Karol Wojtachnia 243702

Projektowanie efektywnych algorytmów

Sprawozdanie z etapu 2:

Algorytm genetyczny

Prowadzący:
dr inż. Jarosław Rudy

Cel zadania

Zadanie polegało na implementacji zadanego algorytmu oraz zbadanie jakości dostarczonych rozwiązań i pokazaniu wpływu najważniejszych parametrów na jego działanie.

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.

Badany w tym etapie algorytm należy 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 genetyczny jest rodzajem heurystycznego przeszukiwania przestrzeni rozwiązań, który naśladuje naturalne procesy ewolucji. Zaliczany jest do algorytmów ewolucyjnych. W metodach ewolucyjnych przeszukiwanie polega na tworzeniu rozwiązania opartego na współpracy osobników (populacji). Osobniki wykorzystują wiedze o przestrzeni rozwiązań i przekazują ją kolejnym pokoleniom wykorzystując do tego mechanizmy ewolucyjne. Kluczem są niewielkie zmiany przystosowania osobników w całej populacji, dążących do możliwie najlepszego rozwiązania spełniającego warunki narzucane przez środowisko.

W implementacji środowisko reprezentowane jest przez populację. Populacja ograniczona jest przez maksymalną ilość osobników, które mogą w niej występować. Każdy z tych osobników jest rozwiązaniem problemu, co w przypadku TSP oznacza ścieżkę odpowiadającą cyklowi Hamiltona dla rozważanego grafu.

W algorytmach genetycznych wykorzystywane są 3 operatory genetyczne:

1. SELEKCJA – wybór z bieżącej populacji osobników, których materiał zostanie poddany operacji krzyżowania oraz mutacji, a następnie przekazany do kolejnej populacji.

2. KRZYŻOWANIE – wymiana materiału genetycznego pomiędzy parami osobników wybranymi podczas selekcji. W wyniku tej operacji powstają nowe osobniki mogące wejść w skład nowej populacji. W efekcie krzyżowania powinny powstać osobniki lepiej przystosowane od swoich rodziców. Zawsze zachodzi z pewnym prawdopodobieństwem.

3. MUTACJA – zmiana wartości losowo wybranej cechy osobnika. Zapewnia zmienność chromosomów oraz stwarza możliwość wyjścia z optimów lokalnych. Tak jak krzyżowanie zawsze zachodzi z pewnym prawdopodobieństwem.

Szczegóły implementacji

I. W implementacji została zastosowana reprezentacja ścieżkowa – każdy osobnik opisany jest jako wektor liczb całkowitych oznaczający kolejne numery wierzchołków odwiedzanych w reprezentowanym przez niego rozwiązaniu.
II. Przystosowanie osobników obliczane jest jako odwrotność długości cyklu Hamiltona opisanego przez danego osobnika. W ten sposób wyższa wartość funkcji przystosowania oznacza krótszą ścieżkę.
III. Populacja początkowa wypełniana jest osobnikami, dla których zostały wygenerowane losowe ścieżki.
IV. W sposobie generowania nowego pokolenia zastosowany został elitaryzm. 10% osobników o najwyższej wartości funkcji przystosowania jest bezpośrednio kopiowana do nowej populacji. Reszta populacji wypełniana jest w następujący sposób:

K01) powtarzaj tak długo, jak populacja nie jest pełna:
K02) wylosuj osobnika ze starej populacji (elita nie została z niej usunięta) i przypisz go do zmiennej *newIndividual*
K03) wylosuj liczbę z przedziału [0,1]
K04) jeżeli jest mniejsza niż prawdopodobieństwo krzyżowania:
K05) wylosuj drugiego osobnika
K06) przeprowadź krzyżowanie z *newIndividual*
K06) *newIndividual* = dziecko lepiej przystosowane
K07) wylosuj liczbę z przedziału [0,1]
K08) jeżeli jest mniejsza niż prawdopodobieństwo mutacji
K09) przeprowadź mutację na *newIndividual*
K10) dodaj *newIndividual* do nowej populacji

