Go Puzzle

Hubert Jackowski, Nikodem Anzel

Opis problemu

Gra Go to klasyczna gra planszowa pochodząca z Chin, której zasady opierają się na otaczaniu przeciwnika oraz kontrolowaniu terytorium. Rozpatrywany system, Go Puzzle, to uproszczona wersja tej gry, która polega polega na znalezieniu optymalnego ciągu ruchów czarnego gracza, który w określonej liczbie tur stara się zdobyć jak najwięcej punktów poprzez wyłożenie określonej liczby kamieni na planszę z założeniem, że gracz biały spasował.

Plansza początkowo zawiera ustalony układ białych i czarnych kamieni. Celem jest takie rozmieszczenie dodatkowych czarnych kamieni, aby zmaksymalizować sumę liczby czarnych kamieni obecnych na planszy w ostatniej turze oraz liczby zbitych kamieni białych.

Programowanie Liniowe

Sformułowanie problemu

$N \in \mathbb{N}$ - rozmiar obrzeża planszy

$K \in \mathbb{N}$ - ilość kamieni do wystawienia

$T = K + 2$ - ilość plansz (tur) rozgrywki

$nh(i,j) = \{(i',j') \in \{(i \pm 1, j), (i, j \pm 1)\} \mid 0 \leq i' < N, 0 \leq j' < N\}$ - zbiór par indeksów sąsiednich pól według sąsiedztwa von Neumanna

$i, j \in \{0, 1, ..., N\}$ - indeksy wiersza i kolumny planszy

$t \in \{0, 1, ..., T\}$ - indeksy rozegranych tur

$b_{i,j} \in \{0, 1, 2\}$ - plansza początkowa:
0 oznacza, że pole (i, j)-e jest puste,
1 oznacza kamień czarny na polu (i, j)-tym,
2 oznacza kamień biały na polu (i, j)-tym

Zmienne decyzyjne

$x_{i,j,t} \in \{0, 1\}$ - macierz binarna decyzji postawienia czarnego kamienia na polu (i, j)-tym w turze t-tej lub jeżeli kamień czarny z tury t-1 pozostaje na planszy

$y_{i,j,t} \in \{0, 1\}$ - macierz binarna decyzji pozostania białego kamienia na polu (i, j)-tym w turze t-tej

$xl_{i,j,t} \in \{0, 1\}$ - macierz binarna decyzji istnienia oddechu czarnego kamienia na polu (i, j)-tym w turze t-tej

$yl_{i,j,t} \in \{0, 1\}$ - macierz binarna decyzji istnienia oddechu białych kamienia na polu (i, j)-tym w turze t-tej

Ograniczenia

Początkowe ustawienie czarnych kamieni

$$\forall_{i,j} \; x_{i,j,0} = \{1 \; jeśli \; b_{i,j} = 1, \; 0 \; wpp\} .$$

Początkowe ustawienie białych kamieni

$$\forall_{i,j} \; y_{i,j,0} = \{1 \; jeśli \; b_{i,j} = 2, \; 0 \; wpp\} .$$

Białe nie mogą być w miejscu gdzie tracą oddech

$$\forall_{i,j,t=1,\dots,T-1} \quad y_{i,j,t} = y_{i,j,t-1} \cdot yl_{i,j,t-1} .$$

Czarne wystawić możemy tylko na polach, gdzie będą mieć oddech

$$\forall_{i,j,t=1,\dots,T-1} \quad x_{i,j,t} \leq xl_{i,j,t-1} .$$

Czarne nie mogą zostać podniesione

$$\forall_{i,j,t=1,\dots,T-1} \quad x_{i,j,t} \geq x_{i,j,t-1} .$$

Na polu może znajdować się tylko jeden kolor kamienia naraz

$$\forall_{i,j,t} \quad x_{i,j,t} + y_{i,j,t} \leq 1.$$

W każdej turze oprócz zerowej wyłożyć możemy tylko jeden dodatkowy kamień czarny

$$\forall_{t=1,\dots,T-2} \sum_{i=0}^{N} \sum_{j=0}^{N} x_{i,j,t} - \sum_{i=0}^{N} \sum_{j=0}^{N} x_{i,j,t-1} \leq 1 .$$

W ostatniej turze nie wykładamy żadnego kamienia

$$\sum_{i=0}^{N}\sum_{j=0}^{N} x_{i,j,T-1} - \sum_{i=0}^{N}\sum_{j=0}^{N} x_{i,j,T-2} = 0 \ .$$

Biały ma oddech na polu (i, j)-ym jeśli jest ono puste, lub jeżeli znajdujący się tam kamień czarny jest częścią grupy sąsiadującej z polem pustym
$\forall_{i,j,t}\ yl_{i,j,t} = ((1 - x_{i,j,t}) \cdot (1 - y_{i,j,t})) \vee (y_{i,j,t} \wedge max_{(i',j')\in nh(i,j)}\ yl_{i',j',t})$.

