Sztuczna Inteligencja i inżynieria wiedzy
INZ002386L

Laboratorium

Zadanie 1

Implementacja i badanie Algorytmów Ewolucyjnych

data aktualizacji: 2.03.2020 * pawel.myszkowski@pwr.edu.pl

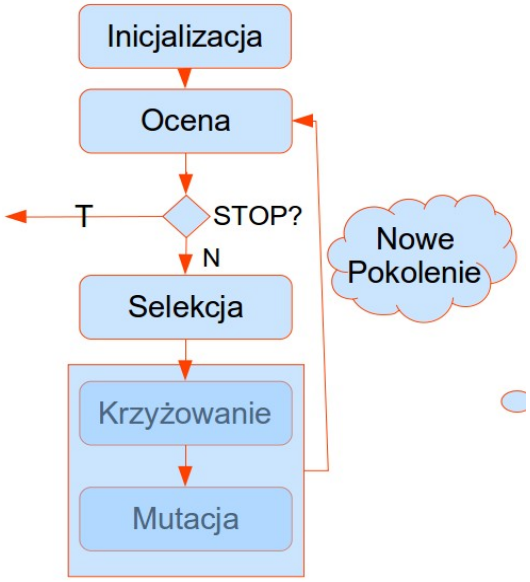
Cel ćwiczenia

Zapoznanie się z metaheurystyką algorytmu ewolucyjnego w praktyczny sposób poprzez samodzielną implementację. Algorytm ewolucyjny jest metaheurystyką (choć nazwa inaczej sugeruje – nie jest to algorytm a metaheurystyka), która naśladuje ewolucję naturalną metodą ciśnienia selekcyjnego i doboru naturalnego. Aby ją zastosować, należy zdefiniować potencjalne rozwiązanie (osobnika), sposoby jego zmiany (mutacja), łączenia (krzyżowania) oraz oceny jakości rozwiązania (funkcja oceny).

Uwaga odnośnie terminologii używanych metod.

Algorytm, wedle definicji jest to skończona sekwencja operacji pozwalający w skończonym czasie rozwiązać problem. Czas działania algorytmu zależy od rozmiaru danych wejściowych (tj. złożoność obliczeniowa algorytmu) i gwarantuje znalezienie rozwiązania optymalnego. W przypadku **heurystyki** złożoność obliczeniowa może być różna (zależy od pomysłowości programisty) i nie gwarantuje znalezienia optymalnego rozwiązania. W tym kontekście **metaheurystyką** nazwiemy metodę, która jest heurystyką wyższego poziomu, tj. heurystyką która określa jakie/jak stosować heurystyki niższego poziomu. Przykładem heurystyki niższego poziomu jest inicjalizacja, selekcja, krzyżowanie itp.

Algorytm ewolucyjny (EA) jest metaheurystyką i opiera się na schemacie przedstawionym na schemacie 1 (patrz Rysunek 1) i w pseudokodzie 1. EA wstępnie inicjalizuje (zwykle losowo) populację rozwiązań. Następnie ocenia jakość poszczególnych osobników. W kolejnym kroku sprawdzane są warunki zatrzymania: czy osiągnięto akceptowalne rozwiązanie i/lub limit pokoleń został przekroczony. Wybór osobników (sugerowana metoda turnieju lub ruletki) do następnej populacji następuje przed działaniem operatora krzyżowania i mutacji, które budują osobniki nowego pokolenia. Dalej, następuje ocena nowych rozwiązań i cykl EA się zamyka przy sprawdzaniu warunków zatrzymania.



Rysunek 1. Ogólny schemat działania Algorytmu Ewolucyjnego

```
begin
t:=0;
initialise( pop(t0) );
evaluate( pop(t0) );
        while (not stop_condition) do
        begin
          pop(t+1) := selection( pop(t) );
          pop(t+1) := crossover( pop(t+1) );
          pop(t+1) := mutation( pop(t+1) );
          evaluate( pop(t+1) );
          t:=t+1;
        end
return the_best_solution
end
```

Pseudokod 1. Ogólny pseudokod Algorytmu Ewolucyjnego

Na Pseudokodzie 2 przedstawiono inny sposób budowy przepływu osobników w Algorytmie Ewolucyjnym. Wykorzystano tam operatory krzyżowania tworzące jednego osobnika, jednak bez większych przeszkód ten pseudokod może być rozszerzony dla operatorów zwracających dwa osobniki.

