AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ

KATEDRA INFORMATYKI STOSOWANEJ

Praca dyplomowa magisterska

Analiza i porównanie wybranych algorytmów dla gry karcianej
Analysis and comparison of selected algorithms for the card game

Autor: *Damian Malarczyk*
Kierunek studiów: *Informatyka*
Opiekun pracy: *dr inż. Edyta Kucharska*

Kraków, 2015

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Podziękowania

Spis treści

1. Wprowadzenie

1.1. Przedmiot pracy

Od kilku lat coraz większą popularnością cieszą się wszelkiego rodzaju gry planszowe i karciane. Przyciągają nie tylko coraz lepszą oprawą graficzną, lecz również ciekawą mechaniką pozwalającą na stosowanie różnych taktyk. Z tego powodu stanowią szerokie pole do testowania algorytmów optymalizujących dostępne ruchy tak, by zapewnić zwycięstwo.

Są również gry, w których kluczową rolę odgrywa tak zwana 'intuicja'. Obliczenie całego drzewa dostępnych ruchów jest zbyt skomplikowane i decyzja musi zostać podjęta na podstawie niepełnych danych. Przykładem takiej gry jest Łove Letter", gra niezbyt skomplikowana, jednak zawierająca dużo interakcji i możliwych ścieżek rozwoju sytuacji.

Inspirując się wyżej wymienioną grą, w poniższej pracy porównuję trzy algorytmy podejmowania decyzji w grze:

– probabilistyczny - który będzie podejmował decyzję w sposób losowy na podstawie prawdopodobieństwa wystąpień kart,

– zachłanny - który będzie wybierał zawsze najbardziej prawdopodobny scenariusz,

– Monte Carlo Tree Search - algorytm heurystyczny, który podejmowane decyzje opiera na symulacjach.

1.2. Cel pracy

1.3. Zawartość pracy

Rozdział 1 jest wprowadzeniem definiującym przedmiot pracy. W rozdziale 2 opisana została gra karciana wraz z problemem który przedstawia. Rozdział 3 przedstawia trzy proponowane algorytmy rozwiązujące problem. W rozdziale 4 zebrane są poszczególne etapy tworzenia aplikacji:

– analiza wymagań,

– koncepcja wykonania,

– wykorzystane technologie,

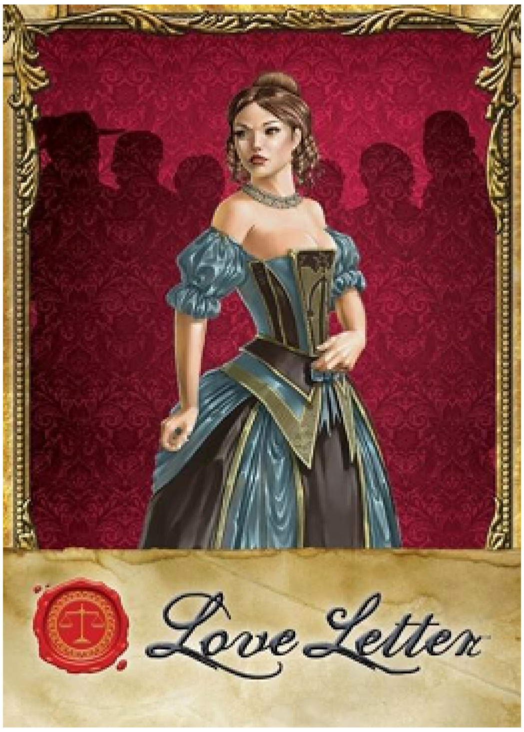
– diagramy,

– prezentacja systemu.

W rozdziale 5 zebrane są statystyki związane z implementacją algortytmów. Rozdział 6 stanowi podsumowanie pracy.

2. Gra karciana Love Letter

W tym rozdziale opisuję kontekst oraz zasady gry Love Letter. Tłumaczę działanie każdej karty oraz przedstawiam główny cel gry - wygranie określonej ilości rund. Następnie przedstawiam problem i analizuję jego złożoność. Wszystkie załączone zdjęcia oraz instrukcja zaczerpnięte są z [1] oraz [2].



Rysunek 2.1. Love Letter - okładka

2.1. Opis zasad gry

W trakcie gry wcielamy się w rolę jednego z adoratorów księżniczki starającego się o zdobycie jej serca. W tym celu przygotowaliśmy list miłosny, który chcemy jej dostarczyć. Niestety, księżniczka pogrążona jest obecnie w żałobie i nie przyjmuje do siebie nikogo obcego, w związku z czym musimy znaleźć inny sposób na przekazanie jej naszego listu. Oprócz księżniczki, na dworze znajdują się inne postacie, z których każda ma mniejszy lub większy dostęp do komnat naszej wybranki i może oddać jej list. Przekazujemy więc naszą przesyłkę swojemu tajnemu posłańcowi, a na koniec gry księżniczka jako pierwszy przeczyta ten list, który został przekazany przez najbardziej zaufaną postać. Serce wybranki zdobywa gracz, który jako pierwszy przekaże w ten sposób od 4 do 7 listów, w zależności od liczby graczy.

