**POLITECHNIKA WROCŁAWSKA**

**Zespół Inżynierii Oprogramowania i Inteligencji Obliczeniowej**

**Katedra Informatyki Technicznej**

**Projektowanie efektywnych algorytmów** **- projekt**

**Kurs: INEK032**

**Sprawozdanie z projektu**

|  |  |
| --- | --- |
| **Wykonał:** | **Michał Madarasz, 238903** |
| **Termin:** | **Czwartek 17:05** |
| **Prowadzący:** | **Mgr inż. Antoni Sterna** |
| **Data oddania sprawozdania:** | **28.01.2019** |
| **Ocena:** |  |

|  |
| --- |
| **Uwagi prowadzącego:** |

# Wstęp

Celem projektu jest zastosowanie 2 algorytmów dla wcześniej wybranego zagadnienia, w tym przypadku problemu komiwojażera a następnie porównanie wyników otrzymanych przy użyciu każdego z nich. W trzecim zadaniu projektowym zbadane zostało działanie algorytmu genetycznego na podstawie napisanego algorytmu, badając go w zależności od najbardziej istotnych parametrów dla wcześniej wybranych instancji testowych. Uzyskane wyniki zostaną potem porównane ze sobą pod względem dokładności otrzymanego wyniku oraz porównane z metodą z poprzedniego etapu projektu, czyli Tabu Search.

# Algorytm genetyczny

**Algorytm genetyczny** jest heurystyką przeszukującą przestrzeń alternatywnych rozwiązań bazującą na zjawisku ewolucji biologicznej. Zadaniem algorytmu jest symulowanie populacji danego rozwiązania dążącego do jak największego przystosowania się do otoczenia, czyli do uzyskania jak najlepszego rozwiązania badanego problemu. Wykorzystując naturalne mechanizmy takie jak rozmnażanie czy mutację osobników staramy się przystosować kolejne pokolenia rozwiązań by coraz bardziej zbliżać się do optymalnego rozwiązania (jednak nie mamy żadnej gwarancji na jego odnalezienie).

Podstawą algorytmu jest **populacja**, czyli grupa osobników danego problemu. To na jej podstawie przebiega dalsze dostosowywanie się osobników do otoczenia, gdyż jest ona bazą dla wykonywanego rozmnażania. Początkowa populacja generowana jest zazwyczaj losowo. Kiedy istnieje już populacja, następuje właściwa część algorytmu zaczynając od **selekcji**. Polega ona na wybraniu jak najbardziej optymalnych osobników do późniejszego rozmnożenia. Zazwyczaj są to osobniki najlepiej przystosowane, jednak nie jest to regułą.

Po przeprowadzeniu selekcji wybrane osobniki przystępują do **rozmnażania**, podczas którego „krzyżują się” wyselekcjonowane wcześniej osobniki. Zazwyczaj brane do krzyżowania są dwa osobniki, z których powstają dwa kolejne (dzieci). Dzięki temu otrzymujemy osobniki nowe, będące przemieszaniem się rozwiązań rodziców, co po wielokrotnym wykonaniu daje nam 2 nową pulę osobników z innym rozwiązaniem. Dodatkowym elementem mogącym zwiększyć różnorodność otrzymanych osobników jest **mutacja**. Pojawia się ona rzadko i zazwyczaj nie wprowadza wielkich zmian, jednak może pomóc w poprawieniu rozwiązania, gdy krzyżowane osobniki nie mogą poprawić rozwiązania przez dłuższy czas. Jako że mutacja przebiega losowo, może ona zarówno poprawić, jak i pogorszyć przystosowanie osobnika. Ostatnim elementem algorytmu jest wybór **nowej populacji**, będącej następną generacją algorytmu. Polega ona na stworzeniu nowej grupy osobników, z do której należy część poprzedniej populacji oraz nowo wygenerowane dzieci skrzyżowanych osobników.

Po wybraniu nowej populacji powtarzana jest cała sekwencja dla nowych osobników, w celu utworzenia kolejnej generacji rozwiązań. Całość algorytmu powtarza się aż do spełnienia warunków zakończenia, wybierając jako wynik końcowy najlepiej przystosowanego osobnika, czyli rozwiązanie z najlepszym uzyskanym wynikiem.

