III Konferencja Młodych Informatyków

Uniwersytet Śląski

Wydział Informatyki i Nauki o Materiałach

Sosnowiec 2003

Programowanie gier planszowych

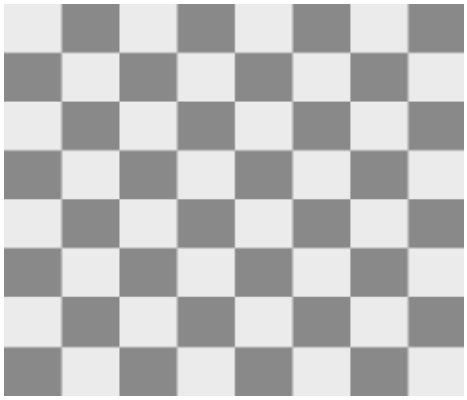
Tomasz Rostański

Streszczenie

W niniejszej pracy zostanie jedynie zasugerowane pewne podejście do programowania strategicznych gier planszowych. Niniejszy referat ma na celu przybliżenie czytelnikowi sposobów na bardzo proste programowanie gier planszowych. Dlatego brak będzie jakichkolwiek skomplikowanych algorytmów, co jednakże nie oznacza, że zaproponowane rozwiązania są złe czy niewydajne. Jak można się niejednokrotnie przekonać proste rozwiązania są niejednokrotnie bardzo skuteczne. W referacie zostaną przedstawione przykłady dla gier: Warcaby i Hexxagon. A następnie zostaną przedstawione otrzymane wyniki dla gry Hexxagon.

PLANSZA

Rozpatrzmy planszę o wymiarach 8x8 pól:



Naturalnym sposobem przedstawienia jest tablica 2-wymiarowa:

j \ ***i***	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***
0								
1								
2								
3								
4								
5								
6								
7					X			

W ten sposób możemy opisać planszę o dowolnej liczbie pól. Położenie obiektu X możemy zatem opisać za pomocą 2 współrzędnych (i,j): ***położenie(X)=(4,7);***

Planszę taką można również przedstawić w postaci tablicy jednowymiarowej, wykorzystując numerowanie pól:

0	***10***	***20***	***30***	***40***	***50***	***60***	***70***
1	***11***	***21***	***31***	***41***	***51***	***61***	***71***
2	***12***	***22***	***32***	***42***	***52***	***62***	***72***
3	***13***	***23***	***33***	***43***	***53***	***63***	***73***
4	***14***	***24***	***34***	***44***	***54***	***64***	***74***
5	***15***	***25***	***35***	***45***	***55***	***65***	***75***
6	***16***	***26***	***36***	***46***	***56***	***66***	***76***
7	***17***	***27***	***37***	***47***	***57***	***67***	***77***

To rozwiązanie ma jednak dwie wady:

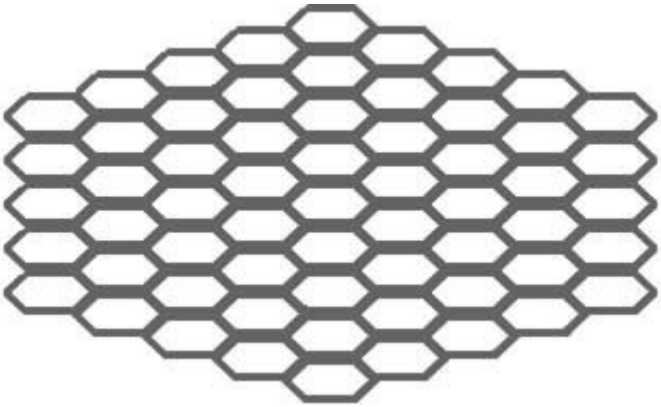
- wprowadza większą zajętość pamięci poprzez pamiętanie pól nie należących do planszy;
- ogranicza rozmiar planszy do 10x10 przy indeksowaniu systemem dziesiętnym i 16x16 przy heksadecymalnym (co jednakże pozwala opisać ogromną większość gier planszowych)

