka \rangle = \frac{|7-2x|}{7}, \tag{4.38}$$

gdzie $x$ jest kolumną szachownicy, na której znajduje się określony wolny pionek. Zatem

$$\langle oddalone\,wolne\,pionki \rangle = \frac{\sum_{i=1}^{n_B} \langle odleglosc\,wolnego\,pionka \rangle_{B_i}}{n_B} - \frac{\sum_{i=1}^{n_C} \langle odleglosc\,wolnego\,pionka \rangle_{C_i}}{n_C}, \tag{4.39}$$

gdzie $<$*odległość wolnego pionka*$>_{Bi}$ oraz $<$*odległość wolnego pionka*$>_{Ci}$ wyznacza odległość $i$-tego wolnego pionka od pionowej linii środkowej odpowiednio białych oraz czarnych, a $n_B$ oraz $nC$ to odpowiednio liczba wszystkich białych oraz czarnych wolnych pionków, przy czym jeśli któraś z nich przyjmuje wartość 0, to związane z nim wyrażenie uzyskuje również wartość 0.

**Wzmocnienie wolnych pionków** określa przewagę jednej ze stron wynikającą z posiadania większej części wolnych pionków bronionych przez inne pionki (niekoniecznie wolne). Na Rys. 4.9. wzmocnionym wolnym pionkiem jest biały pionek b6 broniony przez pionka c5, oraz biały pionek c5 broniony przez pionka d4. Wzmocnienie wolnego pionka przez innego pionka znacznie zwiększa jego wartość pozycyjną, głównie ze względu na praktycznie niemożliwe pozbycie się go dopóki jest wzmocniony. Taki pionek nie będzie zbity przez żadną figurę, gdyż w następnym ruchu zostałaby ona zbita przez pionka, który go wzmacnia. Cecha wyznaczana jest jako różnica średniej liczby wzmocnionych wolnych pionków białych oraz czarnych. Zatem

$$\langle wzmocnienie\,wolnych\,pionkow \rangle = \frac{\langle l.wzm.w.p. \rangle_B}{\langle l.w.p. \rangle_B} - \frac{\langle l.wzm.w.p. \rangle_C}{\langle l.w.p. \rangle_C}, \tag{4.40}$$

gdzie $<l.\ wzm.\ w.p.>_B$ oraz $<l.\ wzm.\ w.p.>_C$ wyznacza liczbę wzmocnionych wolnych pionków odpowiednio białych oraz czarnych, a $<l.\ w.p.>_B$ oraz $<l.\ w.p.>_C$ – liczbę wszystkich wolnych pionków odpowiednio białych oraz czarnych, przy czym jeśli któraś z nich przyjmuje wartość 0, to związane z nim wyrażenie uzyskuje również wartość 0.

**Wzmocnienie wolnych pionków przez wieże** wyraża przewagę jednej ze stron wynikającą z większej części wzmocnionych wolnych pionków przez wieże. Wolny pionek jest wzmocniony przez wieżę, jeśli razem znajdują się w tej samej kolumnie i na drodze pomiędzy nimi nie znajduje się żadna inna figura. Na Rys. 4.9. wolnym pionkiem wzmocnionym przez wieżę jest biały pionek b6. Wzmocnienie pionka przez wieżę powoduje zwiększenie wartości pozycyjnej pionka w każdy miejscu jego drogi do pola przemiany, gdyż utrudnione staje się jego pozbycie przez przeciwnika. Cecha wyznaczana jest jako różnica średniej liczby wzmocnień wolnych pionków przez wieże białe oraz czarne. Zatem

$$\langle wzmocnienie\ wolnych\ pionkow\ przez\ wieze\rangle = \frac{\langle l.wzm.w.p.\rangle_B}{\min\{\langle l.w.p\rangle_B, \langle l.wiez\rangle_B\}} - \frac{\langle l.wzm.w.p.\rangle_C}{\min\{\langle l.w.p\rangle_C, \langle l.wiez\rangle_C\}}, \tag{4.41}$$

gdzie $<l.\ wzm.\ w.p.>_B$ oraz $<l.\ wzm.\ w.p.>_C$ wyznacza liczbę wzmocnionych wolnych pionków przez wieże odpowiednio białych oraz czarnych, $<l.\ w.p.>_B$ oraz $<l.\ w.p.>_C$ – liczbę wszystkich wolnych pionków odpowiednio białych oraz czarnych, a $<l.\ wiez>_B$ oraz $<l.\ wiez>_C$ – liczbę wszystkich wież odpowiednio białych oraz czarnych, przy czym jeśli któraś z nich przyjmuje wartość 0, to związane z nim wyrażenie uzyskuje również wartość 0.

**Blokada wolnych pionków przeciwnika** określa siłę obrony wobec wolnych pionków przeciwnika, polegającą na blokowaniu wolnych pionków przeciwnika. Wolny pionek jest zablokowany, jeśli bezpośrednio przed nim na drodze do pola przemiany stoi figura przeciwnika. Na Rys. 4.9. biały pionek c5 jest wolnym pionkiem zablokowanym przez czarnego skoczka. Cecha wyznaczana jest jako różnica średniej liczby zablokowanych wolnych pionków czarnych oraz białych. Zatem

$$\langle blokada\ wolnych\ pionkow\ przeciwnika\rangle = \frac{\langle l.zabl.\rangle_C}{\langle l.w.p.\rangle_C} - \frac{\langle l.zabl.\rangle_B}{\langle l.w.p.\rangle_B}, \tag{4.42}$$

gdzie $<l.\ zabl.>_B$ oraz $<l.\ zabl.>_C$ wyznacza liczbę zablokowanych wolnych pionków odpowiednio białych oraz czarnych, a $<l.w.p.>_B$ oraz $<l.w.p.>_C$ – liczbę wszystkich wolnych pionków odpowiednio białych oraz czarnych, przy czym jeśli któraś z nich przyjmuje wartość 0, to związane z nim wyrażenie uzyskuje również wartość 0.

**Kandydaci na wolne pionki** wyrażają przewagę jednej ze stron wynikającą z posiadania większej liczby kandydatów na wolne pionki. Kandydatem na wolnego pionka jest pionek, przed którym w jego kolumnie nie znajduje się żaden pionek przeciwnika, ale w przynajmniej jednej sąsiadującej z nim kolumnie taki pionek występuje. Na Rys. 4.8. kandydatem na wolnego pionka jest biały pionek d4. Kandydat na wolnego pionka zyskuje na wartości, gdyż ma on dużą szansę stać się w przyszłości jeszcze groźniejszym wolnym pionkiem. Cecha wyznaczana jest jako iloraz różnicy liczby kandydatów na wolne pionki białych oraz czarnych do maksymalnej z tych liczb. Zatem

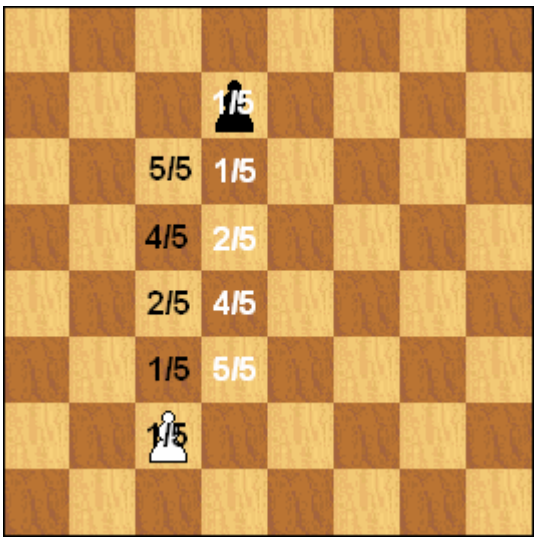
$$\langle kanydaci\,na\,wolne\,pionki\rangle = \frac{\langle l.k.na\,wolne\,pionki\rangle_B - \langle l.k.na\,wolne\,pionki\rangle_C}{\max\{\langle l.k.na\,wolne\,pionki\rangle_B, \langle l.k.na\,wolne\,pionki\rangle_C\}}, \tag{4.43}$$

gdzie $<l.k.\ na\ wolne\ pionki>_B$ oraz $<l.k.\ na\ wolne\ pionki>_C$ wyznacza liczbę kandydatów na wolne pionki odpowiednio białych oraz czarnych. Jeżeli na szachownicy nie znajduje się żaden kandydat na wolnego pionka, to powyższe wyrażenie przyjmuje wartość 0.

**Zaawansowanie kandydatów na wolne pionki** określa przewagę jednej ze stron wynikającą z lepszego zbliżenia kandydatów na wolne pionki do pól przemiany. Zaawansowanie kandydata na wolnego pionka jest tym większe im bliżej znajduje się on pola przemiany (pola w ostatnim wierszu swojej drogi przez szachownicę) – Rys. 4.7. Im bardziej zaawansowany staje się kandydat na wolnego pionka tym większa staje się jego wartość pozycyjna, gdyż nieuchronnie zbliża się on do pola, na którym przeobraża się w dowolną figurę szachową. Cecha wyznaczana jest jako różnica średniej zaawansowania kandydatów na wolne pionki białych oraz czarnych. Zatem

$$\langle zaawansowanie\,kandydatow\,na\,wolne\,pionki\rangle = \frac{\sum_{i=1}^{n_B}\langle zaawansowanie\,k.w.p.\rangle_{B_i}}{n_B} - \frac{\sum_{i=1}^{n_C}\langle zaawansowanie\,k.w.p.\rangle_{C_i}}{n_C}, \tag{4.44}$$

gdzie $<zaawansowanie\ k.w.p.>_{Bi}$ oraz $<zaawansowanie\ k.w.p.>_{Ci}$ wyznacza zaawansowanie *i*-tego kandydata na wolnego pionka odpowiednio białych oraz czarnych, a $n_B$ oraz $n_C$ to odpowiednio liczba wszystkich białych oraz czarnych kandydatów na wolne pionki, przy czym jeśli któraś z nich przyjmuje wartość 0, to związane z nim wyrażenie uzyskuje również wartość 0.




Rys. 4.7. Wartościowanie zaawansowania kandydata na wolnego pionka białego oraz czarnego

**Wzmocnienie kandydatów na wolne pionki** wyraża przewagę jednej ze stron wynikającą z posiadania większej części kandydatów na wolne pionki bronionych przez inne dowolne pionki. Na Rys. 4.9. wzmocnionym kandydatem na wolnego pionka jest biały pionek d4 broniony przez pionka e3. Oczywistą korzyścią ze wzmocnienia kandydata na wolnego pionka jest uniemożliwienie przeciwnikowi jego zdobycia. Cecha wyznaczana jest jako różnica średniej liczby wzmocnionych kandydatów na wolne pionki białych oraz czarnych. Zatem

$$\langle wzmocnienie\,kanydatow\,na\,wolne\,pionki\rangle = \frac{\langle l.wzm.k.w.p.\rangle_B}{\langle l.k.w.p.\rangle_B} - \frac{\langle l.wzm.k.w.p.\rangle_C}{\langle l.k.w.p.\rangle_C}, \tag{4.45}$$

gdzie $<l.\ wzm.\ k.w.p.>_B$ oraz $<l.\ wzm.\ k.w.p.>_C$ wyznacza liczbę wzmocnionych kandydatów na wolne pionki odpowiednio białych oraz czarnych, a $<l.k.w.p.>_B$ oraz $<l.k.w.p.>_C$ – liczbę wszystkich kandydatów na wolne pionki odpowiednio białych oraz czarnych, przy czym jeśli któraś z nich przyjmuje wartość 0, to związane z nim wyrażenie uzyskuje również wartość 0.

**Izolowane pionki** określają stronę, która znajduje się w posiadaniu większej części izolowanych pionków. Pionek jest izolowany, jeśli w dwóch sąsiadujących z nim kolumnach nie występuje pionek tego samego koloru. Na Rys. 4.8. izolowanymi pionkami są: biały pionek a4 oraz czarne pionki a5 i a7. Trudno mówić w przypadku tej cechy o przewadze którejś ze stron, gdyż pośród szachistów są nawet tacy, którzy preferują grę z izolowanymi pionkami, chociaż początkującym szachistom raczej odradza się taką grę. Uważa się, że gra z izolowanymi pionkami potrafi być skomplikowana, a same pionki są tak naprawdę najczęściej łatwym obiektem ataku przez przeciwnika. Cecha wyznaczana jest jako różnica średniej liczby izolowanych pionków białych oraz czarnych. Zatem

$$\langle izolowane\ pionki\rangle = \frac{\langle l.izol.\,pionkow\rangle_B}{\langle l.\,pionkow\rangle_B} - \frac{\langle l.izol.\,pionkow\rangle_C}{\langle l.\,pionkow\rangle_C}, \tag{4.33}$$

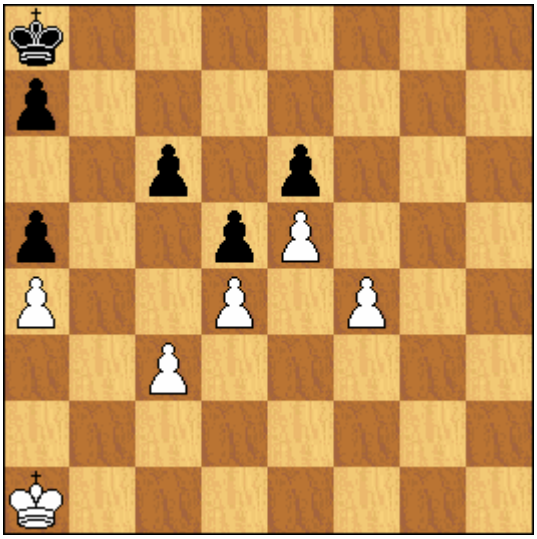
gdzie $<l.\ izol.\ pionkow>_B$ oraz $<l.\ izol.\ pionkow>_C$ wyznacza liczbę izolowanych pionków odpowiednio białych oraz czarnych, a $<l.\ pionkow>_B$ oraz $<l.\ pionkow>_C$ – liczbę wszystkich pionków odpowiednio białych oraz czarnych, przy czym jeśli któraś z nich przyjmuje wartość 0, to związane z nim wyrażenie uzyskuje również wartość 0.

**Zdwojone pionki** określają stronę, która znajduje się w posiadaniu większej części zdwojonych pionków. Pionki danego koloru są zdwojone, jeśli znajdują się w tej samej kolumnie. Na Rys. 4.8. zdwojonymi pionkami są czarne pionki a5 i a7. W przypadku tej cechy również trudno mówić o ewentualnej przewadze którejś ze stron, gdyż całkowicie zależy ona od rozstawienia i wykorzystania reszty figur. Ogólnie uważa się, że powstawanie zdwojonych pionków prowadzi do pogorszenia struktury pionowej, jednakże może nieść również ze sobą pewne korzyści, np. otworzenie linii czy zwiększenie pionów rezydujących w centrum szachownicy. Cecha wyznaczana jest jako różnica średniej liczby zdwojonych pionków białych oraz czarnych. Zatem

$$\langle zdwojone\ pionki\rangle = \frac{\langle l.zdw.\,pionkow\rangle_B}{\langle l.\,pionkow\rangle_B} - \frac{\langle l.zdw.\,pionkow\rangle_C}{\langle l.\,pionkow\rangle_C}, \tag{4.34}$$

gdzie $<l.\ zdw.\ pionkow>_B$ oraz $<l.\ zdw.\ pionkow>_C$ wyznacza liczbę zdwojonych pionków odpowiednio białych oraz czarnych, a $<l.\ pionkow>_B$ oraz $<l.\ pionkow>_C$ – liczbę wszystkich pionków odpowiednio białych oraz czarnych, przy czym jeśli któraś z nich przyjmuje wartość 0, to związane z nim wyrażenie uzyskuje również wartość 0.

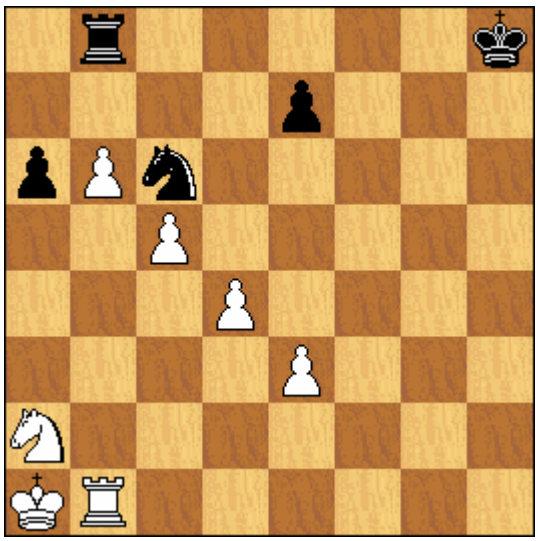
Na poniższych rysunkach 4.11-12. przedstawiono dwie przykładowe pozycje szachowe, dla których wyznaczono wartości prezentowanych w tym rozdziale cech gry związanych z pionkami. Na Rys. 4.8. mamy do czynienia z końcówką, w której obie strony dysponują tylko pięcioma pionkami (tzw. pionkówka). Obserwując wartości wyznaczonych cech gry można przypuszczać, iż białe znajdują się w nieco lepszej sytuacji. Chociaż przewaga pionowa na skrzydłach równoważy się – białe mają przewagę jednego pionka na skrzydle królewskim, a czarne na skrzydle hetmańskim – to jednak białe pionki lepiej i mocniej opanowują centrum (co prawda cechy te miałyby większe znaczenie w środkowej fazie gry). Duże znaczenie ma tutaj posiadanie przez białych kandydata na wolnego pionka, którego wykorzystanie mogłoby być kluczowe do doprowadzenia partii do zwycięstwa. Jednakże wartość takiej cechy została przedstawiona w kolejnym przykładzie.



$$\langle przewaga\ pionowa\ na\ skrzydlach \rangle = 0.0$$
$$\langle aktywnosc\ centralnych\ pionow \rangle = 0.25$$
$$\langle aktywnosc\ przycentralnych\ pionow \rangle = 0.5$$
$$\langle wzmocnienie\ pionowego\ centrum \rangle = 1.0$$
$$\langle izolowane\ pionki \rangle = -0.2$$
$$\langle zdwojone\ pionki \rangle = -0.4$$

Rys. 4.8. Przykładowa pozycja z pierwszą częścią wyznaczonych wartości cech gry związanych z pionkami

Na Rys. 4.9. mamy do czynienia z kolei z nieco bardziej skomplikowaną końcówką, w której występują jeszcze po dwie figury. W pierwszej chwili rzuca się w oczy przewaga materialna białych – dwa pionki więcej, wzmocniona dodatkowo ich wysoką wartością pozycyjną. Białe dysponują większą liczbą wolnych pionków (dwa do jednego), które nie dość, że są połączone, to jeszcze wzmocnione – jeden przez wieżę i pionka, drugi tylko przez pionka. Zaawansowanie białych wolnych pionków jest również o wiele większe, co sugeruje iż białe są już o krok od doprowadzenia jednego z wolnych pionków do pola przemiany i zwiększenia tym samym swojej wartości materialnej. Białe dysponują również jednym kandydatem na wolnego pionka (czarne żadnym), który jest dodatkowo wzmocniony kolejnym pionkiem. Czarne mogą się jedynie pochwalić ustawioną blokadą na jednego białego wolnego pionka, która powstrzymuje jego dalszą podróż w stronę pola przemiany. Niestety w tej pozycji blokada nie potrwa długo, gdyż białe korzystając z połączonych sił swoich pionków bardzo prędko mogą ją zlikwidować.



$$\langle wolne\ pionki \rangle = 0.5$$
$$\langle zaawansowanie\ wolnych\ pionkow \rangle = 0.375$$
$$\langle polaczone\ wolne\ pionki \rangle = 1.0$$
$$\langle oddalone\ wolne\ pionki \rangle = -0.429$$
$$\langle wzmocnienie\ wolnych\ pionkow \rangle = 1.0$$
$$\langle wzmocnienie\ wolnych\ p.\ przez\ wieze \rangle = 1.0$$
$$\langle blokada\ wolnych\ p.\ przeciwnika \rangle = -0.5$$
$$\langle kandydaci\ na\ wolne\ pionki \rangle = 1.0$$
$$\langle zaawansowanie\ kand.na\ wolne\ pionki \rangle = 0.4$$
$$\langle wzmocnienie\ kand.na\ wolne\ pionki \rangle = 1.0$$

Rys. 4.9. Przykładowa pozycja z drugą częścią wyznaczonych wartości cech gry związanych z pionkami

### 4.1.4. Cechy gry związane z królami

W grupie cech gry związanych z królami znajdują się cechy związane z kilkoma strategicznymi motywami dotyczącymi najważniejszych figur na szachownicy – króli. Tak naprawdę większość z nich nabiera znaczenia w końcowej fazie gry, w której król musi pomóc swoim figurom w realizacji pewnych zadań i wtedy wychodzi na środek szachownicy, prowadzi wolne pionki do pola przemiany, bądź blokuje wolne pionki przeciwnika przed dojściem do pola przemiany. Jest jednak jedna cecha związana z bezpieczeństwem króla, która nabiera swojej ważności już w środkowej fazie gry, w której obie strony dążą do zdobywania centrum. Poniżej znajdują się definicje zaproponowanych cech, a po nich przedstawiono jeden przykład pozycji (Rys. 4.10.), dla którego wyznaczono ich wartości.

**Bliskość króli do centrum** wyraża przewagę jednej ze stron wynikającą z lepszego umiejscowienia króla w okolicach centrum szachownicy. Król znajdujący się blisko centrum zyskuje na aktywności w końcowej fazie gry, dzięki której może realizować dalszy wypad na dowolne skrzydło, na którym potrzebna jest jego pomoc. Cecha wyznaczana jest jako różnica odległości czarnego oraz białego króla do centrum. Odległość króla do centrum określana jest jako:

$$\langle odleglosc\ krola\ od\ centrum \rangle = \frac{|x - 3.5| + |y - 3.5|}{6}, \qquad (4.46)$$

gdzie $x$ jest kolumną, a $y$ – wierszem szachownicy, w których znajduje się król. Zatem

$$\langle bliskosc\ kroli\ do\ centrum \rangle = \langle odl.k.od\ cent. \rangle_C - \langle odl.k.od\ cent. \rangle_B, \qquad (4.47)$$

gdzie $<odl.\ k.\ od\ cent.>_B$ oraz $<odl.\ k.\ od\ cent.>_C$ wyznacza odległość odpowiednio białego oraz czarnego króla od centrum.

**Bezpieczeństwo króli** określa stan zabezpieczenia króli obu stron. Król jest bezpieczny (*<bezpieczny krol>*=1.0) jeśli znajduje się w swoim wierszu wyjściowym oraz w kolumnach *a-c* lub *g-h*, a w jego bezpośrednim sąsiedztwie znajduje się przynajmniej jeden pionek tego

samego koloru. Na Rys. 4.10. biały król jest w pozycji bezpiecznej. Bezpieczeństwo króli, utożsamione z jego umiejscowieniem blisko rogu szachownicy za swoimi pionkami, jest szczególnie ważne w środkowej fazie gry. Cecha wyznaczana jest jako różnica wartości bezpiecznego króla białych oraz czarnych. Zatem

$$\langle bezpieczenstwo\ kroli \rangle = \langle bezpieczny\ krol \rangle_B - \langle bezpieczny\ krol \rangle_C, \qquad (4.48)$$

gdzie $<bezpieczny\ krol>_B$ oraz $<bezpieczny\ krol>_C$ wyznacza wartość bezpiecznego króla odpowiednio białego oraz czarnego.

**Królowie przewodnicy wolnych pionków** wyrażają przewagę jednej ze stron wynikającą z torowania drogi do pola przemiany swojemu wolnemu pionkowi. Król jest przewodnikiem wolnego pionka *(*<*krol przewodnik*>=1.0), jeśli w odległości do dwóch wierszy przed pionkiem (po stronie, w którą porusza się pionek) i w odległości do jednej kolumny znajduje się król tego samego koloru co pionek (w przeciwnym przypadku <*krol przewodnik*>=0.0). Na Rys. 4.10. czarny król jest przewodnikiem wolnego pionka d6. Królewskie przewodniczenie wolnemu pionkowi, w celu doprowadzenia go do pola przemiany, staje się niezwykle ważne w końcowej fazie gry, w której oba kolory dysponują już tylko królami i pionkami. Cecha wyznaczana jest jako różnica wartości króla przewodnika białych oraz czarnych. Zatem

$$\langle krolowie\ przewodnicy\ pionow \rangle = \langle krol\ przewodnik \rangle_B - \langle krol\ przewodnik \rangle_C, \qquad (4.49)$$

gdzie $<krol\ przewodnik>_B$ oraz $<krol\ przewodnik>_C$ wyznacza wartość odpowiednio białego oraz czarnego króla przewodnika.

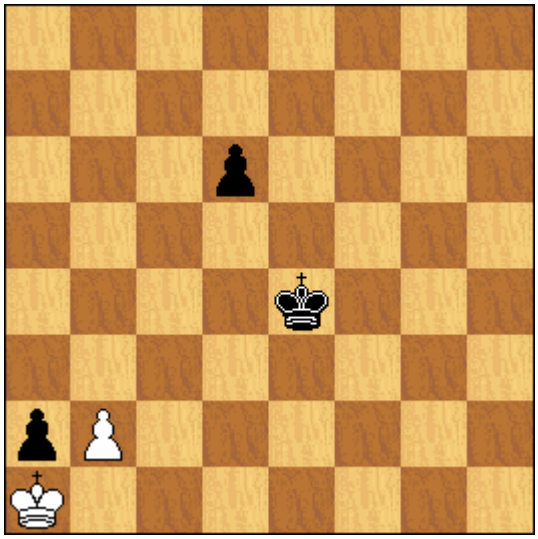
**Królewska blokada wolnego pionka przeciwnika** określa stan obrony wobec wolnych pionków przeciwnika, który polega na blokowaniu wolnego pionka przeciwnika. Król blokuje wolnego pionka przeciwnika *(*<*blokada krola*>=1.0), jeśli w tej samej kolumnie w odległości do dwóch wierszy przed pionkiem (po stronie, w którą porusza się pionek) znajduje się król przeciwnego koloru co pionek. Na Rys. 4.11. biały król blokuje wolnego pionka a2 przeciwnika. Cecha wyznaczana jest jako różnica wartości blokującego króla białych oraz czarnych. Zatem

$$\langle krolewska\ blokada\ wolnego\ pionka\ przeciwnika \rangle = \langle blokada\ krola \rangle_B - \langle blokada\ krola \rangle_C, \qquad (4.50)$$

gdzie $<blokada\ krola>_B$ oraz $<blokada\ krola>_C$ wyznacza wartość odpowiednio białego oraz czarnego króla blokującego.

Na Rys. 4.10. przedstawiono przykładową końcówkę, dla której wyznaczono wartości prezentowanych w tym rozdziale cech gry związanych z królami. Bardzo trudno jednoznacznie ocenić tę końcówkę. Można jedynie skoncentrować się na poszczególnych jej cechach. Król czarnych jest lepiej ustawiony – znajduje się w samym centrum, stąd w przypadku wielu różnych końcówek możliwe jest włączenie go do gry na dowolnym skrzydle. Biały król zajmują pozycję bezpieczną, która jest bardzo istotna w środkowej fazie gry, a w końcówce praktycznie jest bez znaczenia. Czarny król jest przewodnikiem swojego wolnego pionka, co w dużym stopniu sugeruje, przynajmniej w pionkówkach, iż czarny pionek już bez przeszkód będzie mógł dotrzeć do pola przemiany. Biały król blokuje natomiast jednego z wolnych pionków przeciwnika, zmniejszając tym samym

wartość pozycyjną tego pionka i umożliwiając w tym przypadku jego zdobycie w najbliższym posunięciu.



$$\langle bliskosc\ kroli\ do\ centrum \rangle = -1.0$$
$$\langle bezpieczenstwo\ kroli \rangle = 1.0$$
$$\langle krolowie\ przewodnicy\ pionow \rangle = -1.0$$
$$\langle krolewska\ blokada\ wolnego\ pionka \rangle = 1.0$$

Rys. 4.10. Przykładowa pozycja z wyznaczonymi wartościami cech gry związanych z królami

## 4.2. Opis systemu NeuralChess

Głównym zamierzeniem autora było stworzenie systemu, który zapewniłby eksperymentalne poszukiwanie optymalnej strategii gry w szachy, a jednocześnie pozwoliłby przeprowadzać serie eksperymentów mających na celu badanie efektywności stosowanego podejścia. Proces przeprowadzania eksperymentów całkowicie zautomatyzowano, tak aby sztuczny gracz (uczeń) mógł uczyć się strategii gry w automatycznej grze w szachy, umożliwiając przy tym autorowi dostrajanie wielu parametrów mających związek z samą nauką każdorazowo przed rozpoczęciem eksperymentu. Dodatkowo zrealizowano pomocnicze funkcjonalności umożliwiające testowanie wartości wyznaczanych cech gry oraz przeprowadzanie turnieju dla zaimplementowanych na sztywno jak i znalezionych w procesie nauki strategii gry. Efektem końcowym przeprowadzonych eksperymentów było określenie najlepszej znalezionej strategii gry i zaimplementowanie jej w aplikacji realizującej interaktywną grę w szachy z użytkownikiem.

Ostatecznie system NeuralChess podzielono na cztery oddzielne aplikacje:

- *Neural Chess – Eksperymenty* umożliwiającą:
  - skonfrontowanie w pojedynczej partii dowolnych zaimplementowanych strategii gry (ich opis znajduje się w kolejnym podrozdziale) w grze rozpoczynającej się od pozycji wyjściowej bądź losowej,
  - przeprowadzenie eksperymentu - treningu sieci neuronowej (w celu wyznaczenia funkcji oceniającej) w grze od pozycji wyjściowych z dowolną zaimplementowaną strategią gry na przestrzeni ustalonej liczby partii, z możliwością wyboru odpowiedniego zestawu wartości parametrów wpływających zarówno na efektywność nauki jak i jakość samej gry, oraz z możliwością zapisu kolejnych stanów sieci neuronowej wynikających z procesu uczenia do wybranego pliku,
  - bieżący wgląd w wyniki osiągane przez sieć neuronową podczas treningu,
  - bieżący wgląd na szachownicę, na której rozgrywana jest pojedyncza partia bądź jedna z gier treningowych;

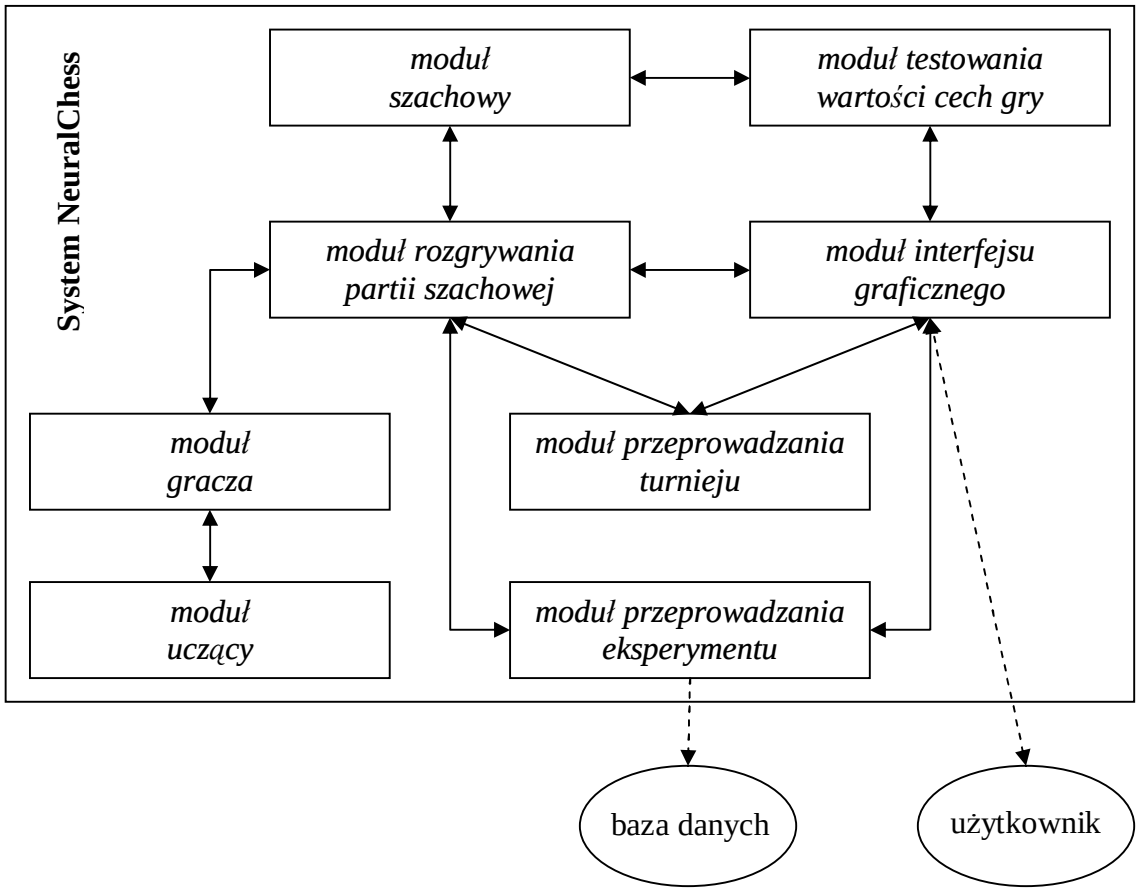
- *Neural Chess – Wartości Cech Gry* umożliwiająca:
  - wgląd w wartościowanie wszystkich zaproponowanych cech gry dla wybranej losowo bądź ręcznie pozycji szachowej;
- *Neural Chess – Turniej* umożliwiająca:
  - przeprowadzenie turnieju dla dowolnej liczby wybranych strategii w systemie każdy z każdym z określoną liczbą rewanżów,
  - bieżący wgląd w wyniki osiągane przez poszczególne strategie uczestniczące w turnieju,
  - bieżący wgląd na szachownicę, na której rozgrywana jest jedna z partii turniejowych;
- *Neural Chess* umożliwiająca:
  - rozegranie przez użytkownika gry w dowolnym kolorze z pozycji wyjściowej bądź losowej ze strategią reprezentowaną przez wyuczoną sieć neuronową, z możliwością ustawienia jej głębokości przeszukiwania drzewa gry,
  - cofanie i powtarzanie ruchów zagranych przez użytkownika.

Dokładny opis dotyczący obsługi wyżej wymienionych programów można znaleźć w dodatkach A-D. Działanie wszystkich powyższych aplikacji bazuje na następujących modułach (Rys. 4.11.):

- *moduł szachowy*, który jest odpowiedzialny za:
  - reprezentację bieżącej pozycji szachowej (rozmieszczenie figur),
  - utrzymanie poprawności wszystkich wykonywanych na szachownicy ruchów,
  - generowanie możliwych do wykonania ruchów przez wszystkie figury odpowiedniego koloru znajdujące się w bieżącej pozycji szachowej,
  - tworzenie historii gry umożliwiające ewentualne cofanie i powtarzanie ruchów,
  - wyznaczanie wartości autorskich cech gry,
- *moduł testowania wartości cech gry*, który w połączeniu z *modułem interfejsu graficznego* pozwala przedstawić użytkownikowi wyliczenia wszystkich autorskich cech gry z *modułu szachowego*;
- *moduł gracza*, który implementuje:
  - przeszukiwanie drzewa gry metodą cięć *Alpha-Beta* dla wszystkich strategii gry na dowolnie ustaloną maksymalną głębokość,
  - strategie gry (opisane dalej):
    - *losową*,
    - *uciekającego króla*,
    - *treningową*,
    - *TDL*, reprezentowaną przez sieć neuronową,
    - *użytkownika* – implementuje działania związane z przemieszczaniem figur na szachownicy wywoływanym przez użytkownika z poziomu *modułu interfejsu graficznego*,
- *moduł uczący* związany ze *strategią TDL modułu gracza*, który odpowiada za wykonywanie procedury uczącej *TD-Leaf*;
- *moduł rozgrywania partii szachowej*, który koordynuje grę pomiędzy dwoma strategiami z *modułu gracza* na szachownicy przechowywanej przez *moduł szachowy*,
- *moduł przeprowadzania eksperymentu*, którego zadaniem jest automatyczne przeprowadzenie eksperymentu uczenia *strategii TDL* dla ustalonych przez użytkownika parametrów, przy wykorzystaniu funkcjonalności *modułu rozgrywania partii szachowej* oraz ewentualnie komunikacji z *Bazą Danych*,

w której zebranych jest prawie pół miliona partii szachistów rozegranych na kafejce szachowej www.szachy.org w okresie od marca 2006 r. do maja 2007 r.

- *moduł przeprowadzania turnieju*, który umożliwia organizację turnieju dla dowolnej liczby strategii z *modułu gracza* przy wykorzystaniu funkcjonalności *modułu rozgrywania partii szachowej*,
- *moduł interfejsu graficznego użytkownika* odpowiedzialny za komunikację użytkownika z modułami *rozgrywania partii szachowej*, *przeprowadzania turnieju*, *przeprowadzania eksperymentu*, *testowania wartości cech gry*.




Rys. 4.11. Komunikacja pomiędzy modułami systemu NeuralChess

### 4.2.1. Strategie gry

W *module gracza* zaimplementowano cztery komputerowe strategie gry w szachy:

- *strategia losowa*,
- *strategia uciekającego króla*,
- *treningowa*,
- *TDL*.

**Strategia losowa** jest jedną z dwóch zdecydowanie najprostszych i najsłabszych strategii grających w szachy. Bazuje ona na całkowicie losowym wyborze ruchu spośród wszystkich możliwych ruchów w grze. Wygrywanie z taką strategią nie powinno stanowić problemu dla żadnej strategii ukierunkowanej na analizę pewnych cech gry. Strategia losowa, ze względu na szybkość podejmowanych decyzji, była przydatna przy wykonywaniu szybkich testów związanych z grą.

