Zaawansowane programowanie

wykład 1: wprowadzenie + algorytmy genetyczne

prof. dr hab. inż. **Marta Kasprzak**
Instytut Informatyki, Politechnika Poznańska

Plan wykładów

1. Wprowadzenie + algorytmy genetyczne
2. Metoda przeszukiwania tabu
3. Inne heurystyki
4. Jeszcze o metaheurystykach
5. Algorytmy dokładne

Zakłada się opanowanie przez uczestników wykładów zagadnień (w sensie wiedzy i umiejętności) z zakresu I stopnia Bioinformatyki, w szczególności tych zrealizowanych w ramach przedmiotów Algorytmy i struktury danych, Podstawy programowania, Projektowanie i programowanie obiektowe.

Problemy kombinatoryczne

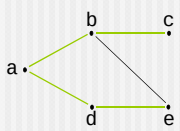
- *Kombinatoryka* jest obszarem matematyki dyskretnej obejmującym operacje na zbiorach elementów i ich atrybutach
- *Problem kombinatoryczny* wyrażony jest w postaci opisu instancji problemu i rozwiązania, którego w danej instancji poszukujemy
- Przykładowe problemy kombinatoryczne: problem plecakowy, problem komiwojażera, kolorowanie mapy, szeregowanie zadań w procesie produkcyjnym

Problemy kombinatoryczne

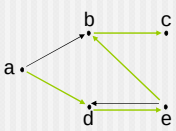
- Problemy kombinatoryczne dzielone są na dwie klasy w odniesieniu do postaci poszukiwanego rozwiązania: klasa problemów *decyzyjnych* i klasa problemów *przeszukiwania*
- W *problemach decyzyjnych* poszukiwanym rozwiązaniem jest odpowiedź „tak” lub „nie” na odpowiednio sformułowane pytanie dotyczące instancji problemu
- W *problemach przeszukiwania* poszukuje się konkretnego rozwiązania — pewnego obiektu czy wartości — spełniającego sformułowane warunki, lub odpowiedzi „nie”, jeśli takie rozwiązanie nie istnieje dla danej instancji
- Większość problemów kombinatorycznych może zostać sformułowana zarówno w postaci decyzyjnej, jak i przeszukiwania

Problemy kombinatoryczne — przykład

- Ścieżka Hamiltona w grafie jest zdefiniowana jako ścieżka odwiedzająca każdy wierzchołek grafu dokładnie raz. Jest to innymi słowy taka permutacja wszystkich wierzchołków, dla której istnieje krawędź lub odpowiednio skierowany łuk w grafie pomiędzy wszystkimi sąsiednimi parami wierzchołków z permutacji



graf nieskierowany



graf skierowany

Problemy kombinatoryczne — przykład

- Sformułowanie problemu poszukiwania ścieżki Hamiltona w grafie skierowanym — wersja decyzyjna
  Instancja: graf skierowany $G=(V, A)$.
  Pytanie: czy istnieje ścieżka Hamiltona w grafie G?
- Sformułowanie problemu poszukiwania ścieżki Hamiltona w grafie skierowanym — wersja przeszukiwania
  Instancja: graf skierowany $G=(V, A)$.
  Rozwiązanie: ścieżka Hamiltona w grafie G.
- Wersja decyzyjna problemu jest nie trudniejsza niż jego wersja przeszukiwania

Problemy optymalizacyjne

- *Problem optymalizacyjny* jest szczególną odmianą problemu przeszukiwania. W problemie takim rozwiązaniem jest pewien obiekt o optymalnej wartości funkcji celu (funkcji kryterialnej) zdefiniowanej w problemie
- Rozwiązanie problemu optymalizacyjnego jest *dopuszczalne*, jeśli spełnia wszystkie warunki jak w sformułowaniu problemu z pominięciem warunku na optymalną wartość funkcji celu. Rozwiązanie o najlepszej wartości funkcji celu spośród dopuszczalnych jest *optymalne*
- Funkcję celu można *maksymalizować* (gdy poszukiwane jest rozwiązanie o najwyższej wartości funkcji) lub *minimalizować* (gdy wartość funkcji ma być jak najniższa)

