Algorytmy ewolucyjne

Piotr Lipiński

Informacje ogólne

- Informacje i materiały dotyczące wykładu będą publikowane na stronie internetowej wykładowcy, m.in.
  - prezentacje z wykładów
    - UWAGA: prezentacja to nie książka, notatki czy skrypt – to raczej streszczenie omawianego materiału, pokazanie wybranych algorytmów, przedstawienie wybranych przykładów – dlatego przejrzenie samych prezentacji czasami może nie wystarczyć do zrozumienia treści całego wykładu
  - listy zadań
  - propozycje minireferatów i minikonkursów
  - materiały dodatkowe rozszerzające treść wykładu
  - ogłoszenia bieżące

Zasady zaliczenia

- Ćwiczenia/pracownie:
  - Będzie można zdobywać punkty za:
    - listy zadań (za 90 punktów łącznie)
      - każda lista zadań będzie dotyczyć pewnego działu eksploracji danych
      - część zadań będzie polegać na zaimplementowaniu pewnych algorytmów i wykorzystaniu ich do analizy przykładowych danych, a część na omówieniu przy tablicy pewnych mechanizmów eksploracji danych
      - na realizację każdej listy zadań będzie określony czas, zwykle 1 lub 2 tygodnie
    - projekt (za 30 punktów)
    - punkty bonusowe za dodatkową aktywność (minireferaty, minikonkursy, itp.)
  - Łącznie będzie do zdobycia minimum 120 punktów (oprócz punktów bonusowych)
  - Na zaliczenie wymagane jest 60 punktów. Progi na poszczególne oceny to:

3.0	60 punktów
3.5	72 punktów
4.0	84 punktów
4.5	96 punktów
5.0	108 punktów

  - Na ocenę bardzo dobrą wymagane jest dodatkowo przygotowanie i wygłoszenie minireferatu.
- Wykład: egzamin

Zasady zaliczenia

- Kilka przykładów:
  - Student, który zdobędzie 40 punktów za listy zadań i 20 punktów za projekt, zaliczy zajęcia z oceną 3.0.
  - Student, który zdobędzie 60 punktów za listy zadań i 0 punktów za projekt, zaliczy zajęcia z oceną 3.0.
  - Student, który zdobędzie 55 punktów za listy zadań, 0 punktów za projekt i 5 punktów za minireferat, zaliczy zajęcia z oceną 3.0.
  - Student, który zdobędzie 85 punktów za listy zadań, 20 punktów za projekt i 5 punktów za minireferat, zaliczy zajęcia z oceną 5.0.
  - Student, który zdobędzie 85 punktów za listy zadań i 25 punktów za projekt, zaliczy zajęcia z oceną 4.5 (na ocenę 5.0 wymagany jest minireferat).
- UWAGA: Projekt może wymagać sporo pracy. Musi zawierać przemyślenie wybranego problemu, opracowanie algorytmu jego rozwiązywania, implementację tego algorytmu, przeprowadzenie eksperymentów obliczeniowych i wykonanie raportu z testowania opracowanego podejścia.

Program wykładu

- Wprowadzenie do problemów optymalizacji
- Wprowadzenie do heurystycznych algorytmów optymalizacji

- Problemy optymalizacji:
  - Klasyczne problemy optymalizacji
  - Wielomodalne problemy optymalizacji
  - Wielokryterialne problemy optymalizacji
  - Dynamiczne problemy optymalizacji

- Algorytmy ewolucyjne:
  - algorytmy genetyczne, strategie ewolucyjne, programowanie genetyczne, programowanie ewolucyjne (m.in. SGA, ES, GP, EP, messyGA, DE)
  - proste algorytmy estymowania rozkładu (m.in. CGA, PBIL, ECGA)
  - algorytmy oparte na sieciach bayesowskich i gaussowskich (EBNA, EGNA, BOA)
  - algorytmy optymalizacji wielokryterialnej (m.in. NSGA, SPEA, MOEA/D)

- Zastosowania:
  - klasyczne problemy optymalizacji (m.in. TSP, QAP, Q3AP)
  - systemy ekspertowe, systemy klasyfikujące, systemy wspomagania decyzji
  - evolutionary robotics, embodied evolution i inne

Klasyczny problem optymalizacji

- Niech $F : \Omega \rightarrow R$ będzie funkcją określoną na zbiorze Ω i przyjmującą wartości rzeczywiste.

- Problem minimalizacji funkcji F na zadanym zbiorze $D \subseteq \Omega$ polega na znalezieniu minimum funkcji F na zbiorze D (i punktu, w którym funkcja to minimum przyjmuje), tzn. na znalezieniu $x_0 \in D$ takiego, że dla każdego $x \in D \setminus \{x_0\}$ zachodzi $F(x_0) < F(x)$.

- Problem maksymalizacji funkcji F na zadanym zbiorze $D \subseteq \Omega$ polega na znalezieniu maksimum funkcji F na zbiorze D (i punktu, w którym funkcja to maksimum przyjmuje), tzn. na znalezieniu $x_0 \in D$ takiego, że dla każdego $x \in D \setminus \{x_0\}$ zachodzi $F(x) < F(x_0)$.

- Każdy problem minimalizacji może zostać przekształcony do problemu maksymalizacji i każdy problem maksymalizacji może zostać przekształcony do problemu minimalizacji.

- UWAGI:
  - Zbiór Ω nazywamy **przestrzenią poszukiwań**.
  - Funkcję F nazywamy **funkcją celu**.
  - Zbiór D nazywamy **zbiorem rozwiązań dopuszczalnych**.

  - W praktyce, często rozszerza się problem zastępując nierówności ostre nieostrymi i starając się znaleźć wszystkie punkty x_0 spełniające powstały warunek.

  - W praktyce, często nie jest ważne znalezienie dokładnych punktów minimów czy maksimów funkcji celu, a jedynie ich przybliżeń lub punktów, w których funkcja celu przyjmuje wartości "dostatecznie niskie" lub "dostatecznie wysokie" z praktycznego punktu widzenia.