# Implementacja algorytmu

Napisany algorytm został napisany na podstawie standardowego schematu Algorytmu genetycznego:

***1.*** *wybór populacji początkowej chromosomów (losowy)*

***2.*** *sprawdzanie warunku zatrzymania*

***a.*** *krzyżowanie chromosomów z populacji rodzicielskiej*

***b.*** *mutacja - może być również wykonana przed krzyżowaniem*

***d.*** *ocena przystosowania chromosomów*

***e.*** *utworzenie nowej populacji*

***3.*** *wyprowadzenie ,,najlepszego’’ rozwiązania*

Algorytm po wczytaniu danych z plików testowych generuje wektor **nowej populacji**. Populacja wygenerowana zostaje w ilości określonej przez użytkownika całkowicie losowo (tworząc drogę z losowo wybranych miast), po czym obliczana jest droga każdego wygenerowanego rozwiązania. Na końcu populacja jest sortowana pod kątem długości drogi, czyli najlepszego wyniku wylosowanej drogi. Każda droga zaczyna się od wierzchołka startowego o indeksie 0.

Po utworzeniu pierwszej populacji, następuje pętla działając aż nie zostanie spełniony warunek końcowy działania algorytmu. W programie zastosowane zostało **ograniczenie czasowe** ustalane w programie. Wewnątrz tej pętli znajduje się główna część algorytmu.

Po utworzeniu populacji początkowej następuje losowanie par osobników do krzyżowania. Pary są losowane przy użyciu odpowiednio zmodyfikowanej metody rand() oraz współczynnika krzyżowania, który w większości przypadków wynosi 0,8.

Po wylosowaniu dwóch osobników, losowane jest na podstawie podanego w konfiguracji prawdopodobieństwa, czy mają zostać skrzyżowane czy też nie. Jeśli odpowiednia wartość zostanie wylosowana, osobniki przechodzą do **krzyżowania metodą OX**. Metoda ta losuje dwa indeksy, pomiędzy którymi nastąpi krzyżowanie się osobników zgodnie z tą metodą. Po wykonaniu krzyżowania zostaje obliczona droga dla dzieci i przekazywane są one do następnego etapu, którym jest mutacja.

Mutacja podobnie jak samo krzyżowanie wykonywana jest z prawdopodobieństwem określonym przez użytkownika w programie. Gdy zostanie wylosowana dla każdego skrzyżowanego dziecka osobno, następuje jego **mutacja.** Rodzaj mutacji może zostać wybrany przez użytkownika w programie. Do wyboru są dwie metody mutacji: **wstawienie (insert)** oraz **odwrócenie podciągu (invert).** Po mutacji zostaje obliczona nowa wartość drogi dla osobnika.

Po krzyżowaniu oraz po mutacji, bez względu na to czy zaszła czy też nie, utworzony osobnik zostaje dodany do nowej populacji. Po dodaniu wszystkich osobników następuje ich redukcja poprzez posortowanie i ustawienie najlepszych osobników na początku. Następnie wszystkie osobniki wykraczające poza ilość populacji zostają usunięte. Po tym procesie wynik zostaje zapisany do pliku oraz zostaje sprawdzony warunek zakończenia programu.

Czas działania algorytmu jest przede wszystkim uzależniony od czasu ograniczającego podanego w konfiguracji. Z kolei jego główna część wewnątrz pętli ograniczonej czasowo ze względu na wykonywaną metodę krzyżowania daje nam **złożoność obliczeniowąO(n2)**.

# Procedura testowania

Testy przygotowanych zestawów zostały wykonane na prywatnym komputerze. Podczas przeprowadzania testów nie były uruchomione żadne dodatkowe aplikacje. A wszystkie procesy w tle zostały ograniczone do minimum poprzez wyłączenie aplikacji w tle oraz odłączenie komputera od Internetu w celu uzyskania jak najlepszych wyników. Konfiguracja sprzętowa maszyny testującej wygląda następująco:

* Procesor: Intel Core i5-5200u 2.2 GHz;
* Pamięć RAM: 8,00 Gb;
* Typ systemu: 64-bitowy system operacyjny;
* System operacyjny: Windows 10 Home;

Jako danych wejściowych użyłem danych z plików dostępnych na stronie: <http://elib.zib.de/pub/mp-testdata/tsp/tsplib/atsp/index.html>. Testowane były trzy instancje problemu TSP: tsp\_43, tsp\_171, tsp\_358.