Problemy optymalizacyjne — przykład

- W *problemie komiwojażera*, dla podanego zbioru miast i odległości między wszystkimi parami miast, poszukiwana jest trasa zamknięta o minimalnej sumarycznej długości odwiedzająca wszystkie miasta
- Funkcją celu jest tu suma odległości pomiędzy parami sąsiednich miast na trasie, jest ona minimalizowana
- Gdy macierz odległości pomiędzy miastami jest kompletna, rozwiązanie dopuszczalne w tym problemie jest łatwo osiągalne i jest nim dowolna permutacja miast
- Znalezienie rozwiązania optymalnego dla problemu komiwojażera jest obliczeniowo trudne

Złożoność obliczeniowa problemów

- Problemy kombinatoryczne dzielimy pod względem ich złożoności obliczeniowej na — w uproszczeniu — problemy łatwe i trudne obliczeniowo
- Problem jest *łatwy obliczeniowo*, jeśli udowodniono, że znalezienie jego dokładnego rozwiązania (dla dowolnej instancji problemu) możliwe jest w czasie ograniczonym od góry wielomianem zależnym od rozmiaru instancji problemu
- Problem jest *trudny obliczeniowo*, jeśli — w uproszczeniu — przeprowadzono dowód jego przynależności do klasy problemów NP-zupełnych (problemy decyzyjne) lub NP-trudnych (problemy przeszukiwania). Dla takich problemów nie są znane dokładne algorytmy wielomianowe

Złożoność obliczeniowa problemów

- W biologii obliczeniowej większość problemów jest trudna obliczeniowo. Zdarza się, że najprostsze teoretyczne modele odzwierciedlające podstawowy wariant problemu biologicznego są rozwiązywalne w czasie wielomianowym. Natomiast uwzględnienie naturalnej złożoności problemu, a zwłaszcza uwzględnienie obecności błędów eksperymentalnych w instancji, skutkuje zazwyczaj trudnym obliczeniowo problemem kombinatorycznym

Złożoność obliczeniowa algorytmów

- Czas obliczeń algorytmu rozwiązującego problem kombinatoryczny — w ogólności — rośnie wraz ze wzrostem rozmiaru instancji problemu
- Funkcja czasu algorytmu utożsamiana jest z liczbą jego elementarnych kroków. Zmienną tej funkcji jest rozmiar instancji problemu n. Jeśli liczba kroków może zostać opisana (ograniczona od góry) funkcją wielomianową zmiennej n, mówimy, że algorytm jest *wielomianowy* (w sensie czasu). W przeciwnym przypadku przyjęło się nazywać algorytm *wykładniczym*
- Jak dotąd nie udowodniono, że problem trudny obliczeniowo można (lub nie) rozwiązać w sposób dokładny algorytmem wielomianowym. W praktyce takie problemy, a także problemy otwarte, rozwiązuje się algorytmami wykładniczymi (choć tylko stosunkowo małe instancje) lub w sposób przybliżony (niedokładny) wielomianowymi *heurystykami*

Heurystyki

- W przypadku problemów trudnych obliczeniowo czas oczekiwania na dokładne rozwiązanie dla większych instancji problemu staje się zbyt duży, nawet dla najlepszych algorytmów wykładniczych
- Użytkownik często w takiej sytuacji jest skłonny zrezygnować z rozwiązania dokładnego na rzecz rozwiązania przybliżonego, za to uzyskanego w akceptowalnym czasie
- *Heurystyką* nazywamy algorytm (metodę) zwracający rozwiązanie przybliżone. Zazwyczaj oczekuje się, że algorytm taki ma złożoność wielomianową i działa szybko w praktyce
- Zaawansowanie heurystyki koreluje z jej złożonością czasową i jakością zwracanych rozwiązań. Przykłady heurystyk: losowa, zachłanna, lokalnego przeszukiwania, metaheurystyka