Cel i ustawienie początkowe

Love Letter rozgrywa się jako serię rund. Grę wygrywa gracz o następującej ilości wygranych rund:

- 7 w grze na 2 graczy,
- 5 w grze na 3 graczy,
- 4 w grze na 4 graczy.

Każda runda dzieli się na tury, w których naprzemiennie jeden z graczy wykonuje ruch. Grę wygrywa ten z nich, który na końcu ostatniej tury posiada kartę o wyższym numerze.

Ustawienie początkowe każdej rundy wygląda następująco:

- przetasuj karty
- odrzuć 1 wierzchnią kartę nie odkrywając jej (nie bierze udziału w rundzie),
- jeśli gra tylko 2 graczy, odrzuć 3 wierzchnie karty, odkryte,
- rozdaj po 1 karcie wszystkim graczom,
- jeśli jest to pierwsza runda, grę zaczyna gracz, który jako ostatni był na randce, w przeciwnym wypadku zwycięzca poprzedniej rundy.

Tura gracza i opis kart

Podczas swojej tury gracz dociąga jedna kartę ze stosu. Następnie wybiera jedną z dwóch kart, które posiada już w ręce, kładzie ją przed sobą tak, by była widoczna dla wszystkich i zastosowuje opisany na niej efekt - nawet jeśli jest negatywny. Zagrana karta pozostaje odkryta przez całą rundę, a druga pozostaje w ręce. Następnie tura przechodzi na osobę po lewej stronie aktywnego gracza.

W grze znajduje się 16 kart, w 8 typach. Każda z typów kart posiada wartość od 1 do 8. Są to kolejno: 4 karty Strażniczki, po 2 karty Kapłana, Barona, Pokojówki i Księcia, oraz po jednej karcie Króla, Hrabiny i Księżniczki. Ich szczegółowy opis wraz z wyglądem znajduje się poniżej:

Rysunek 2.2. Strażniczka

Na rysunku 2.2 przedstawiona jest karta typu Strażniczka. Zagrywając tę kartę należy wskazać jednego z pozostałych graczy i odgadnąć kartę którą posiada. Jeśli karta została prawidłowo odgadnięta, wskazany gracz odrzuca ją i przegrywa rundę.

Rysunek 2.3. Kapłan

Rysunek 2.3 przedstawia kartę typu Kapłan. Zagrywając tę kartę należy podglądnąć kartę wybranego gracza.

Rysunek 2.4. Baron

Na rysunku 2.4 przedstawiona jest karta typu Baron. Po zagraniu tej karty należy w ukryciu porównać drugą posiadaną kartą z wybranym graczem. Następnie ten gracz, który ma kartę o mniejszej wartości odrzuca swoją kartę i przegrywa rundę. W przypadku remisu nic się nie dzieje.

Rysunek 2.5. Pokojówka

Rysunek 2.5 przedstawia kartę typu Pokojówka. Zagranie tej karty sprawia, że gracz jest niewrażliwy na efekt pozostałych kart do czasu swojej następnej tury.

Rysunek 2.6. Książe

Na rysunku 2.6 przedstawiona jest karta typu Książę. Zagranie pozwala wybrać dowolnego gracza (w tym siebie), zmusić go do odrzucenia posiadanej karty i pociągnięcia następnej.

Rysunek 2.7. Król

Rysunek 2.7 przedstawia kartę typu Król. Po jej zagraniu należy wymienić się pozostałą kartą z innym graczem.

Rysunek 2.8. Hrabina

Rysunek 2.8 przedstawia kartę typu Hrabina. Ta karta ma działanie pasywne. Nie wywiera efektu po zagraniu, natomiast zmusza gracza do jej zagrania jeśli równocześnie posiada na ręce kartę typu Książę lub Król.

Rysunek 2.9. Księżniczka

Na rysunku 2.9 przedstawiona jest karta typu Księżniczka. Zagranie tej karty oznacza natychmiastową przegraną w rundzie. Ta zasada działa również, gdy gracz został zmuszony do zagrania tej karty, np. przez efekt karty Książe.

2.2. Definicja i analiza problemu

Z wyżej przedstawionych zasad wynika, że gra cechuje się wysokim stopniem losowości i jest niedeterministyczna, więc można ją przedstawić jako problem optymalizacyjny[4]. W związku z tym powstaje również pytanie, jaka jest najlepsza strategia mieszana[5] maksymalizująca prawdopodobieństwo wygrania gry. Dla dalszych rozważań zakładam, że w gra toczy się pomiędzy dwoma graczami.