Każdy z plików posiada inny zestaw wierzchołków, w różnej liczbie oraz o różnej odległości względem siebie. Również do każdego z nich dołączona jest znaleziona optymalna ścieżka, co pozwoli nam porównać otrzymane wyniki podczas testów z najlepszymi znalezionymi do tej pory, dzięki czemu będziemy mogli określić czy dany algorytm działa prawidłowo. Pliki są wczytywane z formatu .txt. Aby wczytać plik testowy podaje się jedynie nazwę bez rozszerzenia.

Parametry do testowania algorytmu były ustawiane na stałe dla każdej instancji problemu takie same. Parametry wynosiły odpowiednio:

* **Czas działania**: 60 sekund;
* **Rozmiar populacji początkowej**: 50, 100, 150;
* **Prawdopodobieństwo krzyżowania**: 0.8;
* **Prawdopodobieństwo mutacji**: 0.01, 0.02, 0.05, 0.1;

Wyniki pomiarów przedstawiają średnią wartość uzyskaną dla badanej kombinacji parametrów. Aby dowiedzieć się jak bardzo wygenerowane rozwiązania różnią się od rozwiązania optymalnego, dla pewnych danych został obliczony błąd względem rozwiązania optymalnego na podstawie wzoru:

**|f – fopt| / fopt**

gdzie:

**f – wartość obliczona przez testowany algorytm**

**fopt – wartość optymalna – najlepsze znane rozwiązanie**

Wyniki optymalne dla poszczególnych instancji:

* tsp\_43 – 5620;
* tsp\_171 – 2755;
* tsp\_358 – 1163;

# Wynik

## Badanie wpływu wielkości populacji na wyniki dla trzech różnych wartości oraz obu wybranych metod mutacji: inwersji i wstawienia.

### Tsp\_43

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| tsp\_43 – 60 sekund, inwersja | | | | | |
| **Populacja – 50** | | **Populacja – 100** | | **Populacja 150** | |
| **Czas [s]** | **Błąd względny** | **Czas [s]** | **Błąd względny** | **Czas [s]** | **Błąd względny** |
| 0,208501 | 113% | 0,224744 | 205% | 0,36089 | 205% |
| 0,487285 | 111% | 0,493084 | 204% | 0,810784 | 203% |
| 0,591437 | 104% | 0,818613 | 118% | 1,19718 | 116% |
| 0,94396 | 104% | 1,1263 | 115% | 1,50028 | 111% |
| 1,04492 | 104% | 1,3084 | 113% | 1,85871 | 109% |
| 1,2958 | 104% | 1,51088 | 111% | 3,07445 | 17% |
| 1,45452 | 104% | 1,75435 | 110% | 4,20768 | 16% |
| 2,1095 | 103% | 1,97569 | 108% | 5,72245 | 12% |
| 2,25047 | 103% | 2,64146 | 18% | 6,12078 | 9% |
| 2,43021 | 103% | 2,84002 | 16% | 7,32083 | 8% |
| 2,67619 | 102% | 3,5368 | 13% | 9,69601 | 7% |
| 3,013 | 102% | 4,79934 | 12% | 13,3431 | 6% |
| 4,12134 | 101% | 5,22568 | 12% | 16,0869 | 5% |
| 5,44689 | 7% | 5,45098 | 10% | 19,7064 | 4% |
| 6,34858 | 6% | 6,25285 | 9% | 24,1325 | 3% |
| 7,90823 | 5% | 6,45756 | 8% | 31,0287 | 2% |
| 12,0495 | 4% | 7,88609 | 7% | 43,5503 | 1% |
| 16,2794 | 3% | 12,3976 | 6% |  |  |
| 20,1661 | 2% | 17,278 | 5% |  |  |
| 47,8093 | 1% | 22,5421 | 4% |  |  |
| 59,7091 | 1% | 34,9774 | 3% |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| tsp\_43 – 60 sekund, wstawienie | | | | | |
| **Populacja – 50** | | **Populacja – 100** | | **Populacja 150** | |
| **Czas [s]** | **Błąd względny** | **Czas [s]** | **Błąd względny** | **Czas [s]** | **Błąd względny** |
| 0,114161 | 204% | 0,220464 | 205% | 0,374468 | 114% |
| 0,447583 | 113% | 0,618377 | 110% | 1,65577 | 20% |
| 1,09628 | 106% | 1,10118 | 109% | 3,2314 | 19% |
| 1,59347 | 105% | 1,28847 | 18% | 3,53995 | 18% |
| 2,07597 | 104% | 1,47224 | 16% | 3,85449 | 14% |
| 2,22805 | 103% | 1,84562 | 15% | 4,4823 | 13% |
| 3,68668 | 102% | 2,04288 | 14% | 4,81668 | 12% |
| 4,16282 | 7% | 3,01188 | 12% | 5,17343 | 10% |
| 5,71627 | 6% | 3,56483 | 10% | 6,15788 | 9% |
| 7,26776 | 5% | 5,47665 | 9% | 7,75096 | 8% |
| 8,66271 | 4% | 6,98411 | 8% | 8,73621 | 7% |
| 11,1855 | 3% | 8,28939 | 7% | 10,9872 | 6% |
| 13,688 | 2% | 9,71545 | 6% | 13,5962 | 5% |
| 24,6452 | 1% | 14,6913 | 5% | 18,6187 | 4% |
| 59,6535 | 1% | 22,8163 | 4% | 25,6766 | 3% |
|  |  | 28,238 | 3% | 32,322 | 2% |
|  |  | 40,6344 | 2% | 53,1034 | 1% |
|  |  | 59,9365 | 1% | 60,1113 | 1% |

