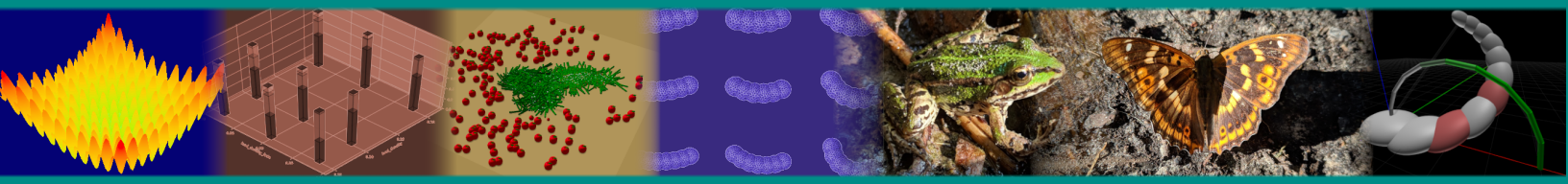
Metaheurystyki, Obliczenia Inspirowane Biologicznie i Sztuczne Życie

[cytowanie tego skryptu]

Maciej Komosiński

2025



Materiały pomocne w przypomnieniu większości zagadnień poruszanych na wykładach z **przedmiotu „Metaheurystyki i Obliczenia Inspirowane Biologicznie”**.

http://www.cs.put.poznan.pl/mkomosinski/site/?q=mioib

Spis treści

Rozdział 1

Optymalizacja

1.1 Przeszukiwanie wykorzystujące sąsiedztwo pojedynczego rozwiązania

Zanim przejdziemy do omówienia inspirowanych biologicznie algorytmów optymalizacji (w tym algorytmów ewolucyjnych), zapoznamy się z działaniem podstawowych, prostszych algorytmów [url].

- Podstawowe zagadnienia dotyczące optymalizacji:
  - `OptWprowadzenie.pdf` (ostatnie slajdy) https://youtu.be/uk-aWMHORvs
  - `MetaheurystykiPodsumowanie.pdf` (porównywanie algorytmów)
- Idea algorytmów lokalnej optymalizacji: `LS.pdf` https://youtu.be/oe94AHDQap0
- Algorytm symulowanego wyżarzania (1983): `SA.pdf` https://youtu.be/gX-X85dCib0
- Algorytm przeszukiwania tabu (1986): `TS.pdf` https://youtu.be/HsGJrSBFQNs

Niezłe wprowadzenie do optymalizacji globalnej oraz opis algorytmów omawianych na tym przedmiocie zawiera książka dostępna online [Wei09].

Metody optymalizacji globalnej można podzielić pod względem własności determinizmu [ENS99]:

- deterministyczne

  - siatki (*grid methods, covering methods*): w przestrzeni rozwiązań tworzy się siatkę (równomierną lub nie – nierównomierna sprzyja lepszej eksploatacji, bo gęstość siatki zależy od jakości rozwiązań), którą się przeszukuje
  - uogólnionego spadku (*generalized descent*)
    * trajektorii cząstki (*trajectory methods*): cząstka (rozwiązanie) porusza się w polu energetycznym wynikającym z optymalizowanej funkcji; strategia takiego poruszania się, choć może być chaotyczna[1], jest deterministyczna
    * kary (*penalty methods*): wielokrotnie uruchamia się metodę optymalizacji lokalnej zmieniając funkcję celu o człon kary, co zmienia krajobraz optymalizowanej funkcji i ma zapobiec zbieganiu do tego samego optimum lokalnego
  - deterministyczna wersja TS (*tabu search*)
- niedeterministyczne
  - poszukiwań losowych: Monte Carlo (MC) które w czystej postaci jest algorytmem losowym; CRS (*Controlled Random Search*) – połączenie MC i metody sympleksu nieliniowego; SA (*simulated annealing*)
  - z wykorzystaniem stochastycznego modelu funkcji celu[2]
  - populacyjne, np. algorytmy ewolucyjne

...ale podczas omawiania rozdziału 2.2.2 przeprowadzimy dyskusję o tym, co to znaczy, że algorytm ma niedeterministyczne elementy, co ta własność wnosi istotnego i kiedy jest niezbędna.

1.2 Dobór wartości parametrów; zastosowanie interakcyjne i wsadowe

Podobnie jak w wielu algorytmach sztucznej inteligencji, optymalne wartości parametrów zależą od charakteru rozwiązywanego zadania, którego jednak a priori (i zwykle również a posteriori) nie znamy. Dobór zależy również od zastosowania – interakcyjnego (*on-line*) lub wsadowego (*off-line*). Przy podejściu wsadowym zainteresowani jesteśmy najlepszym znalezionym podczas pracy algorytmu rozwiązaniem. Przy podejściu interakcyjnym („bieżącym") interesuje nas, by dany algorytm optymalizacji dawał jak najlepsze wyniki przez

[1] Chaos nie oznacza losowości, a wysoką wrażliwość na warunki początkowe.

[2] https://en.wikipedia.org/wiki/Stochastic_optimization, https://en.wikipedia.org/wiki/Stochastic_programming

cały czas. Dla oceny działania w trybach *on-line* i *off-line* De Jong zaproponował specjalne wskaźniki [Gol03, str. 125]; wymyśl dwa najprostsze.

Rozdział 2

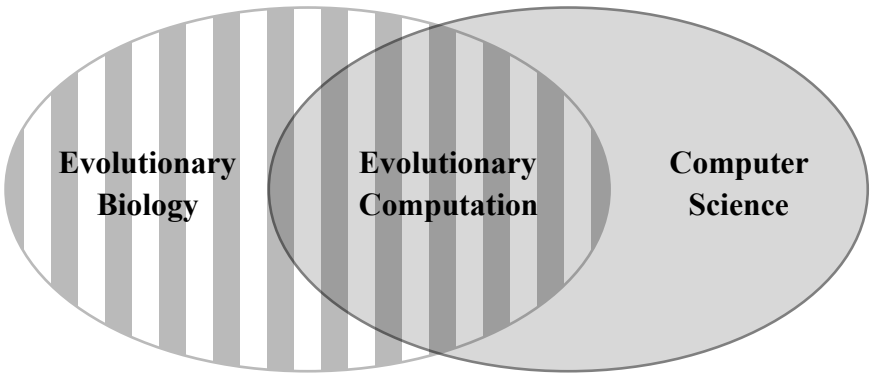
Algorytmy ewolucyjne

Materiały wideo do tego rozdziału:
https://www.youtube.com/playlist?list=PLl33p3GENNQHztKdRM1LVGd85VTDbo2Ub

2.1 Podział

Evolutionary Computation (EC) / Algorithms (EA)

EA is based upon biological observations that date back to Charles Darwin's discoveries in the 19th century: the means of natural selection and the survival of the fittest, and theories of evolution.



Rysunek 2.1: Obliczenia ewolucyjne jako część informatyki i biologii.

- Genetic Algorithms (GA)
- Evolution Strategies (ES)

- Evolutionary Programming (EP)
- Genetic Programming (GP)
- Classifier Systems (CFS), Genetic-Based Machine Learning (GBML)
- Various coevolutionary architectures
- ...

GA: created by John Holland (1973, 1975), made famous by David Goldberg (1989)
EP: created by Lawrence Fogel (1963), developed by his son, David Fogel (1992)
ES: created by Ingo Rechenberg (1973), promoted by Thomas Bäck (1996)
GP: developed by John Koza (1992)

Zastosowania AE:

- optymalizacja funkcji
- badania operacyjne – szeregowanie, optymalizacja, ...
- wielokryterialne wspomaganie decyzji
- przetwarzanie obrazów, rozpoznawanie wzorców
- algorytmy adaptacyjne w grach
- sterowanie; robotyka; projektowanie ewolucyjne/ewolucja konstrukcji
- biologia – symulacje (gatunków, populacji, ...
- nauki społeczne – symulacje grup
- sztuczne życie
- ...

2.2 Algorytmy genetyczne (genetic algorithms)

Wszystkie genotypy są wektorami binarnymi o takiej samej, stałej długości. Jeśli problem optymalizacji wymaga innej reprezentacji (np. permutacji), wtedy rozwiązania potrzebują kodowania, dekodowania i ew. naprawy. Operatory genetyczne są nieświadome oryginalnej reprezentacji rozwiązań, więc mogą być takie same dla każdego problemu optymalizacji. Porównajmy to do natury... bardzo różne gatunki, ten sam podstawowy kod.

2.2.1 Budowa algorytmu i parametry

Główna pętla:

$t := 0$
inicjalizuj $P(t)$
oceń $P(t)$
dopóki (**nie** warunek-zakończenia)
{
    $t := t + 1$
    wybierz $P(t)$ z $P(t-1)$
    zmień $P(t)$
    oceń $P(t)$
}

Parametry:

- wielkość populacji *POPSIZE*
- prawdopodobieństwo krzyżowania *PXOVER*
- prawdopodobieństwo mutacji *PMUT*
- wybór kryterium stopu
- wybór mechanizmu selekcji (pozytywnej i ew. negatywnej)
- wybór wartości parametrów mechanizmu selekcji

Proste demo: http://www.alife.pl/opt/p

Sposób tworzenia kolejnej puli genów w AG: „*steady state*" (algorytm stanu ustalonego/inkrementacyjny) albo „*generational replacement*" (pokoleniowy). W „*steady state GA*" nie wszystkie osobniki ulegają modyfikacji – część trafia do kolejnego pokolenia nie zmieniona. W szczególnym przypadku zmieniamy tylko jednego osobnika i algorytm działa płynnie, bez wyraźnie zaznaczonych pokoleń, i szybciej korzysta z nowopowstałych pomysłów (możliwa jest szybsza zbieżność).

2.2.2 Selekcja

Rola selekcji.

- Do czego potrzebna jest selekcja w algorytmie ewolucyjnym?
- Co działoby się, gdyby selekcja była czysto losowa?
- Co działoby się, gdyby selekcja była deterministyczna i dawała równą szansę każdemu osobnikowi?
- Czy można wyrazić siłę presji selekcyjnej liczbowo?

Dyskusja o rodzajach niedeterminizmu i roli niedeterminizmu w algorytmach i w informatyce.

Rozważ cztery generatory sekwencji losowych: oparty o kolejne liczby i *modulo*, pseudolosowy kiepski, pseudolosowy dobry, i prawdziwie losowy.

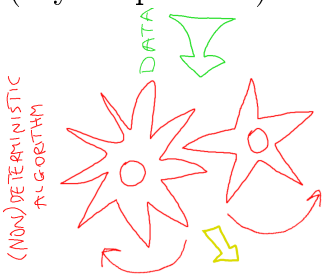
```
int random1(int n)
    static int c=-1
    c++
    c=c%n
    return c

int random2(int n)
    static int c=0
    c=(c·...+...)%n
    return c

int random3(int n)
    //very
    //complicated
    //logic
    return ...

int random4(int n)
    return ☼%n
```

- przykład suplementacji (np. 4 substancje co 2 dni każda, nie znamy interakcji)
- przykład dawania prezentów
- przykład *dithering*'u sygnału (dźwięk, obraz, ...) i zaokrąglania
- i wreszcie przykład algorytmu...

Rola niedeterminizmu w algorytmice – wnioski.

- wymyślasz jakiś algorytm, np. optymalizacyjny (albo inny). Co skłoni Cię do użycia w nim funkcji random()?
  - przykłady: *Greedy* i *Steepest* z wieloma sąsiadami o takiej samej jakości, indukcja drzew decyzyjnych z wieloma atrybutami o równej entropii, ...

  - spróbuj kompletne „udeterministycznić” SA. Czy będą jakieś negatywne konsekwencje?
- w jakich sytuacjach „nieskrępowana”, pełna losowość jest korzystna?
- dlaczego przejmujemy się ...1080777777696980348..., ale nie ...1080735172696980348...? Pierwsza sekwencja jest wprost z RNG, druga jest „poprawiona” (czy na pewno?)



- kiedy prawdziwa losowość jest preferowana względem dobrej pseudolosowości?
- i wreszcie: czy powinien dostać to finansowanie i dlaczego??

Od etapu reprodukcji oczekujemy powielenia osobników dobrych. Im większa będzie dominacja rozwiązań lepszych nad gorszymi (większy napór selekcyjny), tym mniejsza będzie różnorodność otrzymanej populacji. Te dwa aspekty selekcji (preferencja lepszych nad gorszymi oraz utrzymanie różnorodności) są ze sobą w pewnym stopniu sprzeczne, choć oba są też w pewnym stopniu pożądane.

Siłę presji selekcyjnej można wyrazić liczbowo, np. dzieląc prawdopodobieństwo wyboru najlepszego osobnika w populacji przez prawdopodobieństwo wyboru osobnika przeciętnego (mającego przystosowanie równe medianie przystosowań w populacji).

Sterowanie naporem selekcyjnym odbywa się np. za pomocą skalowania ocen osobników. Zbyt silny napór doprowadzi do przedwczesnej zbieżności (do optimum lokalnego), ponieważ osobniki najlepsze w danym momencie uzyskają przewagę liczebną i zdominują pozostałe rozwiązania. Z kolei niski napór selekcyjny zapewni dużą różnorodność osobników w populacjach, co może spowodować nieefektywność całej ewolucji i upodobnienie jej do przeszukiwania losowego.

Oznaczmy przez f_i ocenę osobnika i-tego ($i = 1..POPSIZE$), a przez e_i – liczbę jego oczekiwanych kopii w nowej (kolejnej) populacji, $e_i = POPSIZE \cdot f_i / \sum f_j$.

Najbardziej popularne techniki selekcji [Gol03] to:

- Wybór losowy proporcjonalnie do jakości z powtórzeniami (tj. ze zwracaniem), na-

zywany potocznie zasadą ruletki: osobnikom przydzielane są pola na tarczy ruletki, których wielkość jest proporcjonalna do ich ocen f_i. Następnie kręci się tarczą „ruletki" *POPSIZE* razy wybierając do nowej populacji wylosowanego osobnika. Tą samą zasadę realizuje metoda *stochastic universal sampling*[1], która zapewnia lepsze własności wyboru losowego.

- Wybór losowy według reszt bez powtórzeń: każdy osobnik otrzymuje tyle kopii w nowej populacji, ile wynosi część całkowita jego e_i. Pozostałe wolne miejsca zapełnia się podejmując losową decyzję, dla każdego osobnika z prawdopodobieństwem będącym częścią ułamkową jego e_i, czy ma trafić do nowej populacji. Przykład: 4 osobniki, $\boldsymbol{f}$=[1,3,5,6].
- Wybór według turniejów losowych: wybiera się losowo k osobników, a następnie zwycięzcę (osobnika o najwyższej ocenie) umieszcza się w nowej populacji i proces jest powtarzany aż do wypełnienia wszystkich miejsc w nowej populacji. Bardziej staranny wariant tej techniki dba o to, żeby każdy osobnik wziął udział w tej samej liczbie turniejów.