Czarny ma oddech na polu (i, j)-ym jeśli jest ono puste, lub jeżeli znajdujący się tam kamień czarny jest częścią grupy sąsiadującej z polem pustym
$\forall_{i,j,t}\ xl_{i,j,t} = ((1 - x_{i,j,t}) \cdot (1 - y_{i,j,t})) \vee (x_{i,j,t} \wedge max_{(i',j')\in nh(i,j)}\ xl_{i',j',t})$.

Funkcja celu

$F(x, y) = \sum_{i=0}^{N}\sum_{j=0}^{N} x_{i,j,T-1} + \sum_{i=0}^{N}\sum_{j=0}^{N} y_{i,j,0} - \sum_{i=0}^{N}\sum_{j=0}^{N} y_{i,j,T-1}$ - Zdobyte punkty gracza

Należy znaleźć

$F(x^*, y^*) = max_{x,y} F(x, y)$

Rozwiązanie metaheurystyczne

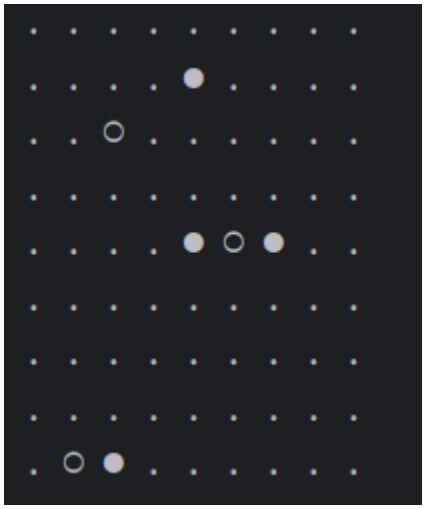
W celu rozwiązania postawionego problemu optymalizacyjnego, polegającego na znalezieniu optymalnego rozmieszczenia T=9 czarnych kamieni na planszy do grze w Go, zaimplementowano metaheurystykę symulowanego wyżarzania. Symulowane wyżarzanie to

probabilistyczna technika optymalizacyjna, inspirowana procesem wyżarzania w metalurgii. W procesie tym metal jest podgrzewany do wysokiej temperatury, a następnie powoli schładzany. Powolne chłodzenie pozwala atomom na uporządkowanie się w strukturę krystaliczną o minimalnej energii, co nadaje materiałowi pożądane właściwości (np. wytrzymałość). Gwałtowne schłodzenie prowadziłoby do zamrożenia atomów w stanie o wysokiej energii, tworząc kruchą strukturę z wieloma defektami.
Główną zaletą symulowanego wyżarzania jest zdolność do unikania optimów lokalnych. W przeciwieństwie do prostych algorytmów zachłannych (które akceptują tylko ruchy poprawiające wynik), SA na początku swojego działania (przy wysokiej temperaturze) z pewnym prawdopodobieństwem akceptuje również ruchy pogarszające aktualne rozwiązanie. Daje mu to szansę na "wydostanie się" z pułapki lokalnego optimum i eksplorację szerszej przestrzeni możliwych rozwiązań. W miarę spadku temperatury, prawdopodobieństwo akceptacji gorszych ruchów maleje, a algorytm zaczyna zachowywać się bardziej zachłannie, koncentrując się na doskonaleniu znalezionego dobrego regionu.

Implementacja problemu

W naszym projekcie, kluczowe elementy algorytmu SA zostały zdefiniowane następująco:

Plansza początkowa na której będziemy szukać rozwiązania:



Reprezentacja Rozwiązania: Pojedyncze rozwiązanie to lista T=9 krotek (x, y), które jednoznacznie określają pozycje nowych czarnych kamieni na planszy.

Funkcja Celu (objective): Aby ocenić jakość danego rozwiązania, zdefiniowano funkcję celu. W naszym przypadku jej wartość jest sumą dwóch składników:

Liczby zbitych białych kamieni (count_white_captures) oraz liczby postawionych kamieni (len(moves)), która w tym problemie jest stała i wynosi 9. Algorytm dąży do maksymalizacji wartości tej funkcji. Ruchy nielegalne (np. na zajęte pole) są silnie penalizowane przez zwrócenie bardzo niskiej wartości (-1e9).

Generowanie Sąsiada (generate_neighbor): Przejście od jednego rozwiązania do drugiego (sąsiedniego) odbywa się poprzez losową modyfikację. Funkcja generate_neighbor bierze aktualny zestaw 9 ruchów, losowo wybiera jeden z nich i zamienia jego koordynaty na inne, losowo wybrane wolne pole na planszy. To prosta, ale skuteczna metoda na eksplorację przestrzeni rozwiązań.

Mechanizm Akceptacji: W każdej iteracji, po wygenerowaniu nowego rozwiązania (sąsiada), podejmowana jest decyzja o jego akceptacji:

Jeśli nowe rozwiązanie jest lepsze od obecnego (delta > 0), jest ono akceptowane bezwarunkowo.

Jeśli nowe rozwiązanie jest gorsze (delta < 0), jest ono akceptowane z prawdopodobieństwem P = exp(delta / temp). Prawdopodobieństwo to zależy od dwóch czynników:

delta: Im gorsze jest nowe rozwiązanie, tym mniejsza szansa na akceptację.

temp: Im wyższa temperatura, tym większa szansa na akceptację.

