

Wstęp do inteligencji komputerowej – zajęcia nr 6

Jarosław Stańczak

WSISiZ

Heurystyki w rozwiązywaniu problemów NP i optymalizacji funkcji wielomodalnych

- algorytmy ewolucyjne
- metoda tabu-search
- algorytmy mrówkowe.

Algorytmy ewolucyjne

początki

- Algorytmy ewolucyjne zostały wymyślone równolegle w kilku liczących się ośrodkach naukowych w różnych krajach (Niemcy, USA) ok. 50 lat temu.
- Wszystkie w różnym stopniu zainspirowane były teorią ewolucji i w różny sposób starały się uchwycić jej zasady.
- Mimo, że różniły się zarówno rozwiązywanymi problemami, jak i sposobem zapisu i samym przebiegiem działania, to jednak po pewnym czasie zauważono ich wspólne korzenie i wszystkie zostały zaklasyfikowane jako metody/algorytmy ewolucyjne.
- Te wspólne cechy algorytmów to:
 - występowanie populacji rozwiązań;
 - użycie selekcji, promującej lepsze rozwiązania do dalszych obliczeń, a odrzucających gorsze (na wzór selekcji naturalnej);
 - wykorzystywanie różnych metod zakodowania problemu (rozwiązań), zamiast rozwiązywania go bezpośrednio;
 - użycie operatorów genetycznych, czyli metod modyfikujących rozwiązania, najczęściej losowo (mutacja) lub przez losowe krzyżowanie cech rozwiązań rodzicielskich (nie wszystkie metody ewolucyjne używały krzyżowania);
 - otrzymanie dobrych rozwiązań wymaga wielu iteracji (pokoleń, generacji, epok) podczas których wymienione wyżej metody oddziałują na populację.

Algorytmy ewolucyjne

podział metod

Nazwy i podział algorytmów ewolucyjnych nie jest jeszcze w pełni ustalony, jednak najczęściej można się spotkać z następującymi kategoriami algorytmów:

- algorytmy genetyczne, charakteryzują się zastosowaniem kodowania binarnego, prostej mutacji i krzyżowania oraz selekcji ruletkowej (bardziej rozwinięte wersje wykorzystują też inne sposoby kodowania, operatory i metody selekcji);
- strategie ewolucyjne, charakteryzują się wykorzystywaniem kodowania rzeczywistoliczbowego, początkowo wykorzystywały tylko mutację, później także krzyżowanie, selekcję rankingową, później zastosowano m.in. samoadaptację;
- programowanie genetyczne – ewoluujące programy, a bardziej wyrażenia, zakodowane jako drzewa;
- programowanie ewolucyjne – ewoluujące automaty, zakodowane jako grafy przejść między stanami;
- metody hybrydowe, specjalizowane, algorytmy memetyczne, ewolucja różnicowa, ... – wszelkiego rodzaju metody ewolucyjne, zawierające elementy powyższych, specjalizowane – zawierające wiedzę o problemie z dodatkowymi metodami optymalizacji lokalnej;
- przeszukiwanie rozproszone – deterministyczna wersja AE z precyzyjnym doбором populacji startowej (tzw. ziarna) optymalizacją lokalną i specyficznym krzyżowaniem;

Algorytmy ewolucyjne

nomenklatura

Biologiczne korzenie AE wpłynęły na powstanie w tej dziedzinie specyficznego słownictwa, zapożyczonego z genetyki i innych działów biologii. Co prawda można je zastąpić mającymi to samo lub podobne znaczenie (w tym przypadku i ujęciu) pojęciami matematyczno-informatycznymi, to jednak pojęcia biologiczne są często (nad)używane, czasem nie mają też jednoznacznych odpowiedników w AE.

Do najczęściej używanych pojęć zapożyczonych z biologii należą:

- chromosom – oryginalnie odpowiednio upakowana nić DNA w jądrze komórki; w AE ciąg kodowy, zakodowany fragment rozwiązania (np. dla 1 wymiaru);
- gen – w genetyce fragment chromosomu, nici DNA, kodujący konkretny aminokwas (podstawową cegiełkę budującą białka); w AE fragment kodu, kodujący jakąś specyficzną właściwość, element rozwiązania, jeden z wymiarów w przestrzeni rozwiązań, często trudny do praktycznego wyodrębnienia;
- allel – w genetyce wariant cechy kodowanej przez gen; w AE konkretna wartość przypisana genowi;
- locus – w genetyce pozycja genu w nici DNA; w AE położenie odpowiedniego genu w kodzie rozwiązania (pojęcie może być trudne do interpretacji przy różnych metodach kodowania);

Algorytmy ewolucyjne

nomenklatura c.d.

- genotyp – w genetyce zapis genetyczny danego organizmu; w AE struktura rozwiązania, rozwiązanie problemu w przestrzeni kodowej;
- fenotyp – w biologii osobnik powstały z genotypu; w AE rozwiązanie zadania w jego naturalnej przestrzeni;
- osobnik – w biologii konkretny organizm; w AE konkretne rozwiązanie zadania;
- populacja – w biologii zbiór osobników jednego gatunku; w AE zbiór rozwiązań;
- mutacja – w genetyce przypadkowa, losowa zmiana w nici DNA; w AE modyfikacja rozwiązania (najczęściej losowana lecz celowa);
- krzyżowanie – w biologii rozmnażanie płciowe osobników z rekombinacją genów rodziców; w AE nowe rozwiązanie utworzone z 2 lub więcej rozwiązań składowych przez rekombinację (odpowiednie zmieszanie) cech (kodu) rozwiązań „rodzicielskich”.