**Strategia uciekającego króla** jest kolejną bardzo trywialną strategią gry, która również bazuje na całkowicie losowym wyborze ruchu spośród wszystkich możliwych ruchów w grze, które utrzymują własnego króla jak najbliżej centrum. Wygrywanie z taką strategią również nie powinno stanowić większego problemu dla innych strategii zwracających uwagę na jakiekolwiek cechy gry. Jedyną trudnością może stać się zamatowanie króla tej strategii, który zawsze będzie chciał krążyć jak najbliżej centrum szachownicy. Podobnie jak strategia losowa, powyższa strategia, ze względu na szybkość podejmowanych decyzji, była przydatna do przeprowadzania szybkich testów.

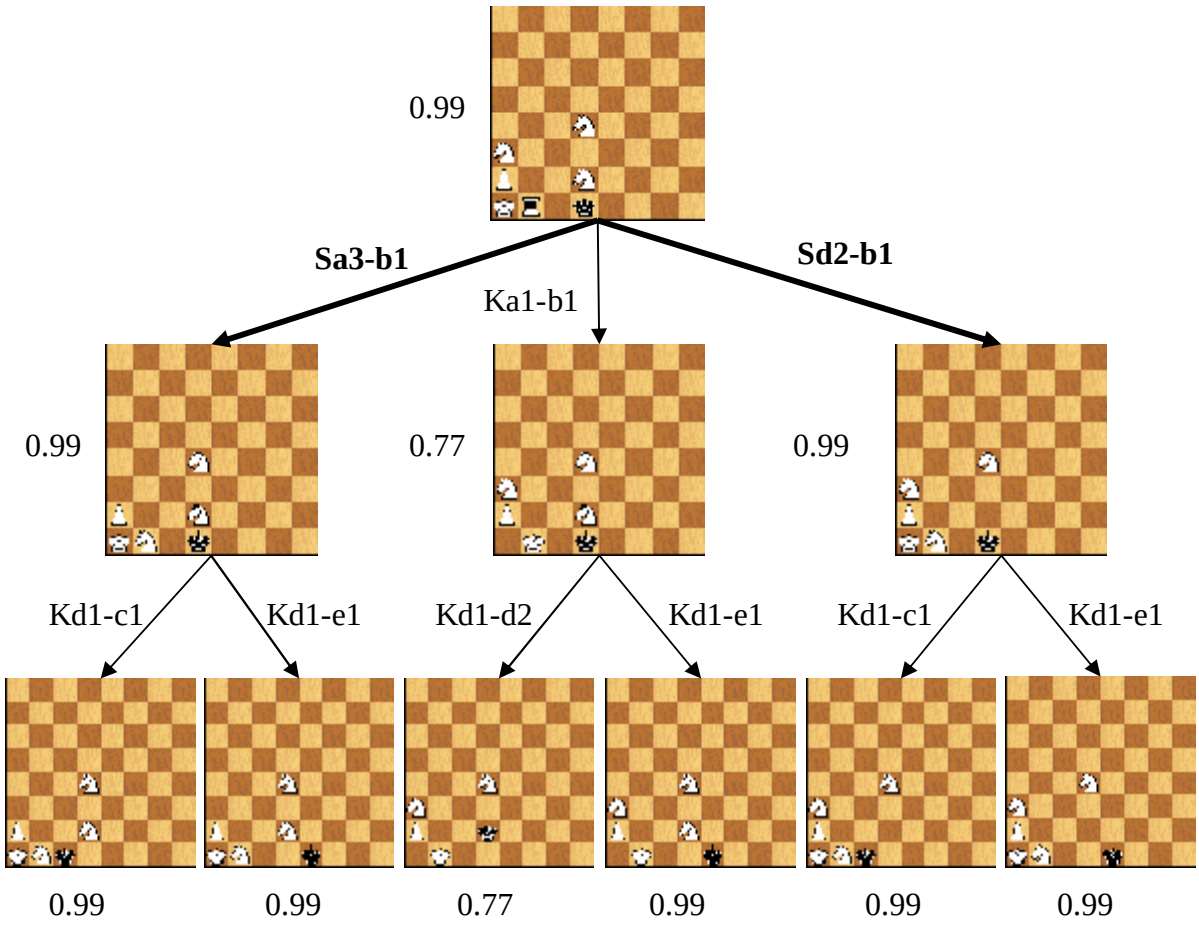
**Strategia treningowa** to już z całą pewnością o wiele lepsza strategia gry w porównaniu do dotychczas omówionych, choć wcale nie grająca na jakimś rewelacyjnym poziomie. Wyboru najlepszego ruchu dokonuje na podstawie analizy drzewa gry do ustalonej maksymalnej głębokości przy wykorzystaniu algorytmu cięć Alpha-Beta oraz zaimplementowanej na sztywno funkcji oceniającej stan gry:

$$\begin{aligned} f(x) &= 10 \cdot (H_B(x) - H_C(x)) + 5 \cdot (W_B(x) - W_C(x)) \\ &+ 3 \cdot (G_B(x) - G_C(x)) + 3 \cdot (W_B(x) - W_C(x)) + (P_B(x) - P_C(x)), \end{aligned} \tag{4.51}$$

gdzie $x$ oznacza pozycję szachową podlegającą ocenie, $H_B(x)$ oraz $H_C(x)$ – liczbę hetmanów odpowiednio koloru białego oraz czarnego znajdujących się w pozycji $x$, $W_B(x)$, $W_C(x)$, $G_B(x)$, $G_C(x)$, $S_B(x)$, $S_C(x)$, $P_B(x)$ oraz $P_C(x)$ – analogicznie liczbę odpowiednio wież, gońców, skoczków oraz pionków. Wyjątkiem od obliczania powyższej funkcji liniowej jest występowanie mata na szachownicy, wówczas $f(x)$=1000, jeśli białe zamatowały czarnego króla, lub $f(x)$=-1000, jeśli czarne zamatowały białego króla. Strategia treningowa jest zatem całkowicie elementarną strategią gry ukierunkowaną, tylko i wyłącznie, na ewentualne zdobycze materialne oraz możliwość matowania przeciwnika przy najbliższej sposobności.

**Strategia TDL** to strategia reprezentowana przez sieć neuronową pełniącą rolę funkcji oceniającej pozycje w grze, która może się jej uczyć przy pomocy algorytmu *Temporal Difference Learning* (a właściwie jej drobnej modyfikacji *TD-Leaf*). Najogólniej rzecz ujmując sieć neuronowa na podstawie informacji o wartościach omówionych wcześniej cech gry ocenia pozycje wynikające z przeszukiwania drzewa gry do ustalonej maksymalnej głębokości przy wykorzystaniu algorytmu cięć *Alpha-Beta* i na tej podstawie wybiera najlepszy możliwy dla siebie wariant gry. Algorytm cięć *Alpha-Beta* jest ulepszeniem algorytmu *Min-Max* o możliwość odcinania pewnych gałęzi drzewa, które na pewno nie będą miały już znaczenia w ocenie najlepszego możliwego wariantu. Działanie to potrafi znacznie skrócić czas przeszukiwań. Wyjściem sieci jest ocena podanej na wejście pozycji (a właściwie informacji o jej cechach). Ponieważ całkowite rozwinięcie drzewa dla gry takiej jak szachy jest praktycznie niewykonalne, więc jak już powiedziano ograniczono jego przeglądanie do pewnej maksymalnej głębokości, na której jego węzły podlegały ocenie przez sieć. Spośród węzłów o tej samej najlepszej ocenie (dla koloru białego największej, a dla czarnego – najmniejszej) w sposób losowy wybierany był jeden, dla którego wykonywany był ruch do niego prowadzący. Opisany sposób działania sieci oceniającej rozwinięte do pewnego poziomu drzewo gry przedstawiono na Rys. 4.12. W korzeniu drzewa gry znajduje się bieżąca pozycja, od której rozwijane są kolejne osiągalne pozycje wynikające z możliwych do wykonania ruchów. Węzłami drzewa są pozycje, a łukami – posunięcia prowadzące do kolejnych pozycji. Drzewo rozwijane jest w głąb. Za każdym razem, po osiągnięciu najgłębszego węzła, następuje jego ocena, która następnie propagowana jest do bezpośredniego jego poprzednika, w którym zapamiętywana jest uzyskana dotychczas najlepsza ocena gry z punktu widzenia danego koloru. Postępując w ten sposób analogicznie w górę do korzenia, otrzymywana jest ostatecznie jego ocena wynikająca z najlepszego

możliwego wariantu gry. Ta ocena będąca jednocześnie wynikiem działania sieci dla najlepszego znalezionego węzła stanowi prognozę związaną z ostatecznym wynikiem gry. Jak widać w prezentowanym przykładzie najlepiej ocenione najgłębsze węzły drzewa można osiągnąć w dwojaki sposób – zarówno po wykonaniu ruchu *Sa3-b1*, jak i *Sd2-b1*. Dlatego też w tej i podobnie pojawiających się sytuacjach przyjęto losowanie ruchu do wykonania z jednakowym prawdopodobieństwem dla każdego wyznaczonego.




Rys. 4.12. Przykład rozwiniętego drzewa gry do drugiego półruchu wraz z przykładową oceną najgłębiej położonych węzłów i wyborem ruchu w grze

## 4.3. Sieć neuronowa do oceny stanu gry

Mając już ogólny obraz działania systemu i idei wykorzystania sieci neuronowej posługującej się informacjami o cechach gry w celu oceny poszczególnych pozycji szachowych, przejdziemy do omówienia architektury zaprojektowanej sieci neuronowej oraz zasady działania algorytmu uczenia *Temporal Difference Learning*.

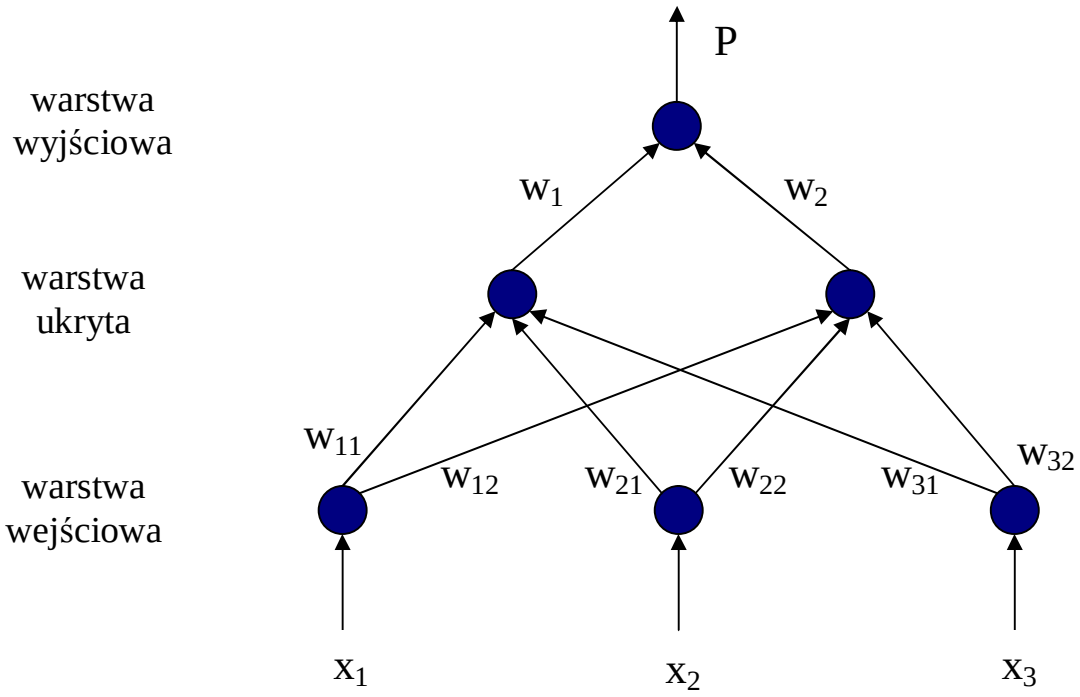
### 4.3.1. Architektura sieci neuronowej

Sieć neuronowa strategii TDL została zbudowana z trzech warstw: wejściowej, ukrytej i wyjściowej (Rys. 4.13.). Pomiędzy warstwami przyjęto stosowanie pełnych połączeń, tzn. każdy neuron niższej warstwy został połączony z każdym neuronem warstwy wyższej. W warstwie wejściowej sieci utworzono 45 neuronów, z których każdy otrzymywał na wejściu rzeczywisto-liczbową wartość odpowiedniej cechy gry zwykle z przedziału [-1,1]. Liczbę neuronów warstwy ukrytej ustalono jako średnią arytmetyczną liczby neuronów

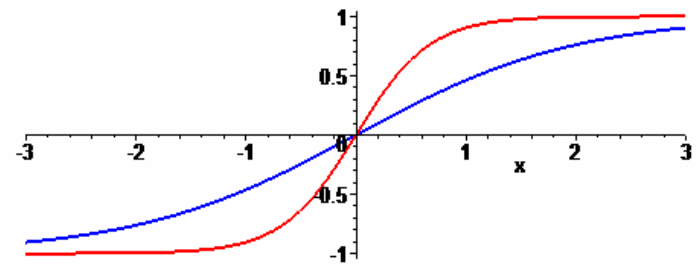
warstw wejściowej i wyjściowej, a więc 23 neurony. Otrzymano w ten sposób 1058 wszystkich połączeń pomiędzy neuronami. Każdy neuron warstwy ukrytej oraz wyjściowej aktywowany był przy wykorzystaniu funkcji sigmoidalnej $f$ przyjmującej wartości z przedziału (-1,1) (Rys. 4.14.):

$$f(x)=\frac{2}{1+e^{-bx}}-1. \tag{4.52}$$

W powyższym wzorze parametr $\beta$ odpowiedzialny jest za nachylenie sigmoidy i może przyjmować dowolną wartość rzeczywistą, z tymże im większa jest jego wartość tym bardziej stroma staje się sigmoida. W trakcie przeprowadzania eksperymentów dobierano różne wartości tego parametru. Wyjście sieci będące wartością z przedziału (-1,1) traktowano jako prognozę dotyczącą ostatecznego wyniku gry dla bieżącej pozycji szachowej. Ustalono, że wygrana białych oznacza 1.0, wygrana czarnych – -1.0, natomiast remis – 0.0 Ustalenie stanu początkowego sieci, tzn. wartości poszczególnych wag na połączeniach, zostało przedstawione przy okazji omawiania eksperymentów.




Rys. 4.13. Przykład trójwarstwowej sieci neuronowej z pełnymi połączeniami między warstwami, gdzie $x_i$ jest wejściem, $w_i$, $w_{ij}$ – wagą na połączeniu, oraz $P$ – predykcją sieci




Rys. 4.14. Wykres funkcji sigmoidalnej f(x)
z parametrem $\beta$=3.0 (kolor czerwony) oraz $\beta$=1.0 (kolor niebieski)

### 4.3.2. Uczenie sieci neuronowej metodą TD(λ)

Sieć neuronowa uczy się na podstawie dostarczanych jej wzorców, którymi w przypadku gry w szachy są pozycje szachowe. Najbardziej typowym algorytmem uczenia sieci jest algorytm *backpropagation* – propagacji wstecznej błędu, w którym aktualizacja stanu sieci (wszystkich wag na połączeniach) odbywa się każdorazowo po otrzymaniu wzorca. Wzorzec oceniany jest przez sieć neuronową, a następnie ocena ta porównywana jest z właściwą odpowiedzią, którą powinna dać sieć dla tego wzorca. Wynikający stąd błąd sieci propagowany jest wstecz (od wyjścia do wejścia) na wszystkie neurony, a następnie na jego podstawie dochodzi do modyfikacji wag. W przypadku gry w szachy zastosowanie tego algorytmu jest niemożliwe ze względu na brak właściwej odpowiedzi, którą powinna dać sieć, dla większości wzorców – pozycji szachowych. Właściwie tylko dla części wzorców – pozycje szachowe, w których zamatowany jest król – znana jest odpowiedź. Dlatego też zdecydowano się zastosować algorytm *Temporal Difference Learning* (*TD(λ)*), który wychodzi naprzeciw takim problemom. W odróżnieniu od algorytmu *backpropagation*, nie porównuje bieżącej predykcji sieci z właściwą odpowiedzią dla danego wzorca (której najczęściej nie ma), tylko porównuje kolejne predykcje, aż do momentu uzyskania wzorca, dla którego właściwa odpowiedź jest znana. Dlatego też w procesie uczenia ważna jest kolejność podawanych na wejście wzorców, a więc w przypadku szachów kolejnych pozycji wynikających z rozgrywania partii. Błąd wynikający z porównania następujących po sobie odpowiedzi sieci propagowany jest wstecz, podobnie jak w przypadku *backpropagation*. Stąd też w szczególnym przypadku – podawania wzorców ze znanymi odpowiedziami – algorytmy *temporal difference learning* oraz *backpropagtion* sprowadzają się do identycznego działania. Jednakże zaleta podejścia *TD(λ)* uwidacznia się w tzw. problemach wielokrokowych, którego przykładem jest analizowana gra w szachy. W problemach tych informacja o prawidłowości poszczególnych predykcji uzyskiwana jest po więcej niż jednym kroku, w których oceniane są podawane na wejście wzorce.

Działanie algorytmu *TD(λ)* zostanie przedstawione na nieco uproszczonym modelu sieci neuronowej (Rys. 4.15.) z ustalonymi początkowymi wartościami wag na połączeniach. Sieci neuronowej zostaną podane na wejście dwa wzorce (kolejno [1.0, 0.0] oraz [1.0, 1.0]), z których tylko dla drugiego będzie znana właściwa odpowiedź (0.0), jaką powinna ona dać. Krok po kroku przeanalizowany zostanie proces aktywacji sieci oraz aktualizacji jej stanu. Załóżmy więc, że na wejściu tej sieci pojawił się impuls $x_1$=1.0 i $x_2$=0.0 będący informacją o pierwszym wzorcu, który propagowany jest dalej przez aktywowane neurony warstwy wejściowej. Dzięki odpowiednim połączeniom pomiędzy warstwami, pobudzeniu podlegają wszystkie neurony warstwy ukrytej. Pobudzenie *i*-tego (licząc od lewej do prawej) neuronu warstwy ukrytej wyznaczane jest jako:

$$net_{Hi} = \sum_{j=0}^{N_I} x_j w_{ji}\,, \tag{4.53}$$

gdzie $N_I$ jest liczbą neuronów w warstwie wejściowej. Stąd w przykładzie mamy

$$\begin{aligned} net_{H1} &= x_1 w_{11} + x_2 w_{21} = 1.0\,, \\ net_{H2} &= x_1 w_{12} + x_2 w_{22} = 0.5\,. \end{aligned} \tag{4.54}$$

Pobudzone neurony podlegają następnie aktywacji poprzez wspomnianą wcześniej funkcję sigmoidalną (4.52):

$$f(net_{H1}) = 0.91, \quad f(net_{H2}) = 0.64. \tag{4.55}$$

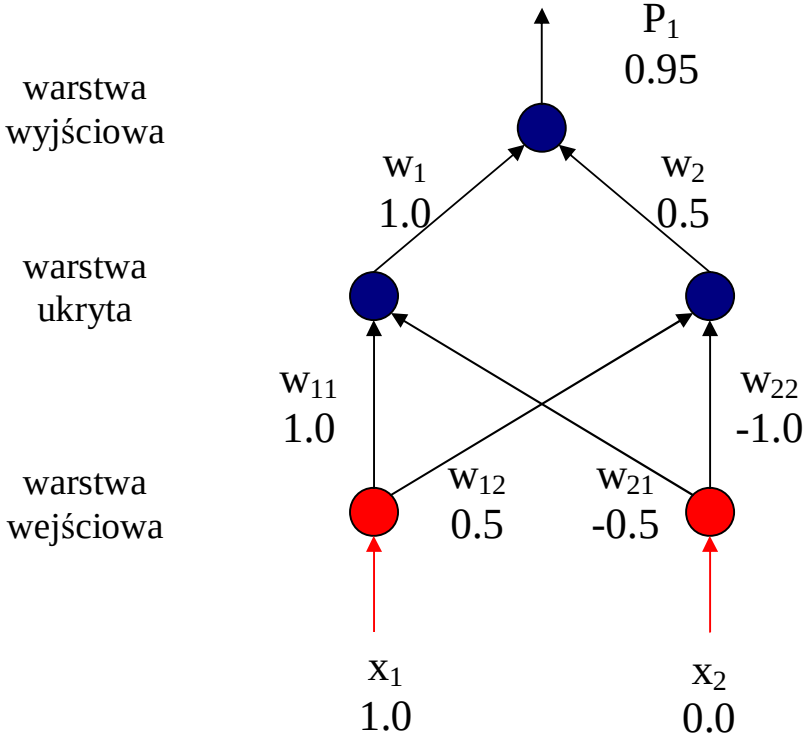
Aktywacja neuronów warstwy ukrytej, analogicznie, wpływa na pobudzenie neuronu warstwy wyjściowej:

$$net_O = \sum_{i=0}^{N_H} f(net_{Hi}) w_i , \tag{4.56}$$

gdzie $N_H$ jest liczbą neuronów w warstwie ukrytej. W ten sposób dochodzi do aktywacji neuronu warstwy wyjściowej:

$$net_O = f(net_{H1}) w_1 + f(net_{H2}) w_2 = 1.23, \quad f(net_O) = 0.95. \tag{4.57}$$

Powyższy wynik jest jednocześnie pierwszą odpowiedzią sieci (predykcją $P_1$) na zadane na wejściu wartości definiujące obserwowany wzorzec.




Rys. 4.15. Trójwarstwowa sieć neuronowa z ustalonymi przykładowymi wagami na połączeniach pobudzona impulsem wejściowym $x_1$=1.0 oraz $x_2$=0.0

Załóżmy teraz, iż dla obserwowanego wzorca nieznana jest właściwa odpowiedź, jaką powinna dać sieć, a na wejściu zaobserwowano kolejną sekwencję impulsów: $x_1$=1.0 i $x_2$=1.0 związaną z pojawieniem się drugiego wzorca, dla których w podobny sposób wyznaczono drugą predykcję sieci: $P_2$=0.44, po czym uzyskano właściwą odpowiedź, jaką powinna zwrócić sieć $P_3$=0.0 (umownie oznaczoną symbolem $P$ z najwyższym indeksem).

Na podstawie uzyskiwanych przez sieć predykcji oraz ostatecznego wzmocnienia, jakim było poznanie właściwiej odpowiedzi, wyliczane są w sposób przyrostowy pojawiające się zmiany wszystkich modyfikowalnych wag:

$$e_t = \nabla_w P_t + l\, e_{t-1}, \tag{4.58}$$

$$\Delta w_t = a\left(P_{t+1} - P_t\right) e_t, \tag{4.59}$$

$$w = w + \sum_{t=1}^{m} \Delta w_t\,, \tag{4.60}$$

gdzie $e_t$ ($e_0$=0) jest pomocniczym wyrażeniem (wektorem) wykorzystywanym do obliczania przyrostów wektora wag $Dw_t$, $l \in [0,1]$ jest parametrem wyznaczającym całą rodzinę metod uczenia TD(λ), który definiuje wartość wykładniczego sposobu ważenia przeszłych gradientów predykcji $\tilde{N}_w P_t$, $a \in (0,1]$ jest współczynnikiem uczenia wpływającym na dynamikę zmian wag, a $m$ wyznacza liczbę zaobserwowanych wzorców, po których otrzymano wzmocnienie. Aktualizacja wag (4.60) może występować na różnym poziomie przyrostowości:

- każdorazowo po uzyskaniu kolejnej predykcji sieci,
- każdorazowo po uzyskaniu ostatecznego wzmocnienia (tak też założono),
- każdorazowo po ustalonym dłuższym okresie czasu (np. po otrzymaniu kilku wzmocnień).

W pracy przyjęto konwencję aktualizacji wag po otrzymaniu wzmocnienia, a więc w przypadku szachów po uzyskaniu ostatecznego wyniku gry. Gradient $\tilde{N}_w P_t$ jest wektorem pochodnych cząstkowych funkcji $P$ względem wszystkich wag, spośród których wyróżnić można dwie klasy: pochodnej cząstkowej po dowolnej wadze na połączeniu między warstwą wejściową a ukrytą:

$$\frac{\partial P_t}{\partial w_{ji}} = f'\left(net_O\right) f'\left(net_{Hj}\right) w_i x_j\,, \tag{4.61}$$

a także pochodnej cząstkowej po dowolnej wadze na połączeniu między warstwą ukrytą a wyjściową:

$$\frac{\partial P_t}{\partial w_i} = f'\left(net_O\right) f\left(net_{Hi}\right), \tag{4.62}$$

gdzie

$$f'(x) = \frac{2be^{-bx}}{\left(1+e^{-bx}\right)^2}\,. \tag{4.63}$$

Wracając do prezentowanego przykładu, w którym znamy już sekwencję $P_1$, $P_2$, $P_3$, dla początkowego stanu sieci wyrażonego przez wektor wag (Rys. 4.15.):

$$w = \left[w_{11}, w_{12}, w_{21}, w_{22}, w_1, w_2\right] = [1.0, 0.5, -0.5, -1.0, 1.0, 0.5], \tag{4.64}$$

przyjmijmy $a$=0.6, $b$=3.0, $l$=0.8 i dokonajmy obliczeń zgodnie z (4.58) oraz (4.59):

$$\begin{aligned} e_1 &= [0.04, 0.07, 0.00, 0.00, 0.13, 0.09], \\ \Delta w_1 &= [-0.01, -0.02, 0.00, 0.00, -0.04, -0.03], \end{aligned} \tag{4.65}$$

$$\begin{aligned} e_2 &= [1.11, 0.59, 1.08, 0.54, 0.87, -0.69], \\ \Delta w_2 &= [-0.30, -0.16, -0.29, -0.14, -0.23, 0.19]. \end{aligned} \tag{4.66}$$

Powyższe wyliczenia pozwalają ostatecznie dokonać aktualizacji stanu sieci zgodnie z (4.60):

$$w = [0.69, 0.32, -0.79, -1.14, 0.73, 0.66]. \tag{4.67}$$

Nowy wektor wag wyznacza jednocześnie nowy stan sieci neuronowej wynikający z procesu uczenia. Okazuje się, że dla zmodyfikowanego w ten sposób stanu sieci odpowiednie jej predykcje dla tej samej sekwencji wejściowej (te same dwa wzorce podawane na wejście) wyglądałyby następująco: $P_1$=0.86 oraz $P_2$=-0.75. Natomiast po czterokrotnym zajściu opisywanego wyżej uczenia stan sieci wyglądałby następująco:

$$w = [0.84, 0.00, -0.46, -1.05, 0.04, 0.06], \tag{4.68}$$

co w rezultacie dla tej samej sekwencji wejściowej sprawiłoby, że predykcje sieci byłyby jeszcze bardziej zbliżone do właściwej odpowiedzi modelowanego systemu: $P_1$=0.06 oraz $P_2$=-0.05.

Powyższe rozważania dotyczące zasad działania algorytmu *TD(λ)* można podsumować następującym pseudokodem mającym miejsce podczas uczenia gry w szachy:

```
ustal parametry uczenia α, β, λ;
ustal stan początkowy sieci neuronowej;
foreach gra in zbiór gier
  t := 1 #krok algorytmu
  e0 := [0, 0, 0, …]
  foreach pozycja in gra
    Pt := ocena-sieci-neuronowej( pozycja )
    et := ∇wPt + λet-1
    if t > 1
      wt-1 := α(Pt-Pt-1)et-1
    endif
    t := t+1
  end
  Pt := wynik( gra )
  wt-1 := α(Pt-Pt-1)et-1
  aktualizacja-stanu-sieci-neuronowej( Σwt )
end
```

Rys. 4.16. Pseudokod algorytmu *TD(λ)* zastosowanego do uczenia sieci neuronowej gry w szachy

# 5. Badania eksperymentalne

Głównym celem niniejszej pracy było stworzenie systemu, w którym optymalna strategia gry w szachy byłaby poszukiwana przy użyciu sieci neuronowej (*strategia TDL*). Celem eksperymentów przedstawionych w niniejszym rozdziale była ocena efektywności zastosowanego podejścia. Początkowo jednak w sposób eksperymentalny dokonano doboru wartości parametrów. Ostatecznie wykonano 5 serii eksperymentów, których zadaniem było zbadanie:

- wpływu stanu początkowego sieci neuronowej na szybkość jej uczenia,
- wpływu wartości parametrów: $\alpha$ (współczynnik uczenia), $\beta$ (nachylenie funkcji sigmoidalnej), $\lambda$ (parametr algorytmu TD($\lambda$)) na efektywność uczenia,
- wpływu głębokości rozwijania drzewa gry na jakość gry oraz efektywność uczenia,
- wpływu sposobu trenowania sieci neuronowej na jakość gry,
- poziomu gry systemu z najlepiej wyuczoną siecią neuronową w konfrontacji z innymi programami szachowymi.

*Trenowanie sieci* przeprowadzano w sposób *aktywny* albo *bierny*.

**Trening aktywny** polega na rozgrywaniu przy użyciu *strategii TDL* ustalonej liczby gier kolorami na przemian ze *strategią treningową* i aktualizacji stanu sieci neuronowej po każdej rozegranej partii na podstawie wszystkich odpowiedzi sieci wynikających z oceny poszczególnych pozycji powstających podczas tej partii. Ocena pozycji wynika z analizy rozwijanego do ustalonej głębokości drzewa. Dla każdej ze strategii można ustalić maksymalną głębokość rozwijania drzewa gry obowiązującą w ciągu całego treningu. W treningu aktywnym sieć neuronowa uczy się grać w szachy w bezpośrednim starciu (biorąc aktywny udział w grze) ze *strategią treningową* grającą na ustalonym niezmiennym poziomie.

**Trening bierny** polega na uczeniu *strategii TDL* na podstawie obserwacji ustalonej liczby losowo wybranych gier ze zbioru prawie pół miliona partii rozegranych przez różnej klasy szachistów na internetowej kafejce szachowej www.szachy.org w okresie od 17.03.2006 do 15.05.2007. Uczenie na podstawie obserwacji gry polega na ocenianiu kolejnych wykonywanych przez szachistów posunięć i ostatecznym zaktualizowaniu stanu sieci neuronowej na podstawie wszystkich tych ocen po przejrzeniu całej partii. Ocena posunięć wynika z oceny pozycji po ich wykonaniu na podstawie analizy rozwijanego do ustalonej głębokości drzewa gry. W treningu biernym *strategia TDL* nie bierze czynnego udziału w grze z przeciwnikiem, tylko biernie przygląda się rozegranej niegdyś partii. Zastosowanie bazy danych partii rozgrywanych przez szachistów na różnym poziomie wydaje się być bardzo interesującym sposobem trenowania sieci, głównie ze względu na pouczający motyw wykorzystywania błędów przeciwnika. W takich partiach bardzo często popełniane są niedociągnięcia w rozgrywaniu odpowiednich faz gry. Z drugiej jednak strony błędy te wykorzystywane są przez przeciwników, którzy zmuszeni są do realizowania własnej przewagi do końca (zamatowania bądź osiągnięcia znacznej przewagi materialnej).

## 5.1. Wpływ stanu początkowego sieci neuronowej na szybkość jej uczenia

Głównym celem pierwszej serii eksperymentów było zbadanie wpływu stanu początkowego sieci neuronowej na szybkość jej uczenia. Spostrzeżenia wynikające z badań zostały wykorzystane w następnych eksperymentach.

**Przebieg**

Ze względu na dużą różnorodność cech gry, których wartości wpływają na ocenę poszczególnych pozycji przez sieć neuronową, przyjęto że korzystnie byłoby umiejscowić stan początkowy sieci w miejscu znacznie zbliżonym do strategii optymalnej, aby można było znacząco skrócić proces jej poszukiwania. W związku z tym podczas dostrajania stanu początkowego sieci największa uwaga skupiona została na elementarnych pojęciach szachowych związanych z wartością materialną figur oraz zamatowaniem przeciwnika. Cechy gry związane z tymi pojęciami to: *siła pionów*, *siła skoczków*, *siła gońców*, *siła wież*, *siła hetmanów* oraz *mat*. W niniejszym eksperymencie modyfikacji podlegały wagi połączeń wychodzących z neuronów z warstwy wejściowej oraz ukrytej. Badania zostały przeprowadzone z wykorzystaniem *treningu aktywnego*. Ustalone arbitralnie niezmienne wartości parametrów zebrano w Tab. 5.1. Przeprowadzono sześć eksperymentów dla różnie zdefiniowanych wartości wag. W Tab. 5.2. opisano przyjęte wartości wag na połączeniach między neuronami sieci neuronowej dla każdego z eksperymentów.

| **Parametr** | **Wartość** |
|---|---|
| Trening | aktywny |
| Liczba gier | 200 |
| Głębokość przeszukiwania drzewa gry przez obie strategie (*TDL* oraz *treningową*) | 2 |
| Współczynnik uczenia (α) | 0.5 |
| Nachylenie sigmoidy (β) | 0.5 |
| Parametr TD (λ) | 0.5 |

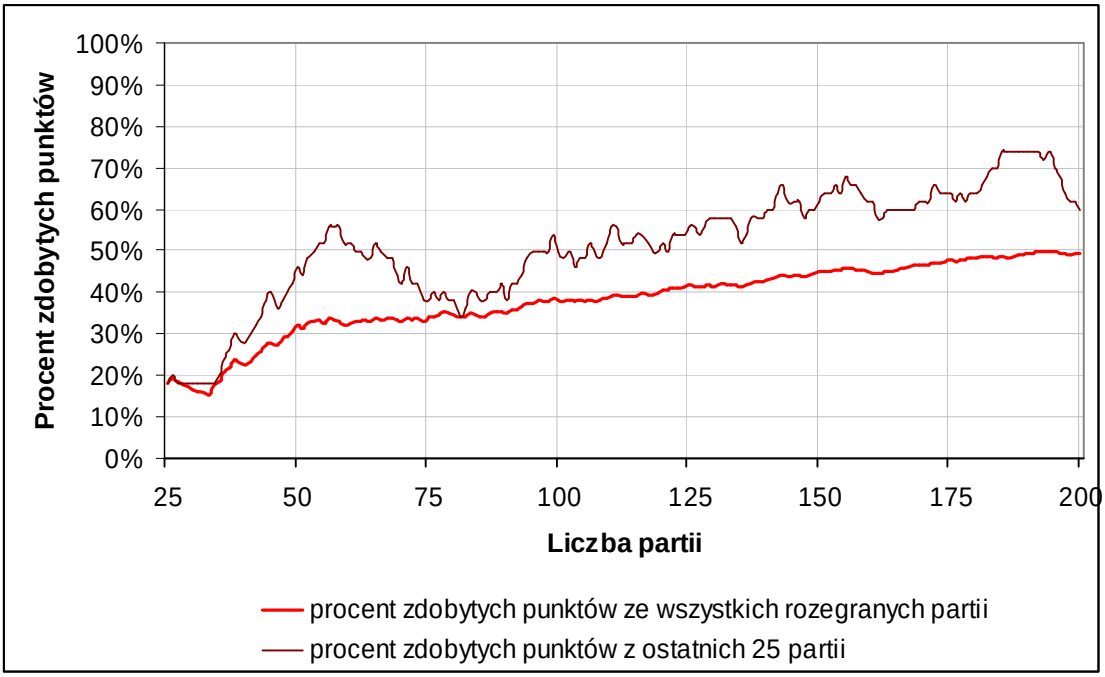
Tab. 5.1. Wartości parametrów pierwszej serii eksperymentów

| **Eksperyment** | **Ustawienia wartości wag** | **Wyniki** |
|---|---|---|
| I | wartości wag na wszystkich połączeniach ustalono na 0.01 | Rys. 5.1. |
| II | wartości wag na wszystkich połączeniach ustalono na 0.1 | Rys. 5.2. |
| III | wartości wag na wszystkich połączeniach losowano z jednakowym prawdopodobieństwem z przedziału [-0.01,0.01] | Rys. 5.3. |
| IV | wartości wag na wszystkich połączeniach losowano korzystając z rozkładu normalnego N(0.01,0.01) | Rys. 5.4. |
| V | wartości wag na połączeniach wychodzących z neuronów wejściowych otrzymujących informacje o wartościach cech *siła pionów*, *siła skoczków*, *siła gońców*, *siła wież*, *siła hetmanów* oraz *mat* ustalono odpowiednio na 0.1, 0.3, 0.3, 0.5, 1.0, 10.0 (starano się oddać w ten sposób ważność najistotniejszych cech gry – wartość materialną poszczególnych figur oraz zamatowanie króla przeciwnika); wartości wag na pozostałych połączeniach ustalono na 0.01 | Rys. 5.5. |
| VI | wartości wszystkich wag ustalono identycznie jak w eksperymencie V z tą drobną różnicą, iż do każdej z tych wartości dodano wartość losową z rozkładu normalnego N(0.00,0.01) | Rys. 5.6. |

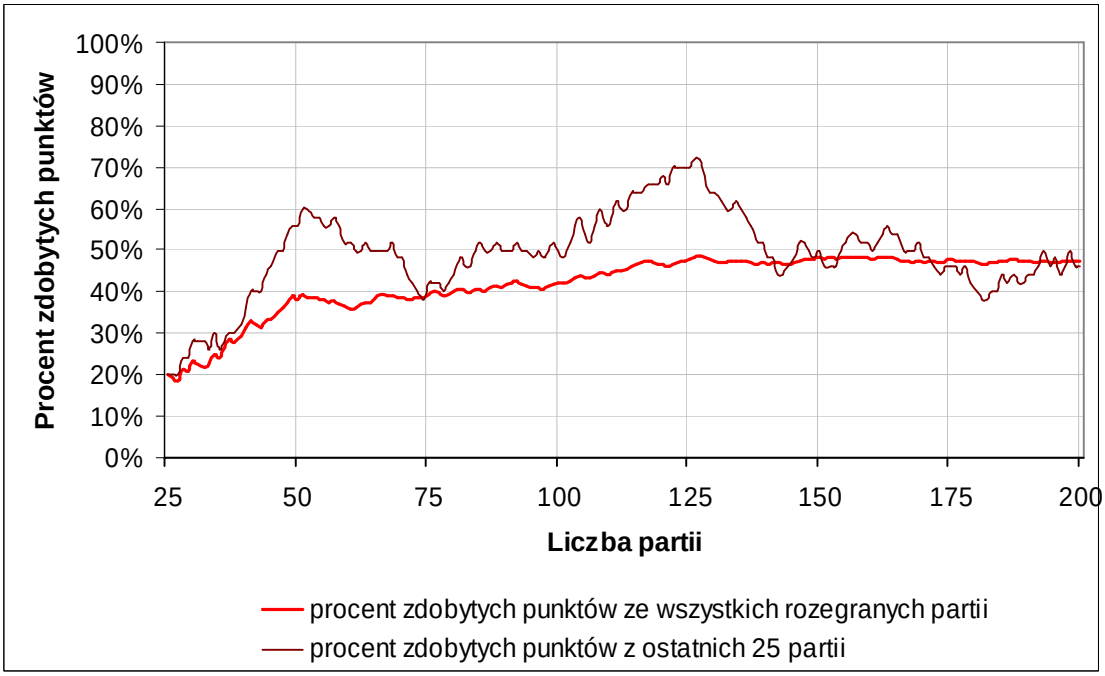
Tab. 5.2. Wartości wag na połączeniach ustalone w eksperymentach

**Wyniki**

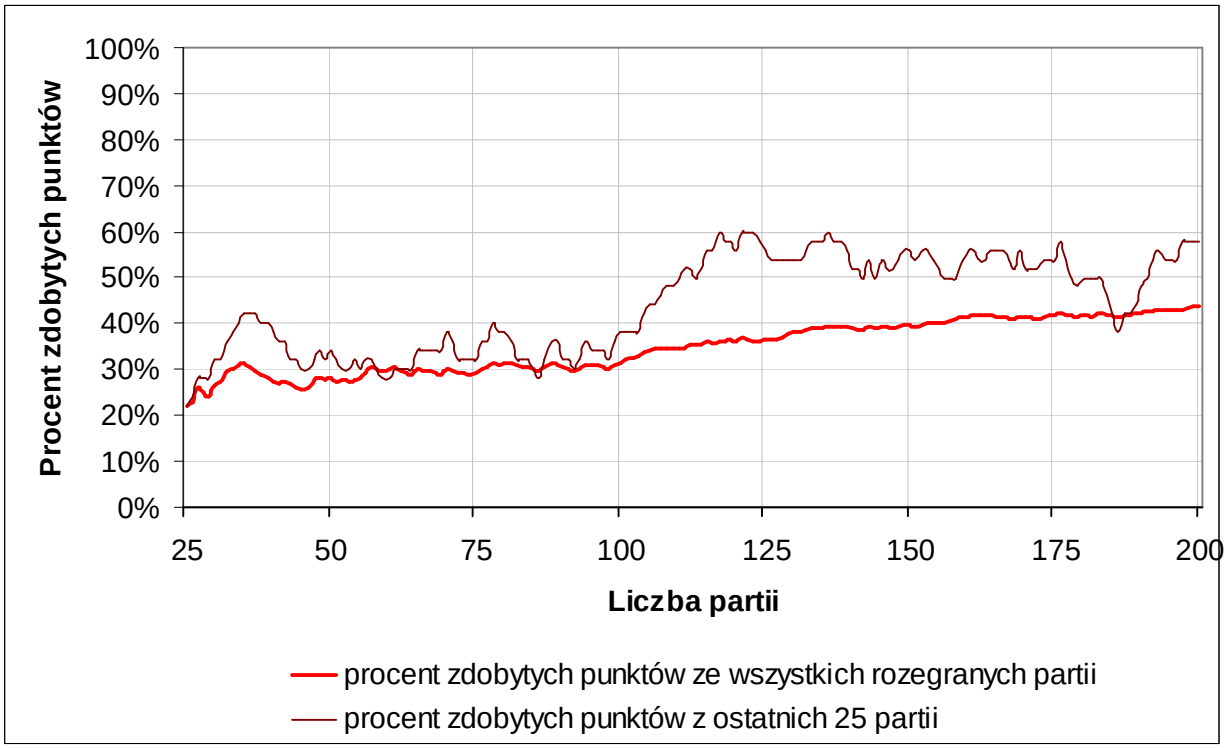
Poniżej zebrano wyniki sześciu przeprowadzonych eksperymentów (Rys. 5.1-6.), w których badano wpływ stanu początkowego sieci neuronowej na szybkość jej uczenia. Wyniki stanowią wykresy procentowego udziału zdobywanych punktów przez *strategię TDL* w grze ze *strategią terningową*.



