Sztuczna Inteligencja i Inżynieria Wiedzy

Laboratorium

Zadanie 1

Algorytm Genetyczny
implementacja i badanie

data aktualizacji: 25.02.2021 * Wiktor Walentynowicz, Artur Zawisza

---

Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z klasą problemów optymalizacyjnych oraz metodami ich rozwiązywania na przykładzie Algorytmu Genetycznego. Zakres ćwiczenia obejmuje zapoznanie się z teorią, własnoręczną implementację metody optymalizacji oraz zdobycie intuicji w pracy z nią poprzez wykonanie serii badań elementów i parametrów metody.

Algorytm. Heurystyka. Metaheurystyka

Algorytm jest metodą spełniającą poniższe kryteria:
1. Jest to **skończona lista kroków**,
2. Wykonanie algorytmu **zawsze się zakończy**,
3. Dla zadanego, konkretnego wejścia, zwrócony wynik jest **zawsze** taki sam,
4. Wynik działania algorytmu jest zawsze **wynikiem optymalnym**.

Metoda przybliżona (metoda heurystyczna, heurystyka) w porównaniu do algorytmu **nie daje żadnych gwarancji** znalezienia rozwiązania optymalnego, ale pozwala na uzyskanie **„jakiegoś" rozwiązania** w **ograniczonym czasie**. Heurystyki są często pewnego rodzaju **intuicjami**, które wydają się **dawać szansę** na znalezienie dobrych rozwiązań.

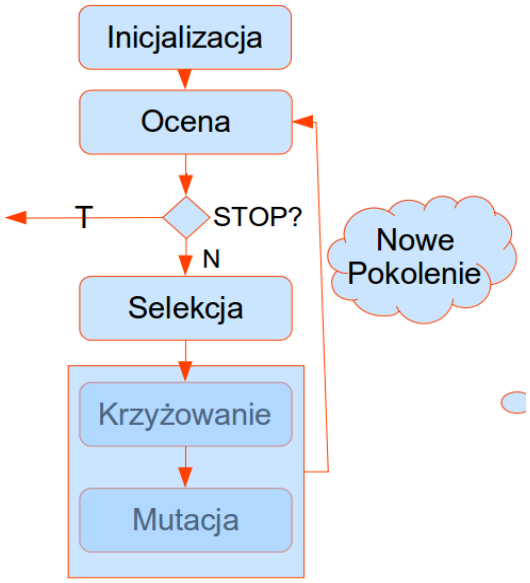
Naturalnym jest, że w celu polepszenia jakości znajdywanych rozwiązań zaczęto **łączyć prostsze heurystyki** w większe metody. **Sposób łączenia** prostszych heurystyk w większą metodę nazywamy ***metaheurystyką***. Można na nie patrzeć jak na **szablony**, które opisują **sposób tworzenia konkretnych heurystyk**.

Algorytm Genetyczny

Algorytm Genetyczny (*ang. Genetic Algorithm*, *GA*) jest przykładem **metaheurystyki** stosowanej do rozwiązywania **problemów optymalizacyjnych**. Optymalizatory utworzone na podstawie schematu Algorytmu Genetycznego również nazywane są Algorytmami Genetycznymi (są to jednak już konkretne heurystyki).

Algorytm Genetyczny zainspirowany został procesem ewolucji organizmów żywych w wyniku doboru naturalnego. Jak wiadomo, organizmy lepiej przystosowane do środowiska naturalnego mają większe szanse na wydanie potomstwa, a przez to utrwalenie swoich cech. Iteracyjny proces przetrwania najlepszych i wymierania najsłabszych osobników skutkuje uzyskaniem populacji coraz lepiej dopasowanej do warunków środowiskowych.

Ideę tę zaadaptowano na potrzeby problemów optymalizacyjnych w sposób przedstawiony na Rysunku 1.



Rysunek 1. Ogólny schemat działania Algorytmu Genetycznego, Autor: Paweł Myszkowski

Pierwszy krok GA stanowi wygenerowanie początkowej, zazwyczaj **losowej populacji rozwiązań**. Nie jest to jednak reguła i możliwe jest wykorzystanie np. wyników działania innych, prostych metod jako źródła populacji początkowej. Rozwiązania w GA, przez analogię do natury, nazywamy **osobnikami**.

W kolejnym kroku, każdy osobnik jest oceniany. **Ocena** polega na określeniu „jak dobrze osobnik jest dopasowany do środowiska", czyli jak dobrze rozwiązuje on problem optymalizacyjny. Sprowadza się to do wyznaczenia wartości funkcji celu, nazywanej tutaj **funkcją przystosowania**, dla każdego z osobników w populacji.