V. Zastosowaną metoda jest OX – krzyżowanie z zachowaniem porządku. Dla rodziców r1 oraz r2 przebiega ona następująco:

Osobniki rodzicielskie

$$p = (\ p_1 \dots \overbrace{|p_{k_1} \dots p_{k_2}|}^{sekcja\ dopasowania} \dots\ p_m\ )$$

$$q = (\ q_1 \dots \underbrace{|q_{k_1} \dots q_{k_2}|}_{sekcja\ dopasowania} \dots\ q_m\ )$$

Osobniki potomne

$$r = (\ r_1 \dots \overbrace{|p_{k_1} \dots p_{k_2}|}^{sekcja\ dopasowania} \dots\ r_m\ )$$

$$s = (\ s_1 \dots \underbrace{|q_{k_1} \dots q_{k_2}|}_{sekcja\ dopasowania} \dots\ s_m\ )$$

1 Przykłady zaczerpnięte z wykładów dr Tomasza Kapłona

Indeksy k1 oraz k2 są wybierane losowo z przedziału od 1 do m. Do r kopiowane są elementy z q, które nie występują w sekcji dopasowania p, zaczynając od indeksu k2+1. Analogicznie postępujemy z s. Do s kopiowane są elementy z p, które nie występują w sekcji dopasowania q, zaczynając od indeksu k2+1.

VI. Zastosowaną metodą mutacji jest inwersja. Dla wylosowanych indeksów k1 oraz k2 odwracana jest kolejność miast zawierających się między tymi indeksami. Na przykład :
Dla indeksów 3 i 7 oraz osobnika [1,9,2,**8**,3,7,4,**5**,0,6] otrzymujemy osobnika [1,9,2,**5**,4,7,3,**8**,0,6].

VII. Warunkiem zatrzymania w algorytmie jest osiągnięcie maksymalnego czasu wykonywania.

Metoda genethicAlgorithm() korzysta z następujących zmiennych:

- iteration – przechowuje informację o aktualnym numerze iteracji
- maxTime – maksymalny czas wykonywania w milisekundach
- mutationProbability – prawdopodobieństwo mutacji jako liczba z przedziału [0,1]
- crossoverProbability – prawdopodobieństwo krzyżowania jako liczba z przedziału [0,1]
- populationSize – wielkośc populacji
- population – wektor wektorów przechowujący wszystkich osobników
- populationFitnessValues – wektor przechowujący wartości funkcji dopasowania dla każdego osobnika z populacji
- bestPath – wektor przechowujący najlepsze znalezione rozwiązanie
- bestPathValue – wartość najlepszego znalezionego rozwiązania
- countTime – klasa pozwalająca na liczenie czasu

Pseudokod metody genethicAlgorithm():

```
K01) for (int i = 0; i < populationSize; i++)
K02)        wygeneruj losowego osobnika
K03)        oblicz długość ścieżki dla tego osobnika
K04)        if (długość nowej ścieżki < bestPathValue) zapisz to rozwiązanie jako
            najlepsze w bestPath i jego wartość w bestPathValue
K05)        dodaj nowego osobnika do population, a wartość jego
            przystosowania do populationFitnessValues
K06) while (czas wykonania < maxTime)
K07)        wygeneruj nową populację zgodnie z algorytmem w punkcie IV
            szczegółów implementacji na stronie 4
K08)        stwórz wektor newFitness dla nowych wartości przystosowania
K09)        for (int i = 0; i < populationSize; i++)
K10)              int pathLength = policz wartość ścieżki dla population[i]
K11)              if (pathLength < bestPathValue) bestPathValue = pathLength
                  oraz bestPath = population[i]
K12)              dodaj pathLength do newFiness[i]
K13)        populationFitnessValues = newFitnes
```

K14) iteration++
K15) zwróć bestPath oraz bestPathValue

Opis badania algorytmu

Aby sprawdzić działanie algorytmu badanie zostało podzielone na etapy:

1. porównanie średniego błędu dla różnych wartości prawdopodobieństwa krzyżowania
2. porównanie średniego błędu dla różnych wielkości populacji
3. porównanie średniego błędu dla różnych wartości prawdopodobieństwa mutacji

Do badania użyto następujących instancji:

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

Wyniki badania algorytmu zostały uśrednione dla 10 pomiarów.

Wyniki badania zaimplementowanych algorytmów

1) Porównanie średniego błędu dla różnych wartości prawdopodobieństwa krzyżowania

Prawdopodobieństwo mutacji zostało ustawione na 1%, a wielkość populacji na 5*rozmiar instancji. Maksymalny czas wykonania to minuta.