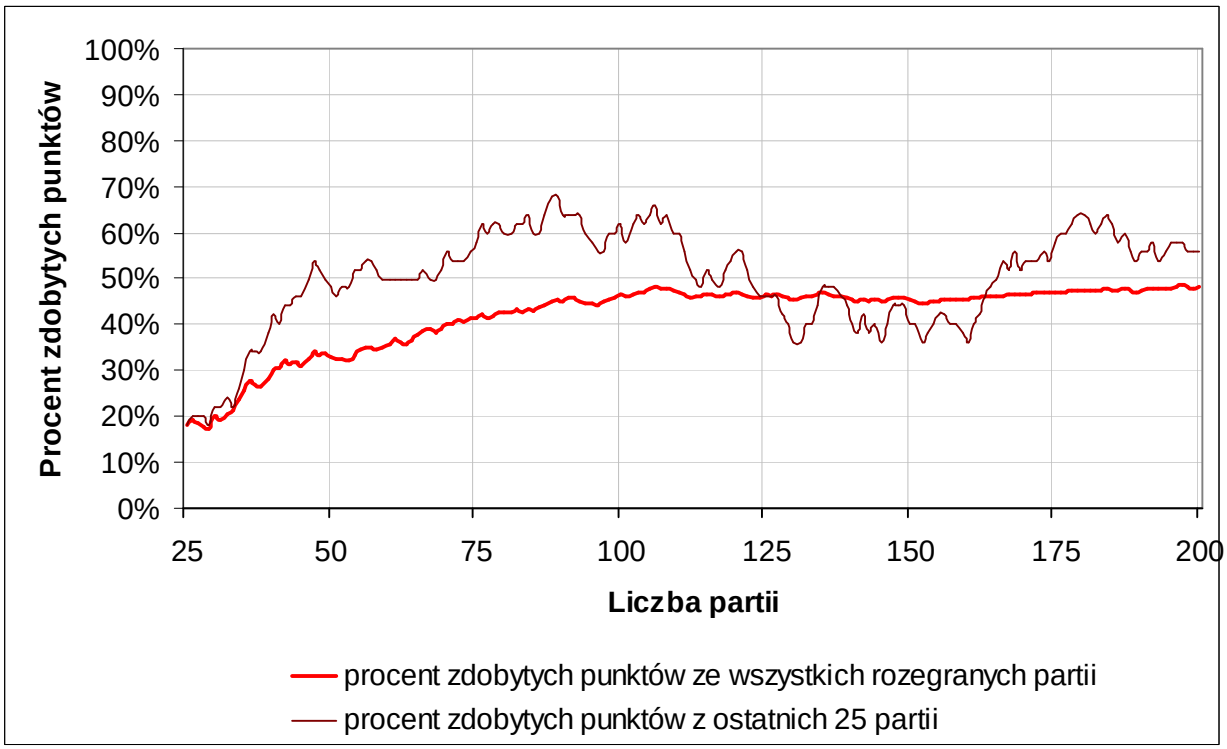

Rys. 5.1. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w I eksperymencie, którego początkowy zestaw wag opisano w Tab. 5.2.



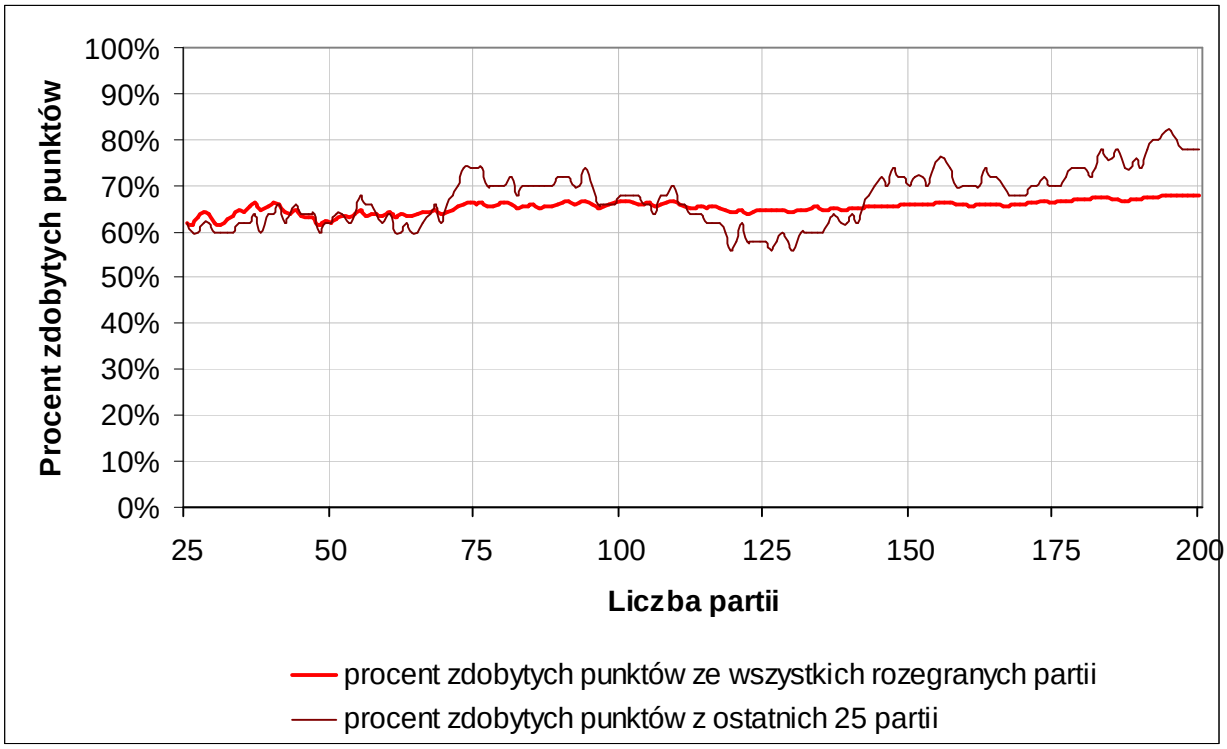

Rys. 5.2. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w II eksperymencie, którego początkowy zestaw wag opisano w Tab. 5.2.



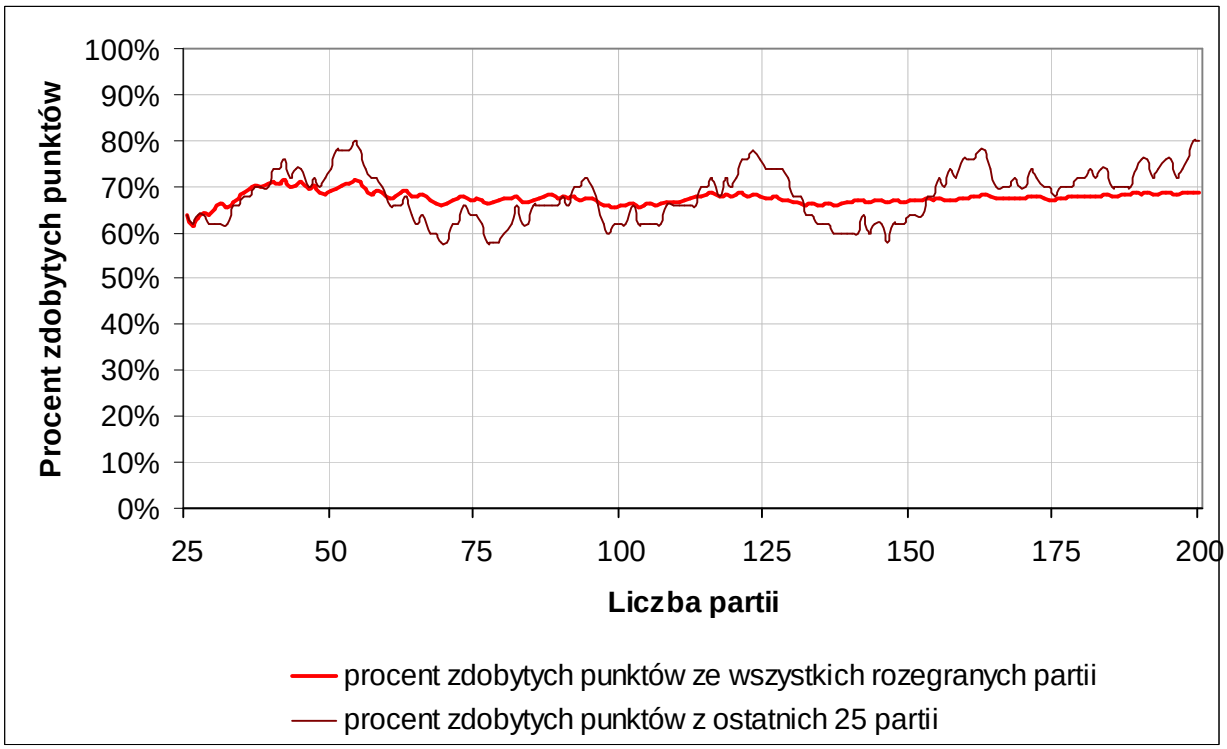

Rys. 5.3. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w III eksperymencie, którego początkowy zestaw wag opisano w Tab. 5.2.




Rys. 5.4. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w IV eksperymencie, którego początkowy zestaw wag opisano w Tab. 5.2.




Rys. 5.5. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w V eksperymencie, którego początkowy zestaw wag opisano w Tab. 5.2.




Rys. 5.6. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w VI eksperymencie, którego początkowy zestaw wag opisano w Tab. 5.2.

**Komentarz**

W pierwszym eksperymencie wartości wag dla wszystkich połączeń sieci neuronowej ustalono na stosunkowo niewielką liczbę bliską zeru (0.01), tak aby w chwili początkowej informacja o każdej cesze gry była jednakowo ważna. Zdecydowano się wybrać dodatnią wartość ze względu na charakterystykę zdefiniowanych cech gry, których dodatnia wartość wyraża ich występowanie na korzyść koloru białego, a w przeciwnym przypadku – koloru czarnego. Skalowanie dodatnią liczbą wszystkich informacji przepływających przez sieć nie zmieniało wspomnianej charakterystyki. Podczas przeprowadzania eksperymentu zaobserwowano, że dla każdego neuronu wejściowego wartości wag na połączeniach z niego wychodzących oraz dla neuronu wyjściowego wartości wag na połączeniach do niego dochodzących zmieniały się identycznie uzyskując za każdym razem tę samą wartość. Zjawisko to jest oczywistym następstwem założenia dotyczącego stanu początkowego sieci neuronowej, które wpływa na jednakową wartość wyznaczanej pochodnej cząstkowej dla wszystkich połączeń wychodzących z danego neuronu wejściowego (równanie (4.61)) oraz dla wszystkich połączeń dochodzących do neuronu wyjściowego (równanie (4.62)). Wyniki pierwszego eksperymentu okazały się być bardzo interesujące (Rys. 5.1.). W ciągu 200 rozegranych partii ze *strategią treningową* zauważalny był ciągły wzrost umiejętności gry *strategii TDL* – o czym świadczy trend wzrostowy wykresów. Z zaledwie 20% zdobytych punktów po pierwszych 25 partiach udało się sieci neuronowej nauczyć grać lepiej zdobywając ostatecznie niecałe 50% punktów ze wszystkich rozegranych partii i uzyskując nawet w najlepszych momentach prawie 75% zdobytych punktów z ostatnich 25 partii. Co ciekawe na koniec eksperymentu wartości wag na połączeniach wychodzących z neuronów wejściowych otrzymujących informacje o cechach *siła pionów*, *siła skoczków*, *siła gońców*, *siła wież* oraz *siła hetmanów* ustaliły się na wartościach odpowiednio 0.10, 0.28, 0.36, 0.46 oraz 0.66 – co w jednoznaczny sposób porządkuje wartości materialne figur zgodnie z logiką szachową. Cechy, dla których wartości wag na połączeniach wychodzących z warstwy wejściowej znacząco się uwydatniły to *opanowanie szachownicy* (0.46), *agresywność figur* (0.28), *otwarte wieże/hetmany* (0.19), *aktywność centralnych pionów* (0.18), *szach* (0.13), *mat* (0.13), *zdwojone pionki* (0.13) oraz *przewaga pionowa na skrzydłach* (0.12). Wartości pozostałych wag na połączeniach wychodzących z warstwy wejściowej ustaliły się w przedziale [-0.1,0.1]. Duże wzmocnienie cech *opanowania szachownicy* oraz *agresywności figur* wskazuje na ukierunkowanie strategii na agresywną grę, w której ważne jest szybkie opanowanie jak największej liczby pól szachownicy oraz atakowanie jak największej liczby figur przeciwnika. Wzmocnienie to może jednak okazać się zdradliwe głównie ze względu na porównywalną wartość wzmocnienia dotyczącą informacji o wartościach materialnych figur – oznacza to, że *strategia TDL* kosztem poświęcenia pewnej figury może starać się na przykład za wszelką cenę uzyskać lepsze opanowanie szachownicy, co niekoniecznie jest dobrym rozwiązaniem. Wartość materialna figur powinna być raczej brana pod uwagę przed wartością pozycyjną figur, co nie stanowi jednak żelaznej reguły. Dużym minusem końcowego stanu sieci neuronowej okazała się być wartość wagi związana z matowaniem (*mat*), która powinna być zdecydowanie najbardziej uwydatnioną cechą. Wyrazem tego stanu rzeczy była duża liczba remisów uzyskanych w wyniku pata, którego bezpośrednią przyczyną była na pewno ważna dla sieci informacja o opanowaniu szachownicy (w pozycji patowej strona mająca wykonać ruch opanowuje tylko tyle pól ile ma figur – jest to zatem minimalna liczba opanowanych pól dla danego zestawu figur).

W drugim eksperymencie postanowiono sprawdzić jak sieć neuronowa będzie się zachowywać w podobnej do poprzedniej sytuacji startując z wartościami wag dla wszystkich połączeń ustalonymi na nieco większą wartość 0.1. Oczywiście podobnie jak poprzednio dla każdego neuronu wejściowego wartości wag na połączeniach wychodzących (a także wartości

wag na połączeniach dochodzących do neuronu wyjściowego) zmieniały się jednakowo uzyskując za każdym razem tę samą wartość. Wyniki drugiego eksperymentu potwierdziły ukierunkowany wzrost siły gry *strategii TDL* (Rys. 5.2.) – trend wzrostowy wykresów. Chociaż wyniki w ogólności okazały się nieco słabsze – 47.5% zdobytych punktów w ciągu całego eksperymentu – to jednak tendencja uzyskiwanych wyników jest analogiczna. Rozpoczynając od 20% zdobytych punktów z pierwszych 25 partii, *strategia TDL* uzyskała ostatecznie niecałe 50% wszystkich możliwych punktów, osiągając miejscami wynik 70% zdobytych punktów z ostatnich 25 partii. Smucić może jedynie ostatnie 50 partii eksperymentu, w których nie doszło już do znacznych zmian w udziale zdobywanych punktów – poziom gry *strategii TDL* utrzymywał się na stałym poziomie. Wartości wag na połączeniach wychodzących z neuronów wejściowych otrzymujących informacje o cechach *siła pionów*, *siła skoczków*, *siła gońców*, *siła wież* oraz *siła hetmanów* ustaliły się na koniec treningu na wartościach odpowiednio 0.15, 0.16, 0.40, 0.47 oraz 0.78, co ponownie potwierdziło jednoznaczny sposób porządkowania wartości materialnych figur szachowych. Co prawda informacja o *sile pionków* stała się jednakowo ważna co informacja o *sile skoczków*, a informacja o *sile gońców* prawie tak samo ważna jak informacja o *sile wież*, ale samo uporządkowanie wartości tych cech zgodnie z odpowiadającą im ważnością wydaje się być po raz kolejny miłym doświadczeniem. Zadziwiający stał się przy tym fakt, iż pomimo całkowicie innego ustawienia stanu początkowego sieci (wartości wag na połączeniach ustalono na wartości dziesięciokrotnie większe od tych z pierwszego eksperymentu) wartości wag na połączeniach związanych ze wspomnianymi cechami oscylowały na koniec eksperymentu w bliskim sąsiedztwie wartości. Potwierdzeniem tego faktu stało się również uwydatnienie podobnego zestawu cech: *opanowanie szachownicy* (0.46), *aktywność centralnych pionów* (0.27), *agresywność figur* (0.23), *mat* (0.16), *szach* (0.15), *połączone wolne pionki* (0.14), *izolowane pionki* (0.13), *otwarte wieże/hetmany* (0.13), *przestrzeń figur* (0.11), *zdwojone pionki* (0.11). Wartości pozostałych wag na połączeniach wychodzących z warstwy wejściowej ustaliły się w przedziale [-0.1,0.1]. W tym przypadku można by posłużyć się niemalże identycznym komentarzem odnoszącym się do powyższych wyników tak jak to miało miejsce w pierwszym eksperymencie.

W trzecim eksperymencie postanowiono urozmaicić nieco stan sieci wprowadzając element losowości w doborze stanu początkowego sieci neuronowej – wartości wag losowano z jednakowym prawdopodobieństwem z przedziału niewielkich wartości bliskich zeru [-0.01,0.01]. Wprowadzenie elementu losowości miało zapewnić wyeliminowanie opisywanego w dwóch poprzednich eksperymentach zjawiska jednakowych zmian wartości wag na poszczególnych połączeniach. Jak się później okazało stan końcowy sieci neuronowej (po eksperymencie) znacznie odbiegał od stanów końcowych sieci neuronowych z dwóch poprzednich eksperymentów. Chociaż uzyskano wyniki zdecydowanie najsłabsze (Rys. 5.3.) – ok. 44% zdobytych punktów w ciągu całego eksperymentu – to jednak po raz kolejny zauważalny był wzrostowy trend uzyskiwanych wyników. Ponownie rozpoczynając od 20% zdobytych punktów z pierwszych 25 partii, *strategia TDL* uzyskała ostatecznie niecałe 45% punktów, osiągając miejscami wynik 60% zdobytych punktów z ostatnich 25 partii. Losowy dobór wartości wag na połączeniach sieci sprawił, że mniej więcej z jedną połową połączeń związane były ostatecznie ujemne wartości a z drugą – dodatnie. W poprzednich przypadkach wartości wag na większości połączeń sieci ustalała się na wartościach dodatnich. Różnego znaku wartości wag na połączeniach wychodzących z warstwy ukrytej nie pozwalają jednoznacznie ocenić wartości wag na pozostałych połączeniach, ponieważ bardzo trudno jest w tym przypadku przeanalizować w jakim stopniu dana informacja wpływa na zwiększenie aktywacji neuronu wyjściowego. Można jedynie zauważyć pewną tendencję do wzmacniania ważności informacji o pewnych cechach, dla których wartości wag na połączeniach wychodzących z neuronu wejściowego oscylują

w większych przedziałach wartości: *siła pionów* [-0.08,0.07], *siła skoczków* [-0.69,0.62], *siła gońców* [-0.62,0.56], *siła wież* [-1.02,0.92], *siła hetmanów* [-1.30,1.16], *szach* [-0.2,0.18], *mat* [-0.2,0.19], *agresywność figur* [-0.52,0.46], *otwarte wieże/hetmany* [-0.52,0.46], *wzmocnienie ciężkich figur w kolumnie* [-0.13,0.11], *opanowanie szachownicy* [-1.01,0.89], *aktywność centralnych pionów* [-0.62,0.54], *połączone wolne pionki* [-0.30,0.27], *oddalone wolne pionki* [-0.22,0.24], *kandydaci na wolne pionki* [-0.26,0.29], *bliskość króli do centrum* [-0.16,0.18]. Pozostałe wartości wag na połączeniach wychodzących z warstwy wejściowej ustaliła się co najwyżej w przedziałach [-0.1,0.1]. Co ciekawe po raz kolejny cechy takie jak *opanowanie szachownicy*, *aktywność centralnych pionów*, *agresywność figur*, *mat*, *szach* oraz *otwarte wieże/hetmany* zdecydowanie zyskują na znaczeniu.

W czwartym eksperymencie utrzymano urozmaicony stan sieci zakładając tym razem losowanie wartości wag dla wszystkich połączeń sieci przy wykorzystaniu rozkładu normalnego N(0.01,0.01). Ze względu na najlepsze wyniki pierwszego eksperymentu zdecydowano losować wartości wokół punktu 0.01. Podobnie jak poprzednio w stanie końcowym sieci neuronowej mniej więcej połowa wartości wag na połączeniach była znaku dodatniego, a druga – ujemnego, aczkolwiek wartości wag z dodatnim znakiem osiągały większe wartości bezwzględne. Tym razem uzyskano wyniki nieco słabsze od pierwszego eksperymentu (Rys. 5.4.), głównie ze względu na postój w nauce pomiędzy 110. i 160. partią, gdzie *strategia TDL* utrzymywała podobny udział zdobytych punktów z ostatnich 25 partii. Niemniej jednak po raz kolejny rozpoczynając od 20% zdobytych punktów z pierwszych 25 partii, *strategia TDL* uzyskała ostatecznie niecałe 50% wszystkich możliwych punktów zdobywając miejscami ponad 60% punktów z ostatnich 25 partii. Najciekawsze wartości wag na połączeniach wychodzących z neuronów wejściowych pojawiły się przy cechach: *siła pionów* [-0.02,0.05], *siła skoczków* [-0.04,0.51], *siła gońców* [-0.07,0.90], *siła wież* [-0.05, 0.76], *siła hetmanów* [-0.09, 1.31], *szach* [0.00,0.21], *mat* [-0.01,0.23], *agresywność figur* [0.00,036], *otwarte wieże/hetmany* [-0.03,0.44], *opanowanie szachownicy* [-0.08, 0.97], *aktywność centralnych pionów* [-0.01,0.29], *oddalone wolne pionki* [-0.28,0.02]. Pozostałe wartości wag na połączeniach wychodzących z warstwy wejściowej ustaliła się co najwyżej w przedziałach [-0.1,0.1].

W piątym eksperymencie postanowiono wreszcie znacznie przybliżyć stan sieci neuronowej do stanu reprezentującego pewną strategią optymalną gry w szachy. Wartości wag na połączeniach wychodzących z neuronów wejściowych otrzymujących informacje o wartościach cech *siła pionów*, *siła skoczków*, *siła gońców*, *siła wież*, *siła hetmanów* oraz *mat* ustalono odpowiednio na 0.1, 0.3, 0.3, 0.5, 1.0 oraz 10.0. Głównym zamierzeniem tego podejścia było wzmocnienie i uporządkowanie ważności informacji dotyczących wartości materialnych figur oraz matowania króla przeciwnika. Sposób w jaki wyznaczono wartości wag dla cech związanych z siłami figur odpowiada sposobowi w jaki definiowana jest wartość materialna figur (opisana w rozdziale 2.). Wartości wag na pozostałych połączeniach ustalono na stałą wartość 0.01. W wyniku powyższych zabiegów uzyskane wyniki eksperymentu znacząco odbiegają od wyników poprzednich eksperymentów. *Strategia TDL* już po rozegraniu 25 pierwszych partii uzyskała nieco powyżej 60% punktów. Dalszy trening pozwolił jeszcze nieco poprawić ten rezultat poprzez zdobycie w ostatecznym rozrachunku 68% punktów, utrzymując przy tym w ostatnich pięćdziesięciu partiach wynik ponad 70% (dochodząc nawet do 80%) zdobywanych punktów z ostatnich 25 partii. Na koniec eksperymentu wartości wag na połączeniach wychodzących z neuronów wejściowych otrzymujących informacje o cechach *siła pionów*, *siła skoczków*, *siła gońców*, *siła wież* oraz *siła hetmanów* ustaliły się na wartościach odpowiednio 0.16, 0.46, 0.53, 0.60 oraz 1.01 – co nadal w jednoznaczny sposób porządkowało wartości materialne figur zgodnie z intuicją szachową. Cechy, które znacząco się uwydatniły to *kolor* (0.45), *agresywność figur* (0.20),

*otwarte wieże/hetmany* (0.29), *opanowanie szachownicy* (0.31) oraz *połączone wolne pionki* (0.15). Pozostałe wartości wag na połączeniach wychodzących z neuronów wejściowych oscylowały w przedziale wartości [-0.1,0.1].

W ostatnim szóstym już eksperymencie zdecydowano się połączyć dwie własności stanu początkowego sieci neuronowej: różnorodność oraz bliskość do stanu reprezentującego pewną strategię optymalną gry w szachy. W związku z tym wartości wag na wszystkich połączeniach ustalono identycznie jak w eksperymencie piątym dodając jedynie do każdej z nich wartość losową z rozkładu normalnego N(0.00,0.01). Wyniki tego eksperymentu potwierdziły jedynie wnioski płynące z wyników poprzedniego eksperymentu. Odpowiednie dostrojenie stanu początkowego sieci neuronowej znacząco wpłynęło na grę *strategii TDL* już od początku treningu – ponad 60% punktów zdobytych w pierwszych 25 partiach. Kolejne partie rozgrywane były już właściwie na niezmiennym poziomie, w którym *strategia TDL* zdobyła w grze ze *strategią treningową* około 70% punktów ze wszystkich rozegranych partii. Na koniec eksperymentu wartości wag na połączeniach wychodzących z neuronów wejściowych otrzymujących informacje o cechach *siła pionów*, *siła skoczków*, *siła gońców*, *siła wież* oraz *siła hetmanów* ustaliły się na w przedziałach odpowiednio [0.12,0.15], [0.44,0.49], [0.45,0.49], [0.74,0.81], [1.00,1.04]. Jak widać wartości te w procesie nauki zostały powiększone – w trzech przypadkach nawet o 60%. Cechy, które znacząco się uwydatniły to *kolor* [0.36,0.45], *agresywność figur* [0.20,0.26], *otwarte wieże/hetmany* [0.10,0.15] oraz *opanowanie szachownicy* [0.39,0.47]. Wartości pozostałych wag na połączeniach wychodzących z neuronów wejściowych nie wykraczały poza przedział [-0.1,0.1].

W wyniku przeprowadzonej serii eksperymentów dokonano dwóch znaczących obserwacji. Po pierwsze w każdym eksperymencie, a w szczególności w pierwszych czterech, zauważalny był ciągły wzrost umiejętności gry *strategii TDL* reprezentowanej przez trenowaną sieć neuronową. W związku z tym w pełni uzasadnione staje się stosowanie metody *temporal difference learning* (z jej drobną modyfikacja *TDLeaf*) do poszukiwania optymalnej strategii gry. Po drugie odpowiednie dostrojenie stanu początkowego sieci neuronowej znamiennie wpływało na proces jej nauki. Poprzez zwrócenie należytej uwagi na najważniejsze cechy gry, *strategia TDL* potrafiła już od początku zdecydowanie lepiej radzić sobie ze swoim przeciwnikiem – *strategią treningową*. Co ciekawe w każdym z eksperymentów stan końcowy sieci neuronowej za każdym razem wskazywał na wzmacnianie znaczenia takich cech gry jak *opanowanie szachownicy*, *agresywność figur*, *otwarte wieże/hetmany* oraz *aktywność centralnych pionów*. Są to cechy, na które warto zwracać uwagę szczególnie w początkowej oraz środkowej fazie gry. Najprawdopodobniej ich występowanie w grze ze *strategią treningową* w sposób istotny przyczyniało się do zwycięstw *strategii TDL*.

## 5.2. Wpływ wartości parametrów α, β oraz λ na efektywność uczenia

Głównym celem drugiej serii eksperymentów było zbadanie wpływu wartości parametrów α (współczynnik uczenia), β (nachylenie funkcji sigmoidalnej) oraz λ (parametr metody *TD*) na efektywność uczenia sieci neuronowej. Spostrzeżenia poczynione podczas przeprowadzania eksperymentów zostały wykorzystane w następnych eksperymentach.