W Pseudokod 2 populacja bieżąca *Pop(t)* stanowi podstawę do operacji dla selekcji – wybierane są osobniki *P1* i *P2*, a następnie sprawdzane jest prawdopodobieństwo krzyżowania. Jeśli zachodzi, operacja jest wykonywana a osobnik potomny *O1* jest mutowany. Aby nie „marnować” przebiegu EA, w przypadku braku operacji krzyżowania osobnik *O1* jest kopiowany od rodzica *P1*, a dalej mutowany. W dalszej kolejności podlega ocenie i dodania do populacji następnej – *Pop(t+1)*. Procedura może być łatwo rozbudowana o operację krzyżowania, która zwraca dwa osobniki potomne.

```
begin
t:=0;
initialise( pop(t) );
evaluate( pop(t) );
```

```
        while (not stop_condition) do
        begin
          while ( pop(t+1).size()!=pop_size )
               P1 := selection( pop(t) );
               P2 := selection( pop(t) );
               if ( rand[0,0..1,0] < Px)
                  O1:= crossover( P1, P2 );
               else O1:=P1;
               O1 := mutation( O1, Pm );
               evaluate( O1 );
               pop(t+1).add( O1 );
               if ( the_best_solution > O1 )
                           the_best_solution:=O1
          end
        t:=t+1;
        end
return the_best_solution
end
```

Pseudokod 2. Pseudokod jako przepływ osobników dla Algorytmu Ewolucyjnego

W zadaniu rozwiązywany będzie klasyczny problem komiwojażera (*Travelling Salesman Problem*), który jest problemem NP-trudnym, gdzie szukamy najkrótszej trasy przejazdu przez wszystkie zadane miasta. Osobnik EA dla problemu TSP może być **reprezentowany** jako sekwencja miast, np. 35124. Wystartujemy z miasta nr 3, potem odwiedzimy miasto 5, potem 1 itd. Na koniec z miasta 4 wracamy do miasta 3 (początkowe). W takiej reprezentacji każde z miast musi wystąpić tylko raz.

Operator **mutacji** dla powyższej reprezentacji wektorowej sprowadzać się może do zamiany losowo dwóch lokalizacji (tzw *swap)*. Przykład działania mutacji przedstawiono na Rysunku 2. Polega ona na wyznaczeniu dwóch „miejsc” w osobniku i zamianie ich wartości. Na przykładzie pokazano zamianę pozycji drugiej (miasto 6) i przedostatniej (miasto 3). W wyniku wymiany powstaje nowy osobnik.

Można także rozpatrzyć mutację typu inwersja (ang. *inversion*) – przykład jej działania przedstawiono na Rysunku 2. Jej działanie polega na wyznaczeniu losowo dwóch miejsc w osobniku i w ramach tego obszaru „odwracamy” kolejność elementów (miast).

Swap – |5671234| → |5371264|
Inwersja |5671234| → |5321764|

Rysunek 2. Schemat działania mutacji typu SWAP i Inwersja

Operator **krzyżowania** łączy dwa osobniki tworząc potomny (lub dwa, zależnie od operatora). W najprostszej postaci krzyżowanie znajduje losowy punkt przecięcia i dla potomka pierwszą część bierze od jednego rodzica, drugą część od drugiego. Uwaga! Po tej operacji należy sprawdzić czy nie brakuje jakieś lokalizacji i/lub która występuje podwójnie w osobniku potomnym. Wymagana jest procedura „naprawy” osobnika. Przykłady operatorów krzyżowania (OX, CX, PMX) zostały przedstawione na wykładzie.

