Karol Wojtachnia 243702

Projektowanie efektywnych algorytmów

Sprawozdanie z etapu 2:

Algorytmy poszukiwania z zakazami oraz symulowanego wyżarzania dla problemu komiwojażera

Prowadzący:
dr inż. Jarosław Rudy

Cel zadania

Zadanie polegało na implementacji zadanych algorytmów oraz zbadanie jakości dostarczonych rozwiązań i pokazanie wpływu najważniejszych parametrów na szybkość ich działania.

Wstęp teoretyczny

Problem komiwojażera (ang. travelling salesman problem – TSP) polega na znalezieniu w grafie ważonym najkrótszej ścieżki łączącej wszystkie wierzchołki. Przez każdy wierzchołek można przejść tylko raz.

Badana w tym etapie algorytmy należą do grupy algorytmów metaheurystycznych. Oznacza to, że znajdują rozwiązanie problemu przy rozsądnych (akceptowalnych z punktu widzenia celu) nakładach obliczeniowych. Nie dają gwarancji optymalności rozwiązania ani jakości znalezionego rozwiązania.

Algorytm poszukiwania z zakazami (ang. tabu search – TS) można określić jako metodę dynamicznej zmiany sąsiedztwa danego rozwiązania. Opiera się na iteracyjnym przeszukiwaniu przestrzeni rozwiązań, wykorzystywaniu zmiennego sąsiedztwa i zapamiętywaniu niektórych ruchów (transformacji rozwiązań) w celu unikania minimów lokalnych. W algorytmie wykorzystywana jest lista ruchów zakazanych (ang. tabu list). Zapisywane są w niej, na określoną liczbę iteracji algorytmu (kadencja), wcześniej wykonane transformacje rozwiązania. Algorytm po wyznaczeniu rozwiązania początkowego, na przykład losowo, przypisuje je jako najlepsze znalezione. Następnie w sąsiedztwie tego rozwiązania wyszukiwane jest rozwiązanie o najlepszej wartości funkcji oceny ruchu, które nie wymaga wykonania ruchu znajdującego się w liście tabu. Stare rozwiązanie jest zastępowane przez nowo znalezione, a ruch, który utworzył nowe rozwiązanie jest dodawany do listy ruchów zakazanych. Sprawdza się, czy nowe rozwiązanie jest lepsze od najlepszego dotychczas znalezionego i w pozytywnym przypadku zapamiętuje się je. Na koniec należy zmniejszyć kadencję każdego elementu w liście tabu i usunąć z niej te, których kadencja się skończyła (jest równa 0). Algorytm jest wykonywany iteracyjnie aż do osiągnięcia warunku zakończenia, którym może być wykonanie określonej liczby iteracji lub przekroczenie zadanego czasu wykonywania.

Algorytm symulowanego wyżarzania bazuje na algorytmie Metropolisa, który został skonstruowany na podstawie zachowania układów termodynamicznych.
Głównym parametrem decydującym o działaniu algorytmu jest temperatura. W trakcie przebiegu algorytmu jej wartość się obniża. Na podstawie temperatury

wyliczane jest prawdopodobieństwo przyjęcia niepoprawnego rozwiązania ze wzoru:

$$\exp\left(-\frac{|f(x)-f(y)|}{T}\right)$$

gdzie:

f(x) jest wartością aktualnie przyjętego rozwiązania
f(y) jest wartością nowo utworzonego rozwiązania
T jest aktualną wartością temperatury.

Gdy T dąży do 0, co w algorytmie jest symulowane poprzez jej ciągłe obniżanie, wartość tego prawdopodobieństwa dąży do 0. Dzięki temu niepoprawne rozwiązania są akceptowane coraz rzadziej. Mechanizm ten pozwala na wychodzenie z ekstremów lokalnych i bardziej optymalne przeszukanie przestrzeni rozwiązań. W algorytmie ważną rolę odgrywa epoka. Jej długość wskazuje, jak wiele iteracji algorytmu zostanie wykonanych w takiej samej temperaturze.

Algorytm zaczyna się od wyznaczenia temperatury i długości epoki oraz wylosowania początkowego rozwiązania. Wartość wylosowanego rozwiązania przypisuje się jako najlepszą znalezioną. Następnie do warunku zatrzymania wykonuje się kolejne kroki algorytmu. Tyle razy, ile wynosi długość epoki, wyszukuje się nowe rozwiązanie w sąsiedztwie aktualnego rozwiązania. Za każdym wyszukaniem, jeżeli wartość nowego rozwiązania jest lepsza niż aktualnego zapamiętuje się nowe rozwiązanie jako aktualne. W przeciwnym wypadku losuje się liczbę z przedziału (0,1) i liczy się wartość prawdopodobieństwa opisanego powyżej. Jeżeli wylosowana liczba jest mniejsza niż prawdopodobieństwo, nowe rozwiązanie jest zapamiętane zamiast aktualnego. Na koniec epoki wykonuje się zmniejszenie temperatury według przyjętego schematu chłodzenia (w projekcie został zastosowany schemat geometryczny). Algorytm musi też pamiętać i aktualizować najlepszą znalezioną wartość rozwiązania.