Symulacja

W celu znalezienia jak najlepszego rozwiązania metaheurystycznego przeprowadziliśmy test jakości na poszczególnych parametrach modelu. Głównym wyznacznikiem tego czy dany parametr jest lepszy jest wartość funkcji celu, natomiast w przypadku gdy algorytm znalazł dla obu parametrów taką samą wartość funkcji celu braliśmy rozwiązanie o mniejszej złożoności obliczeniowej (krótszym czasie wykonywania algorytmu). Kluczowym elementem wpływającym na skuteczność algorytmu jest schemat chłodzenia, czyli sposób, w jaki temperatura (temp) jest obniżana. W naszym modelu przetestowaliśmy następujące schematy:

-Logarytmiczny schemat chłodzenia - dla którego testowaliśmy parametry takie jak : liczba iteracji oraz temperatura początkowa
-Geometryczny schemat chłodzenia - dla którego testowaliśmy parametry takie jak : współczynnik alpha chłodzenia oraz temperatura początkowa
-Liniowy schemat chłodzenia - dla którego testowaliśmy parametry takie jak : liczba iteracji oraz temperatura początkowa.

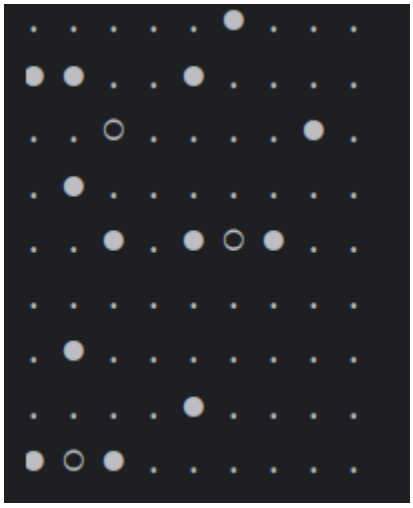
Następnie po wyłonieniu najlepszych parametrów dla każdego schematu chłodzenia porównałem je ze sobą w celu wyłonienia najlepszego.

Wyniki

Dla chłodzenia logarytmicznego przyjąłem sobie następujące parametry początkowe:
temperatura końcowa = 0.01
liczba iteracji ∈ {100, 1000, 10000}
temperatura początkowa ∈ {N*N*T, N*N*T * ¾, N*N*T *½ }

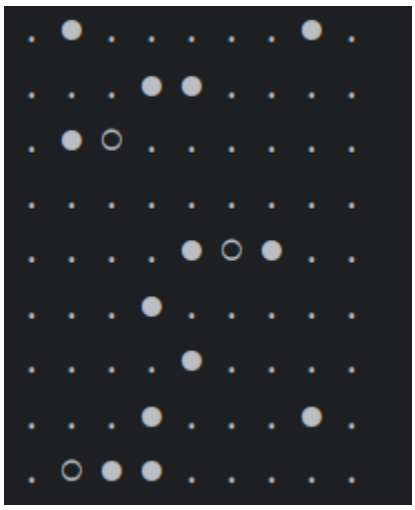
Wykonałem zatem 9 kombinacji testując każdą możliwość ułożenia parametrów i otrzymałem następujące wyniki:

Dla liczby iteracji równej 100 oraz temperatury początkowej równej N*N*T:
Najlepszy wynik równy 9 (żaden biały kamień nie został zbity)
Plansza końcowa:

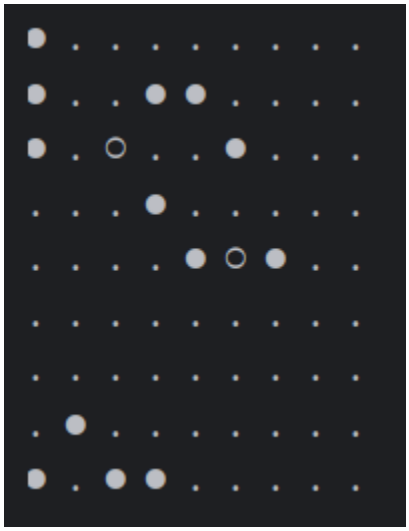


Dla liczby iteracji równej 100 oraz temperatury początkowej równej N*N*T * ¾:
Najlepszy wynik równy 9 (żaden biały kamień nie został zbity)

Plansza końcowa:

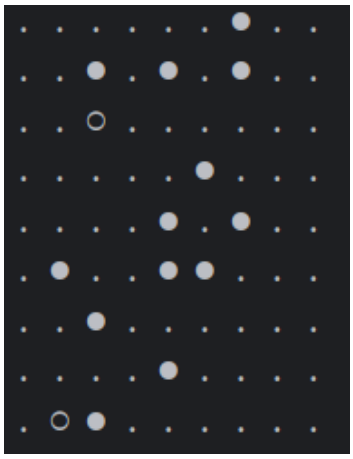


Dla liczby iteracji równej 100 oraz temperatury początkowej równej N*N*T * ½
Najlepszy wynik równy 10, plansza końcowa:



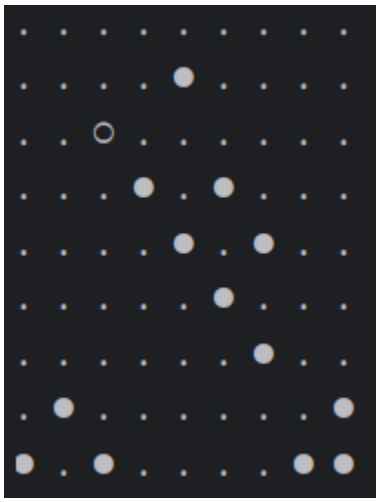
Dla liczby iteracji równej 1000 oraz temperatury początkowej równej N*N*T:

Najlepszy wynik równy 10, plansza końcowa:



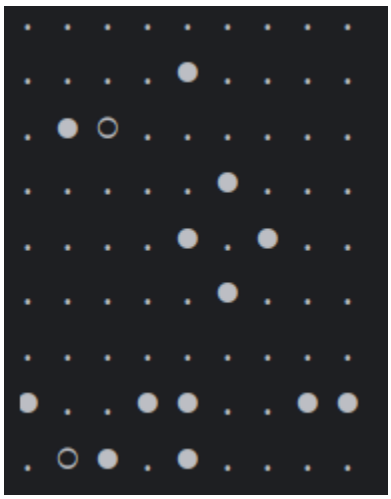
Dla liczby iteracji równej 1000 oraz temperatury początkowej równej N*N*T *¾:

Najlepszy wynik równy 11, plansza końcowa:

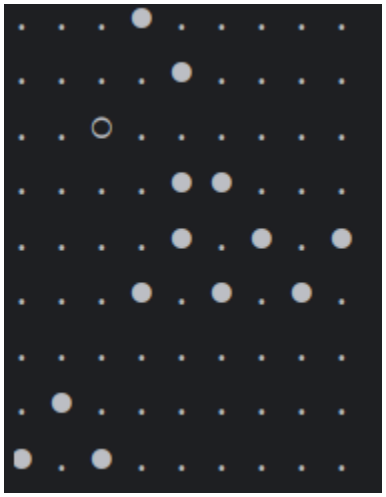


Dla liczby iteracji równej 1000 oraz temperatury początkowej równej N*N*T *½ :

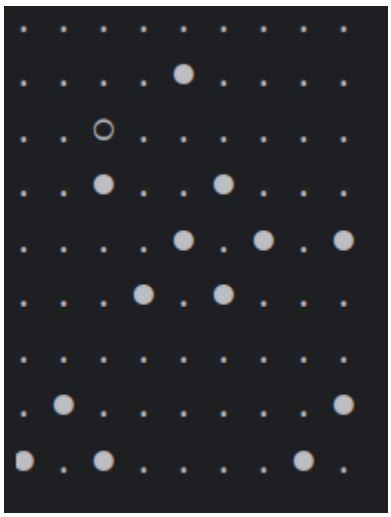
Najlepszy wynik równy 10, plansza końcowa:



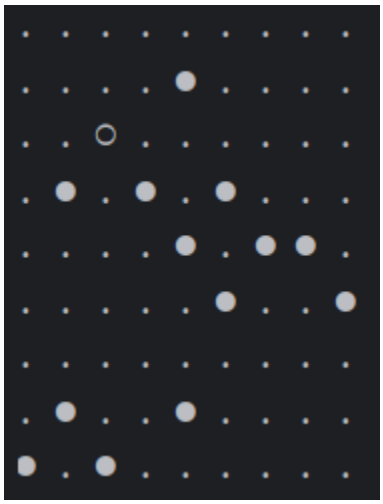
Dla liczby iteracji równej 10000 oraz temperatury początkowej równej N*N*T:
Najlepszy wynik równy 11, plansza końcowa:



Dla liczby iteracji równej 10000 oraz temperatury początkowej równej N*N*T*3/4:
Najlepszy wynik równy 11, plansza końcowa:



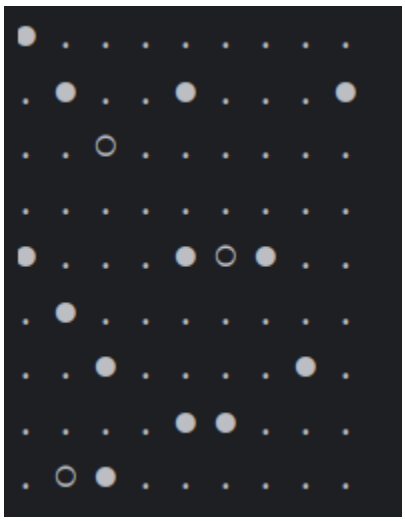
Dla liczby iteracji równej 10000 oraz temperatury początkowej równej N*N*T*1/2:
Najlepszy wynik równy 11, plansza końcowa:



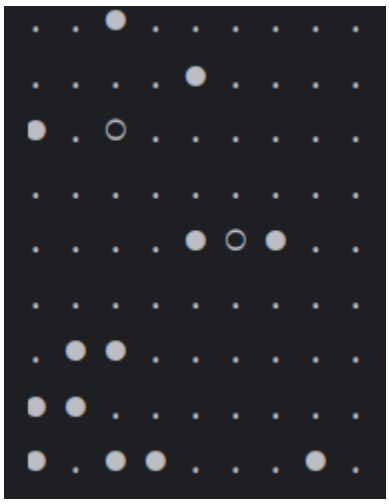
Zatem wyszło że najlepszym doborem parametrów dla tego sposobu obniżania temperatury jest temperatura początkowa równa N*N*T * ¾ oraz liczba iteracji równa 1000 przekłada się to na wynik równy 11.