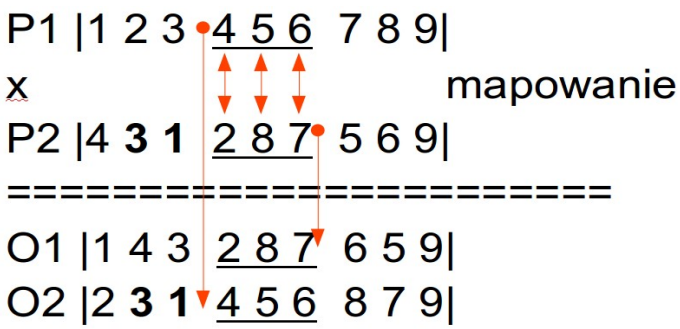
Operator krzyżowania OX (*Ordered Crossover*) polega na wyznaczeniu dwóch miejsc w jednym osobniku rodzicielskim. Ten podciąg będzie wpisywany bez zmian do osobnika potomnego bez zmian. Z drugiego osobnika rodzicielskiego będzie kopiowana kolejność miast. Przykład działania operatora krzyżowania OX przedstawiono poniżej.

Podciąg od jednego rodzica, uzupełnij kolejnością od drugiego

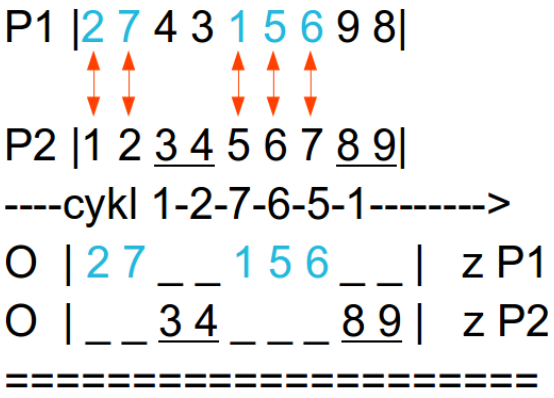
P1 |123456789| x P2 |574913628|

→ O: |793456128|

Operator krzyżowania PMX (*Partially-Matched Crossover)* jest nieco bardziej skomplikowany w działaniu. Wyznacza się dwa losowe punkty, które będą wyznaczać podsekcję krzyżowania. W osobników rodzicielskich wyznacza się sekcję, która jest mapowana, wyznaczane są miasta sobie odpowiadające. W osobnikach potomnych sekcja mapowania jest przepisywana od jednego osobnika rodzicielskiego, a brakujące miasta przepisywane są w kolejności z drugiego rodzica. Przykład działania operatora krzyżowania PMX przedstawiono poniżej:



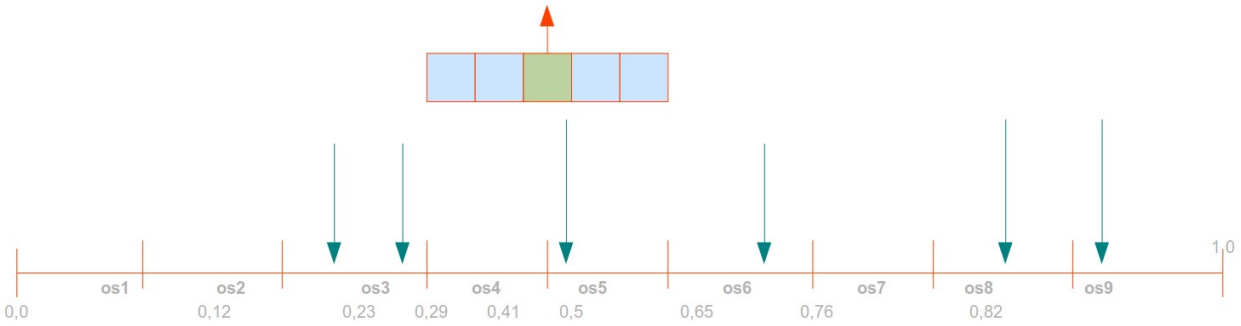
Ciekawym przykładem operatora krzyżowania jest CX (*Cycle Crossover*). Zasada działania tego operatora polega na zestawieniu obu rozwiązań rodzicielskich i wyznaczeniu „cyklu”, czyli które miasta sobie odpowiadają pozycją, a następnie na tej podstawie wyznaczenie „trasy” podróży (cyklu). W dalszych krokach buduje się osobnik potomny, który oparty jest na miastach cyklu odpowiednio z jednego z rodziców. Pozostałe miasta uzupełnia wg kolejności miast drugiego rodzica. Przykład działania operatora krzyżowania CX przedstawiono poniżej:



Operatory krzyżowania opisane są literaturze podanej na końcu dokumentu. Najbardziej polecaną jest pozycja [4].