Położenie obiektu na planszy będziemy opisywali jedną tylko liczbą – numerem pola:

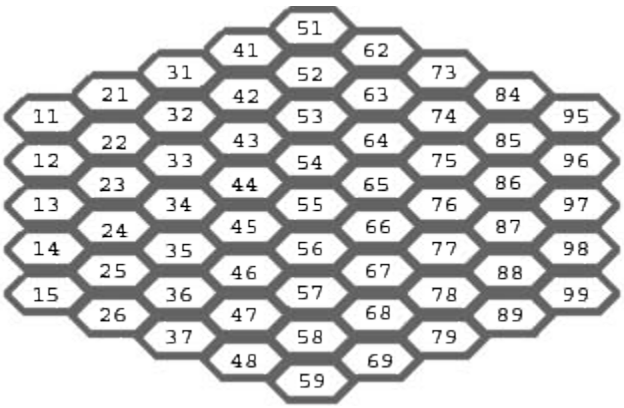
położenie(X) = 47;

Wykorzystanie tablicy jednowymiarowej indeksowanej numeracją pól jest wygodniejsze od wariantu dwuwymiarowego, toteż będziemy rozważać tylko taki przypadek.

Rozpatrzmy planszę do gry Hexxagon o dość nietypowym kształcie:



Również w tym przypadku dokonujemy numeracji pól, ale w nieco inny sposób niż poprzednio (dlaczego tak – o tym później)



WŁAŚCIWOŚCI PIONKA

Niezależnie od rodzaju czy też zasad gry, pionek powinien posiadać pewną wiedzę. Powinien znać:

- przynależność do gracza (odpowiednik koloru pionka);
- położenie na planszy;
- typ (jeśli w grze występuje kilka typów pionków).

Powinien również posiadać następujące umiejętności:

- wykonanie ruchu prawidłowego ruchu;
- oszacowanie korzyści i strat wynikających z możliwego ruchu;
- możliwość zbicia pionka przeciwnika czy też jego przejęcia.

W zależności od gry pionek może posiadać jeszcze inne dodatkowe umiejętności.

Wygodnie jest stworzenie klasy *Pionek* posiadającej ww. właściwości i metody.
Dodatkowo wygodnie posłużyć się tablicą obiektów klasy *Pionek*, która będzie oznaczała wszystkie pionki jednego gracza (np. w warcabach) albo obydwu graczy (takie rozwiązanie jest wygodniejsze w Hexxagon ze względu na częste zmiany właściciela pionka).

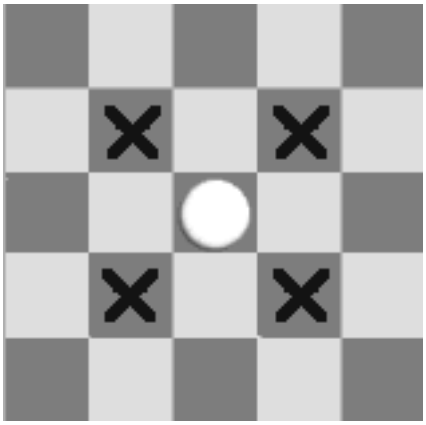
KRYTERIUM STOPU

Przed przystąpieniem do programowania gry należy zdać sobie sprawę z momentu jej zakończenia. Należy przeanalizować wszystkie możliwe sytuacje. Błędne bądź niekompletne sformułowanie kryterium stopu będzie prowadziło do zapętlenia programu.
Gra warcaby kończy się w momencie zbicia wszystkich pionków przeciwnika ale również w przypadku zablokowania przeciwnika. Należy tą drugą ewentualność także rozpatrzyć.
Gra Hexxagon kończy się natomiast w momencie zapełnienia całej planszy, w momencie zablokowania ruchu przeciwnika oraz w przypadku utraty wszystkich pionków przez jednego z graczy.