### Przebieg

Odpowiedni dobór parametrów wpływających na proces uczenia sieci neuronowej jest jedną z najbardziej istotnych spraw. Spośród wszystkich możliwych wartości – $\alpha \in (0,1)$, $\beta > 0$, $\lambda \in [0,1]$ – należy dobrać taki zestaw wartości tych parametrów, aby proces uczenia sieci przebiegał prawidłowo, tzn. bez większych skoków po przestrzeni wszystkich możliwych rozwiązań. Optymalne dostrojenie wartości badanych parametrów nie jest rzeczą prostą

i zależy ono w dużej mierze od stanu początkowego sieci neuronowej. W przypadku całkowicie losowej inicjalizacji stanu sieci najprawdopodobniej najbardziej pożądany byłby zestaw takich wartości parametrów, który wpływałby na większą dynamikę zmian stanu sieci neuronowej. W eksperymentach przeprowadzonych w tej części początkowy stan sieci neuronowej ustalany był zgodnie z założeniami poczynionymi w ostatnim eksperymencie poprzedniej serii, tzn. wartości wag na połączeniach wychodzących z neuronów wejściowych otrzymujących informację o cechach *siła pionów*, *siła skoczków*, *siła gońców*, *siła wież*, *siła hetmanów* oraz *mat* ustalono odpowiednio na wartości 0.1, 0.3, 0.3, 0.5, 1.0 oraz 10.0, natomiast pozostałe wartości wag ustalono na wartości 0.01 – dodatkowo do każdej wagi dodawano wartość losową z rozkładu normalnego N(0.00,0.01). W związku z powyższym założeniem dotyczącym stanu początkowego sieci neuronowej poszukiwany zestaw wartości parametrów związanych z uczeniem nie powinien zbyt dynamicznie wpływać na zakres zmian stanu sieci. Dlatego też w niniejszej serii eksperymentów modyfikacji podlegały trzy parametry i zdecydowano się wybrać następujący zestaw badanych wartości (Tab. 5.3.).

| **Parametr** | **Wartości** | | |
|---|---|---|---|
| α (współczynnik uczenia) | 0.1 | 0.5 | 0.9 |
| β (nachylenie sigmoidy) | 0.1 | 0.5 | 0.9 |
| λ (parametr metody *TD*) | 0.1 | 0.5 | 0.9 |

Tab. 5.3. Wartości parametrów dotyczących uczenia sieci neuronowej

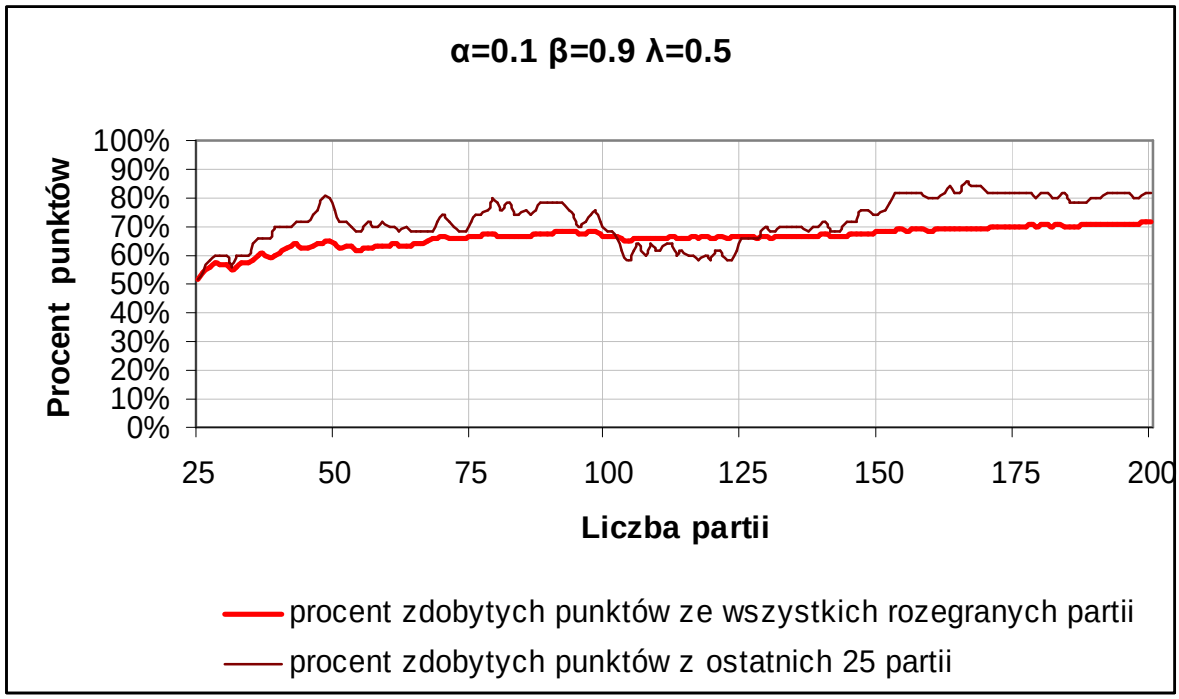
Dla każdego parametru wybrano po trzy wartości do zbadania. W niniejszej serii przeprowadzono 27 eksperymentów – w każdym testowano jedną z możliwych kombinacji. Wartości parametrów ustalono dosyć arbitralnie, zwracając jednak uwagę na rozłożenie ich w dziedzinie. Dla parametrów α oraz λ są to wartości z środka oraz z obu krańców dopuszczalnych dziedzin wartości. W przypadku parametru β sytuacja była podobna, gdyż ostatecznie założono, że jego wartość nie powinna przekraczać 1.0. W Tab. 5.4. zestawiono ustalone wartości dodatkowych parametrów dla wszystkich 27 eksperymentów.

| **Parametr** | **Wartość** |
|---|---|
| Trening | aktywny |
| Liczba gier | 200 |
| Głębokość przeszukiwania drzewa gry przez obie strategie (*TDL* oraz *treningową*) | 2 |

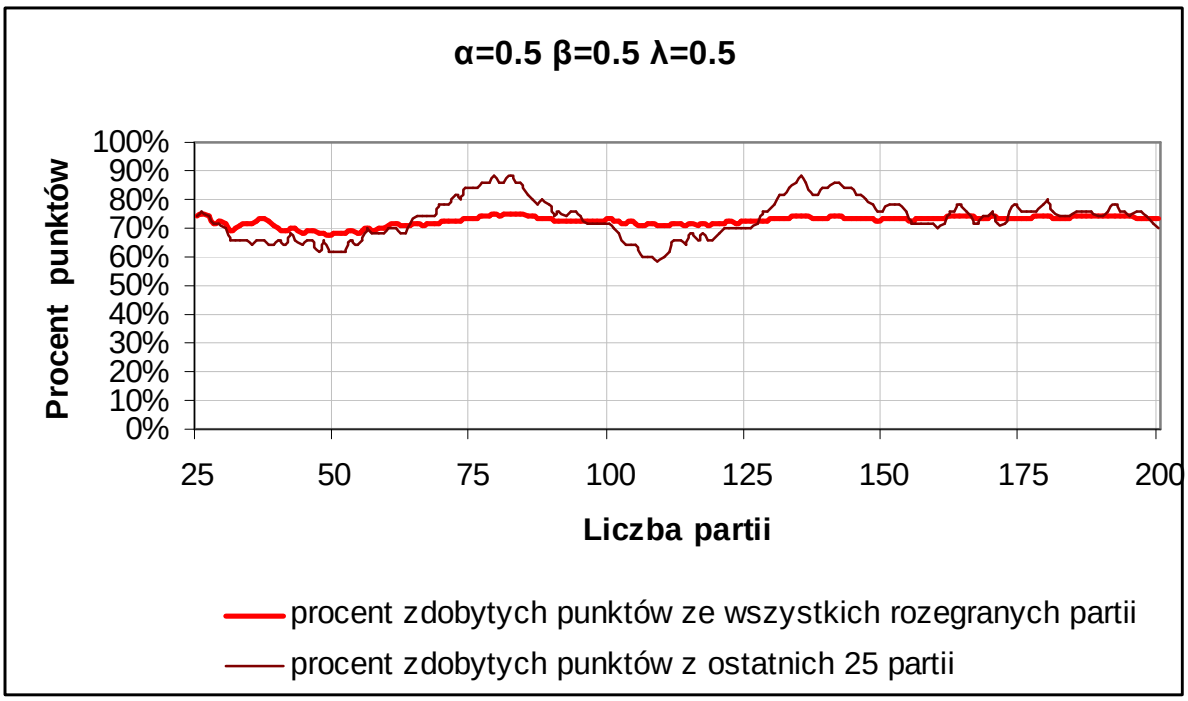
Tab. 5.4. Wartości parametrów całej serii eksperymentów

**Wyniki**

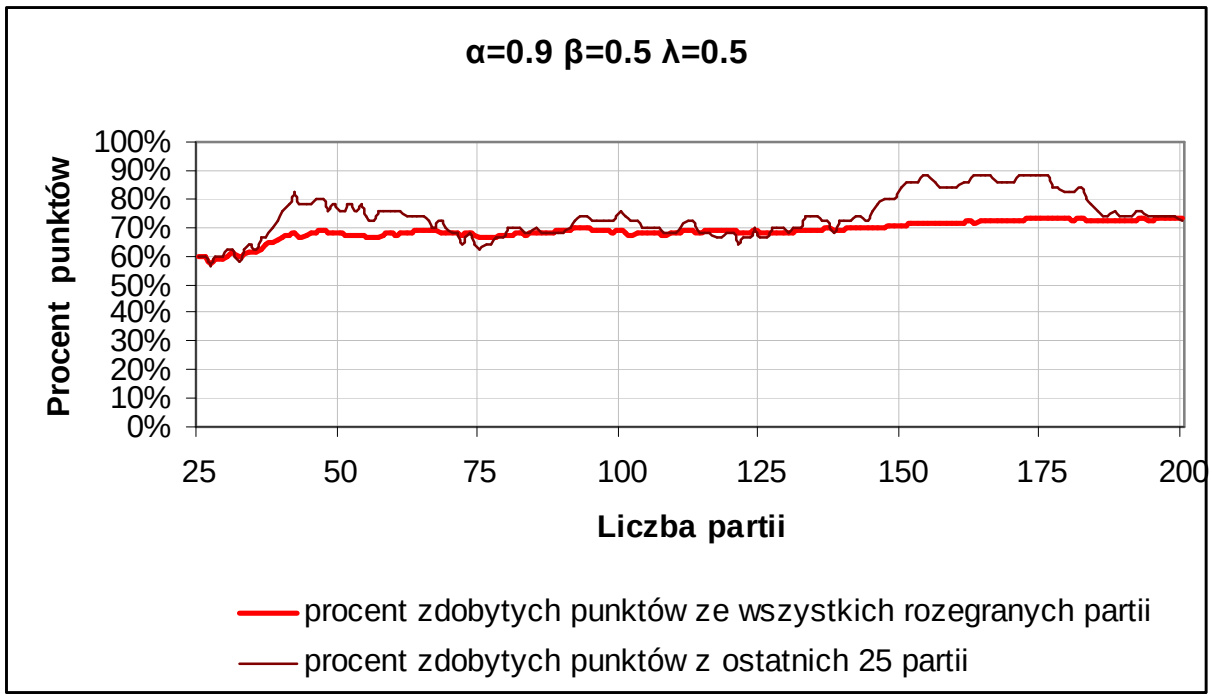
Poniżej (Rys. 5.7.-16.) przedstawiono wyniki wybranych 10 eksperymentów spośród wszystkich 27 przeprowadzonych, w których badano wpływ wartości parametrów α, β oraz λ na efektywność uczenia. Wyniki stanowią udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową*. Wyniki wszystkich 27 eksperymentów załączono w dodatku F.



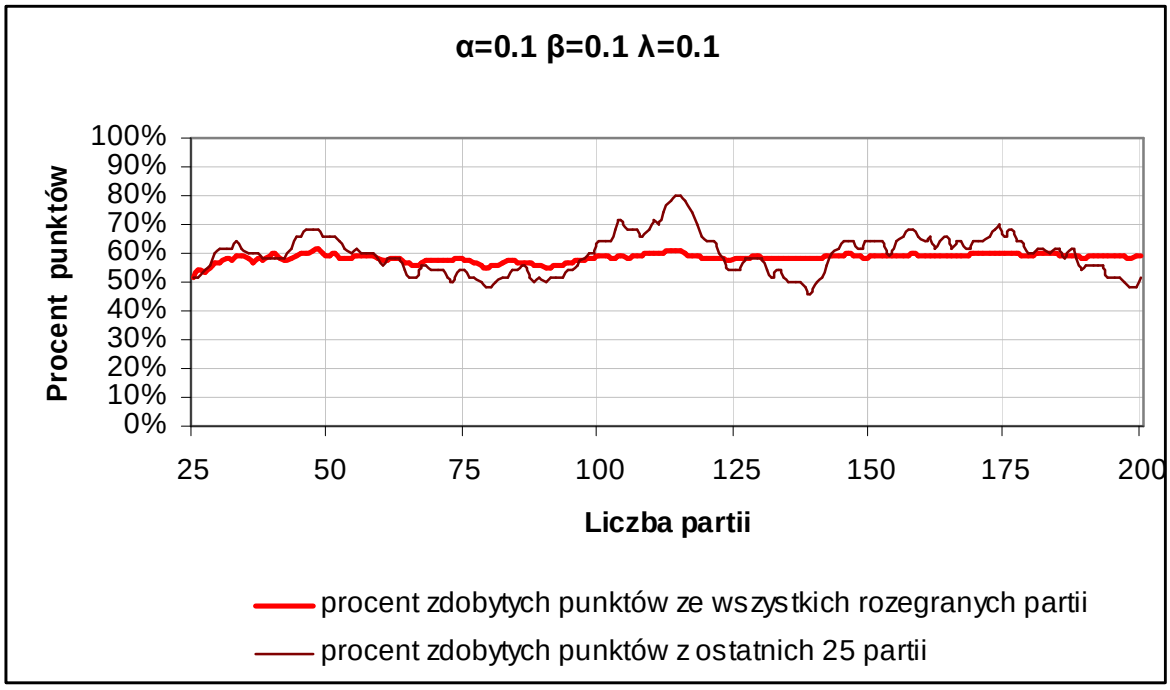

Rys. 5.7. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w eksperymencie z ustalonymi parametrami α=0.1, β=0.9 oraz λ=0.5



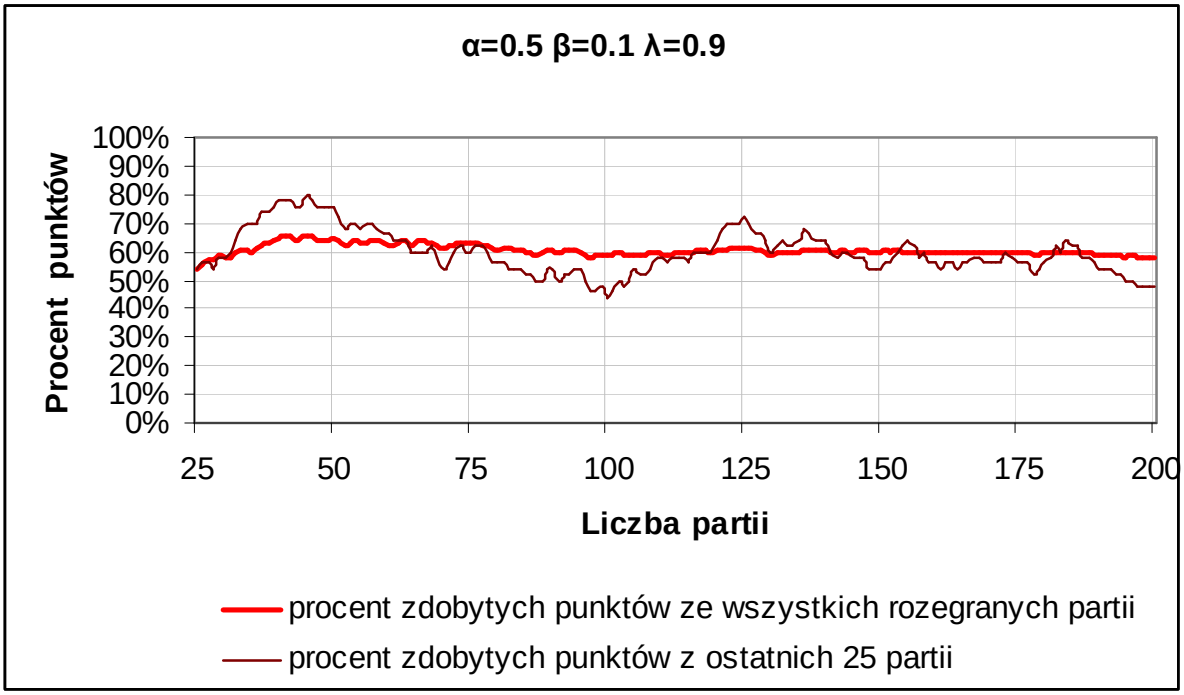

Rys. 5.8. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w eksperymencie z ustalonymi parametrami α=0.5, β=0.5 oraz λ=0.5



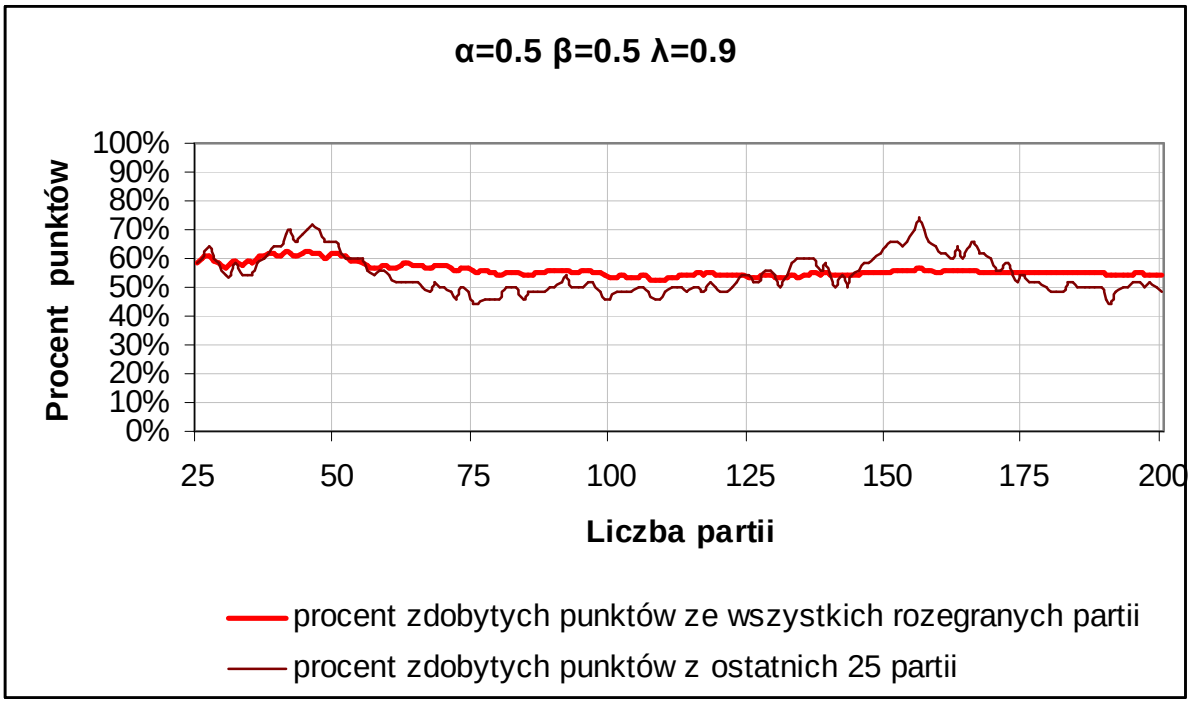

Rys. 5.9. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w eksperymencie z ustalonymi parametrami α=0.9, β=0.5 oraz λ=0.5



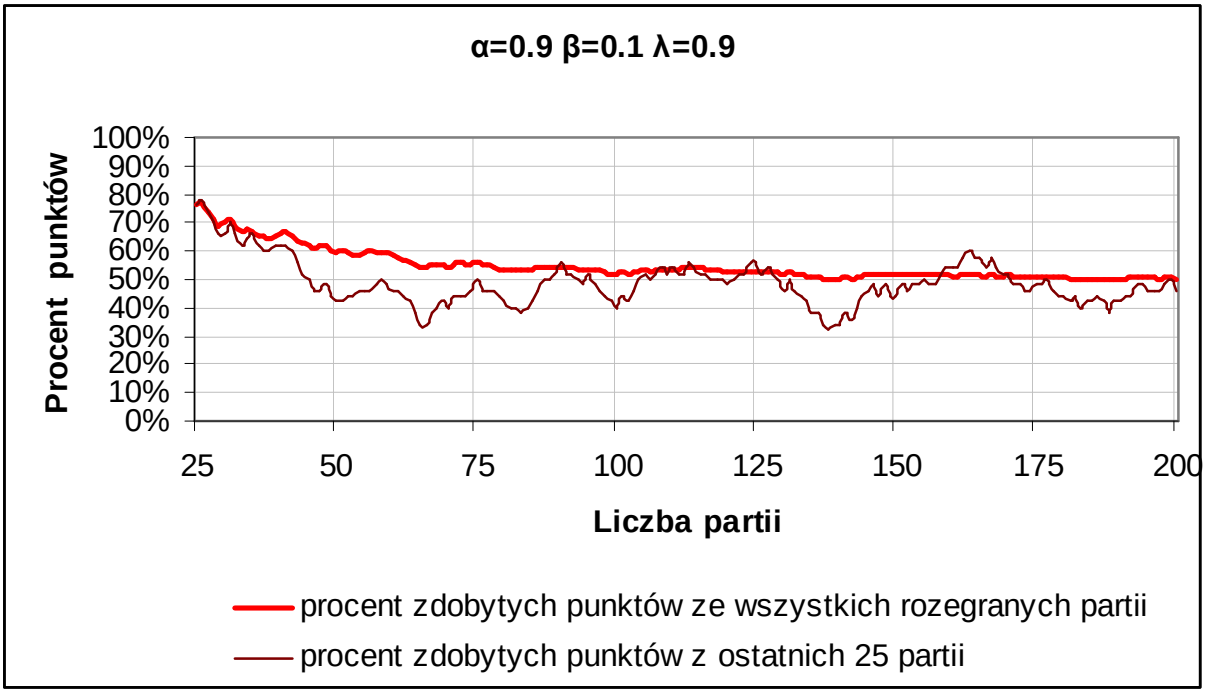

Rys. 5.10. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w eksperymencie z ustalonymi parametrami α=0.1, β=0.1 oraz λ=0.1



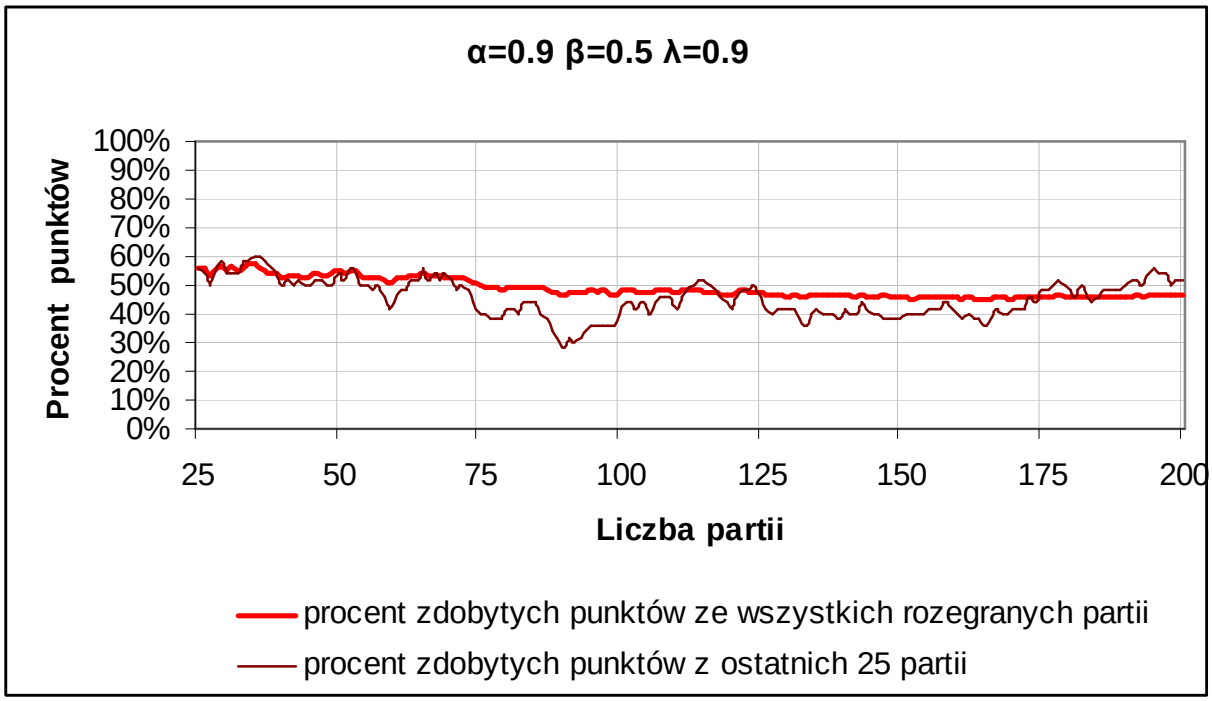

Rys. 5.11. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w eksperymencie z ustalonymi parametrami α=0.5, β=0.1 oraz λ=0.9



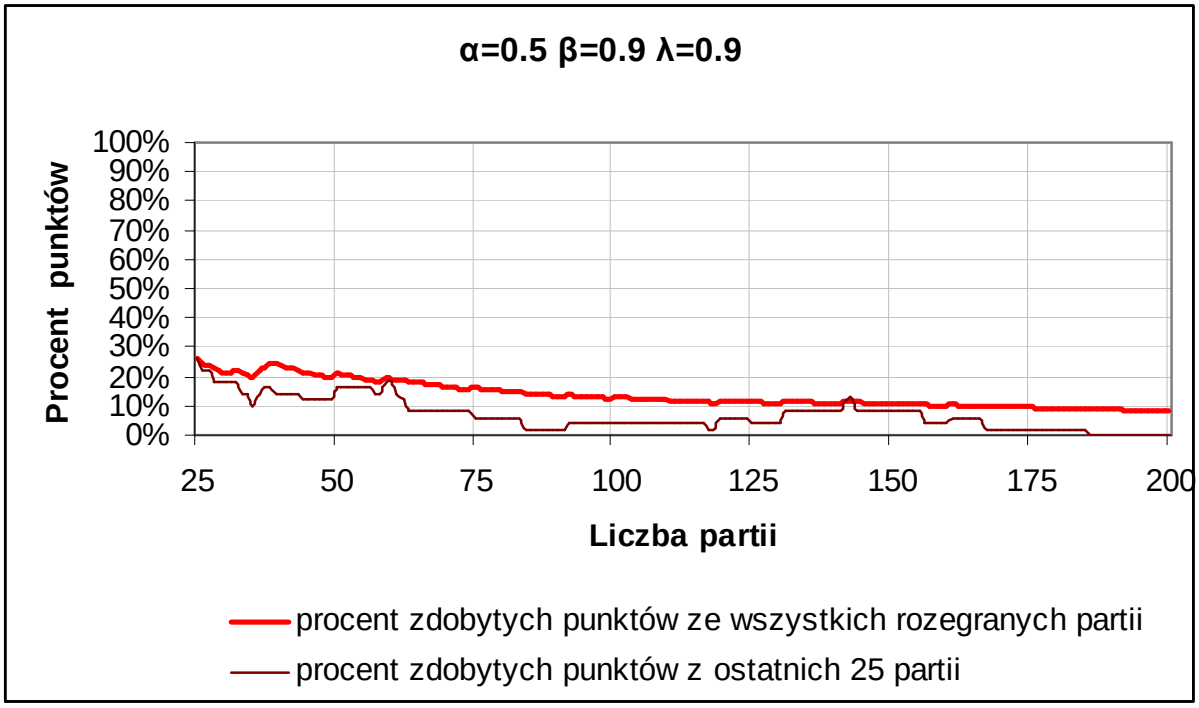

Rys. 5.12. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w eksperymencie z ustalonymi parametrami α=0.5, β=0.5 oraz λ=0.9



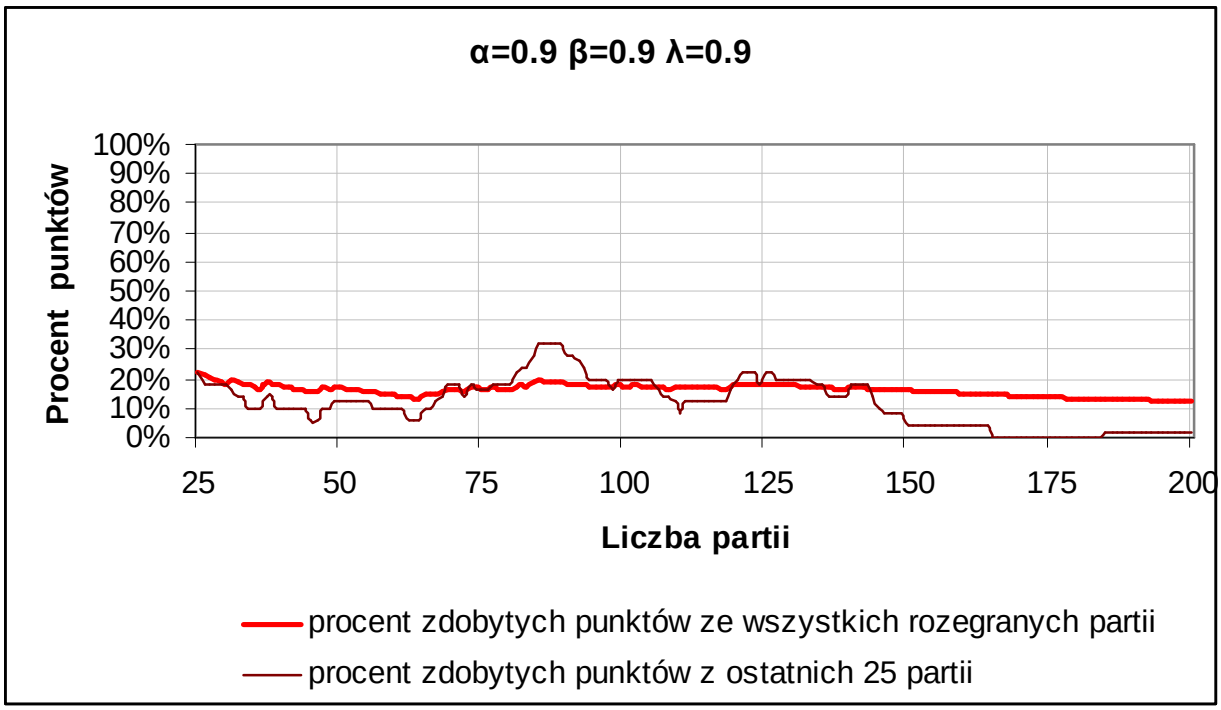

Rys. 5.13. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w eksperymencie z ustalonymi parametrami α=0.9, β=0.1 oraz λ=0.9




Rys. 5.14. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w eksperymencie z ustalonymi parametrami α=0.9, β=0.5 oraz λ=0.9




Rys. 5.15. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w eksperymencie z ustalonymi parametrami α=0.5, β=0.9 oraz λ=0.9




Rys. 5.16. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w eksperymencie z ustalonymi parametrami α=0.9, β=0.9 oraz λ=0.9

**Komentarz**

Niestety w wyniku przeprowadzonych eksperymentów nie znaleziono odpowiedzi na pytanie, który zestaw wartości parametrów jest rzeczywiście najlepszy. W większości przypadków eksperymenty przebiegały w podobny sposób, tzn. poziom gry *strategii TDL* tylko nieznacznie ulegał zmianie na lepsze. Na koniec większości eksperymentów udział zdobytych przez *strategię TDL* punktów kształtował się przy granicy 70%. Jedynie dla eksperymentów z wartościami parametrów [$\alpha$=0.1, $\beta$=0.9, $\lambda$=0.5 – Rys. 5.7.], [$\alpha$=0.5, $\beta$=0.5, $\lambda$=0.5 – Rys. 5.8.] oraz [$\alpha$=0.9, $\beta$=0.5, $\lambda$=0.5 – Rys. 5.9.] ostateczne wyniki ukształtowały się powyżej 70% zdobytych punktów ze wszystkich partii – odpowiednio 71%, 73% oraz 73%. W tych i kilku innych eksperymentach pojawiały się również ciekawe okresy gry *strategii TDL*, w których zdobywała ona ze *strategią treningową* około 80% punktów z ostatnich 25 partii. Natomiast kilka eksperymentów wykazało, że pewne kombinacje wartości parametrów $\alpha$, $\beta$ oraz $\lambda$ wpływają niekorzystnie na proces uczenia sieci neuronowej. W eksperymentach z wartościami parametrów [$\alpha$=0.1, $\beta$=0.1, $\lambda$=0.1 – Rys. 5.10.], [$\alpha$=0.5, $\beta$=0.1, $\lambda$=0.9 – Rys. 5.11.], [$\alpha$=0.5, $\beta$=0.5, $\lambda$=0.9 – Rys. 5.12.], [$\alpha$=0.9, $\beta$=0.1, $\lambda$=0.9 – Rys. 5.13.] oraz [$\alpha$=0.9, $\beta$=0.5, $\lambda$=0.9 – Rys. 5.14.] wynik końcowy kształtował się w zakresie 50-60% zdobytych punktów ze wszystkich rozegranych partii, a przebieg tych eksperymentów wykazywał nieznaczne pogarszanie gry *strategii TDL*. W szczególności eksperymenty z wartościami parametrów [$\alpha$=0.5, $\beta$=0.9, $\lambda$=0.9 – Rys. 5.15.] oraz [$\alpha$=0.9, $\beta$=0.9, $\lambda$=0.9 – Rys. 5.16.] wykazały całkowite rozstrojenie sieci neuronowej, po którym *strategia TDL* uzyskiwała zaledwie 10% punktów w grze ze *strategią treningową*. W tych dwóch przypadkach sieć neuronowa traciła całkowicie zdolność postrzegania najważniejszych cech gry związanych z ewentualną przewagą materialną czy możliwością zamatowania króla przeciwnika.

Jedyną ważną obserwacją (potwierdzającą regułę) wypływającą z przeprowadzonej serii eksperymentów jest charakter zmian wyników gry *strategii TDL* w zależności od wielkości parametrów $\alpha$, $\beta$ oraz $\lambda$. Mniejsze wartości tych parametrów wpływają na mniejszą dynamikę zmian stylu gry *strategii TDL*, a tym samym na dłuższe utrzymywanie podobnych udziałów zdobywanych z ostatnich gier punktów, a nieco większe wartości parametrów wpływają z kolei na zwiększenie tej dynamiki. Jednakże po przekroczeniu pewnej (nieokreślonej) granicy wpływają już tylko negatywnie na proces całej nauki sieci. W szczególności większość eksperymentów, w których parametr $\lambda$ był ustalony na wartość 0.9, kończyły się mniejszym lub większym niepowodzeniem. Przypuszczalnie najlepszymi wartościami dla badanych parametrów są wartości bliskie 0.5, przy czym w przypadku parametru $\lambda$ powinny to być najprawdopodobniej wartości nie większe niż 0.5.

Ostatecznie przyjęto, że do dalszych eksperymentów najlepszym zestawem wartości będzie: $\alpha$=0.5, $\beta$=0.5 oraz $\lambda$=0.5. Warto niemniej jednak pamiętać o tym, iż nie są to na pewno wartości optymalne, a tym bardziej uniwersalne, które pasują do każdego typu sieci neuronowej. W rzeczywistości najlepszy zestaw wartości tych parametrów będzie bardzo często zależał przynajmniej od stanu początkowego sieci neuronowej, ale również od samej jej architektury, a także poczynionych innych założeń. Do tego wszystkiego dochodzi również sposób trenowania (uczenia) sieci neuronowej. W powyższym przypadku sieć neuronowa poszukiwała optymalnej strategii gry przeciw prostej *strategii treningowej* i okazywało się, że w większości przypadków *strategia TDL* przez nią reprezentowana już od początku treningów doskonale sobie poczynała. W związku z czym na przykład przekroczenie bariery 80% zdobywanych punktów było już nieosiągalne ze względu na brak „dobrego trenera”.

## 5.3. Wpływ głębokości rozwijania drzewa gry na efektywność uczenia

Głównym celem trzeciej serii eksperymentów było zbadanie wpływu głębokości rozwijania drzewa gry przez obie strategie (*TDL* oraz *treningową*) na efektywność uczenia sieci neuronowej oraz poziom gry wyuczonej *strategii TDL*.

### Przebieg

Na przestrzeni ostatnich kilkudziesięciu lat najwięcej prac związanych z oprogramowaniem szachowym skupiało się na zwiększaniu głębokości i szybkości przeszukiwania drzewa gry. Niniejsza seria eksperymentów pozwoliła sprawdzić jak bardzo głębokość ta wpływa na poziom gry strategii i czy w ogóle warto poszukiwać o wiele bardziej złożonych strategii gry kosztem redukowania głębokości przeszukiwań drzewa gry. Ze względu na bardzo czasochłonny proces trenowania sieci w niniejszej serii ograniczono się zaledwie do czterech eksperymentów, w których wykonywano *treningi aktywne* z różnymi głębokościami przeszukiwań drzewa gry (2 lub 3 pół-ruchy) dla każdej ze strategii. W każdym eksperymencie stan początkowy sieci neuronowej ustalano analogicznie jak w poprzedniej serii eksperymentów, tzn. wartości wag na połączeniach wychodzących z neuronów wejściowych otrzymujących informację o cechach *siła pionów*, *siła skoczków*, *siła gońców*, *siła wież*, *siła hetmanów* oraz *mat* ustalano odpowiednio na wartości 0.1, 0.3, 0.3, 0.5, 1.0 oraz 10.0, a resztę wartości wag ustalano na 0.01. Dodatkowo każda waga była modyfikowana o wartość losową z rozkładu normalnego N(0.00,0.01). Stałe wartości parametrów dla każdego eksperymentu zebrano w Tab. 5.5.

| **Parametr** | **Wartość** |
|---|---|
| Trening | aktywny |
| Liczba gier | 1000 |
| $\alpha$ (współczynnik uczenia) | 0.5 |
| $\beta$ (nachylenie sigmoidy) | 0.5 |
| $\lambda$ (parametr metody *TD*) | 0.5 |

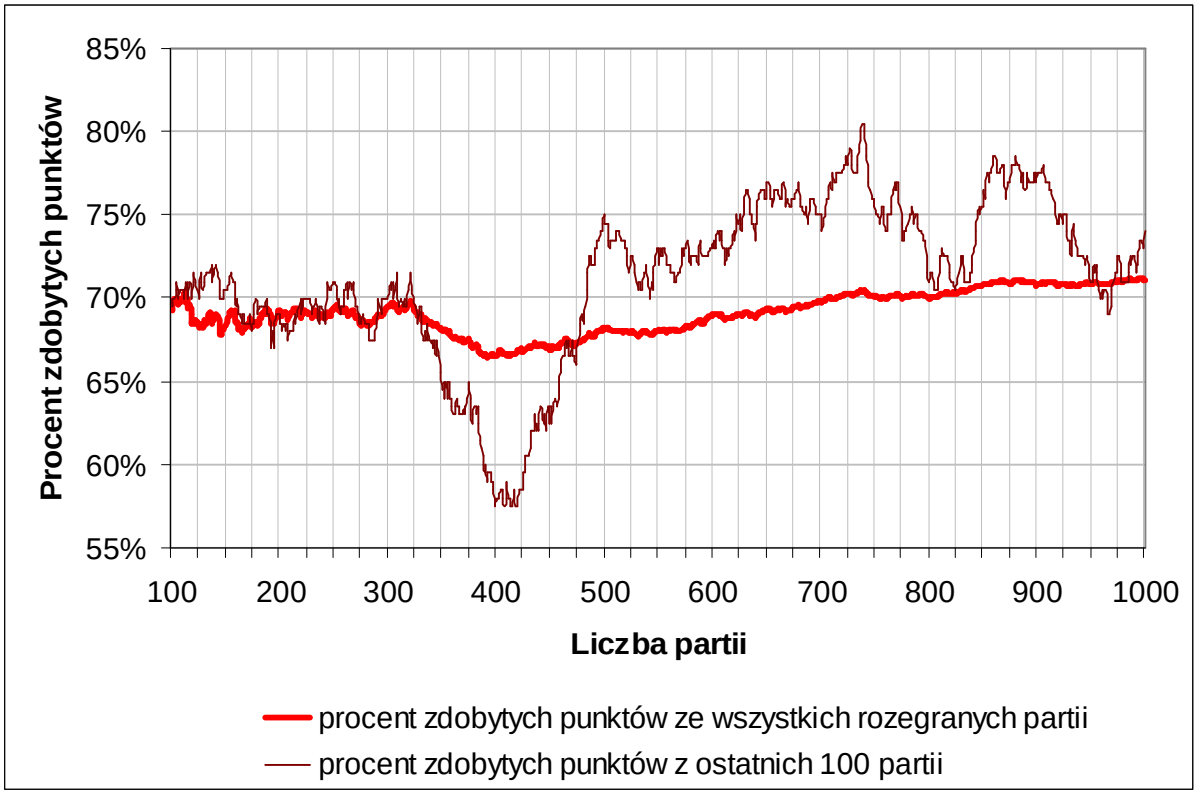
Tab. 5.5. Wartości parametrów dla całej serii eksperymentów