Algorytmy ewolucyjne

ogólny schemat działania

```
t=0
Inicjacja populacji startowej P(0)
Ocena rozwiązań w P(0)
do {
    T(t)=Reprodukcja P(t) //powielenie i krzyżowanie osobników
    O(t)=Modyfikacja T(t) //mutacja osobników
    Ocena rozwiązań w O(t)
    P(t+1)=Selekcja/Sukcesja(O(t) lub O(t)∪P(t))
    t=t+1
}
while(warunek stopu)
```

Algorytmy ewolucyjne

ogólny schemat działania cd.

- Metoda polega na iteracyjnym powtarzaniu tych samych czynności: tworzenia nowej populacji (potomnej) przez modyfikację rozwiązań z populacji poprzedniej (rodzicielskiej) oraz selekcję w jakimś sensie lepszych rozwiązań z tej zmodyfikowanej (lub obu).
- Operatory modyfikujące populację (krzyżowanie i mutacja) posiadają prawdopodobieństwa wykonania raczej mniejsze od 1, nie każdy osobnik musi być przez nie zmieniony. Mutacja najczęściej ma bardzo małe ($\sim 0,01$), a krzyżowanie duże ($\sim 0,9$) prawdopodobieństwo wykonania.
- Jak widać do działania metody optymalizacyjnej potrzebna jest jedynie informacja o jakości rozwiązań (łatwa do obliczenia funkcja celu, często nazywana funkcją przystosowania, ang. fitness function).
- Metoda selekcji jak i metody reprodukcji i modyfikacji rozwiązań zależą nieco od wybranej konkretnej wersji AE i są w dużej mierze dobierane do wymagań rozwiązywanego problemu spośród wielu możliwych.
- Nowoczesne metody ewolucyjne bazują raczej na kryterium użyteczności danego zestawu metod (reprodukcji, mutacji, selekcji) do danego problemu niż na ortodoksyjnej poprawności i zgodności z metodami zaproponowanymi przez ich twórców.

Algorytmy ewolucyjne

wady metody

- Obliczenia są dość powolne i zasobochłonne.
- Dobre efekty można uzyskać tylko po dobrym przystosowaniu metody do wymogów zadania (wybór odpowiedniej metody kodowania, operatorów, selekcji, dostrojeniu parametrów). Standardowa wersja metody, stosowana do wszystkich problemów raczej nie spełni oczekiwań,
- Metoda często cierpi na chorobę „przedwczesnej zbieżności”, czyli zablokowania się rozwiązań w optimach lokalnych i stagnacji obliczeń.
- Problemowi „przedwczesnej zbieżności” trudno jest zapobiegać, wynika on często ze źle dobranych parametrów metody, ale też i ograniczeń środowiska komputerowego (skończona dokładność obliczeń, słabe parametry generatorów liczb losowych).
- Z problemem przedwczesnej zbieżności wiąże się też problem małego zróżnicowania populacji. Jeśli osobniki są prawie jednakowe, to efektywność obliczeń ewolucyjnych znacząco spada (wymiana jednakowej informacji między osobnikami).
- Brak sensownego kryterium stopu, ewolucja zapewne nie ma końca (koniec świata?) i trzeba zapewnić je sztucznie.
- Jak wszystkie metody heurystyczne nie gwarantuje znalezienia optimum w skończonej liczbie iteracji (dlatego symulacja AE prawie zawsze kończy się stagnacją, bo nie możemy czekać na wynik w nieskończoność).

Algorytmy ewolucyjne

zalety metody

- Nie potrzebne są założenia o postaci funkcji celu, istnieniu pochodnych, ciągłości, itp.
- Tego samego schematu rozwiązywania można używać do szerokiej gamy zadań.
- Obliczenia prowadzone są równoległe przez dużą grupę „osobników” - rozwiązań, więc jest spora szansa na odwiedzenie wielu obiecujących obszarów.
- Obliczenia trudniej utykają w ekstremach lokalnych i łatwiej mogą je opuścić dzięki wymianie informacji pomiędzy „osobnikami”.
- Obliczenia można łatwo zrównoleglić w systemach wieloprocesorowych, rozproszonych, itp.
- Metoda jest skalowalna, łatwo ją przystosować do konkretnego zadania i jego rozmiaru.
- Mimo braku gwarancji na otrzymanie rozwiązań optymalnych, dobrze dobrana metoda działa szybko i skutecznie.

Algorytmy ewolucyjne

eksploracja i eksploatacja

- Jednym ze sposobów walki z „przedwczesną zbieżnością” jest zbalansowanie działania operatorów genetycznych mutacji i krzyżowania (lub analogicznych).
- Operator mutacji wprowadza losowe zaburzenie do rozwiązania, jest odpowiedzialny za właściwości eksploracyjne ewolucji (operator globalny), dzięki jego działaniu możliwe jest tworzenie nowych cech w osobnikach (np. przeniesienie osobnika do innego obszaru dziedziny, tworzenie nowych sekwencji w rozwiązaniu), niestety pozytywne zmiany generowane są dość rzadko i aby zminimalizować niekorzystne, mutacja wykonywana jest dość rzadko, z niskim prawdopodobieństwem. W naturze organizmy również wkładają dużo wysiłku, aby bronić się przed mutacjami. Zbyt intensywna mutacja powoduje, że AE zaczyna przypominać metodę Monte-Carlo – losowe przeszukiwanie przestrzeni rozwiązań bez dziedziczenia informacji po „przodkach”. Zbyt słaba mutacja zatrzyma eksplorację w AE.
- Operator krzyżowania odpowiedzialny jest za eksploatacyjne właściwości ewolucji. Nowy osobnik powstaje z rekombinacji cech rodziców, czyli jest do nich podobny (operator lokalny) możliwe, że odziedziczy najlepsze ich cechy, operator ten wykonywany jest bardzo często (z prawdopodobieństwem bliskim 1). W naturze krzyżowanie (rozmnażanie płciowe) zostało wynalezione prawdopodobnie „dość późno”, niecały 1 mld lat temu (historia życia na ziemi to ok. 3,7 mld lat), jednakże stało się to początkiem gwałtownego rozwoju życia.