Znając **wartość przystosowania** każdego z osobników można sprawdzić, czy osiągnięto założoną jakość rozwiązania i czy można przerwać proces optymalizacji – **sprawdzenie warunków stopu**.

Jeżeli warunki zatrzymania **nie** zostały spełnione, Algorytm Genetyczny generuje kolejne pokolenie osobników – nową populację. Wykorzystywane są tutaj składowe heurystyki nazywane **operatorami genetycznymi**. W najprostszej, podstawowej wersji, GA zawiera następujące operatory:

- **Operator selekcji**, wybierający osobniki - rodziców generujących potomstwo,
- **Operator krzyżowania**, tworzący osobniki/a – dzieci na podstawie wybranych rodziców,
- **Operator mutacji**, wprowadzający losowe zmiany w kodzie genetycznym dzieci.

Po wykonaniu powyższych operatorów i utworzeniu nowej populacji o rozmiarze identycznym do rozmiaru poprzedniej, proces ewolucji jest powtarzany.

Kolejne kroki Algorytmu Genetycznego przedstawia Pseudokod 1. Warto zaznaczyć, że operatory genetyczne w tym pseudokodzie tworzą od razu całe populacje tymczasowe.

Populacje można też generować w inny sposób – „krokowo", czyli po jednym osobniku. Schemat takiego postępowania przedstawiono na Pseudokodzie 2. Warto zwrócić uwagę, że w tym przykładzie operator krzyżowania *crossover* generuje tu tylko jedno dziecko, nie ma jednak najmniejszego problemu z zaadaptowaniem tego sposobu postępowania na operatory generujące większą liczbę dzieci.

```
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 Genetycznego, Autor: Paweł Myszkowski

Z aktualnej populacji *pop(t)* wybierane są dwa osobniki *P1* i *P2* na rodziców osobnika *O1*. Następnie, z prawdopodobieństwem *Px* dokonywane jest krzyżowanie generujące dziecko *O1*. Jeśli do krzyżowania nie dojdzie, aby nie marnować przebiegu algorytmu, wykonywana jest kopia jednego z rodziców np. *P1*. Następnie dziecko O1 jest poddawane mutacji z prawdopodobieństwem *Pm*. Kroki powtarza się do czasu uzyskania populacji *pop(t + 1)* o rozmiarze równym rozmiarowi *pop(t)*.

Taki schemat działania pozwala ograniczyć zużycie pamięci i zapisać algorytm w sposób ułatwiający jego modyfikację i rozszerzanie.

