Sprawozdanie

Metaheurestyki inspirowane naturą w optymalizacji kombinatorycznej

Amadeusz Filipek

Laboratorium komputerowe WFiIS AGH

1. Wstęp

Celem ćwiczenia jest wykorzystanie algorytmu symulowanego wyżarzania do rozwiązania zadanego problemu kombinatorycznego.

Algorytm symulowanego wyżarzania jest klasyczną metodą optymalizacji zainspirowanej zjawiskami zachodzacymi w naturze mianowicie dynamiką układu stygnącego ciała stałego. Przeszukiwanie obszaru rozwiązań odbywa się w sposób iteracyjny i jest generowane losując nowe, poblliskie rozwiązanie. Algorytm ten wykorzystuje parametr zwany temperaturą, który określa prawdopodobieństwo przyjęcia nowego gorszego rozwiązania w kolejnym kroku iteracji. Przyjęcie rozwiązań gorszych zapobiega utykaniu algorytmu w minimach lokalnych. Temperatura maleje wraz z kolejnymi iteracjami algorytmu, zatem prawdopodobieństwo akceptacji gorszych rozwiązań również. Wobec tego algorytm na początku przeszukuje w sposób dynamiczny duży obszar rozwiązań a w dalszym etapie iteracji dokładniej przeszukuje bliskie otoczenie obszaru do którego doszedł.

Problem optymalizacji kombinatorycznej posiada skończony zbiór rozwiązań co oznacza, że optymalizowana funkcja kosztu lub celu ma dyskretny zbiór argumentów. Przykładowymi problemami optymalizacji kombinatorycznej są problem komiwojażera oraz problem minimalnego drzewa rozpinającego.

2. Realizacja ćwiczenia

W ramach ćwiczenia zaimplementowano algorytm symulowanego wyżarzania i wykorzystano go do problemu szeregowania zadań w systemie multiprocesorowym. Dana jest macierz informująca o tym ile czasu zajmuje i – temu procesorowi rozwiązanie j – te zadania. Procesory wykonują przypisane im zadania jeden po drugim niezależnie. Rozwiązanie problemu polega na znalezieniu takiej kombinacji przypisanych poszczególnych zadań do procesorów aby sumaryczny czas pracy najbardziej obciążonego procesora był najkrótszy. Sumaryczny czas pracy najbardziej obciążonego procesora określany jest mianem funkcji kosztu, dla której szukane jest minimum globalne. W ćwiczeniu dane jest 35 zadań, które należy porozdzielać na 11 procesorów. Warto policzyć ile kombinacji należaloby sprawdzić metodą „exhaustive search” (brute force) aby znaleźć dokładnie najlepsze rozwiązanie. Liczba wszystkich możliwości jest równa wariacji z powtórzeniami, gdzie każdemu zadaniu należy przydzielić jeden z 11 procesorów, zadań jest 35 więc:

$$W_{11}^{35} = 11^{35} > 10^{35}$$

Widać zatem, że liczba wszystkich możliwości jest ogromna i metoda brute force potrzebowałaby bardzo dużo czasu i nakładów obliczeniowych aby znaleźć rozwiązanie optymalne. Ustalono $N = 1000$ iteracji algorytmu. Rozwiązanie początkowe inicjalizowane jest w sposób losowy. Rozwiązanie nowe

generowane jest w sposób następujący: losowane jest jedno zadanie i następnie dla wylosowanego zadania losowany jest nowy procesor. W ten sposób rozwiązanie nowe od aktualnego różni się jednym zadaniem przesuniętym na inny procesor. Dla nowego zadania liczona jest funkcja kosztu i wywoływana jest reguła Metropolisa. Reguła ta ma za zadanie zadecydować czy nowe rozwiązanie jest akceptowane czy odrzucane na rzecz starego. Reguła ta sprawdza czy wartość funkcji kosztu dla nowego rozwiązania jest mniejsza od wartości dla rozwiązania starego. Jeśli tak to nowe rozwiązanie jest akceptowane. Jeśli nie to losowana jest liczba z przedziału $[0, 1]$ i sprawdzany jest warunek:

$$U(0,1) > exp\left[\frac{-\Delta c}{T(i)}\right]$$

gdzie Δc to różnica wartości funkcji kosztu dla rozwiązania nowego i starego oraz $T(i)$ to temperatura w i – tym kroku iteracji, która wyrażona jest wzorem:

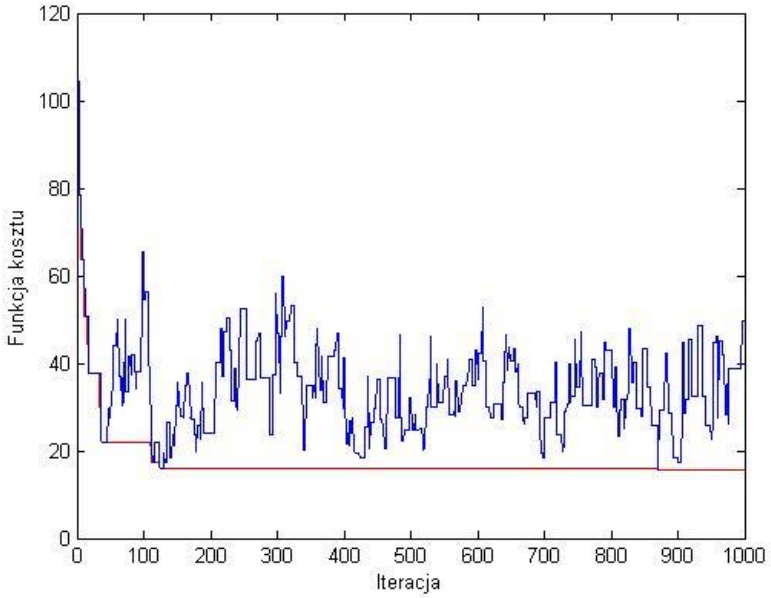
$$T(i) = \frac{T_0}{i}$$

Jeśli warunek ten jest spełniony to nowe rozwiązanie jest akceptowane pomimo, że uzyskuje niższą wartość funkcji kosztu. Prawdopodobieństwo przyjęcia rozwiązania gorszego zależy od temperatury, w szczególności od ustalonej temperatury początkowej T_0. Temperatura początkowa ustalana jest arbitralnie. Zbadano jak ustalone wartości temperatury początkowej T_0 wpływają na wyniki pracy algorytmu. Zweryfikowano także wyniki pracy algorytmu przy zmianie operatora generowania sąsiedztwa z jednego zadania na zamianę dwóch zadań. Każdą konfigurację algorytmu wywołano 20 razy ze względu na statystyczny charakter pracy algorytmu. Uzyskane średnie wartości funkcji kosztu wraz z ich odchyleniem standardowym przedstawione są w poniższej tabeli.

Tabela 1. Wartości średnie oraz odchylenia standardowe uzyskanych wartości funkcji kosztu dla dwóch operatorów generowania sąsiedztwa przy różnych temperaturach poczatkowych.

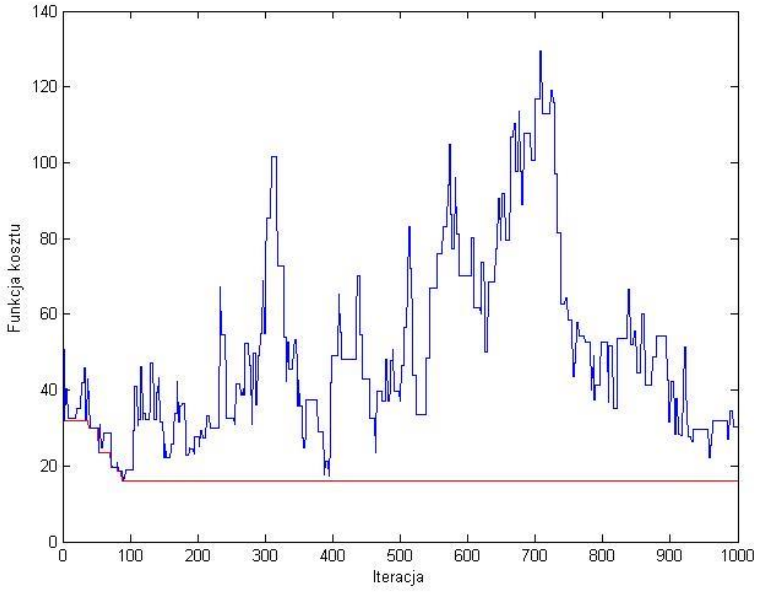
	zmiana 1 zadania		zmiana 2 zadań	
T_0	μ	σ	μ	σ
1	16.09	2.17	16.23	2.07
5	15.34	1.63	15.39	1.81
10	15.98	2.19	15.79	1.6
50	15.64	2.27	15.6	2.15
100	14.78	2.14	15.75	1.72
500	15.56	2.05	16.45	2.06
1000	16.57	2.33	14.79	2.35