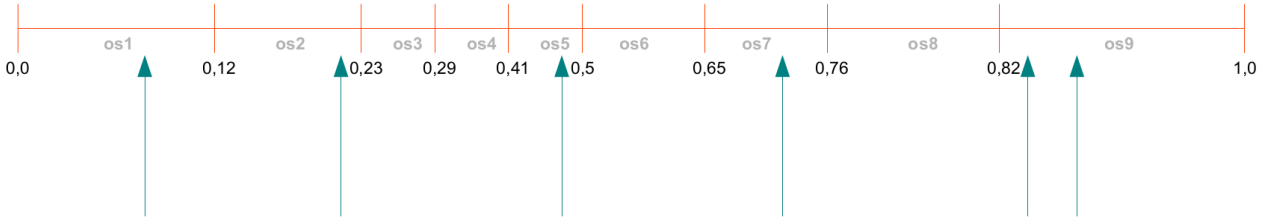
Operator **selekcji** stanowi bardzo ważny element EA – to on decyduje o ciśnieniu selekcyjnym. Jeśli ciśnienie jest zbyt duże populacja szybko się ujednolica i utyka w lokalnym optimum. Jeśli ciśnienie jest za małe (lub go brak), jest EA ma trudność ze zbieżnością. Najczęściej stosowanymi metodami selekcji jest turniej i ruletka.

Metoda selekcji **turnieju** (patrz Rysunek 2) polega na losowym wyborze N[1] osobników z populacji i wskazanie w wyniku osobnika z najlepszym przystosowaniem. Jeśli w $N=1$ wybór osobników jest losowy i brak jakiegokolwiek ciśnienia selekcyjnego. Dla N równego rozmiarowi populacji metoda turniejowa bliższa jest metodzie elitarnej (opis poniżej), czyli wyboru najlepszego osobnika z populacji.



Rysunek 2. Schemat działania operatora selekcji turniejowej (n=5)

Inna metoda selekcji, tzw. **ruletka** (patrz Rysunek 3), która pozwala na wybór osobnika z prawdopodobieństwem proporcjonalnym do jego jakości (funkcji przystosowania). Warto wskazać, że ruletka dla problemów kombinatorycznych przy dużych wartościach funkcji oceny i małych (%) różnicach w wartościach może powodować za małe ciśnienie selekcyjne – wtedy należy rozważyć skalowania funkcji przystosowania, gdzie uwypukla się różnice w wartościowaniu funkcji oceny dla wskazanej populacji.



Rysunek 3. Schemat działania selekcji ruletki

Dodatkowy parametrem jest **elitaryzm**, który czasem też jest nazywamy metodą selekcji. W praktyce jest to wybór M najlepszych osobników z populacji. Wartość M – podobnie jak wartość N dla turnieju może być arbitralna lub procentowa – określa ile osobników bez zmian przechodzi do następnego pokolenia. Parametr elitarny może być bardzo przydany dla dużych populacji (nie gubimy najlepszych osoników) jednak dla małych jest bardzo zgubne, bo może „kotwiczyć” populację w lokalnych optimach.

Metoda **inicjalizacji** jest kluczowa dla działania EA nie tylko dla początku procesu przeszukiwania przestrzeni rozwiązań. Za mało różnorodne osobniki początkowe mogą szybko spowodować utknięcie EA w lokalnym optimum. Z tego względu jako metodę inicjalizacji zwykle używa się losowych punktów startowych. Można też zastosować metody, które wstępnie przeszukają przestrzeń i dadzą EA lepsze punkty startowe. Przykładem takiej metody może być algorytm zachłanny (opisany poniżej), który buduje rozwiązanie lepsze od losowego.