## Tsp\_171

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| tsp\_171 – 60 sekund, inwersja | | | | | |
| **Populacja – 50** | | **Populacja – 100** | | **Populacja 150** | |
| **Czas [s]** | **Błąd względny** | **Czas [s]** | **Błąd względny** | **Czas [s]** | **Błąd względny** |
| 0,90 | 787,80% | 2,32973 | 7,72922 | 3,35835 | 7,88494 |
| 1,93 | 783,16% | 6,9345 | 7,63412 | 10,0821 | 7,78802 |
| 2,96 | 774,59% | 9,00094 | 7,61343 | 13,3304 | 7,67913 |
| 3,84 | 768,78% | 11,1333 | 7,56878 | 16,6059 | 7,58004 |
| 8,19 | 754,37% | 13,5283 | 7,46425 | 23,1188 | 7,54737 |
| 9,26 | 753,47% | 15,4282 | 7,26207 | 29,9522 | 7,48893 |
| 10,13 | 752,45% | 17,6617 | 7,23593 | 33,498 | 7,48748 |
| 11,17 | 741,67% | 19,7311 | 7,13503 | 36,8884 | 7,42577 |
| 15,69 | 740,04% | 23,8465 | 7,098 | 50,3414 | 7,32341 |
| 18,01 | 736,37% | 26,0826 | 6,93721 | 54,0896 | 7,33866 |
| 20,24 | 732,56% | 39,4891 | 6,86134 | 60,6893 | 7,26824 |
| 23,65 | 727,66% | 45,8237 | 6,83485 |  |  |
| 31,42 | 715,43% | 47,7665 | 6,81125 |  |  |
| 35,80 | 710,60% | 53,7641 | 6,75898 |  |  |
| 38,80 | 705,05% | 55,9364 | 6,75717 |  |  |
| 41,35 | 702,80% | 60,3961 | 6,73902 |  |  |
| 44,76 | 701,27% |  |  |  |  |
| 45,82 | 697,24% |  |  |  |  |
| 50,53 | 693,32% |  |  |  |  |
| 52,66 | 692,09% |  |  |  |  |
| 55,81 | 691,62% |  |  |  |  |
| 57,03 | 689,33% |  |  |  |  |
| 59,37 | 685,08% |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| tsp\_171 – 60 sekund, wstawienie | | | | | |
| **Populacja – 50** | | **Populacja – 100** | | **Populacja 150** | |
| **Czas [s]** | **Błąd względny** | **Czas [s]** | **Błąd względny** | **Czas [s]** | **Błąd względny** |
| 1,16645 | 770% | 2,37824 | 781% | 3,49216 | 749% |
| 2,88549 | 775% | 7,50692 | 776% | 6,99359 | 736% |
| 4,81828 | 759% | 14,6005 | 765% | 13,9153 | 726% |
| 7,44649 | 755% | 19,2473 | 755% | 17,4679 | 721% |
| 11,1427 | 750% | 26,1282 | 742% | 31,7107 | 716% |
| 15,5056 | 740% | 28,4509 | 741% | 35,1793 | 703% |
| 16,7327 | 739% | 30,8992 | 737% | 53,2303 | 702% |
| 18,5856 | 732% | 33,1239 | 730% | 56,6451 | 695% |
| 23,7655 | 729% | 42,26 | 729% | 60,3106 | 695% |
| 24,8457 | 722% | 52,6455 | 723% |  |  |
| 26,7228 | 712% | 55,2937 | 715% |  |  |
| 27,8333 | 709% | 58,1737 | 711% |  |  |
| 30,6861 | 701% |  |  |  |  |
| 42,6724 | 695% |  |  |  |  |
| 46,3202 | 692% |  |  |  |  |
| 47,6493 | 691% |  |  |  |  |
| 49,8103 | 690% |  |  |  |  |
| 56,5734 | 682% |  |  |  |  |
| 59,8128 | 682% |  |  |  |  |