Klasyczny problem optymalizacji

- Często zakłada się, że przestrzeń poszukiwań Ω jest przestrzenią metryczną z pewną metryką ρ. Wówczas można definiować pojęcie otoczenia punktu, na przykład otoczenie punktu x_0 o promieniu $\varepsilon > 0$ można określić jako $S = \{x : \rho(x_0, x) < \varepsilon\}$.
- Mówimy, że funkcja F przyjmuje minimum lokalne w punkcie $x_0 \in \Omega$, jeśli istnieje otoczenie S punktu x_0 takie, że dla każdego $x \in S \setminus \{x_0\}$ zachodzi $F(x_0) < F(x)$.
- Mówimy, że funkcja F przyjmuje maksimum lokalne w punkcie $x_0 \in \Omega$, jeśli istnieje otoczenie S punktu x_0 takie, że dla każdego $x \in S \setminus \{x_0\}$ zachodzi $F(x) < F(x_0)$.

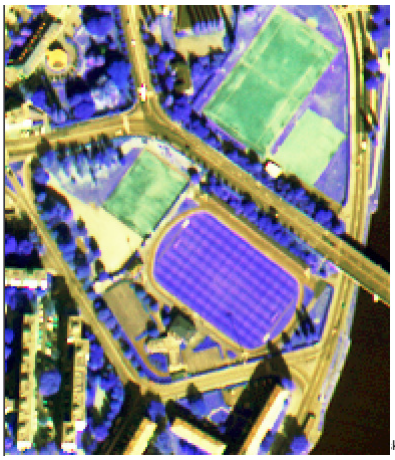
Klasyczny problem optymalizacji

- Przykład 1 (funkcje analityczne):
  - Znaleźć minimum funkcji $F(x) = 3x^2 – 2x + 1$.
  - Znaleźć minimum funkcji $F(x, y) = x^{17} y^{13} – 2x^{11} + 3y^7 + 5$.
  - Znaleźć minimum funkcji F(x), określonej jak wyżej, ale rozwiązaniami dopuszczalnymi są jedynie liczby całkowite.
- Przykład 2 (problemy kombinatoryczne):
  - Problem komiwojażera.
  - Problem QAP.
  - Problem Q3AP.
- Przykład 3 (funkcje "obliczeniowe" / Black Box Optimization):
  - W praktyce często mamy do czynienia z funkcjami, których dokładne matematyczne definicje i wzory nie są znane. Funkcje takie są obliczane w zadany sposób, na przykład w procesie symulacji pewnych zjawisk. Cała nasza wiedza o takiej funkcji to znajomość procedury obliczania jej wartości dla zadanego argumentu.
- Przykład 4:
  - Funkcje pochodzące z systemów uczenia maszynowego: systemów ekspertowych, systemów klasyfikujących, systemów wspomagania decyzji.
- Przykład 5:
  - Funkcje określone na drzewach, grafach czy innych złożonych obiektach.

Przykład – klasyfikacja wielospektralnych obrazów satelitarnych

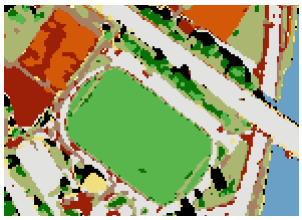
Wielospektralny obraz satelitarny:

- obraz wielospektralny składa się z pewnej liczby n warstw (zazwyczaj n jest duże, np. $n = 115$) (analogicznie jak obraz RGB składa się z 3 warstw: R, G i B)
- każdy punkt obrazu q jest więc opisany przez n parametrów: $q_1, q_2, \ldots, q_n$ (parametry te mogą być liczbami jednobajtowymi lub ogólniej liczbami rzeczywistymi)

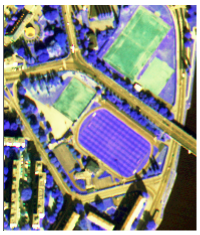


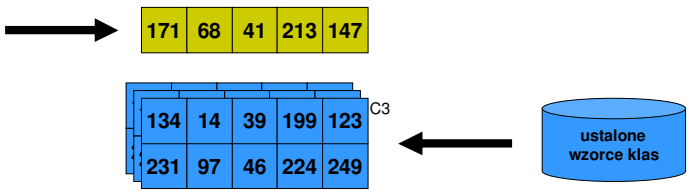
Klasyfikacja:

- na obrazie chcemy rozpoznawać pewne klasy punktów, na przykład odpowiadające budynkom, drogom, trawnikom, zbiornikom wodnym, itp.
- pewien fragment obrazu został wcześniej wzorcowo sklasyfikowany, możemy użyć go do utworzenia/nauczenia systemu automatycznej klasyfikacji
- niech K oznacza liczbę rozważanych klas, zaś C1, C2, …, CK będą etykietami tych klas



Przykład – klasyfikacja wielospektralnych obrazów satelitarnych





punkt obrazu jest zgodny ze wzorcem klasy C3
(parametry zawierają się w odpowiednich przedziałach)

Przykład – klasyfikacja wielospektralnych obrazów satelitarnych

Proponowane podejście:

- Dla zadanego punktu q, klasyfikator będzie kolejno określał czy punkt q należy do klasy C1, klasy C2, …, klasy CK. Jeśli punkt będzie należał do więcej niż jednej klasy, to klasyfikator zwróci błąd (punkt nieokreślony).
- Klasyfikator będzie określał czy punkt q należy do klasy C przez sprawdzenie warunków postaci:

$$q_1 \in [a_1^{(C)}, b_1^{(C)}] \wedge q_2 \in [a_2^{(C)}, b_2^{(C)}] \wedge \ldots \wedge q_n \in [a_n^{(C)}, b_n^{(C)}],$$

gdzie $a_1^{(C)}, a_2^{(C)}, \ldots, a_n^{(C)}, b_1^{(C)}, b_2^{(C)}, \ldots, b_n^{(C)}$ to ustalone parametry określające klasę C.

Problem: Jak wyznaczyć właściwe parametry określające klasy?

Można sprowadzić to do problemu optymalizacji:

- Przestrzeń poszukiwań:
  - Zbiór wszystkich możliwych konfiguracji systemu (ciągów parametrów), tzn. zbiór wektorów liczb rzeczywistych o długości 2 n K.
- Funkcja celu:
  - argumentem funkcji celu jest ciąg parametrów
    $[a_1^{(C1)}, a_2^{(C1)}, \ldots, a_n^{(C1)}, b_1^{(C1)}, b_2^{(C1)}, \ldots, b_n^{(C1)},$
    $a_1^{(C2)}, a_2^{(C2)}, \ldots, a_n^{(C2)}, b_1^{(C2)}, b_2^{(C2)}, \ldots, b_n^{(C2)}, \ldots,$
    $a_1^{(CK)}, a_2^{(CK)}, \ldots, a_n^{(CK)}, b_1^{(CK)}, b_2^{(CK)}, \ldots, b_n^{(CK)}]$
  - wartość funkcji celu jest obliczana przez uruchomienie klasyfikatora z zadaną przez argument funkcji celu konfiguracją systemu (ciągiem parametrów) dla danych uczących (fragmentu obrazu wzorcowo sklasyfikowanego) i policzenie liczby błędnie sklasyfikowanych punktów obrazu
- Wyznaczenie właściwej konfiguracji systemu polega więc na znalezieniu minimum funkcji celu na przestrzeni poszukiwań.