Uzyskane wyniki są sobie bliskie co do wartości. Wyniki uzyskane przy użyciu jednokrotnej zmiany zadania między procesorami mają najmniejsze wartości dla pośrednich temperatur początkowych. Ta sama zależność widoczna jest dla dwukrotnej zmiany zadań między procesorami, jednakże dla $T_0 = 1000$ uzyskano wartość najmniejszą (najlepszą). Co więcej, operator dwukrotnej zmiany zadań uzyskał mniejsze wartości odchyleń standardowych w 4 na 7 przypadkach. Dla dwóch najlepszych rezultatów (oznaczonych szarym tłem w tabeli 1.) zbadano wartości przystosowania oraz najlepszego znalezionego przystosowania w kolejnych iteracjach algorytmu. Rezultaty przedstawiono na poniższych wykresach.



Wykres 1. Wartość funkcji kosztu w kolejnych iteracjach algorytmu na niebiesko, wartość najlepsza funkcji kosztu w kolejnych iteracjach algorytmu na czerwono, dla $T_0 = 100$ *przy jednokrotnej zmianie zadania*

Na powyższym wykresie widać, że przystosowanie w kolejnych krokach iteracji ma charakter oscylacyjny. Na początku działania algorytmu najlepsza wartość szybko maleje i ustala się na ok. 130 kroku iteracji, jednakże algorytmowi udaje się znaleźć wartość lepszą w około 870 kroku iteracji. Na wykresie widać także, że wraz ze spadkiem temperatury oscylacje przystosowania maleją i algorytm lepiej penetruje otoczenie dobrego rozwiązania, co skutkuje znalezieniem rozwiązania minimalnie lepszego.



Wykres 2. Wartość funkcji kosztu w kolejnych iteracjach algorytmu na niebiesko, wartość najlepsza funkcji kosztu w kolejnych iteracjach algorytmu na czerwono, dla $T_0 = 1000$ *przy dwukrotnej zmianie zadania*

Wartości uzyskane na powyższym wykresie wykazują znaczne oscylacje, dużo większe niż rezultaty przedstawione na wykresie 1. Należy zaznaczyć, że algorytm szybko bo w poniżej 100 iteracjach znajduje kombinację o najlepszym przystosowaniu, której nie udaje się już przekroczyć w późniejszym etapie obliczeń. Silne oscylacje związane są z faktem zarówno wysokiej temperatury początkowej jak i zmiany dwóch zadań w jednym kroku iteracji, co razem powoduje silne, chaotyczne przeszukiwanie przestrzeni rozwiązań. Co z jednej strony jest dobre, ponieważ algorytm szybko wychodzi z minimów lokalnych, jednakże algorytm nie penetruje dokładniej obszaru dobrego rozwiązania.

3. Podsumowanie

W ramach ćwiczenia zaimplementowano algorytm symulowanego wyżarzania i wykorzystano go do problemu szeregowania zadań w układzie wieloprocesorowym. Zbadano wyniki pracy algorytmu w zależności od parametru temperatury początkowej a także dla dwóch konfiguracji operatora generacji nowego rozwiązania. Zaobserwowano, że algorytm uzyskuje najlepsze wyniki przy najmniejszych odchyleniach standardowych dla pośrednich wartości temperatury początkowej, przy czym różnice te są małe. Przy zamianie operatora generacji nowego rozwiązania z jednokrotnej zmiany zadania na dwukrotną nie zaobserwowano istotnych zmian w rezultatach. Zbadano także przebieg pracy algorytmu dla dwóch przypadków dla $\mathrm{T}_0 = 100$ przy jednokrotnej zmianie zadania oraz dla $\mathrm{T}_0 = 1000$ przy dwukrotnej zmianie zadania. Zaobserwowano, że zmiana temperatury oraz operatora generacji powodują znaczny wzrost oscylacji wartości przystosowania w kolejnych iteracjach algorytmu. Dobór odpowiedniej wartości T_0 oraz operatora generacji nowego rozwiązania podyktowane są zatem kompromisem pomiędzy chęcią eksploracji nowych, odległych rozwiązań a także dokładną penetracją obszaru otaczającego dobre rozwiążanie. Z jednej strony algorytm może działać chaotycznie a z drugiej strony istnieje ryzyko utknięcia w minimum lokalnym.