Algorytmy ewolucyjne

eksploracja i eksploatacja c.d.

- Wyłączenie mutacji w AE skutkowałoby szybką zbieżnością do najbliższych optimów lokalnych. Algorytm ewolucyjny bez krzyżowania działałby zdecydowanie wolniej, tzn. suboptymalne lub przy odrobinie szczęścia optymalne rozwiązania pojawiłyby się po bardzo długim czasie, lecz droga do nich nie byłaby zamknięta. Zastosowanie obu operatorów z dobrze dobranymi prawdopodobieństwami wykonania zdecydowanie przyspiesza obliczenia i zwiększa efektywność metody.
- Duże znaczenie w walce z przedwczesną zbieżnością ma też zastosowana metoda selekcji, która wprowadza presję selekcyjną. Silną mutację można też zrównoważyć silną selekcją, promującą tylko najlepsze osobniki.
- Zwiększenie właściwości eksploracyjnych uzyskuje się też przy zastosowaniu selekcji nowej populacji tylko z osobników z populacji tymczasowej (pomijając populację rodziców), jest to tzw. strategia rozwoju populacji (μ, λ) - stosowana jest głównie przy optymalizacji zadań niestacjonarnych, bardziej zachowawcza (eksploatacyjna) jest strategia rozwoju populacji $(\mu + \lambda)$, w której nową populację wybiera się z obu podpopulacji (rodzicielskiej i tymczasowej), stosowana jest w rozwiązywaniu problemów stacjonarnych.

Algorytmy ewolucyjne

kodowanie rozwiązań

W algorytmach ewolucyjnych operuje się nie na rzeczywistym rozwiązaniu, a jego symbolicznej reprezentacji, którą w każdym momencie powinno się dać odkodować do konkretnego rozwiązania.

W tradycyjnych metodach ewolucyjnych używano kodowania:

- **binarnego**, w którym należy odpowiednio zakodować problem tak, aby można było go wyrazić przy użyciu zapisu binarnego, np. dla przypadku optymalizacji funkcji ciągłej skończony podzbiór zbioru liczb rzeczywistych (osi liczbowej) dzieli się na liczbę przedziałów o skończonej długości i liczbie będącej całkowitą potęgą 2 (w przypadku wielowymiarowym, operację przeprowadza się dla każdego wymiaru), przedziałom przydziela się kolejne numery zapisane binarnie, chcąc uzyskać większą dokładność obliczeń należy użyć większej liczby bitów, przekodowanie na liczbę rzeczywistą uzyskujemy ze wzoru:

$$z = z_0 + \frac{z_k - z_0}{2^n - 1} * \sum_{i=0}^{n-1} x_i * 2^i$$

gdzie: z_0 , z_k - początek i koniec przedziału, n - liczba bitów, z - odkodowana liczba, x_i - wartość bitu o numerze i (licząc od najmniej znaczącego bitu zerowego).

- **rzeczywistoliczbowego**, używano go oczywiście tylko do optymalizacji funkcji ciągłych, w przypadku funkcji wielowymiarowych mamy do czynienia z wektorem liczb rzeczywistych.

Obecne metody ewolucyjne używają w zasadzie dowolnych struktur danych do zakodowania rozwiązań: wektorów, tablic, drzew zawierających symbole binarne, całkowitoliczbowych i rzeczywiste, automatów skończonych, a nawet ewoluujących programów ...

Algorytmy ewolucyjne

operatory genetyczne – inne przypadki

W przypadku kodowania rzeczywistoliczbowego mutacja to zwykła modyfikacja liczby występującej w rozwiązaniu o wartość wylosowaną z rozkładu normalnego (czasem też Cauchy'ego):

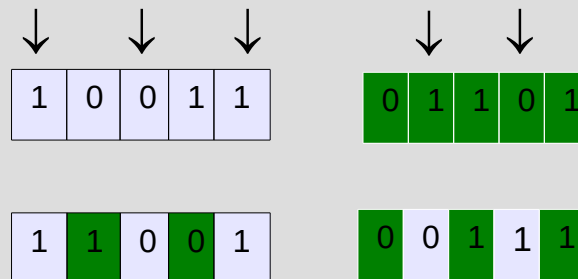
$$X^{t+1}_i = X^t_i + N(0,1)$$

a krzyżowanie to tzw krzyżowanie uśredniające:

$$X^{t+1}_i = \alpha * X^t_i + (1-\alpha) * Y^t_i \quad Y^{t+1}_i = \alpha * Y^t_i + (1-\alpha) * X^t_i$$

X^{t+1}_i, Y^{t+1}_i – i -ta współrzędna osobnika potomnego, X^t_i, Y^t_i – i -ta współrzędna osobnika rodzicielskiego, α – wartość losowana z rozkładu $U(0,1)$.

W kodowaniu binarnym i rzeczywistoliczbowym (tylko w przypadku wielowymiarowym) można stosować też krzyżowanie z maską (równomierne). Polega ono na losowym wybieraniu do osobników potomnych wartości kolejnych współrzędnych osobników rodzicielskich. Ta która zostanie wylosowana trafi do pierwszego potomka, elementy niewylosowane utworzą 2 osobnika:



Obecnie stosuje się najrozmaitsze sposoby modyfikacji rozwiązań: losowe i deterministyczne, łącznie z metodami bazującymi na wiedzy o problemie, prostymi metodami optymalizacji lokalnej i heurystykami.