Przykład – problem QAP

Quadratic Assignment Problem (QAP):

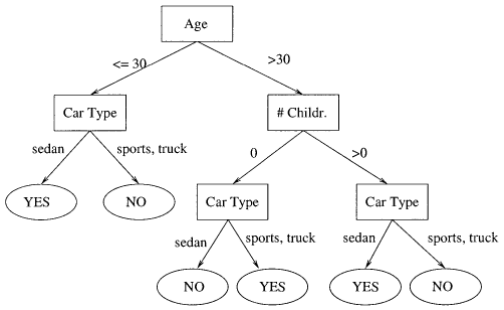
- rozpatrujemy n miejscowości, w których należy zbudować n fabryk (w każdej miejscowości ma powstać dokładnie jedna fabryka)
- **przypisanie** fabryk do miejscowości (rozmieszczenie fabryk) to permutacja $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ liczb 1, 2, …, n, taka że p_k oznacza numer miejscowości, w której zostanie zbudowana fabryka numer k
- dana jest **macierz odległości** między miejscowościami – macierz D rozmiaru n x n, o elementach d(i, j) oznaczających odległość miejscowości i od miejscowości j
- dana jest **macierz przepływu** towarów między fabrykami – macierz F rozmiaru n x n, o elementach f(k, l) oznaczających wartość przepływu towarów z fabryki k do fabryki l
- należy znaleźć takie rozmieszczenie fabryk **p**, żeby łączny koszt przepływu towarów c(**p**) był minimalny:

$$c(\mathbf{p}) = \sum_{k=1}^{n} \sum_{l=1}^{n} f(k,l) d(p_k, p_l)$$

Przykład – programowanie genetyczne

- Przestrzenią poszukiwań może być zbiór drzew (na przykład drzew klasyfikujących). Funkcja celu jest wówczas określona na tych drzewach (na przykład jako sprawność klasyfikatora liczona na pewnych danych uczących). Zagadnienie optymalizacji polega na znalezieniu maksimum funkcji celu, czyli znalezieniu drzewa spełniającego pewne kryteria (na przykład najlepiej klasyfikującego dane uczące).
- Programowanie genetyczne to algorytmy zajmujące się problemami optymalizacji określonymi na zbiorach drzew.

Record Id	Car	Age	Children	Subscription
1	sedan	23	0	yes
2	sports	31	1	no
3	sedan	36	1	no
4	truck	25	2	no
5	sports	30	0	no
6	sedan	36	0	no
7	sedan	25	0	yes
8	truck	36	1	no
9	sedan	30	2	yes
10	sedan	31	1	yes
11	sports	25	0	no
12	sedan	45	1	yes
13	sports	23	2	no
14	truck	45	0	yes



Klasyczny problem optymalizacji

- Popularne metody rozwiązywania:
  - analiza matematyczna (funkcji jednej i wielu zmiennych)
  - programowanie liniowe i kwadratowe
  - algorytm simpleks (dla wypukłych problemów optymalizacji)
  - algorytmy aproksymacyjne
  - algorytmy przeszukiwania
    - algorytm pełnego przeglądu
    - algorytmy heurystyczne
      - Simulated Annealing
      - Tabu Search
      - Hill Climbing
      - algorytmy przeszukiwania populacyjnego, w tym algorytmy ewolucyjne
- Dla wielu praktycznych problemów algorytmy optymalizacji gwarantujące poprawność rozwiązania nie mogą zostać zastosowane. Algorytmy aproksymacyjne, nawet jeśli mogą być zastosowane do takich problemów, to mimo że gwarantują określoną dokładność rozwiązania, często dostarczają rozwiązań zbyt słabych. Prowadzi to do konieczności stosowania heurystycznych algorytmów przeszukiwania.

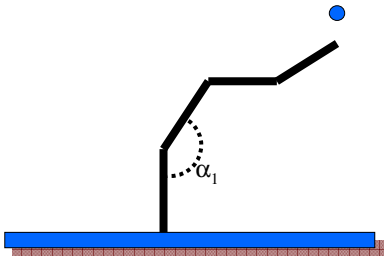
Uogólnione problemy optymalizacji

- Wielomodalny problem optymalizacji
  - Celem nie jest znalezienie maksimum globalnego funkcji celu, ale określonej liczby największych maksimów lokalnych funkcji celu.
  - Przykład: projektowanie kształtów spełniających pewne kryteria (funkcja celu mierzy stopień spełniania tych kryteriów), użytkownik jednak chciałby mieć możliwość wybrania rozwiązanie gorszego (o niższej wartości funkcji celu), ale "ładniejszego" (a tego kryterium użytkownik nie potrafi sprecyzować).
- Wielokryterialny problem optymalizacji
  - Zamiast jednej funkcji celu jest ich kilka, często ze sobą sprzecznych. Celem jest znalezienie rozwiązania kompromisowego, optymalizującego kilka sprzecznych ze sobą kryteriów.
  - Przykład: maksymalizacja oczekiwanego zysku z inwestycji przy jednoczesnej minimalizacji ryzyka.
- Dynamiczny problem optymalizacji
  - Funkcja celu zmienia się w czasie. Zmiany są na tyle częste, że nie można rozpatrywać tego problemu jako ciągu statycznych problemów optymalizacji, ale należy w algorytmie optymalizacji uwzględnić fakt, że funkcji celu zmienia się w trakcie jego wykonywania.
  - Przykład: system wspomagania decyzji dla środowisk dynamicznych, m.in. opartych na szeregach czasowych wysokiej częstotliwości.