1 N jest parametrem metody Turniej -- może mieć wartość liczbową lub procentową w stosunku do rozmiaru populacji na której pracuje, np. 5%

Realizacja ćwiczenia

Zadanie ma kilka celów, które mogą być sprecyzowane jako:

- Zapoznanie się z metaheurystyką – algorytmy ewolucyjne
- Określenie problemu optymalizacyjnego do rozwiązania – minimalizacja długości trasy w klasycznym problemie komiwojażera TSP (ang. *Travelling Salesman Problem)*.
- Zbudowanie algorytmu genetycznego: osobnik, funkcja oceny, zarządzanie populacją
- Zbudowanie **operatorów** genetycznych typu:
  - krzyżowanie (np. OX, CX, PMX),
  - mutacja (np. SWAP i INVERSION),
  - selekcja – np. Turniej i Ruletka
  - inicjalizacja – np. losowa i zachłanna
- Implementacja modelu w dowolnym języku obiektowym. Sugeruje się używanie języków programowania: Java, Python, C/C++ lub C#.
- Zbadanie wpływu wartości dla różnych parametrów (prawd. mutacji Pm, krzyżowania Px, selekcji, rozmiar populacji pop_size, liczba pokoleń gen) na efektywność i skuteczność algorytmu ewolucyjnego
- Sporządzenie sprawozdania z ćwiczenia
- Pokazanie na wykresach zmianę wartości przystosowania (wartość optymalizowanej funkcji celu) w poszczególnych pokoleniach: najlepszy osobnik, średnia wartość w populacji i najgorszy osobnik
- Porównaj działanie algorytmu genetycznego z wybranymi przez siebie, nieewolucyjnymi metodami optymalizacji (np.: metoda losowego przeszukiwania, algorytm zachłanny).
- Proszę zwrócić uwagę na uzyskany wynik, czas działania oraz liczbę wartościowań optymalizowanej funkcji celu. Proszę pokazać i omówić najciekawsze wyniki.

Raport z ćwiczenia powinien zawierać wszystkie punkty wymagane w realizacji zadania.

Implementacja – wskazówki realizatorskie

Przy realizacji ćwiczenia sugeruje się utworzyć program komputerowy, który swoimi funkcjonalnościami obejmie:

- wczytywanie pliku z danymi, zapis problemu do pamięci komputera (logger, problem),
- oceny rozwiązania (osobnika) w kontekście problemu TSP,
- zarządzanie parametrami algorytmu (konfiguracja),
- sterowania przebiegiem algorytmu ewolucyjnego (EA),
- przechowywania rozwiązania (osobnik) i operatorów genetycznych (krzyżowanie, mutacja),
- zarządzania populacją – w szczególności selekcja, inicjalizacja oraz tworzenie nowej,
- logowania postępów metody (logger) do pliku zewnętrznego.

Kod programu powinien być na zaprojektowany na tyle elastycznie aby możliwe było użycie go przy realizacji następnych zadań laboratoryjnych.

Do realizacji zadania potrzebne jest **porównanie** wyników EA do innych nieewolucyjnych metod. Wskazane są dwie metody: losowa i zachłanna. W przypadku losowej, wystarczy uruchomić metodę tyle razy ile jest rozwiązań przejrzanych przez EA (liczba pokoleń x liczba osobników) aby je porównać. W przypadku algorytmu zachłannego, jest on deterministyczny (zawsze daje to samo rozwiązanie) i opiera się na sekwencyjnej konstrukcji rozwiązania opierając się na lokalnej informacji – zawsze wybiera najbliższe miasto. Tutaj, podobnie jak w algorytmie losowym, należy powtórzyć działanie dla różnych punków startowych. Wyniki dla działania metod należy uśrednić z 10 uruchomień.

Mała podpowiedź
Z punktu widzenia programowania sugeruje się rozważyć czy przy tworzeniu osobników czy potrzebujemy **kopii głębokiej** czy też wystarczy kopia płytka? Dla optymalności kodu proszę też rozważyć jak kodujemy osobnika (czy/jaka kolekcja jest użyta) – będzie to miało bardzo duży wpływ na szybkość działania kodu.

Podobne rozważania proszę zrobić przy implementacji turnieju – proszę rozważyć która (i czy?) kolekcja jest przydatna?

Instancje TSP używane w zadaniu

W Tabeli 1. przedstawiono instancje wskazane do realizacji zadania. Podano tam liczbę miast oraz znane optymalne rozwiązanie. W przypadku instancji berlin11_modified dokonano sztucznej redukcji miast do 11 aby umożliwić analizę „ręczną” rozwiązania.

