**AMEBA konkrety**

**Selekcja**

* N - wielkość populacji, f\_i - fitness i-tego osobnika w pop.
* Ruletkowa: N razy losowy wybór proporcjonalnie do *f\_i*
* Turniejowa: N razy najlepszy osobnik z losowej k-kombinacji (jeśli k=N, to steepest, jeśli k=1, to random walk -> napór selekcyjny proporcjonalny do k)
* Losowa według reszt (z lub bez powtórzeń): tyle i-tego osobnika kopii w nowej pop. ile część całkowita z *e\_i=N\*f\_i/sum(f)*, pozostałe miejsca losowo z prawd. równym części ułamkowej *e\_i* (albo w malejącej kolejności jeśli deterministyczna, albo ruletkowo jeśli z powtórzeniami)
* Porządkowa: losowo według funkcji prawdopodobieństwa od miejsca w rankingu *f\_i*
* Elitarna: dowolna metoda selekcji, ale najlepszy zawsze przechodzi
* Model ze ściskiem: nowe osobniki zastępują najbardziej podobne w starej pop. (symulacja walki o ograniczone zasoby w ekologicznych niszach)
* Model wyspowy: podział na podpopulacje, w których ewolucja odbywa się niezależnie, z okresową migracją -> większa eksploracja
* Selekcja konwekcyjna: podział na podpopulacje na podstawie podobieństwa wartości funkcji celu osobników [najlepsi, średni, najsłabsi] -> poprawa eksploracji przez balans presji selekcyjnej
* Selekcja negatywna: usuwanie genotypów z pop., żeby zrobić miejsce na inne

**Skalowanie** (pomaga utrzymać stały napór selekcyjny podczas ewolucji)

* Selekcja np. turniejowa i rankingowa nie zależą od konkretnych wart. f\_i, więc skalowanie nie ma sensu
* Prawd. wybrania i-tego osobnika przed skalowaniem: p\_i = f\_i/sum(f)
* Liniowe: f’ = a\*f + b
* Sigma-obcinające: f’ = max(0, f - mu + c\*sigma) [mu - średni fitness pop., sigma - odch std. fitnessu pop., c - napór selekcyjny (im większe c, tym mniejszy napór)]
* Mediana: p’ = 1/(1 + exp( (f\_i - median(f)) / sigma ))

**Krzyżowanie**

* Jednopunktowe, wielopunktowe, jednorodne
* Tasowanie: losowa zamiana *pozycji* bitów w rodzicach przed krzyżowaniem (po krzyżowaniu pozycje są przywracane)
* Adaptacyjne: miejsca cięcia podlegają ewolucji, dynamiczny wybór operatora
* Z wieloma przodkami: liczba rodziców ~ różnorodność

**Strategie ewolucyjne**

* Dwuelementowa (1+1)-ES: osobnik mutant zastępuje przodka jeśli jest lepszy i dopuszczalny (podobnie do Local Search)
* Wieloelementowa: populacja i krzyżowanie dwóch osobników - potomek zastępuje najgorszego w populacji
* Wieloelem. (mu/rho , lambda)-ES: selekcja tylko z potomków (mu - n rodziców, rho - n rodziców potomków, lambda - n potomków)
* Wieloelem. (mu/rho + lambda)-ES: selekcja z rodziców i potomków
* Reprezentacja osobnika: wektor zmiennych, gdzie zmienna to jej wartość, odch. std. (stałe lub ewoluowane) i kąt odchylenia (opcjonalnie)
* Mutacja: zmiana wektora cech o losową wartość z rozkładu normalnego
* Krzyżowanie: jednorodne lub arytmetyczne
* CMA-ES: adekwatna strategia dla problemów źle uwarunkowanych (gdzie błędy numeryczne zbyt wpływają na wynik)

1. ustal środek populacji
2. próbkuj rozwiązania z macierzy kowariancji
3. oceń rozwiązania
4. przesuń środek populacji (rankingowa średnia ważona jakością)
5. rozproszenie nowych osobników jest proporcjonalne do prędkości z jaką przemieszcza się środek populacji
6. aktualizuj macierz kowariancji, żeby rozciągnąć rozkład w kierunku przemieszczenia środka populacji (podążanie za gradientem oczekiwanej jakości rozwiązań)

**Programowanie genetyczne**

* Generowanie drzew (np. do populacji początkowej): **Full** (drzewa o tej samej głębokości) / **Grow** (różna głębokość i kształt) / **Ramped half-and-half** (połowa pop. Full, połowa pop. Grow - *zapewnia zróżnicowanie pop. początkowej*)
* Krzyżowanie: wymiana losowych poddrzew u obu rodziców
* Mutacja: zastąpienie losowego poddrzewa losowo wygenerowanym poddrzewem
* Własności: **domknięcie** (funkcje działają dla dowolnego inputu), **wystarczalność** (elementy pozwalają na zbudowanie rozwiązania)
* Problemy: **puchnięcie** (rozwiązanie: kara za rozmiar w f-i celu / limit na rozmiar)

**Koewolucja**

* kilka gatunków wpływających na siebie; ocena osobników jednej pop. może zależeć od osobników w innej pop.)