Heurystyki

- W przypadku *problemów optymalizacyjnych* metoda heurystyczna ma na celu zwrócenie rozwiązania dopuszczalnego o wartości funkcji celu jak najbliższej optymalnej
- Najczęstszym podejściem jest generowanie kolejnych rozwiązań dopuszczalnych z dążeniem do poprawy wartości funkcji celu. Możliwa jest także chwilowa rezygnacja z dopuszczalności bieżących rozwiązań, z odpowiednią korektą rozwiązania pod koniec obliczeń

13

Heurystyki

- W problemie komiwojażera z kompletną macierzą odległości wygenerowanie rozwiązania dopuszczalnego było proste, co nie jest regułą dla dowolnego problemu optymalizacyjnego. Metoda heurystyczna może nie zwrócić żadnego rozwiązania, jeśli nie znajdzie dopuszczalnego
- Jakość heurystyki można ocenić m.in. na podstawie różnicy w wartości funkcji celu osiągniętej przez nią i optymalnej. Wartość optymalną można dla niektórych problemów/instancji lepiej lub gorzej oszacować, można ją także uzyskać (dla mniejszych instancji) istniejącym algorytmem dokładnym

14

Metaheurystyki

- *Metaheurystyki* to złożone metody heurystyczne stosowane do rozwiązywania problemów optymalizacyjnych
- Metaheurystyka jest ogólnym schematem konstrukcji algorytmów, który jest dopasowywany przez twórców do konkretnego problemu
- W ogólności dzielimy metaheurystyki na oparte na pojedynczym rozwiązaniu i na oparte na populacji rozwiązań
- Przykładem metaheurystyki pierwszego rodzaju jest metoda przeszukiwania tabu (ang. *tabu search*), metaheurystyka drugiego rodzaju to np. algorytmy genetyczne (ang. *genetic algorithms*)

15

Metaheurystyki

- Konstruując metaheurystykę, należy realizować jednocześnie dwie sprzeczne strategie: intensyfikacji i dywersyfikacji
- Strategia *intensyfikacji* dąży do iteracyjnej poprawy bieżącego rozwiązania (lub populacji rozwiązań). Realizowana jest zazwyczaj przez wbudowaną heurystykę lokalnego przeszukiwania
- Strategia *dywersyfikacji* dąży do jak najszerszej eksploracji przestrzeni rozwiązań, co wiąże się z dopuszczeniem znacznego nawet pogorszenia bieżącego rozwiązania (lub populacji rozwiązań). W zamian uzyskuje się możliwość wyjścia z obszaru lokalnego optimum
- Strategie te przeplatają się wewnątrz algorytmu

16

Algorytmy genetyczne

[John H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975]

- *Algorytmy genetyczne* oparte są na mechanizmie naturalnej selekcji i ewolucji, zaczerpniętym ze świata rzeczywistego
- W tym podejściu *reguły probabilistyczne* są preferowane nad deterministycznymi
- Niejednokrotnie podstawowy, głównie losowy schemat dochodzenia do suboptymalnego rozwiązania uzupełniany jest deterministyczną heurystyką wspomagającą. Taki algorytm genetyczny nazywany jest *hybrydowym*
- Algorytmy genetyczne operują na *populacji rozwiązań*. Początkowa populacja jest zazwyczaj generowana losowo

17

Algorytmy genetyczne

- Potencjalne rozwiązania wchodzące w skład populacji nazywane są *osobnikami* (ang. *individuals*)
- Z każdym osobnikiem skojarzona jest wartość *funkcji przystosowania* (ang. *fitness function*). Wartość ta wpływa na decyzję, czy dany osobnik będzie *reprodukował się* czy nie. Funkcja ta w pewien sposób odpowiada funkcji kryterialnej w problemie (nie musi być identyczna)
- W każdym cyklu reprodukcji wybierane są najlepsze osobniki (o najlepszym przystosowaniu), kojarzone w pary *rodziców* i *krzyżowane* w celu wyprodukowania następnej populacji
- Powyższy schemat wykorzystuje losowość zaimplementowaną w wielu czynnościach, np. w wyborze miejsc krzyżowania, w *mutacjach* (przypadkowe zmiany w pojedynczym osobniku), w losowym doborze części rodziców, par rodziców, itp.