Dla geometrycznego sposobu chłodzenia przyjąłem sobie następujące parametry :
Temperatura końcowa równa 0.01
Współczynnik alpha ∈ {0.99, 0.9, 0.7}
Temperatura początkowa ∈ {N*N*T, N*N*T*¾, N*N*T*½ }

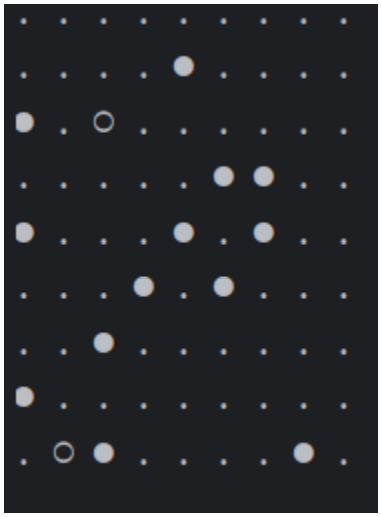
Dla współczynnika alpha równego 0.99 oraz temperatury początkowej równej N*N*T:
Najlepszy wynik równy 9, plansza końcowa:



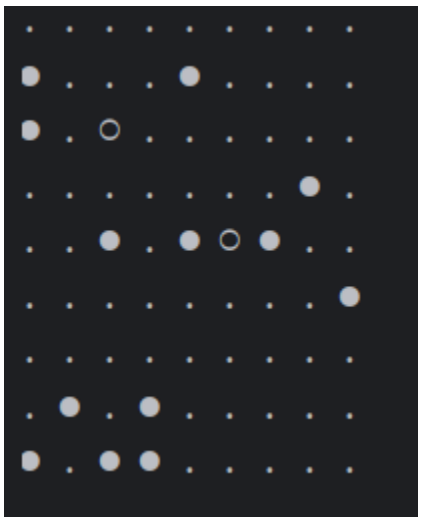
Dla współczynnika alpha równego 0.99 oraz temperatury początkowej równej N*N*T * ¾ :
Najlepszy wynik równy 10, plansza końcowa:



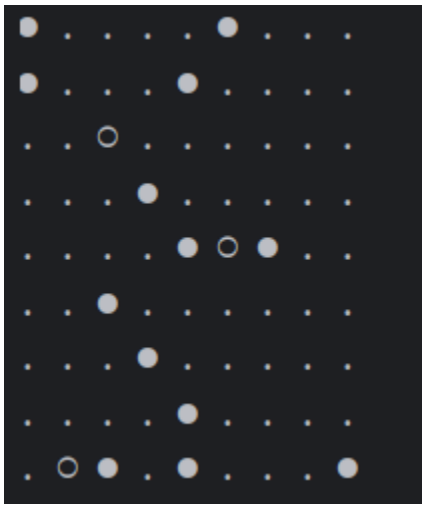
Dla współczynnika alpha równego 0.99 oraz temperatury początkowej równej N*N*T * 1/2 :
Najlepszy wynik równy 10, plansza końcowa:



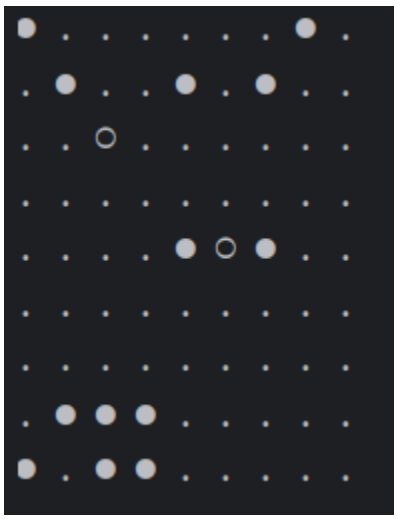
Dla współczynnika alpha równego 0.9 oraz temperatury początkowej równej N*N*T
Najlepszy wynik równy 10, plansza końcowa:



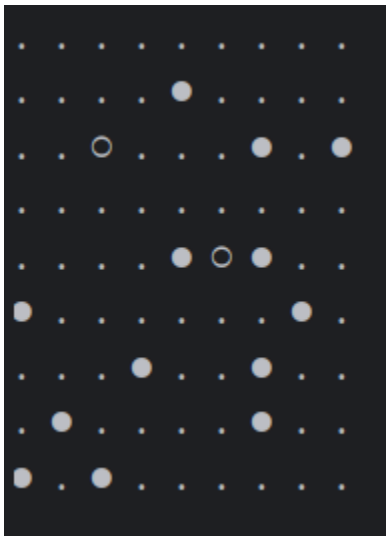
Dla współczynnika alpha równego 0.9 oraz temperatury początkowej równej N*N*T * 3/4
Najlepszy wynik równy 9, plansza końcowa:



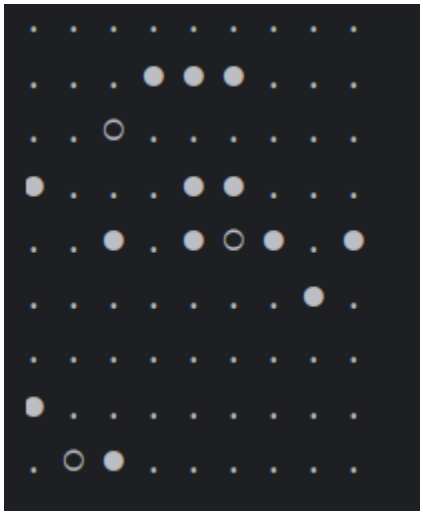
Dla współczynnika alpha równego 0.9 oraz temperatury początkowej równej N*N*T * 1/2
Najlepszy wynik równy 10, plansza końcowa:



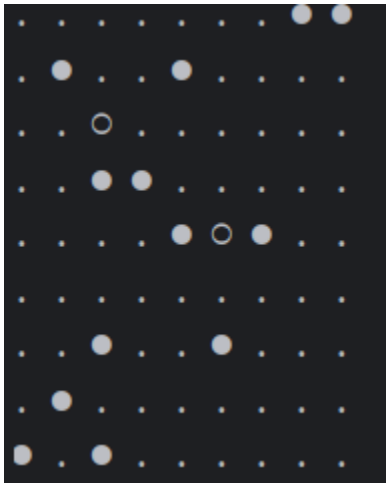
Dla współczynnika alpha równego 0.7 oraz temperatury początkowej równej N*N*T
Najlepszy wynik równy 10, plansza końcowa :



Dla współczynnika alpha równego 0.7 oraz temperatury początkowej równej N*N*T*3/4
Najlepszy wynik równy 10, plansza końcowa :



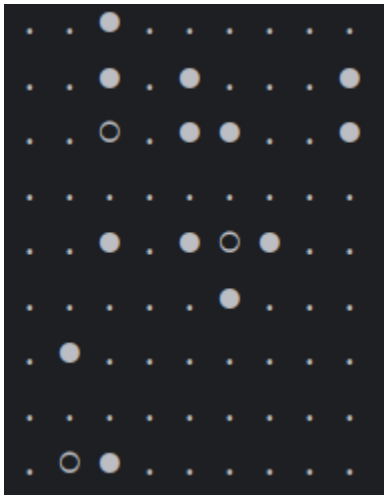
Dla współczynnika alpha równego 0.7 oraz temperatury początkowej równej N*N*T*1/2
Najlepszy wynik równy 10, plansza końcowa :



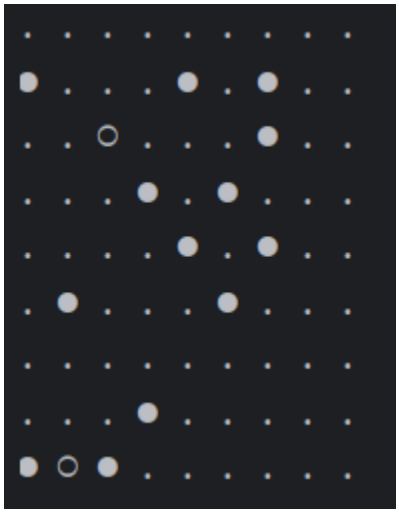
Dla geometrycznego sposobu chłodzenia najlepszym wynikiem jest wynik 10. Najszybciej osiągnęliśmy to przy parametrach modelu równych : Dla współczynnika alpha równego 0.7 oraz temperatury początkowej równej N*N*T*1/2

Dla chłodzenia liniowego:
Przyjąłem sobie następujące parametry:
Temperatura końcowa: 0.01
Liczba iteracji ∈ {100, 1000, 10000}
Temperatura początkowa ∈ {N*N*T, N*N*T*3/4 , N*N*T*½}

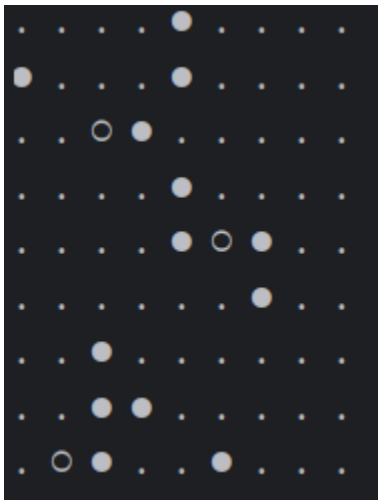
Dla liczby iteracji równej 100 oraz temperatury początkowej równej N*N*T:
Wynik równy 9, plansza końcowa równa :