## Tsp\_358

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| tsp\_358 – 60 sekund, inwersja | | | | | |
| **Populacja – 50** | | **Populacja – 100** | | **Populacja 150** | |
| **Czas [s]** | **Błąd względny** | **Czas [s]** | **Błąd względny** | **Czas [s]** | **Błąd względny** |
| 5,44985 | 463% | 10,4829 | 462% | 15,1198 | 462% |
| 10,605 | 456% | 20,7499 | 456% | 30,1394 | 455% |
| 24,0883 | 455% | 41,7242 | 451% | 60,000 | 455% |
| 30,5083 | 451% | 60,000 | 451% |  |  |
| 37,0007 | 446% |  |  |  |  |
| 60,000 | 446% |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| tsp\_358 – 60 sekund, inwersja | | | | | |
| **Populacja – 50** | | **Populacja – 100** | | **Populacja 150** | |
| **Czas [s]** | **Błąd względny** | **Czas [s]** | **Błąd względny** | **Czas [s]** | **Błąd względny** |
| 5,93 | 466% | 11,2609 | 460% | 16,79 | 465% |
| 11,57 | 461% | 22,4356 | 455% | 34,05 | 461% |
| 17,27 | 458% | 56,2261 | 451% | 51,49 | 458% |
| 22,78 | 456% | 60,000 | 443% | 60,000 | 448% |
| 28,38 | 457% |  |  |  |  |
| 39,47 | 456% |  |  |  |  |
| 56,18 | 453% |  |  |  |  |
| 60,000 | 453% |  |  |  |  |

## Badanie wpływu współczynnika mutacji na wyniki dla ustalonego współczynnika krzyżowania 0,8 oraz populacji o wartości 50

### tsp\_43

### tsp\_171

### tsp\_358

# Wnioski

Badając wpływ współczynnika mutacji można było zauważyć, że współczynnik powinien być większy niż 0,01. Algorytm dawał lepsze wyniki dla wyższych wartości współczynnika. Jest to prawdopodobnie tym, że dla niższego współczynnika zmiana ścieżki w trakcie trwania algorytmu jest za wolna, nie następuje z optymalną częstotliwością. Natomiast dla wyższych współczynników mutacji następuje znaczne polepszenie wyników.

Jeżeli chodzi o wielkość populacji, podczas porównywania wykresów można zauważyć, że nie należy wybierać dużych populacji początkowych. Najlepsze uśrednione wyniki wychodziły dla populacji o wartości 50.

Metody mutacji także nie zmieniły wiele, chociaż widać, że dla małej populacji metoda inwersji jest nieznacznie lepsza. Za to dla większych populacji daje sobie radę trochę gorzej niż metoda wstawiania.

Ciekawe obserwacje wyniknęły z porównywania wyników dla Algorytmu Genetycznego oraz algorytmu Przeszukiwania Tabu. Algorytm Tabu Search dawał znacznie lepsze wyniki w tym samym czasie wykonania. Poza tym algorytm ten znajdował najmniejsze rozwiązanie w krótkim czasie, a następnie do zakończenia programu nie znajdował już żadnej poprawy.

Ze wszystkich obserwacji dokonanych podczas tworzenia programu oraz testowania go wynika, że nie da się jednoznacznie określić parametrów dla każdej instancji problemu komiwojażera. Każdą z instancji należy traktować osobno i sprawdzać ją dla wielu parametrów oraz współczynników.