Algorytmy ewolucyjne

selekcja osobników do populacji potomnej

selekcja proporcjonalna (ruletkowa)

- Każdy osobnik ma przyporządkowaną wartość funkcji dopasowania $f(j)$.
- Prawdopodobieństwo wyboru i wartość oczekiwana liczby potomków to:

$$p(j) = \frac{f(j)}{\sum_{l=1}^k f(l)}$$

$$E(p(j)) = \frac{k * f(j)}{\sum_{l=1}^k f(l)} = \frac{f(j)}{\bar{f}}$$

gdzie: k – liczność populacji, $f(j)$ – wartość funkcji dopasowania osobnika j , \bar{f} – średnia wartość funkcji dopasowania populacji.

- Procedurę losowego wyboru przeprowadzamy k (dokładniej tyle razy, ile osobników ma mieć populacja potomna, nie zawsze musi to być taka sama liczba jak dla populacji rodzicielskiej) razy.

Wniosek!

Osobniki o większym prawdopodobieństwie powinny mieć więcej potomków niż osobniki o mniejszym, a tym samym populacja powinna ewoluować w kierunku lepszego przystosowania – lepszych rozwiązań.

Algorytmy ewolucyjne

selekcja osobników do populacji potomnej

- **selekcja rangowa** - polega na nadaniu rozwiązaniom wartości (rang) na podstawie wartości funkcji dopasowania przez ich posortowanie i nadanie kolejnych wartości rang według wzorów $r(X)=a+k(1-f(X)/r_{max})$ lub $r(X)=a+k(r_{max}-f(X))^b$, prawdopodobieństwo wyboru określa się przez normalizację rang, a dalszy ciąg działania tak jak w selekcji proporcjonalnej, a , b i k to parametry metody, $r(X)$, r_{max} – ranga rozwiązania X i ranga maksymalna $f(X)$ – wartość funkcji dopasowania rozwiązania X . Metoda ta ma lepsze parametry niż s. ruletkowa i jest stosowana w rozwiązaniach praktycznych.
- **wybór najlepszych osobników z populacji** – algorytm działania jest oczywisty (sortowanie), w metodzie tej oczywiście należy wybierać rozwiązania z większej populacji niż docelowa, metoda powoduje dość łatwe wpadanie populacji w minima lokalne lub unifikację populacji, nie ma wobec tego najlepszych parametrów i nie jest zbyt chętnie stosowana.
- **selekcja turniejowa** – z populacji losowanych jest k osobników ($k \geq 2$), najlepszy jest wybierany, możliwe są wersje ze zwracaniem i bez; jest to zdecydowanie jedna z najlepszych i jednocześnie najprostszych metod selekcji, jest i powinna być używana w rozwiązaniach praktycznych, jej użycie w rozważaniach teoretycznych jest dość skomplikowane.

Algorytmy ewolucyjne

selekcja osobników do populacji potomnej

- **selekcja deterministyczna** – części całkowite wartości oczekiwanej liczby potomków obliczane są jak w selekcji proporcjonalnej, stają się liczbami potomków dla osobników, ewentualny nadmiar lub niedomiar jest odpowiednio poprawiany przez dołączenie kolejnych najlepszych lub usunięcie najśłabszych; metoda ta charakteryzuje się bardzo silnym naciskiem selekcyjnym, jednakże prowadzi to do szybkiej unifikacji populacji (straty różnorodności) i osiadania w optimach lokalnych, metoda nie jest więc zbyt często używana.
- **selekcje sterowane** – istnieją metody selekcji, które zmieniają swoje parametry w trakcie działania programu lub składają się z kilku metod składowych o różnej sile nacisku selekcyjnego i zmieniają, w zależności od stanu populacji, częstość ich stosowania.
- **czas życia osobnika** – istnieją rozwiązania, w których zarówno licznosc populacji jak i selekcja nie są typowo ustalone, a każdy osobnik otrzymuje czas życia w iteracjach algorytmu, zależny od jego jakości;

Algorytmy ewolucyjne

selekcja osobników do populacji potomnej

ćwiczenie

W pewnym algorytmie ewolucyjnym maksymalizującym pewną funkcję mamy populację rodzicielską o następujących wartościach funkcji celu: {5,5; 4,5, 1,2; 3,3}. W wyniku działań operatorów genetycznych powstała populacja potomna o wartościach funkcji celu: {6,1; 4,9, 3,2; 3,4}. Jak będzie wyglądać skład nowej populacji rodzicielskiej jeśli do wyboru osobników do niej użyto:

- metody selekcji najlepszych osobników i strategii $(\mu+\lambda)$,
- metody selekcji najlepszych osobników i strategii (μ,λ) ,
- metody selekcji deterministycznej i strategii (μ,λ) ?

Algorytmy ewolucyjne

uwzględnienie ograniczeń

Bardzo często mamy do czynienia z problemem, w którym występują ograniczenia. AE sam z siebie jest metodą optymalizacji bez ograniczeń i aby je uwzględnić, należy zastosować jedną z kilku możliwych metod:

- w przypadku ograniczeń „kostkowych” można sprawdzać, czy w trakcie inicjacji lub po działaniu operatorów genetycznych nie pojawiają się rozwiązania niedopuszczalne i je odpowiednio „przycinać” lub rzutować
- w przypadku bardziej skomplikowanych ograniczeń funkcyjnych można stosować transformację zmiennych – metody znane z tradycyjnej optymalizacji funkcji rzeczywistych
- w pewnych przypadkach (np. gdy ograniczenia wynikają z samej idei problemu) można zastosować odpowiednie rodzaje kodowania i specjalnie zaprojektowane w tym celu operatory genetyczne, które nie pozwolą na wygenerowanie rozwiązań niedopuszczalnych (np. w problemie TSP zakodowanie w postaci listy miast, krzyżowanie jako operator PMX i mutacja jako operator inwersji)