### Wyniki

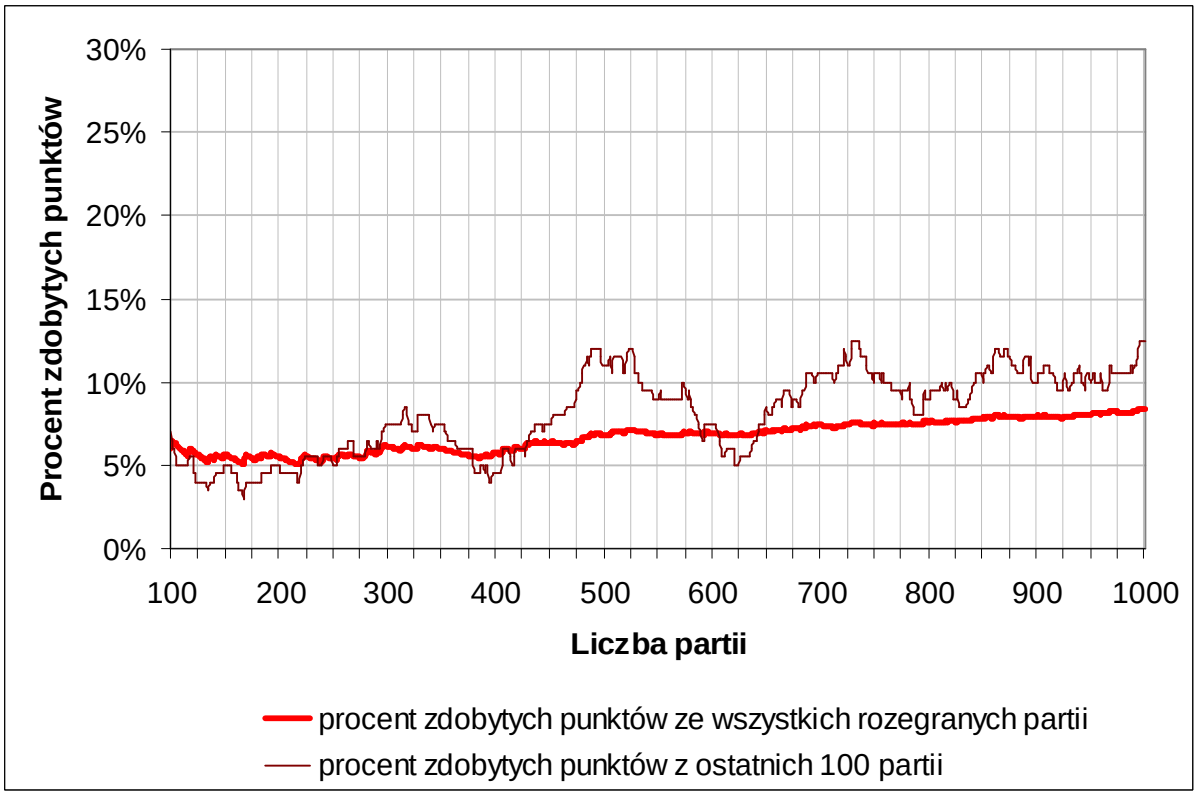
W Tab. 5.6. zebrano wyniki przeprowadzonych eksperymentów wraz z założeniami.

| **Eksperyment** | **Głębokość przeszukiwań drzewa gry** | | **Wyniki** | **Czas trwania eksperymentu** |
|---|---|---|---|---|
| | *strategia TDL* | *strategia treningowa* | | |
| I | 2 | 2 | Rys. 5.17. | 3 h |
| II | 2 | 3 | Rys. 5.18. | 16 h |
| III | 3 | 2 | Rys. 5.19. | 32 h |
| IV | 3 | 3 | Rys. 5.20. | 41 h |

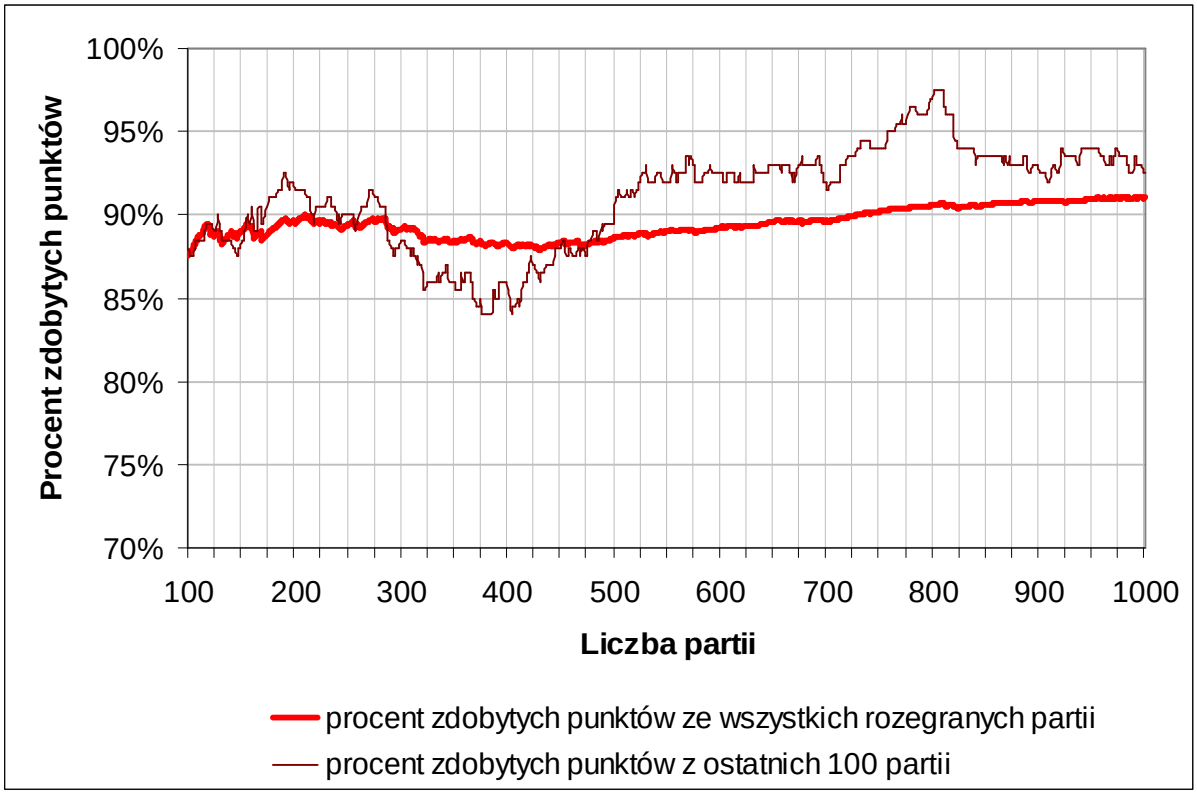
Tab. 5.6. Założenia i wyniki przeprowadzonych eksperymentów



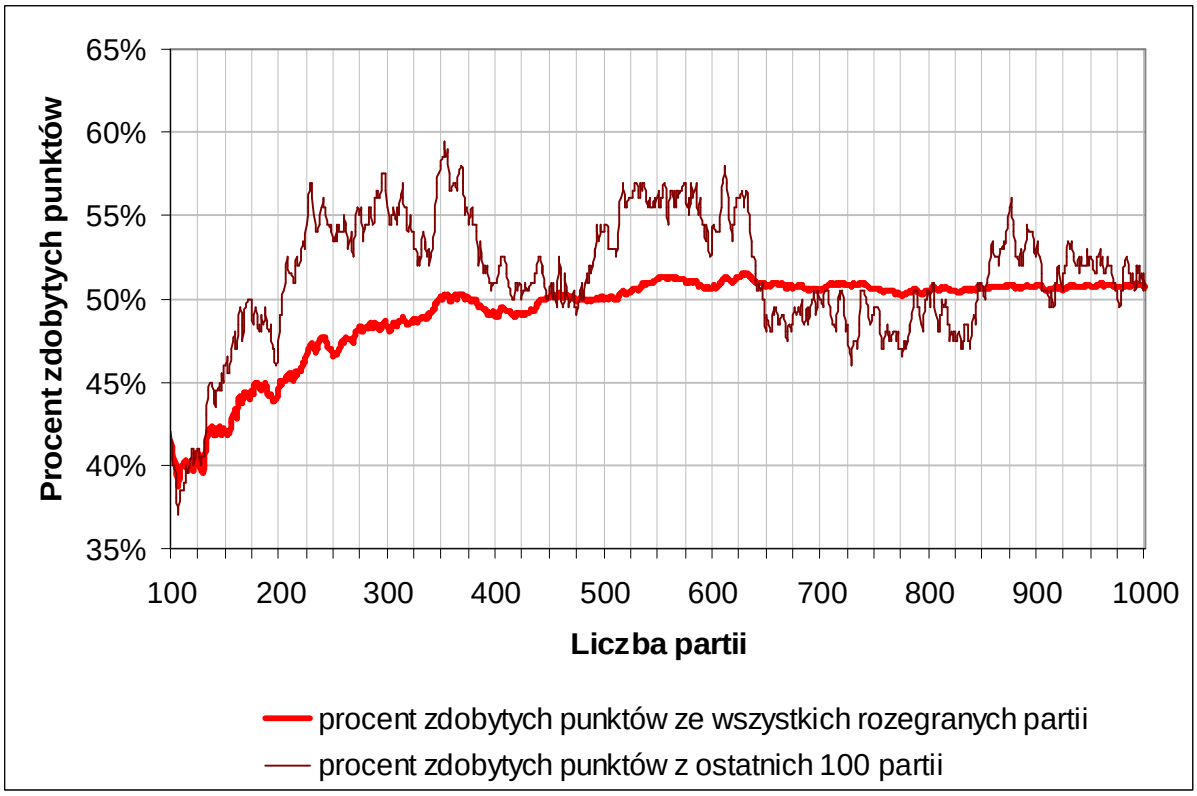

Rys. 5.17. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w I eksperymencie (opis parametrów w pierwszym wierszu w Tab. 5.6.)




Rys. 5.18. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w II eksperymencie (opis parametrów w drugim wierszu w Tab. 5.6.)




Rys. 5.19. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w III eksperymencie (opis parametrów w trzecim wierszu w Tab. 5.6.)




Rys. 5.20. Udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową* w IV eksperymencie (opis parametrów w czwartym wierszu w Tab.5.6.)

**Komentarz**

Wyniki przeprowadzonych eksperymentów potwierdziły jednoznacznie, iż głębokość przeszukiwań drzewa gry znacząco wpływa na poziom gry dowolnej strategii.

W pierwszym eksperymencie (Rys. 5.17.) *strategia TDL* oraz *strategia treningowa* przeszukiwały drzewo gry do drugiego pół-ruchu – identycznie jak w dotychczas przeprowadzanych eksperymentach. Tym razem jednak, przy odpowiednio założonych parametrach, pozwolono im rozegrać 1000 partii. Po raz kolejny okazało się, iż *strategia TDL* znacząco przewyższa siłę gry *strategii treningowej* już od początku trwania eksperymentu. Po 100 rozegranych partiach *strategia TDL* miała już na swoim koncie 69.5 punktów. Wyniki kolejnych 200 partii utrzymywały się na jednakowym poziomie oscylując w pobliżu 70% zdobytych punktów. Jednakże kolejne 100 partii (między 300. a 400.) wykazały znaczne pogorszenie gry *strategii TDL*, która coraz rzadziej zaczęła wygrywać osiągając nawet w najgorszym momencie 58% zdobytych punktów z ostatnich 100 rozegranych partii. Był to jednak przełomowy moment, po którym *strategia TDL* zaczęła grać zdecydowanie lepiej i w kolejnych 100 rozegranych partiach zdobyła 75% punktów. Od tej pory wyniki gry *strategii TDL* utrzymywały ciągły trend wzrostowy i do końca trwania eksperymentu udział zdobytych punktów z ostatnich 100 partii oscylował w przedziale 70%-80%. Ostatecznie *strategia TDL* zdobyła 71% punktów ze wszystkich rozegranych w eksperymencie partii, wygrywając 56% gier, remisując 30% i przegrywając zaledwie 14%. Na osiągnięte wyniki zdecydowany wpływ musiała mieć postać wyuczonej funkcji oceniającej, która zwracała uwagę na cechy gry nieznane *strategii treningowej*.

W drugim eksperymencie (Rys. 5.18.) *strategia TDL* nadal przeszukiwała drzewo gry do drugiego pół-ruchu naprzód, a jej przeciwnik – *strategia treningowa* – już do trzeciego pół-ruchu naprzód. W porównaniu do pierwszego eksperymentu wyniki tego eksperymentu okazały się być diametralnie inne. Zwiększenie głębokości przeszukiwania drzewa gry o jeden pół-ruch do przodu dla *strategii treningowej* przyczyniło się do znacznego zwiększenia jej poziomu gry. *Strategia TDL* po pierwszych 100 rozegranych partiach zdobyła zaledwie 6 punktów – wygrywając dwie partie, remisując osiem i resztę przegrywając. W poprzednim eksperymencie przeciwnik *strategii TDL* wydawał się być za słaby, a tym razem okazał się być zbyt silny, aby mierzyć się z nim jak równy z równym. Jedynym pozytywnym wrażeniem płynącym z wyników przeprowadzonego eksperymentu jest ciągła tendencja wzrostowa udziału zdobywanych przez *strategię TDL* punktów. Już po pięciuset rozegranych partiach *strategia TDL* zdołała zdobyć 12 punktów z ostatnich 100 partii (dwa razy więcej niż na początku). Był to nadal bardzo słaby wynik, ale dowodził ciągłemu wzrostowi umiejętności gry *strategii TDL*. W ostatnich trzystu partiach eksperymentu *strategia TDL* zdobyła 10.5% punktów, a ostatecznie 8.35% punktów ze wszystkich rozegranych partii. Tym razem fakt zwracania uwagi na wiele niematerialnych cech gry przez *strategię TDL* był niewystarczający wobec faktu głębszego przeszukiwania drzewa gry przez *strategię treningową*, aby osiągnąć lepszy poziom gry.

W trzecim eksperymencie (Rys. 5.19.) postanowiono odwrócić role – tym razem *strategia TDL* przeszukiwała drzewo gry do trzeciego pół-ruchu naprzód, a *strategia treningowa* – do drugiego. Zgodnie z oczekiwaniami poziom gry *strategii TDL* w ciągu całego eksperymentu był o klasę lepszy od poziomu gry *strategii treningowej* – głównie za sprawą głębszego przeszukiwania drzewa gry. Już po pierwszych stu partiach *strategia TDL* zdobyła 87.5 punktów. Poziom ten utrzymywał się przez kolejne dwieście partii. Jednak co ciekawe, podobnie jak w pierwszym eksperymencie, pomiędzy 300. a 400. partią doszło do zauważalnego pogorszenia wyników. Przyczyny pojawiającego się zjawiska zmiany stylu gry na gorsze można upatrywać jedynie w procesie samej nauki, podczas której w trakcie dostrajania stanu sieci neuronowej mógł pojawić się niekorzystny zestaw wartości wag na połączeniach. Przyglądając się jednak dalszym wynikom eksperymentu łatwo

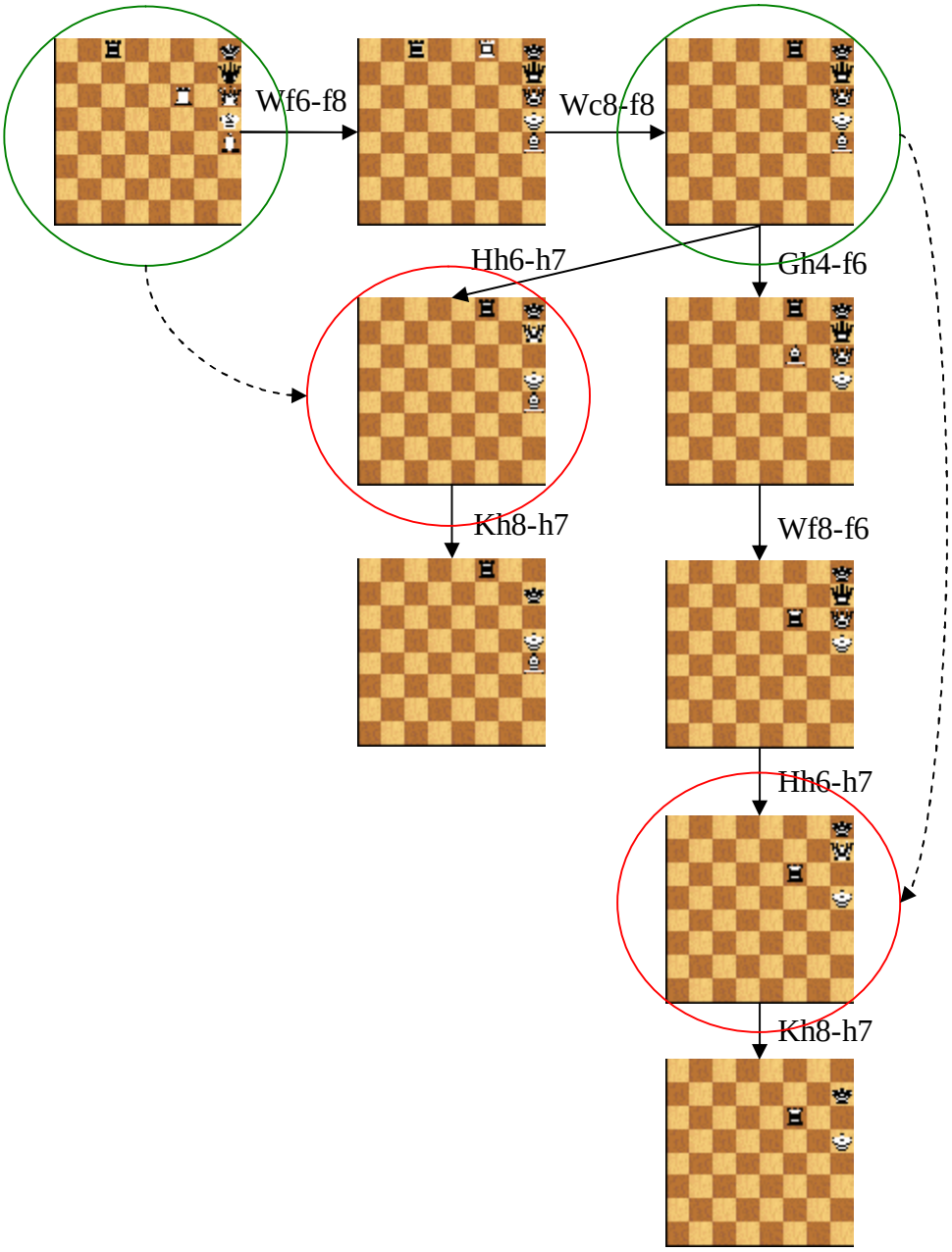
zauważyć, iż podczas kolejnych partii udział zdobytych przez *strategię TDL* punktów znacząco rósł, osiągając najlepszy dotychczas rezultat 93 punktów zdobytych z ostatnich 100 partii. Zatem można przypuszczać, iż w procesie dalszej nauki stan sieci neuronowej uległ prawidłowemu skorygowaniu. Od tej chwili, już do końca eksperymentu *strategia TDL* utrzymywała bardzo wysoki poziom gry, zdobywając średnio 93% punktów ze stu ostatnio rozegranych partii, a w najlepszych chwilach nawet 97.5%. Ostatecznie *strategia TDL* zdobyła 91% punktów ze wszystkich przeprowadzonych partii w całym eksperymencie, wygrywając 84.5% gier, remisując 13% i przegrywając zaledwie 2.5%.

W czwartym eksperymencie (Rys. 5.20.) skonfrontowano wreszcie ze sobą obie strategie przy założeniu, że przeszukują one drzewo gry do trzeciego pół-ruchu naprzód. Założenia tego eksperymentu mocno przypominają założenia poczynione w pierwszym eksperymencie – obie strategie przeszukują drzewo gry do tej samej głębokości. Można więc było się spodziewać podobnych wyników, ale jak się okazało wyniki tego eksperymentu znacznie odbiegły od wyników pierwszego eksperymentu. Po pierwsze poziom gry *strategii treningowej* okazał się być na początku nieco lepszy od *strategii TDL*, która po pierwszych stu partiach zdobyła tylko 41.5 punktów. Na szczęście kolejne 250 partii wskazywało na ciągłą poprawę stylu gry *strategii TDL*, która zaczęła zdobywać coraz więcej punktów, osiągając poziom nawet 59.5 zdobytych punktów ze stu ostatnich partii oraz 50% zdobytych punktów ze wszystkich rozegranych dotychczas partii. Od tej pory nie doszło już do znaczących zmian w udziale zdobywanych przez *strategię TDL* punktów. Jedyną ciekawą obserwację można dokonać przyglądając się wykresowi związanemu z udziałem procentowym zdobytych punktów z ostatnich stu partii. Wyniki te można podzielić na cztery okresy: 400-500, 500-650, 650-850, 850-1000, w których utrzymywał się poziom odpowiednio ok. 51%, 55%, 48%, 53% zdobytych punktów z ostatnich stu rozegranych partii. Skoki pomiędzy tymi okresami świadczą o ciągłym poszukiwaniu optymalnej strategii gry wyrażonym zmianami stanu sieci neuronowej, które miały wpływ na styl gry *strategii TDL*. Ostatecznie *strategia TDL* zdobyła 50.8% punktów ze wszystkich rozegranych w eksperymencie partii, wygrywając 37% gier, remisując 27.5% i przegrywając 35.5%.

Wszystkie przeprowadzone w tej części eksperymenty wykazały znaczący wpływ głębokości rozwijanego drzewa gry na poziom gry obu strategii i efektywność uczenia *strategii TDL*. Zwiększenie głębokości przeszukiwania drzewa o jeden pół-ruch naprzód znamiennie wpływa na styl gry dowolnej strategii. Poszukiwanie bardziej złożonej strategii gry kosztem płytszego przeszukiwania drzewa gry mija się z celem. Stworzenie silnego programu grającego w szachy możliwe jest przy wykorzystaniu nawet najprostszej funkcji oceniającej. Jedynym ważnym czynnikiem staje się umożliwienie możliwie najgłębszego rozwijania drzewa gry. Co prawda najprawdopodobniej przy większych głębokościach przeszukiwania drzewa gry różnica jednego pół-ruchu nie wpływa już tak drastycznie na poziom gry strategii, ale nie ulega wątpliwości, że zawsze będzie miała duże znaczenie.

Pomimo znaczących różnic dotyczących udziału zdobywanych punktów, wyniki wszystkich eksperymentów wykazywały ukierunkowany wzrost siły gry *strategii TDL*. Zatem proces uczenia z wykorzystaniem sieci neuronowej oraz algorytmu *TDLeaf* przebiegał za każdym razem prawidłowo, dając zadowalające efekty. Jedynym mankamentem staje się szybkość przeszukiwania drzewa gry, którą należałoby znacząco poprawić. W wyniku płytkiego rozwijania drzewa gry *strategia TDL* wcale nie poszukuje globalnie optymalnej funkcji oceniającej, tylko lokalnie najlepszej dla ustalonej głębokości przeszukiwań, a także w tym przypadku dla gry z określoną *strategią treningową*. W bardzo wielu przypadkach dużą przeszkodą w realizacji przewagi okazywał się krótki horyzont gry, tzn. analiza drzewa gry tylko do drugiego bądź trzeciego pół-ruchu naprzód, który przyczyniał się do braku planu gry w zwycięskich pozycjach, a które wobec tego kończyły się remisem (najczęściej wynikającym z wykonania pięćdziesięciu ruchów

bez bicia). Remisowanie wygranych pozycji z pewnością niekorzystnie wpływało na uczenie się sieci. Krótki horyzont miał również dosyć często negatywny wpływ na ocenę pozycji pojawiających się w środku wielo-ruchowych wymian bądź ślepo widzianej zdobyczy materialnej poprzedzonej szachem. Przykładem takiej pozycji prowadzącej do „klęski materialnej" wywołanej ślepym zapatrzeniem w zdobycz może być sytuacja przedstawiona na Rys. 5.21.




Rys. 5.21. Ślepe zapatrzenie w zdobycz przy krótkim horyzoncie gry na wybranym przykładzie rozwiniętego drzewa gry z przedstawionymi tylko dwiema gałęziami – zielonymi okręgami zaznaczono miejsca, od których strategia rozwija drzewo do trzech pół-ruchów naprzód, gdzie wybiera najlepiej ocenioną pozycję zaznaczoną okręgiem czerwonym

Załóżmy, zgodnie z przykładem, iż omawiana strategia rozpoczyna rozwijanie drzewa gry do trzech pół-ruchów naprzód z pozycji zaznaczonej zielonym okręgiem w lewym górnym rogu. Po przejrzeniu całego drzewa, strategia ocenia, iż najlepszą pozycją jest ta zaznaczona

czerwonym okręgiem z lewej strony rysunku, ponieważ doprowadza do przewagi materialnej – poświęca wieżę, która daje szacha przeciwnikowi, a następnie zdobywa jego hetmana. Niestety brak rozwinięcia gałęzi o jeden pół-ruch do przodu mści się złą oceną pozycji, gdyż w następnym posunięciu czarne jedynym możliwym sposobem odbijają białego hetmana. Po wykonaniu ruchu wieżą zgodnie z planem oraz odpowiedzi przeciwnika, strategia rozpoczyna ponownie rozwijanie drzewa gry do trzech pół-ruchów naprzód tym razem z pozycji zaznaczonej zielonym okręgiem w prawym górnym rogu. W tej chwili już bez problemu ocenia, że wcale nie zdobędzie hetmana przeciwnika bijąc go swoim hetmanem, więc poszukuje innego potencjalnie dobrego wariantu. Po przejrzeniu całego drzewa, strategia w podobny sposób ocenia, że może zdobyć hetmana przeciwnika po ówczesnym zaszachowaniu jego króla gońcem i niestety po raz kolejny brak rozwinięcia gałęzi o jeden pół-ruch dalej przyczynia się do ostatecznej klęski materialnej białych, które bezpodstawnie pozbawiły siebie dwóch figur – wieży i gońca. Jak się jednak okazuje wydłużenie horyzontu gry wcale nie zapewniłoby jeszcze ominięcia tego typu problemu – choć na pewno zdarzałby się on rzadziej. Jedynym rozwiązaniem jest wprowadzenie testu na tzw. pozycje stabilne, o których była już mowa w drugim rozdziale. Pozycje takie charakteryzują się najczęściej sytuacją, w której nie może już dojść do ostrych wymian figurowych, przy których ocena kolejnych pozycji ulega diametralnym zmianom. Stąd też praktykowana jest często metoda rozwijania drzewa gry do najbliższej pozycji stabilnej leżącej nie wyżej niż maksymalna głębokość przeszukiwań. Z powodu krótkiego czasu na realizację programu szachowego nie zastosowano jednak tego typu rozwiązania, gdyż niezoptymalizowana własna implementacja znacznie wydłużała czas przeprowadzania doświadczeń. Dlatego też wszystkie omawiane doświadczenia zostały obarczone takimi „błędami", które zaszumiały ich wyniki i wpływały niekorzystnie na proces uczenia sieci neuronowej. Natomiast podobne „błędy" popełniane były zarówno przez *strategię treningową* jak i *strategię TDL*, zatem pod tym względem szanse były wyrównane. Omawiany tutaj horyzont gry miał zapewne decydujący wpływ na różnicę w wynikach pomiędzy eksperymentem pierwszym, gdzie obie strategie rozwijały drzewo gry do drugiego pół-ruchu naprzód, a eksperymentem czwartym, gdzie obie strategie rozwijały drzewo gry do trzeciego pół-ruchu naprzód. Prawdopodobnie *strategia TDL* uzyskuje znacznie lepsze wyniki w grze ze *strategią treningową* przy rozwijaniu drzewa gry do parzystej liczby pół-ruchów naprzód, a nie nieparzystej. Chodzi tutaj przede wszystkim o kwestię widzenia odpowiedzi przeciwnika na swój ostatni pół-ruch w rozwijanej gałęzi drzewa.

## 5.4. Wpływ sposobu trenowania sieci neuronowej na jakość gry

Głównym celem czwartej serii eksperymentów było zbadanie wpływu sposobu trenowania sieci neuronowej na jakość gry wyuczonej *strategii TDL*.

### Przebieg

We wszystkich dotychczasowych eksperymentach optymalną strategię gry poszukiwano przy wykorzystaniu *treningu aktywnego*. W niniejszej serii eksperymentów postanowiono wypróbować innego sposobu uczenia sieci – *treningu biernego*, który opiera się na obserwacji partii rozegranych przez różnej klasy szachistów. Po zakończeniu *treningu biernego*, w celu zweryfikowania jakości gry wyuczonej *strategii TDL* przeprowadzono turniej z wykorzystaniem autorskiej aplikacji *Neural Chess – Turniej*, która służy do organizowania turniejów dla zaimplementowanych w systemie strategii.

*Trening bierny* przeprowadzono na podstawie tysiąca losowo wybranych partii szachowych spośród zbioru prawie pół miliona wszystkich partii. Założono również, iż każda pozycja (pojawiająca się podczas obserwacji losowo wybranej partii) oceniana będzie przez sieć na podstawie rozwinięcia drzewa gry do trzeciego pół-ruchu naprzód.

Stan początkowy sieci neuronowej ustalono identycznie jak dotychczas - wartości wag na połączeniach wychodzących z neuronów wejściowych otrzymujących informację o cechach *siła pionów*, *siła skoczków*, *siła gońców*, *siła wież*, *siła hetmanów* oraz *mat* ustalano odpowiednio na wartości 0.1, 0.3, 0.3, 0.5, 1.0 oraz 10.0, resztę wartości wag ustalano na 0.01, a następnie każdą wagę zmodyfikowano o wartość losową z rozkładu normalnego N(0.00,0.01).

Turniej przeprowadzono z udziałem trzech strategii: *strategii treningowej*, *strategii TDL* wyuczonej na podstawie *treningu aktywnego* (wykorzystano wyniki czwartego eksperymentu poprzedniej serii) oraz *strategii TDL* wyuczonej w tej serii eksperymentów na zasadzie *treningu biernego*. Każda z wymienionych strategii rozwijała drzewo gry do trzeciego pół-ruchu naprzód. W turnieju każda strategia rozegrała z każdą inną strategią 15 partii przy losowym doborze koloru figur.

Wszystkie stałe wartości parametrów dotyczących uczenia podczas *treningu biernego* zebrano w Tab. 5.7.

| **Parametr** | **Wartość** |
|---|---|
| Trening | bierny |
| Liczba gier | 1000 |
| Głębokość rozwijania drzewa gry | 3 |
| $\alpha$ (współczynnik uczenia) | 0.5 |
| $\beta$ (nachylenie sigmoidy) | 0.5 |
| $\lambda$ (parametr metody *TD*) | 0.5 |

Tab. 5.7. Wartości parametrów dla eksperymentu uczenia sieci neuronowej na podstawie *treningu biernego*

**Wyniki**

W przypadku pierwszego przeprowadzonego eksperymentu – uczenia sieci neuronowej na zasadzie *treningu biernego* – można było obserwować jedynie zmianę stanu sieci neuronowej następującej po każdej przejrzanej partii. Spośród tysiąca losowo wybranych partii szachistów, 493 zakończone były zwycięstwem białych, 40 – remisem, a 467 – zwycięstwem czarnych. Największy zakres zmian wartości wag pojawił się na połączeniach wychodzących z neuronów wejściowych otrzymujących informacje o cechach takich jak *szach* [-0.94;0.79], *siła wież* [-0.93;0.67], *opanowanie szachownicy* [-0.88,0.52], *bliskość króli do centrum* [-0.74,0.63] oraz *siła gońców* [-0.52,0.47]. Najmniejszy zakres zmian wartości wag pojawił się z kolei na połączeniach wychodzących z neuronów wejściowych otrzymujących informacje o cechach takich jak *niebezpieczne wieże* [-0.06;-0.03], *wzmocnienie kandydatów na wolne pionki* [-0.02;0.01], *mat* [-0.03;0.00] oraz *potencjalne przyczółki* [0.04;0.07].

Wyniki przeprowadzonego turnieju zestawiono w Tab. 5.8.

| **Miejsce** | **Strategia** | **Pkt** | **Wygrane** | **Remisy** | **Przegrane** |
|---|---|---|---|---|---|
| 1. | *Strategia TDL* (*trening aktywny*) | 18.0 | 10 | 16 | 4 |
| 2. | *Strategia treningowa* | 15.5 | 12 | 7 | 11 |
| 3. | *Strategia TDL* (*trening bierny*) | 11.5 | 4 | 15 | 11 |

Tab. 5.8. Wyniki turnieju dla trzech wybranych strategii gry, który został rozegrany na zasadzie „każdy z każdym po 15 partii”

**Komentarz**

Zinterpretowanie końcowego stanu sieci neuronowej po zakończeniu *treningu biernego* nie jest właściwie możliwe, ze względu na różnorodność wartości wag na połączeniach sieci objawiającej się mniej więcej równym podziałem na wartości ujemne i dodatnie. Z tego też względu ciężko ocenić, które informacje o cechach mają mniej lub bardziej znaczący wpływ na wybór najlepszego posunięcia przez funkcję oceniającą *strategii TDL*. Wyraźne są natomiast zmiany jakie zaszły w sieci neuronowej w procesie uczenia, a które zostały przedstawione powyżej w wynikach. Oznacza to, że *trening bierny* może równie dobrze nadawać się do poszukiwania optymalnej strategii gry co *trening aktywny*.

Wyniki turnieju nie pozastawiają jednak złudzeń. *Strategia TDL* wyuczona podczas *treningu aktywnego* poradziła sobie najlepiej w turnieju zdobywając 60% wszystkich możliwych punktów, przegrywając przy tym zaledwie cztery partie. Natomiast *strategia TDL* wyuczona podczas *treningu biernego* zajęła ostatnie miejsce zdobywając niecałe 40% punktów i wygrywając przy tym tylko cztery partie. Powyższe wyniki pozwalają przypuszczać, iż znacznie lepsze efekty poszukiwania optymalnej strategii gry daje uczenie podczas *treningu aktywnego*. Warto jednak zwrócić uwagę na niedociągnięcia, które nie zostały wyeliminowane podczas przeprowadzania *treningu biernego*, a które mogły mieć niekorzystny wpływ na naukę sieci neuronowej, pomimo wielu korzyści płynących z obserwacji partii rozegranych przez różnej klasy szachistów. Zbiór prawie pół miliona partii nie został odpowiednio przefiltrowany z szumu. W zbiorze tym znajdowała się znaczna część partii, która kończyła się wynikiem odbiegającym od rzeczywistego stanu rzeczy na szachownicy. Wszystkie partie pochodzą z bazy danych gier rozegranych na kafejce internetowej www.szachy.org. Partie te grane w większości z wykorzystaniem limitu czasowego nie zawsze kończyły się wygraną strony posiadającej rzeczywistą przewagę. Część z nich z pewnością kończona była w wyniku przekroczenia limitu czasowego strony wygrywającej, co jest równoznaczne z jej przegraną. Ten sam problem mógł pojawiać się w pozycjach remisowych lub całkowicie niejasnych (takich, w których trudno ocenić czyjąś przewagę). Tego typu sytuacje wprowadzają uczoną sieć neuronową w duży błąd, przez co wyniki treningu mogą być mocno zniekształcone. Korzystnie byłoby więc starannie dobrać odszumiony tysiąc partii szachowych, aby wyeliminować niepożądane zjawisko.

Niemniej jednak zwycięska strategia z turnieju została w nagrodę wykorzystana w ostatniej serii eksperymentów.

## 5.5. Konfrontacja najlepiej wyuczonej sieci neuronowej z innymi programami grającymi w szachy

Głównym celem ostatniej serii eksperymentów było zbadanie poziomu gry najlepiej wyuczonej sieci neuronowej (najlepszej *strategii TDL*) poprzez skonfrontowanie jej w grze z innymi programami grającymi w szachy.

**Przebieg**

Do niniejszej serii eksperymentów wybrano *strategię TDL*, która zwyciężyła turniej w ostatnim przeprowadzonym eksperymencie. Poziom gry *strategii TDL* pozostawia jeszcze wiele do życzenia, głównie za sprawą płytkiego przeszukiwania drzewa gry. Warto jednak skonfrontować jej możliwości z programami grającymi w szachy na porównywalnym poziomie. Do eksperymentów wybrano kilka różnych ogólnodostępnych programów szachowych umożliwiających regulację swojego poziomu gry:

- *TalvMenni*[18] – program grający w szachy będący wynikiem pracy magisterskiej nad algorytmami równoległego przeszukiwania drzewa gry obronionej na Uniwersytecie Wysp Owczych,
- *Brutal Chess*[19] – otwarty program szachowy z interesującym trójwymiarowym interfejsem graficznym,
- *Capa*[20] – otwarty program szachowy,
- *Crafty*[21] – jeden z najlepszych otwartych programów szachowych rozwijany od ponad 20 lat,
- *Mobile Chess* – program do gry w szachy znajdujący się standardowo na komórce *Samsung SGH X480*.

Ze względu na ustaloną maksymalną głębokość przeszukiwań *strategii TDL* (trzy pół-ruchy naprzód) wypadałoby wyrównać szanse gry z wybranymi programami poprzez jednakowe ustalenie głębokości rozwijania drzewa gry. Niestety niewiele programów szachowych dostarcza opcji bezpośredniego określania tego typu parametrów. Poziom gry większości programów ustalany jest pośrednio przez kilkustopniową skalę typu: *Easy*, *Normal*, *Hard*. Spośród wybranych programów jedynie *Crafty* umożliwia ustalenie głębokości przeszukiwania drzewa gry z poziomu interfejsu użytkownika poprzez wywołanie komendy *sd* z parametrem liczbowym (w pomocy opisanej jako: *sets absolute search depth*). W pozostałych przypadkach ustalenie tego parametru wymaga przejrzenia kodu źródłowego i odpowiedniego jego zmodyfikowaniu. Czynność tę wykonano jedynie w przypadku programu *TalvMenni*. W pozostałych przypadkach ograniczono się do gry na dostarczonych w interfejsie użytkownika poziomach. Odpowiednie ustawienia programów zestawiono w Tab. 5.9.

| **Program** | **Ustalony poziom gry** |
|---|---|
| *Mobile Chess* | *Easy*, *Normal*, *Hard* |
| *TalvMenni* | głębokość przeszukiwania drzewa gry ustalono na trzy pół-ruchy naprzód poprzez ręczne zmodyfikowanie odpowiedniego fragmentu kodu źródłowego |
| *Capa* | *Easy* |
| *Brutal Chess* | *Easy* |
| *Crafty* | głębokość przeszukiwania drzewa gry ustalono na trzy pół-ruchy naprzód poprzez wykorzystanie komendy *sd* z poziomu interfejsu użytkownika |

Tab. 5.9. Ustawienia programów szachowych, z którymi skonfrontowano najlepszą *strategię TDL*

Ze względu na nieposiadanie przydatnego interfejsu, który pozwoliłby w sposób automatyczny skonfrontować wybrane programy, ograniczono się do ręcznego przeprowadzenia wszystkich rozgrywek i ostatecznie założono, że z każdym z wymienionych wyżej programów szachowych *strategia TDL* rozegra tylko dwie partie – jedną kolorem białym, a drugą kolorem czarnym. Jedynym wyjątkiem był program *Mobile Chess*, który umożliwia użytkownikowi grę tylko i wyłącznie kolorem białym.