**Koewolucja kooperacyjna**

* Optymalizacja złożonych problemów będzie skuteczniejsza, jeśli zostaną zdekomponowane na części
* Usuwanie gatunku: jeśli wkład gatunku do współpracy jest poniżej progu (różnica między fitnessem z nim i bez niego)
* Dodawanie gatunku: gdy stagnacja (brak wzrostu fitnessu)

**Koewolucja konkurencyjna**

* osobnik reprezentuje wiedzę o strategii, ocena np. przez wiele partii z pozostałymi osobnikami
* Problemy: chcemy ciągłej konkurencji (arms race), nie chcemy stagnacji (MSS - mediocre stable state), ciężko rozróżnić wygraną z przeciętnym i złym osobnikiem
* Rozwiązania problemów (cykle, nieprzechodność relacji porównania, brak postępu)
  + CFS (competitive fitness sharing): każdy osobnik ma jednostkę zasobu, którą dzieli po równo i oddaje osobnikom z którymi przegrał
  + HoF (hall of fame): każdy osobnik musi grać z każdym osobnikiem z hall of fame

**Reprezentacja zmiennoprzecinkowa**

* Krzyżowanie
  + Arytmetyczne: dziecko = r\_1\*p + r\_2\*(1-p)
  + Simpleksowe: c = centroid rodziców - najgorszy rodzic
* Mutacja
* Naprawianie mutacji
  + Pochłaniaj (clip), Powtarzaj (generuj aż wyjdzie), Zastępuj (generuj z losowego przodka aż wyjdzie)
  + Ignoruj (zostaw wartość przodka), Zawijaj (modulo), Flat (random uniform)
  + Odbijaj (jak x = d powyżej górnej granicy, to ustaw x = d poniżej górnej granicy; jak x = d poniżej dolnej granicy, to ustaw x = d powyżej dolnej granicy)

**Pytania “z dyskusji”**

* Czy do dobrego działania algorytmu ewolucyjnego potrzebny jest niedeterministyczny element: **NIE**
* Czy mutacja jest niezbędna - **TAK**, ponieważ zabezpieczenia algorytm przez zbyt szybką zbieżnością, umożliwia wyjście z lokalnego optimum, odpowiedzialna jest za eksplorację (wprowadzenie różnorodności)
* Czy krzyżowanie jest potrzebne: **NIE**, mutacja jest wystarczajaca, aczkolwiek umiejętne wykorzystanie krzyżowania pozwala nam zachować odpowiedni balans eksploracji/eksploatacji
* Sensowny sposób radzenia sobie z mutantami spoza dozwolonego przedziału: **ODBIJAJ**, bo nie zmienia rozkładu wartości i zachowuje podobieństwo do oryginalnej wart.
* Różnice między AE steady state i generacyjnym - w steady state tylko kilka osobników w populacji zostaje zmieniona, w generacyjnym wszystkie osobniki ulegają modyfikacji (selekcja i reprodukcja)
* **Epistaza to stopień zależności pomiędzy różnymi genami w chromosomie i ich łączny wpływ na fitness**. Dla skuteczności AE tym **lepiej, im mniejsza epistaza**, lecz dla niektórych definicji funkcji celu, reprezentacji i użytych operatorów, zjawisko epistazy jest korzystne. Ostatecznie w optymalizacji chodzi nie o niską epistazę, tylko o korzystny związek między topologią przestrzeni przeszukiwania, a krajobrazem przystosowania.
* Wielkość populacji ~ bezwładność algorytmu (przy większych pop. algorytm reaguje wolniej, ale mniejsze ryzyko utknięcia w opt. lokalnym)
* Prawdopodobieństwo mutacji: 1/liczba zmiennych decyzyjnych, albo 1/wielkość populacji
* Selekcja: losowa według reszt (bez lub z powtórzeniami), albo turniejowa

**Inne bzdury**

* Niepożądany algorytm genetyczny
* Operatory łączenia, cięcia i mutacji (prosta) łańcuchu bitów, selekcja turniejowa
* Eksperymentalnie dobre w problemach zwodniczych
* Hierarchiczny algorytm genetyczny
  + automatyczne odkrywanie zależnych elementów
  + próbkowanie specjalnie skonstruowanych rozwiązań pozwala zdekomponować
  + po dekompozycji niezależna optymalizacja
* Problem zwodniczy
* Ewolucja różnicowa:
  + -
* DPX (distance preserving crossover): dist(d, r1) = dist(d, r2) = dist(r1, r2)
* Globalna wypukłość
* FDC (fitness distance correlation)
  + Jeśli korelacja jest wysoka, to warto zaczynać od dobrych rozwiązań. Wtedy dywersyfikację można uzyskać niewielkimi zakłóceniami rozwiązań.
* Embriogeneza: mapowanie genotyp -> fenotyp
* Specjalizacja: dwa organizmy dobre w dwóch różnych zadaniach są lepsze niż dwa średnie w obu (a\*b > a/2 \* b/2)
* Specjacja: specjalizacja pod postacią podziału na gatunki o różnych niszach
  + podobne osobniki w okolicach ekstremum można nazwać gatunkiem
* Tworzenie skutecznych operatorów krzyżowania:
  + znajdź cechy wpływające na wartość f-i celu
  + stwórz miary odległości bazujące na tych cechach
  + jeśli miary determinują globalną wypukłość, wykorzystaj te cechy budując operator DPX