Algorytmy ewolucyjne

uwzględnienie ograniczeń

- metoda naprawy – polega na „naprawieniu”, czyli przekształceniu rozwiązania w taki sposób, aby było dopuszczalne, metoda taka w niektórych przypadkach może być kosztowna obliczeniowo („przycinanie” i rzutowanie na ograniczenie też można potraktować jako naprawę).
- metoda funkcji kary – gdy nie można zastosować innych metod, to należy stosować pogorszenie funkcji celu rozwiązań niedopuszczalnych tak, aby zostały wyeliminowane przez selekcję, granicznym przypadkiem takiego podejścia jest metoda „kary śmierci”, która prowadzi do natychmiastowej eliminacji rozwiązań niedopuszczalnych przez nadanie im maksymalnie nieodpowiedniej (np. $-\infty$ lub $+\infty$) wartości funkcji dopasowania. W przypadku AE funkcje kary mogą nie posiadać pochodnych lub być nieciągłe, inaczej niż w przypadku optymalizacji klasycznej.

Algorytmy ewolucyjne

kryterium stopu

Ewolucja nie ma naturalnego kryterium stopu (koniec świata?). Z tego też powodu nie mają jej i AE.

Ponieważ są one metodami heurystycznymi, więc nie potrafią odróżnić rozwiązania optymalnego i jego osiągnięcie (choć mało prawdopodobne) w ogólnym przypadku nie może być kryterium stopu. Można wyróżnić następujące kryteria

- **kryteria maksymalnego kosztu** – najczęściej po określonej liczbie wywołań funkcji oceny następuje zakończenie obliczeń. Jednym z jego wariantów (zresztą bardzo często używanym) jest narzucona maksymalna liczba iteracji/generacji;
- **kryterium osiągnięcia w trakcie symulacji zadowalającej wartości rozwiązania;**
- **kryterium minimalnej szybkości poprawy** – gdy od ostatniej poprawy rozwiązania minie odpowiednio dużo iteracji, program jest przerywany;
- **kryterium zaniku różnorodności populacji**, która jest jak wiadomo niezbędna do efektywnego działania ewolucji, jej zanik to praktycznie koniec możliwości znalezienia lepszych rozwiązań;
- **kryterium zaniku adaptowanego zasięgu operatora mutacji**, możliwe do zastosowania tylko tam, gdzie go zastosowano (np. w SE), mały zasięg uniemożliwia szerszą eksplorację przestrzeni i tym samym oznacza, że algorytm osiadł w ekstremum.

Ewolucja różnicowa (DE)

Ewolucja różnicowa jest jedną z nowszych metod ewolucyjnych. Najlepiej nadaje się do optymalizacji problemów rzeczywistoliczbowych, choć istnieją wersje binarne i całkowitoliczbowe. Metoda działa bardzo efektywnie, a jest bardzo prosta. Ogólny algorytm jest zbliżony do typowego algorytmu ewolucyjnego, jednak szczegółowe operacje są specyficzne dla metody. W typowej wersji osobniki populacji o rozmiarze S są n -wymiarowymi wektorami liczb rzeczywistych.

Ewolucja różnicowa (DE)

Algorytm

begin

$t:=0$

 inicjacja populacji startowej X^0

 ocena X^0

 while (not warunek_stopu) do

 begin

$V^t := \text{mutacja } X^t$

$U^t := \text{krzyżowanie } V^t \text{ i } X^t$

 ocena U^t

$X^{t+1} := \text{selekcja/sukcesja lepszego z pary rodzic/potomek w populacjach } X^t \text{ i } U^t\}$

$t:=t+1$

 end

end

Ewolucja różnicowa (DE)

Mutacja polega na wytworzeniu nowego rozwiązania (mutanta v) na bazie kilku (najczęściej 3, czasem 5 i więcej) wylosowanych (różnych!) osobników:

$$v_i = x_i^1 + F * (x_i^2 - x_i^3) \quad \text{lub} \quad v_i = x_i^1 + F * (x_i^2 - x_i^3) + F * (x_i^4 - x_i^5)$$

Krzyżowanie w DE polega na zastosowaniu metody krzyżowania równomiernego, zwanego także krzyżowaniem z maską (losowy wybór współrzędnych z rodzica x^0 lub mutanta v)

$$u_i = \begin{cases} v_i & \text{jeśli } U(0,1) < CR \\ x_i^0 & \text{w przeciwnym przypadku} \end{cases}$$

gdzie: F – współczynnik amplifikacji (wzmocnienia), najczęściej ok. 0,5; CR – prawdopodobieństwo mieszania cech.

Selekcja – osobnik próbny u jest porównywany z rodzicem x^0 , lepszy przechodzi do kolejnej generacji.

Ewolucja różnicowa (DE)

Istnieje skrócony, symboliczny zapis wariantu metody: DE/x/y/z w którym:

x – to osobnik bazowy poddawany mutacji, może to być osobnik losowy(rand), najlepszy (best) i bieżący – kolejny (current);

y – liczba różnic osobników biorących udział w mutacji, dla 3 osobników jest to wartość 1, dla 5 osobników jest to 2, ...

z – model krzyżowania, obecnie stosowany w którym lepsze rozwiązanie zastępuje gorsze to bin.

Czyli najprostsza wersja z 1 różnicą i mutowaniem każdego osobnika populacji to:
DE/current/1/bin

Algorytm przeszukiwania z tabu

tabu search (TS)

Metoda **tabu search** jest kolejnym krokiem rozwoju algorytmów optymalizacji lokalnej (podobnych do algorytmu wzrostu), polegających na przeszukiwaniu sąsiedztwa ostatnio zaakceptowanego rozwiązania. Metoda szczególnie nadaje się do rozwiązywania problemów kombinatorycznych, dyskretnych, zadania ciągłe mogą być rozwiązywane dopiero po odpowiedniej dyskretyzacji.