Szczegóły implementacji

a)Algorytm poszukiwania z zakazami

Warunek zatrzymania został przyjęty jako przekroczenie czasu wykonywania algorytmu. Listę zakazów zdefiniowano jako macierz trójkątną N x N bez głównej przekątnej, której element o indeksach [i][j] oznacza zamianę i-tego oraz j-tego elementu ścieżki miejscami, a jego wartość oznacza aktualną kadencję tej transformacji. W implementacji występuje sąsiedztwo typu *invert.* Zaimplementowany w algorytmie przegląd sąsiedztwa sprawdza tylko połowę

macierzy N x N, gdzie N jest wielkością instancji TSP, gdyż zamiana i-tego wierzchołka z j-tym jest równa zamianie j-tego z i-tym. Algorytm zawiera także mechanizm chroniący przed stagnacją. Jeżeli w danej iteracji algorytmu żadna zamiana wierzchołków nie daje rozwiązania lepszego niż aktualne, aktualne rozwiązanie zastępowane jest przez wygenerowaną losowo ścieżkę. Implementacja zawiera także kryterium aspiracji. Jeżeli dana transformacja ścieżki ma pozwolić na uzyskanie rozwiązania lepszego niż dotychczasowe najlepsze, to się ją wykonuje nawet jeżeli jest ona obecna w liście tabu. Kadencja w algorytmie ma stałą wartość zdefiniowaną na początku algorytmu.

Zaimplementowany algorytm korzysta z następujących zmiennych:

- time – klasa licząca czas i potrafiąca zwrócić czas wykonania algorytmu
- maxTime – wartość czasu w ms po której algorytm się zatrzyma
- tabu – lista zakazów
- bestPath – wektor przechowujący kolejne numery wierzchołków najlepszej znalezionej ścieżki
- bestPathValue – wartość ścieżki bestPath
- currentPath – wektor przechowujący kolejne numery wierzchołków aktualnej ścieżki
- currentPathValue – wartość ścieżki currentPath
- newPath – wektor przechowujący kolejne numery wierzchołków ścieżki po ostatniej transformacji
- newPathValue – wartość ścieżki newPath
- stagnation – wartość 0 oznacza brak stagnacji, wartość 1 oznacza wystąpienie stagnacji
- size – wielkość instancji
- bestSwap1, bestSwap2 – numery wierzchołków, których zamiana daje najlepszą poprawę rozwiązania w sąsiedztwie aktualnego rozwiązania
- cadency – długość kadencji

Pseudokod funkcji tabuSearch()

K0) zdefiniuj kadencję
K1) wygeneruj losową ścieżkę i przypisz ją do currentPath
K2) oblicz wartość currentPath i przypisz do currentPathValue
K3) bestPath = currentPath oraz bestPathValue = currentPathValue
K4) while (time.giveTime() < maxTime) powtarzaj
K5) if (stagnacja = 1)
K6) wykonaj K1 oraz K2
K7) if (currentPathValue < bestPathValue)

K8) wykonaj K3
K9) stagnacja = 0
K10) bestSwap1=0 i bestSwap2=0
K11) for (int i=0;i<size;i++)
K12) for(int j=i+1; j<size;j++)
K13) zamien i-ty i j-ty element currentPath i przypisz do newPath
K14) policz wartość newPath i przypisz do newPathValue
K15) if(newPathValue < bestPathVale)
K16) bestPath = newPath oraz bestPathValue=newPathValue
K17) bestSwap1=i oraz bestSwap2=j
K18) else if(newPathValue < currentPathValue && tabu[i][j-i-1] ==0)
K19) bestSwap1=i oraz bestSwap2=j
K20) currentPathValue = newPathValue
K21) if((bestSwap1 | bestSwap2) != 0)
K22) zamień w currentPath elementy o indeksach bestSwap1, bestSwap2
K23) tabu[bestSwap1][bestSwap2 – bestSwap1 -1] =cadence
K24) stagnation = 0
K25) else stagnation++
K26) zredukuj kadencję każdego elementu listy tabu o 1, jeżeli jest różna 0
K27) wyświetl bestPath i bestPathValue

b)Algorytm symulowanego wyżarzania

Warunkiem zatrzymania jest osiągnięcie przez algorytm minimalnej temperatury zadeklarowanej przed rozpoczęciem algorytmu. Zaimplementowany schemat chłodzenia jest schematem geometrycznym. Program używa sąsiedztwa typu *invert* – nowa ścieżka jest generowana przez zamianę dwóch losowych wierzchołków w aktualnej ścieżce. W kodzie zaimplementowano także mechanizm chroniący przed stagnacją. Jeżeli N zamian wierzchołków z kolei nie spowoduje znalezienia krótszej ścieżki niż aktualna, program losuje nową, całkowicie losową ścieżkę i przypisuje ją jako aktualną.