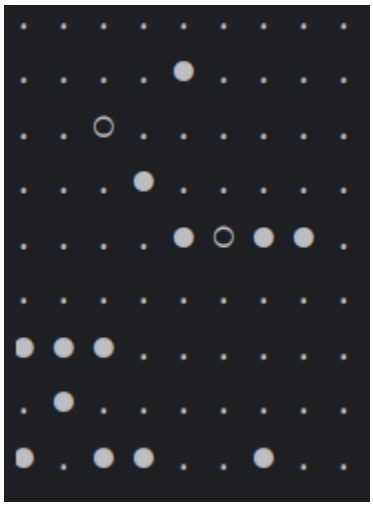
Dla liczby iteracji równej 100 oraz temperatury początkowej równej N*N*T*3/4:
Wynik równy 10, plansza końcowa równa :



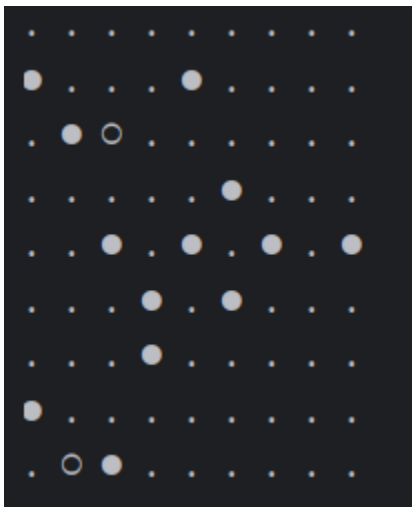
Dla liczby iteracji równej 100 oraz temperatury początkowej równej N*N*T*1/2:
Wynik równy 9, plansza końcowa równa :



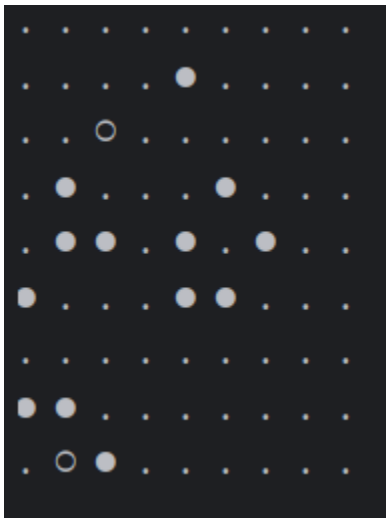
Dla liczby iteracji równej 1000 oraz temperatury początkowej równej N*N*T:
Wynik równy 10, plansza końcowa równa :



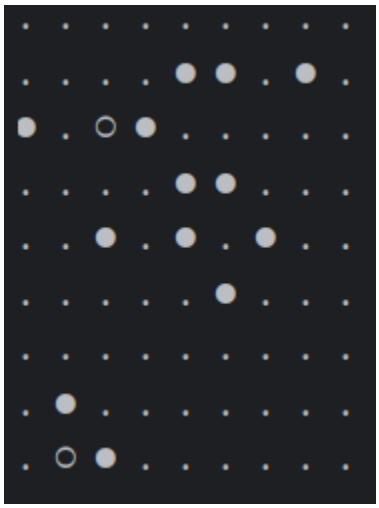
Dla liczby iteracji równej 1000 oraz temperatury początkowej równej N*N*T*3/4:
Wynik równy 10, plansza końcowa równa :



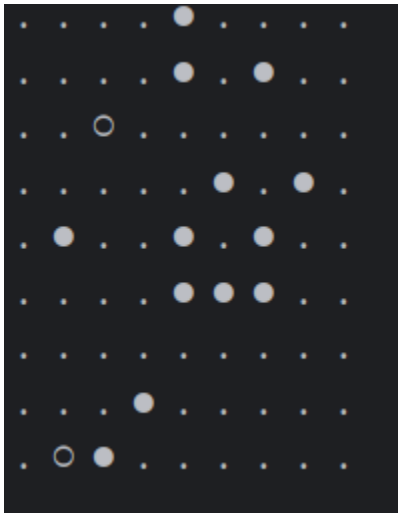
Dla liczby iteracji równej 1000 oraz temperatury początkowej równej N*N*T*1/2:
Wynik równy 10, plansza końcowa równa :



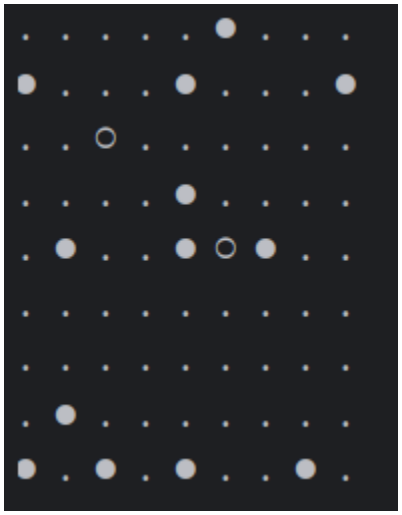
Dla liczby iteracji równej 10000 oraz temperatury początkowej równej N*N*T:
Wynik równy 10, plansza końcowa równa :