Jak wiadomo metody optymalizacji lokalnej dość łatwo wpadają w optima lokalne lub ulegają zapętleniu i w związku z tym ich efektywność nie jest duża. Badacze usiłują poprawić te metody, wprowadzając mechanizmy umożliwiające opuszczenie optimów lokalnych znane np. z symulowanego wyżarzania, algorytmu „małych światów”, metody wielostartu lub restartu, makromutacje zapobiegające stagnacji itp. Jedną ze skuteczniejszych metod jest też wprowadzenie reguł sterujących przebiegiem algorytmu. Reguły te mają w tym przypadku postać zakazów pewnych ruchów, wyborów pewnych rozwiązań ponownie i w związku z tym zostały nazwane „tabu” od nazwy zakazów religijnych (czasowych lub nieograniczonych czasowo) dotyczących przebywania w pewnych miejscach, stykania się z pewnymi ludźmi, wykonywania pewnych czynności, występujących pod tą nazwą w religiach Polinezji.

Algorytm przeszukiwania z tabu

tabu search (TS)

W algorytmie zakazy te są tworzone przez samą metodę na podstawie istniejących w niej metareguł, czyli są przykładem „uczenia się” algorytmu (algorytm posiada struktury pamięci do przechowywania odpowiednich faktów i reguł), innymi słowy adaptacji do zaistniałych warunków.

Dodatkowo algorytm został również wyposażony w reguły przełamывania tabu (tzw. kryterium aspiracji), jeśli zaistniała sytuacja jest wyjątkowo korzystna i odrzucenie dobrego rozwiązania byłoby ewidentną stratą dla obliczeń lub gdy przez dłuższy czas nie można znaleźć nowego dopuszczalnego rozwiązania bo wszystkie lub prawie wszystkie możliwe „sąsiednie” rozwiązania, możliwe do wygenerowania przez operator przeszukiwania sąsiedztwa, są na liście tabu.

Algorytm przeszukiwania z tabu

Reguły tabu mogą występować na trzech poziomach działania metody (nie wszystkie muszą wystąpić w każdej realizacji algorytmu) lub też jako różne rodzaje pamięci działań metody:

- pamięć krótkotrwała pamięta ostatnio odwiedzone rozwiązania i wprowadza (najczęściej czasowy) zakaz powrotu do nich – jest to dość naturalny zakaz, zabezpieczający metodę przed zapętleniem;
- pamięć średnioterminowa wzmacniająca reguły umożliwiające przeszukiwanie bardziej obiecujących obszarów przestrzeni rozwiązań;
- pamięć długoterminowa zawiera reguły dywersyfikujące (różnicujące) umożliwiające przeszukiwanie nowych obszarów przestrzeni rozwiązań.

Dane w pamięci mogą być przechowywane w sposób bezpośredni np. jako gotowe rozwiązania lub pośrednio np. jako ślady wykonywanych działań (np. przestawień elementów rozwiązań) lub częstości występowania pewnych zdarzeń, czyli pewne statystyki procesu przeszukiwania.

W przypadku bardziej skomplikowanych reguł tabu, opartych na częstotliwościach wystąpień pewnych atrybutów, reguły tabu przybierają raczej postać funkcji kary, osłabiania lub wzmacniania pewnych akcji proporcjonalnie do stopnia naruszenia zakazu niż bezwzględnej eliminacji danego działania.

Algorytm przeszukiwania z tabu

schemat działania

1. Inicjalizacja danych w strukturach pamięci algorytmu
2. Wybór/wylosowanie rozwiązania startowego x_0 oraz jego ocena
3. Rozwiązanie startowe staje się najlepszym aktualnie rozwiązaniem $x^*=x_0$
4. do {
 5. Wygenerowanie n nowych zmodyfikowanych rozwiązań w sąsiedztwie x^*
 6. Ocena nowych rozwiązań z uwzględnieniem wpływu reguł tabu (funkcje kary, odrzucenie rozwiązań zakazanych,...)
 7. Wybór x' najlepszego z wygenerowanych rozwiązań
 8. Jeśli jest ono lepsze od x^* to $x^*=x'$
 9. Aktualizacja danych w pamięci: reguł tabu, częstości przejść, ...}
10. while (warunek stopu)

Algorytm przeszukiwania z tabu

schemat działania

Wygenerowane rozwiązania (i sposób ich wygenerowania) są oceniane przy pomocy funkcji celu oraz reguł tabu. W tym kroku niektóre rozwiązania mogą zostać odrzucone lub słabiej ocenione w efekcie zadziałania reguł tabu. W wyniku zadziałania tzw. kryterium aspiracji rozwiązanie „będące na indeksie”, lecz bardzo dobre (lub najlepsze) mogą być „ułaskawione” i dopuszczone do dalszych etapów metody.

Z pozostałych po ocenie rozwiązań, wybierane jest najlepsze x' .

Jeśli jest ono lepsze od x^* , to staje się nowym x^* .

Aktualizacja struktur pamięci polega na zmniejszeniu indeksu tabu rozwiązań i ruchów, które się nie pojawiły, zwiększenia indeksu tym, które się pojawiły, zaakceptowane, odpowiednio powiększenia/zmniejszenia statystyk zadziałania operatora (-ów) modyfikujących rozwiązania.

Kryterium stopu metody jest typowe, jak dla innych metod heurystycznych.

Algorytm przeszukiwania z tabu

schemat działania

W trakcie inicjalizacji struktur pamięci wszelkie reguły tabu są zerowane, ich ustawienie nastąpi w trakcie działania metody.

Do wygenerowania rozwiązania początkowego można użyć losowania lub też innych bardziej wyspecjalizowanych metod (algorytmu zachłannego, metody losowej+algorytm optymalizacji lokalnej, wybór najlepszego z kilku wylosowanych, itp.). Rozwiązanie to staje się początkowym bieżącym najlepszym rozwiązaniem.