Zaimplementowany algorytm korzysta z następujących zmiennych:

- eraLength – długość jednej epoki
- temperaturę – aktualna temperatura
- minmalTemp – minimalna temperatura osiągana przez algorytm
- bestPath – wektor przechowujący kolejne numery wierzchołków najlepszej znalezionej ścieżki
- bestPathValue – wartość ścieżki bestPath

- currentPath – wektor przechowujący kolejne numery wierzchołków aktualnej ścieżki
- currentPathValue – wartość ścieżki currentPath
- newPath – wektor przechowujący kolejne numery wierzchołków ścieżki po ostatniej transformacji
- newPathValue – wartość ścieżki newPath
- stagnation – zmienna przechowująca ilość zamian wierzchołków bez poprawy aktualnej ścieżki
- stagnationMax – maksymalna wartość stagnacji

Pseudokod funkcji tabuSearch()

K0) zdefiniuj długość epoki
K1) zdefiniuj temperaturę początkową i minimalna
K2) zdefiniuj maksymalną wartość stagnacji
K3) wygeneruj losową ścieżkę i przypisz ją do currentPath
K4) oblicz wartość currentPath i przypisz do currentPathValue
K5) bestPath = currentPath oraz bestPathValue = currentPathValue
K6) while (temperaturę > minimalTemp) **{**
K7) for(int i=0;i<eraLength;i++) **{**
K8) if(stagnation > stagnationMax)
K9) wylosuj nową ścieżkę i przypisz do currentPath
K10) policz wartość currentPath
K11) if(currentPathValue < bestPathvalue)
K12) bestPath=currentPath i bestPathValue=currentPathValue
K13) stagnation = 0
K14) wylosuj indeksy dwóch wierzchołków ze ścieżki
K15) zamień miejscami wylosowane wierzchołki i przypisz nową ścieżkę do newPath, a jej długość do newPathValue
K16) if(newPathvalue < currentPathValue)
K17) currentPathValue = newPathValue i currentPath = newPath
K18) if(currentPathValue < bestPathValue)
K19) bestPath=currentPath i bestPathValue=currentPathValue
K20) elseif (rand(0,1) < exp(-(newPathValue-currentPathValue)/temperature)
K21) currentPathValue = newPathValue i currentPath = newPath
K22) stagnation=0
K23) else stagnation++ **}**
K24) iteration++
K25) wykonaj chłodzenie**}**
K26) zwróć bestPath

Opis badania algorytmów

Aby sprawdzić działanie algorytmów badanie zostało podzielone na etapy:

1. porównanie średniego błędu dla różnych czasów działania algorytmu tabu search
2. porównanie średniego błędu dla różnej długości kadencji w algorytmie tabu search
3. porównanie średniego błędu dla różnej temperatury początkowej algorytmu SA
4. porównanie dla średniego błędu różnej wielkości epoki algorytmu SA

Do badania algorytmów użyto instancji:

- m16.atsp – 16 wierzchołków
- gr21.tsp – 21 wierzchołki
- gr48.tsp – 48 wierzchołków
- ftv70.atsp – 70 wierzchołków
- gr120.tsp – 120 wierzchołków
- kroB150.tsp – 150 wierzchołków
- gr229.tsp – 229 wierzchołków

Wyniki badania algorytmów zostały uśrednione dla 10 pomiarów.

Wyniki badania zaimplementowanych algorytmów

1) Porównanie średniego błędu dla różnych czasów działania algorytmu tabu search

Kadencja była równa wielkości danej instancji.

		WIELKOSC INSTANCJI						
		16	21	48	70	120	150	229
CZAS WYKONANIA	1 min	0,00%	0,00%	16,47%	50,30%	53,26%	82,47%	92,01%
	2 min	0,00%	0,00%	15,94%	43,54%	53,63%	78,01%	94,94%
	3 min	0,00%	0,00%	13,67%	41,90%	52,87%	88,33%	90,50%
	4 min	0,00%	0,00%	8,03%	44,26%	53,52%	78,79%	87,90%

Wyniki nie pokazują, aby czasu działania algorytmu dawało jednoznaczne polepszenie działania algorytmu. Jedynie dla instancji gr48.tsp 4 krotne wydłużenie czasu dało 2 krotne polepszenie średniego wyniku. W pozostałych przypadkach zmiany w średnim błędzie są zbyt niewielkie, aby można było wyciągnąć jednoznaczne wnioski.