Inne techniki selekcji:

- Wybór deterministyczny według reszt: każdy osobnik otrzymuje tyle kopii w nowej populacji, ile wynosi część całkowita jego e_i, a pozostałe wolne miejsca w populacji zapełnia się w kolejności malejących części ułamkowych e_i osobników.
- Wybór losowy według reszt z powtórzeniami: każdy osobnik otrzymuje tyle kopii w nowej populacji, ile wynosi część całkowita jego spodziewanej liczby kopii (e_i). Pozostałe miejsca zapełnia się według zasady ruletki, wykalibrowanej według części ułamkowych e_i.
- Wybór porządkowy: osobnikom nadaje się rangi odpowiadające ich pozycjom w uszeregowaniu od najlepszego do najgorszego. Wybór następuje na podstawie funkcji prawdopodobieństwa określonej nie na ocenach osobników, a na ich pozycjach w rankingu. Stosuje się różne funkcje prawdopodobieństwa – linowe i nieliniowe, przy czym parametry tych funkcji pozwalają na sterowanie naporem selekcyjnym.

Zadanie: podziel wymienione 6 technik na dwie kategorie – zależnie od sposobu, w jaki wykorzystują wartości funkcji przystosowania.

Dodatkowe własności selekcji:

[1] https://en.wikipedia.org/wiki/Stochastic_universal_sampling

- Model elitarny (elitarystyczny): spełnia oczekiwanie, że proces selekcji nie powinien prowadzić do utraty najlepszego znalezionego dotąd osobnika. Jeśli taki osobnik nie trafia w naturalny (wynikający z zastosowanej metody selekcji) sposób do kolejnej populacji, włącza się go do niej i w ten sposób informacja o najlepszym dotychczasowym rozwiązaniu zostaje zawsze zachowana.
- Model ze współczynnikiem zatłoczenia (ze ściskiem): podobnie jak w naturze, gdzie gatunki wypełniające niszę ekologiczną muszą walczyć o ograniczone zasoby – w modelu ze ściskiem (ang. *crowding model*) nowe osobniki zastępują osobniki stare (z poprzedniej populacji) z uwzględnieniem ich podobieństwa, tzn. nowe osobniki zajmują miejsce najbardziej podobnych do nich starych osobników. Współczynnik ścisku (parametr) wpływa na sposób zastępowania osobników [DJ75; Mah92].

Meta-schematy selekcji:

W poniższych metodach selekcji, części populacji (podpopulacje) mogą być niezależnie przetwarzane – metody te mogą zatem pełnić również rolę schematu rozpraszania i zrównoleglania ewolucji.

- Model wyspowy: dzieli całą populację na podpopulacje, w których działa wybrany schemat selekcji (np. turniejowa, ruletkowa czy inny). Ewolucja odbywa się niezależnie na każdej wyspie, z okresową migracją części genotypów między wyspami. Ten model zwiększa zdolność eksploracji.
- Selekcja konwekcyjna: inaczej niż w tradycyjnym modelu wyspowym, podział na podpopulacje następuje wedle podobieństwa wartości funkcji celu rozwiązań. Selekcja konwekcyjna poprawia zdolność eksploracji AE dzięki odpowiedniemu zbalansowaniu presji selekcyjnej [KU17; KM18]. Animacje sposobu działania tutaj. Dyskusja: jak zadziała ten meta-schemat w wariantach *EqualNumber* i *EqualWidth* [KM18, Rys. 3], kiedy w podpopulacjach selekcja będzie losowa (np. rozmiar turnieju = 1), w porównaniu do modelu wyspowego i do standardowego, jednopopulacyjnego AE z losową selekcją?

Wymienione techniki selekcji mają swoje wady i zalety; w szczególności pierwsza z nich – metoda ruletki – cechuje się wysokim rozrzutem losowym, co powoduje duże różnice pomiędzy faktycznie uzyskiwanymi a oczekiwanymi liczbami osobników. Stąd powstało wiele technik (jak np. wybór losowy według reszt bez powtórzeń) pozbawionych tej wady. Wybór metody ma duży wpływ na zachowanie algorytmu, w szczególności na zdolność przekraczania siodeł[2] podczas optymalizacji [CG97].

[2] https://pl.wikipedia.org/wiki/Punkt_siod%C5%82owy

Niekiedy (zależnie od przyjętej architektury AG) oprócz zastosowania selekcji pozytywnej konieczne jest wykorzystanie również selekcji negatywnej. Jej rolą jest zrobienie miejsca w populacji na nowe genotypy: selekcja negatywna decyduje, które genotypy usunąć z populacji. Można stosować podobne mechanizmy, jak przy selekcji pozytywnej; dwie przykładowe, naiwne metody to usuwanie najgorszego i usuwanie losowego.

Przykładowe pytania

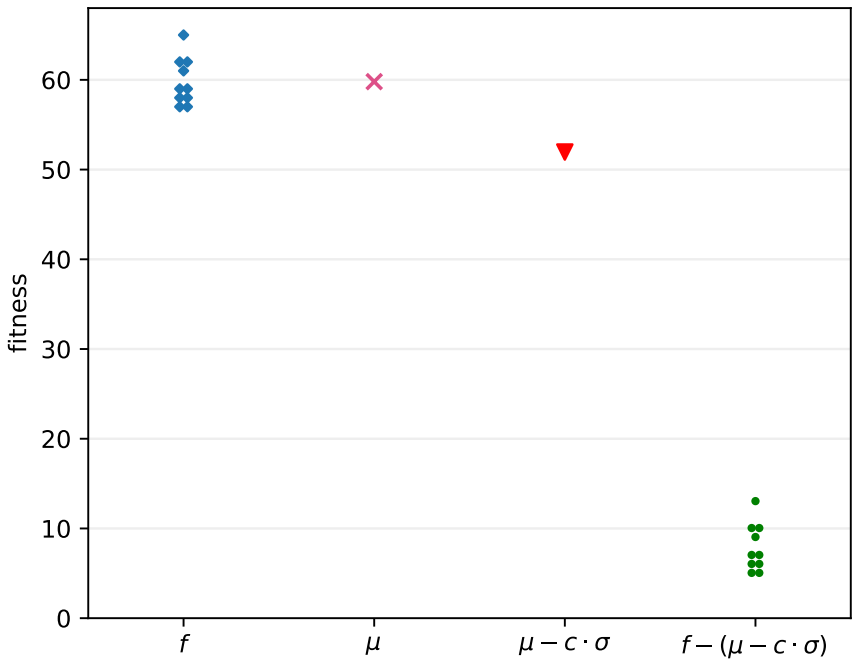
- Wymień i omów poznane metody selekcji.
- Jakie wady i jakie zalety mają poszczególne metody?

2.2.3 Skalowanie

Dyskusja: skąd wynika potrzeba skalowania? Analiza zachowania selekcji ruletkowej na początku ewolucji oraz w późniejszych etapach.

- Skalowanie liniowe: $f' = af + b$. Współczynniki a i b dobiera się w ten sposób, by przystosowanie f' osobnika najlepszego było zadaną wielokrotnością (na przykład $2\times$) dopasowania osobnika „średniego". Po skalowaniu mogą pojawić się ujemne wartości przystosowania – można wówczas wyzerować je lub dokonać innego przekształcenia liniowego.
- Skalowanie potęgowe: $f' = f^k$. Współczynnik k zależy od konkretnego rozwiązywanego zadania, stąd metoda ta nie jest specjalnie przydatna.
- Skalowanie σ-obcinające (obcinanie na poziomie zależnym od odchylenia standardowego). Wartości przystosowania zależą nie tylko od wartości pierwotnych ocen osobników, ale także od rozkładu ocen w populacji. Wyznacza się średnie dopasowanie populacji μ oraz odchylenie standardowe ocen w populacji σ, a przystosowanie (w przypadku maksymalizacji) $f' = f - (\mu - c \cdot \sigma)$. Ujemne wartości f' zastępuje się zerem. Współczynnik c określa napór selekcyjny: im większe c, tym mniejszy napór.

Np. 10 osobników o ocenach 57, 57, 58, 58, 59, 59, 61, 62, 62, 65. Średnia μ wynosi 59.8, a odchylenie standardowe $\sigma = 2.6$.



Rysunek 2.2: Wykres skalowania (z odcięciem na poziomie zależnym od odchylenia standardowego) dla współczynnika $c = 3$.

Zadanie obliczeniowe (i interpretacyjne): przed skalowaniem, prawdopodobieństwo wybrania osobnika i-tego p_i to iloraz wartości dopasowania osobnika (f_i) i sumy dopasowań wszystkich osobników ($j = 1..n$).

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j} = \frac{f_i}{n\mu}$$

Ile wyniesie p'_i, jeśli $f'_i = f_i - (\mu - c\sigma)$?

Wyraź p'_i jako funkcję p_i i innych możliwie łatwo interpretowalnych wyrażeń.

Odpowiedź:

$$p'_i = \frac{f_i - \mu}{nc\sigma} + \frac{1}{n} \quad = \quad \frac{\mu}{c\sigma}\left(p_i - \frac{1}{n}\right) + \frac{1}{n}$$

Zinterpretuj powyższy wzór i odnieś do Rys. 2.2. Czym jest $\frac{1}{n}$? Rozważ sytuację, gdzie μ jest małe a σ duże (początek ewolucji) oraz odwrotną, gdy ewolucja jest już zaawansowana.

Przypomnienie: jaki brzmiał jeden z pomysłów pozwalający reprezentować „presję selekcyjną" jako pojedynczą wartość?

Mechanizm skalowania pomaga utrzymać stały napór selekcyjny podczas całej ewolucji, niezależnie od własności optymalizowanej funkcji. Kontrolowanie naporu selekcyjnego jest bardzo ważne – bez tego niemożliwa jest efektywna optymalizacja.

Stosowanie skalowania ma sens, jeśli użyta metoda selekcji wykorzystuje ilorazowość skali wartości przystosowania (a np. selekcja turniejowa i rankingowa nie wykorzystują).

Inna (lepsza niż σ-obcinające?) funkcja skalowania: prawdopodobieństwo wybrania osobnika,

$$p'_i = \frac{1}{1 + \exp(\frac{f_i - M}{\sigma})}$$

f_i – wartość funkcji celu osobnika,
M – **mediana** funkcji celu wszystkich osobników w populacji,
σ – odchylenie standardowe funkcji celu w populacji.

Dla maksymalizacji należy odwrócić znak argumentu exp().

Zalety:

- Minimalny wpływ krańcowo dobrze/źle przystosowanych osobników (w ciągu całego procesu ewolucji), więc znika problem przedwczesnej zbieżności. Dlaczego minimalny? Bo użyto mediany, a nie średniej. Krańcowy osobnik wpłynie tylko na umiarkowane zwiększenie σ.

- Skuteczne rozdzielenie osobników przeciętnie przystosowanych na dwie grupy (powyżej i poniżej mediany).
- Na początku ewolucji zwykle występuje duże σ, czyli małe zróżnicowanie p'_i osobników. Pod koniec ewolucji – zbieżność więc małe σ, zatem osobniki lepsze dużo silniej promowane.

Aby zobaczyć przykładowy rozkład wartości funkcji oceny w populacji podczas ewolucji, zobacz [KU17, górny wykres na rys. 9]. Jak wyjaśnisz utrzymujące się przez długi czas poziome linie (grupy osobników o podobnych, szczególnych wartościach funkcji celu)?

2.2.4 Krzyżowanie

Dyskusja: czy krzyżowanie jest niezbędne w algorytmie ewolucyjnym?

Metody klasyczne:

- Krzyżowanie proste – jednopunktowe.

  ```
  a b c d e f|g h        a b c d e f G H
  A B C D E F|G H        A B C D E F g h
  ```

- Uogólniony, wielopunktowy operator krzyżowania (parametrem jest liczba punktów cięcia):
  - Dla parzystej liczby cięć łańcuchy bitów traktuje się jako zamknięte pierścienie.
  - Dla nieparzystej liczby (jak np. 1 w krzyżowaniu prostym) uwzględnia się początek lub koniec łańcucha bitów jako jeden z krańców zamienianego odcinka.

  Doświadczenia dowiodły, że zwiększanie liczby punktów cięcia dla pewnych reprezentacji rozwiązań pogarsza osiągi algorytmu genetycznego. Przyczyną jest większy wpływ niszczący takiego krzyżowania: im więcej punktów cięcia, tym intensywniej rozrywane są wartościowe schematy (zob. twierdzenie o schematach, 2.2.7).
- Krzyżowanie odcinkowe, w którym ustalony współczynnik zmiany odcinka s określa prawdopodobieństwo wystąpienia punktu cięcia w danym punkcie genotypu. Dla genotypu o długości l oczekujemy $l \cdot s$ punktów cięcia, choć liczba ta waha się ze względu na czynnik losowy (prawdopodobieństwo s).
- Krzyżowanie jednorodne. Dla każdego bitu powstającego potomka pierwszego decyduje się (z prawdopodobieństwem p), od którego z dwóch rodziców otrzyma on dany bit. Drugi potomek otrzymuje bit od pozostałego rodzica. Przy $p = \frac{1}{2}$ potomkowie

mają równe szanse na odziedziczenie poszczególnych bitów od jednego lub drugiego rodzica. Gdy p maleje, potomkowie upodabniają się do rodziców. Krzyżowanie jednorodne jest jeszcze bardziej szczegółowe od wielopunktowego – wymianie podlegają bity, nie odcinki. Dla zadań, w których wzajemna lokalizacja bitów nie gra roli, taki rodzaj krzyżowania może być korzystny.

Metody bardziej zaawansowane:

- *Tasowanie*: dodatkowy mechanizm [ECS89] stosowany przy niektórych rodzajach krzyżowania. Polega on na losowej zamianie miejsc bitów w łańcuchach rodziców (u obu tak samo), dokonaniu krzyżowania i przywróceniu pierwotnego porządku bitów u potomków (zastanów się, po co stosuje się taką operację i dla jakich operatorów krzyżowania ma sens?)
- Krzyżowanie adaptacyjne: w genotypie oprócz bitów rozwiązania mogą być przechowywane informacje o miejscach, w których nastąpiło krzyżowanie. Rozwiązania słabe zanikają (wraz z informacją o miejscach krzyżowania), podczas gdy rozwiązania dobre (i informacja o korzystnych punktach krzyżowania) utrzymują się i ulepszają. Tak więc miejsca cięcia podlegają ewolucji wraz z populacją rozwiązań. Rozszerzeniem tego pomysłu jest rejestrowanie dopasowania (oceny) różnych operatorów genetycznych i wybór operatora z uwzględnieniem jego dopasowania.
- Krzyżowanie z wieloma przodkami: rekombinacji podlegają geny losowane z puli genów rodziców uzyskanych w wyniku selekcji, włączając w to przypadek tzw. „orgii", w których rozwiązanie-potomek może mieć więcej niż dwóch rodziców (zastanów się, jaki wpływ ma liczba rodziców?)