Uwaga! Do realizacji zadania dobrane są pliki, które mogą dać wstępnie lepsze wyniki dla algorytmu zachłannego niż dla EA. Przy dobrej konfiguracji EA i nieco większego czasu obliczeń, EA jest skuteczniejszy od algorytmu zachłannego.

Algorytm Ewolucyjny ma element niedeterminizmu i zwracane wyniki za kolejnymi uruchomieniami mogą się różnić. Z tego powodu wyniki badań przeprowadzonych przy użyciu tych plików (spis w Tabela 1) muszą być powtórzone 10-krotnie, uśrednione i podane wartości odchylenia standardowego.

Tabela 1. Pliki dla problemu TSP

instancja	Liczba miast	Optymalne rozwiązanie
Easy – sprawdzenie czy metoda działa...		
berlin11_modified	11[2]	4038
berlin52	52	7542
Medium – strojenie metody		
kroA100	100	21282
kroA150	150	26524
kroA200	200	29368
Hard – badanie efektywności/skuteczności metody		
fl417	417	11861
ali535	535	202339
gr666	666	294358

Dla każdego wyniku/wykresu należy podać użytą konfigurację metody: rozmiar populacji (*pop_size*), liczba pokoleń (*gen[erations]*), prawdopodobieństwo krzyżowania (*Px*), oraz pradopodobieństwo mutacji (*Pm*). Dodatkowo, jaki typ krzyżowania/mutacji/selekcji został użyty. W przypadku, gdy wymaga ona dodatkowych parametrów, np. rozmiar turnieju (*Tour*), tę wartość także należy podać w konfiguracji.
Dla przedstawienia wyników badań sugeruje się użycia tabelki zbiorczej formatu jak Tabela 2.

Tabela 2. Format sugerowany do przedstawienia wyników działania EA

instancja	Opt.wynik	Alg.losowy [10k]				Alg.zachłanny [N]				Alg.Ewolucyjny [10x]			
		best	worst	avg	std	best	worst	avg	std	best	worst	avg	std
xyz2	101												
...	...	...	...	...	...	...	...	...	...	...	...	...	...

2 Instancja celowo „obcięta” do 11 miast. Pozwala to na dokładną analizę rozwiązania i rozpisanie go na przysłowiowej kartce. Szczególnie może być przydatne przy analizie działania operatorów genetycznych i sprawdzania poprawności działania.

Gdzie:

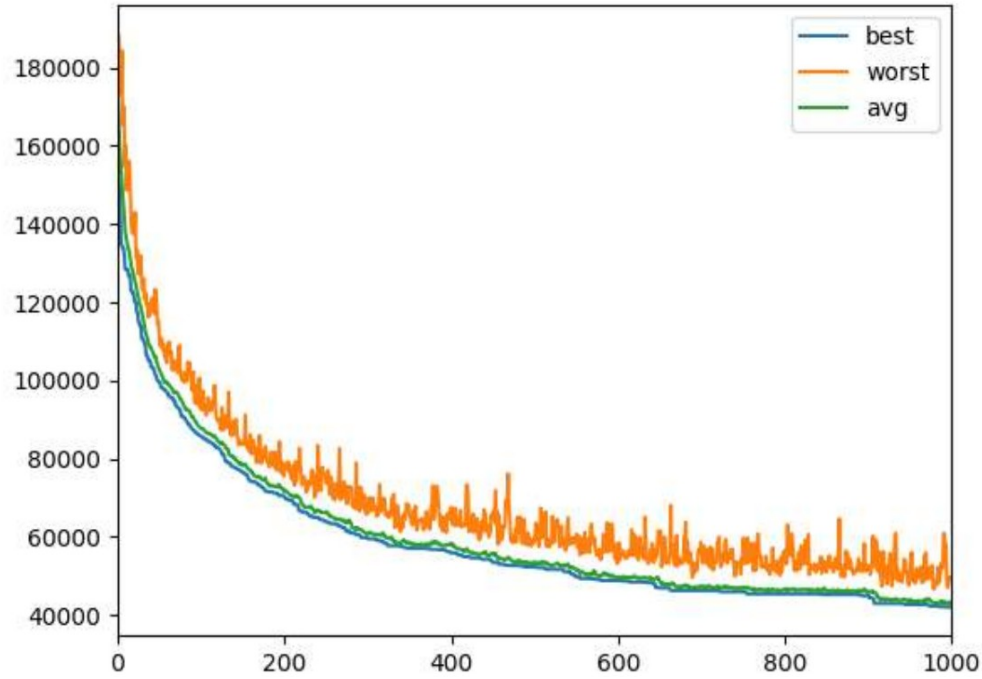
Alg.losowy [10k] oznacza 10.000 uruchomień algorytmu losowego
Alg.zachłanny [N] oznacza N uruchomień algorytmu zachłannego[3]
Alg.Ewolucyjny [10x] oznacza 10-krotne uruchomienie algorytmu ewolucyjnego i podanie wartości statystycznych dla tych uruchomień.