Przykład – problem Inverse Kinematics

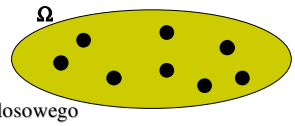
Inverse Kinematics:

- Dane jest ramię robota, przytwierdzone do podłoża, składające się z kilku segmentów o ustalonej długości, które może się wyginać (może zmieniać się kąt między segmentami).
- Celem jest takie ustawienie ramienia robota (określenie wartości kątów między segmentami), żeby koniec ramienia dosięgał podanego punktu docelowego.
- Problem rozpatrujemy w dwóch (łatwiej) lub trzech (trudniej) wymiarach.
- Problem można skomplikować:
  - dodając ograniczenia na zakres wartości każdego kąta (segmenty nie mogą przekręcać się o 360 stopni),
  - wprowadzając przeszkody, które ramię musi ominąć,
  - rozpatrując problem w aspekcie dynamicznym (punkt docelowy lub przeszkody mogą się poruszać).
- Problem taki można sformułować jako problem optymalizacji i rozwiązywać algorytmami ewolucyjnymi.

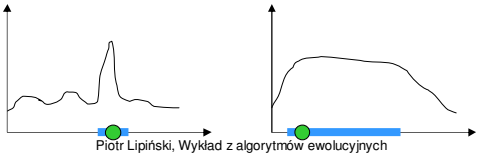


Heurystyczne algorytmy przeszukiwania

- *Dla wielu praktycznych problemów algorytmy optymalizacji gwarantujące poprawność rozwiązania nie mogą zostać zastosowane. Algorytmy aproksymacyjne, nawet jeśli mogą być zastosowane do takich problemów, to mimo że gwarantują określoną dokładność rozwiązania, często dostarczają rozwiązań zbyt słabych. Prowadzi to do konieczności stosowania heurystycznych algorytmów przeszukiwania.*
- Podejścia naiwne:
  - algorytm pełnego przeglądu
    - algorytm polega na przejrzeniu całej przestrzeni poszukiwań – policzeniu wartości funkcji celu dla każdego elementu przestrzeni poszukiwań i wyboru minimum/maksimum
    - zazwyczaj niestosowalny ze względu na wielkość przestrzeni poszukiwań (zbyt duży czas obliczeń)

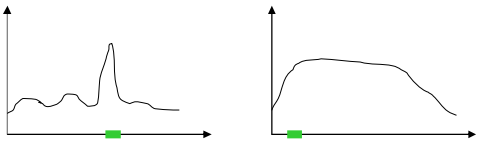


  - algorytm przeglądu losowego
    - algorytm polega na wylosowaniu kolejno, z jednostajnym rozkładem prawdopodobieństwa, ustalonej liczby elementów przestrzeni poszukiwań, policzeniu wartości funkcji celu dla każdego z nich i wyboru minimum/maksimum
    - prawdopodobieństwo, że algorytm zwróci prawdziwe optimum funkcji celu (lub jego przybliżenie o zadanej dokładności) zależy od liczby losowanych elementów oraz od trudności problemu



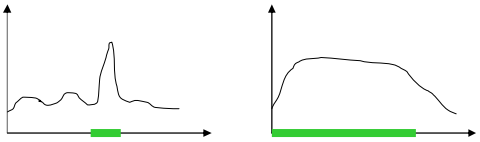
Heurystyczne algorytmy przeszukiwania

- Podejścia nieco usprawnione:
  - algorytm przeglądu losowego z pewnym usprawnieniem
    - algorytm losuje z jednostajnym rozkładem prawdopodobieństwa element x przestrzeni poszukiwań
    - algorytm wyznacza sąsiedztwo wylosowanego elementu x o ustalonym promieniu, sprawdza wszystkie elementy y tego sąsiedztwa, wybiera z nich minimum/maksimum i zapamiętuje je
    - powyższe dwa kroki algorytm powtarza ustaloną liczbę iteracji
    - algorytm wybiera minimum/maksimum z zapamiętanych elementów
    - prawdopodobieństwo, że algorytm zwróci prawdziwe optimum funkcji celu (lub jego przybliżenie o zadanej dokładności) jest większe niż w poprzednim przypadku
    - UWAGA: potrzebne jest określenie relacji sąsiedztwa na przestrzeni poszukiwań



Heurystyczne algorytmy przeszukiwania

- Podejścia nieco usprawnione:
  - algorytm przeglądu losowego z większym usprawnieniem (zwanym Hill Climbing)
    - algorytm losuje z jednostajnym rozkładem prawdopodobieństwa element x przestrzeni poszukiwań
    - algorytm wyznacza sąsiedztwo wylosowanego elementu x o ustalonym promieniu, sprawdza wszystkie elementy y tego sąsiedztwa, wybiera z nich minimum/maksimum i zastępuje nim element x
    - powyższy krok algorytm powtarza aż element x przestanie się zmieniać i zapamiętuje go
    - powyższe trzy kroki algorytm powtarza ustaloną liczbę iteracji
    - algorytm wybiera minimum/maksimum z zapamiętanych elementów
    - prawdopodobieństwo, że algorytm zwróci prawdziwe optimum funkcji celu (lub jego przybliżenie o zadanej dokładności) jest większe niż w poprzednim przypadku
    - UWAGA: potrzebne jest określenie relacji sąsiedztwa na przestrzeni poszukiwań



- Kilka uwag:
  - Rozpatrywane sąsiedztwa muszą być na tyle małe, aby dało się przejrzeć wszystkie ich elementy w krótkim czasie.
  - Jeśli rozpatrywane sąsiedztwa są zbyt duże, można zmodyfikować algorytm – zamiast pełnego przeglądu sąsiedztw, można zastosować przegląd losowy.
  - Hill Climbing będzie utykał w optimach lokalnych, niekoniecznie globalnych.

Heurystyczne algorytmy przeszukiwania

- Algorytm symulowanego wyżarzania (ang. Simulated Annealing)
  - minimalizacja funkcji celu $F : \Omega \rightarrow R$ określonej na przestrzeni poszukiwań Ω, na której można określić relację sąsiedztwa (m.in. przestrzenie metryczne)
- Parametry:
  - NumberOfIterations – liczba iteracji algorytmu
  - T – ustalona nierosnąca funkcja (reprezentująca "temperaturę" wyżarzania)
- Algorytm

```
x = Initial-Random-Solution()
t = 0
while t < NumberOfIterations
    y = Random-Neighbour(x)
    if F(y) ≤ F(x) then
        x = y
    if F(y) > F(x) then
        if (exp((F(x)-F(y)) / T(t)) < UniformRandom(0,1)) then
            x = y
    t = t + 1
return x
```