```
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 := copy(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
```

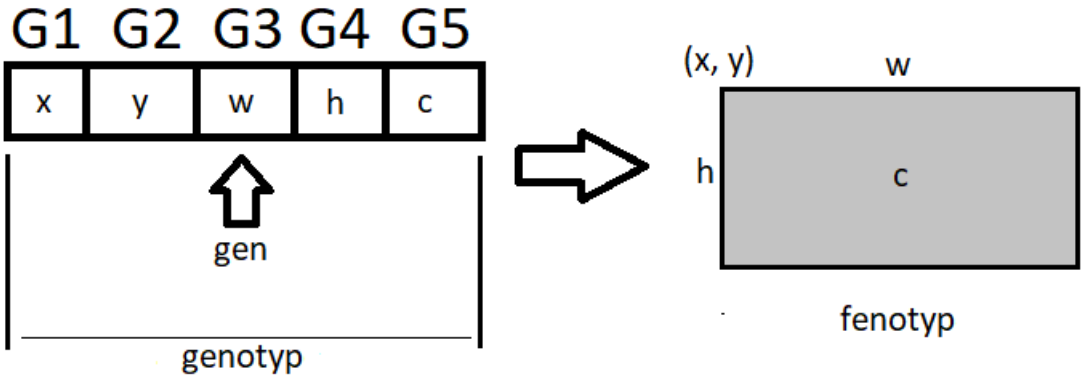
Pseudokod 2. Algorytm Genetyczny w podejściu „krokowym”, Autor: Paweł Myszkowski

Genotyp. Fenotyp. Kodowanie osobnika

Projektując własny Algorytm Genetyczny pierwszym krokiem jest **zdefiniowanie osobnika**.

Klasycznie, w GA **osobnik** jest reprezentowany przez **listę/tablicę o stałym rozmiarze**. Ze względu na inspirację biologiczną, każda **wartość** w osobniku nazywana jest **genem**, a cała **lista genów** – **genotypem**.

Należy przy tym zaznaczyć, że **genotyp** osobnika **nie musi** być równoznaczny z rozwiązaniem. Najczęściej jedynie koduje rozwiązanie. Właściwe, odkodowane rozwiązanie nazywamy **fenotypem**. Porównanie genotypu i fenotypu przedstawiono na Rysunku 2.



Rysunek 2. Przykład genotypu i odpowiadającego mu fenotypu (reprezentacja kolorowego prostokąta)

Istnieje wiele **rodzajów kodowań**, które można wykorzystać projektując osobnika:

- **Kodowanie binarne** – gen przyjmuje wartość 0 lub 1,
- **Kodowanie całkowitoliczbowe** – gen przyjmuje wartość całkowitą z pewnego zbioru,
- **Kodowanie rzeczywistoliczbowe** – gen przyjmuje wartości rzeczywistoliczbowe,

Wybór kodowania jest bardzo istotnym etapem projektu algorytmu, gdyż **determinuje** ono jakie **operatory** mogą być użyte dalej. Osobnik może korzystać z jednego rodzaju kodowania lub mieszać wiele rodzajów.

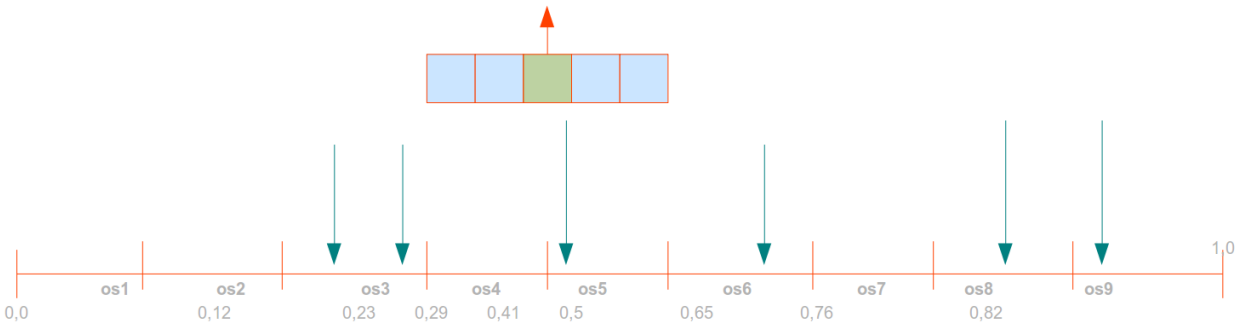
Operator selekcji. Turniej. Ruletka

Pierwszym operatorem używanym w GA jest **operator selekcji** (*ang. selection*). Służy on do **wyboru kandydatów** na rodziców na podstawie **wartości przystosowania** osobników w populacji.

Operator selekcji jest bardzo istotny, gdyż jego zły wybór może doprowadzić do zatrzymania dalszego rozwoju populacji z powodu problemów z tzw. **presją/ciśnieniem selekcyjnym**. Ciśnienie selekcyjne jest generowane przez operator selekcji w wyniku „podbijania" nawet niewielkich różnic w wartości przystosowania. Jeśli ciśnienie jest zbyt wielkie, najlepszy osobnik szybko zdominuje populację, jeśli ciśnienie jest zbyt małe – proces ewolucji przestaje być zbieżny.

Dwa najprostsze, ale i popularne operatory selekcji to **selekcja turniejowa** i **selekcja ruletkowa**.

Operator selekcji turniejowej dokonuje **losowania bez powtórzeń** N osobników z populacji, spośród których wybierany jest osobnik najlepiej przystosowany. Parametr **N** nazywany **rozmiarem turnieju** może być zarówno pewną **wartością stałą** lub być określony jako pewien **procent rozmiaru populacji**. Dla N = 1 wybierany jest osobnik losowy (brak ciśnienia selekcyjnego), dla N równego rozmiarowi populacji wybierany jest najlepszy osobnik z populacji (szybka zbieżność). Rysunek 3 przedstawia schemat selekcji turniejowej.



Rysunek 3. Schemat działania operatora selekcji turniejowej dla N=5 (warto zwrócić uwagę, że os3 został wylosowany dwukrotnie – losowanie powtórzono), Autor: Paweł Myszkowski