Najprostszym sposobem na znalezienie takiej strategii, byłoby stworzenie drzew probabilistycznych dla wszystkich możliwych stanów początkowych gry a następnie opracowanie algorytmu podejmowania decyzji opartego na danych statystycznych. Jednakże ilość i rozmiar tych drzew może być zbyt duża by znaleźć rozwiązanie problemu w rozsądnym czasie. Z tego powodu postanowiłem najpierw oszacować ile jest wszystkich możliwych przebiegów gry. Nim przejdę do obliczeń, wprowadzę kilka definicji by ustandaryzować używane pojęcia:

– *Decyzja* - inaczej *Zagranie*, jest to typ karty wraz ze sposobem jej wykorzystania. Przykładowo: Zagranie karty typu Strażniczka z wyborem karty typu Król, lub zagranie karty typu Książę z wyborem na gracza przeciwnego.

– *Podjęcie decyzji* - to wybór zagrania, które zostanie użyte jako ruch w grze. Podjęcie decyzji to inaczej zwrócenie zagrania przez algorytm.

– *Strategia* - algorytm, który podejmuje decyzje.

– *Scenariusz* - inaczej przebieg gry, chronologiczny spis decyzji podjętych przez obu graczy od początku do końca rundy; Jedna ze ścieżek w drzewie probabilistycznym dla danego stanu początkowego rundy.

Analiza złożoności

Na przebieg każdej rundy wpływ mają następujące czynniki:

– Kolejność kart w talii na początku rundy

– Zagrywanie kart przez graczy

Zacząłem od oszacowania ilości możliwych stanów początkowych. W każdej rundzie bierze udział wszystkie 16 kart. Zakładając, że każda z nich jest unikalna, to liczba wszystkich możliwych kolejności kart to permutacja, którą obliczam wzorem podanym w [3]:

$$P_n = n!\text{ , gdzie } n \in N^+$$

Dla n = 16, $n! = 20922789888000$. Część kart się powtarza, więc tę liczbę należy jeszcze podzielić przez permutacje powtarzających się kart Strażniczki, Kapłana, Barona, Pokojówki oraz Księcia. Razem jest to $4! * 2! * 2! * 2! * 2! = 384$. Ostatecznie wynika, że liczba unikalnych kolejności kart wynosi:

$$20922789888000/32 = 54486432000$$ - 54mld, 864mln i 432 tys.

Nie jest to jednak liczba wszystkich dostępnych scenariuszy. W każdej turze gracz ma do wyboru co najmniej dwa zagrania, ponieważ tyle ma dostępnych kart. Jednakże, w przypadku karty Strażniczki możliwości jest więcej, ponieważ można wytypować 7 typów kart, a w jednym przypadku może to pokonać przeciwnika i skończyć rundę. W związku z tym, by oszacować liczbę scenariuszy na zadanej kolejności kart, posłużyłem się następującą metodą:

– zgodnie z zasadami gry dla dwóch graczy, odrzucam łącznie 4 pierwsze karty (1 zakryta, 3 odkryte).

– rozdaję po 1 karcie obu graczom. Pozostaje 10 kart w talii.

– zakładając, że żadna decyzja nie spowoduje przerwania rundy, gracze łącznie 10 razy pociągną kartę, więc podejmą 10 decyzji.

– każdą decyzję można przedstawić jako 0 (zagranie posiadanej karty) lub 1 (zagranie pociągniętej karty) - w tym miejscu jeśli istnieje możliwość zagrania karty w wieloraki sposób, upraszczam to do jednego zagrania niekończącego rundę.

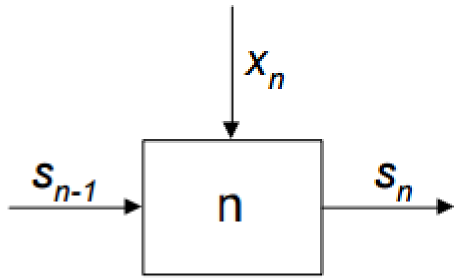
Na podstawie powyższego można oszacować, że możliwych scenariuszy dla danej kolejności kart jest $2^{10} = 1024$. Łącząc tę liczbę z ilością możliwych kolejności kart, utrzymujemy przybliżoną liczbę scenariuszy:

$$54486432000 * 1024 = 55794106368000 \approx 5.8 * 10^{12}$$

Z uwagi na rząd wielkości, stworzenie strategii na podstawie analizy statystycznej wszystkich dostępnych scenariuszy jest problemem *NP-trudnym*. Z tego powodu, zamiast odpowiadać na pytanie „Jaka jest najlepsza strategia podejmowania decyzji?”, dużo łatwiej będzie odpowiedzieć na pytanie „która z podanych strategii jest najlepsza?”, gdyż jest to charakterystyczna cecha problemów klasy *NP*. Kierując się tą zasadą, w następnym rozdziale opisałem wybrane strategie, których skuteczność sprawdzę implementując je w napisanej przeze mnie aplikacji. Pozostaje jeszcze sformalizować przedstawiony problem za pomocą modelu matematycznego.