Możliwe jest również wykonywanie operacji krzyżowania i mutacji jednocześnie – utworzenie operatora rekombinacji wykonującego te dwie operacje w jednym przebiegu.

Podczas ewolucji chcemy uniknąć nadmiernego podobieństwa osobników (i utrzymać różnorodność populacji), a zarazem wymusić odpowiedni napór selekcyjny. Ponieważ operacja krzyżowania działa w kierunku ujednolicania populacji, umiejętne jej stosowanie (w obecności mutacji) może służyć utrzymaniu odpowiedniej różnorodności i właściwego naporu selekcyjnego. Miarą różnorodności osobników może być ich entropia; korzystne jest wówczas uzależnienie od niej prawdopodobieństwa krzyżowania i/lub mutacji.

2.2.5 Mutacja

Dyskusja: czy mutacja jest niezbędna w algorytmie ewolucyjnym?

Mutacja zapobiega przedwczesnej zbieżności oraz utracie fragmentów rozwiązania z powodu

zbytniego ujednolicenia populacji. Mutacja prosta polega na zamianie bitu na przeciwny z pewnym prawdopodobieństwem. W początkowej fazie ewolucji korzystne jest zwykle wysokie prawdopodobieństwo mutacji, które zapobiega zbieżności w kierunku lokalnego optimum; pod koniec oczekuje się dokładniejszego dostrojenia do najlepszego rozwiązania. Stąd wniosek, iż prawdopodobieństwo mutacji powinno spadać wraz z upływem czasu – np. malejąca funkcja numeru pokolenia (co przypomina nieco efekt „temperatury" w SA). Innym rozwiązaniem jest uzależnienie tempa mutacji od różnorodności populacji (podobnie, jak przy krzyżowaniu). Wówczas zbytnie ujednolicenie populacji będzie sygnałem do zwiększenia prawdopodobieństwa mutacji.

Z reguły korzystne jest to, że po mutacji osobnik jest podobny w pewnym stopniu do rodzica, oraz że każda mutacja powoduje zmiany zbliżonej wielkości. Czy taka sytuacja ma miejsce, kiedy zastosujemy standardowe kodowanie binarne liczb? Nie. Mutacje będą się zdarzać raz na bitach mniej, raz bardziej znaczących, powodując czasem niewielkie zmiany, czasem ogromne.

Jeśli zastosujemy kod Gray'a, w którym sąsiednie liczby różnią się tylko jednym bitem (odległość Hamminga), to zawsze sąsiednie liczby są osiągalne przez pojedynczą mutację. Ale czy to coś pomoże?

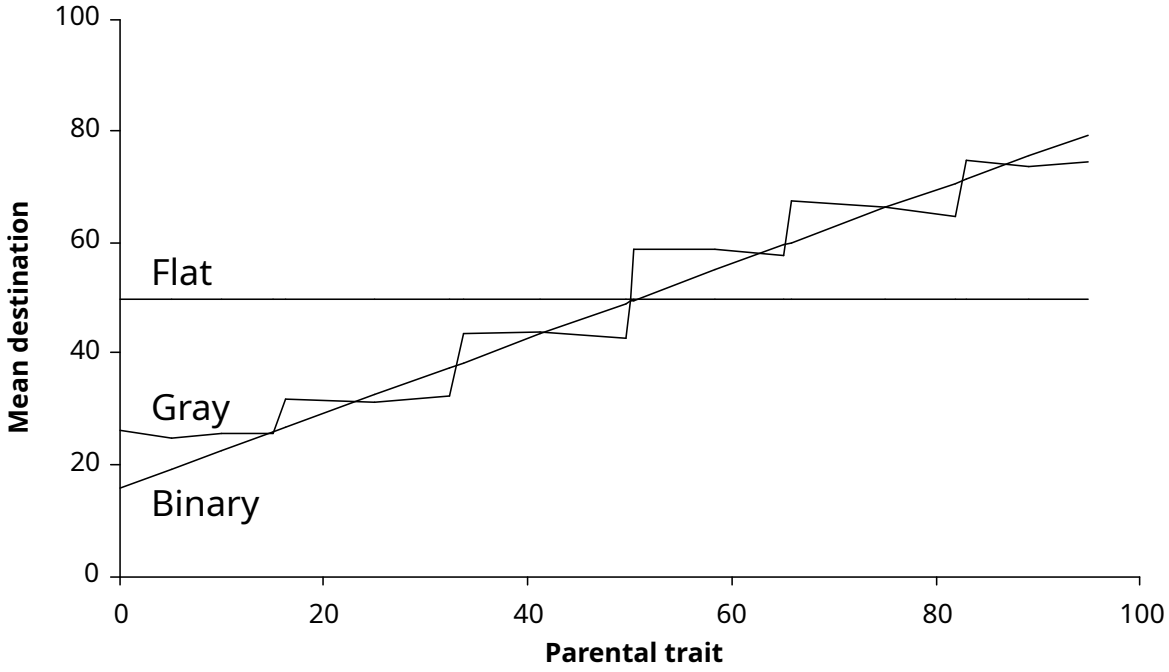
Przedstawienie graficzne: bity = osie X, Y, Z.

	bit 0	bit 1	bit 2
0	0	0	0
1	1	0	0
2	1	1	0
3	0	1	0
4	0	1	1
5	1	1	1
6	1	0	1
7	0	0	1

cykl Hamiltona 3D (graniczne różnią się też o 1)

	bit 0	bit 1	bit 2
0	0	0	0
1	1	0	0
2	1	1	0
3	0	1	0
4	0	1	1
5 (*)	0	0	1
6	1	0	1
7 (*)	1	1	1

ścieżka Hamiltona 3D



Rysunek 2.3: Silne nieciągłości operatora mutacji mają stały wpływ (*bias*) na ewolucję [Bul99, str. 69]. Flat: mutacja polega na wylosowaniu wartości genu z dozwolonego przedziału. Kod Gray'a nie pozwala na uniknięcie wad związanych z mutowaniem zakodowanych wartości liczbowych.

Uwaga: przy kodowaniu Gray'a każda mutacja zmienia wartość genu, ale także jego parzystość. Jeśli *wartość genu* decyduje wraz z jego **parzystością** o jakiejś właściwości (np. *nasilenie* **lewo/prawo**-ręczności), to z każdą mutacją będzie się zmieniał i *stopień* „ręczności", i jej **kierunek** (lewo/prawo).

Dyskusja: jakie kodowanie spełniłoby oczekiwanie, że każda mutacja zmienia zakodowaną wartość o ± 1?

2.2.6 Podsumowanie parametryzacji AG

Wprowadzanie różnych udoskonaleń i rozszerzeń AG (przykład: adaptacyjne prawdopodobieństwa mutacji) prowadzi niekiedy do powstania kolejnych parametrów. Z drugiej strony część takich modyfikacji pozwala na automatyczne dostrojenie wartości parametrów. Nowe parametry są często łatwiej interpretowalne i/lub ich wpływ na działanie algorytmu i uzyskiwane wyniki staje się bardziej przewidywalny.

Dyskusja: wyobraź sobie, że masz jakiś konkretny problem optymalizacji. Jakie wartości parametrów ustalisz i jakie mechanizmy wybierzesz? Jakimi kryteriami kierujesz się wybierając akurat te, a nie inne wartości?

Wielkość populacji ma wpływ przede wszystkim na bezwładność algorytmu. Przy większych populacjach algorytm reaguje wolniej, dlatego przy zastosowaniach *on-line* zaleca się względnie małe populacje. Duże populacje, szczególnie na początku ewolucji, potrzebują więcej czasu by dotrzeć do rozwiązań dobrych. Z kolei dla zastosowań *off-line* duże populacje są korzystne, ponieważ dysponują większą ilością informacji i pozwalają na dokładniejsze przeszukanie przestrzeni rozwiązań. Duża liczba osobników zmniejsza zarazem ryzyko utknięcia w optimum lokalnym. Trzeba jednak tak dobrać wielkość populacji, żeby zdążyła ona zbiec do regionu dobrych rozwiązań w dostępnym czasie.

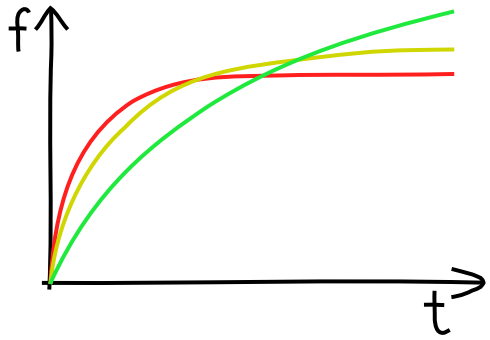
Prawdopodobieństwo krzyżowania wpływa na stopień wymiany informacji w populacji, a także na jej ujednolicanie. Typowa wartość zalecana w prostym AG, by uzyskać kompromis pomiędzy efektywnością *off-line* i *on-line*, to 60%. Przy stosowaniu metod selekcji o niskim rozrzucie losowym (tzn. bardziej deterministycznych, niż np. reguła ruletki) zaleca się wyższe prawdopodobieństwo krzyżowania – nawet 100% [Gol03, str. 130]. Te zalecenia trzeba jednak traktować z dużym dystansem – trudno podać uniwersalną, dobrą wartość nie znając znaczenia rozwiązania w danym problemie optymalizacji, sposobu kodowania binarnego i charakterystyki (skuteczności) operatora krzyżowania.

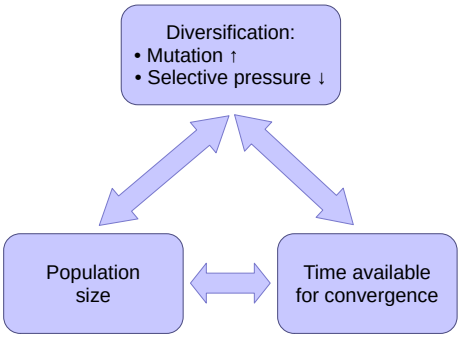
Prawdopodobieństwo mutacji odpowiada za liczbę losowych zmian bitów osobników. Zaleca się, by było ono odwrotnością liczby zmiennych decyzyjnych (jeśli zmienne decyzyjne są binarne, to mutacji ilu genów spodziewamy się w każdym pokoleniu?). Czasem spotyka się prawdopodobieństwo mutacji będące odwrotnością liczby osobników w populacji (mutacji ilu genów spodziewamy się w każdym pokoleniu?)

Mechanizm selekcji o właściwościach korzystniejszych od mechanizmu ruletki to opisany wcześniej wybór losowy według reszt, bez powtórzeń lub z powtórzeniami, albo metody turniejowe. Polecany mechanizm skalowania to odcięcie na poziomie zależnym od odchylenia standardowego, ewentualnie połączone ze skalowaniem liniowym.

Dobrym kryterium stopu przy zastosowaniach *off-line* jest liczba pokoleń bez poprawy (chociaż bez wprowadzenia progu minimalnej zmiany algorytm może działać zbyt długo z powodu ciągłych, nieznacząco małych polepszeń). Można też monitorować średnie podobieństwo w populacji i kończyć optymalizację, gdy zaniknie różnorodność.

Interakcje parametrów:





2.2.7 Twierdzenie o schematach

Twierdzenie o schematach[3] (ang. *schema theorem*) opisuje zasadę działania algorytmów ewolucyjnych. Dla AG, „schemat”[4] to ciąg symboli 0, 1 oraz * (gwiazdka zastępuje 0 lub 1). Dla specyficznego AE, jakim jest AG z mutacją prostą i krzyżowaniem prostym, określamy *rząd schematu* jako liczbę ustalonych pozycji w schemacie, oraz *rozpiętość schematu* jako odległość między jego skrajnymi ustalonymi pozycjami.

Przystosowaniem schematu jest średnia z przystosowania wszystkich pasujących do niego genotypów. Dość proste obliczenia prowadzą do wniosku–twierdzenia, że *liczba genotypów pasujących do schematów posiadających wyższe od średniego przystosowanie, niski rząd i małą rozpiętość, będzie rosła wykładniczo w kolejnych pokoleniach.*

Hipoteza cegiełek mówi z kolei, że za efektywnym działaniem AE stoi zdolność do korzystnego łączenia krótkich ciągów genów (efekt synergii), zobacz też Epistaza (str. 26), Tasowanie (str. 20), Inwersja (str. 63), Nieporządny AG (rozdział 2.2.9).

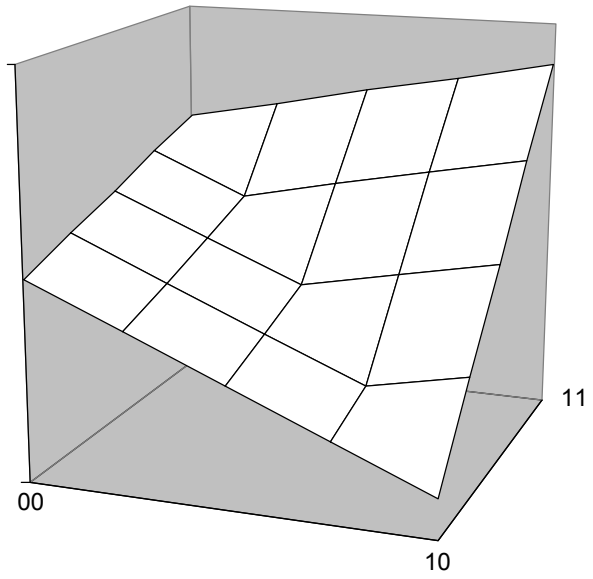
2.2.8 Problemy zawodne (trudne dla AG) i twierdzenie „No Free Lunch”

NFL: dla wszystkich problemów optymalizacji jakie mogą teoretycznie zaistnieć, skuteczność wszystkich algorytmów jest średnio taka sama (analogiczne twierdzenia obowiązują w uczeniu maszynowym, kompresji danych, itd.). Rozważ konsekwencje tego twierdzenia i porównaj: hiperheurystyki z rozdziału 2.6.2.

Aby skonstruować taki problem, który sprawi trudność algorytmowi genetycznemu, postarajmy się zanegować hipotezę cegiełek. Chcemy, by dobre „cegiełki” po połączeniu tworzyły niekorzystną strukturę. Najprostszy taki przypadek można uzyskać dla genotypów o długości 2 (jest to problem zwodniczy rzędu drugiego).