18

Algorytmy genetyczne

- Rozwiązaniem jest najlepszy osobnik wygenerowany w trakcie działania algorytmu (niekoniecznie z ostatniej populacji)
- Strategia intensyfikacji realizowana jest poprzez dobór osobników o najlepszej wartości funkcji przystosowania. W przypadku algorytmów hybrydowych także przez deterministyczną heurystykę wspomagającą bardziej efektywne dążenie do poprawy rozwiązania (np. przez odpowiedni dobór i krzyżowanie osobników)
- Strategia dywersyfikacji realizowana jest poprzez duży udział losowych wyborów w algorytmie (np. losowy dobór niezbyt dobrych rodziców czy mutacje osobników)

Algorytmy genetyczne — parametry

- Parametrami tej metaheurystyki, umożliwiającymi dostosowanie jej do indywidualnych potrzeb, są:
  - *rozmiar populacji*; najczęściej rozmiar jest stały przez cały czas trwania obliczeń, ale także zdarza się zmienny; duża populacja sprzyja różnorodności, ale wydłuża obliczenia
  - *liczba cykli reprodukcyjnych* (iteracji); generalnie im większa, tym lepszego rozwiązania możemy oczekiwać, przy czym w pewnym momencie osiągamy zbieżność kolejnych populacji i dalsze iteracje nie przynoszą poprawy, jedynie wydłużenie czasu obliczeń; można przyjąć z góry zadaną liczbę iteracji albo np. liczbę iteracji bez poprawy funkcji celu (lub funkcji przystosowania)

Algorytmy genetyczne — parametry

- Dalsza część parametrów algorytmów genetycznych:
  - *prawdopodobieństwo mutacji*; zwykle bardzo małe, aby mutacja dotknęła nieliczne osobniki
  - *prawdopodobieństwo krzyżowania*; określa, jaka część populacji zastępowana jest nowymi osobnikami powstałymi w wyniku reprodukcji rodziców; można dopuścić przeznaczenie części miejsc na kopie najlepszych osobników z poprzedniej populacji (w celu intensyfikacji poszukiwań)
  - ew. *dodatkowe kryterium stopu*; np. limit czasu obliczeń albo jeśli różnorodność populacji spadnie poniżej pewnego wymaganego minimum

Algorytmy genetyczne — reprezentacja

- „Genetyczna” reprezentacja osobnika, czyli odpowiadający mu w algorytmie łańcuch znaków, nazywany jest *chromosomem*
- Pierwotnie chromosom kodowany był binarnie, co sprzyjało losowym operacjom w metaheurystyce (np. mutacjom) i było bliskie 4-znakowemu alfabetowi DNA. W celu utrzymania dopuszczalności rozwiązania stosowane były procedury naprawy chromosomu po krzyżowaniu lub mutacji
- Reprezentacja binarna może być np. ciągiem binarnie zapisanych indeksów elementów występujących w problemie. Może być ciągiem bitów — po jednym na każdy element — mówiącym, czy dany element pojawi się czy też nie w rozwiązaniu. Może być także liniowym zapisem macierzy binarnej, w której jedynka na przecięciu wiersza *i* i kolumny *j* oznacza, że element *i* poprzedza element *j* w rozwiązaniu

Algorytmy genetyczne — reprezentacja

- Obecnie stosuje się także inne metody kodowania, np. w postaci permutacji indeksów elementów problemu (np. miast, zadań, wierzchołków grafu). Wymagają one bardziej zaawansowanych operatorów krzyżowania i mutacji
- W takiej permutacji kolejne indeksy mogą odpowiadać po prostu kolejnym elementom w uporządkowanym rozwiązaniu (ścieżka, cykl). W innym schemacie kodowania indeks *i* na pozycji *j* może oznaczać, że element *j* poprzedza element *i* w rozwiązaniu
- Chromosom, na początku algorytmu zazwyczaj losowy, ewoluując, dąży do poprawy swojego przystosowania. Zakodowane w nim rozwiązanie dopuszczalne może pokrywać cały chromosom lub tylko jego część. Tradycyjnie chromosomy wszystkich osobników są tej samej długości (nie muszą być)