Wygenerowanie rozwiązań zmodyfikowanych w sąsiedztwie najlepszego to oczywiście konieczność użycia pewnych specjalizowanych operatorów (inteligentnych?). W metodzie standardowo jest jeden, ale heurystyki są po to, żeby je rozwijać...

Algorytm przeszukiwania z tabu

szczegóły techniczne

Do przechowywania całych rozwiązań potrzebna jest oczywiście macierz, wektor lub lista o odpowiedniej wielkości, np. o długości tabu (kolejka). W zasadzie można również przechowywać bardziej ogólne informacje np. o „klasach” lub regionach bardzo zbliżonych rozwiązań (niewiele się różniących).

Do przechowywania informacji o zmianach w rozwiązaniu powodowanych przez operator sąsiedztwa potrzebna będzie macierz kwadratowa (lub kilka takich macierzy) o rozmiarze liczby elementów rozwiązania (np. liczby miast), w której po zastosowaniu danej zmiany wstawia się do odpowiednich komórek liczby mówiące jak długo takie przejście jest zabronione. Macierze takie mogą zawierać znacznie więcej elementów (kilka macierzy lub macierz bardziej złożonych struktur danych), zależnie od wykorzystywanych atrybutów. Należy też zauważyć, że ich odświeżanie po każdej iteracji może być kosztowne obliczeniowo.

Algorytm przeszukiwania z tabu

rozszerzenia metody

- czasy obowiązywania tabu mogą być zmienne w trakcie działania algorytmu
- przy nieskomplikowanych operatorach sąsiedztwa, postulowano przejście wszystkich sąsiadów, jednak w bardziej skomplikowanych przypadkach jest to niemożliwe i stosuje się przejście pewnej ich części (np. wylosowanej)
- stosuje się bardzo różnorodne rodzaje reguł tabu, oparte na różnych danych możliwych do uzyskania od procesu optymalizacji
- stosuje się różne rodzaje kryteriów aspiracji np. zmniejsza się okres trwania tabu
- stosuje się metody hybrydowe np. połączenia z algorytmami ewolucyjnymi lub przeszukiwaniem rozproszonym – jako wykorzystywane tam algorytmy optymalizacji lokalnej
- pomysły wykorzystania reguł tabu wykorzystuje się też do sterowania innymi procesami, nie tylko optymalizacyjnymi

Algorytmy mrówkowe

Algorytmy mrówkowe, podobnie jak wiele innych heurystyk, mają źródło w biologii i są jednymi z pierwszych algorytmów „odzwierzęcych” (obecnie jest ich kilkanaście). W tym przypadku obserwacje działania pojedynczych mrówek, jak i całego mrowiska oraz sposobów komunikacji między owadami zainspirowały badaczy do wykorzystania ich jako pewnej heurystyki, na podstawie której można stworzyć algorytm optymalizacyjny.

W omawianym przypadku największe znaczenia ma przekazywanie informacji między owadami (np. dotyczących drogi do pożywienia) drogą sygnałów chemicznych (feromonowych) – mrówki starają się podążać po śladzie feromonowym pozostawionym przez inne mrówki. Zauważono również, że mrówki po pewnym czasie wybierają coraz krótszą drogę do pożywienia, co oznacza, że występuje u nich mechanizm optymalizacji drogi, dzięki większej koncentracji feromonu na krótszej drodze niż na dłuższej – jest to efekt dodatniego sprzężenia zwrotnego: wybieranie krótszej drogi i wzrost stężenia feromonu oraz częstotliwości przechodzenia mrówek zachęca mrówki do jeszcze częstszego odwiedzania, itd.

Algorytmy mrówkowe

Sposób działania algorytmu mrówkowego powoduje, że dość intuicyjnie można go zastosować do problemów związanych z grafami, optymalizacją drogi (np. TSP), nie znaczy to jednak, że metoda może służyć tylko do tego celu. Stworzony algorytm jest metodą uniwersalną i może być wykorzystywany do rozwiązywania dowolnych zadań dyskretnych - po odpowiednich modyfikacjach np. przedstawieniu problemu w postaci grafu przejść.

Algorytmy mrówkowe

algorytm metody

1. Ustalenie początkowego poziomu feromonu w miejscach (krawędziach grafu) odpowiadających przejściom między kolejnymi elementami rozwiązywanego problemu
- do {
2. Odparowanie feromonu (czyli zmniejszenie wartości wag feromonowych krawędzi).
 3. Wylosowanie punktu startu agenta-mrówki.
 4. Wylosowanie ścieżek na podstawie poziomu feromonu (wag) na przejściach między kolejnymi elementami rozwiązania.
 5. Naniesienie feromonu (zwiększenie wag) krawędzi wykorzystanych w rozwiązaniu.
 6. Operacje dodatkowe – np. zbalansowanie feromonu ścieżek (próba wyjścia z optimum lokalnego), dodatkowe wzmocnienie pewnych krawędzi, itp.
- }
- while (warunek stopu)

Algorytm mrówkowy

schemat działania

Schemat działania AS wymaga doprecyzowania kilku parametrów:

- liczba mrówek – wpływa na dokładność i czas działania metody
- poziom feromonu początkowego (wagi feromonowe początkowe krawędzi) – nie może być zbyt duży, gdyż działanie metody będzie przypominać losową generację rozwiązań – nie zaistnieje dodatnie sprzężenie zwrotne, umożliwiające poprawę rozwiązań
- odparowanie feromonu (zmniejszenie wag feromonowych) – również musi być na odpowiednim poziomie: zbyt duże przetnie sprzężenie zwrotne, zbyt małe może doprowadzić do dominacji lokalnie lepszych rozwiązań
- punkt startowy losuje się dowolnie, natomiast wyborem kolejnych ścieżek zarządza prawdopodobieństwo wyboru drogi, które opisane jest wzorem:

$$p_{i,j}^k(t) = \frac{[\tau_{i,j}(t)]^\alpha * [\eta_{i,j}]^\beta}{\sum_{l=1}^n ([\tau_{i,l}(t)]^\alpha * [\eta_{i,l}]^\beta)}$$