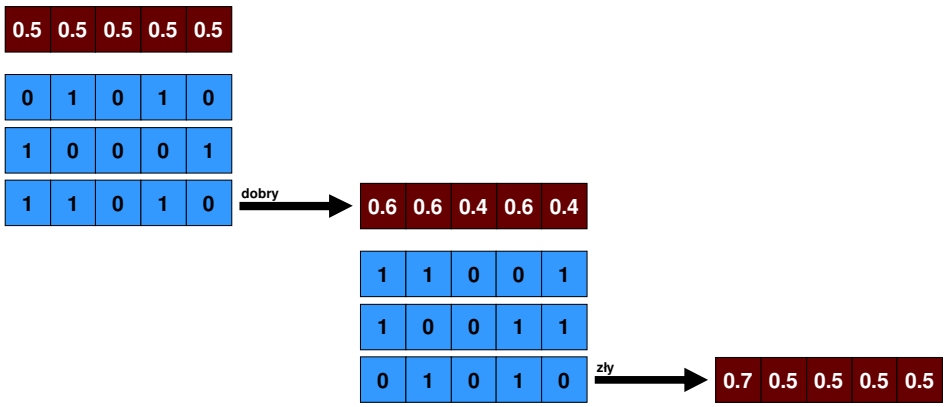
Heurystyczne algorytmy przeszukiwania

- Algorytm przeszukiwania z tabu (ang. Tabu Search)
  - idea polega na ograniczeniu powtórnego rozpatrywania tych samych elementów przestrzeni poszukiwań:
    - algorytm tworzy listę tabu ostatnio sprawdzanych elementów przestrzeni poszukiwań
    - po każdym wylosowaniu sąsiada y elementu x, sprawdza czy wylosowany element y jest na liście tabu
    - jeśli element y jest na liście tabu, to algorytm powtarza losowanie (wybiera innego sąsiada elementu x)
    - jeśli element y nie jest na liście tabu, to zapisuje go tam (usuwając najstarszy element z tej listy) i przetwarza go zgodnie ze standardowym algorytmem
  - szczegóły – dobry pomysł na minireferat

Heurystyczne algorytmy przeszukiwania

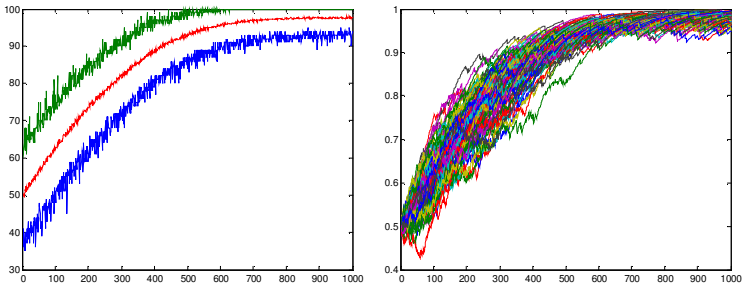
- Rozważane dotąd podejścia nie starały się zdobywać żadnej wiedzy o problemie.
  - Czy wylosowane już elementy nie mogłyby dostarczyć nam jakiejś wiedzy, którą można byłoby wykorzystać przy losowaniu następnych?
  - Czy losowanie z jednostajnym rozkładem prawdopodobieństwa jest najlepszym pomysłem?
- Przykład:
  - Rozpatrzmy przestrzeń poszukiwań $\Omega = \{0, 1\}^n$ i funkcję celu F określoną na niej.
  - Początkowo, element przestrzeni poszukiwań jest losowany z jednostajnym rozkładem prawdopodobieństwa (tzn. prawdopodobieństwo wylosowania każdego wektora binarnego o długości n jest takie samo i wynosi $1/2^n$). Inaczej mówiąc, wektor binarny jest losowany bit po bicie, a dla każdego bitu, prawdopodobieństwo wylosowania 0 i 1 jest takie samo i wynosi 1/2.
  - W kolejnych iteracjach, prawdopodobieństwa wylosowania 0 i 1 dla różnych bitów zmieniają się na podstawie wiedzy uzyskanej z wcześniej wylosowanych elementów:
    - jeśli wylosowany element był dobry (np. lepszy od dotychczas najlepszego) i jeśli miał 1 na pewnym miejscu, to prawdopodobieństwo wylosowania 1 na tym miejscu zostaje zwiększone, a jeśli miał 0 na pewnym miejscu, to prawdopodobieństwo wylosowania 1 na tym miejscu zostaje zmniejszone.
    - jeśli wylosowany element był zły (np. gorszy od dotychczas najgorszego), to analogicznie
- Algorytm CGA
- Algorytm PBIL

Ilustracja



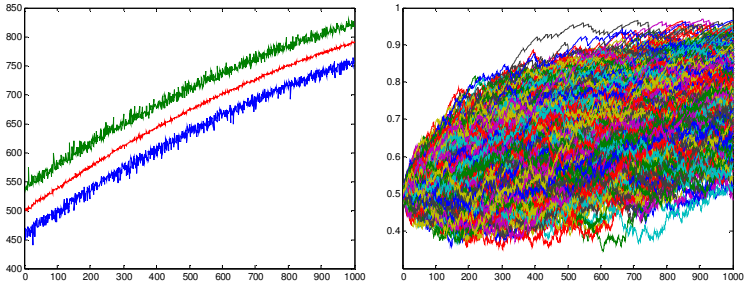
Ilustracja działania PBIL

- PBIL na OneMax o długości d = 100
  - Przestrzeń poszukiwań to wektory binarne długości d = 100.
  - Funkcja celu OneMax zwraca liczbę jedynek w ciągu binarnym. Jej maksimum to wektor samych jedynek.
  - T = 1000 iteracji
  - N = 125 osobników
  - $\theta_1 = 0.01, \theta_2 = 0.05, \theta_3 = 0.01$
- Algorytm znalazł rozwiązanie problemu po ok. 500 iteracjach.
- Algorytm skonstruował w miarę dobry model probabilistyczny po ok. 800 iteracjach.
- UWAGA: W końcowym modelu probabilistycznym nie wszystkie prawdopodobieństwa są równe 1. Czy to dobrze czy źle?



Ilustracja działania PBIL

- PBIL na OneMax o długości d = 1000
  - Przestrzeń poszukiwań to wektory binarne długości d = 1000.
  - Funkcja celu OneMax zwraca liczbę jedynek w ciągu binarnym. Jej maksimum to wektor samych jedynek.
  - T = 1000 iteracji
  - N = 125 osobników
  - $\theta_1 = 0.01$, $\theta_2 = 0.05$, $\theta_3 = 0.01$
- Algorytm nie znalazł rozwiązania problemu (potrzeba więcej iteracji lub zmiany parametrów uczenia).
- Algorytm nie skonstruował też właściwego modelu probabilistycznego.