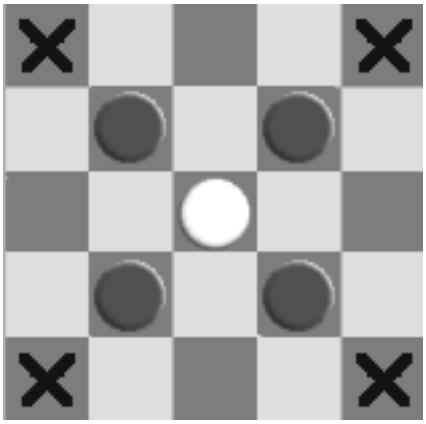
RUCH

WARCABY

Pionek w warcabach ma możliwość wykonania ruchu na następujące pola:

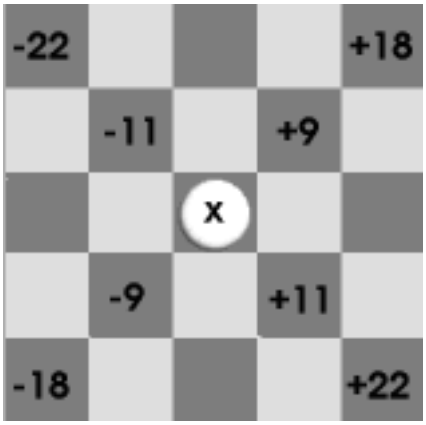


W przypadku sąsiadowania z pionkiem przeciwnika, rozpatrywany pionek ma możliwość ew. zbicia pionka przeciwnika:



Następnie rekurencyjnie sprawdzamy czy z nowej lokalizacji nie będzie możliwości kolejnego bicia (bicie wielokrotne).

Pola wokół naszego pionka mają następujące numery względne (liczone od numeru pionka - x):



Algorytm wykonujący ruch musi zatem sprawdzić wpierw czy można wykonać bicie i ew. czy bić jest więcej. W przypadku nie możności zbicia rozważy wykonanie ruchu:

jeżeli na polu (x-11) znajduje się wrogi pionek i pole (x-22) jest puste odnotuj możliwość bicia
jeżeli na polu (x-9) znajduje się wrogi pionek i pole (x-18) jest puste odnotuj możliwość bicia
jeżeli na polu (x+11) znajduje się wrogi pionek i pole (x+22) jest puste odnotuj możliwość bicia
jeżeli na polu (x+9) znajduje się wrogi pionek i pole (x+18) jest puste odnotuj możliwość bicia
jeżeli liczba bić = 1 wykonaj bicie i zakończ algorytm
w przeciwnym wypadku gdy liczba bić > 1 wybierz a następnie wykonaj jedno najlepsze bicie i zakończ algorytm
w przeciwnym wypadku
    jeśli pole (x-11) jest puste odnotuj możliwość ruchu
    jeśli pole (x+9) jest puste odnotuj możliwość ruchu
    jeśli możliwe do wykonania są 2 ruchy wybierz i wykonaj najlepszy
    w przeciwnym wypadku jeśli możliwy jest jeden to go wykonaj
    w przeciwnym wypadku powiadom ze nie możesz wykonać ruchu

Algorytm ten dla podanego pionka wykona najlepszy lokalnie ruch.
Algorytm dla damki jest analogiczny, ale związany z analizą większej ilości pól i kierunków ruchu.

Algorytm wybierający najlepszy ruch ze wszystkich , będzie kierował się maksymalnym zyskiem wynikającym z pojedynczego ruchu. Zysk będzie różnicą pomiędzy zbitą liczbą pionków przeciwnika a swoją.