$p_{i,j}^k(t)$ – prawdopodobieństwo wyboru drogi; $\tau_{i,j}(t)$ – natężenie feromonu, $\eta_{i,j}$ – waga krawędzi wynikająca z optymalizowanego zadania, np. odwrotność odległości między miastami w przypadku TSP; α, β - parametry określające wpływ feromonu i wagi krawędzi wynikającej z zadania na p. wyboru drogi; i, j - indeksy elementów rozwiązania (np. miast); k – numer mrówki; n – liczba punktów dopuszczalnych, które można włączyć do rozwiązania w danym momencie.

Algorytm mrówkowy

schemat działania cd.

- następny krok do przejścia jest losowany na podstawie prawdopodobieństwa z poprzedniego slajdu metodą ruletkową (proporcjonalną), parametry α , β umożliwiają sterowanie znaczeniem feromonu (informacji uzyskanej w trakcie działania algorytmu) i wagi pochodzącej od zadania, czasem konieczne może być uwzględnienie tu ograniczeń
- aktualizacja feromonu na wygenerowanej ścieżce wykonywana jest według następującego wzoru:

$$\tau_{i,j}(t+1) = (1-\rho) * \tau_{i,j}(t) + \sum_{l=1}^n \Delta \tau_{i,j,l}(t)$$

$\tau_{i,j}(t)$ – natężenie feromonu; ρ - współczynnik parowania feromonu, zazwyczaj 0,5...0,9; $\Delta \tau_{i,j,l}(t)$ – feromon zostawiony przez mrówki wykorzystujące dane połączenie.

- aktualizacja feromonu jest jednym z najczęściej modyfikowanych i rozwijanych kroków algorytmu, odkładanie feromonu przez mrówki jest jedyną formą komunikacji mrówek i metoda wykonywania tego kroku ma decydujący wpływ na pojawienie się informacyjnego sprzężenia zwrotnego między mrówkami oraz efektywność algorytmu; możliwych jest kilka metod odkładania feromonu:
 - z poziomem stałym: mrówka odkłada jednakową ilość feromonu na wykorzystanych ścieżkach grafu przejść;
 - z poziomem średnim: ilość odkładanego feromonu zależy od wagi (np. długości) krawędzi;

Algorytm mrówkowy

schemat działania cd.

- z poziomem cyklicznym: ilość odkładanego feromonu zależy od wagi (np. długości) całego rozwiązania (w tym przypadku aktualizacja musi być koniecznie osobnym punktem algorytmu, w pozostałych dwóch aktualizacje można wykonywać na bieżąco wraz z wyborem kolejnych ścieżek);
- często wprowadza się pojęcie **elity** – zbioru najlepszych ścieżek, rozwiązania, a właściwe ich krawędzie składowe są dodatkowo punktowane za posiadanie ich fragmentów przez dodanie dodatkowych porcji feromonów (ta część algorytmu może też odpowiadać pkt. 6 metody);
- w tzw. systemach kolonii mrówkowych stosuje się połączenie 2 rodzajów aktualizacji globalnego i lokalnego, czyli w zasadzie aktualizacji z poziomem cyklicznym i aktualizacji z poziomem stałym lub średnim z tą różnicą, że aktualizacji globalnej dokonuje tylko najlepsza mrówka w danej iteracji, aktualizacji lokalnej dokonują wszystkie według wzoru:

$$\Delta \tau'_{i,j}(t) = (1 - \xi) * \Delta \tau_{i,j}(t) + \xi * \tau_0$$

$\Delta \tau_{i,j}(t)$ – feromon jak w metodzie z poziomem cyklicznym równy $1/L_k$ (odwrotność długości/wagi drogi);

ξ - parametr z zakresu (0,1); τ_0 – parametr o znaczeniu zbliżonym do startowego poziomu feromonu.

Algorytmy mrówkowe

rozszerzenia metody

- **elitarny system mrówkowy (EAS)** – najlepsze globalnie rozwiązanie pozostawia dodatkową porcję feromonu po każdej iteracji.
- **system mrówkowy Max-Min (MMAS)** - feromon jest pozostawiany tylko przez najlepszą globalnie mrówkę lub najlepszą w danej iteracji. Wszystkie krawędzie są ustawiane na wartości startowe (reset) w przypadku stagnacji algorytmu.
- **system kolonii mrówkowych (ACO)** – prezentowane wcześniej – najlepsza globalnie mrówka jest odpowiedzialna za aktualizację globalną, pozostałe za aktualizację lokalną.
- **rankingowy system mrówkowy (ASRank)** – wszystkie rozwiązania/mrówki otrzymują rangę powiązaną z jakością wyniku, ilość pozostawianego feromonu jest ważona współczynnikiem proporcjonalnym do wagi rozwiązania, lepsze zostawiają więcej, gorsze niewiele feromonu.
- **ortogonalny ciągły system mrówkowy (COAC)** metoda rozwiązywania zadań ciągłych ze specjalnym sposobem przeglądania wybranych fragmentów zdyskretyzowanej przestrzeni poszukiwań (metoda ortogonalnego przeglądu).
- **rekurencyjny system mrówkowy (RACO)** – rekurencja polega na sprawdzaniu sąsiednich miast do tych odwiedzonych przez najlepsze mrówki (coś w rodzaju dodatkowej optymalizacji lokalnej).

Koniec

Dziękuję za uwagę.