2.3. Model matematyczny problemu

Patrząc z perspektywy jednego z graczy, cała gra składa się z szeregu etapów (tur), gdzie w każdym etapie gracz podejmuje decyzję, a stan początkowy następnego etapu jest wynikiem podjętej decyzji oraz reakcji gracza przeciwnego. Takie zachowanie można przedstawić jako wieloetapowy proces podejmowania decyzji w warunkach niepewności[6], który ogólnie zdefiniowałem następująco:



Rysunek 2.10. Schemat modelu

- n - etap gry
- $s_{n-1} \in S_{n-1}$ - stan wejściowy
- $x_i \in X_n$ - podjęta decyzja
- $s_n \in S_n$ - stan wyjściowy
- X_n - zbiór dopuszczalnych decyzji dla etapu n-tego
- S_n - zbiór dopuszczalnych stanów dla etapu n-tego
- $T_n : \{x_i, s_{n-1}\} \rightarrow \{s_n, X_n\}$ - funkcja przejścia
- $Q = max(P_w)$ - funkcja celu, gdzie P_w oznacza prawdopodobieństwo wygranej
- $D : \{X_n, s_{n-1}\} \rightarrow x_n$ - strategia podejmowania decyzji

Funkcja przejścia T_n wynika bezpośrednio z opisanych wcześniej zasad oraz zawiera reakcję gracza przeciwnego, w efekcie zwracając jeden z możliwych stanów należących do S_n oraz zbiór dopuszczalnych decyzji X_n. Również wszystkie $S_0..S_n$ oraz $X_0..X_n$ wynikają z zasad gry. Jedynym nieokreślonym elementem pozostaje D, a moje propozycje na jego zdefiniowanie przedstawiam w następnym rozdziale.

3. Przegląd wybranych algorytmów i innych rozwiązań

W tym rozdziale przedstawiam kilka wybranych algorytmów, które zaimplementuję w następnym rozdziale. Dla każdego z nich opisuję jego użycie w odniesieniu do gry 'Love Letter'.

3.1. Algorytm losowy

3.1.1. Opis

Jest to najprostszy algorytm podejmowania decyzji. Na wejściu otrzymuje listę dostępnych zagrań, z których w całkowicie losowy sposób wybiera jedno. Każde z dostępnych zagrań ma takie samo prawdopodobieństwo bycia wybranym przez ten algorytm.

3.1.2. Sposób wykorzystania

Z uwagi na prostotę tego algorytmu, używam go jako punktu odniesienia dla wszystkich pozostałych strategii. Dla każdej z nich algorytm losowy spełnia rolę drugiego gracza i w ten sposób można łatwo porównać je ze sobą. Dla tego algorytmu zastosowałem jedną drobną modyfikację: nigdy nie podejmie decyzji o zagraniu Księżniczki - oznacza to natychmiastową przegraną niezależnie od momentu gry, co mogłoby mocno wypaczyć wyniki innych algorytmów.

3.1.3. Zapis pseudokodem

function $D_{losowa}(X_n, s_{n-1})$
    for all x_i **do**
        if $x_i ==$ Księżniczka **then**
            $X_n \leftarrow X_n - x_i$
        end if
    end for
    return wylosujJeden(X_n)
end function

3.2. Algorytm zachłanny

3.2.1. Opis

Algorytm zachłanny polega na wyborze najlepszego możliwego zagrania dostępnego w danej chwili, nie analizując jego konsekwencji w przyszłości. Pomimo, że takie podejmowanie decyzji krótkowzroczne, jest łatwe w implementacji i daje atrakcyjne wyniki w niektórych problemach, np. przy szukaniu minimalnego drzewa rozpinającego[7].

3.2.2. Sposób wykorzystania

W kontekście gry 'Love Letter', implementacja algorytmu zachłannego wymaga pewnego doprecyzowania. Najważniejszą częścią jest funkcja kryterialna oceniająca wartość zagrania, która w niektórych przypadkach musi być oparta o probabilistykę wystąpienia kart u przeciwnika.

Rozważmy przypadek, w którym algorytm musi podjąć decyzję o zagraniu karty Strażniczki, lub karty Barona. Jest to pierwszy ruch gracza, a w widocznych kartach odrzuconych na starcie są odrzucone karty Króla, Księcia i Pokojówki. Oznacza to, że w talii pozostało 9 kart, a jedna z odrzuconych jest niewidoczna, niemniej jednak ją też trzeba brać pod uwagę. Wyliczenie prawdopodobieństwa *P* jaką kartę ma przeciwnik jest tym momencie proste, jednak trzeba jeszcze wziąć pod uwagę drobny szczegół - czy do liczenia *P* wliczać karty posiadane w ręce. Z jednej strony wydaje się to nielogiczne i może prowadzić do wybierania nieoptymalnych decyzji (co przeczyłoby idei algorytmu zachłannego), z drugiej strony można to potraktować jako element blefu, który jest nieodłączną częścią każdej gry towarzyskiej. W swojej implementacji założyłem absolutną zachłanność algorytmu i karty posiadane na ręce są pomijane w obliczeniach. Wobec powyższego, prawdopodobieństwo wystąpienia kart u przeciwnika rozkłada się następująco:

Tablica 3.1. Przykład rozkładu prawdopodobieństwa

Karta	$P(Karta)$
Strażniczka	30%
Kapłan	20%
Baron	10%
Pokojówka	10%
Książę	10%
Hrabina	10%
Księżniczka	10%

Zagranie karty Baron oznacza porównanie drugiej karty z kartą przeciwnika. Łatwo policzyć, że w 70% przypadków skończyłoby to się porażką, a w 30% nie było by żadnego efektu. Drugim dostępnym ruchem jest zagranie karty Strażniczki, a w jej przypadku najlepszym wyborem jest wytypowanie

Kapłana, co daje 20% szans na zwycięstwo i 80% szans, że nie nastąpi żaden efekt. Zauważmy, że gdybyśmy wliczali posiadane karty do obliczenia prawdopodobieństwa, wystąpienie Barona i Kapłana byłoby tak samo możliwe. W takich przypadkach algorytm powinien zawsze celować w kartę z wyższym numerem. By formalnie stwierdzić, jaka decyzja powinna zostać podjęta, musimy obliczyć funkcję kryterialną dla dostępnych zagrań i wybrać to zagranie, dla której funkcja przyjmuje wyższy wynik. Przyjmijmy, że funkcja kryterialna wygląda następująco:

$$F(Karta) = 1 + prawdopodobienstwo\_wygranej - prawdopodobienstwo\_przegranej$$

Po podstawieniu otrzymujemy:

$$F(Strazniczka) = 1.2 \text{ i } F(Baron) = 0.3$$
$$F(Strazniczka) > F(Baron) => Decyzja = Zagranie(Pokojowka + Kaplan)$$

Najlepszą decyzją w tym wypadku jest zagranie karty Strażniczki z wyborem karty Kapłana. Jak jednak na podstawie powyższego wzoru ocenić zagranie karty Kapłana, Pokojówki lub Króla? Każda z nich wymaga unikalnej funkcji kryterialnej. Mając na uwadze, że w kontekście strategii zachłannej decyzja zawsze powinna być optymalna w ujęciu chwili, ustaliłem następujące warunki oceny zagrania karty:

– Strażniczka - ocena zagrania wzrasta gdy pozwala wyeliminować przeciwnika.
– Kapłan - w ujęciu chwili jest to karta neutralna i ocena jej zagrania będzie zawsze stała.
– Baron - ocena zagrania wzrasta gdy mamy drugą kartę silniejszą niż może mieć przeciwnik.
– Pokojówka - podobnie jak w przypadku kapłana, ocena zagrania karty będzie stała.
– Książę - ocena zagrania na przeciwnika wzrasta z prawdopodobieństwem Księżniczki u przeciwnika. Ocena zagrania na siebie jest stała, lecz 0 gdy druga karta to Księżniczka.
– Król - ocena stała
– Hrabina - ocena stała, jednak musimy ją zagrać gdy druga posiadana karta to Król lub Książe
– Księżniczka - nie może być nigdy wyrzucona.
– Dodatkowo, jeśli oba zagrania mają taką samą wartość, powinna być podjęta decyzja o zagraniu karty o niższej wartości.

3.2.3. Zapis pseudokodem

function $D_{zachlanna}(X_n, s_{n-1})$
  for all $rodzajKarty \in rodzajeKart$ **do** ▷ Tworzenie tablicy prawdopodobieństwa
    $P[rodzajKarty] \leftarrow$ prawdopodobieństwo wystąpienia karty danego rodzaju u przeciwnika

```
    end for
    decyzja ← NULL                          ▷ Szukanie zagrania o najwyższej wycenie
    for all x_i ∈ X_n do        ▷ Obliczenie funkcji kryterialnej dla każdego zagrania
        if F(decyzja, P[] < F(x_i, P[] then
            decyzja ← x_i
        end if
    end for
    return decyzja
end function
```

Gdzie funkcja kryterialna wygląda następująco:

```
function F(x_i, P[])
    switch x_i do
        case Strażniczka + Kaplan           ▷ Zagranie Strażniczki na dany typ karty
            return 1 + P[Kaplan] + 0.08
        case Strażniczka + Baron
            return 1 + P[Baron] + 0.08
        ...
        case Strażniczka + Księżniczka
            return 1 + P[Ksiezniczka] + 0.08
        case Kapłan
            return 1 + 0.07
        case Baron      ▷ Kryterium zależy od wartości W() drugiej posiadanej karty (DK)
            szansePrzegranej ← 0
            szanseWygranej ← 0
            for all typKarty do
                if W(DK) < W(typKarty) then
                    szansePrzegranej ← szansePrzegranej + P[typKarty]
                else if W(DK) > W(typKarty) then
                    szanseWygranej ← szanseWygranej + P[typKarty]
                end if
            end for
            return 1 − szansePrzegranej + szanseWygranej + 0.06
        case Pokojówka
            return 1 + 0.05
        case Książe na przeciwnika
            return 1 + P[Ksiezniczka] + 0.04
        case Książe na siebie
```

```
            if DK == Ksiezniczka then
                return 0
            else
                return 1 + 0.04
            end if
        case Król
            return 1 + P[Ksiezniczka] + P[Hrabina] + 0.03
        case Hrabina, gdy druga posiadana karta to Król lub Książe
            return 10 + 0.02
        case Hrabina
            return 1 + 0.02
        case Księżniczka
            return 0
end function
```