[18] Źródła programu *TalvMenni* możliwe są do ściągnięcia ze strony http://sourceforge.net/projects/talvmenni.
[19] Źródła programu *BrutalChess* możliwe są do ściągnięcia ze strony http://sourceforge.net/projects/brutalchess.
[20] Źródła programu *Capa* możliwe są do ściągnięcia ze strony http://sourceforge.net/projects/capa.
[21] Źródła programu *Crafty* możliwe są do ściągnięcia z ftp://ftp.cis.uab.edu/pub/hyatt/.

**Wyniki**

Zapis wszystkich rozegranych partii zamieszczono na końcu pracy w dodatku G. Wyniki zebrano w Tab. 5.10.

| Program | Wyniki |
|---|---|
| *Mobile Chess* | 1.0 : 0.0 (*Easy*), 1.0 : 0.0 (*Normal*), 1.0 : 0.0 (*Hard*) |
| *TalvMenni* | 1.5 : 0.5 |
| *Capa* | 1.0 : 1.0 |
| *Brutal Chess* | 0.5 : 1.5 |
| *Crafty* | 0.5 : 1.5 |

Tab. 5.10. Wyniki pojedynków najlepszej *strategii TDL* z wybranymi programami szachowymi

**Komentarz**

*Strategia TDL* najlepsze wyniki osiągnęła z programem *Mobile Chess* wygrywając z nim bez problemu wszystkie trzy partie na każdym poziomie trudności. Bardzo dobrze poradziła sobie również z programem *TalvMenni* wygrywając po wyrównanej grze 1.5:0.5. Chociaż w pierwszej partii granej białymi od początku wszystko wskazywało na przegraną *strategii TDL*, która w dwunastym posunięciu straciła hetmana kosztem dwóch lekkich figur przeciwnika. Jednak kilka posunięć później udało jej się odbić jeszcze jedną lekką figurę doprowadzając tym samym do bardzo interesującej gry środkowej, w której trudno mówić o przewadze którejś ze stron. Ostatecznie partia zakończyła się w niejasnej pozycji remisem spowodowanym trzykrotnym powtórzeniem pozycji. W drugiej rewanżowej partii granej przez *strategię TDL* kolorem czarnym początek również nie zapowiadał się ciekawie. Program *TalvMenni* o wiele lepiej rozgrywał grę w sensie pozycyjnym – zdobywając centrum oraz rozwijając swobodnie swoje figury. Około siedemnastego posunięcia *strategia TDL* doprowadziła do wątpliwej wymiany hetmana za gońca i wieżę. Jak się jednak później okazało dzięki szybkiej aktywacji figur czarne zaczęły zdobywać inicjatywę i po trzydziestym posunięciu doprowadziły do znaczącej przewagi, którą udało im się już wykorzystać. Zadowalający wynik został również osiągnięty przez *strategię TDL* z programem *Capa* – 1.0:1.0 (dwa remisy). W pierwszej partii rozegranej białymi po bardzo wyrównanej grze doszło ostatecznie do remisu w wyniku trzykrotnego powtórzenia pozycji. Po raz kolejny jednak *strategia TDL* pokazała, że bardzo źle rozgrywa początek partii w sensie pozycyjnym. *Capa* mogła w tej części pochwalić się znacznie lepszym ustawieniem strategicznym wszystkich swoich figur – nie wykorzystała ona jednak do końca swojej przewagi pozycyjnej. Druga partia potwierdziła poprzednie obserwacje. Ponownie *Capa* znacznie lepiej rozstawiała swoje figur, podczas gdy *strategia TDL* wędrowała swoim hetmanem po całej szachownicy oddając przy tym łatwym łupem jedną ze swoich lekkich figur. Przewaga materialna programu *Capa* utrzymywała się już do końca partii, jednak w całkowicie wygranej końcówce (król+wieża+skoczek przeciwko królowi) nie potrafiła zrealizować swojej przewagi, remisując ostatecznie w wyniku wykonania pięćdziesięciu posunięć bez bicia. Nieco gorsze wyniki zostały osiągnięte przez *strategię TDL* w pojedynkach z *Brutal Chess* oraz *Crafty* – oba pojedynki zakończyły się porażką 0.5:1.5. Chociaż w pierwszym przypadku, w grze z *Brutal Chess*, wydawało się, że *strategia TDL* ma szansę na wygraną. W pierwszej partii granej białymi z *Brutal Chess*, *strategia TDL* pomimo niezadowalającej gry w fazie początkowej, potrafiła zdobyć przewagę materialną jednej lekkiej figury, której niestety jednak nie zdołała wykorzystać i doprowadziła do remisu w wyniku

trzykrotnego powtórzenia pozycji. Podobnie w drugiej partii, granej czarnymi, *strategia TDL* mało ciekawie rozgrywała fazę początkową gry, co w tym przypadku zemściło się utratą hetmana w dwudziestym szóstym posunięciu, po którym bez najmniejszych problemów doszło już do zwycięstwa programu *Brutal Chess*. W przypadku gry z *Crafty*, pierwsza partia grana przez *strategię TDL* białymi zakończyła się zwycięstwem programu *Crafty*, który już w siódmym posunięciu zdobył przewagę materialną jednej lekkiej figury, a która następnie w dwudziestym posunięciu zamieniła się już w przewagę materialną jednej ciężkiej figury. Przewaga ta została ostatecznie wykorzystana. Kolejna partia to był ponownie pokaz nienajlepszej gry początkowej *strategii TDL*, która po niepotrzebnej wędrówce hetmana i wynikającemu stąd niedorozwoju figur, w czternastym posunięciu straciła za darmo skoczka. Pomimo znacznej przewagi pozycyjnej oraz materialnej programu *Crafty*, doszło do remisu wynikającego z trzykrotnego powtórzenia pozycji, którego przyczyną był brak planu gry ze strony *Crafty*.

Uzyskane w tej części wyniki pojedynków z wybranymi programami szachowymi (odpowiednio skonfigurowanymi) wskazują na niski poziom gry wyuczonej *strategii TDL*. Główną przyczyną słabej gry *strategii TDL* jest oczywiście płytkość rozwijanego drzewa gry. Rażącą wadą stylu gry *strategii TDL* jest początkowa faza jej gry, której daleko jest od poprawności strategicznej (pozycyjnej). Pomimo tego warto podkreślić, iż ze wszystkimi programami (odpowiednio skonfigurowanymi) grała na porównywalnym poziomie. Niestety z każdym programem przeprowadzono tylko dwa pojedynki różnymi kolorami. Zapewne przy większej liczbie partii udział procentowy zdobytych punktów mógłby znacznie odbiegać od uzyskanych tutaj wyników. Jednakże warto zauważyć, iż w większości przypadków programy szachowe wykazują deterministyczny styl gry, tzn. dla określonej pozycji wykonywane jest zawsze to samo posunięcie. Aby temu zapobiec, tylko niektóre programy wprowadzają niewielki szum losowy w swojej funkcji oceniającej w celu zapewnienia różnorodności gry takiego programu. Być może przeprowadzenie długodystansowych *treningów* sieci neuronowej (kilkanaście tysięcy partii) owocowałoby znacznie lepszymi wynikami w konfrontacji z innymi programami szachowymi. Niemniej jednak, najbardziej kluczowym składnikiem dobrze grającej strategii byłoby zwiększenie głębokości przeszukiwanego drzewa gry.

## 6. Podsumowanie pracy i dalsze kierunki jej rozwoju

W niniejszej pracy dokonano przeglądu inteligentnych metod wykorzystywanych w usprawnianiu programów szachowych. Wspomógł on wybór metody uczenia sieci neuronowej. Jest to metoda różnic czasowych (algorytm *Temporal Difference Learning*), która została zaimplementowana w autorskiej aplikacji *Neural Chess*. Sieć neuronowa została wykorzystana jako element oceniający stan gry, w poszukiwaniu optymalnej strategii gry w szachy. Zaproponowano szereg cech gry w szachy istotnych z punktu widzenia strategicznego, na które powinna była zwracać uwagę uczona sieć neuronowa. Na koniec przeprowadzono kilka serii eksperymentów badających wpływ wartości wielu parametrów związanych z uczeniem sieci neuronowej oraz jakość gry najlepszej znalezionej strategii.

Zaprezentowane wyniki eksperymentów dają nadzieję na odniesienie większych sukcesów przy ewentualnych dalszych pracach w przyszłości. W procesie dostrajania sieci neuronowej odkryto silną strategię radzącą sobie ze strategią treningową. Niestety nie osiągnięto jednak w całości głównego celu pracy, tzn. nie znaleziono optymalnej strategii gry, która radziłaby sobie w każdej sytuacji, z każdym graczem. Odkrytej strategii daleko jest do ideału, jak również do dobrze grających programów szachowych. Niemniej jednak uzyskane wyniki można uznać za zadowalające. Obie zaproponowane metody trenowania sieci (gra z inną strategią oraz obserwowanie partii z bazy danych) wydają się być bardzo dobrym sposobem poszukiwania najlepszego rozwiązania. Przegląd partii z bazy danych daje pożądaną różnorodność gry, której nie zapewnia ciągła gra z tą samą strategią. Ważny jest jednak staranny dobór tych partii, tak aby ich wyniki były adekwatne do powstającej na końcu pozycji. W części eksperymentalnej opisano nieco szerzej niekorzystne zjawisko zaszumienia wyników partii. Z kolei trening z jedną strategią może prowadzić do wyuczenia jedynie strategii lepszej od treningowej, co nie zapewnia od razu jej optymalności.

Proces dostrajania sieci neuronowej oraz ulepszania programu szachowego można byłoby kontynuować nadal. Jednak zważywszy na ograniczone ramy czasowe jak i samodzielną pracę w implementacji całego programu, należało potraktować pewne rzeczy bardziej powierzchownie. Na pewno pozostało wiele miejsc, które można byłoby dalej rozwijać. Implementacja oprogramowania ukierunkowana była w pierwszej kolejności na działanie i realizację pomysłów, a nie na ich optymalizację.

Największym mankamentem powstałego programu szachowego jest jego szybkość, która zadecydowała o znacznym spłyceniu przeszukiwania drzewa gry w celu przeprowadzenia na czas wszystkich doświadczeń i eksperymentów. Problem zwiększania szybkości i głębokości przeszukiwania drzewa gry od wielu lat leży w kręgu zainteresowań matematyków i informatyków. We współczesnych programach szachowych implementowanych jest wiele technik usprawniających ten proces, tj. tablice haszujące, heurystyka pustych ruchów, generowanie ruchów w najlepszym możliwym porządku, od którego może zależeć wcześniejsze podcinanie gałęzi drzewa, poszukiwanie pozycji stabilnych, heurystyki przeszukiwania selektywnego, rezygnującego z góry z przeszukiwania innych gałęzi drzewa, które niekoniecznie są gorsze [11].

Innym problemem jest także szybkość oceny pozycji, która w przypadku zastosowania sieci neuronowej jest złożonym procesem obliczeniowym na wartościach rzeczywistoliczbowych. We współczesnych programach funkcje oceniające stanowią najczęściej kombinację liniową pewnych przeskalowanych cech, bazującą na obliczeniach jedynie całkowitoliczbowych. Ważne jest zatem odpowiednie wyważenie długości czasu potrzebnego na ocenę określonej pozycji szachowej w stosunku do głębokości przeszukiwań drzewa gry. Przeprowadzone eksperymenty jednoznacznie wykazały znaczący wpływ głębokości rozwijania drzewa gry na poziom gry strategii.

Wielkim wyzwaniem jest cały proces nauki sieci neuronowej, który bardzo często jest procesem żmudnym i długotrwałym (złożoność obliczeń związanych z aktualizacją stanu sieci), uzależnionym niestety od wielu czynników. Szybkość znalezienia i trafność rozwiązania może często zależeć od zadanych warunków początkowych. W pracy starano się doświadczalnie dobrać możliwie najlepszy początkowy zestaw wartości wag na połączeniach sieci jak również parametrów α (współczynnik uczenia), β (współczynnik funkcji aktywacji neuronów), λ (współczynnik algorytmu TD). Duży wpływ na osiągane wyniki ma również długość trwania treningów, w których rozgrywanych było zaledwie kilkaset partii w porównaniu do kilkuset tysięcy, które na przykład rozgrywano przy nauce programu *TD-GAMMON*, będącego największym sukcesem zastosowania metody różnic czasowych do poszukiwania optymalnej strategii gry. Innym pomysłem jest również eksperymentowanie z wielowarstwową strategią złożoną z kilku sieci neuronowych odpowiedzialnych za poszczególne fazy gry, w najprostszym przypadku: rozpoczęcie, gra środkowa, końcówka. Można by dodatkowo rozdzielić zaproponowane cechy gry na różne jej fazy. Bardzo ciekawym pomysłem jest również podłączenie systemu uczącego do dowolnej internetowej kafejki szachowej, w której gra z szachistami daje pożądaną różnorodność stylów gry – zastosowanie tej techniki w praktyce mogłoby bardzo poprawić działanie systemu.

# Dodatek A. Neural Chess – Wartości Cech Gry

Aplikacja *Neural Chess – Wartości Cech Gry* (Rys. A.) służy do testowania wyznaczanych wartości cech gry zaproponowanych przez autora niniejszej pracy. Aplikacja umożliwia losowanie n-figurowej pozycji bądź ręczne ustawienie dowolnej pozycji, dla której chcemy zbadać wartości cech gry. Wartości cech gry odczytywane są z przewijanego okienka znajdującego się po prawej stronie.

*Losowanie pozycji* odbywa się poprzez wprowadzenie liczby całkowitej z zakresu od 2 do 64 w pole obok etykiety '*Liczba figur:*' i naciśnięcie przycisku '*Losuj*'.

*Ustawienie dowolnej pozycji* odbywa się poprzez wprowadzenie rozstawienia poszczególnych figur w polu poniżej etykiety '*Rozstawienie figur na szachownicy:*' i naciśnięcie przycisku '*Wybierz*'. Rozstawienie figur należy wprowadzić w algebraicznej notacji angielskiej rozdzielanej spacjami pomiędzy kolejnymi figurami – pierwsza wielka litera dotyczy nazwy figury białej, pierwsza mała litera dotyczy nazwy figury czarnej, natomiast następne dwie litery oznaczają odpowiednie pole na szachownicy.


Rys. A. Zrzut ekranu aplikacji *Neural Chess – Wartości Cech Gry*

## Dodatek B. Neural Chess – Eksperymenty

Aplikacja *Neural Chess – Eksperymenty* (Rys. B.) służy przede wszystkim do przeprowadzania eksperymentów z nauką sieci neuronowej w *treningu aktywnym*, ale również do testowania gry zaimplementowanych strategii w pojedynczej grze. W lewym górnym rogu aplikacji znajduje się szachownica, na której możliwy jest cały czas wgląd w aktualnie rozgrywaną partię. W prawym górnym rogu znajduje się panel do przeprowadzania pojedynczych rozgrywek pomiędzy różnymi graczami. Tuż poniżej znajduje się panel informacyjny przedstawiający na bieżąco informacje o ostatnio wykonanym ruchu, liczbie wykonanych ruchów od początku partii, liczbie wykonanych ruchów od ostatniego bicia mającego miejsce na szachownicy oraz ostatecznym wyniku partii. W dolnej części aplikacji znajduje się panel do przeprowadzania eksperymentów, w którym możliwe jest odpowiednie ustalenie parametrów związanych z *treningiem aktywnym* oraz bieżący wgląd w aktualne wyniki uczonej sieci neuronowej.


Rys. B. Zrzut ekranu aplikacji *Neural Chess – Eksperymenty*

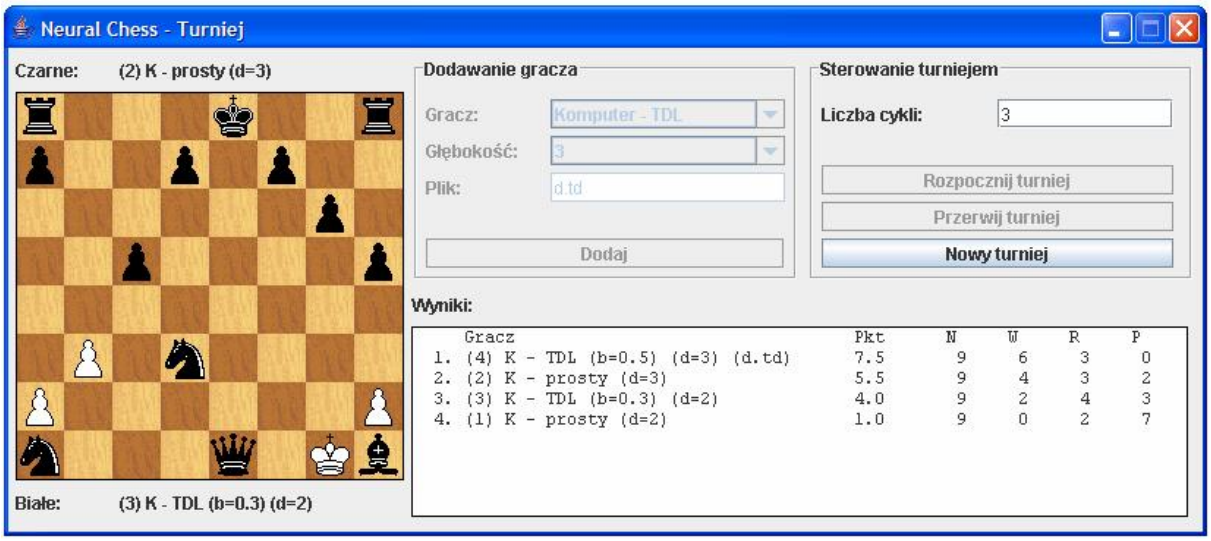
*Przeprowadzenie pojedynczej rozgrywki* odbywa się przy pomocy panelu znajdującego się w prawym górnym rogu. W celu przeprowadzenia pojedynku należy ustalić jego graczy poprzez ich wybranie z rozwijanych list znajdujących się obok etykiet '*Białe:*' oraz '*Czarne:*'. W grze może wziąć udział użytkownik, *strategia losowa*, *strategia uciekającego króla*, *strategia treningowa* (z głębokością przeszukiwania drzewa gry od dwóch do czterech pół-ruchów naprzód) albo *strategia TDL* (z głębokością przeszukiwania drzewa gry od dwóch do trzech pół-ruchów naprzód). Stan sieci neuronowej reprezentującej *strategię TDL* ustalany jest w następujący sposób: jeśli w katalogu, w którym znajduje się aplikacja znajduje się plik *d.td*, to stan sieci jest z niego pobierany, w przeciwnym przypadku wartości wag na połączeniach sieci neuronowej ustalane są wszędzie na wartość 0.01, z wyjątkiem wartości wag na połączeniach wychodzących z neuronów wejściowych otrzymujących informacje o cechach *siła pionów*, *siła skoczków*, *siła gońców*, *siła wież*, *siła hetmanów* oraz *mat*, które ustalane są odpowiednio na wartości 0.1, 0.3, 0.3, 0.5, 1.0 oraz 10.0. Dodatkowo wartość każdej wagi modyfikowana jest o wartość losową z rozkładu normalnego N(0.00,0.01). Po wybraniu odpowiednich graczy do pojedynku należy określić jego pozycję początkową: *wyjściową* albo *losową*. Po ustaleniu wszystkich wymienionych opcji i wciśnięciu przycisku '*Start*' dochodzi do rozpoczęcia pojedynczej rozgrywki. Od tej chwili grę można w każdej chwili spauzować (wciskając przycisk '*Pauza*') albo całkowicie zatrzymać (wciskając przycisk '*Stop*'). Można również dowolnie modyfikować czas odpowiedzi pomiędzy ostatnio zagranym na szachownicy ruchem a rozpoczęciem przeszukiwania drzewa gry przez kolejnego gracza poprzez wpisanie odpowiedniej liczby milisekund w pole obok etykiety '*Czas odpowiedzi:*'. W dowolnej chwili istnieje także możliwość odwracania szachownicy o 180 stopni poprzez wciśnięcie przycisku '*Odwróć szachownicę*'. Przeprowadzenie pojedynczej rozgrywki możliwe jest tylko i wyłącznie wtedy gdy nie przeprowadzany jest eksperyment.

*Przeprowadzenie eksperymentu* odbywa się przy pomocy panelu znajdującego się poniżej szachownicy. W celu przeprowadzenia eksperymentu należy przede wszystkim ustalić parametry uczenia: $\alpha$ (współczynnik uczenia), $\beta$ (nachylenie sigmoidy funkcji aktywacji neuronów) oraz $\lambda$ (parametr algorytmu uczenia *temporal difference learning*). Następnie należy wybrać przeciwnika, z którym będzie się uczyć grać *strategia TDL*. Może to być *strategia treningowa* (z głębokością przeszukiwania drzewa gry do dwóch lub trzech pół-ruchów naprzód), *strategia losowa*, *strategia uciekającego króla* albo sam użytkownik. Następnie należy zdefiniować liczbę gier potrzebnych do rozegrania w eksperymencie oraz sposób zmiany kolorów gry pomiędzy kolejnymi partiami. Na koniec należy określić głębokość przeszukiwania drzewa gry przez uczoną *strategię TDL* oraz ewentualnie nazwę pliku, do którego zapisywany będzie stan sieci po każdej rozegranej partii. Po ustaleniu wszystkich powyższych parametrów i wciśnięciu przycisku '*Rozpocznij trening*' dochodzi do rozpoczęcia eksperymentu. Od tej chwili wyniki eksperymentu na bieżąco uaktualniane są w miejscu tuż nad dolnymi przyciskami po każdej rozegranej partii. Znajdują się tutaj między innymi informacje o liczbie rozegranych dotychczas partii w eksperymencie, liczbie (oraz udziale procentowym) wygranych, zremisowanych oraz przegranych partii przez *strategię TDL*, a także liczbie (oraz udziale procentowym) zdobytych punktów. Wykresy na bieżąco przedstawiają zmiany udziału procentowego zdobywanych punktów – wykres czerwony określa udział procentowy zdobytych punktów ze wszystkich rozegranych do tej pory partii, a wykres szary określa udział procentowy zdobytych punktów z ostatnich 25 rozegranych partii. Oś pionowa określa udział procentowy punktów od 0 do 100%. Oś pozioma określa liczbę rozegranych partii od 0 do ustalonej liczby gier. Eksperyment może zostać w każdej chwili przerwany poprzez wciśnięcie przycisku '*Przerwij trening*'. W każdej chwili można również odwracać szachownicę o 180 stopni poprzez wciśnięcie przycisku '*Odwróć szachownicę*' oraz ustalać czas odpowiedzi pomiędzy ostatnio zagranym ruchem na

szachownicy a rozpoczęciem przeszukiwania drzewa gry przez kolejnego gracza poprzez wpisanie odpowiedniej liczby milisekund w pole obok etykiety '*Czas odpowiedzi:*'. Stan początkowy sieci neuronowej reprezentującej *strategię TDL* ustalany jest identycznie jak zostało to opisane w przypadku *przeprowadzania pojedynczej rozgrywki*. Przeprowadzenie eksperymentu możliwe jest tylko i wyłącznie wtedy gdy nie przeprowadzana jest pojedyncza rozgrywka.

## Dodatek C. Neural Chess – Turniej

Aplikacja *Neural Chess – Turniej* (Rys. C.) służy do przeprowadzania turniejów dla *strategii treningowej* oraz *strategii TDL*. W turnieju może brać udział dowolna liczba graczy (przynajmniej dwóch). Każda ze strategii może rozwijać drzewo gry od jednego do trzech pół-ruchów naprzód. Z lewej strony aplikacji znajduje się szachownica, na której zawsze przedstawiana jest aktualnie rozgrywana w turnieju partia. Poniżej oraz powyżej szachownicy znajdują się dodatkowo informacje o aktualnie grających graczach. Z prawej strony aplikacji znajdują się z kolei trzy panele: panel dodawania nowego gracza, panel sterowania turniejem oraz panel z aktualnymi wynikami turnieju.




Rys. C. Zrzut ekranu aplikacji *Neural Chess - Turniej*

*Przeprowadzenie turnieju* odbywa się poprzez zdefiniowanie odpowiednich parametrów turniejowych i wciśnięcie przycisku '*Rozpocznij turniej*'. Przed rozpoczęciem turnieju należy określić graczy w nim uczestniczących. Odbywa się to poprzez panel dodawania gracza. Gracz wybierany jest z listy rozwijanej znajdującej się obok etykiety '*Gracz:*'. Na liście rozwijanej znajduje się '*Komputer – prosty*' czyli *strategia treningowa* oraz '*Komputer – TDL*' czyli *strategia TDL*. Dla każdego gracza można zdefiniować głębokość rozwijania drzewa gry poprzez wybranie odpowiedniego poziomu z listy rozwijanej znajdującej się obok etykiety '*Głębokość:*'. W przypadku wyboru *strategii TDL* można dodatkowo określić plik, z którego ma zostać pobrany stan sieci neuronowej. Plik musi się znajdować w tym samym katalogu co aplikacja. Jeśli plik nie zostanie podany, to stan sieci neuronowej zostanie ustalony w następujący sposób: wszystkie wartości wag na połączeniach sieci neuronowej ustawiane są na 0.01, z wyjątkiem wartości wag na połączeniach wychodzących z neuronów wejściowych otrzymujących informacje o cechach *siła pionów*, *siła skoczków*, *siła gońców*, *siła wież*, *siła hetmanów* oraz *mat*, które ustawiane są odpowiednio na wartości 0.1, 0.3, 0.3, 0.5, 1.0 oraz 10.0. Dodatkowo wartość każdej wagi modyfikowana jest o wartość losową z rozkładu normalnego N(0.00,0.01). Po określeniu parametrów gracza należy dodać go do turnieju poprzez wciśnięcie przycisku '*Dodaj*'. Procedurę dodawania gracza należy przeprowadzić przynajmniej dwukrotnie, aby w turnieju uczestniczyło co najmniej dwóch graczy. Turniej rozgrywany jest na zasadzie 'każdy z każdym po n partii'. Liczbę partii do rozegrania pomiędzy każdymi dwoma graczami ustala się poprzez wpisanie odpowiedniej liczby w pole obok etykiety '*Liczba cykli:*'. Po zdefiniowaniu wszystkich powyższych parametrów i wciśnięciu przycisku '*Rozpocznij turniej*' dochodzi do wystartowania turnieju.

Do każdej kolejnej partii turnieju losowanych jest dwóch graczy ze zbioru wytypowanych przez użytkownika graczy zgodnie z ograniczeniem ‘każdy z każdym po n partii’. Po każdej partii uaktualniane są wyniki turnieju znajdujące się w prawej dolnej części aplikacji. Wyniki przedstawiają listę graczy posortowanych względem liczby zdobytych w turnieju punktów. W kolumnie *Gracz* znajdują się opisy graczy – w nawiasie podany jest numer startowy (kolejność wprowadzenia do turnieju przez użytkownika), następnie nazwa gracza oraz dodatkowe informacje dotyczące głębokości przeszukiwanego drzewa gry (np. (d=3)) w przypadku obu strategii, a także wartości parametru β (nachylenia sigmoidy funkcji aktywacji neuronów, np. (b=0.5) oraz ewentualnie nazwy pliku, z którego wczytano stan sieci neuronowej (np. (d.td)) w przypadku ***strategii TDL***. W kolejnych kolumnach znajdują się informacje o liczbie punktów (***Pkt***), liczbie rozegranych partii (***N***), liczbie wygranych partii (***W***), liczbie zremisowanych partii (***R***) oraz liczbie przegranych partii (***P***).

## Dodatek D. Neural Chess

Aplikacja *Neural Chess* (Rys. D.) służy do gry użytkownika ze *strategią TDL*, dla której stan sieci neuronowej wczytywany jest z pliku *d.td* znajdującego się w tym samym katalogu co aplikacja. W przypadku braku tego pliku aplikacja nie zostanie uruchomiona. Wraz z aplikacją dołączono plik *d.td*, w którym zapisany jest stan sieci neuronowej reprezentujący najlepszą wyłonioną podczas eksperymentów *strategię TDL*.


Rys. D. Zrzut ekranu aplikacji *Neural Chess*

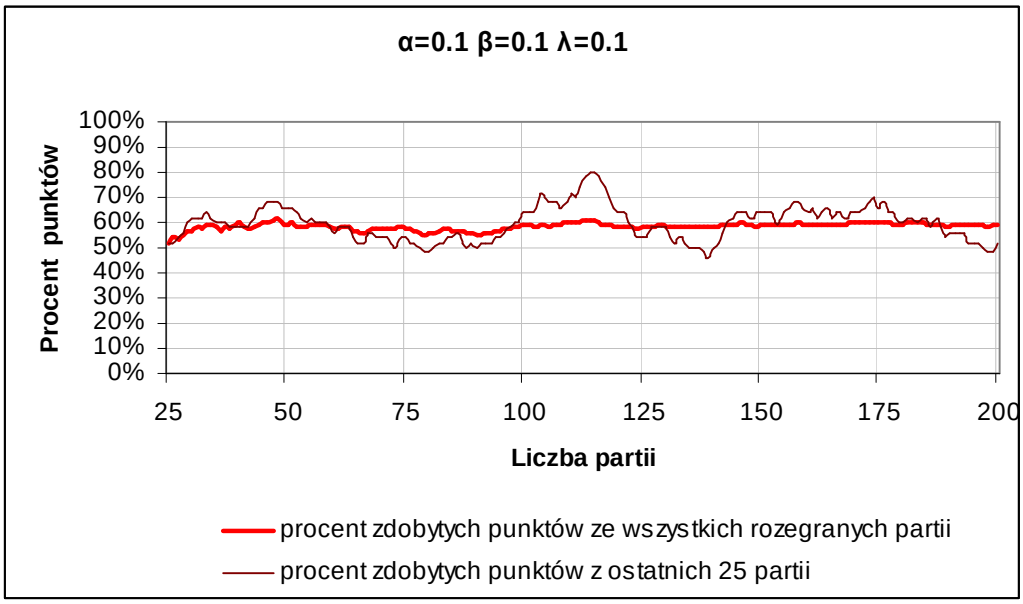
*Przeprowadzenie gry ze strategią TDL* odbywa się poprzez zdefiniowanie koloru gry użytkownika (z listy rozwijanej znajdującej się obok etykiety '*Kolor użytkownika*'), określenie pozycji początkowej (z listy rozwijanej znajdującej się obok etykiety '*Pozycja początkowa:*') – *wyjściowej* lub *losowej*, ustalenie głębokości przeszukiwania drzewa gry przez *strategię TDL* (z listy rozwijanej znajdującej się obok etykiety '*Głebokość:*') – od jednego do czterech pół-ruchów naprzód, i wciśnięcie przycisku '*Nowa gra*'. Po rozpoczęciu partii możliwe staje się jej przerwanie w dowolnej chwili (poprzez wciśnięcie przycisku '*Przerwij grę*'), jak również cofanie (przycisk '*Cofnij*') i powtarzanie (przycisk '*Powtórz*') dowolnej liczby ruchów wykonanych podczas partii. W dowolnym momencie można także odwrócić szachownicę o 180 stopni wciskając przycisk '*Odwróć strony*'.

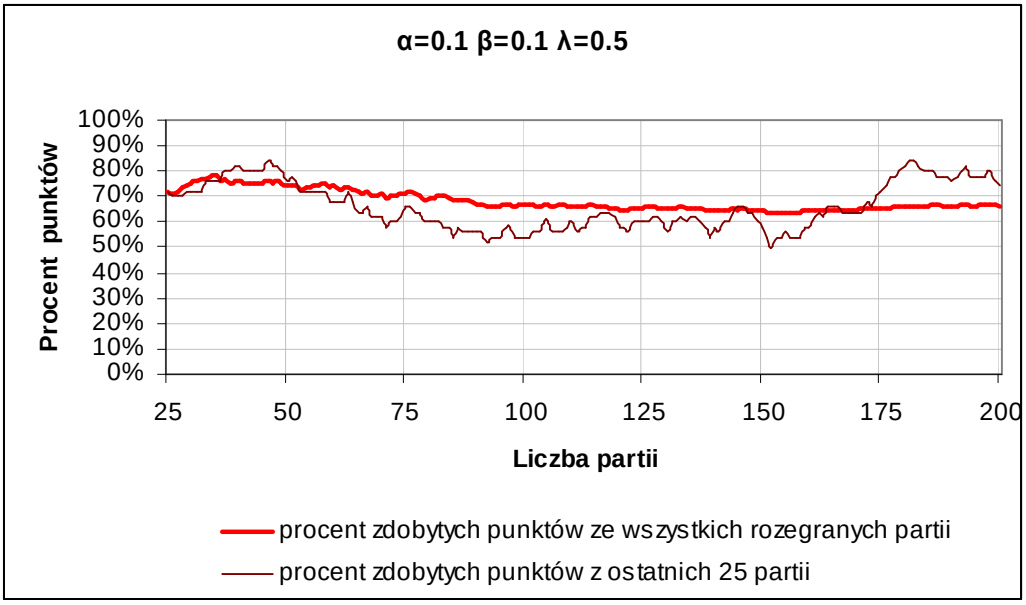
## Dodatek E. Wejście sieci neuronowej – numeracja wejść

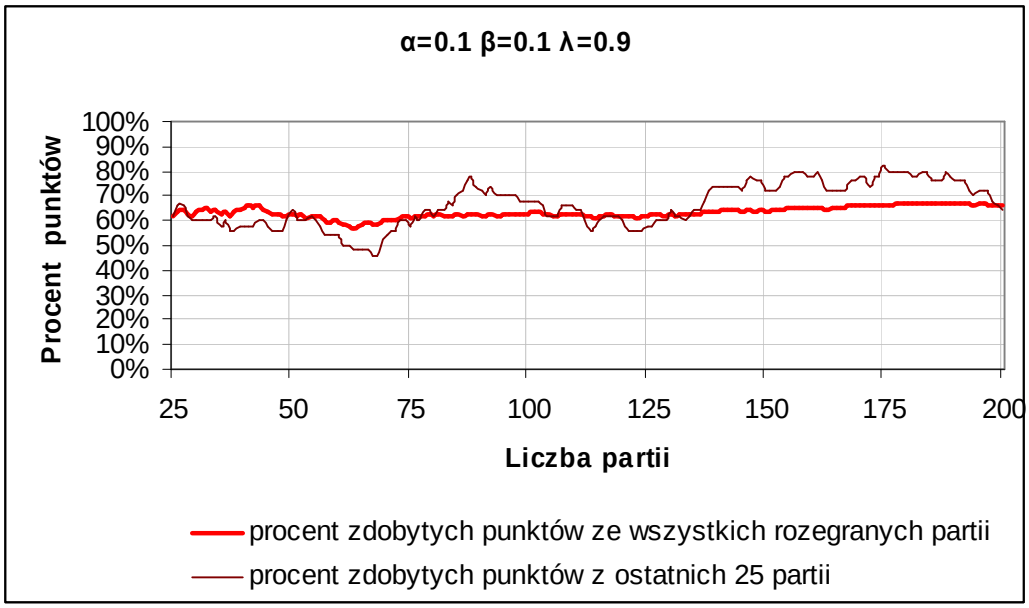
| Numer neuronu | Cecha gry |
|---|---|
| 0 | siła pionów |
| 1 | siła skoczków |
| 2 | siła gońców |
| 3 | siła wież |
| 4 | siła hetmanów |
| 5 | szach |
| 6 | szach z możliwą ucieczką króla |
| 7 | mat |
| 8 | kolor |
| 9 | bliskość figur do poziomej linii środkowej |
| 10 | bliskość figur do pionowej linii środkowej |
| 11 | aktywność figur |
| 12 | agresywność figur |
| 13 | przestrzeń figur |
| 14 | otwarte linie |
| 15 | otwarte wieże/hetmany |
| 16 | niebezpieczne wieże |
| 17 | wzmocnienie ciężkich figur w kolumnie |
| 18 | potencjalne przyczółki |
| 19 | przyczółki |
| 20 | związanie figur z królem/hetmanem |
| 21 | baterie |
| 22 | gońce Gorwitza |
| 23 | końskie widełki |
| 24 | opanowanie szachownicy |
| 25 | przewaga pionowa na skrzydłach |
| 26 | aktywność centralnych pionów |
| 27 | aktywność przycentralnych pionów |
| 28 | wzmocnienie pionowego centrum |
| 29 | wolne pionki |
| 30 | zaawansowanie wolnych pionków |
| 31 | połączone wolne pionki |
| 32 | oddalone wolne pionki |
| 33 | wzmocnienie wolnych pionków |
| 34 | wzmocnienie wolnych pionków przez wieże |
| 35 | blokada wolnych pionków przeciwnika |
| 36 | kandydaci na wolne pionki |
| 37 | zaawansowanie kandydatów na wolne pionki |
| 38 | wzmocnienie kandydatów na wolne pionki |
| 39 | izolowane pionki |
| 40 | zdwojone pionki |
| 41 | bliskość króli do centrum |
| 42 | bezpieczeństwo króli |
| 43 | królowie przewodnicy pionów |
| 44 | królewska blokada wolnego pionka |

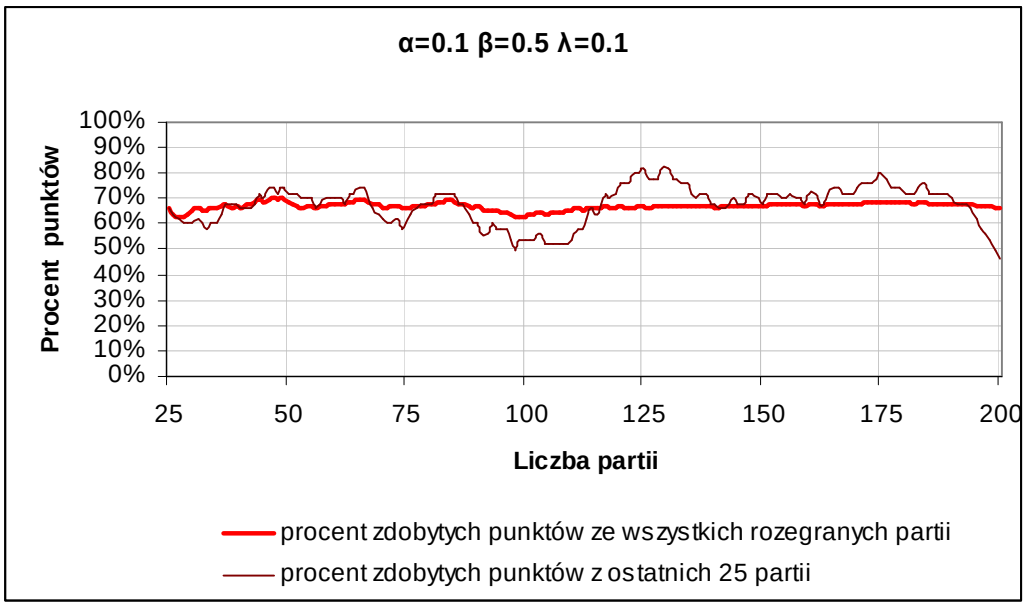
# Dodatek F. Wyniki eksperymentów badających wpływ wartości parametrów uczenia na jego efektywność

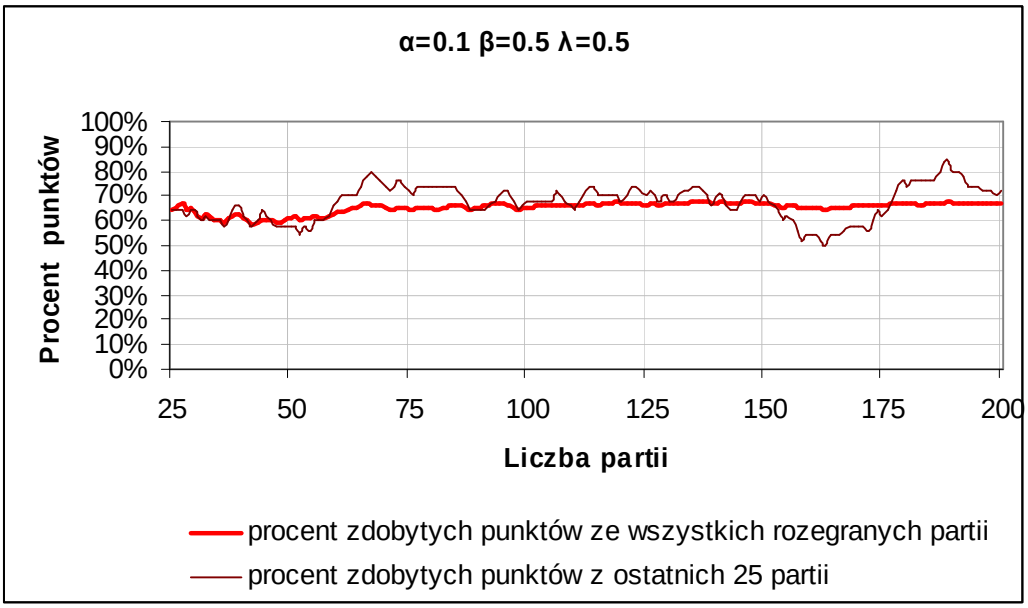
W niniejszym dodatku zebrano wyniki wszystkich 27 eksperymentów, które przeprowadzono w celu zbadania wpływu wartości parametrów α (współczynnik uczenia), β (stromość sigmoidy) oraz λ (parametr metody TD) na efektywność uczenia sieci neuronowej. Eksperymenty te zostały opisane w rozdziale 5.2. niniejszej pracy. Wyniki stanowią udział procentowy zdobywanych punktów przez *strategię TDL* w grze ze *strategią treningową*.