Załóżmy, że istnieją cztery schematy:

[3] https://en.wikipedia.org/wiki/Holland%27s_schema_theorem
[4] https://en.wikipedia.org/wiki/Schema_(genetic_algorithms)



Rysunek 2.4: Przykładowy problem zwodniczy rzędu drugiego, typu I.

```
*** 0 *** 0 ***
*** 0 *** 1 ***
*** 1 *** 0 ***
*** 1 *** 1 ***
```

Gwiazdki odpowiadają dowolnej liczbie symboli nieustalonych (ale pozycje ustalone są we wszystkich schematach na tych samych miejscach). Oznaczmy średnie oceny schematów przez f_{00}, f_{01}, f_{10}, f_{11} i schemat z dwoma jedynkami jest globalnym optimum (f_{11}). Aby problem był zwodniczy, chcemy, by schematy z jednym ustalonym zerem były lepsze od odpowiednich schematów z jedynką, to znaczy, by spełniona była przynajmniej jedna z nierówności:

$f_{0*} > f_{1*}$ $\qquad\qquad$ $f_{0*} = (f_{00} + f_{01})/2$, itd.
$f_{*0} > f_{*1}$

Taki problem nazywany jest minimalnym problemem zwodniczym (ang. *minimal deceptive problem, MDP*), ponieważ nie istnieje problem zwodniczy rzędu pierwszego. Są dwa typy problemów zwodniczych rzędu drugiego (por. Rys. 2.4):

Typ I: $f_{01} > f_{00}$
Typ II: $f_{00} \geq f_{01}$

W *MDP* funkcja przystosowania *nie może* być przedstawiona jako liniowa kombinacja poszczególnych alleli, czyli w postaci $f(x_1, x_2) = b + \sum_{i=1}^{2} a_i x_i$.

To, że problem jest zwodniczy nie oznacza jeszcze, iż algorytm genetyczny nie znajdzie optimum. Oznacza to jednak, że funkcja przystosowania (przedstawiona na osi pionowej) nie może być wyrażona jako liniowa kombinacja poszczególnych bitów, zatem występuje zjawisko epistazy – nieliniowości. Zwykle taki problem nie okazuje się jednak AG-trudny, czyli algorytm genetyczny może znaleźć rozwiązanie optymalne. Zachowanie algorytmu zależy jednak od wielu czynników, takich jak na przykład początkowa obecność schematów w populacji osobników. W szczególności, jeśli wszystkie cztery schematy występują w populacji początkowej, problem zwodniczy rzędu 2 typu I nie jest AG-trudny. Dla zadań typu II zachowanie algorytmu zależy od proporcji występowania schematów w populacji: jeśli schemat `00` będzie występował w przewadze, algorytm może być zbieżny do rozwiązania nieoptymalnego (choć taka sytuacja występuje rzadko). Bardziej szczegółowa analiza problemów zwodniczych znajduje się w [Gol03, str. 63–70, 375–382].

Powyższe rozważania dotyczyły klasycznego algorytmu genetycznego, z krzyżowaniem prostym i mutacją prostą. Przedstawiony problem zwodniczy jest reprezentantem zadań, które mogą sprawiać trudność algorytmom genetycznym z powodu występowania izolowanych optimów. Oczywiście takie problemy stanowią również trudność dla innych algorytmów optymalizacji.

Zaproponowano wiele udoskonaleń algorytmów genetycznych, które mają zapobiegać opisanej sytuacji. Można jej uniknąć stosując specyficzne kodowanie (potrzebna jest wtedy wiedza o optymalizowanej funkcji), można zastosować operator inwersji (str. 63), można również stosować nieporządne (rozdział 2.2.9) lub hierarchiczne (rozdział 2.2.10) algorytmy genetyczne.

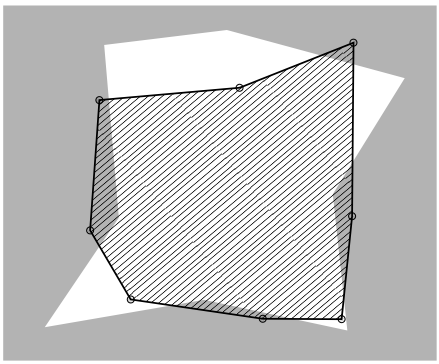
Epistaza (ang. *epistasis*): cecha reprezentacji (i ew. operatorów dla niej) – stopień zależności pomiędzy różnymi genami w chromosomie [Dav90]. Jeśli dana reprezentacja posiada wysoką epistazę, fenotypowy efekt pewnych genów zależy od alleli (wartości) innych genów (poligeniczność).

Zerowa epistaza: każdy gen niezależnie wpływa na wartość funkcji celu (i wtedy nie warto stosować AE). Dla skuteczności AE tym lepiej, im mniejsza epistaza. Dla niektórych definicji funkcji celu, projektując reprezentację i operatory może opłacać się zaakceptować niewielki wzrost epistazy, jeśli zyskamy korzystniejszy związek między topologią przestrzeni przeszukiwania, a krajobrazem przystosowania (por. Rys. 2.8).

Oszacuj mniej więcej epistazę dla następujących funkcji celu $f(x_1, x_2)$, gdzie x to liczby – wartości genów: $x_1 + x_2$, $x_1 - x_2$, $x_1 \cdot x_2$, $x_1 + x_2 \cdot x_2$, $x_1 + x_1 \cdot x_2$, $\frac{x_1}{x_2}$, $x_2 + \frac{x_1}{x_2}$.

Przykład.

Optymalizujemy kształt figury 2D (albo przekrój obiektu 3D); geny opisują rozmieszczenie wierzchołków w przestrzeni (współrzędne x_i, y_i).



Porównaj epistazę dla następujących scenariuszy:

1. Same geny x_i, y_i oraz operator mutacji przesuwający losowy wierzchołek.
2. Wprowadzamy dwa dodatkowe geny: obrotu i skali. Mutacja tych pojedynczych genów zmienia orientację (obrót) i rozmiar całej figury.
3. Zamiast tych dwóch dodatkowych genów wprowadzamy specjalne operatory mutacji-obrotu i mutacji-skalowania.

W scenariuszu (1) zmiana orientacji i rozmiaru figury jest możliwa jedynie przez bardzo wiele niezależnych mutacji współrzędnych wierzchołków. Tymczasem w pewnych zadaniach (np. w optymalizacji online, gdy docelowy kształt otworu się zmienia), możliwość szybkiego obrotu i skalowania może przyspieszyć zbieżność i polepszyć jakość uzyskiwanych rozwiązań. W scenariuszu (2) rozważ wpływ konkretnej implementacji (matematyczne działania przekształcające genotyp $\rightarrow$ fenotyp) – chodzi o efekty złożenia mutacji (1) i (2).

———

Epistazę, jako stopień interakcji pomiędzy genami, można zamodelować tak [RW95; NK00]:

$$
\begin{aligned}
f(s) = {} & \text{stała} \\
& + \sum_{i=0}^{l-1} \text{efekt } s_i \\
& + \sum_{i=0}^{l-2} \sum_{j=i+1}^{l-1} \text{interakcja między } s_i \text{ i } s_j \\
& + \ldots \\
& + \text{interakcja między } s_0, s_1, \ldots, s_{l-1} \\
& + \text{losowy błąd.}
\end{aligned}
$$

s_i – gen i-ty,
l – liczba genów (numerowane od zera).

Interakcje mogą być dodatnie i ujemne: jeśli np. interakcja pomiędzy s_i i s_j ma taki sam znak, co s_i i s_j, to taka interakcja wzmacnia działanie tych dwóch genów i może być pożądana.

Problemy optymalizacyjne sprawiają trudność algorytmom ewolucyjnym (i innym algorytmom optymalizacji) z trzech powodów:

- *isolation* (igła w stogu siana)
- *deception* (nieskuteczność współpracy schematów/cegiełek budujących)
- *multimodality* (liczne optima lokalne)

Dwie ostatnie cechy nie są ani konieczne, ani wystarczające, by problem był GA-trudny.

Istnieją różne miary **przewidywania trudności** problemu dla algorytmu ewolucyjnego; znane są w szczególności proste miary *zmienności epistazy* oraz *korelacji odległości od optimum z wartością dopasowania* (por. rozdział 2.5.2). Praca [NK00] zawiera ich definicje, charakterystyki oraz dyskusję skuteczności.

Aby oszacować trudność problemu próbkuje się przestrzeń rozwiązań, a potem oblicza się funkcję oceny dla wybranych rozwiązań. Następnie wylicza się różne wskaźniki, biorące pod

uwagę odległość pomiędzy rozwiązaniami (według przyjętej reprezentacji genetycznej) i między ich ocenami. Obecnie miary te nie są jednak zadowalające: da się skonstruować takie problemy, dla których owe wskaźniki zawodzą (problem okazuje się dla algorytmu optymalizacji łatwy, jednak według miar jest trudny, lub na odwrót). Bardziej wiarygodne są miary biorące pod uwagę zachowanie AE podczas rozwiązywania danego problemu (tzw. miary *on-line*), jednak ich podstawową wadą jest złożoność – czas obliczeń potrzebny do oceny problemu.

2.2.9 Nieporządny algorytm genetyczny

Ang. *messy genetic algorithm*.

Podejście „nieporządne" (niechlujne) ma na celu, podobnie jak opisany na str. 63 operator inwersji, polepszenie własności algorytmu genetycznego poprzez skuteczniejsze konstruowanie, wykorzystanie i przekształcanie schematów. Nieporządny algorytm genetyczny [Gol+93][Mic96, str. 122] wykorzystuje szczególną reprezentację osobników: genotypy są zmiennej długości, złożone z par (*etykieta*, *wartość*). Etykieta to opis znaczenia genu – podobnie jak w operatorze inwersji, etykieta może być pierwotnym numerem genu. Dozwolone są genotypy niekompletne (niedospecyfikowane), tzn. nie określające wartości wszystkich genów. Genotyp może zawierać również geny nadmiarowe, a nawet sprzeczne.

Używa się trzech operatorów: cięcia, łączenia i mutacji. Operator cięcia rozcina z pewnym prawdopodobieństwem, w losowo wybranym miejscu, łańcuch bitów. Operator łączenia skleja z pewnym prawdopodobieństwem dwa genotypy. Mutacja jest identyczna z mutacją prostą.

Stosowana jest selekcja turniejowa. Proces ewolucji składa się z dwóch (potencjalnie wielokrotnie wykonywanych) faz: wyboru bloków budujących i stosowania operatorów. Liczebność populacji jest zmienna w trakcie działania algorytmu.

Nadmiarowość informacji w genotypie można prosto rozwiązać wybierając pierwszą napotkaną wartość danego genu w genotypie, ale istnieją też inne metody – np. uśrednianie wszystkich wartości genu albo stosowanie pewnego rodzaju głosowania, którą wybrać. Niedoprecyzowane genotypy, o ile nie są akceptowalne w danym problemie optymalizacji, rozwiązuje się uzupełniając brakujące geny najlepszymi znanymi wartościami danego genu z wcześniejszej fazy algorytmu.

Nieporządne algorytmy genetyczne w problemach zwodniczych działały kilkakrotnie lepiej od klasycznych algorytmów genetycznych z krzyżowaniem punktowym.

2.2.10 Hierarchiczny algorytm genetyczny

Dyskusja: czy można w jakiś sposób „wykryć" epistazę?

Podobnie jak w przypadku nieporządnych algorytmów genetycznych, motywacją do rozwoju H-GA była chęć automatycznego odkrywania stopnia współzależności (ang. *linkage*, nie mylić z genetic linkage) elementów rozwiązania w celu dekompozycji problemu. Poprzez próbkowanie specjalnie skonstruowanych rozwiązań można z pewnym prawdopodobieństwem określić zależność lub niezależność genów i grup genów, a następnie dla wykrytych niezależnych grup (modułów) prowadzić niezależną optymalizację [JTW04].

Aby zbadać w pełni niezależność dwóch genów od reszty rozwiązania, należałoby wygenerować zbiór rozwiązań, w których wszystkie możliwe pary wartości tych dwóch genów są otoczone wszystkimi możliwymi wartościami pozostałych genów (stanowiących „kontekst"). Następnie wszystkie te rozwiązania należałoby ocenić i wyznaczyć zależność między wartościami genów a wartością funkcji celu. Byłoby to bardzo kosztowne obliczeniowo, a przecież to byłby tylko test dla jednej pary genów! Dlatego też stosuje się próbkowanie, które umożliwia oszacowanie potencjalnej niezależności dla wszystkich podzbiorów genów w rozwiązaniu.

Wizja automatycznej dekompozycji problemu optymalizacji jest bardzo atrakcyjna – pozostaje kwestia metod i ich wydajności, dlatego to zagadnienie jest wciąż aktywnie badane.

Wykrywanie zależności: techniki statystyczne i empiryczne

Metody wykrywania zależności między genami dzieli się czasem na statystyczne (np. H-GA albo rodzina metod GOMEA) i empiryczne (np. DLED). Te pierwsze bazują na osobnikach w populacji oraz ich ocenach i na tej podstawie statystycznie szacują niezależność genów. Te drugie próbkują i oceniają pełne otoczenie konkretnego osobnika, zatem w jego lokalnym sąsiedztwie i dla jego konkretnego zestawu wartości genów uzyskują kompletną informację o (nie)zależności.

Bazując na tej informacji dowiadujemy się, czy i jak można dokonać dekompozycji problemu – co może mieć miejsce już w trakcie działania algorytmu [PKF21, Rozdział 5]. Algorytm może dzięki temu odpowiednio zarządzać podpopulacjami optymalizującymi potencjalnie niezależne podproblemy, dostosowywać operator krzyżowania, mutacji, itp.

Szacowanie epistazy i dekompozycja – technika DLED

Załóżmy, że mamy osobnika, którego genotyp składa się z co najwyżej pięciu elementów. Istnienie każdego elementu reprezentujemy jednym bitem (np. cztery z pięciu elementów to np. osobnik `10111`).

Współzależności między genami uwypuklają się w optimach lokalnych. Dlatego jeśli nam

na tym zależy, można poddawać analizie osobniki będące optimami lokalnymi – można je np. wcześniej zoptymalizować metodą Greedy (z losową kolejnością sąsiadów–genów) zamieniając pojedyncze $1 \to 0$ i $0 \to 1$.