3.3. Algorytm min-max

3.3.1. Opis

Algorytm minimaksowy polega na „minimalizowaniu maksymalnych możliwych strat" bądź alternatywnie na „maksymalizacji minimalnego zysku"[8]. Zgodnie z [10] często stosuje się go do gier o następujących zasadach:

- występuje dwóch graczy
- ruchy wykonywane są naprzemiennie
- w każdym stanie istnieje skończona liczba decyzji do podjęcia
- stan i podjęta decyzja jednoznacznie wyznaczają stan następny
- każdy stan może zakwalifikować do jednej z następujących kategorii:
  - wygrana pierwszego gracza
  - wygrana drugiego gracza
  - remis
  - sytuacja nierozstrzygnięta

Najczęstsze implementacje polegają na przeszukiwaniu drzewa dalszych przebiegów gry począwszy od zadanego stanu początkowego, tak jak na przykład w warcabach[9]. Istnieją także implementacje opierające się na liczbowej ocenie ruchu[10] i na tej idei opieram swoją implementacje algorytmu minimaksowego.

3.3.2. Sposób wykorzystania

W podanym powyżej opisie, gra Love Letter niezgodna jest w punkcie czwartym. Ze względu na niepewność jaką kartę posiada przeciwnik, jaką pociągnie ze stosu w swojej turze, oraz jaki wykona ruch, podjęcie decyzji x_i w stanie s_{n-1} nie określa jednoznacznie stanu s_n. Mimo to, jesteśmy w stanie statystycznie ocenić możliwości przeciwnika, a tym samym zminimalizować naszą maksymalną stratę. Ponieważ zysk należy rozumieć jako zwiększenie szansy na wygraną (tak jak w strategii zachłannej), to strata oznacza zwiększenie szansy na przegraną.

Rozważmy scenariusz, w którym gracz pierwszy posiada kartę Księcia oraz Pokojówki. Na stosie pozostało 2 karty, 1 karta jest zakryta zgodnie z zasadami rozpoczęcia rundy i przeciwnik również posiada 1 kartę. Wśród tych 4 nieznanych graczowi kart są karty Strażniczki, Barona, Księcia oraz Księżniczki. Prawdopodobieństwo wystąpienia każdej z nich u przeciwnika wynosi 25% i razem stanowią one tablicę prawdopodobieństwa $P[]$. Przyjmijmy, że wykorzystywana w poprzedniej strategii funkcja kryterialna $F(Karta, P[])$ to maksymalizacja zysku i nazwijmy ją $F_{max}(Karta, P[])$ Zgodnie z jej definicją $F_{max}(Ksiaze, P[]) = 1.29$ i $F_{max}(Pokojowka, P[]) = 1.05$, więc zagranie Księcia maksymalizuje szansę na wygraną. Jeśli jednak weźmiemy pod uwagę, że w 75% przypadków nie wygrywamy, wówczas musimy rozważyć odpowiedź przeciwnika. W tym wypadku jest 6 par kart które może posiadać przeciwnik, więc dla każdej z nich, zgodnie ze strategią zachłanną, należy obliczyć funkcję kryterialną oraz szansę na przegraną. Dodatkowo należy zauważyć, że z perspektywy przeciwnika w każdym przypadku gracz pierwszy ma inny rozkład prawdopodobieństwa wystąpienia karty. Pełne obliczenia znajdują się w tabeli poniżej.

Skróty : *Str* - Strażniczka; *Bar* - Baron; *Pok*-Pokojówka; *K-e* - Książę; *K-a* - Księżniczka
Gracz pierwszy po zagraniu Księcia posiada kartę Pokojówki.

Tablica 3.2. Scenariusze reakcji przeciwnika na decyzję w stanie s_{n-1}

Karty 1 i 2	$P(Karta)$ pierwszego gracza	$F(Karta_1)$	$F(Karta_2)$	Szanse przegranej
Str ; Bar	$P(Pok) = P(K\text{-}e) = P(K\text{-}a) = 33.(3)\%$	1.41	0.6	33.(3)%
Str ; K-e	$P(Pok) = P(Bar) = P(K\text{-}a) = 33.(3)\%$	1.41	1.37	33.(3)%
Str ; K-a	$P(Str) = P(Bar) = P(K\text{-}e) = 33.(3)\%$	1.41	0	33.(3)%
Bar ; K-e	$P(Str) = P(Pok) = P(K\text{-}a) = 33.(3)\%$	1.39	1.37	100%
Bar ; K-a	$P(Str) = P(Pok) = P(K\text{-}e) = 33.(3)\%$	2.06	0	100%
K-e ; K-a	$P(Str) = P(Pok) = P(Bar) = 33.(3)\%$	1.04	0	0%