2) Porównanie średniego błędu dla różnej długości kadencji w algorytmie tabu search

Czas wykonania algorytmu został ustawiony na 1 min.

		WIELKOSC INSTANCJI						
		16	21	48	70	120	150	229
DŁUGOŚĆ KADENCJI	size	0,00%	0,00%	15,10%	44,05%	52,10%	86,01%	88,90%
	2*size	0,00%	0,00%	12,82%	64,31%	57,61%	77,06%	94,94%
	5*size	0,00%	0,00%	10,92%	45,69%	54,71%	87,31%	90,50%
	10*size	1,48%	5,21%	11,55%	47,49%	55,07%	89,14%	92,13%

W przypadku gr21.tsp oraz m16.atsp zbyt duża wartość kadencji sprawiła, że optymalne rozwiązanie zostało pominięte. Pozostałe wyniki pozwalają jedynie stwierdzić, że w zaimplementowanym algorytmie długość kadencji nie ma zbyt wielkiego znaczenia, gdyż wyniki dla każdej instancji są do siebie bardzo zbliżone.

3) Porównanie średniego błędu dla różnej temperatury początkowej algorytmu symulowanego wyżarzania.

Temperatura minimalna wynosiła 0,0001.
Chłodzenie polegało na pomnożeniu aktualnej temperatury przez 0,9999.
Stagnacja maksymalna została ustawiona na 100*wielkość isntancji.

		WIELKOSC INSTANCJI						
		16	21	48	70	120	150	229
TEMP POCZĄTKOWA	1	0,00%	5,21%	23,78%	51,33%	89,95%	74,71%	84,26%
	10	0,00%	0,00%	12,33%	17,74%	26,25%	66,23%	72,89%
	50	0,00%	0,00%	0,00%	10,26%	13,73%	24,50%	74,00%
	100	0,00%	0,00%	0,00%	11,59%	9,51%	16,85%	54,56%

Wyniki pomiarów pokazują, że wraz ze wzrostem temperatury początkowej maleje średni błąd. Dla każdej wielkości instancji otrzymywane wyniki były coraz lepsze, ich poprawa znacząca.

4) Porównanie dla średniego błędu różnej wielkości epoki algorytmu SA

Temperatura początkowa wynosi 100.
Temperatura minimalna wynosiła 0,0001.
Chłodzenie polegało na pomnożeniu aktualnej temperatury przez 0,9999.

		WIELKOSC INSTANCJI						
		16	21	48	70	120	150	229
DŁUGOŚC EPOKI	size/4	0,00%	0,00%	1,21%	14,41%	12,06%	23,96%	36,79%
	size/3	0,00%	0,00%	1,88%	13,80%	13,96%	20,98%	55,24%
	size/2	0,00%	0,00%	0,75%	14,56%	12,07%	23,54%	51,10%
	size	0,00%	0,00%	1,61%	10,31%	4,86%	12,98%	38,97%

Uzyskane wyniki mogą sugerować, że im dłuższa epoka, tym lepsze wyniki można otrzymać. Oprócz dwóch największych instancji wyniki są do siebie bardzo zbliżone i nie pozwalają wyprowadzić jednoznacznych wniosków.

Wnioski

Podczas realizacji projektu udało się zaimplementować działające algorytmy zarówno dla SA, jak i TS. Przeprowadzone badanie obu implementacji pozwala stwierdzić, że implementacja symulowanego wyżarzania daje znacznie lepsze wyniki, niż tabu search.
Dla poszukiwania z zakazami badane parametry nie pokazały istotnego wpływu na jakość otrzymywanych rozwiązań. Zaimplementowany algorytm TS bardzo słabo radzi sobie z opuszczaniem ekstremum lokalnego.
W przypadku symulowanego wyżarzania wyniki pokazują, że im większa jest temperatura początkowa, tym lepszej jakości są zwracane rozwiązania. Dla wyższych temperatur algorytm częściej przyjmuje ścieżki, które nie poprawiają wyniku, co pozwala na efektywniejsze przeszukanie przestrzeni rozwiązań.
Badane algorytmy bazują na losowaniu rozwiązań, SA także na losowaniu wierzchołków do zamiany. Aby wyeliminować wpływ tej losowości na otrzymywane wyniki należałoby przeprowadzić dłuższe badania na większej ilości instancji, a każdą z instancji przebadać dokładniej. Uśrednienie większej ilości pomiarów pozwoliłoby na jeszcze lepszą stabilizację wyników.