Na analizowanym genotypie osobnika wykonujemy dekompozycję typu DLED [PKF21]:

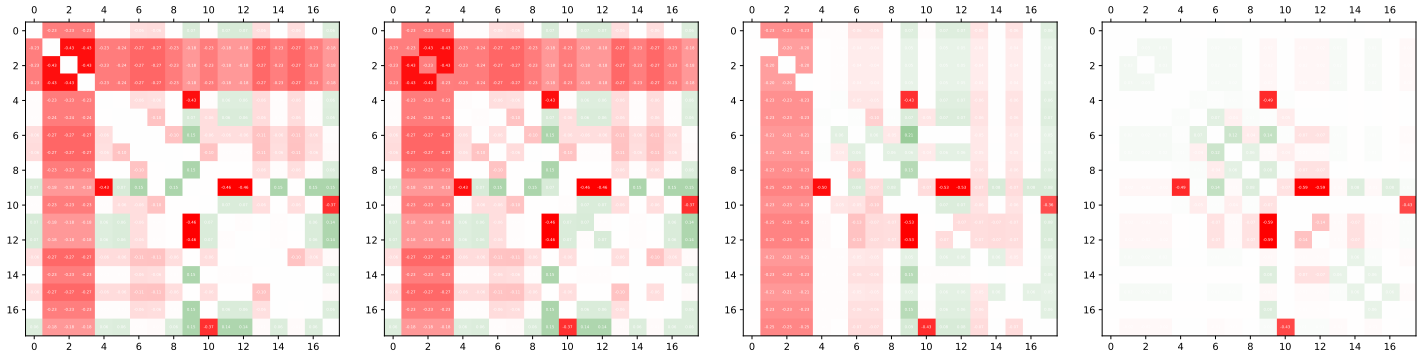
1. Dla każdego genu A wprowadzamy perturbację (zmieniamy wartość na odwrotną).
2. Sprawdzamy wszystkie pozostałe geny, jak dla genu A po perturbacji wartość innego genu B wpływa na fitness jeśli pozostaje niezmieniona oraz jeśli będzie zmieniona.
3. Decyzja o zależności jest binarna i wynika ze spełnienia warunku/warunków poniżej.

Warunki z artykułu [PTK23]:

> The situation changed when the Direct Empirical Linkage Discovery (DLED) was proposed [17]. To check the dependency between genes g, h, DLED requires computing $f(\boldsymbol{x})$, $f(\boldsymbol{x}, g)$, $f(\boldsymbol{x}, h)$, $f(\boldsymbol{x}, g, h)$ values, where $(\boldsymbol{x}, g)$ and $(\boldsymbol{x}, h)$ are the genotypes of individual $\boldsymbol{x}$ with genes g and h flipped, respectively. Finally, $f(\boldsymbol{x}, g, h)$ is the genotype of $\boldsymbol{x}$ with both genes flipped. In DLED, genes g and h are considered dependent if at least one of the conditions holds:
> C1. $f(\mathbf{x}) < f(\mathbf{x}, g) \ \ \& \ \ f(\mathbf{x}, h) \geq f(\mathbf{x}, g, h)$
> C2. $f(\mathbf{x}) = f(\mathbf{x}, g) \ \ \& \ \ f(\mathbf{x}, h) \neq f(\mathbf{x}, g, h)$
> C3. $f(\mathbf{x}) > f(\mathbf{x}, g) \ \ \& \ \ f(\mathbf{x}, h) \leq f(\mathbf{x}, g, h)$
> C4. $f(\mathbf{x}) < f(\mathbf{x}, h) \ \ \& \ \ f(\mathbf{x}, g) \geq f(\mathbf{x}, g, h)$
> C5. $f(\mathbf{x}) = f(\mathbf{x}, h) \ \ \& \ \ f(\mathbf{x}, g) \neq f(\mathbf{x}, g, h)$
> C6. $f(\mathbf{x}) > f(\mathbf{x}, h) \ \ \& \ \ f(\mathbf{x}, g) \leq f(\mathbf{x}, g, h)$
> The above conditions can be interpreted as the following statement. If the modification of one gene changes the fitness relations for the values of the other gene, then these two genes are dependent. DLED is an ELL technique proven to report only the direct dependencies.

Epistaza – przykład ciągły

- Co powinno być wierszami i kolumnami (w tym przykładzie to było „wyłączanie genów", ale czy o to zawsze chodzi?)
- Jak wykorzystać te informacje w algorytmie podczas optymalizacji?
  - optymalizować niezależne podzbiory genów osobno → zmniejszenie złożoności obliczeniowej
  - zaprojektować krzyżowanie i mutację tak, aby zachowywały korzystne epistatyczne interakcje genów → traktować współpracujące epistatycznie grupy genów



Rysunek 2.5: (1) Zmiana w wartości fitness po wyłączeniu par genów w przykładowym osobniku. (2) Zmiana w wartości fitness po wyłączeniu par genów; na przekątnej efekt wyłączenia jednego genu. (3) Odjęcie wartości na przekątnej od wierszy: jak interpretować efekt? (4) Addytywność wpływu lub jej brak.

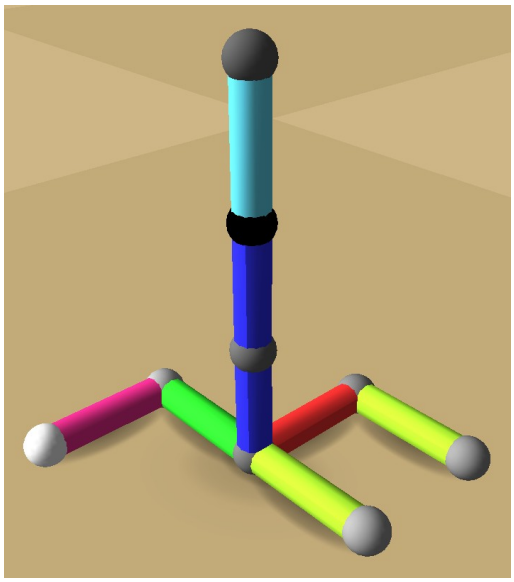
jako integralne zespoły → ochrona i skuteczna propagacja „cegiełek" budujących dobre rozwiązania [GT12; TB13]

- przykład: operator *Optimal Mixing* (OM) w algorytmie GOMEA. Wybiera z populacji rodzica i donora, a następnie tworzy potomka przenosząc z donora do rodzica allele współzależnych (współdziałających) genów. Akceptuje potomka tylko gdy jest on co najmniej tak dobry, jak rodzic [Thi18].

- Jak zastosować analogiczne podejście do wykrycia współzależności trzeciego rzędu? (między trójkami genów?)

Bohater powyższej analizy: genotyp, fenotyp i fitness.

Genotyp: /**9**/UDDDLFBFBRFBBFBFBR



Fitness: wysokość środka ciężkości konstrukcji
(0.47 dla oryginalnego genotypu – dobrego, ale nie lokalnie optymalnego).

2.2.11 Empiryczna i teoretyczna ocena AG

Badanie efektywności i zachowania AE można prowadzić teoretycznie albo eksperymentalnie (empirycznie). Analiza teoretyczna daje ugruntowaną, pewną wiedzę, ale często da się tak badać tylko bardzo proste modele (tj. z wieloma założeniami). Analiza empiryczna prowadzi do mniej pewnej wiedzy, trudniej o uogólnienia, ale zawsze da się taką analizę przeprowadzić. Oprócz testowania skuteczności algorytmu na docelowym problemie optymalizacji, często używa się znanych problemów testowych[5] albo sparametryzowanych modeli problemów pozwalających na sterowanie nasileniem epistazy, takich jak model NK[6] oraz jego warianty z neutralnością: NKP (probabilistyczny) i NKQ (z kwantyzacją).

2.2.12 Neutralność

Konsekwencje neutralności w krajobrazach przystosowania [Gea+02]:

- potocznie/tradycyjnie uważa się, że krajobraz przystosowania „składa się z pagórków”, a mutacje powodują zmianę przystosowania
- prowadzi to do koncepcji „optimów lokalnych”, w których rozwiązania lub ich populacje mogą utknąć

[5] https://en.wikipedia.org/wiki/Test_functions_for_optimization
[6] https://en.wikipedia.org/wiki/NK_model

- badania molekularne nad krajobrazami różnych sposobów zwijania RNA sugerują, że duża część mutacji na poziomie molekularnym jest selekcyjnie neutralna
- neutralność jest również obecna w wielu rzeczywistych problemach optymalizacyjnych (popularne „płaskowyże" lub sąsiedzi tej samej jakości)
- zatem: wiele genotypów [→ jeden fenotyp] → jedna wartość funkcji celu
- neutralność: jedna z przyczyn równowag przestankowych
- jeśli neutralność występuje często w krajobrazie, ryzyko uwięzienia populacji AE w lokalnych optimach jest niskie
- w związku z tym rośnie rola charakterystyk operatorów rekonfiguracji (mutacji/sąsiedztwa, krzyżowania) i rola dryfu genetycznego
- kluczowa rola ostrej i nieostrej nierówności w implementacjach przeszukiwania lokalnego, o której mówiliśmy wcześniej (ostra nierówność → wszystkie neutralne ruchy są potencjalnymi końcami)

2.2.13 Dryf genetyczny

Główne siły prowadzące ewolucję:

- co kieruje ewolucją? co wpływa na trajektorię populacji?
- krajobraz przystosowania kontra operatory rekonfiguracyjne
- co się stanie, gdy wszystkie lub większość mutacji będzie niekorzystna, a dawne osobniki nie będą zachowywane?
- „bias"/„neutralność" krajobrazu przystosowania vs. bias/neutralność operatorów rekonfiguracyjnych
- operatory rekonfiguracyjne: zmiany w częstości alleli, ale zwykle chcemy uniknąć „biasu"!
- widzisz populację osób o niebieskich, zielonych i brązowych oczach;
  po iluś pokoleniach wszyscy mają zielone oczy. Dlaczego?

Dryf genetyczny:

- wyłączmy presję selekcyjną, mutację i krzyżowanie
- przykład kamyków w słoiku

- dyskretny charakter populacji (składających się z odrębnych osobników), więc idealnie równy rozkład alleli nie jest możliwy (np. 1/64 wśród 5000 osobników)
- wąskie gardła (populacja tymczasowo kurczy się do bardzo małych rozmiarów)
  - efekt założyciela
- wpływ rozkładu cech i wielkości populacji
  - prawdopodobieństwo, że dana cecha ostatecznie zdominuje całą populację, to jej częstość w populacji
  - oczekiwana liczba pokoleń do wystąpienia całkowitej dominacji jest proporcjonalna do wielkości populacji
- wpływ losowości (nieograniczonej) vs. „sprawiedliwość" – przypomnienie: nasza wcześniejsza szczegółowa dyskusja

---

- jeśli nie znasz tych zjawisk, ich konsekwencje mogą być sprzeczne z intuicją, błędnie interpretowane i mylnie przypisywane innym mechanizmom!
- interakcja krajobrazu przystosowania (selekcja), operatorów rekonfiguracji i dryfu genetycznego.

2.2.14 Przykład analizy teoretycznej: brak mutacji

Oto przykład prostej analizy teoretycznej algorytmu genetycznego zero-jedynkowego, z selekcją ruletkową i krzyżowaniem, bez mutacji. Rozważamy populację m osobników o długości k. Ile jest różnych stanów populacji? Ile jest stanów, które są atraktorami?

Użyjemy łańcuchów Markowa; niech π – stan, P – macierz prawdopodobieństw przejścia[7]. Jest 2^{mk} różnych stanów. Rozkład prawdopodobieństwa stanów, w których możemy wylądować po n krokach[8] od stanu π to π-ty wiersz macierzy P^n – zapiszmy go jako πP^n. Ponieważ nie ma mutacji, pewne stany są absorbujące (nie ma z nich wyjścia) – wszystkie osobniki są równe; takich stanów jest $a = 2^k$. Zatem macierz P możemy wyrazić jako[9]

$$P = \begin{bmatrix} I_a & 0 \\ R & Q \end{bmatrix}$$

[7] https://en.wikipedia.org/wiki/Stochastic_matrix
[8] https://www.youtube.com/watch?v=nnssRe5DewE
[9] https://en.wikipedia.org/wiki/Absorbing_Markov_chain

gdzie I_a to macierz $a \times a$ wypełniona zerami oprócz jedynek po przekątnej, R to podmacierz $t \times a$ opisująca przejścia do stanu absorbującego, Q to podmacierz $t \times t$ opisująca przejścia do stanów nie-absorbujących; $t = 2^{mk} - a$. Dla n kroków będziemy mieli

$$P^n = \begin{bmatrix} I_a & 0 \\ N_n R & Q^n \end{bmatrix}$$

gdzie $N_n = I_t + Q + Q^2 + Q^3 + \ldots + Q^{n-1}$, I_t to macierz $t \times t$ wypełniona zerami oprócz jedynek po przekątnej. Granicznie

$$\lim_{n\to\infty} P^n = \begin{bmatrix} I_a & 0 \\ (I_t - Q)^{-1} R & 0 \end{bmatrix}.$$

A zatem, co można łatwo zobaczyć mnożąc przykładowe macierze w numpy czy nawet w arkuszu kalkulacyjnym, nasz algorytm zaczynając ze stanu nie-absorbującego (prawdopodobieństwa $(I_t - Q)^{-1}R$) wyląduje na pewno w jakimś stanie absorbującym i w nim pozostanie (I_a). Obliczmy prawdopodobieństwo trafienia do stanu absorbującego [Fog00, str. 105]. Niech $\Gamma = \{0, 1\}$. Po n iteracjach, nasz algorytm znajdzie się w stanie γ, $\gamma \in (\Gamma^k)^m$:

$$\Pr(\gamma \in A) = \sum_{i=1}^{a} \left(\pi^* P^n\right)_i = \sum_{i=1}^{a} \left(\pi^* \begin{bmatrix} I_a \\ N_n R \end{bmatrix}\right)_i$$

gdzie $(\cdot)_i$ oznacza i-ty element poziomego wektora, A to zbiór wszystkich stanów absorbujących, a π^* to wektor poziomy opisujący prawdopodobieństwa rozpoczęcia algorytmu w każdym stanie populacji. Graniczne prawdopodobieństwo absorpcji

$$\lim_{n\to\infty} \sum_{i=1}^{a} \left(\pi^* \begin{bmatrix} I_a \\ N_n R \end{bmatrix}\right)_i = \sum_{i=1}^{a} \left(\pi^* \begin{bmatrix} I_a \\ (I - Q)^{-1} R \end{bmatrix}\right)_i = 1.$$

Zadanie dla chętnych. Stosujemy pokoleniowy algorytm ewolucyjny, populacja n osobników, g z n osobników jest dobrych, $n - g$ jest złych. Wykorzystujemy selekcję turniejową (wielkość turnieju k) ze zwracaniem, w której jeśli osobnik dobry spotka się z złym, wygrywa dobry.

Załóż, że krzyżowanie i mutacja nie zmieniają charakterystyki osobnika (dobry zostaje dobrym, zły pozostaje złym).

1. Zakładając, że w populacji jest połowa osobników dobrych ($g = \frac{n}{2}$), jak zmieni się ta proporcja w kolejnym pokoleniu (tzn. po jednej selekcji)?
2. Zakładając, że wszystkie osobniki podlegają mutacji i mutacja nigdy nie polepsza złego osobnika, ale pogarsza osobnika dobrego z prawdopodobieństwem m, jakie musi być co najmniej g, żeby liczba dobrych w populacji nie spadała?