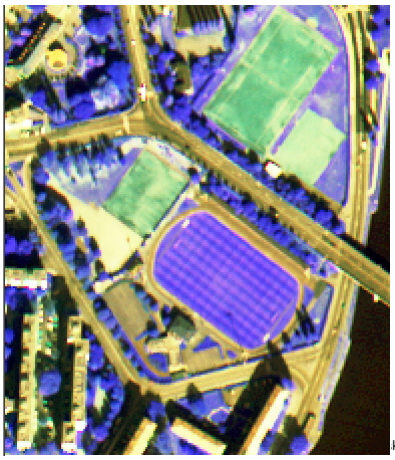
Ilustracja działania PBIL

- Rozpatrywać będziemy uproszczony problem klasyfikacji wielospektralnych obrazów satelitarnych:
  - interesują nas tylko 3 klasy punktów, które chcemy rozpoznawać na obrazie (pozostałe punkty nas nie interesują),
  - dostępne są utworzone wcześniej reguły klasyfikujące,
  - algorytmu PBIL użyjemy do wyboru optymalnego zestawu reguł klasyfikujących.
- Dany jest obraz wielospektralny o 3 spektrach i rozmiarze 106 x 148 punktów.
  - Każdy punkt obrazu można przedstawić jako wektor rozmiaru 3 x 1, a więc cały obraz można przedstawić jako macierz rozmiaru 3 x 15688.
- Dana jest wzorcowa klasyfikacja tego obrazu.
  - Jest to bitmapa o 11 kolorach i rozmiarze 106 x 148 punktów.
  - Można przedstawić ją jako wektor rozmiaru 1 x 15688 o wartościach ze zbioru {1, 2, …, 11}.
- Interesują nas tylko 3 klasy punktów, pozostałe punkty więc usuwamy z danych, otrzymując:
  - macierz X rozmiaru 3 x 9350 (punkty obrazu wielospektralnego),
  - wektor C rozmiaru 1 x 9350 o wartościach ze zbioru {1, 2, 3} (wzorcowa klasyfikacja).
- Dostępnych jest 266 utworzonych wcześniej reguł klasyfikujących.
  - Każda reguła to określona funkcja, która na wejściu dostaje punkt obrazu (wektor rozmiaru 3 x 1), a na wyjściu zwraca etykietę klasy (liczbę ze zbioru {1, 2, 3}).
  - Reguły te zostały utworzone algorytmem programowania genetycznego (będziemy o nim mówić w dalszej części wykładu).
  - Dla wygody dostępne są już policzone wartości każdej reguły klasyfikującej dla każdego punktu obrazu, a nie same definicje tych reguł.
- Dla danego punktu obrazu, każda reguła zwraca proponowaną etykietę klasy. Ostateczna decyzja o klasie punktu może zostać podjęta "większością głosów" wszystkich reguł klasyfikujących.
- Czy warto jednak słuchać wszystkich reguł klasyfikujących? Może wybrać podzbiór zbioru wszystkich reguł i słuchać tylko jego? Jak znaleźć optymalny zestaw reguł klasyfikujących?

Przykład – klasyfikacja wielospektralnych obrazów satelitarnych

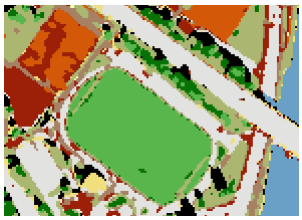
Wielospektralny obraz satelitarny:

- obraz wielospektralny składa się z pewnej liczby n warstw (zazwyczaj n jest duże, np. $n = 115$) (analogicznie jak obraz RGB składa się z 3 warstw: R, G i B)
- każdy punkt obrazu q jest więc opisany przez n parametrów: $q_1, q_2, \ldots, q_n$ (parametry te mogą być liczbami jednobajtowymi lub ogólniej liczbami rzeczywistymi)



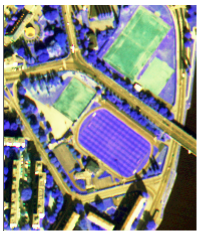
Klasyfikacja:

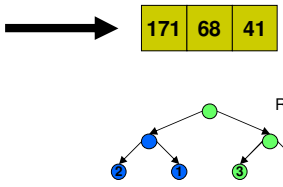
- na obrazie chcemy rozpoznawać pewne klasy punktów, na przykład odpowiadające budynkom, drogom, trawnikom, zbiornikom wodnym, itp.
- pewien fragment obrazu został wcześniej wzorcowo sklasyfikowany, możemy użyć go do utworzenia/nauczenia systemu automatycznej klasyfikacji
- niech K oznacza liczbę rozważanych klas, zaś C1, C2, …, CK będą etykietami tych klas



ki, Wykład z algorytmów ewolucyjnych

Przykład – klasyfikacja wielospektralnych obrazów satelitarnych





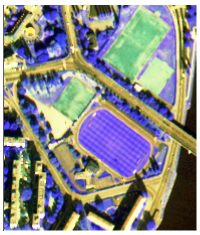
reguła klasyfikująca nr R113 uznała, że punkt obrazu jest klasy C3

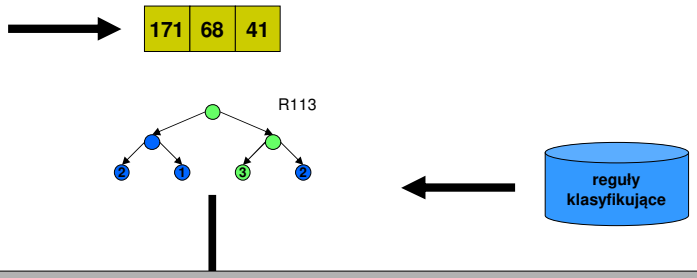
wyniki pozostałych reguł:

3	1	2	1	2	2	3	3	1	3	3	1	3	2	3	3	3	1	…	3

ostateczna decyzja o klasie punktu może być podjęta "większością głosów" wszystkich reguł klasyfikujących

Przykład – klasyfikacja wielospektralnych obrazów satelitarnych





wyniki pozostałych reguł:

3	1	2	1	2	2	3	3	1	3	3	1	3	2	3	3	3	1	…	3

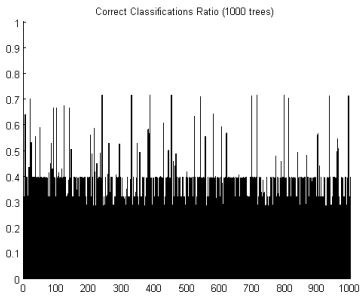
ostateczna decyzja o klasie punktu może być podjęta "większością głosów" wszystkich reguł klasyfikujących

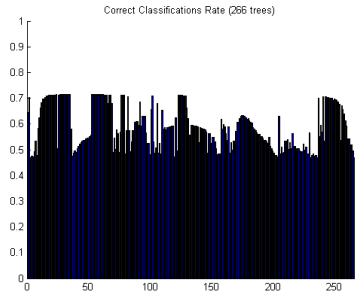
jeśli rozpatrywalibyśmy jedynie pewien podzbiór zbioru wszystkich reguł klasyfikujących, decyzja mogłaby być inna

3	1	2	1	2	2	3	3	1	3	3	1	3	2	3	3	3	1	…	3

Ilustracja działania PBIL

- Decyzja oparta na pojedynczej regule klasyfikującej daje dość słabe wyniki
  - w najlepszych przypadkach poprawnie poklasyfikowanych jest ok. 70% punktów obrazu

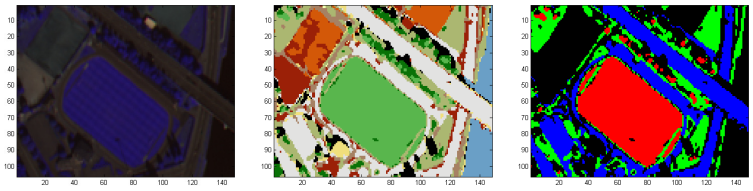




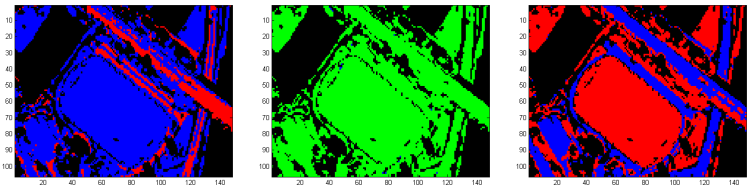
- Decyzja oparta na wszystkich 266 regułach klasyfikujących ("większością głosów") daje tylko 64.53% poprawnie poklasyfikowanych punktów obrazu.
- Znajdując odpowiedni podzbiór reguł klasyfikujących i opierając na nim decyzję można uzyskać nawet ponad 98% poprawnie poklasyfikowanych punktów obrazu.

Ilustracja działania PBIL

- Dane wejściowe: oryginalny obraz wielospektralny, oryginalna klasyfikacja wzorcowa, uproszczona klasyfikacja wzorcowa (tylko 3 klasy punktów).

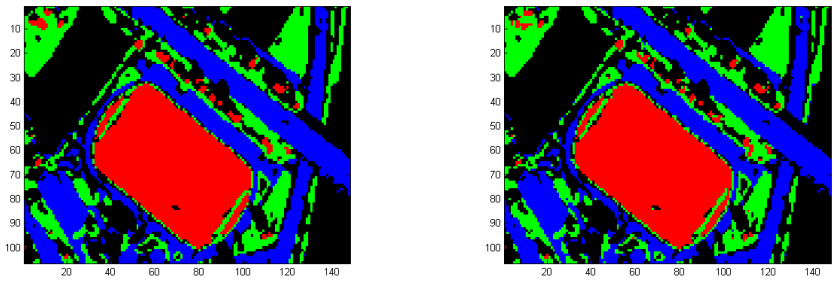


- Przykłady obrazów uzyskiwanych w klasyfikacji opartej na pojedynczej regule klasyfikującej.



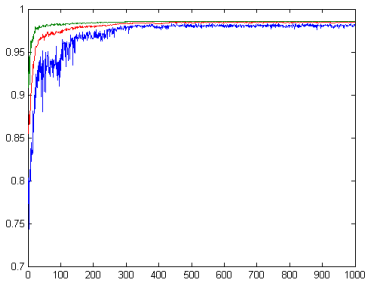
Ilustracja działania PBIL

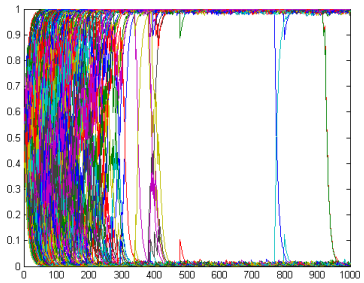
- Po lewej: klasyfikacja wzorcowa.
- Po prawej: obraz uzyskany w klasyfikacji opartej na zbiorze reguł klasyfikujących skonstruowanym algorytmem PBIL (ponad 98% poprawnie poklasyfikowanych punktów obrazu).



Ilustracja działania PBIL

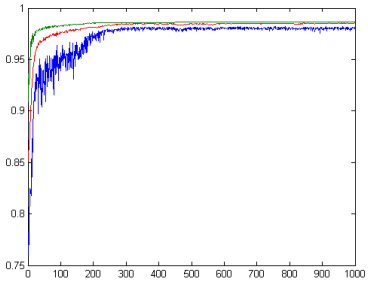
- PBIL
  - Przestrzeń poszukiwań to wektory binarne długości d = 266.
  - Funkcja celu zwraca frakcję poprawnie poklasyfikowanych punktów (zakres wartości od 0.00 do 1.00).
  - T = 1000 iteracji
  - N = 50 osobników
  - czas obliczeń: ok. 2h
  - najlepsze znalezione rozwiązanie: 0.985775

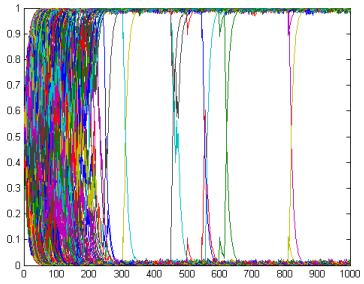




Ilustracja działania PBIL

- PBIL
  - Przestrzeń poszukiwań to wektory binarne długości d = 266.
  - Funkcja celu zwraca frakcję poprawnie poklasyfikowanych punktów (zakres wartości od 0.00 do 1.00).
  - T = 1000 iteracji
  - N = 100 osobników
  - czas obliczeń: ok. 4h
  - najlepsze znalezione rozwiązanie: 0.986631





Heurystyczne algorytmy przeszukiwania

- Czy można prosto uogólnić takie podejście na przypadek ciągły (tzn. funkcję celu określoną na wektorach liczb rzeczywistych) ?
  - dyskretny rozkład prawdopodobieństwa należy zastąpić ciągłym
  - potrzebne są jeszcze procedury uaktualniania parametrów rozkładu
  - przykład: algorytm PBILc (będzie później)

- Proponowane podejścia prowadzą do algorytmów estymowania rozkładów (ang. Estimation of Distribution Algorithms).
  - Jest to nowoczesna gałąź algorytmów ewolucyjnych, których celem jest nie tyle znalezienie rozwiązania problemu optymalizacji (elementu przestrzeni poszukiwań), co skonstruowanie rozkładu prawdopodobieństwa, który pozwala z dużym prawdopodobieństwem wylosować takie rozwiązanie.