Dla liczby iteracji równej 10000 oraz temperatury początkowej równej N*N*T*3/4:
Wynik równy 10, plansza końcowa równa :



Dla liczby iteracji równej 10000 oraz temperatury początkowej równej N*N*T*1/2:
Wynik równy 10, plansza końcowa równa :



Dla liniowej metody zmniejszania wartości temperatury najlepszym wynikiem (uwzględniając optymalizację czasową) jest : Dla liczby iteracji równej 100 oraz temperatury początkowej równej N*N*T*¾ - wynik równy 10

Z powyższej symulacji wynika iż najlepszą metodą oraz najlepszymi parametrami naszej metaheurystyki jest logarytmiczna metoda zmniejszania temperatury o parametrach równych : temperatura początkowa równa N*N*T

* ¾ oraz liczba iteracji równa 1000 co przekłada się na wynik równy 11.

Porównanie

Celem niniejszego rozdziału jest ocena jakości i efektywności zaimplementowanej metaheurystyki Symulowanego Wyżarzania (SA) w kontekście rozwiązywania problemu Go Puzzle. Aby uzyskać miarodajne wyniki, konieczne jest porównanie rozwiązań generowanych przez SA z rozwiązaniami optymalnymi, znalezionymi przy użyciu metody dokładnej. Poniżej przedstawiono szczegółowo kolejne etapy projektowania i realizacji eksperymentu.

Eksperyment przeprowadzono na instancjach o zredukowanej skali: plansza 5x5, na której należy optymalnie rozmieścić K=3 czarne kamienie. Z uwagi na ograniczone możliwości licencyjne zdecydowaliśmy się właśnie na taki rozmiar planszy który jest jednocześnie wystarczająco złożony, aby stanowić wyzwanie dla metaheurystyki jak i nie wymaga posiadania licencji.

Algorytm SA był uruchamiany z konfiguracją parametrów, która została wcześniej zidentyfikowana jako najkorzystniejsza:

Schemat chłodzenia: logarytmiczny

Liczba iteracji: 1000

Temperatura początkowa: $T_{start} = N^2 * T * 0.75$

Eksperyment polegał na wykonaniu pętli N=50 razy. W każdej iteracji:

Generowano nową, losową planszę startową 5x5 z 3 czarnymi i 3 białymi kamieniami.

Uruchamiano metodę dokładną (rozwiązanie za pomocą solvera gurobipy) w celu znalezienia wyniku optymalnego (optimal_score).

Uruchamiano metaheurystykę SA w celu znalezienia jej propozycji rozwiązania (sa_score).

Rejestrowano wyniki oraz czas wykonania obu algorytmów.

Wskaźnik Skuteczności (Success Rate): Określa, w jakim procencie przypadków metaheurystyka była w stanie znaleźć rozwiązanie o wartości równej globalnemu optimum.
Skuteczność = (Liczba sukcesów / Całkowita liczba testów) * 100%

Średnia Jakość Względna (Average Quality Ratio): Mierzy, jak blisko optimum znajduje się rozwiązanie SA, uśredniając stosunek wyniku heurystycznego do optymalnego po wszystkich testach. Wartość bliska 100% świadczy o bardzo wysokiej jakości generowanych rozwiązań, nawet jeśli nie są one idealnie optymalne.
Jakość = (1/N) * Σ (sa_score / optimal_score) * 100%

Wyniki prezentują się następująco :

```
========================================
=== PODSUMOWANIE EKSPERYMENTU ===
========================================
Liczba przeprowadzonych poprawnych testów: 50
----------------------------------------
1. Wskaźnik Skuteczności (znaleziono optimum): 94.00%
2. Średnia Jakość Względna (do optimum): 98.67%
----------------------------------------
Średni czas działania SA: 0.2678 sekundy
Średni czas działania Metody Dokładnej: 0.1214 sekundy
========================================
```

Wnioski końcowe

Wyniki eksperymentu dowodzą, że zaimplementowana metaheurystyka Symulowanego Wyżarzania jest wyjątkowo skutecznym narzędziem do rozwiązywania postawionego problemu Go Puzzle na badanej skali.

Wskaźnik Skuteczności na poziomie 94.00% jest wynikiem doskonałym. Oznacza to, że w 47 na 50 losowych przypadków algorytm SA, mimo swojej probabilistycznej natury i ograniczonego budżetu obliczeniowego (zaledwie 1000 iteracji), był w stanie znaleźć rozwiązanie o wartości równej globalnemu optimum. Taka powtarzalność świadczy o dużej niezawodności metody dla tej klasy problemów.

Średnia Jakość Względna wynosząca 98.67% dodatkowo wzmacnia ten wniosek. Metryka ta pokazuje, że nawet w

tych nielicznych przypadkach (6%), w których SA nie znalazło idealnego rozwiązania, znalezione przez nie wyniki były niezwykle bliskie optimum – średnio stanowiły 98.67% jego wartości. Oznacza to, że "porażki" algorytmu nie są katastrofalne, a sprowadzają się do znalezienia rozwiązania suboptymalnego o znikomej stracie jakości.