Algorytmy genetyczne — pierwsza populacja

- Sposób wygenerowania populacji początkowej ma wpływ na dalsze obliczenia. Jeśli odpowiednia różnorodność osobników w tej populacji nie zostanie zapewniona, algorytm osiągnie przedwczesną zbieżność
- Dobrze widziana w innych metodach (np. *tabu search* czy *branch and bound*) generacja rozwiązania początkowego prostą wstępną heurystyką tutaj nie sprawdzi się
- Losowy sposób generowania populacji początkowej jest najczęściej stosowany
- Innym sposobem może być celowe generowanie początkowych osobników w taki sposób, aby umieścić ich w odległych obszarach przestrzeni rozwiązań

Algorytmy genetyczne — przystosowanie

- Funkcja przystosowania, związana z każdym osobnikiem, umożliwia dobór najlepszych z nich do procesu reprodukcji
- Dla każdego osobnika musi być możliwość obliczenia wartości tej funkcji, tak więc albo wszystkie chromosomy będą dopuszczalne (lub ich fragmenty), albo zostanie dodane narzędzie naprawy niedopuszczalnych chromosomów. Pozostawienie niedopuszczalnych chromosomów bez naprawy powodowałoby ich natychmiastową eliminację
- Funkcja przystosowania może być identyczna z funkcją celu ze sformułowania problemu. Może także być funkcją celu po zastosowaniu dodatkowych operacji, np. normalizacji. Jeśli obliczenie wartości funkcji celu jest czasochłonne, stosuje się pewne jej przybliżenie

25

Algorytmy genetyczne — selekcja

- Podstawową zasadą rządzącą mechanizmem selekcji jest: „im bardziej przystosowany osobnik, tym większa jego szansa na reprodukcję”
- Z drugiej strony, gorzej przystosowane osobniki nie powinny być z gruntu odrzucane, gdyż mogą mieć pożyteczne fragmenty chromosomu; są także czynnikiem dywersyfikującym przeszukiwanie
- Metod selekcji jest wiele, przykładowe to: metoda ruletki, metoda rankingowa, metoda turniejowa. Dopuszczalne są rozmaite modyfikacje w podstawowych schematach selekcji

26

Algorytmy genetyczne — selekcja

- *Metoda ruletki* (ang. *roulette wheel selection*) jest najczęściej używana. Można ją porównać do obrotu kołem ruletki w celu wylosowania osobników do reprodukcji, na którym to kole osobniki bardziej przystosowane mają szersze przedziały
- Każdemu osobnikowi przypisywane jest prawdopodobieństwo selekcji proporcjonalne do jego przystosowania (zazwyczaj jest to przystosowanie podzielone przez sumę przystosowań wszystkich osobników w populacji). Wartości przystosowania można przeskalować w przypadku zagrożenia dominacją jednego lub kilku najlepszych osobników
- W celu wygenerowania populacji o n osobnikach przeprowadza się n losowań z zachowaniem wyliczonego prawdopodobieństwa selekcji (n losowań pozycji na kole ruletki)

27

Algorytmy genetyczne — selekcja

- *Metoda rankingowa* (ang. *rank-based selection*) nie korzysta z wartości funkcji przystosowania, lecz z rankingu osobników sporządzonego na podstawie tych wartości. Najbardziej przystosowany osobnik ma największą wartość w rankingu, najmniej przystosowany ma 1
- W ten sposób zacierane są znaczne czasami różnice pomiędzy osobnikami (jeden nie zdominuje przestrzeni losowań). Nadal najlepsze osobniki mają największe szanse na wylosowanie
- Prawdopodobieństwo selekcji osobnika do procesu reprodukcji jest funkcją oddającą jego pozycję w rankingu. W literaturze funkcja ta jest różnie definiowana