Operator selekcji ruletkowej przypisuje każdemu osobnikowi z populacji **prawdopodobieństwo** bycia wylosowanym zależne od tego jak **dobrze przystosowany** jest dany osobnik. Dla metody ruletki definiuje się często tzw. **funkcję wagową**, której celem jest przeliczanie **przystosowania na wagi** oraz dbanie o odpowiednie **ciśnienie selekcyjne**. Najczęściej stosuje się funkcje liniowe lub wykładnicze.

Wybór osobnika polega na wylosowaniu liczby z przedziału <0, 1> i na jej podstawie wskazaniu osobnika, na którego „pole koła ruletki" wypada ta wartość. Metoda daje szansę każdemu z osobników na zostanie rodzicem. Przykład działania metody ruletkowej przedstawiono na Rysunku 4.

Rysunek 3. Schemat działania selekcji ruletki, Autor: Paweł Myszkowski

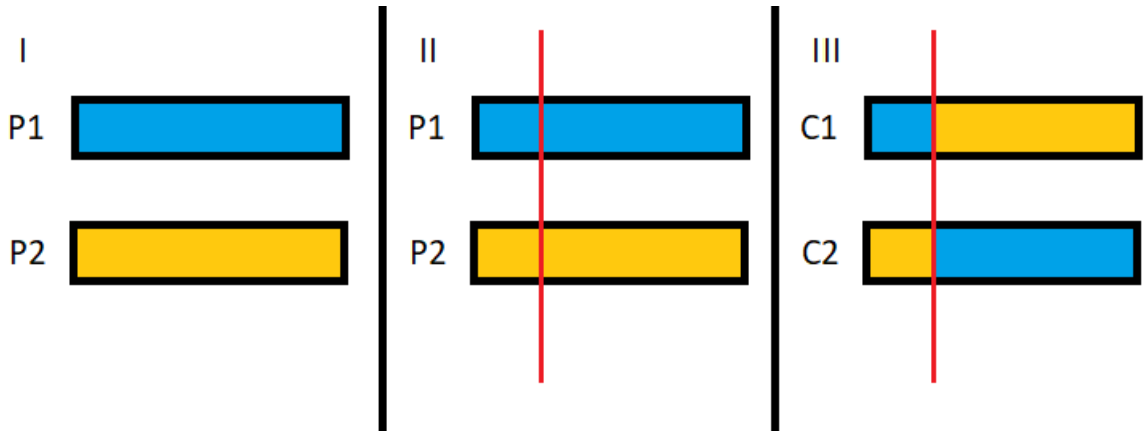
Operator krzyżowania

Drugim w kolejności operatorem genetycznym pojawiającym się w podstawowym Algorytmie Genetycznym jest **operator krzyżowania** (*ang. crossover*) służący do **tworzenia osobników potomnych** na podstawie rodziców. Operatory krzyżowania mogą generować jedno lub więcej dzieci. Krzyżowanie odbywa się zazwyczaj z pewnym prawdopodobieństwem *Px*.

Tego typu operatory są **silnie zależne od kodowania** osobnika, gdyż dla pewnych problemów wartości jednych genów są zależne od innych i **naiwne krzyżowanie** może tworzyć **niepoprawne osobniki**. Dla tego typu problemów albo stosuje się **dedykowane kodowania i operatory krzyżowania**, albo rozszerza Algorytm Genetyczny o dodatkowy **operator naprawczy**, który poprawia uszkodzone genotypy.

Do najprostszych operatorów krzyżowania można zaliczyć **krzyżowanie jednopunktowe**, **krzyżowanie wielopunktowe** oraz **krzyżowanie równomierne**.

Operator krzyżowania jednopunktowego wybiera **losowy punkt** między genami, rozcina oboje rodziców w tym punkcie i tworzy dzieci poprzez wzięcie **jednej części z jednego** rodzica, a **drugiej z drugiego**. Schemat krzyżowania jednopunktowego przedstawiono na Rysunku 4.



Rysunek 4. Schemat działania operatora krzyżowania jednopunktowego

Operator krzyżowania wielopunktowego uzyskujemy z operatora krzyżowania jednopunktowego wybierając **więcej punktów przecięcia**.

Operator krzyżowania równomiernego przechodzi po obojgu rodziców gen po genie i z prawdopodobieństwem *Px* zamienia geny. Można go traktować jako graniczny przypadek krzyżowania wielopunktowego.

Operator mutacji

Ostatnim z operatorów pojawiającym się w podstawowym GA jest **operator mutacji** (*ang. mutation*). Ma on za zadanie wprowadzać do genotypu rzadkie, **niewielkie, losowe zmiany**. Celem tych zmian jest umożliwienie populacji **ucieczkę z obszaru ekstremum lokalnego**, w którym proces ewolucji mógł utknąć.

Operatory mutacji **bardzo silnie** zależą od przyjętego sposobu kodowania rozwiązań.
Najczęściej polegają one, lecz nie są ograniczone, do:

- negacji bitu (kodowanie binarne),
- dodania wartości ±1 do genu (kodowanie całkowitoliczbowe),
- dodania wartości ±ε do genu (kodowanie rzeczywistoliczbowe).

Dla specjalizowanych, dedykowanych sposobów kodowań stosowane są dedykowane metody mutacji. Możliwe jest także wykonywanie naiwnej mutacji i późniejsza naprawa genotypu.