2.3 Strategie ewolucyjne (*evolutionary strategies*)

Strategie ewolucyjne (ang. *Evolutionary Strategies*, ES) rozwijały się przez pewien czas niezależnie od AG, jako metody służące do optymalizacji numerycznej. Wiele aspektów różni je od AG; wspólne jest wykorzystanie mechanizmów ewolucji podczas optymalizacji.

ES – naturalne pochodzenie: jedno z pierwszych zastosowań (1964) to *evolutionary design* (patrz rozdział 5.10.3). W celu minimalizacji oporów przepływu wody i optymalizacji kształtów rur, aby ocenić konstrukcję lub rurociąg, nie symulowano jej, tylko budowano [Rec84, zob. rys. na str. 123]; zmiany konstrukcji odpowiadały „mutacjom". Był to zatem sposób postępowania realizujący algorytm optymalizacji.

Wczesne strategie ewolucyjne używały jedynie operatora mutacji, który modyfikował jedynego przetwarzanego osobnika. Inaczej niż w algorytmach genetycznych, osobnik był parą składającą się z wektora wartości zmiennych i wektora odchyleń standardowych (stałego podczas całego procesu ewolucji). Mutacja polegała na zmianie każdej zmiennej wektora wartości o losowy czynnik wygenerowany zgodnie z rozkładem normalnym o odpowiednim odchyleniu standardowym (określonym w wektorze odchyleń standardowych). Osobnik po mutacji zastępował swojego przodka jedynie wówczas, gdy był od niego lepszy i dopuszczalny. Taka strategia została nazwana dwuelementową (ponieważ w pewnym momencie istnieje jeden potomek i jeden przodek) i jest oznaczana (1+1)-ES, a jej działanie przypomina *Local Search*.

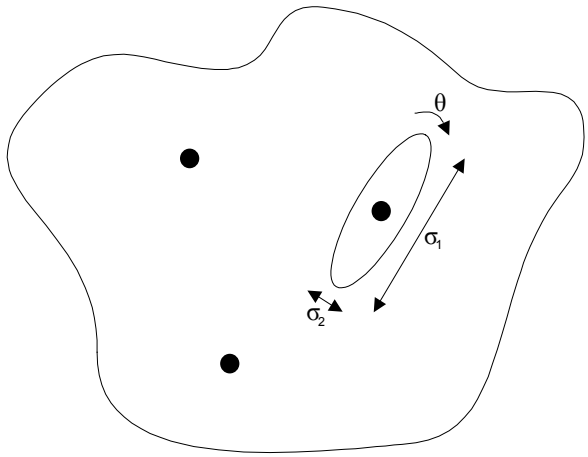
Ulepszeniem strategii dwuelementowej jest strategia wieloelementowa, w której, podobnie jak w AG, istnieje populacja osobników. Wprowadza się dodatkowo operację krzyżowania jednorodnego, jednak nie stosuje się jej do wszystkich osobników, tylko do dwóch z nich – tak, że powstaje jeden potomek, który zastępuje osobnika najsłabszego (jeden nowy osobnik – a więc analogicznie, jak w algorytmach ewolucyjnych typu *steady-state*).

Kolejnym udoskonaleniem było stosowanie krzyżowania wiele razy w jednym kroku (powstawało wielu potomków), a następnie wybór z przodków i potomków *POPSIZE* osobników (tzw. selekcja typu „plus"). W innym podejściu wybiera się osobników do następnego pokolenia tylko z grupy potomków (tzw. selekcja typu „przecinek", ang. *comma selection*), co

jest korzystne w zadaniach ze zmieniającym się optimum.

Ogólny i zwięzły zapis architektury ES ma postać $(\mu/\rho, \lambda)$-ES albo $(\mu/\rho + \lambda)$-ES, gdzie μ to liczba rodziców, $\rho \leq \mu$ to liczba rodziców z których wywodzą się potomkowie, λ to liczba potomków, przecinek oznacza selekcję tylko ze zbioru potomków, a plus – z obu zbiorów: rodziców i potomków.

Stosuje się ulepszony operator mutacji, który zmienia nie tylko wartość zmiennej, ale także odchylenie standardowe zmian, które podlega również ewolucji. Do reprezentacji osobnika, oprócz wartości zmiennych i odchyleń standardowych, można wprowadzić dodatkowo informację o preferowanym kącie odchylenia podczas procesu przeszukiwania i w ten sposób poprawić szybkość zbieżności strategii ewolucyjnych. Zmienna jest wówczas reprezentowana przez jej wartość, odchylenie standardowe i kąt odchylenia, i wszystkie te wielkości podlegają ewolucji pozwalając na samoadaptację i umożliwiając dokładne dostrojenie lokalne. Używa się również krzyżowania arytmetycznego (średnia ważona rodziców).



Rysunek 2.6: Parametry mutacji w strategiach ewolucyjnych.

W szczególności znana z wydajności jest strategia ewolucyjna dostosowująca macierz kowariancji mutacji, CMA-ES,[10] której implementacja jest dostępna np. w bibliotece DEAP.[11] Ta strategia jest też adekwatna dla problemów źle uwarunkowanych (ang. ill-conditioned).

Metoda CMA-ES ma wiele parametrów, istnieje też wiele alternatywnych mechanizmów dla każdego kroku tej metody. Można stosować wartości domyślne oraz polityki wielu uruchomień uwalniające użytkownika od konieczności podawania wartości jakichkolwiek parametrów.

Główna idea CMA-ES:

[10] https://en.wikipedia.org/wiki/CMA-ES

[11] https://deap.readthedocs.io/en/master/examples/cmaes.html

- ustalamy środek populacji,
- próbkujemy rozwiązania z wielowymiarowego (n) rozkładu normalnego (dany jednym parametrem – izometryczny, albo n parametrami – skalowanie równoległe do osi, albo $\binom{n}{2}$ parametrami czyli macierzą kowariancji – umożliwia obracanie),
- oceniamy wszystkie rozwiązania,
- przesuwamy środek populacji: ustawiamy go w miejscu średniej ważonej jakością (rankingową) najlepszych osobników. Rankingowość powoduje nieczułość na niewielkie zaburzenia oceny („chropowatość" krajobrazu) oraz jego wyginanie – stopień wklęsłości,
- rozproszenie nowych (próbkowanych) osobników jest proporcjonalne do prędkości, z jaką przemieszcza się środek populacji: wolniejsze przemieszczanie → mniejsze rozproszenie,
- aktualizujemy macierz kowariancji tak, żeby rozciągnąć nieco wielowymiarowy rozkład normalny w kierunku przemieszczenia środka populacji. W ten sposób będziemy dalej podążali zgodnie z aproksymowanym gradientem oczekiwanej jakości rozwiązań.

2.4 Ewolucja różnicowa (*differential evolution*)

Specyficzną cechą ewolucja różnicowej jest mutacja różnicowa [SP97]. W każdej iteracji ewolucji, dla każdego osobnika o z populacji N osobników powtarzamy:

- losujemy n unikatowych z N osobników, wybieramy z nich osobnika bazowego β i osobnika różnicy δ (dla $n = 3$, β może być wybrany z nich losowo, a δ może być różnicą dwóch pozostałych osobników),
- tworzymy osobnika tymczasowego („donora") $\omega = \beta + F\delta$ (F – stała),
- krzyżujemy ω z o,
- decydujemy czy efekt krzyżowania ma zastąpić oryginalnego o, czy nie (selekcja).

DE jest znana ze swojej prostoty, małej liczby parametrów (przykładowa implementacja) i szybkiej zbieżności. Nie wymaga określenia osobnego, niezależnego rozkładu prawdopodobieństwa dla mutacji – mutacja wynika ze stanu populacji. Warianty DE są konkurencyjne w stosunku do innych algorytmów w corocznych konkursach optymalizacyjnych.

Tworzenie osobnika tymczasowego ω: porównaj krzyżowanie simpleksowe z rozdziału 2.5.1.

2.5 Programowanie ewolucyjne (*evolutionary programming*)

Zbigniew Michalewicz wprowadził na takie podejście nazwę „program ewolucyjny" [Mic96].

EP, originally conceived[12] by Lawrence J. Fogel[13] in 1960, is a stochastic OPTIMIZATION strategy similar to GAs, but instead places emphasis on the behavioral linkage between PARENTs and their OFFSPRING, rather than seeking to emulate specific GENETIC OPERATORS as observed in nature.

Three ways in which EP differs from GAs:

1. There is no constraint on the representation. The typical GA approach involves encoding the problem solutions as a string of representative tokens, the GENOME. In EP, the representation follows from the problem. Example: a neural network can be represented in the same manner as it is implemented, because the mutation operation does not demand a linear encoding. (In this case, for a fixed topology, real-valued weights could be coded directly as their real values and mutation operates by perturbing a weight vector with a zero mean multivariate Gaussian perturbation. For variable topologies, the architecture is also perturbed).
2. The mutation operation simply changes aspects of the solution according to a statistical distribution: minor variations in the behavior of the offspring are highly probable, substantial variations are unlikely. The severity of mutations is often reduced in time.
3. EP typically does not use any CROSSOVER as a GENETIC OPERATOR.

Obecnie „evolutionary programming" jest nazwą rzadko używaną, zamiast niej mówi się o algorytmie ewolucyjnym – co oznacza ogólnie, że użyto algorytmu przystosowanego do danego problemu. Stopień jego przystosowania bywa różny; najczęściej obejmuje reprezentację i operatory.

Wykorzystuje się wiele reprezentacji osobników: zbiór, lista, permutacja[14], drzewo, graf nieskierowany, graf skierowany, macierz, wyrażenia logiczne, reguły (jak w GBML, rozdział 2.8), sieci neuronowe, automaty, wyrażenia opisane gramatyką (np. zapisane w ONP), wyrażenia o strukturze drzewa, programy (jak w GP, rozdział 2.6), ...

Przykładem takiej specyficznej reprezentacji jest ta stosowana w „gładkim algorytmie ewo-

[12] https://www.kanadas.com/whats-ep.html, email from https://en.wikipedia.org/wiki/David_B._Fogel

[13] https://en.wikipedia.org/wiki/Lawrence_J._Fogel

[14] Operatory krzyżowania dla permutacji: OX, PMX, ERO, inne: https://hrcak.srce.hr/file/163313

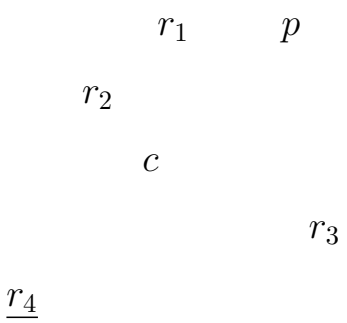
lucyjnym" [Gut94; Gut97]. Osobnik modeluje tu krzywe ciągłe i gładkie. Zastosowania: wygładzanie krzywych/serii danych o nieznanym modelu analitycznym, usuwanie zbędnego tła w sygnale fizycznym itp. Genotyp opisuje (dyskretną) formę krzywej za pomocą skończonej liczby par (x_i, y_i), $i = 1..n$, n – liczba przedziałów. x_i można traktować jako i, choć przedziały mogą być również niejednakowe. Stosowane są również specyficzne operatory genetyczne, w szczególności gładkie krzyżowanie i mutacja.

2.5.1 Reprezentacja liczb rzeczywistych

Bardzo często w AE (i w optymalizacji w ogóle!) stosuje się reprezentację wartości ciągłych. Geny kodują liczby rzeczywiste w takim formacie, jak jest to przyjęte na procesorach (zmienna precyzja w zależności od bezwzględnej wartości liczby). Pytanie: jakie można zaproponować operatory krzyżowania i mutacji (oprócz typowych, takich jak wymiana genów albo wielopunktowe) dla wektora liczb? Proponując operatory pamiętaj o celu krzyżowania i mutacji.

Krzyżowanie: np. średnia z rodziców, lub średnia ważona by uzyskać dwóch różnych potomków. Średnia ważona to krzyżowanie **arytmetyczne** – dzieci są liniową kombinacją rodziców: $d_1 = r_1 \cdot a + r_2 \cdot (1-a)$, $d_2 = \ldots$. Wagę a można losować przy każdym wykonaniu.

Inny operator krzyżowania – krzyżowanie simpleksowe: wyznaczamy centroid c rodziców r. Wariant bez dostępu do jakości rozwiązań – SPX [TYH99]: losujemy potomka z (powiększonej) przestrzeni kombinacji liniowych rodziców (odsuniętych od c o ε – *the expanding rate*). Wariant z uwzględnieniem jakości: tworzymy potomka p jako przesunięcie od najgorszego osobnika/rodzica dalej poprzez punkt c.



Mutacja:

- *uniform random* (**Flat**) – ustal gen na wartość losową z dozwolonego przedziału

- *creep* – zmień gen o wartość losowaną z pewnego rozkładu (np. normalnego albo jednostajnego – np. $-3..+3$, itp.)