- Czy w praktyce większą wartość ma
  - znalezienie rozwiązania problemu optymalizacji (elementu przestrzeni poszukiwań)

  czy
  - skonstruowanie rozkładu prawdopodobieństwa, który pozwala z dużym prawdopodobieństwem wylosować takie rozwiązanie ?

Parę słów o rozkładzie jednostajnym

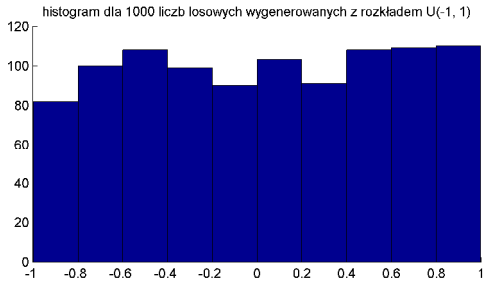
- gęstość rozkładu jednostajnego U(α, β) na odcinku [α, β] to

$$f(x;\alpha,\beta)=\begin{cases}\dfrac{1}{\beta-\alpha}, & \text{dla } x\in[\alpha,\beta]\\ 0, & \text{dla } x\notin[\alpha,\beta]\end{cases}$$

- wykresem przykładowej gęstości jest poniższa krzywa

Parę słów o rozkładzie jednostajnym

- generując liczby losowe z rozkładem U(α, β) powinniśmy dostawać histogramy zgodne z odpowiednią krzywą gęstości
- przykład: histogram dla 1000 liczb losowych wygenerowanych z rozkładem U(α, β)

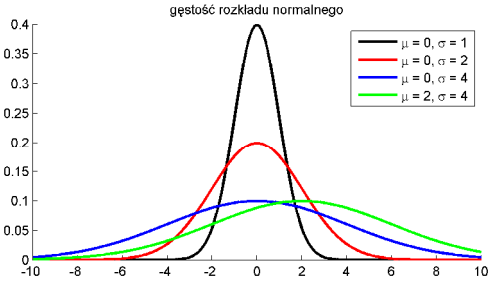


Parę słów o rozkładzie normalnym

- gęstość rozkładu normalnego $N(\mu, \sigma^2)$ o wartości oczekiwanej μ i wariancji σ^2 to

$$f(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- wykresem gęstości jest krzywa Gaussa, wartość oczekiwana wpływa na przesunięcie krzywej, a wariancja na jej kształt



Parę słów o rozkładzie normalnym

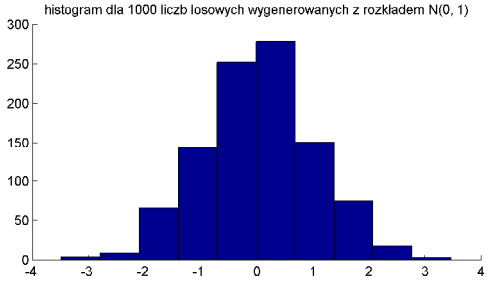
- prawdopodobieństwo, że zmienna losowa $X \sim N(\mu, \sigma^2)$ przyjmie wartość z pewnego przedziału [a, b], to pole powierzchni pod wykresem krzywej Gaussa na odcinku [a, b]
- prawdopodobieństwa takie można wyliczyć numerycznie lub odczytać z tablic statystycznych
- można więc precyzyjnie odpowiadać na pytania:
  - jakie jest prawdopodobieństwo, że wartości zmiennej losowej X będą w przedziale [a, b] ?
  - jakie jest prawdopodobieństwo, że wartości zmiennej losowej X przekroczą zadany próg a ?
  - w jakim przedziale będą wartości X z prawdopodobieństwem 95% ?
- zakładając więc, że zaburzenie analizowanych danych ma określony rozkład, można precyzyjnie szacować dokładność obliczeń podczas dalszego przetwarzania danych

Parę słów o rozkładzie normalnym

- generując liczby losowe z rozkładem $N(\mu, \sigma^2)$ powinniśmy dostawać histogramy zgodne z odpowiednią krzywą Gaussa

  UWAGA: Jest kilka popularnych algorytmów generowania liczb pseudolosowych z rozkładem normalnym (zazwyczaj są one już zaimplementowane w popularnych narzędziach programistycznych), m.in. algorytm Boxa-Mullera czy algorytm Ziggurata.

- przykład: histogram dla 1000 liczb losowych wygenerowanych z rozkładem N(0, 1)



Heurystyczne algorytmy przeszukiwania

- Czy można prosto uogólnić takie podejście na przypadek ciągły (tzn. funkcję celu określoną na wektorach liczb rzeczywistych) ?
  - dyskretny rozkład prawdopodobieństwa należy zastąpić ciągłym
  - potrzebne są jeszcze procedury uaktualniania parametrów rozkładu
  - przykład: algorytm PBILc (będzie później)

- Proponowane podejścia prowadzą do algorytmów estymowania rozkładów (ang. Estimation of Distribution Algorithms).
  - Jest to nowoczesna gałąź algorytmów ewolucyjnych, których celem jest nie tyle znalezienie rozwiązania problemu optymalizacji (elementu przestrzeni poszukiwań), co skonstruowanie rozkładu prawdopodobieństwa, który pozwala z dużym prawdopodobieństwem wylosować takie rozwiązanie.

- Czy w praktyce większą wartość ma
  - znalezienie rozwiązania problemu optymalizacji (elementu przestrzeni poszukiwań)

  czy
  - skonstruowanie rozkładu prawdopodobieństwa, który pozwala z dużym prawdopodobieństwem wylosować takie rozwiązanie ?