dla każdego pionka

jeżeli na polu (x-11) znajduje się wrogi pionek i pole (x-22) jest puste odnotuj możliwość bicia

jeżeli na polu (x-9) znajduje się wrogi pionek i pole (x-18) jest puste odnotuj możliwość bicia

jeżeli na polu (x+11) znajduje się wrogi pionek i pole (x+22) jest puste odnotuj możliwość bicia

jeżeli na polu (x+9) znajduje się wrogi pionek i pole (x+18) jest puste odnotuj możliwość bicia

jeżeli liczba bić = 1 zaznacz bicie (pionek + numer pola + liczba zbitych pionków) i zakończ analizę pionka

w przeciwnym wypadku gdy liczba bić > 1 wybierz a następnie zaznacz jedno najlepsze bicie i zakończ analizę pionka

w przeciwnym wypadku

jeśli pole (x-11) jest puste odnotuj możliwość ruchu

jeśli pole (x+9) jest puste odnotuj możliwość ruchu

jeśli możliwe do wykonania są 2 ruchy wybierz i zaznacz najlepszy (pionek + numer pola + liczba zbitych pionków=0)

w przeciwnym wypadku jeśli możliwy jest jeden to zaznacz go

w przeciwnym zaznacz brak ruchu (pionek + numer pola = pole zza tablicy + liczba zbitych pionków=0)

„postaw" pionka na wybranym polu i sprawdź stratę symulując ruch przeciwnika

otrzymaną wartość odejmij od liczby zbitych pionków

jeżeli nie było pionków lub braku możliwego ruchu zakończ grę

wybierz pionka dającego maksymalny zysk i wykonaj ruch

Kolejna modyfikacja może polegać na budowie drzewa ruchów i oszacowaniu korzyści płynących z ruchu (sekwencji ruchów) w dłuższej perspektywie, podobnie jak to (czasami nieświadomie) czyni człowiek.

BUDOWANIE DRZEWA RUCHÓW

W drzewie ruchów każdy z węzłów musi posiadać o aktualnej pozycji pionków własnych i przeciwnika. Mówiąc krótko musi zawierać pełen opis planszy do gry (żeby można było obliczyć kolejne ruchy) oraz wskaźniki na wszystkich jego potomków. Można również wprowadzić informację o zysku/stracie wynikłej z wykonania ruchu w stosunku do stanu początkowego .
W zależności od rozmiaru problemu, można stosować pełne drzewa gry albo drzewa częściowe (o zadanej wysokości).
Po zakończeniu generacji drzewa należy wybrać optymalny ruch, kierując się np. zasadą minimaksu (wybierając gałąź dającą najmniejszą stratę).

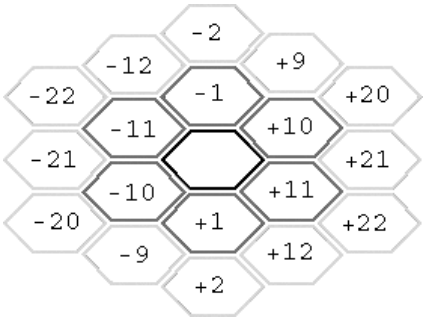
Algorytm generujący drzewo może mieć postać:

w korzeniu drzewa zapisz bieżącą sytuację na planszy
i = 0
dla każdego poziomu i < od zadanej wysokości
    dla każdego pionka
        dla każdego ruchu
            zasymuluj ruch i zapisz sytuację na planszy w nowym węźle na poziomie i+1
ze wszystkich liści wybierz jeden stosując założoną metodę oceny ruchu
przesuń się po niej o 1 poziom w jego kierunku

Pełne drzewo gry może zostać wygenerowane na początku rozgrywki, wówczas 'gra' będzie polegała na wyborze odpowiedniej krawędzi w drzewie.
Drzewo częściowe może być generowane dla każdego ruchu, albo rozgrywka może się toczyć jak przy drzewie pełnym z tą różnicą, że w momencie dotarcia do liścia zostanie wygenerowane na nowo.

HEXXAGON

W grze Hexxagon pionek może wykonać ruch na następujące pola liczone od jego aktualnej pozycji:



W przypadku pól sąsiednich następuje klonowanie pionka, w przypadku pól odleglejszych - skok. Postawienie pionka na polu sąsiadującym z pionkiem wrogim powoduje włączenie go do własnej drużyny.