28

Algorytmy genetyczne — selekcja

- *Metoda turniejowa* (ang. *tournament selection*) wybiera do reprodukcji osobniki najlepsze wewnątrz mniejszych podgrup. Daje dzięki temu szansę wyboru osobnikom słabszym (choć nie najsłabszym)
- Podgrupę o zadanej liczności losuje się spośród wszystkich osobników populacji (bez powielania osobników wewnątrz podgrupy i bez preferencji co do wartości przystosowania). Wewnątrz podgrupy rozgrywa się „turniej”, w którym zwycięzcą jest osobnik o najlepszym przystosowaniu. Ten osobnik jest wybierany do procesu reprodukcji
- Procedurę tę powtarza się n razy, gdzie n jest licznością populacji. W efekcie wyselekcjonowane osobniki mogą się powtarzać

29

Algorytmy genetyczne — reprodukcja

- Rodzice wybrani w procedurze selekcji zostają dobrani w pary, które, krzyżując się, stworzą potomstwo. Dobór jest najczęściej losowy
- Przy założeniu stałej liczności populacji, każda para rodziców wyprodukuje dwa nowe (zazwyczaj różne) osobniki, które odziedziczą po nich cechy — w zamierzeniu najlepsze
- Jeśli prawdopodobieństwo krzyżowania jest mniejsze niż 1, część z rodziców zostanie skopiowana do kolejnej populacji bez zmian
- Czasem dopuszcza się krzyżowanie więcej niż dwóch rodziców

30

Algorytmy genetyczne — krzyżowanie

- Najprostszy operator krzyżowania jest *jednopunktowy*. Oznacza to, że w chromosomach rodziców wybierane jest losowo jedno miejsce krzyżowania

rodzice potomstwo

- Możliwość zastosowania takiego operatora zależy od sposobu zakodowania instancji. W wielu przypadkach chromosomy potomstwa przestaną reprezentować poprawne rozwiązanie (np. w przypadku permutacji elementów)

Algorytmy genetyczne — krzyżowanie

- Uogólnieniem jednopunktowego operatora krzyżowania jest operator dwu- i wielopunktowy

rodzice potomstwo

rodzice potomstwo

Algorytmy genetyczne — krzyżowanie

- W krzyżowaniu *równomiernym* (ang. *uniform crossover*) losowany jest wektor binarny o długości chromosomu. Wartości 1 i 0 utożsamiane są z pozycjami w jednym i drugim rodzicu, które kopiowane są do potomka. Drugi potomek powstaje przez symetryczne zastosowanie reguły kopiowania

0010001110 wektor binarny

rodzice potomstwo

Algorytmy genetyczne — krzyżowanie

- W odniesieniu do chromosomów reprezentujących permutację elementów stosuje się bardziej zaawansowane operatory krzyżowania. Przykładem jest *krzyżowanie z zachowaniem porządku* (ang. *order crossover*)
- Losowane są dwa punkty w chromosomach rodziców. Do pierwszego potomka kopiowany jest fragment pomiędzy nimi z pierwszego rodzica. Fragment ten jest uzupełniany, począwszy od drugiego punktu krzyżowania, elementami z drugiego rodzica, które nie są jeszcze obecne w potomku (także zaczynając od drugiego punktu krzyżowania). Po dojściu do końca chromosomu uzupełniany jest jego początek
- Drugi potomek powstaje przez symetryczne zastosowanie tej reguły

Algorytmy genetyczne — krzyżowanie

- Krzyżowanie z zachowaniem porządku

rodzice krok 1

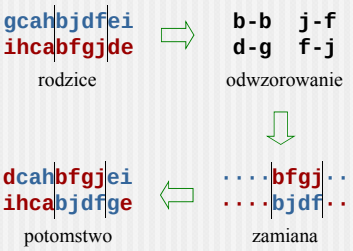
potomstwo krok 2

Algorytmy genetyczne — krzyżowanie

- Innym operatorem, mającym zastosowanie przy permutacjach elementów, jest *krzyżowanie z częściowym odwzorowaniem* (ang. *partially mapped crossover*)
- Losowane są dwa punkty w chromosomach rodziców. Odcinek pomiędzy nimi definiuje odwzorowanie: element z danej pozycji w jednym rodzicu zostanie zastąpiony w potomstwie elementem z tej samej pozycji w drugim rodzicu (dotyczy to wszystkich wystąpień tych elementów, także spoza wylosowanego odcinka). Pozostałe elementy kopiowane są bez zmian z rodzica do potomka
- W przypadku, gdy w potomku element jest już obecny, a mamy ciąg odwzorowań typu a ↔ b, b ↔ c, stosowany jest łańcuch zamian, który daje w efekcie a ↔ c