Każdy rozważany scenariusz ma taką samą szansę wystąpienia, czyli $\frac{1}{6}$. Statystycznie szansa na przegraną wynosi więc:

$$\frac{1}{6} * (\frac{1}{3} + \frac{1}{3} + \frac{1}{3} + 1 + 1 + 0) = \frac{1}{6} * 3 = 0.50$$

Warunkiem wystąpienia możliwości przegrania, jest brak wygranej po zagraniu Księcia, czyli ostatecznie szanse na przegraną wynoszą:

$$F_{min}(Ksiaze, P[]) = 0.75 * 0.50 = 0.375$$

Modyfikując o tę wartość ocenę zagrania Księcia otrzymujemy finalnie:

$$K(Ksiaze) = F_{max}(Ksiaze, P[]) - F_{min}(Ksiaze, s_{n-1}) = 1.29 - 0.375 = 0.915$$

W przypadku zagrania Pokojówki szanse przegrania wynoszą 0%, ponieważ zgodnie z zasadami gry jesteśmy odporni na działanie kart przeciwnika:

$$K(Pokojowka) = F_{max}(Pokojowka, P[]) - F_{min}(Pokojowka, s_{n-1}) = 1.05 - 0 = 1.05$$

Wynika z tego, że decyzją która minimalizuje maksymalną stratę jest $Zagranie(Pokojowka)$.

3.3.3. Zapis pseudokodem

function $D_{minimaksowa}(X_n, s_{n-1})$
    for all $rodzajKarty \in rodzajeKart$ **do** ▷ Tworzenie tablicy prawdopodobieństwa
        $P[rodzajKarty] \leftarrow$ prawdopodobieństwo wystąpienia karty danego rodzaju u przeciwnika
    end for
    for all $x_i \in X_n$ **do** ▷ Obliczenie funkcji kryterialnej dla każdego zagrania
        $K[x_i] \leftarrow F_{max}(x_i, P[]) - F_{min}(x_i, s_{n-1})$
    end for
    $decyzja \leftarrow x_0$ ▷ Szukanie zagrania o najwyższej wycenie
    for all x_i **do**
        if $K[decyzja] < K[x_i]$ **then**
            $decyzja \leftarrow x_i$
        end if
    end for
    return $decyzja$
end function

Funkcja $F_{max}(Karta)$ jest identyczna do $F(Karta)$ występującej w strategii zachłannej, w związku z czym zapiszę wyłącznie $F_{min}(Karta)$.

function $F_{min}(x_i, s_{n-1})$
    $L[Y_n] \leftarrow$ lista możliwych zbiorów decyzji przeciwnika
    switch x_i **do**
        case Strażniczka + Kaplan ▷ Szanse, że Strażniczka nie przyniesie zwycięstwa
            return $(1 - P[Kaplan]) * R(s_{n-1}, L[Y_n])$
        case Strażniczka + Baron
            return $(1 - P[Baron]) * R(s_{n-1}, L[Y_n])$
        ...
        case Strażniczka + Księżniczka

return $(1 - P[Ksiezniczka]) * R(s_{n-1}, L[Y_n])$
case Kapłan
return $R(s_{n-1}, L[Y_n])$
case Baron ▷ Szanse, że porównanie zakończy się remisem
$szanseRemisu \leftarrow P[DK]$
return $szanseRemisu * R(s_{n-1}, L[Y_n])$
case Pokojówka
return 0
case Książe na przeciwnika
return $(1 - P[Ksiezniczka]) * R(s_{n-1}, L[Y_n])$
case Książe na siebie
if $DK == Ksiezniczka$ **then**
return 0
else
return $R(s_{n-1}, L[Y_n])$
end if
case Król
return $R(s_{n-1}, L[Y_n])$
case Hrabina, gdy druga posiadana karta to Król lub Książe
return $R(s_{n-1}, L[Y_n])$
case Hrabina
return $R(s_{n-1}, L[Y_n])$
case Księżniczka
return 0
end function

Funkcja $R(s_{n-1}, L[Y_n])$ oznacza szansę porażki po reakcji przeciwnika.

function $R(s_{n-1}, L[Y_n])$
for all $Y_n \in L[X_n]$ **do**
$Z[] \leftarrow D_{zachlanna}(Y_n, s_{n-1})$ ▷ Lista potencjalnych reakcji
end for
for all $x_i \in Z[]$ **do** ▷ Sprawdzenie szansy porażki
switch x_i **do**
case Strazniczka + DK ▷ DK - druga karta pierwszego gracza
$P_{porazki}[x_i] \leftarrow (Z[x_i] - 1.08)$
case Baron
if $W(DK) < W(DK_{przeciwnika})$ **then**