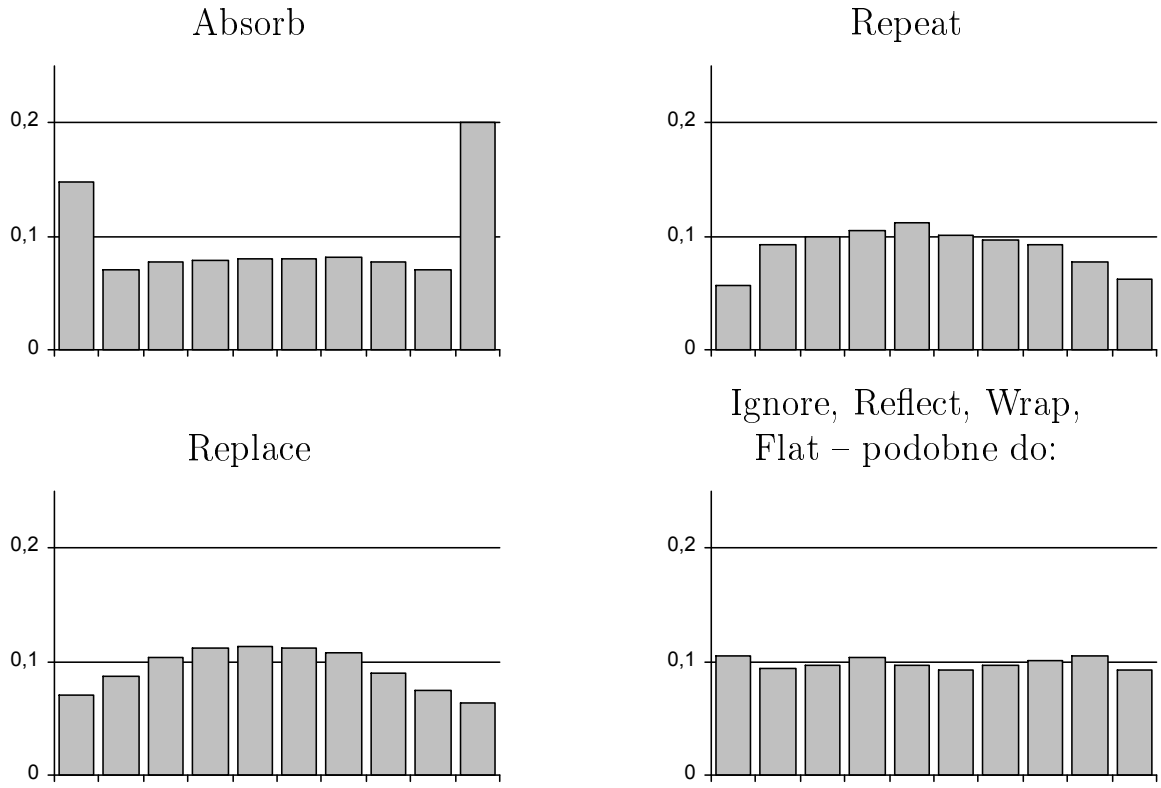
Aby uniezależnić się od „osi" (tzn. aby mutacja nie przebiegała tylko równolegle do osi, czyli nie dotyczyła pojedynczych parametrów, co byłoby niekorzystne gdyby funkcja celu była np. obróconą wersją funkcji zależnej wprost od parametrów), mutuje się naraz wszystkie geny (i wtedy stosujemy rozkład normalny losowanej zmiany a nie równomierny – zastanów się dlaczego). Chcąc zapewnić, że taka mutacja naraz n elementów wektora przesunie bieżące rozwiązanie o taką samą odległość w n-wymiarowej przestrzeni, jak zrobiłaby to mutacja tylko jednego wymiaru, przez jaką wartość należy podzielić (znormalizować) każdą z n wylosowanych wartości zmiany w n-wymiarowym wektorze?

Co zrobić, jeśli po zmutowaniu wartość genu wychodzi poza dozwolony zakres? Jakie metody rozwiązania tego problemu można zaproponować? [Bul99; Bul01]

- **Absorb**: Illegal mutant values are truncated to the nearest boundary.
- **Repeat**: Mutant values are repeatedly generated, until a legal value is obtained.
- **Replace**: Any offspring for which illegal trait values are generated is replaced by a new offspring, re-choosing parents.
- **Ignore**: Mutation events which transgress legal bounds are ignored. Rather than inherit an illegal mutant value, offspring inherit the parental value.
- **Reflect**: Mutant values lying a distance of d above (or below) the legal range are replaced by vales a distance of d below (or above) the nearest boundary.
- **Wrap**: The trait is treated as if it were periodic. The edges of its legal range "wrap" around. Mutant values are calculated modulo the trait's range.

Czy ma znaczenie, którą z metod wybierzemy? Tak! [dyskusja pożądanych cech mutacji i wybór zwycięskiej metody].

Wyniki eksperymentalne: jak często zdarzały się po mutacji i „naprawie" różnymi metodami określone wartości genu? Przodek miał równe szanse na każdą wartość. Oś pozioma – przedział zmienności genu, oś pionowa – częstość występowania wartości potomka w podprzedziałach:

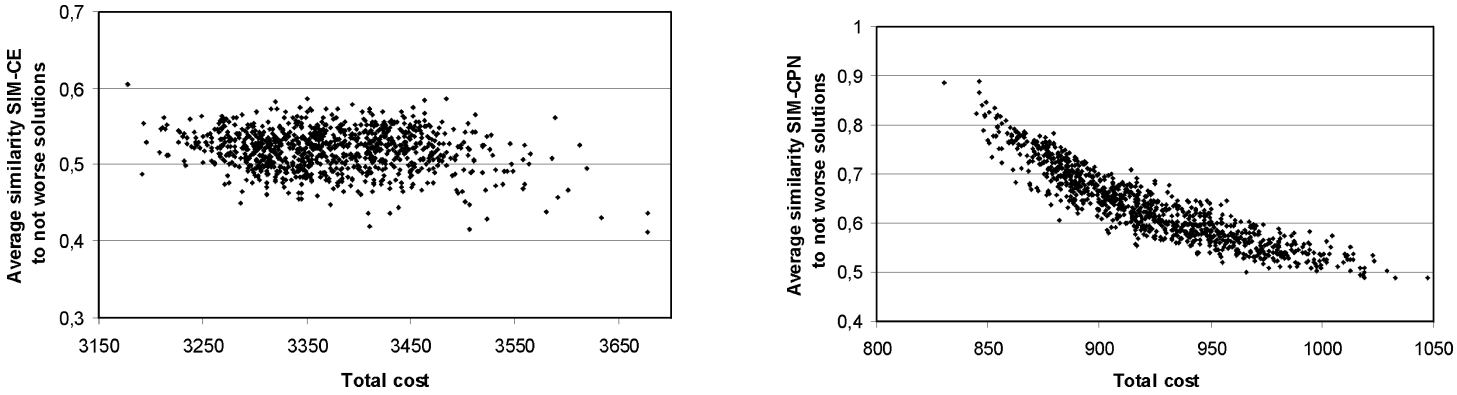


To, że w metodach **Ignore**, **Reflect**, **Wrap**, uzyskano taki sam rozkład jak we **Flat** nie oznacza, że metody te zachowują się tak samo. Faktycznie zasada ich działania jest zupełnie inna. Przykład: metoda **Ignore**. W okolicy granic przedziału więcej mutacji będzie nielegalnych i ignorowanych. Skoro ignorowanych, to rzadziej też będą trafiać się w ogóle wartości bliskie granicom. Kiedy już się trafią, rzadko doprowadzą do poprawnej mutacji...

Zatem to, że rozkład jest taki sam, nie gwarantuje jeszcze, że tak samo często zdarzają się efektywne (faktycznie zmieniające wartość genu) mutacje w poszczególnych podprzedziałach. Stąd oprócz rozkładu częstości wartości genu dla różnych metod należy też porównać inne parametry – np. ile liczb ubywa z określonego podprzedziału (przed mutacją) i przybywa (po mutacji), jaki jest między nimi związek, itp. W tym świetle metody **Ignore**, **Reflect**, **Wrap** różnią się między sobą, i żadna z nich nie zachowuje się tak jak **Flat**.

Rozpatrywaliśmy tu nieskomplikowany rodzaj mutacji i proste mechanizmy, a jednak te drobne elementy mają duży wpływ na proces ewolucji. Na przykład **Absorb** ukierunkowuje (*bias*) **cały czas** dryf genetyczny ku krańcowym wartościom dozwolonych przedziałów. Nie będąc tego świadomym można wyciągnąć wniosek, że są one optymalne i ewolucja je faworyzuje – tymczasem jest to ciągły wpływ mutacji.

Przegląd operatorów mutacji i krzyżowania dla problemów numerycznych zawierają książki–zbiory [Gwi07a; Gwi07b].



Rysunek 2.7: Podobieństwo rozwiązań a podobieństwo ich ocen – przykład z [Kub04].

2.5.2 Krzyżowanie i mutacja a globalna wypukłość

Globalna wypukłość: dobre rozwiązania są do siebie bardziej podobne, niż do gorszych. Da się obiektywnie i ilościowo zmierzyć! (*fitness–distance correlation*, FDC). Jeśli wysoka, to warto zaczynać optymalizację z dobrych rozwiązań, a dywersyfikacja może polegać na niewielkich zaburzeniach rozwiązania (zamiast zaczynać algorytm zupełnie od nowa).

Krzyżowanie zachowujące odległość (*distance-preserving crossover*, DPX):
dist(parent1,parent2)=dist(parent1,child)=dist(parent2,child).

Świadome tworzenie skutecznych operatorów krzyżowania:

- pomyśl, jakie cechy rozwiązania wpływają na wartość funkcji celu,
- stwórz miary odległości bazujące na tych cechach,
- zbadaj, czy te miary determinują globalną wypukłość (Rys. 2.7),
- jeśli tak, wykorzystaj te cechy budując operator DPX.

Analogicznie postępujemy, gdy chcemy stworzyć skuteczny operator mutacji (sąsiedztwa) – opieramy się na takich aspektach podobieństwa rozwiązań, które ujawniają powyższą korelację.

Miara podobieństwa rozwiązań ma liczne zastosowania – przydaje się m.in. do:

- testowania pomysłów na operator krzyżowania – różne cechy rozwiązań i wartości FDC,
- prowadzenia selekcji ze ściskiem – rozdział 2.2.2,
- szacowania różnorodoności w populacji i oceny zbieżności,

- analizy struktury populacji; analizy skupień w zbiorze rozwiązań,
- utrzymywania „gatunków” podczas ewolucji – rozdział 2.7.3,

oraz wszędzie tam, gdzie występuje potrzeba wyznaczenia różnicy dwóch rozwiązań, np.

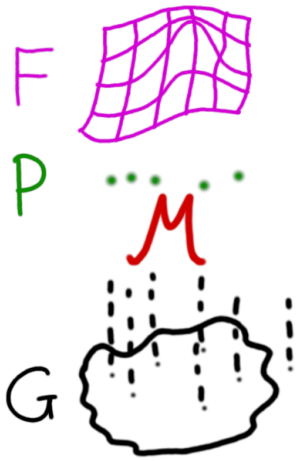
- w ewolucji różnicowej – rozdział 2.4,
- przy krzyżowaniu simpleksowym – rozdział 2.5.1.

Jeśli rozwiązania posiadają prostą reprezentację (rozważ kilka przykładów), wtedy pomysły na miary podobieństwa mogą nasuwać się same. Dla złożonych reprezentacji (rozważ kilka przykładów) przydatne mogą być pojęcia odległości edycyjnej[15] oraz odległości spychaczowej[16] (tłum. MK).

2.5.3 Embriogeneza

Embriogeneza: mapowanie genotyp → fenotyp. Dla prostych reprezentacji i równomiernych, jednorodnych przestrzeni takich jak kompletna przestrzeń bitów, liczb lub permutacji, pierwszym (domyślnym) pomysłem jest trywialne, bezpośrednie mapowanie 1:1.

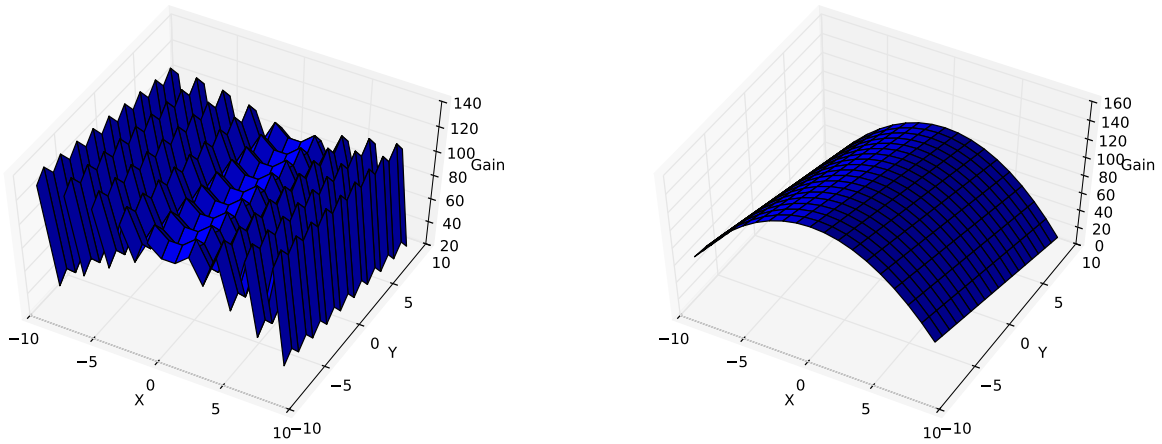
Ale czy takie mapowanie jest najlepszym wyborem?



Przypomnij sobie RGB ↔ HSL, sygnał ↔ widmo, kran ↔ , ...

Zastanów się, w jakich sytuacjach mapowanie genotyp → fenotyp powinno (albo musi?) być bardziej skomplikowane. Jakie cechy mapowania powinna zapewnić procedura mapująca przestrzeń genotypów w przestrzeń fenotypów?

[15] https://en.wikipedia.org/wiki/Edit_distance
[16] https://en.wikipedia.org/wiki/Earth_mover%27s_distance



Pomyśl teraz o naturze, o organizmach żywych i o biologicznym mapowaniu genotyp → fenotyp. Jakie ono jest i czy jest korzystne? Czy dałoby się zrealizować to mapowanie lepiej?

Jeśli przestrzeń fenotypów jest inna niż genotypów (co ma miejsce np. wtedy, kiedy rozwiązania są bardzo skomplikowane – wyobraź sobie optymalizację mostu, samochodu, robota, ...), potrzebna jest procedura „mapowania" jednej przestrzeni w drugą (Rys. 2.8). W biologii ten proces to embriogeneza (rozwój z genotypu do stadium zarodka – budowanie ciała). Ale nawet dla identycznych przestrzeni, niebezpośrednie mapowanie może przynieść korzyści.

Embriogeneza – decyzje i ich konsekwencje [Rot06]:

- nadmiarowość: wiele genotypów → jeden fenotyp
  - sąsiednia (*synonymous*): genotypy tworzące ten sam fenotyp są sąsiadami
    - równoliczna każdy fenotyp powstaje z takiej samej liczby genotypów
    - różnoliczna: przeciwnie
  - odległa (*non-synonymous*): niekorzystne dla optymalizacji
- skalowanie alleli: na ile jednakowy jest wpływ alleli na fitness
- lokalność: podobieństwo (bliskość) genotypów skorelowane z podobieństwem odpowiadających im fenotypów
  - wysoka: dobrze! mapowanie nie zwiększa trudności problemu
  - niska: zwiększa trudność problemu

Powyższe własności można oszacować liczbowo.