Algorytmy genetyczne — krzyżowanie

- Krzyżowanie z częściowym odwzorowaniem

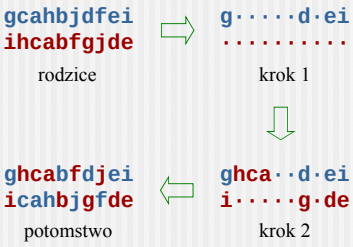


Algorytmy genetyczne — krzyżowanie

- *Krzyżowanie cykliczne* (ang. *cycle crossover*) tworzy potomstwo, którego każda pozycja w chromosomie jest skopiowana z odpowiedniej pozycji jednego z rodziców
- Kopiowanie rozpoczynamy od pierwszego elementu w chromosomie. Dokonany wybór implikuje następny, jeśli nie chcemy powielić dwóch takich samych elementów w jednym chromosomie. Po wyczerpaniu takiego cyklu w podzbiorze elementów wybieramy któregokolwiek z rodziców do skopiowania kolejnej wolnej pozycji i powtarzamy ciąg decyzyjny

Algorytmy genetyczne — krzyżowanie

- Krzyżowanie cykliczne



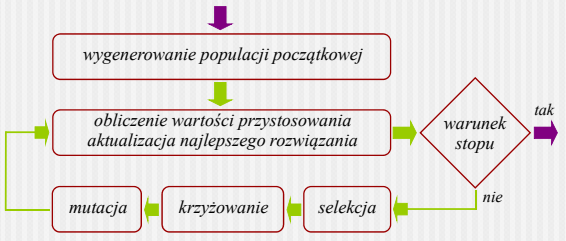
Algorytmy genetyczne — mutacja

- Mutacja w chromosomach binarnych jest zmianą wartości bitu na losowo wybranej pozycji (z 0 na 1 i odwrotnie). W zależności od wybranej metody kodowania chromosom będzie lub nie wymagał naprawy
- W innych chromosomach mutacja może być zmianą jednej z wartości na inną dopuszczalną (losowo wybraną na losowej pozycji). W przypadku permutacji elementów należy zadbać o poprawność ciągu wynikowego
- Dla permutacji można stosować następujące warianty mutacji: przeniesienie elementu lub podciągu elementów w inne miejsce chromosomu, zamiana miejscami dwóch elementów, odwrócenie podciągu elementów (lub odwrócenie i przemieszczenie), przemieszanie elementów w wybranym podciągu, itp.

Algorytmy genetyczne — mutacja

- Mutacje wprowadzane są zazwyczaj po etapie krzyżowania, w nowej populacji, przed wyliczeniem dla jej elementów funkcji przystosowania
- Zakłada się, że mutacje nie występują zbyt często i nie dotykają większości osobników. Przykładowo, można dopuścić mutację jednego lub kilku miejsc w jednej populacji

Ogólny schemat algorytmów genetycznych



Literatura

- Michael R. Garey, David S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, W.H. Freeman & Co., San Francisco, 1979.
- Jacek Błażewicz, *Złożoność obliczeniowa problemów kombinatorycznych*, WNT, Warszawa, 1988.
- El-Ghazali Talbi, *Metaheuristics: From Design to Implementation*, Wiley, Hoboken, 2009.
- John H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge (MA), 1992.
- David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading (MA), 1989.
- Zbigniew Michalewicz, David B. Fogel, *Jak to rozwiązać czyli nowoczesna heurystyka*, WNT, Warszawa, 2006.