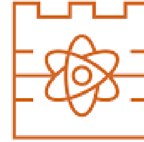


Modelowanie komputerowe

Symulacja wyżarzania – Metoda Monte Carlo

Autorzy:
Budziło Maria,
Kubański Marcin,
Stypuła Hiacynta

25.01.2022

1. Wstęp. Opis problemu

Wyżarzanie to proces chłodzenia stopionej substancji, którego celem jest skondensowanie materii w krystaliczne ciało stałe. Wyżarzanie może być traktowane jako proces optymalizacji. Konfiguracja układu podczas wyżarzania jest określona przez zbiór położeń atomów r_i. Konfiguracja układu jest ważona przez jej współczynnik prawdopodobieństwa Boltzmanna:

$$exp(-E(r_i)/kT),$$

gdzie $E(r_i)$ jest energią konfiguracji, k jest stałą Boltzmanna, a T jest temperaturą. Kiedy substancja jest poddawana wyżarzaniu, jest ona utrzymywana w każdej temperaturze przez czas wystarczająco długi, aby osiągnąć równowagę termiczną. Problemem, z którym się zetknięto było osiągnięcie procedury symulacji wyżarzania.

2. Kryterium Metropolis

Technika iteracyjnego udoskonalania dla optymalizacji kombinatorycznej może być porównana do szybkiego hartowania stopionych metali. Podczas szybkiego hartowania stopionej substancji energia jest gwałtownie odbierana z układu poprzez kontakt z potężnym zimnym substratem. W wyniku gwałtownego schładzania powstają stany metastabilne układu; w metalurgii w wyniku gwałtownego schładzania otrzymuje się substancję szklistą, a nie krystaliczne ciało stałe. Analogia pomiędzy iteracyjnym udoskonalaniem i szybkim chłodzeniem metali wynika z faktu, że iteracyjne usprawnianie i szybkie chłodzenie metali akceptuje tylko te konfiguracje systemu, które zmniejszają funkcję kondycji. W procesie wyżarzania (powol-

nego chłodzenia) nowa konfiguracja układu, która nie poprawia funkcji kosztu, jest akceptowana na podstawie współczynnika prawdopodobieństwa Boltzmanna tej konfiguracji. To kryterium akceptacji nowego stanu systemu nazywane jest **kryterium Metropolis**.

3. Rozwiązanie problemu zbyt skrajnych temperatur

Jeśli temperatura początkowa jest zbyt niska, proces bardzo szybko ulega zahamowaniu i znaleźć można tylko lokalne optimum. Jeśli temperatura początkowa jest zbyt wysoka, proces przebiega bardzo wolno. Do przeszukiwania używane jest tylko jedno rozwiązanie, co zwiększa prawdopodobieństwo utknięcia rozwiązania w lokalnym optimum. Zmiana temperatury oparta jest na zewnętrznej procedurze, która nie jest związana z aktualną jakością rozwiązania, czyli tempo zmiany temperatury jest niezależne od jakości rozwiązania. Problemy te mogą być rozwiązane poprzez zastosowanie zbioru zamiast pojedynczego rozwiązania. Mechanizm wyżarzania może być również sprzężony z jakością bieżącego rozwiązania poprzez uwrażliwienie szybkości zmiany temperatury na jakość rozwiązania.

4. Algorytm bazowy

Poniżej przedstawiona jest procedura symulowanego wyżarzania. Symulowane wyżarzanie polega w istocie na powtarzaniu procedury Metropolis dla różnych temperatur. Temperatura jest stopniowo obniżana w każdej iteracji algorytmu symulowanego wyżarzania.

Procedura symulowanego wyżarzania:

początek

t <- 0

ustawiamy temperaturę T

wybieramy losowo aktualny ciąg v_c

oceniamy sprawność v_c

powtarzamy

powtarzamy

wybieramy nowy ciąg v_n

w pobliżu v_c

poprzez odwrócenie pojedynczego bitu v_c

jeśli f(v_c)< f(v_n)

wtedy v_c <- v_n

inaczej, jeśli losowo[0,1] < exp((f(v_n) - f(v_c)/T)

wtedy v_c <- v_n

dopóki (warunek zakończenia)