Mutacja odbywa się z pewnym prawdopodobieństwem *Pm*. **Prawdopodobieństwo mutacji** najczęściej oznacza prawdopodobieństwo zmiany **pojedynczego genu** (mutacja „po genie”), ale zdarzają się też operatory, dla których prawdopodobieństwo to oznacza szansę **zmiany osobnika** (mutacja „po osobniku”).

Rozszerzenia Algorytmu Genetycznego

Przedstawiony dotychczas schemat Algorytmu Genetycznego **nie jest** schematem sztywnym i pozwala na wprowadzanie wielu zmian. Najprostszym rozszerzeniem jest definiowanie dodatkowych operatorów genetycznych np. operatora naprawczego, wielostopniowej selekcji, krzyżowania i mutacji, podział populacji na izolowane grupy (tzw. model wyspowy) itd.

Jedną z takich modyfikacji jest wprowadzenie tzw. elityzmu. ***Elityzm*** oznacza, że część populacji złożona z pewnej liczby/procentu najlepszych osobników zostaje bez zmian przeniesiona do kolejnej populacji. Zachowanie elity pozwala ustabilizować przebieg ewolucji jednak źle dobrana proporcja może poskutkować przedwczesną zbieżnością.

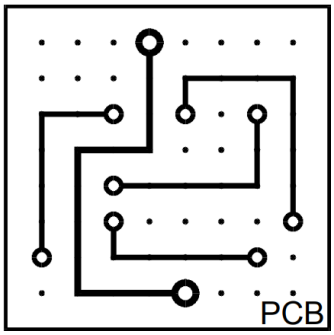
Ćwiczenie – problem projektowania płytek drukowalnych (PCB)

Algorytmy ewolucyjne znajdują zastosowanie w projektowaniu płytek drukowanych. W zadaniu laboratoryjnym zbadane zostaną możliwości zastosowania algorytmów ewolucyjnych do rozwiązania problemu połączeń fizycznych na jednowarstwowych płytkach PCB[7 s. 23-44]. W następującym problemie dane są:

- ograniczony, spójny obszar płaszczyzny zwany dalej płytką drukowaną
- uporządkowany zbiór punktów lutowniczych
- funkcja przyporządkowująca każdemu punktowi ze zbioru ***P*** jego pozycję na płytce zadaną parą współrzędnych kartezjańskich
- uporządkowany zbiór planowanych połączeń strukturalnych ***S*** między punktami ze zbioru ***P***, zadanych jako pary punktów, które powinny zostać połączone fizycznie

Problem polega na zaprojektowaniu sieci fizycznych połączeń w taki sposób, aby dowolne dwa punkty lutownicze zostały fizycznie połączone wtedy i tylko wtedy, jeżeli występowało między nimi planowane połączenie strukturalne. Poniżej znajdują się założenia problemu:

- płytka składa się z jednej warstwy, w kształcie prostokąta o ustalonych rozmiarach
- punkty leżą na współrzędnych całkowitych
- fizyczne połączenia mogą być prowadzone tylko wzdłuż nałożonej na powierzchnię płytki siatki, zbudowanej z kwadratów o boku równym 1
- fizycznej połączenia które przechodzą przez siebie powodują połączenie wszystkich par punktów w jedną całość → fizyczne ścieżki między dwiema różnymi parami **NIE MOGĄ** się przecinać



Rysunek 5. Przykładowy projekt płytki PCB

Sposób kodowania osobnika

Każdy osobnik powinien przedstawiać rozwiązanie zawierające wszystkie wymagane połączenia fizyczne. Każde takie połączeni składa się z sekwencji segmentów o różnej długości. Segmenty te mogą być pionowe bądź poziome o całkowitoliczbowej długości. Aby dało się odczytać ścieżkę złożoną z takich segmentów, należy przed kodowaniem przyjąć, który punkt z pary jest punktem początkowym. Chromosom ma strukturę hierarchiczną, a reprezentacja ścieżki jest zmiennej długości. Trzeba o tym pamiętać podczas projektowania operatorów krzyżowania i mutacji.