Algorytm zachłanny

Bardzo prosty algorytm można zapisać w postaci:

dla każdego pionka

dla pól (y=x±1, y=x±2, y=x±9, y=x±10, y=x±11, y=x±12, y=x±20, y=x±21,y= x±22)

dla pól (y+1, y+10, y+11)

jeśli na polu znajduje się pionek przeciwnika zwiększ zysk

wybierz pole o maksymalnym zysku

jeżeli nie było pionków lub braku możliwego ruchu zakończ grę

ze wszystkich pionków wybierz ten, którego ruch da największy zysk i wykonaj go

przejmij wszystkie sąsiadujące pionki wroga

Algorytm ten wykona ruch optymalny w danym momencie ruch nie patrząc na skutki nim wykonane. Praktyka dowodzi, że algorytm ten przynosi bardzo dobre wyniki.

Algorytm mrowiskowy

Mrówka

Każda mrówka-pionek pamięta nie tylko swoją bieżącą pozycję, ale również całą przebytą do tej pory drogę. Będzie odnotowywała ponadto liczbę przejętych wrogich pionków .

Umiejętności naszych mrówek oprócz wyboru drogi, przemieszczania się i odkładania feromonu, będą poszerzone o możliwość przejmowania pionków przeciwnika i powoływania do życia kolejnych.
Mrówka dokonując wyboru ruchu musi wziąć pod uwagę wartość zysku wywołanego ruchem (ilości punktów zdobytych). Następnie dokona obliczenia współczynników prawdopodobieństwa (zależnych od zysku i wartości feromonu na analizowanej ścieżce) każdego możliwego ruchu, a następnie dokonuje wyboru na zasadzie ruletki.

dla każdej mrówki

dla pól (y=x±1, y=x±2, y=x±9, y=x±10, y=x±11, y=x±12, y=x±20, y=x±21,y= x±22)

dla pól (y+1, y+10, y+11)

policz współczynnik pk zgodnie ze wzorem:

oblicz losową wartość q $p_k(i,j) = \tau_{ij}(t) \cdot zysk(i,j)^{\beta}$

jeśli q<=q0

wybierz pole o maksymalnym pk

w przeciwnym wypadku

każdemu z pól przypisz przedział na osi liczbowej proporcjonalnie do wartości pk a następnie wylosuj liczbę z zakresu osi

wybierz pole któremu odpowiada wylosowany przedział

jeżeli nie było mrówek lub braku możliwego ruchu zakończ grę

wybierz mrówkę dającą największą wartość pk z otrzymanych i wykonaj ruch na to pole

przejmij wszystkie sąsiadujące pionki wroga

dokonaj aktualizacji lokalnej feromonu na ścieżce łączącej oba pola

$$\mathrm{zysk(i,j)} = \frac{\sum_{j \in wolne\_otoczenie} \mathrm{punkt} + typ\_ruchu}{\sum_{j \in otoczenie\_dalsze} \mathrm{punkt}}$$

$$punkt = \begin{cases} 1, & pion\_przeciwnika\_na\_polu\_j \\ 0, & przeciwnie \end{cases}$$

$$typ\_ruchu = \begin{cases} 1, & kolonowanie \\ 0, & skok \end{cases}$$

$wolne\_otoczenie = \{x, x \in otoczenie \wedge x \notin TABU\}$

$otoczenie = sąsiedztwo \cup sąsiedztwo2$

$sąsiedztwo = \{i \pm 1, i \pm 10, i \pm 11\}$

$sąsiedztwo2 = \{i \pm 22,.., i \pm 20, i \pm 12, i \pm 9, i \pm 2\}$

Tablica TABU.

Tablica TABU jest tablicą jednowymiarową, indeksowaną numerami pól. Są w niej zapisane informacje o polach, które są zabronione. Polem zabronionym jest pole, na którym znajduje się już jakaś mrówka lub pole wyłączone z planszy. Tablica ta jest wspólna dla obu kast mrówek. Jest ona odpowiednikiem planszy.