*best** – oznacza najlepszą znalezioną wartość
worst – oznacza najgorszą znalezioną wartość
avg – oznacza średnią znalezioną wartość
std – oznacza wartość odchylenia standardowego liczonego przy średniej (*avg*)

Uwaga: wartość *best** dla algorytmu ewolucyjnego jest pobierana jako najlepsza wartość z 10 uruchomień. Średnia (avg) oznacza wartość liczoną z najlepszego rozwiązania z każdego uruchomienia. Na tej podstawie można utworzyć wykres jak na Rysunek 1.

Tabela 2 przestawia skuteczność zbadanych rozwiązań, czyli jak dobre rozwiązania (średnio) jesteśmy w stanie uzyskać uruchamiając metodę.

Wykres z Rysunku 2 pozwala na wyciągnięcie bardzo wielu cennych wskazówek co do działania i konfiguracji EA.



Rysunek 2. Wykres przykładowego działania EA w rozwiązywaniu problemu TSP

Na Rysunku 2 przedstawiono wykres poprawnego działania skonfigurowanego EA w rozwiązywaniu problemu TSP. Najlepszy osobnik (*best*) schodzi bardzo dynamicznie, choć w wykresie widać „tąpnięcia", które wskazują że występują sytuacje gdy najlepszy osobnik w populacji jest gubiony (nie przechodzi do następnego pokolenia). W przypadku najgorszego osobnika w populacji (*worst*) wykres jest bardziej dynamiczny, co wskazuje na pozytywne zachowanie i ciągłe przeszukiwanie przestrzeni. Odległość wykresu najgorszego i najlepszego wskazuje, że w populacji występuje wystarczająca różnorodność. Trzeci wykres średniego przystosowania (*avg*) wskazuje, że różnorodność jest na wystarczającym poziomie.

3 Dla problemu TSP mamy „tylko" *N* miast. Algorytm zachłanny jest deterministyczny i wystarczy aby każde miasto było startowe i ograniczamy się do *N* uruchomień algorytmu zachłannego. Na tej podstawie wyciągamy średnią.

Ocena realizacji ćwiczenia

Realizacja zadania jest oceniana na podstawie składowych wg poniższego schematu punktów:

2pkt	Zbudowanie modelu EA
2pkt	Zbadanie działania EA na 5 plikach – instancje podane wraz z treścią. Testy muszą być przeprowadzone na: 1 łatwa, 3 średnie i 1 trudna instancja.
1pkt	Zbadanie wpływu rozmiaru populacji *pop_size* i liczby pokoleń *gen* na wyniki działania EA
1pkt	Zbadanie wpływu 2 metod selekcji na skuteczność EA – turniej i ruletka
2pkt	Zbadanie operatorów (po jednym: krzyżowania i mutacji) i wpływu na wyniki EA
2pkt	Porównanie skuteczności EA z wynikami dwóch innych metod nieewolucyjnych

Uwaga: Przy badaniu wyników dla danej metody należy brać pod uwagę przynajmniej 10 uruchomień i uśrednianie wyniku (podajemy wartość średniej i odchylenie standardowe).