Operatory

Operator krzyżowania musi uwzględniać zmienną liczbę segmentów w ścieżce. Dlatego też tworzenie operatora działającego na poziomie pojedynczych ścieżek powodowałoby problemy z zaproponowanym sposobem reprezentacji chromosomu. Sugeruje się wprowadzenie operatora krzyżowania, który polega na wymianie całych ścieżek pomiędzy rodzicami.

Operator mutacji powinien działać na poziomie ścieżki – oznacza to, że powinien modyfikować przebieg ścieżki pomiędzy dwoma punktami. Przykładowym operatorem mutacji może być modyfikacja długości wybranych segmentów o stałą wartość, usunięcie i utworzenie nowego segmentu i tym podobne. Operator

mutacji NIE POWINIEN zmieniać orientacji segmentu (pionowej, poziomej). Mutacja może też działać na ograniczonym obszarze do fragmentu segmentu.

Funkcja oceny

Funkcja oceny powinna minimalizować długość ścieżek oraz liczbę segmentów, z których się składają. Oprócz tego powinna karać osobników niespełniających ograniczenia. W przypadku, gdy minimalizujemy wartość funkcji oceny należy pamiętać o tym fakcie przy implementacji operatora selekcji jakim jest ruletka. Przykładowym zbiorem cech dla funkcji oceny mogą być:

- liczba przecięć
- sumaryczna długość ścieżek
- sumaryczna liczba segmentów tworzących ścieżki
- liczba ścieżek poza płytką
- sumaryczna długość części ścieżek po za obszarem płytki

Każda taka cecha posiada swoją wagę aby zróżnicować istotność danego elementu w funkcji oceny. Ostateczna funkcja oceny składa się z sumy wartości wszystkich cech razy ich wagi.

Realizacja ćwiczenia

Zadanie ma kilka celów, które mogą być sprecyzowane jako:

- Zapoznanie się z metaheurystyką – algorytmy ewolucyjne
- Określenie problemu optymalizacyjnego do rozwiązania – zaprojektowanie płytki drukowanej spełniającej wymagane ograniczenia
- Zbudowanie algorytmu genetycznego: osobnik, funkcja oceny, zarządzanie populacją
- Zbudowanie operatorów genetycznych typu:
    - Inicjalizacja
    - selekcja
    - krzyżowanie
    - mutacja
- Implementacja modelu w dowolnym języku obiektowym. *Sugeruje się używanie języków C/C++, C#, Java, Python*.
- Zbadanie wpływu wartości dla różnych parametrów na **efektywność** i **skuteczność** algorytmu ewolucyjnego:
    - selekcji
    - prawd. krzyżowania
    - prawd. mutacji
    - rozmiaru populacji
    - liczby pokoleń
- Sporządzenie sprawozdania z ćwiczenia
- Pokazanie na wykresach zmianę wartości przystosowania w poszczególnych pokoleniach:
    - najlepszy osobnik
    - średnia wartość w populacji
    - najgorszy osobnik

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

Implementacja – wskazówki realizatorskie

Przy realizacji ćwiczenia utworzony zostanie program komputerowy. Sugerowane właściwości:

- wczytywanie pliku z danymi, zapis problemu do pamięci komputera
- ocena rozwiązania w kontekście problemu projektowania płytki drukowanej
- zarządzanie parametrami algorytmu
- sterowanie przebiegiem algorytmu ewolucyjnego
- przechowywanie rozwiązania i operatorów genetycznych
- zarządzanie populacją - selekcja, inicjalizacja
- logowanie postępów metody do pliku zewnętrznego

Podpowiedź