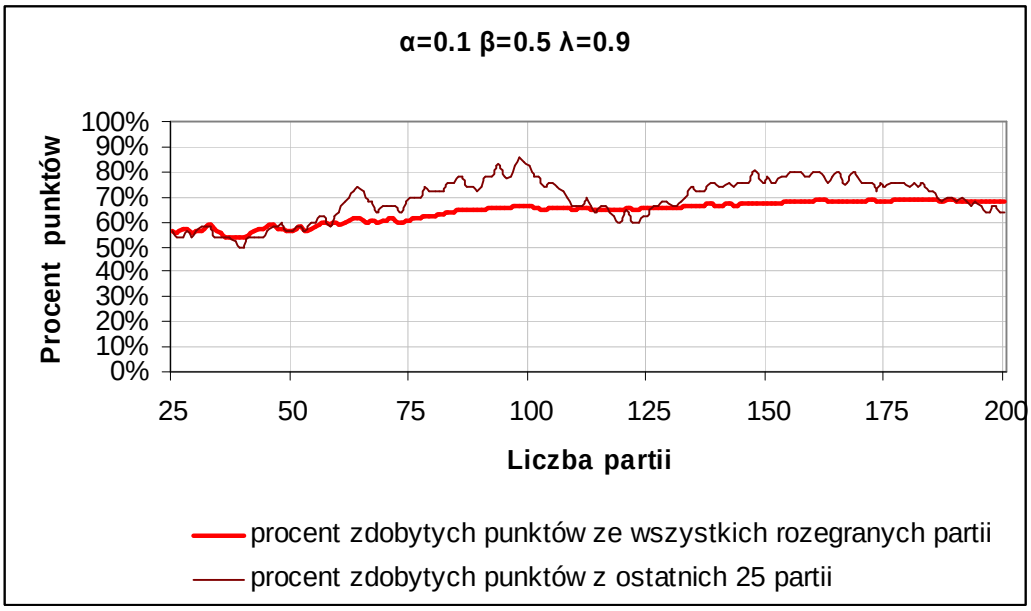


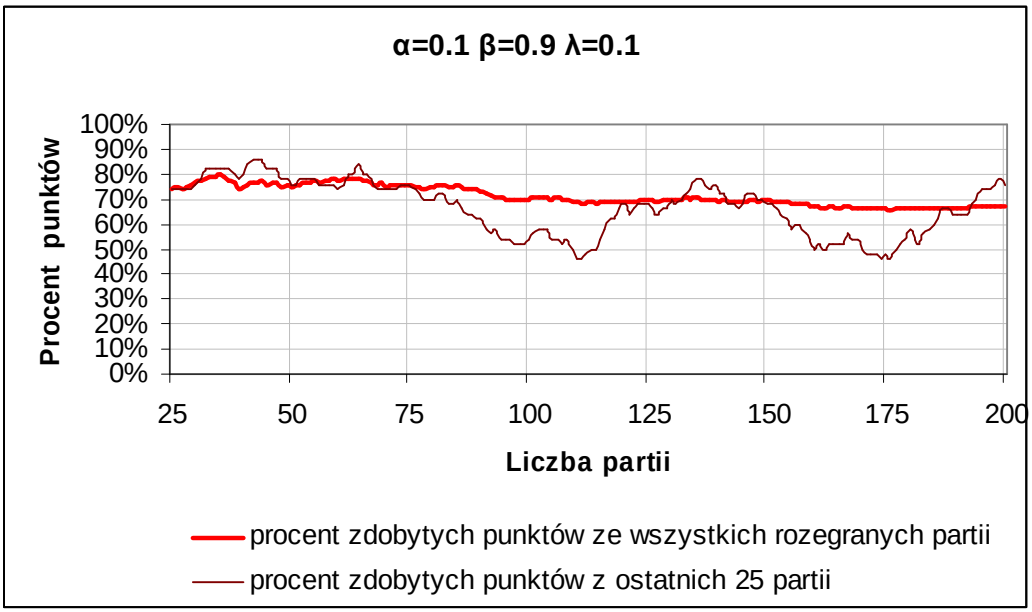



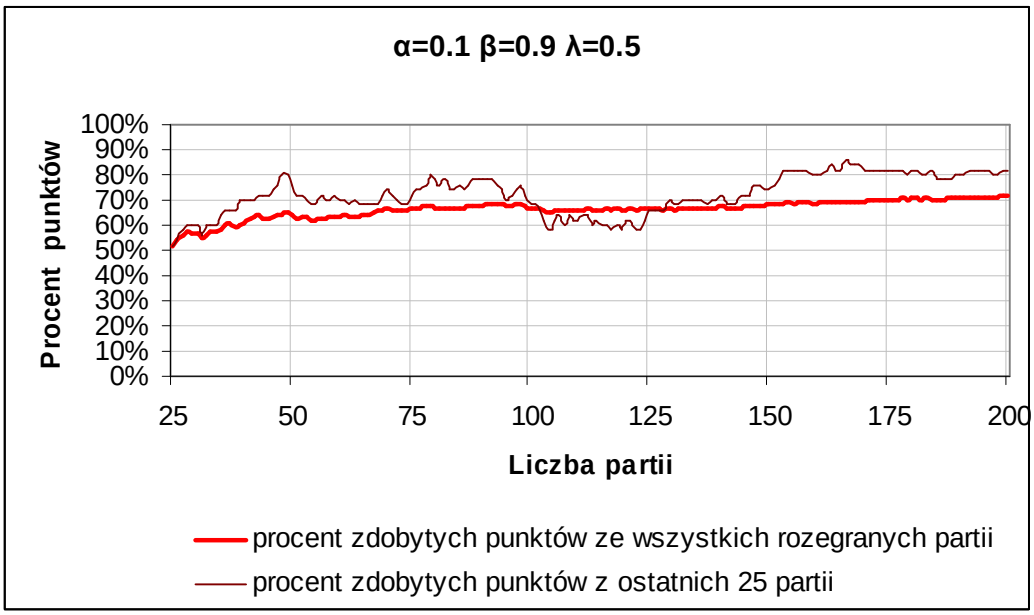

α=0.1 β=0.1 λ=0.9
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

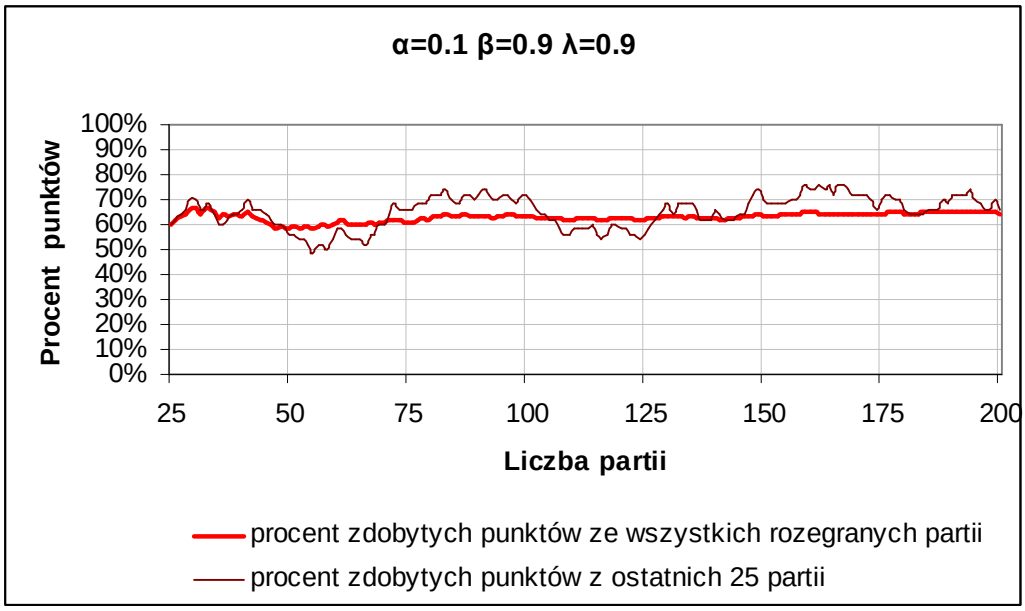

α=0.1 β=0.5 λ=0.1
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

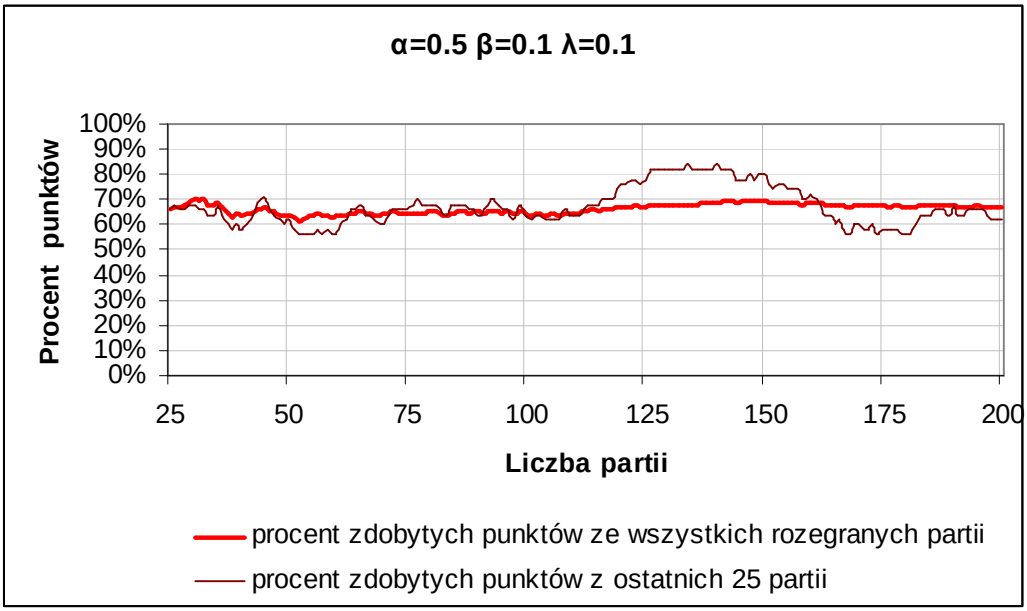

α=0.1 β=0.5 λ=0.5
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

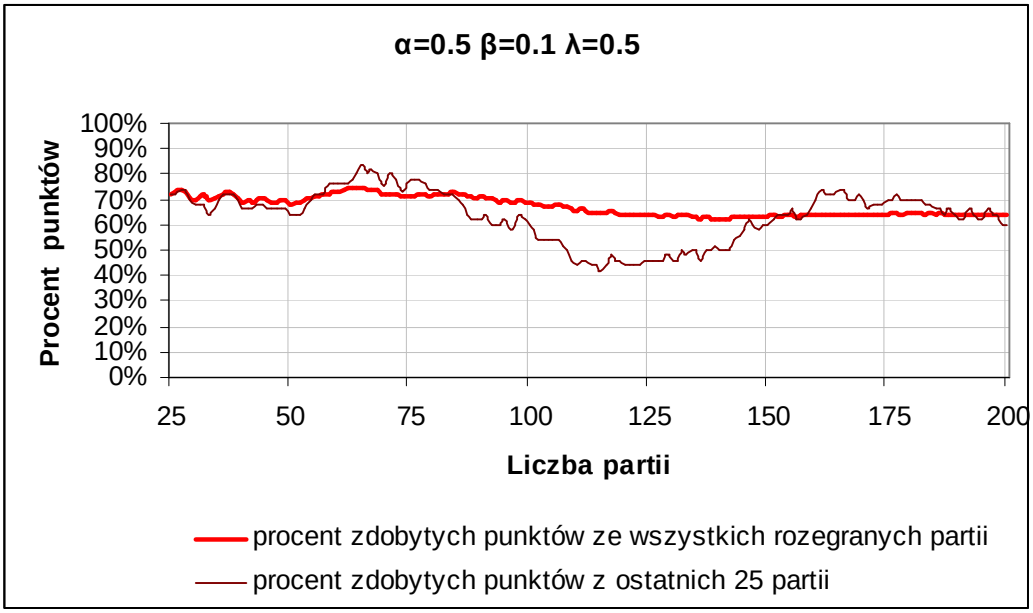

α=0.1 β=0.5 λ=0.9
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

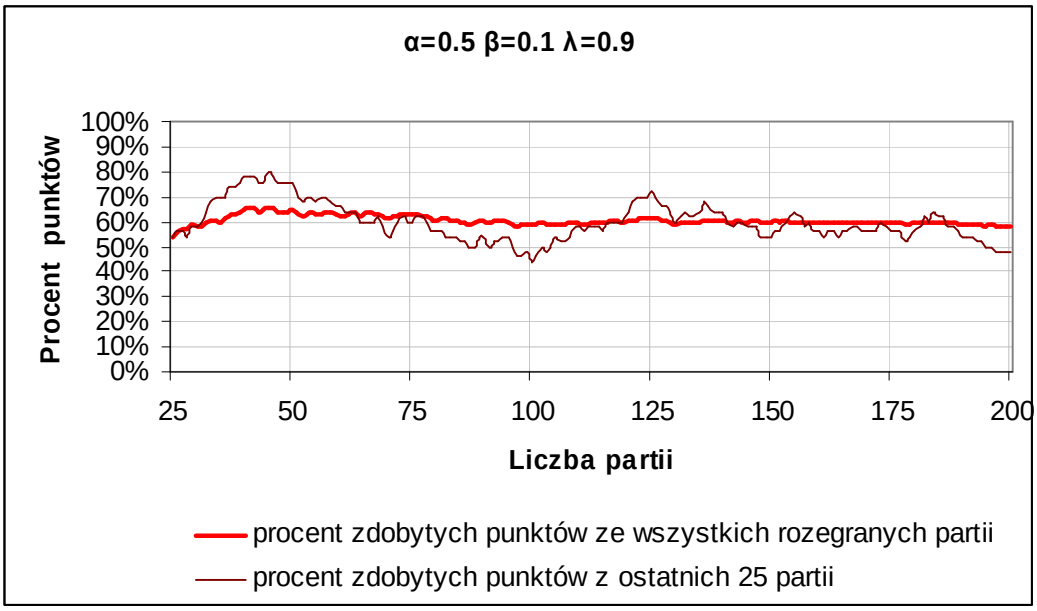

α=0.1 β=0.9 λ=0.1
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

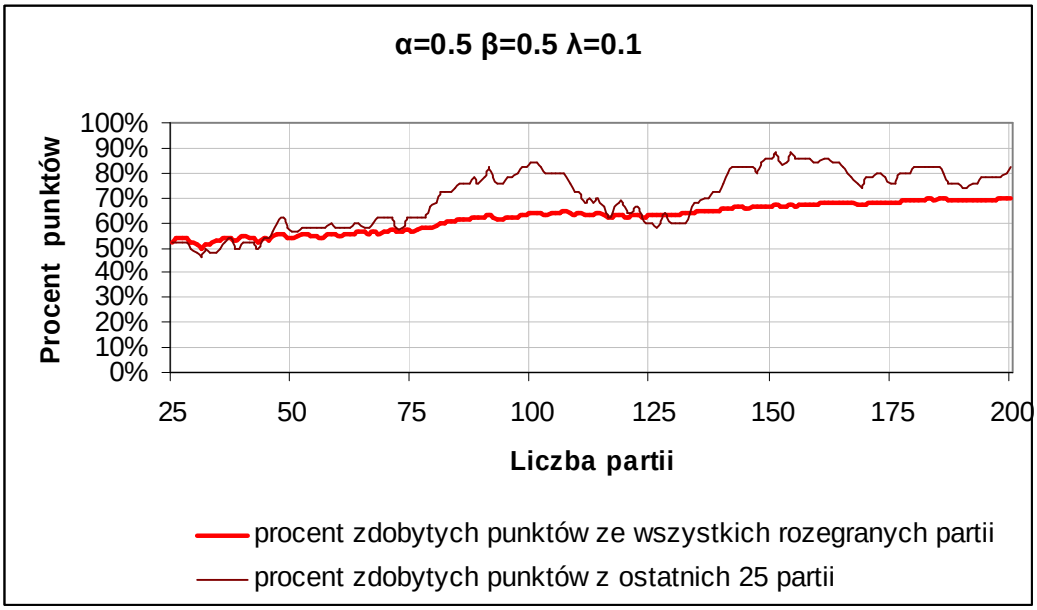

α=0.1 β=0.9 λ=0.5
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

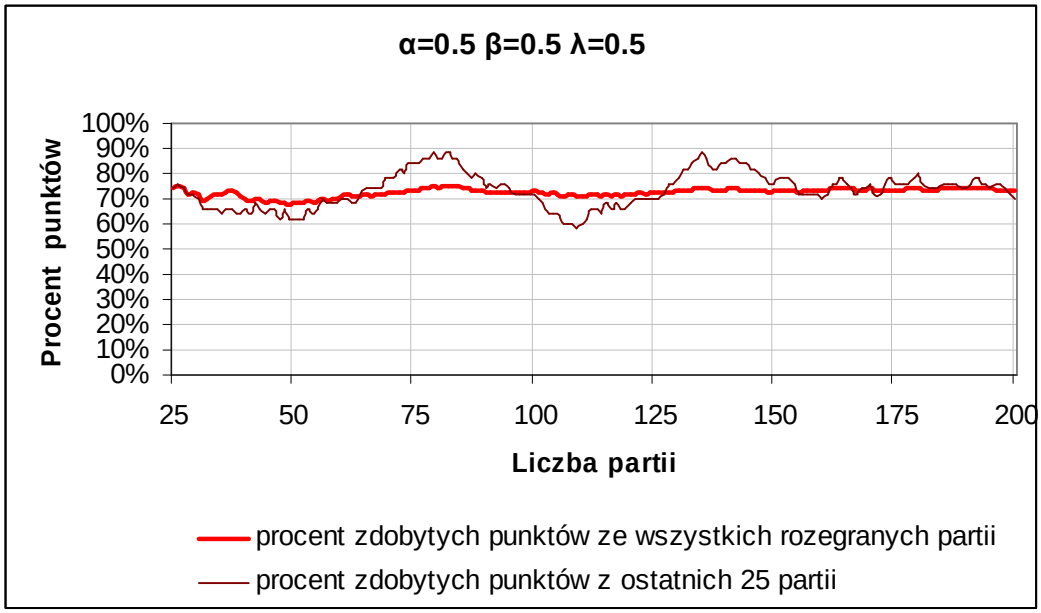

α=0.1 β=0.9 λ=0.9
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

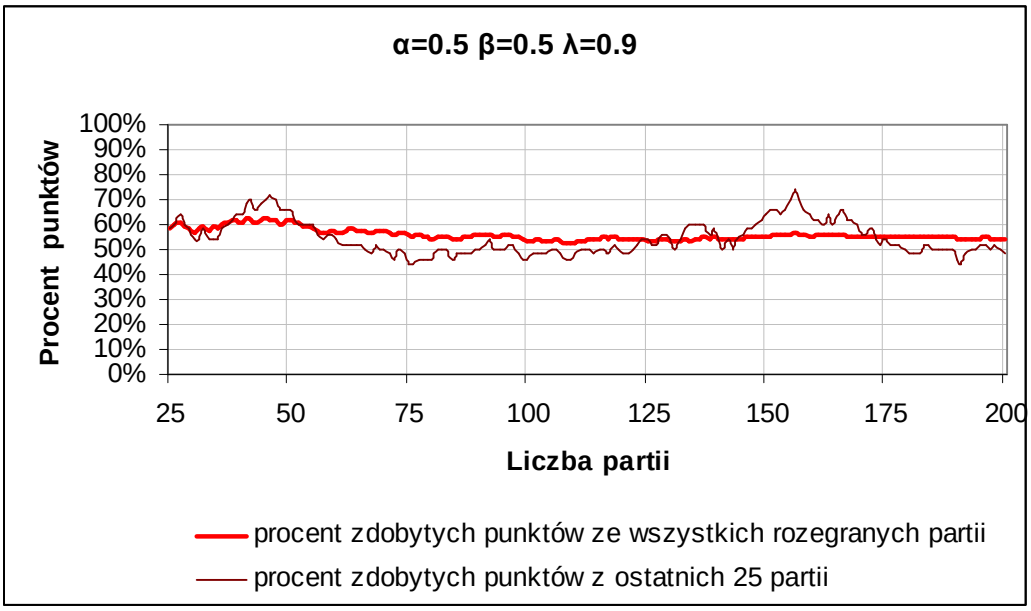

α=0.5 β=0.1 λ=0.1
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

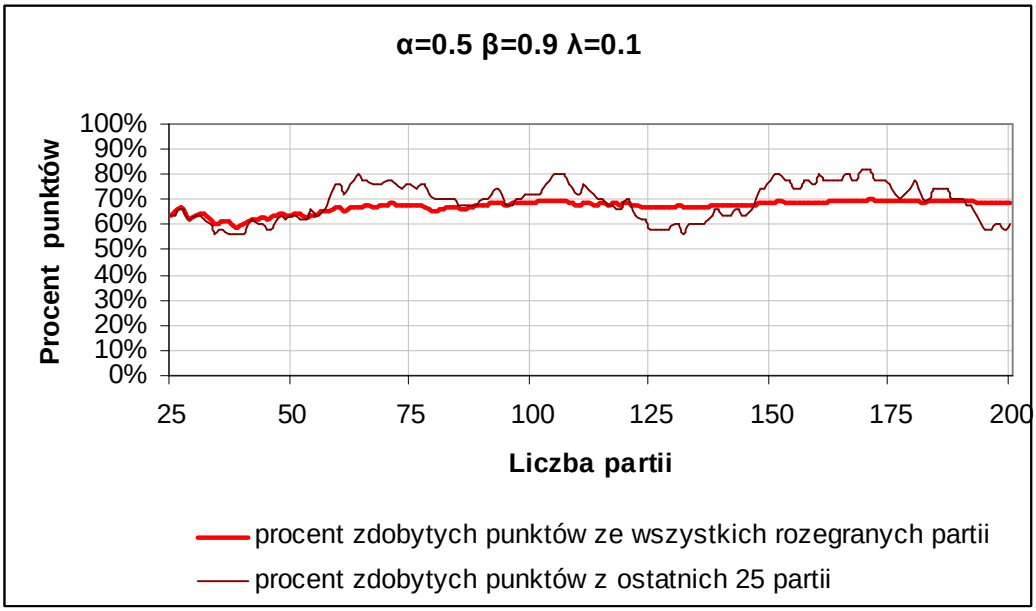

α=0.5 β=0.1 λ=0.5
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

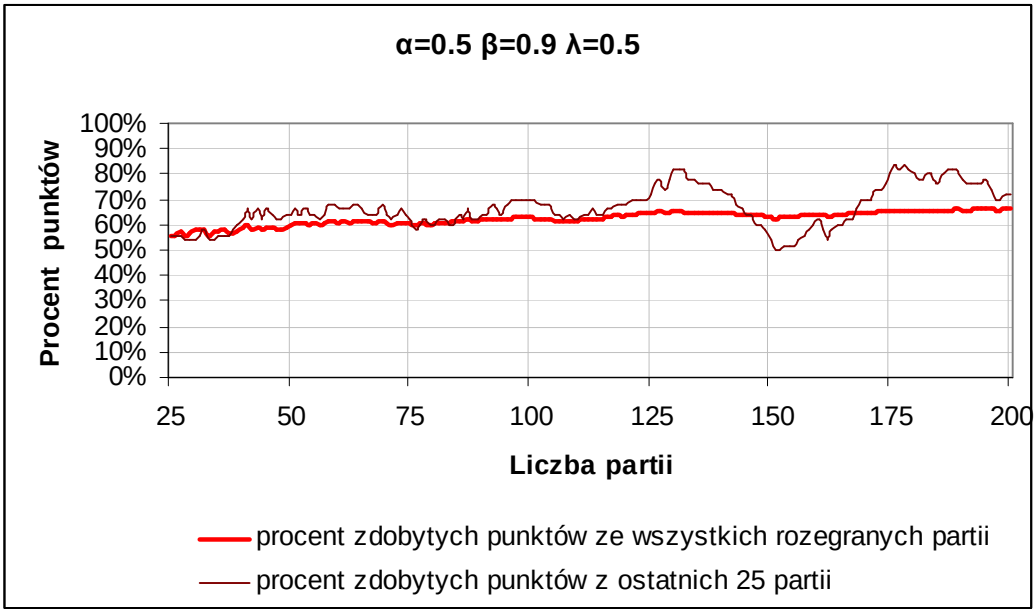

α=0.5 β=0.1 λ=0.9
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

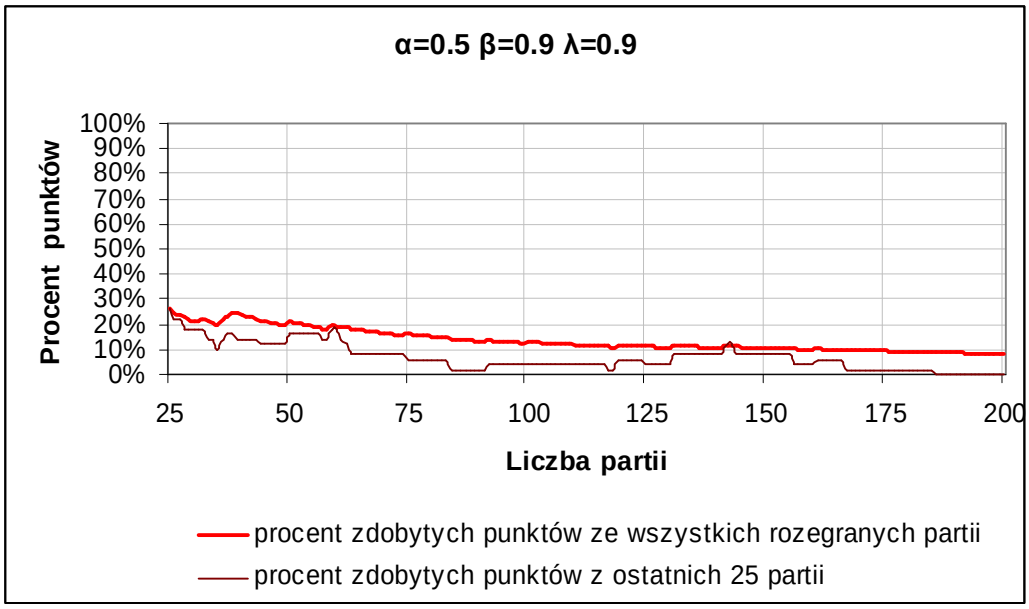

α=0.5 β=0.5 λ=0.1
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

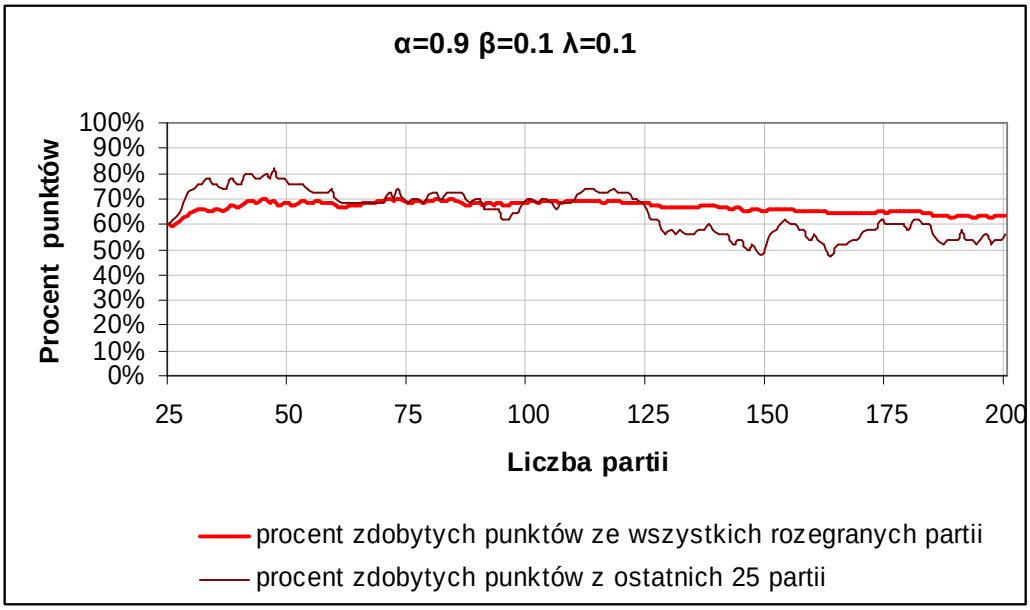

α=0.5 β=0.5 λ=0.5
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

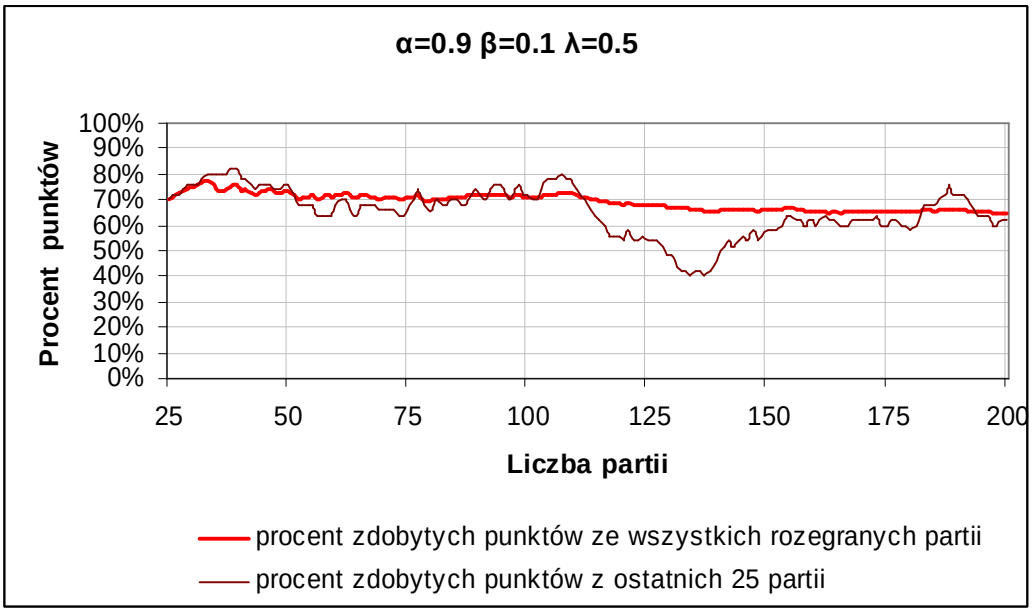

α=0.5 β=0.5 λ=0.9
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

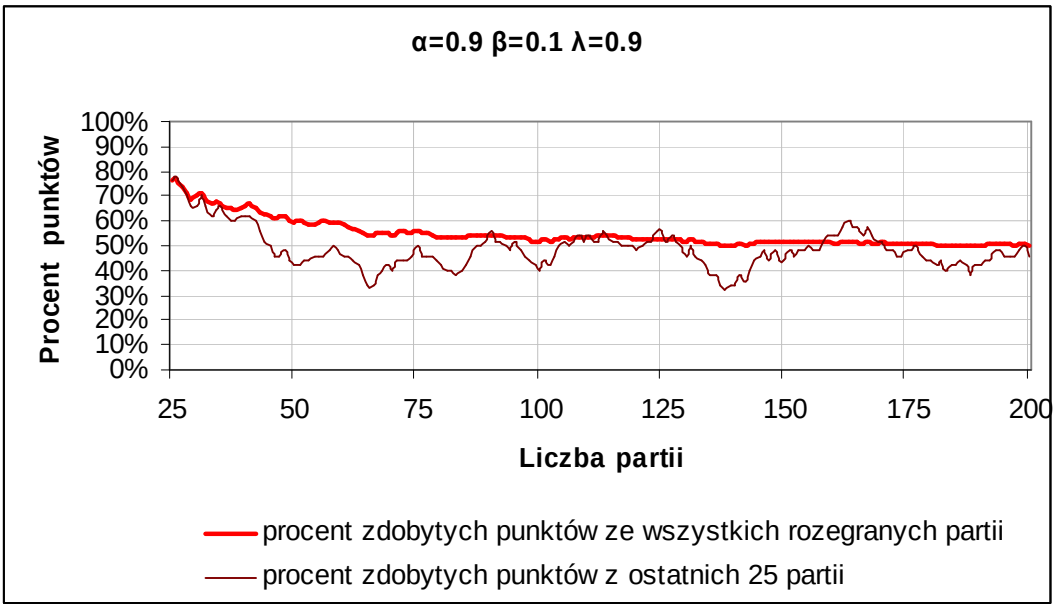

α=0.5 β=0.9 λ=0.1
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

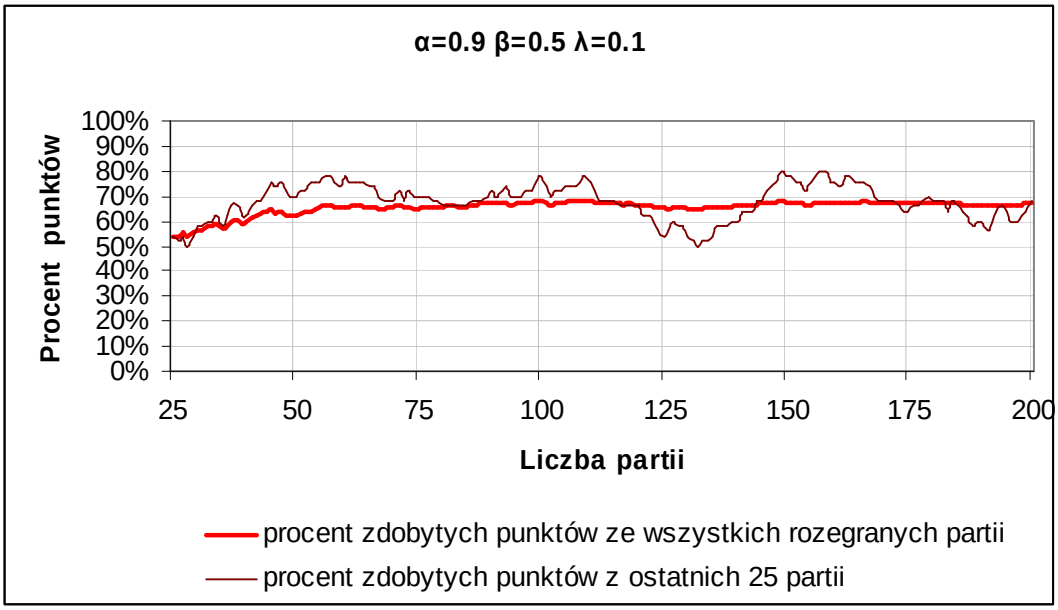

α=0.5 β=0.9 λ=0.5
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

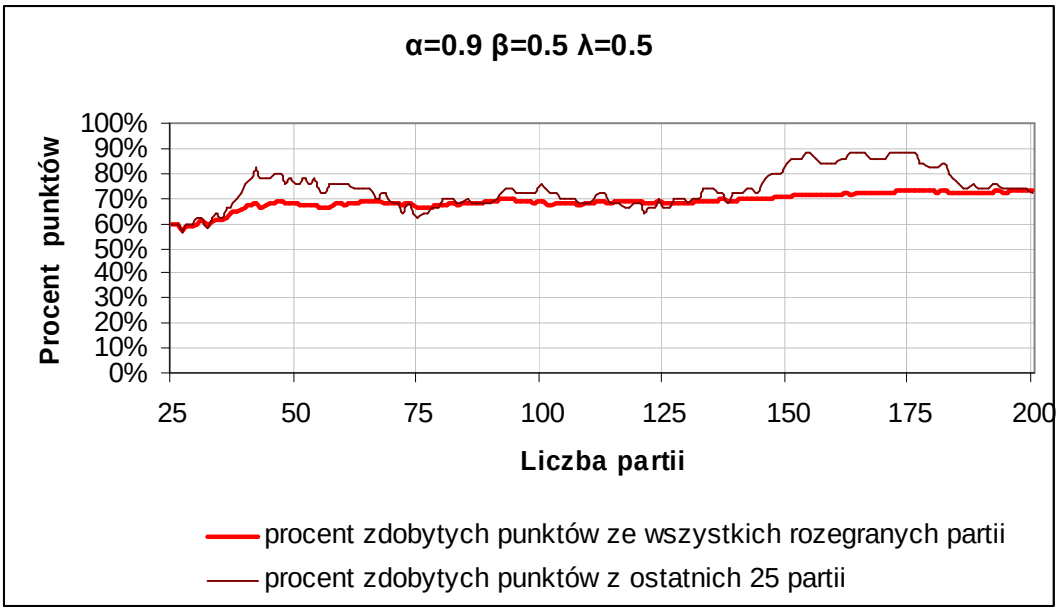

α=0.5 β=0.9 λ=0.9
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

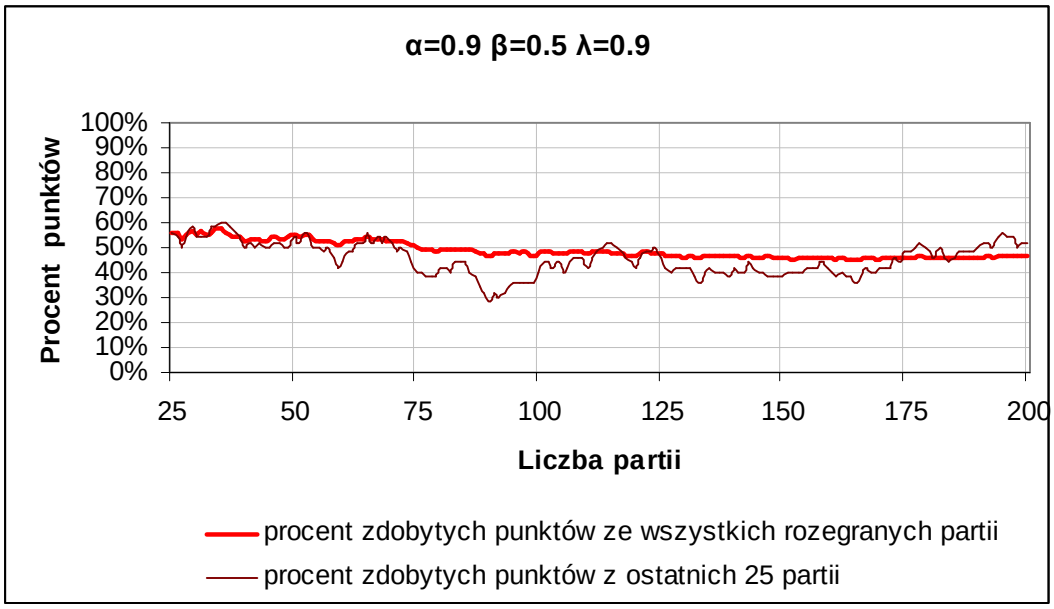

α=0.9 β=0.1 λ=0.1
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

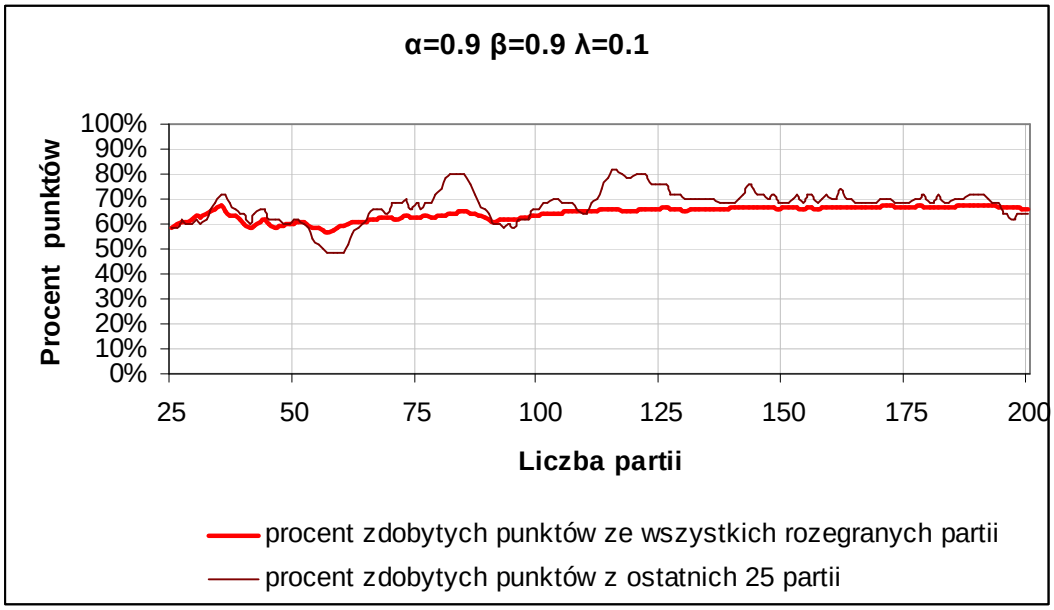

α=0.9 β=0.1 λ=0.5
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

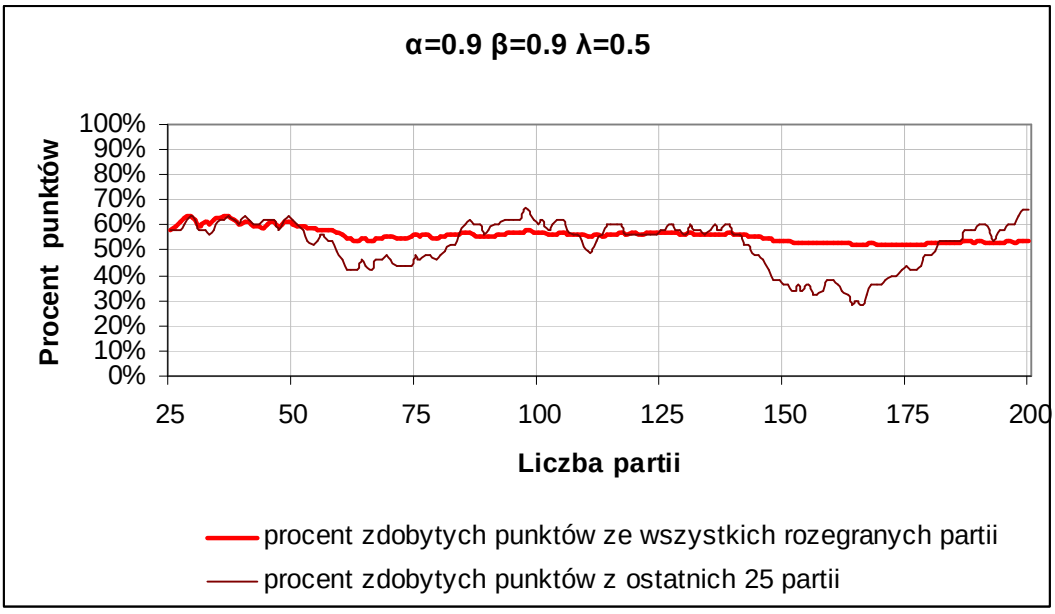

α=0.9 β=0.1 λ=0.9
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

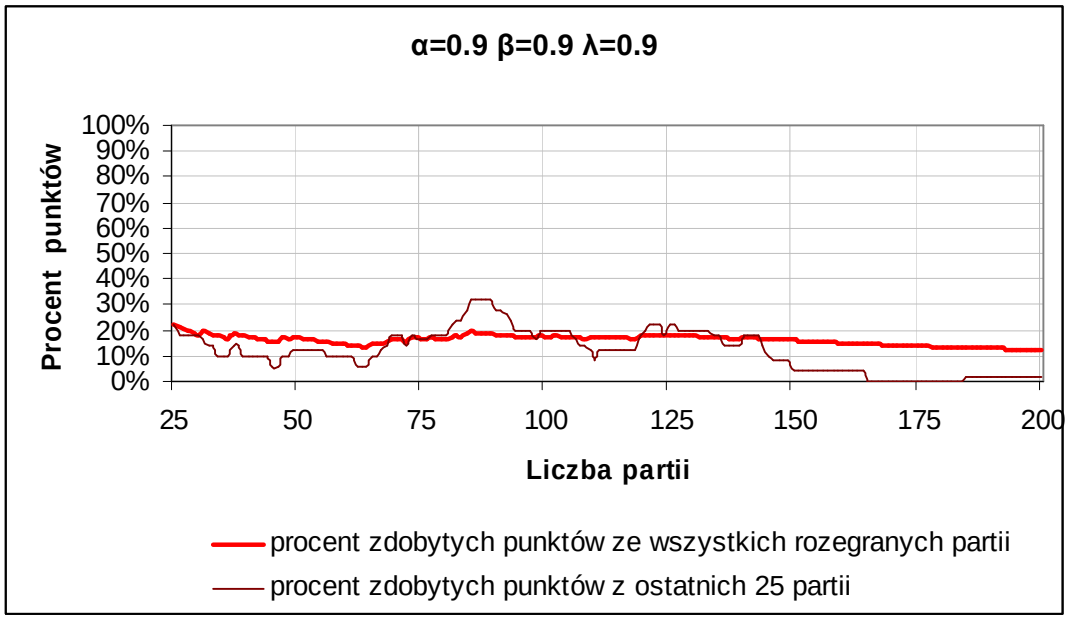

α=0.9 β=0.5 λ=0.1
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii


α=0.9 β=0.5 λ=0.5
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii


α=0.9 β=0.5 λ=0.9
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii


α=0.9 β=0.9 λ=0.1
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii


α=0.9 β=0.9 λ=0.5
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii


α=0.9 β=0.9 λ=0.9
Procent punktów
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
25
50
75
100
125
150
175
200
Liczba partii
procent zdobytych punktów ze wszystkich rozegranych partii
procent zdobytych punktów z ostatnich 25 partii

## Dodatek G. Partie rozegrane podczas konfrontacji

**Strategia TDL – Mobile Chess [Easy]**
1. e2-e3 d7-d5 2. Hd1-h5 a7-a5 3. Gf1-d3 c7-c5 4. Hh5-e5 h7-h5 5. Sb1-c3 a5-a4 6. Sc3-d5 h5-h4 7. Sd5-c7 Ke8-d7 8. Sc7-a8 b7-b6 9. He5-b8 g7-g5 10. Hb8-e5 e7-e6 11. He5-h8 f7-f5 12. Hh8-g8 e6-e5 13. Gd3-f5 Kd7-e8 14. Gf5-g6 Ke8-e7 15. Hg8-h7 Ke7-f6 16. Hh7-f7 **1:0**

**Strategia TDL – Mobile Chess [Normal]**
1. e2-e3 d7-d5 2. Hd1-h5 g7-g5 3. Hh5-g5 a7-a5 4. Gf1-e2 b7-b5 5. Ge2-d1 c7-c5 6. Hg5-e5 f7-f5 7. He5-h8 e7-e5 8. Gd1-h5 Ke8-e7 9. Hh8-h7 Gf8-g7 10. Hh7-g7 Ke7-e6 11. Hg7-f7 Ke6-d6 12. Hf7-g6 Sg8-f6 13. a2-a4 G5-a4 14. Wa1-a4 c5-c4 15. Sb1-c3 e5-e4 16. Sc3-d5 Kd6-d5 17. Gh5-e2 c4-c3 18. Wa4-d4 Kd5-c6 19. Wd4-d8 c3-d2 20. Gc1-d2 a5-a4 21. Wd8-c8 Kc6-b7 22. Hg6-f5 a4-a3 23. b2-a3 Wa8-a3 24. Gd2-c1 Wa3-e3 25. Gc1-e3 Sf6-h5 26. Ge2-h5 Sb8-a6 27. Hf5-d7 Sa6-c7 28. Hd7-c7 Kb7-a6 29. Gh5-e2 **1:0**

**Strategia TDL – Mobile Chess [Hard]**
1. e2-e3 d7-d5 2. Hd1-h5 g7-g5 3. Hh5-g5 a7-a5 4. Gf1-e2 b7-b5 5. Ge2-d1 c7-c5 6. Hg5-e5 f7-f5 7. He5-h8 e7-e5 8. Gd1-h5 Ke8-e7 9. Hh8-h7 Gf8-g7 10. Hh7-g7 Ke7-e6 11. Hg7-f7 Ke6-d6 12. d2-d4 c5-d4 13. e3-d4 e5-d4 14. Gc1-f4 Kd6-c5 15. Sg1-f3 a5-a4 16. Ke1-g1 a4-a3 17. Sb1-a3 Wa8-a3 18. b2-a3 b5-b4 19. a3-b4 Kc5-b4 20. Wa1-b1 Kb4-a4 21. Wb1-b8 Sg8-f6 22. Wb8-a8 Ka4-b4 23. Wf1-b1 Kb4-c5 24. Hf7-a7 Hd8-b6 25. Wa8-c8 **1:0**

**Strategia TDL – TalvMenni**
1. e2-e3 Sg8-f6 2. Gf1-d3 Sf6-d5 3. Hd1-f3 Sd5-b4 4. Gd3-f5 e7-e6 5. Hf3-e4 Gf8-e7 6. Gf5-g4 Ke8-g8 7. Gg4-d1 f7-f5 8. He4-c4 d7-d5 9. Hc4-b5 c7-c6 10. Hb5-a4 b7-b5 11. Ha4-a3 Sb4-c2 12. Gd1-c2 Ge7-a3 13. Sb1-a3 Hd8-a5 14. Ke1-f1 Wf8-f6 15. d2-d4 Ha5-b4 16. Sg1-e2 e6-e5 17. d4-e5 Wf6-e6 18. Gc2-f5 We6-e5 19. Gf5-c8 Hb4-c5 20. Gc8-g4 b5-b4 21. f2-f4 We5-e4 22. Gg4-f5 We4-e7 23. Sa3-c2 h7-h6 24. Gf5-d3 Sb8-d7 25. a2-a3 b4-a3 26. b2-b4 Hc5-d6 27. Wa1-a3 Hd6-e6 28. Sc2-d4 He6-g4 29. Sd4-c6 We7-e6 30. h2-h3 Hg4-h5 31. Sc6-d4 We6-f6 32. g2-g4 Hh5-e5 33. Kf1-g2 He5-c7 34. Sd4-b5 Hc7-c6 35. Sb5-a7 Hc6-e6 36. Gc1-b2 Wf6-f7 37. e3-e4 d5-e4 38. Gd3-c2 g7-g6 39. Wh1-a1 Kg8-h7 40. Se2-d4 He6-e7 41. Sd4-c6 He7-e6 42. Sc6-d4 He6-e7 43. Sd4-c6 He7-e6 44. Sc6-d4 **0.5:0.5**

**TalvMenni – Strategia TDL**
1. Sg1-f3 e7-e6 2. Sf3-e5 Gf8-d6 3. Se5-c4 Gd6-f4 4. e2-e3 d7-d5 5. e3-f4 d5-c4 6. Gf1-c4 Hd8-d4 7. Gc4-b5 Sb8-c6 8. Ke1-g1 Hd4-e4 9. d2-d3 He4-b4 10. Sb1-c3 Gc8-d7 11. a2-a3 Hb4-c5 12. Gc1-e3 Sc6-d4 13. Ge3-d4 Hc5-d4 14. Sc3-e2 Hd4-b2

15. Wa1-b1 Hb2-a3 16. Wb1-b3 Ha3-c5 17. d3-d4 Hc5-b5
18. Wb3-b5 Gd7-b5 19. d4-d5 Ke8-c8 20. f2-f3 Sg8-e7 21. Wf1-f2
Wd8-d5 22. Se2-d4 Gb5-a4 23. Wf2-e2 Wh8-d8 24. Hd1-d3 Ga4-b5
25. c2-c4 Wd5-d4 26. Hd3-b3 Gb5-c4 27. Hb3-a4 Gc4-e2
28. Ha4-c2 Wd4-d1 29. Kg1-f2 Wd8-d2 30. Hc2-h7 Ge2-d3
31. Kf2-g3 Gd3-h7 32. h2-h4 Se7-g6 33. h4-h5 Sg6-e7 34. Kg3-h2
Se7-d5 35. Kh2-g3 Gh7-f5 36. h5-h6 g7-h6 37. Kg3-h2 Sd5-f4
38. Kh2-g3 e6-e5 39. Kg3-h4 Wd1-h1 40. Kh4-g3 Wd2-g2 **0:1**

**Strategia TDL – Capa 0.1 beta 2 [Easy]**
1. e2-e3 e7-e5 2. Hd1-h5 Sb8-c6 3. Gf1-c4 d7-d5 4. Gc4-e2
Sg8-f6 5. Hh5-h4 Gc8-f5 6. Ge2-d1 Gf5-e4 7. f2-f3 Ge4-f5
8. d2-d3 Gf8-b4 9. Ke1-f1 Hd8-d6 10. Sg1-e2 Hd6-c5 11. a2-a4
b7-b6 12. Wa1-a2 Gf5-g6 13. d3-d4 e5-d4 14. e3-d4 Hc5-e7
15. Gc1-f4 Gg6-h5 16. c2-c3 Gb4-a5 17. Wh1-g1 Gh5-g6 18. b2-b4
Gg6-b1 19. Wa2-b2 Gb1-d3 20. b4-a5 b6-a5 21. Hh4-g5 He7-e6
22. Hg5-g7 Wh8-g8 23. Hg7-h6 Wg8-g6 24. Hh6-h4 h7-h6 25. c3-c4
Gd3-c4 26. Gf4-h6 Gc4-a6 27. Gh6-d2 Ke8-c8 28. Hh4-f4 Sf6-h5
29. Hf4-h4 Sh5-f6 30. Hh4-f4 Sf6-h5 31. Hf4-h4 Sh5-f6
32. Hh4-f4 **0.5:0.5**

**Capa 0.1 beta 2 [Easy] – Strategia TDL**
1. d2-d4 e7-e6 2. c2-c4 Hd8-h4 3. Sb1-c3 Gf8-d6 4. Sg1-f3
Hh4-f6 5. Gc1-g5 Hf6-f5 6. e2-e4 Hf5-g4 7. e4-e5 Gd6-e5
8. d4-e5 Sb8-c6 9. Hd1-e2 Sc6-d4 10. Sf3-d4 Hg4-g5 11. Sc3-b5
Ke8-d8 12. He2-e4 Sg8-e7 13. Gf1-d3 a7-a6 14. h2-h4 Hg5-h5
15. g2-g4 Hh5-h6 16. Sb5-c3 c7-c5 17. Sd4-b3 Se7-c6 18. Sb3-c5
a6-a5 19. g4-g5 Hh6-h5 20. Gd3-e2 Hh5-g6 21. He4-f4 b7-b6
22. Ge2-d3 Hg6-h5 23. Gd3-e2 Hh5-g6 24. Ge2-d3 Hg6-h5
25. Sc5-e4 Gc8-b7 26. Se4-g3 Sc6-e5 27. Sg3-h5 Se5-d3
28. Ke1-d2 Sd3-f4 29. Sh5-f4 Wa8-c8 30. Wh1-e1 Wc8-c4
31. Kd2-e3 e6-e5 32. Sf4-d3 Wh8-e8 33. Ke3-d2 Gb7-c6
34. Sd3-e5 Wc4-d4 35. Kd2-c2 Kd8-c8 36. g5-g6 h7-g6 37. h4-h5
g6-h5 38. b2-b4 a5-b4 39. Se5-c6 b4-b3 40. Kc2-b2 Wd4-d2
41. Kb2-b3 We8-e1 42. Sc6-a7 Kc8-b8 43. Wa1-e1 Kb8-a7
44. Sc3-b5 Ka7-b7 45. f2-f4 Wd2-d3 46. Kb3-c2 Wd3-d5
47. Sb5-c3 Wd5-c5 48. We1-e7 Kb7-c8 49. We7-f7 g7-g6
50. Wf7-f8 Kc8-b7 51. Wf8-f7 Kb7-c8 52. Wf7-f8 Kc8-b7
53. Wf8-f6 Wc5-c4 54. a2-a4 d7-d5 55. Wf6-g6 h5-h4 56. Wg6-g7
Kb7-c8 57. Wg7-g8 Kc8-d7 58. Wg8-g7 Kd7-e6 59. Wg7-g6 Ke6 f5
60. Wg6-b6 Kf5-f4 61. Wb6-f6 Kf4-e5 62. Wf6-f7 Ke5-e6
63. Wf7-g7 Ke6-f6 64. Wg7-d7 h4-h3 65. Wd7-d5 h3-h2 66. Wd5-h5
Wc4-a4 67. Sc3-a4 h2-h1H 68. Wh5-h1 Kf6-f7 69. Wh1-h7 Kf7-g6
70. Wh7-e7 Kg6-f6 71. We7-d7 Kf6-g5 72. Wd7-g7 Kg5-f6
73. Wg7-d7 Kf6-f5 74. Wd7-f7 Kf5-e6 75. Wf7-c7 Ke6-d6
76. Wc7-f7 Kd6-e6 77. Wf7-c7 Ke6-d6 78. Wc7-b7 Kd6-c6
79. Wb7-b6 Kc6-c7 80. Kc2-b1 Kc7-d8 81. Wb6-d6 Kd8-e7
82. Wd6-d5 Ke7-e6 83. Sa4-c3 Ke6-e7 84. Wd5-e5 Ke7-f6
85. We5-d5 Kf6-e6 86. Kb1-a1 Ke6-e7 87. Wd5-e5 Ke7-f6
88. We5-d5 Kf6-e6 89. Wd5-d4 Ke6-f6 90. Sc3-d5 Kf6-e6
91. Sd5-f4 Ke6-e5 92. Wd4-c4 Ke5-d6 93. Wc4-d4 Kd6-e5

94. Wd4-c4 Ke5-d6 95. Wc4-e4 Kd6-c5 96. We4-e5 Kc5-d4
97. We5-d5 Kd4-e4 98. Wd5-d7 Ke4-f4 99. Wd7-f7 Kf4-e5
100. Wf7-e7 Ke5-d6 101. We7-f7 Kd6-e6 102. Wf7-c7 Ke6-d6
103. Wc7-f7 Kd6-e6 104. Wf7-c7 Ke6-d6 105. Wc7-b7 Kd6-c6
106. Wb7-e7 Kc6-d6 107. We7-g7 Kd6-e6 108. Wg7-g6 Ke6-f5
109. Wg6-g7 Kf5-f6 110. Wg7-d7 Kf6-e6 111. Wd7-b7 Ke6-d6
112. Wb7-b6 Kd6-d5 113. Wb6-b5 Kd5-d6 114. Wb5-b6 Kd6-d5
115. Wb6-b5 Kd5-d6 116. Wb5-b7 Kd6-d5 117. Wb7-d7 Kd5-e6