Podpowiedzi

Wstępne wartości parametrów przydane do strojenia EA:

pop_size=100, gen=100, Px=0,7, Pm=0,1, Tour=5

Uwaga! optymalne wartości parametrów będą się różnić, w zależności od użytych operatorów oraz rozwiązywanego pliku TSP. Dodatkowo, dla wartości prawdopodobieństwa mutacji należy rozważyć użycie prawdopodobieństwa mutacji (*Pm*) w zależności od:

- **osobnika** dla mutacji działających na osobniku. Przykładem takiej mutacji jest operator inwersji, gdzie niejawnie zakłada się jej tylko jednorazowe działanie na/w danym osobniku. Wtedy *Pm=0,1* oznacza, że statystycznie 10% osobników w populacji będzie mutowane. Dla przykładu, w populacji 100 osobników około 10 osobników będzie zmutowane.
- **genu** dla mutacji działających tylko na małym fragmencie osobnika. Przykładem takiej mutacji może być operator *swap* (w TSP zamiana kolejności dwóch miast). W takim wypadku sugeruje się rozpoczęcie badania od Pm=0,01. Przykładowo, dla problemu TSP z 100 miast i 100 osobników w populacji, średnio będzie zmienione około 200 genów (100*100*0,01 operacji na 2 genach).

Mając logowanie do pliku **.csv* wartości statystyczne dla każdego pokolenia w formacie:

```
<nr_pokolenia; najlepsza_ocena; średnia_ocen; najgorsza_ocena;;;>
```

można „za darmo” uzyskać wykresy w Excelu (jak Rysunek 1). Wtedy można skupić się na merytoryce zadania, mniej na interfejsie, GUI, interaktywnych wykresach itp.

Realizacja zadania

Proponuje się systematyczną realizację zadania wg następującej kolejności etapów.

Zajęcia 1.
Zapoznanie się z problemem.
Implementacja loadera.
Implementacja losowego osobnika TSP – tj. „algorytmu losowego”
Implementacja funkcji f(x) – długość trasy
Implementacja algorytmu zachłannego

Zajęcia 2.
Implementacja operatorów mutacji i krzyżowania.
Implementacja selekcji i zarządzania populacją w algorytmie ewolucyjnym.

Implementacja narzędzi logowania → wykresy.
Wstępne strojenie algorytmu ewolucyjnego.
Badanie wpływu wartości parametrów na skuteczność/efektywność EA.
Przygotowanie sprawozdania: tabelki, wykresy, wnioski.

Zajęcia 3.
(c.d.) Badanie wpływu wartości parametrów na skuteczność/efektywność EA.
(c.d.) Przygotowanie sprawozdania: tabelki, wykresy, wnioski.

Pytania pomocnicze

1. Jak selekcja (rozmiar turnieju *Tour*) wpływa na działanie EA? Co się dzieje jeśli mamy *Tour=1* a co się dzieje jak *Tour=pop_size*?
2. Czym się różni działanie EA z Tour=1 od EA z Tour>1?
3. Czy ruletka działa poprawnie dla problemu TSP?
4. Czy mutacji może być za mało/dużo?
5. Jaką rolę pełni operator mutacji?
6. Czy krzyżowania może być za mało/dużo?
7. Jaką rolę pełni operator krzyżowania?
8. Co się dzieje jeśli wyłączymy krzyżowanie i/lub mutację?
9. Czy selekcja elitarna jest potrzebna?
10. Jak rozmiar populacji i liczba osobników wpływa na efektywność/skuteczność EA?

Uwaga: przy badaniu efektywności metody proszę brać pod uwagę „liczbę urodzeń”, tj. liczbę osobników odwiedzonych przez EA. W praktyce można to sprowadzić się to do określenia wartości *pop_size*gen*

Literatura

1. Arabas J. Wykłady z algorytmów ewolucyjnych (http://staff.elka.pw.edu.pl/~jarabas/ksiazka.html)
2. Goldberg D. Algorytmy genetyczne i ich zastosowanie
3. Michalewicz Z. Algorytmy genetyczne + struktury danych = programy ewolucyjne, WNT.
4. Krunoslav Puljić, Robert Manger, „Comparison of eight evolutionary crossover operators for the vehicle routing problem”, MATHEMATICAL COMMUNICATIONS Math. Commun. 18(2013), 359–375
5. Potvin, Jean-Yves. 1996. Genetic algorithms for the the traveling salesman problem,Annals of Operations Research, Volume 63, pages 339-370.
6. http://edu.pjwstk.edu.pl/wyklady/nai/scb/wyklad10/w10.htm