$P_{porazki}[x_i] \leftarrow 1$
else
$P_{porazki}[x_i] \leftarrow 0$
end if
case Książę
if $DK == Ksiezniczka$ **then**
$P_{porazki}[x_i] \leftarrow 1$
else
$P_{porazki}[x_i] \leftarrow 0$
end if
end for
return $\sum_{i=0}^{Z[].size} \frac{1}{Z[].size} * P_{porazki}[i]$
end function

3.4. Algorytm Monte Carlo Tree Search

3.4.1. Opis

W wielkim skrócie opis jest w [11]. Bardziej szczegółowo jest to opisane w [12]. Przykładem zastosowania jest Hex [15] czy Rome Total War [13]. Co więcej, pokazano że da się zastosować ten algorytm w grach niedeterministycznych jak Osadnicy z Catanu [14].

3.4.2. Sposób wykorzystania

Brutalne symulacje z minimaksowym przeciwnikiem. Modyfikacja - nie zagrywa Księżniczki.

3.4.3. Zapis pseudokodem

Zapis pseudoliryczny, bardzo ogólnikowy. Nie ma co się rozwalać na 4 strony.

4. Implementacja

4.1. Analiza wymagań

Implementacja zasad i modelu przedstawionego w w rozdziale 2. Use case'y - wybór dwóch S.I. które będą przeciwko sobie grać, ustalanie ilości partii, parametryzacja MCTS (przeciwnik, czas na ruch). Program ma zwracać statystyki zwycięstw.

4.2. Koncepcja wykonania

Java w konsoli. Albo Python. Się zobaczy.

4.3. Wykorzystane technologie

j. w.

4.4. Diagramy

Wrzucić use case'y, UML klas,

4.5. Prezentacja systemu

Parę screenów z wynikami.

4.6. Problemy napotkane w trakcie realizacji

Czas, czas, czas.

5. Rezultaty

5.1. Czas działania na 1000 partii

5.2. Statystyki zwycięstw

5.3. Zbieżność algorytmów podejmowania decyzji

5.4. Wnioski

6. Podsumowanie

Bibliografia

[1] Alderac Entertainment Group, *Love Letter*, 2012.

[2] Alderac Entertainment Group, *Love Letter*, http://www.alderac.com/tempest/love-letter, 2016-02-08.

[3] Alicja Cewe, Halina Nahorska, Irena Pancer, *Tablice matematyczne*, Wydawnictwo Podkowa Gdańsk 2002, rozdział *Kombinatoryka*.

[4] *Wikipedia*, hasło *Problem optymalizacyjny*. https://pl.wikipedia.org/wiki/Problem_optymalizacyjny, 2016-02-21.

[5] *Wikipedia*, hasło *Strategia w teorii gier*. https://pl.wikipedia.org/wiki/Strategia_mieszana, 2016-04-10.

[6] *Wikipedia*, hasło *Teoria decyzji*. https://pl.wikipedia.org/wiki/Teoria_decyzji, 2016-04-10.

[7] Sanjoy Dasgupta, Christos Papadimitriou, Umesh Vazirani, *Algorytmy*, Wydawnictwo Naukowe PWN, Warszawa 2012, rozdział *Algorytmy Zachłanne*, s. 133.

[8] *Wikipedia*, hasło *Algorytm min-max*. https://pl.wikipedia.org/wiki/Algorytm_min-max, 2016-04-24.

[9] *Funkcja oceniająca do algorytmu minimaksu w grze warcaby*, http://sequoia.ict.pwr.wroc.pl/ witold/aiarr/2009_projekty/warcaby/, 2016-05-04.

[10] *Studia Informatyczne*, Sztuczna inteligencja/SI Moduł 8 - Gry dwuosobowe. http://wazniak.mimuw.edu.pl/index.php?title=Sztuczna_inteligencjaSI_Moduł_8_-_Gry_dwuosobowe, 2016-04-24.

[11] *Introduction to Monte Carlo Tree Search*, https://jeffbradberry.com/posts/2015/09/intro-to-monte-carlo-tree-search/, 2016-05-04.

[12] Guillaume Chaslot, Sander Bakkes, Istvan Szita, Pieter Spronck, *Monte-Carlo Tree Search: A New Framework for Game AI*, Universiteit Maastricht / MICC. P.O. Box 616, NL-6200 MD Maastricht, The Netherlands

[13] *Monte-Carlo Tree Search in TOTAL WAR: ROME II's Campaign AI*, http://aigamedev.com/open/coverage/mcts-rome-ii/, 2016-05-04.

[14] Istvan Szita, Guillaume Chaslot, Pieter Spronck, *Monte-Carlo Tree Search in Settlers of Catan* Volume 6048 of the series Lecture Notes in Computer Science, s. 21-32, 2010.

[15] Broderick Arneson, Ryan Hayward, Philip Henderson, *MoHex Wins Hex Tournament*, ICGA Journal Vol. 32 No. 2 s. 114–116, Czerwiec 2009