		WIELKOŚĆ INSTANCJI				
		21	48	70	120	229
PRAWDOPODOBIEŃSTWO KRZYŻOWANIA	10%	3,95%	4,46%	88,82%	22,62%	151,08%
	30%	4,02%	3,11%	80,74%	24,35%	267,56%
	50%	3,55%	5,46%	60,04%	88,23%	452,91%
	70%	0%	0%	128%	87,10%	441,67%
	85%	0%	0%	213,74%	234,26%	598,32%

Dla mniejszych instancji badanie dało oczekiwane rezultaty. Dla prawdopodobieństwa krzyżowania 70% i 85% algorytm był dokładny i zwracał rozwiązania optymalne. Jednak dla instancji większych błędy rosły wraz ze

wzrostem badanego prawdopodobieństwa. Prawdopodobną przyczyną takiej sytuacji jest zbyt duża złożoność obliczeniowa zaimplementowanego operatora krzyżowania.

2) Porównanie średniego błędu dla różnych wielkości populacji.

Prawdopodobieństwo mutacji zostało ustawione na 1%, a prawdopodobieństwo krzyżowania na 85%. Maksymalny czas wykonania to minuta.

		WIELKOŚĆ INSTANCJI				
		21	48	70	120	229
WIELKOŚĆ POPULACJI	50	0,00%	28,32%	190,36%	265,11%	711,11%
	100	0,00%	20,13%	165,88%	227,54%	645,94%
	150	0,00%	15,62%	212,36%	237,05%	630,15%
	200	0,00%	15,93%	194%	208,49%	626,11%

Oprócz fvt70.atsp wszystkie instancje zachowały się zgodnie z oczekiwaniami. Wzrost wielkości populacji skutkował poprawieniem otrzymywanych wyników.

3) Porównanie średniego błędu dla różnych wartości prawdopodobieństwa mutacji

Prawdopodobieństwo krzyżowania ustawiono na 85%.
Wielkość populacji to 200 osobników.

		WIELKOŚĆ INSTANCJI				
		21	48	70	120	229
PRAWDOPODOBIEŃSTWO MUTACJI	1%	0,00%	14,45%	190,24%	212,45%	638,04%
	5%	0,00%	6,04%	125,59%	422,45%	427,80%
	10%	0,00%	4,64%	149,23%	66,98%	347,42%
	20%	0,00%	3,71%	133,10%	43,97%	313,85%
	50%	0,00%	4,62%	142,92%	21,52%	267,25%

Osiągane wartości nie są zgodne z oczekiwanymi, które zakładają, że algorytm genetyczny działa najlepiej dla niskich wartości prawdopodobieństwa mutacji.

Wnioski

Algorytm został zaimplementowany zgodnie z założeniami algorytmów genetycznych. Mimo to, badania przeprowadzone na algorytmie pokazują, że nie zachowuje się on tak, jak powinien.

Zgodnie z teorią prawdopodobieństwo krzyżowania w algorytmie genetycznym powinno być wysokie, wynosić około 85%. Tymczasem badany algorytm daje znacznie lepsze wyniki dla niskich wartości tego parametru. Przyczyną może być implementacja operatora krzyżowania. Kiedy rośnie prawdopodobieństwo krzyżowania, rośnie ilość wywołań tego operatora. Widać to szczególnie dla bardzo dużych instancji. W wyniku tego algorytm w czasie jednej minuty wykonuje mniej iteracji, niż dla niższych wartości prawdopodobieństwa krzyżowania. Dla dokładniejszego zbadania tej tezy należałoby zmienić kryterium zatrzymania na osiągnięcie maksymalnej liczby iteracji.

Podobne wnioski można wyciągnąć z badania prawdopodobieństwa mutacji. Algorytm genetyczny powinien osiągać wartości optymalne dla niskich wartości tego parametru – rzędu 1%. Stworzony algorytm działa tym lepiej, im wyższe jest prawdopodobieństwo zajścia mutacji.

Dla dokładniejszego zbadania tej implementacji należałoby przetestować algorytm ze zmienionym sposobem tworzenia nowej populacji, aby wykluczyć wpływ tej części implementacji na efektywność działania.