Z punktu widzenia programowania sugeruje się rozważyć czy przy tworzeniu osobników potrzebujemy **kopii głębokiej** czy też wystarczy kopia płytka? Dla optymalności kodu proszę też rozważyć sposób kodowania osobnika (jakich kolekcji użyjemy). Na to samo proszę zwrócić przy realizacji turniejowego operatora selekcji.

Instancje PCB używane w zadaniu

Do zadania udostępniono dwa zbiory problemów: problemy testowe, które są w pełni opisane, aby można było sprawdzić poprawność implementacji programu oraz problemy badawcze, na których mają być przeprowadzone badania i wykonany raport. Dla każdego z tych problemów należy wygenerować dane wyjściowe, na podstawie których opracowane zostaną tabele i wykresy.
Dla każdego wyniku/wykresu należy podać użytą konfigurację metody:

- rozmiar populacji
- liczba pokoleń
- prawd. krzyżowania
- prawd. mutacji
- typ krzyżowania, mutacji, selekcji
- *dodatkowe parametry wynikające z konfiguracji*

Dla przedstawienia wyników badań sugeruje się użycie tabeli zbiorczej, takiej jak poniższa tabela

instancja	Alg.ewolucyjny [10x]				Metoda losowa [N]			
	best	worst	avg	std	best	worst	avg	std
xyz2								
...	...	...	...	...	...	...	...	...

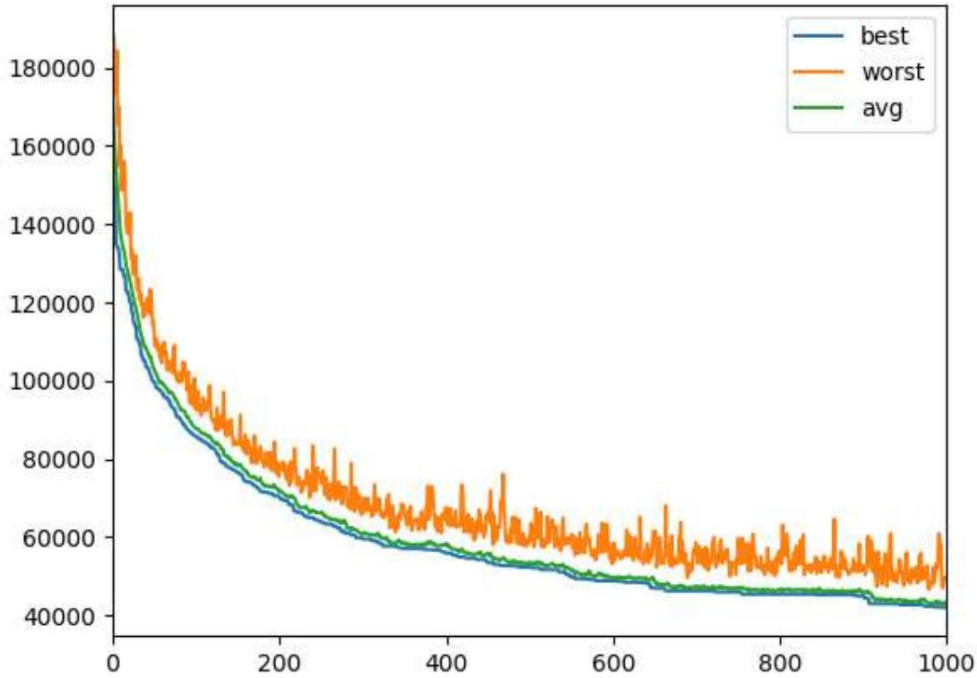
Gdzie:

Alg. Ewolucyjny [10x] oznacza 10-krotne uruchomienie algorytmu ewolucyjnego i podanie wartości statystycznych dla tych uruchomień.
Metoda losowa [N] – uruchomiona tyle razy ile rozwiązań przejrzało 10-krotne uruchomienie algorytmu ewolucyjnego.

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

Uwaga

Wartość *best** dla algorytmu ewolucyjnego jest pobierana jako najlepsza wartość z 10 uruchomień. Średnia oznacza wartość liczoną z najlepszego rozwiązania z każdego uruchomienia. Na tej podstawie można utworzyć wykres taki jak na Rysunku 5. Z wykresu można odczytać wiele ważnych informacji o poprawności konfiguracji metody, a w konsekwencji o jej działaniu.



Rysunek 5. Wykres przykładowego działania GA, Autor: Paweł Myszkowski

Punktacja

Zajęcia 1 (max 2 pkt)
- implementacja ładowania danych oraz funkcji przystosowania (1 pkt),
- implementacja metody losowej (1 pkt)

Ocena:
Ocenie podlegać będzie kod – poprawność implementacji oraz logiczne rozbicie programu na klasy. Wymagane jest uruchomienie obu metod celem stwierdzenia poprawności wyników.
Ładowanie danych może być zaimplementowane przed zajęciami – w domu.

Zajęcia 2 (max 4 pkt)
- implementacja 2 operatorów selekcji – turniej i ruletka (2 pkt),
- implementacja wybranego operatora krzyżowania (1 pkt),
- implementacja wybranego operatora mutacji (1 pkt),

Ocena:
Ocenie podlegać będzie poprawność implementacji operatorów. Weryfikacja poprawności nastąpi poprzez uruchomienie całego algorytmu genetycznego (GA) przy oddawaniu zadania, celem pokazania, że zachowuje się on zgodnie z oczekiwaniami. Z tej przyczyny, szkielet GA zaleca się przygotować przed zajęciami. Alternatywnie, można przygotować zestaw testów, które wykażą poprawność działania poszczególnych operatorów. W tym wypadku GA może zostać skończony przez studenta już po zajęciach. W przypadku braku jakiejkolwiek możliwości weryfikacji, liczba punktów będzie obcinana o 50%.