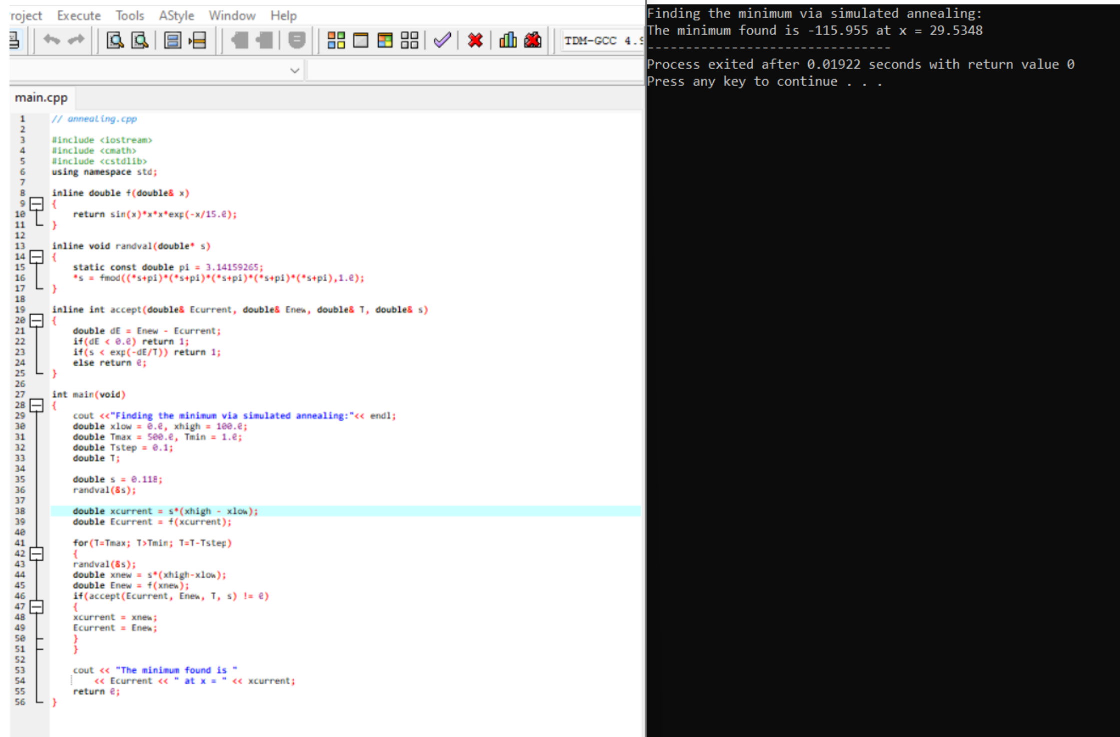
T <- g(T,t)

t <- t+ 1

aż do (kryterium zatrzymania)

koniec

5. Rozwiązanie problemu. Implementacja kodu

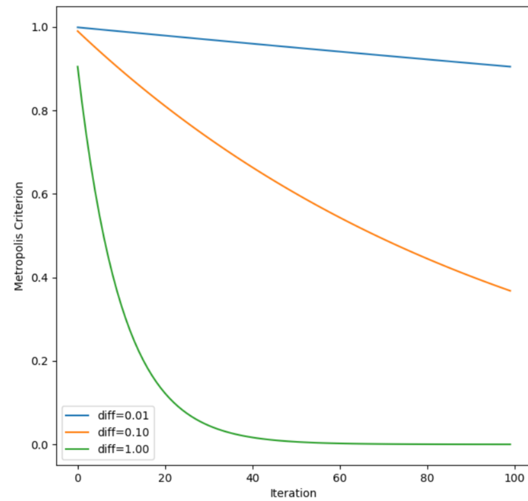


6. Porównanie z innymi rozwiązaniami

wyz.py

```python

# explore metropolis acceptance criterion for simulated annealing
from math import exp
from matplotlib import pyplot
# total iterations of algorithm
iterations = 100
# initial temperature
initial_temp = 10
# array of iterations from 0 to iterations - 1
iterations = [i for i in range(iterations)]
# temperatures for each iterations
temperatures = [initial_temp/float(i + 1) for i in iterations]
# metropolis acceptance criterion
differences = [0.01, 0.1, 1.0]
for d in differences:
    metropolis = [exp(-d/t) for t in temperatures]
    # plot iterations vs metropolis
    label = 'diff=%.2f' % d
    pyplot.plot(iterations, metropolis, label=label)
# inalize plot
pyplot.xlabel('Iteration')
pyplot.ylabel('Metropolis Criterion')
pyplot.legend()
pyplot.show()
```



7. Bibliografia

Materiały udostępnione w pliku „`GeneticAlgorithms.pdf`”