Macierz feromonu

Macierze feromonowe są osobne dla obu mrowisk (graczy). Macierz feromonowa to tablica zawierająca listy feromonowe dla każdego z pól planszy. Na liście takiej znajdują się wszystkie sąsiednie pola i wartość feromonu odłożona pomiędzy nimi.

Lokalna aktualizacja feromonu

Aktualizacja lokalna będzie dokonywana przez każdą mrówkę po wykonaniu przez nią ruchu. Lokalne uaktualnienie śladu wyraża się wzorem:

$$\tau_{ij}(t,t+1) = (1-\alpha)\cdot\tau_{ij}(t) + \alpha\cdot\Delta\tau_{ij}$$

gdzie:

$$\Delta\tau = \tau_0 \cdot \frac{przyrost\_punktów}{61}$$

61 – liczba pól planszy

Aktualizacja globalna.

Po każdej wygranej rozgrywce globalna aktualizacja feromonu dokonywana będzie przez wszystkie mrówki na całej przebytej drodze. W ten sposób wyróżniona zostanie cała sekwencja ruchów od pozycji startowej aż do ostatniego ruchu.

W celu większego zmobilizowania mrówek aktualizacja będzie również wykonywana w przypadku zwiększenia uzyskanej liczby punktów w kolejnej partii. Globalne wzmocnienie określa wzór:

$$\tau_{ij}(t,t+n) = (1-\alpha)\cdot\tau_{ij}(t) + \alpha\cdot\Delta\tau_{ij_K}$$

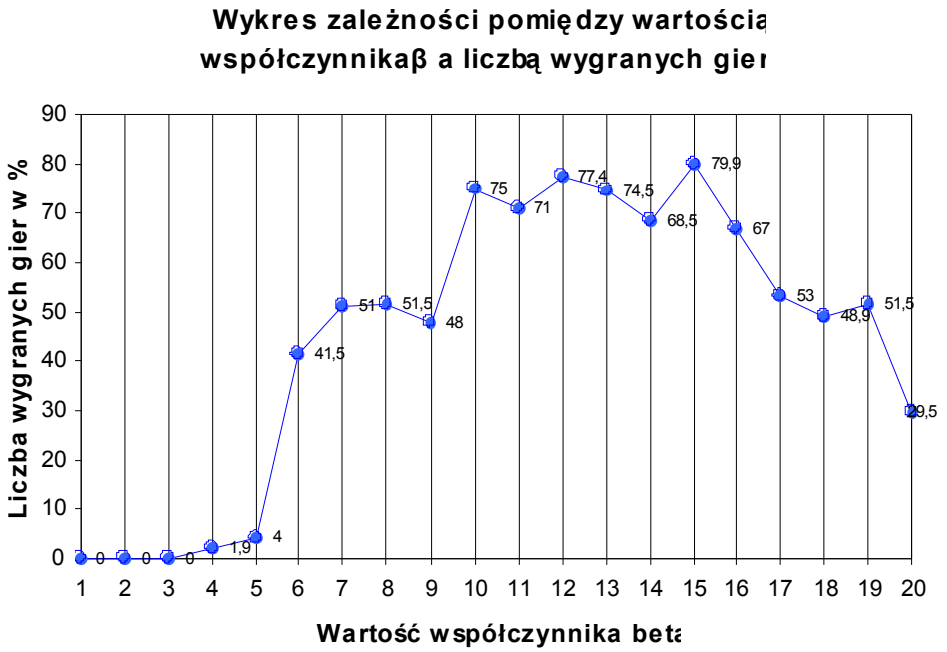
gdzie:

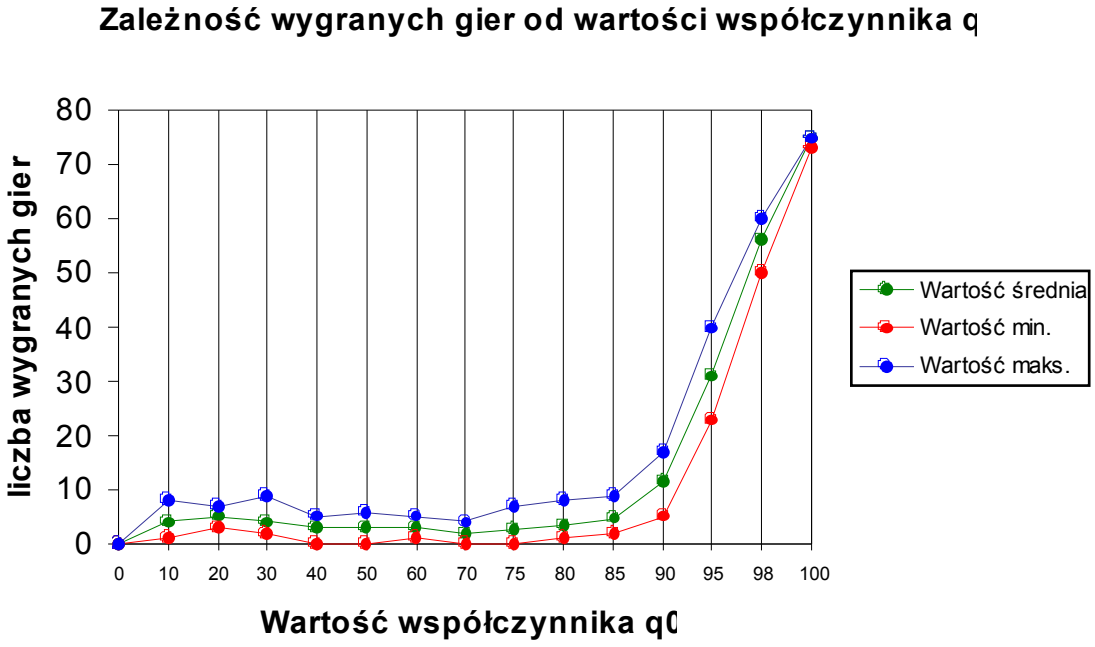
$$\Delta\tau_K(i,j) = \begin{cases} \frac{M}{61} & i,j \in L_K \\ 0 & i,j \notin L_K \end{cases}$$

L_k – droga zapamiętana przez najlepszą mrówkę

M – ilość zdobytych pionów przeciwnika

Wyznaczenie optymalnych parametrów algorytmu mrowiskowego dla gry Hexxagon





Wyniki uzyskane w przypadku dwóch równorzędnych graczy dla gry Hexxagon

W przypadku rozpoczynania przez gracza 1:

	Nr eksperymentu										Liczba wygranych partii	
Wygrane	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**średnio**	**%**
Gracz 1	487	493	472	488	481	528	528	505	488	501	497,1	49,8
Gracz 2	512	506	527	511	510	471	471	494	511	498	501,1	50,2

W przypadku rozpoczynania przez gracza 2:

	Nr eksperymentu										Liczba wygranych partii	
Wygrane	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**średnio**	**%**
Gracz 1	507	502	501	498	490	495	491	485	511	505	498,5	49,9
Gracz 2	492	497	498	501	509	504	508	514	488	494	500,5	50,1

Porównanie algorytmu mrowiskowego z zachłannym w grze Hexxagon

Gracz 1 – algorytm mrowiskowy o następujących parametrach:

Gracz 1	
Parametr	**Wartość**
α	0,01
τ0	0,002
β	14
q0	100

Gracz 2 – algorytm zachłanny

W przypadku rozpoczynania przez gracza 1:

	Nr eksperymentu										Liczba wygranych partii	
Wygrane	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**średnio**	**%**
Gracz 1	691	671	691	671	691	671	691	671	691	671	681	68,17
Gracz 2	308	328	308	328	308	328	308	328	308	328	318	31,83

W przypadku rozpoczynania przez gracza 2:

	Nr eksperymentu										Liczba wygranych partii	
Wygrane	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**średnio**	**%**
Gracz 1	671	691	671	691	671	691	671	691	671	691	681	68,17
Gracz 2	328	308	328	308	328	308	328	308	328	308	318	31,83