118. Wd7-b7 Ke6-d6 119. Wb7-g7 Kd6-e5 120. Wg7-g5 Ke5-f6
121. Wg5-d5 Kf6-e6 122. Wd5-d4 Ke6-e5 123. Wd4-c4 Ke5-d5
124. Wc4-c3 Kd5-d4 125. Wc3-c7 Kd4-e5 126. Wc7-c5 Ke5-d6
127. Wc5-c4 Kd6-d5 128. Wc4-c3 Kd5-d4 129. Wc3-c7 Kd4-e5
130. Wc7-c5 Ke5-d6 131. Wc5-c3 Kd6-d5 132. Wc3-d3 Kd5-e5
133. Wd3-e3 Ke5-d5 134. We3-d3 Kd5-e5 135. Wd3-e3 Ke5-d5
136. We3-f3 Kd5-e4 137. Wf3-f7 Ke4-d5 138. Wf7-f5 Kd5-e6
139. Wf5-f4 Ke6-e5 140. Wf4-f3 Ke5-e4 141. Wf3-f7 Ke4-e5
142. Wf7-g7 Ke5-f6 143. Wg7-b7 Kf6-e6 144. Wb7-a7 Ke6-d6
145. Wa7-a6 Kd6-c5 146. Wa6-a5 Kc5-c6 147. Wa5-a6 Kc6-c5
148. Wa6-a5 Kc5-c6 **0.5:0.5**

**Strategia TDL – Brutal Chess 0.5.2 [Easy]**
1. e2-e3 d7-d5 2. Hd1-h5 Sg8-f6 3. Hh5-e5 Sb8-c6 4. Gf1-b5
Gc8-d7 5. Gb5-c6 Gd7-c6 6. h2-h3 Sf6-e4 7. d2-d4 Hd8-c8
8. Sg1-e2 Gc6-b5 9. He5-d5 Gb5-c6 10. Hd5-e5 Hc8-d7 11. d4-d5
Gc6-d5 12. He5-d4 Hd7-b5 13. Sb1-c3 Hb5-a5 14. Hd4-d5 Ha5-b4
15. a2-a3 Se4-c3 16. Se2-c3 Hb4-b6 17. Sc3-a4 Hb6-a6
18. Sa4-c3 Ha6-b6 19. Sc3-a4 Hb6-a6 20. Sa4-c3 Ha6-b6 **0.5:0.5**

**Brutal Chess 0.5.2 [Easy] – Strategia TDL**
1. d2-d4 e7-e6 2. Gc1-f4 Hd8-h4 3. Gf4-c7 Gf8-b4 4. c2-c3
Gb4-e7 5. Sg1-f3 Hh4-h5 6. Hd1-a4 b7-b5 7. Ha4-b3 Sb8-c6
8. e2-e4 Ke8-f8 9. Gf1-b5 Sc6-a5 10. Gc7-a5 Wa8-b8 11. Ga5-c7
Wb8-b5 12. Hb3-c4 Wb5-b2 13. Gc7-f4 Gc8-b7 14. Sb1-d2 Gb7-c6
15. g2-g4 Hh5-g4 16. Ke1-c1 Ge7-a3 17. Gf4-d6 Ga3-d6 18.
Kc1-b2 Gc6-b7 19. Hc4-b3 Gb7-c6 20. Kb2-b1 Hg4-g2 21. Hb3-c4
Hg2-f2 22. Wd1-f1 Hf2-e3 23. Wf1-e1 He3-f2 24. We1-e2 Gc6-d5
25. Hc4-c8 Kf8-e7 26. We2-f2 Gd5-c6 27. c3-c4 e6-e5 28. d4-e5
Gd6-c5 29. Wf2-e2 f7-f6 30. e5-f6 Ke7-f6 31. e4-e5 Kf6-e7
32. e5-e6 Ke7-d6 33. e6-e7 Sg8-e7 34. Hc8-h8 Kd6-c7 35. Hh8-g7
Gc5-d6 36. We2-e7 h7-h5 37. We7-e6 Gd6-f4 38. Kb1-b2 Gc6-b7
39. Kb2-c3 h5-h4 40. Kc3-d4 h4-h3 41. Hg7-e7 a7-a5 42. He7-c5
Kc7-b8 43. We6-e8 Gb7-c8 44. We8-c8 Kb8-b7 45. Wh1-b1 Kb7-a6
46. Hc5-b6 **1:0**

**Strategia TDL – Crafty 20.14 (sd 3)**
1. e2-e3 Sg8-f6 2. Gf1-d3 e7-e5 3. Hd1-f3 Gf8-d6 4. Sg1-e2
Ke8-g8 5. Hf3-f5 g7-g6 6. Hf5-g5 e5-e4 7. Gd3-e4 Sf6-e4
8. Hg5-d5 Hd8-h4 9. g2-g3 Hh4-g4 10. h2-h3 Hg4-f3 11. Ke1-g1
c7-c6 12. Hd5-d4 Se4-g5 13. Hd4-d6 Sg5-h3 14. Kg1-h2 Hf3-e2
15. Kh2-g2 Wf8-e8 16. Sb1-c3 He2-g4 17. f2-f3 Hg4-h5
18. Wf1-h1 We8-e6 19. Wh1-h3 Hh5-h3 20. Kg2-h3 We6-d6

21. d2-d4 Wd6-f6 22. Kh3-g2 d7-d5 23. Gc1-d2 Gc8-e6 24. Wa1-h1 Kg8-g7 25. f3-f4 Ge6-f5 26. e3-e4 d5-e4 27. Gd2-e3 Wf6-e6 28. Kg2-f2 f7-f6 29. Wh1-h4 h7-h5 30. Kf2-e2 Sb8-d7 31. d4-d5 c6-d5 32. Sc3-d5 Wa8-c8 33. Sd5-c3 Sd7-b6 34. g3-g4 Gf5-g4 35. Ke2-f1 Sb6-c4 36. Ge3-d4 Sc4-b2 37. Kf1-g1 e4-e3 38. Sc3-d5 e3-e2 39. Gd4-f6 We6-f6 40. Kg1-f2 Wc8-c2 41. Sd5-f6 Sb2-d3 42. Kf2-e3 e2-e1H 43. Ke3-d3 He1-d1 44. Kd3-e3 Wc2-e2 **0:1**

**Crafty 20.14 (sd 3) – Strategia TDL**

1. d2-d4 e7-e6 2. Gc1-d2 Hd8-h4 3. Sg1-f3 Hh4-e4 4. Sb1-c3 He4-g4 5. h2-h3 Hg4-h5 6. e2-e4 Gf8-e7 7. Hd1-e2 Ge7-d8 8. Ke1-c1 Sb8-c6 9. He2-c4 Sc6-a5 10. Hc4-a4 c7-c6 11. g2-g4 Hh5-g6 12. Sf3-e5 Hg6-f6 13. Sc3-e2 Hf6-f2 14. Gd2-a5 Gd8-g5 15. Ga5-d2 Gg5-e7 16. Se2-f4 b7-b5 17. Ha4-a5 Sg8-f6 18. Gf1-d3 Hf2-d4 19. Gd2-c3 Hd4-e3 20. Gc3-d2 He3-d4 21. Gd2-c3 Hd4-e3 22. Gc3-d2 He3-d4 **0.5:0.5**

## Literatura


[1] Bain M. E. *Learning Logical Exceptions in Chess*. Praca doktorska. Department of Statistics and Modelling Science, Univeristy of Strathclyde. Szkocja. 1994.

[2] Baxter J., Tridgell A., Weaver L. *Learning To Play Chess Using Temporal Differences*. Kulwer Academic Publisher. Netherlands. 2001.

[3] Campbell M. *An Enjoyable Game: How HAL plays chess. HAL`s Legacy: 2001`s Computer as a Dream Reality.* MIT Press, 1997. http://mitpress.mit.edu/e-books/Hal/

[4] Flann N. S. *Learning appropriate abstractions for planning in formation problems.* W *Proceedings of the 6th International Workshop on Machine Learning*, s. 235-239. Morgan Kaufmann. 1989.

[5] Flann N.S., Dietterich T.G. *A study of explanation-based methods for inductive learning*. W *Machine Learning 4*, s. 187-226.

[6] Friedel F. *A short history of computer chess.* http://www.chessbase.com/columns/column.asp?pid=102.

[7] Friedel F. *Cheating in Chess.* Advances in Computer Chess 9. Computer Science Department, Maastricht University. 2001.

[8] Fürnkranz Johannes. *Machine Learning in Computer Chess: The Next Generation*. Austrian Research Institute for Artificial Intelligence. Wien 1996.

[9] Kerner Y. *Learning strategies for explanation patterns: Basic game patterns with application to chess.* W *Proceedings of the 1st International Conference on Case-Based Reasoning*. Berlin. Springer-Verlag. 1995.

[10] Krulwich B. T. *Flexible Learning in a Multi-Component Planning System*. Praca doktorska. The Institute for the Learning Sciences. Northwestern University. 1993.

[11] Marsland T.A. *Computer Chess and Search*. Computing Science Department. University of Alberta. Edmonton. 1991.

[12] Michniewski T. *Samouczenie programów szachowych*. Praca magisterska. Wydział Matematyki i Mechaniki Uniwersytetu Warszawskiego. 1995.

[13] Minton S. *Constraint based generalization: Learning game playing plans from single examples.* W *Proceedings of the 2nd National Conference on Artificial Intelligence*, s. 251-254. Austin, TX. 1984.

[14] Morales E. *Learning features by experimentation in chess*. W *Proceedings of the 5th European Working Session on Learning*, s. 494-511. Springer Verlag. 1991.

[15] Newell A., Shaw J. C., Simon H. A. *Chess-Playing Programs and the Problem of Complexity*. IBM Journal. 1958.

[16] Nimzowitsch A. I. *Mój system*. MARABUT. Tychy. 1995.

[17] Nunn J. *Secrets of Rook Endings*. Batsford. 1992.

[18] Nunn J. *Secrets of Panwless Endings.* Batsford. 1994.

[19] Quinlan J. R. *Learning efficient classification procedures*. W Michalski R. S., Carbonell J. G., Mitchell T. M. *Machine Learning: An Artificial Intelligence Approach*, s. 463-482. Palo Alto: Tioga. 1983.

[20] Pitrat J. *A program to learn to play chess*. W *Pattern Recognition and AI*, s. 399-419. Academic Press. 1976.

[21] Pitrat J. *Realization of a program learning to find combinations at chess*. W *Computer Oriented Learning Process*. Noordhoff. 1976.

[22] Puget J. F. *Goal regression with opponent*. W *Progress in Machine Learning*, s. 121-137. Sigma Press. 1987.

[23] Randell B. *From Analitycal Engine to Electronic Digital Computer: The Contributions of Ludgate, Torres, and Bush*. MIT. 1980. http://www.cs.ncl.ac.uk/research/pubs/articles/papers/398.pdf.

[24] Russel S. J., Noriv P. *Artificial Intelligence – A Modern Approach*. Prentice-Hall. A Simon & Schuster Company. Englewood Cliffs. New Jersey. 1995.
[25] Riesbeck C. K., Schank R. C. *Inside Case-Based Reasoning*. Hillsade, NJ. Lawrence Erlbaum Associates. 1989.
[26] Scherzer T. L., Tjaden D. *Learning in BEBE*. W *Computers, Chess, and Cognition*. Rozdział 12. s. 197-216. Springer Verlag. 1990.
[27] Standage T. *The Turk. The Life and Times of the Famous Eighteenth-Century Chess-Playing Machine*. Walker Publishing Company. 2002.
[28] Shannon C. *Programming a Computer for Playing Chess*. Philosophical Magazine, Ser. 7, Vol. 41, No. 314. 1950.
[29] Spohrer J. C. *Learning plans through experience: A first pass in the chess domain*. W *Intelligent Robots and Computer Vision*, s. 518-527. 579 wolumen *Proceedings of the SPIE – The International Society for Optical Engineering*.
[30] Sutton R. S. *Learning to predict by the methods of temporal differences*. W *Machine Learning 3*, s. 9-44. Kluwer Academic Publishers. Boston. 1988.
[31] Sutton R. S., Barto A. G. *Reinforcement Learning: An Introduction. 11.1. TD-Gammon*. MIT Press. Cambridge, MA. A Bardford Book. 1998.
[32] Tadepalli P. *Lazy explanation-based learning: A solution to the intractable theory problem*. W *Proceedings of the 11th International Joint Conference on AI*, s. 694-700. Morgan Kaufmann. 1989.
[33] Tadeusiewicz R. *Sieci neuronowe*. Akademicka Oficyna Wydawnicza RM. Warszawa. 1993.
[34] Tesauro G. *Temporal difference learning of backgammon strategy*. W *Proceedings of the 9th International Conference on Machine Learning 8*, s. 451-457. 1992.
[35] Thrun S. *Learning to play the game of chess*. W *Advances in Neural Information Processing Systems 7*. 1995.
[36] Tunstall-Pedoe W. *Genetic algorithms optimizing evaluation functions*. W *ICCA Journal 14(3)*, s. 119-128.
[37] Turing A. *Computing Machinery and Intelligence*. Mind. 1950. 59, 433-460. http://loebner.net/Prizef/TuringArticle.html.
[38] *Von Kempelen`s Chess Turk Recreated*. 2004 http://www.chessbase.com/newsdetail.asp?newsid=1574.
[39] Wall B. *Computer Chess History*. 2006. http://www.geocities.com/SiliconValley/Lab/7378/comphis.htm.