Podczas implementacji GA należy zwrócić szczególną uwagę na sposób przekazywania osobników do operatorów. Przy nadmiernym kopiowaniu algorytm będzie działał długo, a przy nadmiernym współdzieleniu osobników można doprowadzić do niepoprawnego działania całości, np. podczas krzyżowania jest wstępnie wykonywany rzut mający stwierdzić, czy krzyżowanie nastąpi, czy też nie.

W przypadku wystąpienia krzyżowania powinny powstać nowe osobniki i być przekazane dalej – do operatora mutacji. Jeżeli jednak krzyżowanie nie nastąpi, to oryginalni rodzice są przekazywani dalej. Jeżeli na tym etapie nie zostaną wykonane kopie rodziców, to operator mutacji zmodyfikuje osobniki populacji

źródłowej! W konsekwencji kolejne osobniki do nowej populacji powstawać będą na bazie nowej, zmienionej wersji i całość zacznie działać w sposób nieprzewidywalny.

Jeżeli chodzi o testowanie operatorów osobno, to można: stworzyć małą populację (np. o rozmiarze 5, nie losową, można przekazać stałe ziarno do generatora), wyznaczyć wartości przystosowania osobników populacji i pokazać, że np. turniej faktycznie zwraca poprawnego osobnika tj. tego o największej wartości przystosowania, albo ruletka odpowiednio przydziela fragmenty koła do osobników. Jeśli populacja testowa będzie identyczna miedzy uruchomieniami, to można pokazać,
że w selekcji turniejowej działa losowanie osobników i wybór, a w ruletce wybór proporcjonalny. Analogicznie, można stworzyć przykładowych osobników i pokazać, że krzyżowanie poprawnie wymienia geny, a mutacja poprawnie zmienia wybranego osobnika. Dodatkowo warto też pokazać, że nowe osobniki są lub nie są kopiami tych wejściowych.

Operatory selekcji powinny zawsze zwracać stałe referencje (const & w C++), krzyżowanie powinno zawsze zwracać nowe osobniki, niezależne od rodziców, a mutacja powinna zawsze modyfikować zadanego osobnika.

Zajęcia 3 (max 4 pkt)
- zbadanie wpływu parametrów: (1,25 pkt)
  - rozmiar populacji,
  - liczba pokoleń,
  - rozmiar turnieju,
- porównanie operatorów selekcji – ruletka i turniej, (1 pkt)
- zbadanie wpływu parametrów: (1 pkt)
  - prawdopodobieństwo krzyżowania,
  - prawdopodobieństwo mutacji,
- porównanie algorytmu genetycznego z metodami „naiwnymi” – metoda losowa (0,75 pkt)

Ocena:
Ocenie podlegać będzie sprawozdanie z przeprowadzonych badań. Badania powinny być przeprowadzone zgodnie z zaleceniami podanymi w treści ćwiczenia (liczba powtórzeń, stopnie trudności problemu). Sprawozdanie powinno zawierać podsumowanie badań w postaci tabeli wyników oraz wykresów, a także zapisane wnioski z poszczególnych badań. Każde badanie powinno: posiadać nazwę, mieć podany cel badania oraz zestaw parametrów metod np. rozmiar populacji, liczba generacji, użyte operatory, wybrane pliki danych, etc. aby umożliwić ich ewentualne odtworzenie. Ze względu na wymagania i długotrwałość procesu, większość lub nawet całość sprawozdania należy przygotować w domu. Podczas oddawania należy spodziewać się pytań odnośnie uzyskanych wyników, teorii itp.

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
7. Pasek R. „Techniki Ewolucyjne w projektowaniu układu ścieżek na płytkach drukowanych”