Możliwe powody skłaniające do stosowania nietrywialnego mapowania [Ben99]:

- ograniczenie (zmniejszenie) przestrzeni przeszukiwania (rekurencyjne, hierarchiczne, itd.),
- lepsze próbkowanie przestrzeni przeszukiwania (tworzące topologię zwiększającą FDC – opis w rozdziale 2.5.2),
- bardziej złożone rozwiązania w przestrzeni fenotypów („procedura wzrostu" zapisana w genotypie),
- lepsza obsługa ograniczeń (mapowanie **każdego** genotypu na poprawny fenotyp),

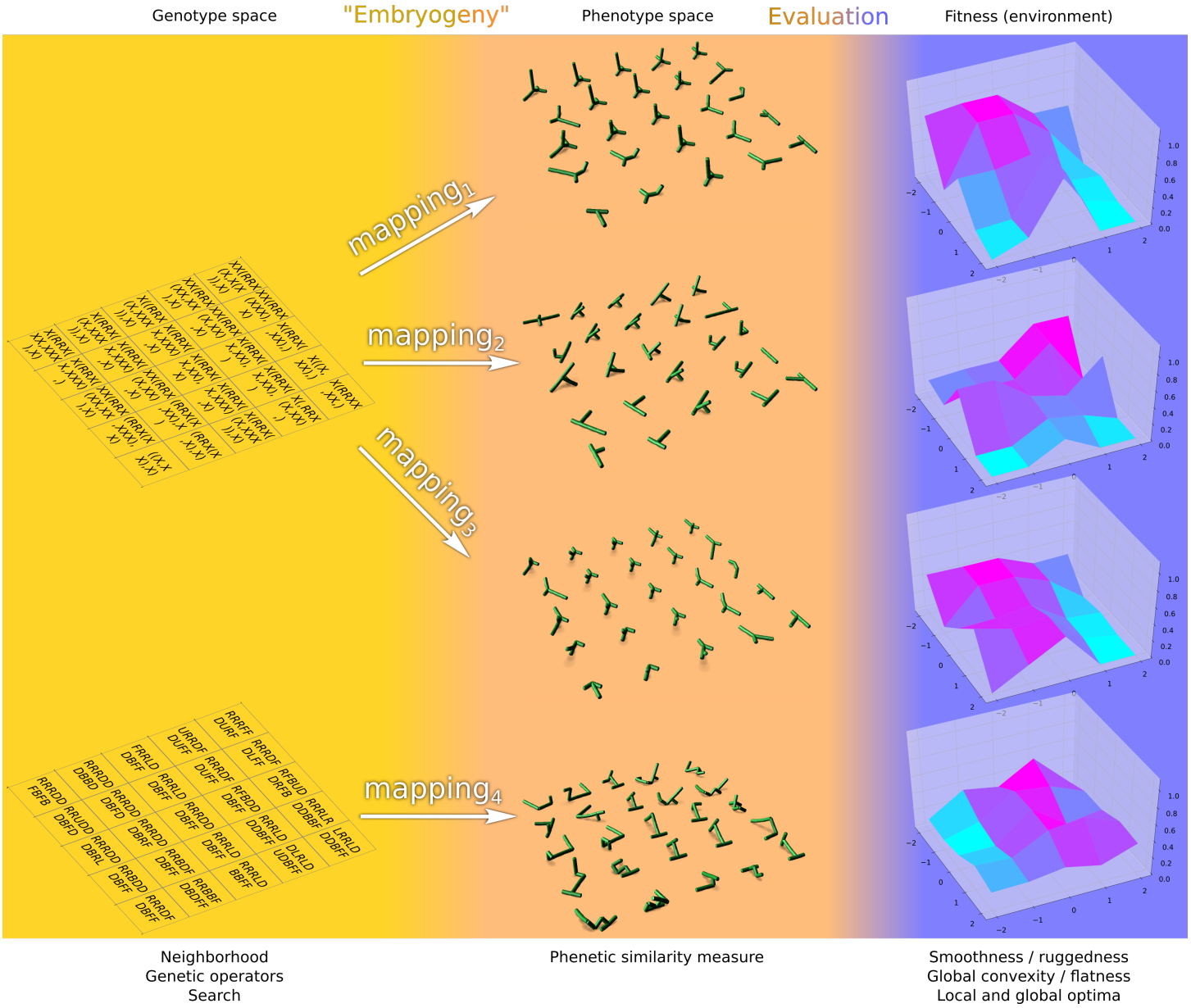
oraz:

- kompresja: proste genotypy opisują skomplikowane fenotypy,
- powtarzanie: genotypy mogą opisywać symetrię, modularność, podprocedury, itd.,
- adaptacja: fenotypy mogą rozwijać się „adaptacyjnie" (żeby dopasować się do ograniczeń, albo żeby naprawiać błędy).

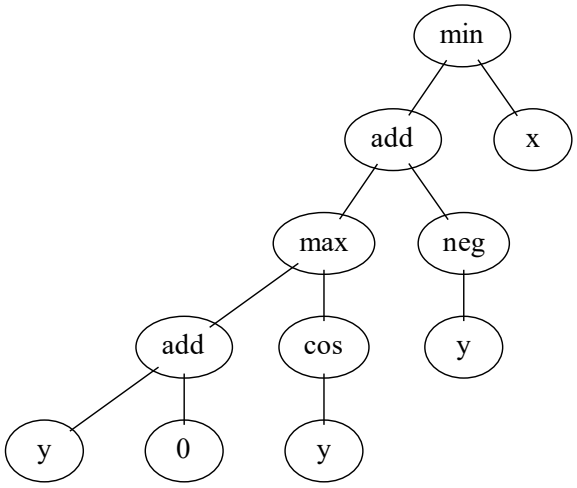
ale:

- potrzeba dużego doświadczenia, aby ręcznie zdefiniować embriogenezę zapewniającą wymienione na poprzednim slajdzie zalety,
- trudno automatycznie wyewoluować embriogenezę (potrzeba specjalizowanych operatorów zabezpieczających przed „puchnięciem" (ang. *bloat*) genotypów i fenotypów, epistazą i psuciem potomków przez operatory lub słabym dziedziczeniem informacji).

W większości zastosowań embriogeneza to niezmienne, zaprojektowane przez człowieka reguły odwzorowujące genotyp w jego znaczenie (*external embryogeny*). Więcej informacji – patrz rozdział 5.10.3 o optymalizowaniu konstrukcji/ewolucyjnym projektowaniu.



Rysunek 2.8: Związki między przestrzenią genetyczną, fenetyczną, i krajobrazem przystosowania. Zauważ, że różne embriogenezy (zatem też różne zbiory fenotypów, ich topologie i krajobrazy przystosowania) mogą być wynikiem (1) różnych reprezentacji genetycznych i przeznaczonych dla nich operatorów (na schemacie są pokazane dwie), (2) różnych interpretacji (pokazano trzy) genów w ramach danej reprezentacji, i (3) tej samej reprezentacji genetycznej i tej samej interpretacji genów, ale różnych operatorów mutacji/sąsiedztwa (schemat tej sytuacji nie pokazuje).



Rysunek 2.9: Wyrażenie `min(add(max(add(y, 0), cos(y)), neg(y)), x))` czyli `min(max(y + 0, cos(y)) + (-y), x)` czyli `min(x, max(y, cos(y)) - y)`.

2.6 Programowanie genetyczne (*genetic programming*)

Programowanie genetyczne[17] służy do optymalizacji wyrażeń i programów. Cechą charakterystyczną jest drzewiasta reprezentacja rozwiązań pozwalająca na zakodowanie programów – Rys. 2.9, choć istnieje też mniej popularny wariant z kodowaniem liniowym.[18]

Wyrażenia przechowywane w populacji składają się z elementów należących do zbioru funkcji $\boldsymbol{F}$ (węzły drzewa) i zbioru terminali $\boldsymbol{T}$ (liście drzewa). Zbiory te można dobrać wedle potrzeb i dostosować do rozwiązywanego problemu. Przestrzeń rozwiązań stanowią wszystkie kombinacje wyrażeń złożonych z elementów obu zbiorów.

Zbiór funkcji	
Rodzaj	Przykłady
Arytmetyczne	+, *, /
Matematyczne	sin, cos, exp
Logiczne	AND, OR, NOT
Warunkowe	IF-THEN-ELSE
Pętle	FOR, REPEAT

Zbiór terminali	
Rodzaj	Przykłady
Zmienne	$\vec{x}$, y, x172
Stałe	3, 0.45, π
Procedury	rand, go_left, read_proximity

„Procedury” mogą byc funkcjami lub akcjami bezargumentowymi.

Pożądane są dwie cechy zbiorów $\boldsymbol{F}$ i $\boldsymbol{T}$:

1. *domknięcie* (ang. *closure*) – każda funkcja działa dla dowolnych wartości i typów

[17] Darmowa książka: http://www0.cs.ucl.ac.uk/staff/W.Langdon/ftp/papers/poli08_fieldguide.pdf
[18] https://en.wikipedia.org/wiki/Linear_genetic_programming

argumentów zwracanych przez dowolną funkcję lub terminal,

2. *wystarczalność* (ang. *sufficiency*) – dostępne w obu zbiorach elementy pozwalają na zbudowanie rozwiązania stawianego problemu.

```
from deap import gp
# https://deap.readthedocs.io/en/master/tutorials/advanced/gp.html
# https://deap.readthedocs.io/en/master/examples/gp_symbreg.html

pset = gp.PrimitiveSet("MAIN", 2) # two arguments (x and y)
pset.addPrimitive(operator.add, 2)
pset.addPrimitive(operator.sub, 2)
pset.addPrimitive(operator.mul, 2)
pset.addPrimitive(operator.neg, 1)
pset.addPrimitive(min, 2)
pset.addPrimitive(max, 2)
pset.addPrimitive(math.cos, 1)
pset.addPrimitive(math.sin, 1)
pset.addEphemeralConstant("rand101", lambda: random.randint(-1,1))
pset.renameArguments(ARG0='x')
pset.renameArguments(ARG1='y')
```

Zastanów się, jak można zapewnić właściwość *domknięcia*.

Właściwość *domknięcia* można uzyskać zabezpieczając odpowiednio funkcje (np. licząc zawsze wartość bezwzględną dla argumentu pierwiastka) albo karając niepoprawne wyrażenia (obniżając im wartość dopasowania). Albo ustawić flagi procesora/programu/systemu operacyjnego tak, żeby wszelkie operacje nie powodowały wyjątków... (tutaj wspomnienie długiej symulacji numerycznej pod linuxem i różnica tej samej symulacji pod Windows).

```
def protectedDiv(left, right):
    try:
        return left / right
    except ZeroDivisionError:
        return 1

pset.addPrimitive(protectedDiv, 2)
```

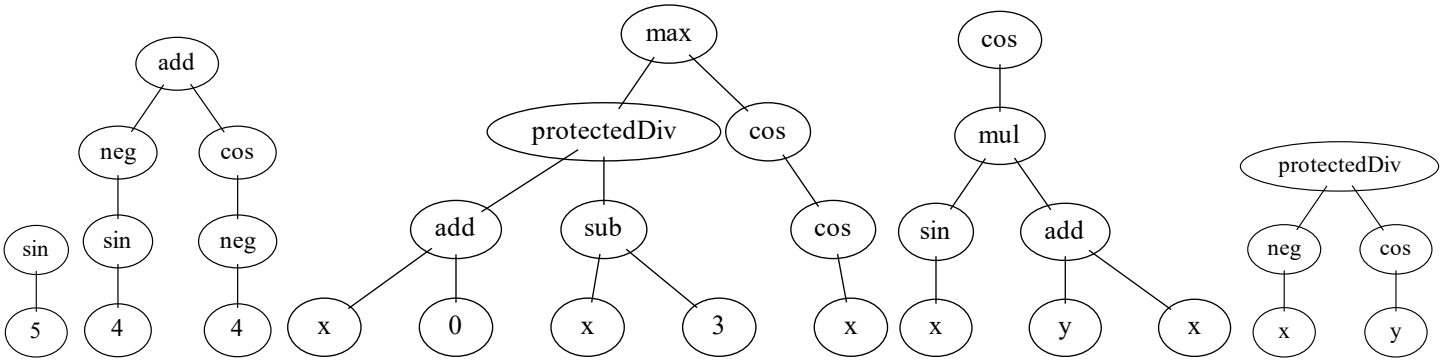
Jeśli nie zapewnimy *wystarczalności*, GP będzie starało się znaleźć (najlepsze) przybliżenie rozwiązania za pomocą dostępnych środków.

Podstawowe metody tworzenia populacji początkowej:

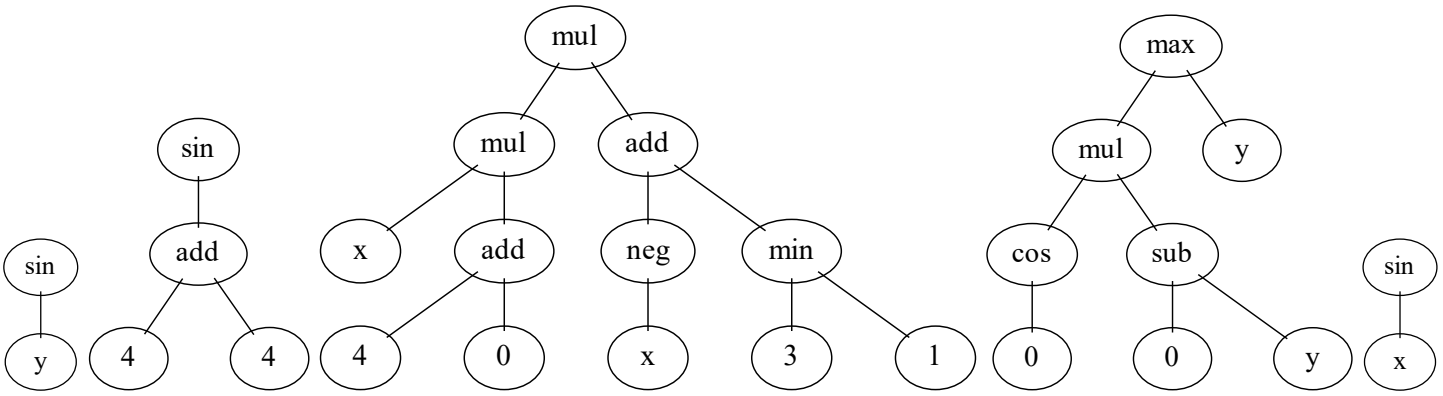
- *Full*: wybieraj węzły z $\boldsymbol{F}$ jeśli głębokość jest poniżej ustalonego progu, a jeśli nie, to z $\boldsymbol{T}$. Wszystkie drzewa będą miały taką samą głębokość – przykłady na Rys. 2.10.

- *Grow*: wybieraj węzły z $\boldsymbol{F} \cup \boldsymbol{T}$ jeśli głębokość jest poniżej ustalonego progu, a jeśli nie, to z $\boldsymbol{T}$. Drzewa będą miały różną głębokość i kształt – przykłady na Rys. 2.11.
- *Ramped half-and-half*: połowa populacji metodą *full*, połowa metodą *grow* – zapewnia zróżnicowanie populacji początkowej.

```
toolbox.register("expr", gp.genHalfAndHalf, pset=pset, min_=1, max_=2) #
    tree height range
toolbox.register("individual", tools.initIterate, creator.Individual,
    toolbox.expr)
toolbox.register("population", tools.initRepeat, list, toolbox.individual
    )
```



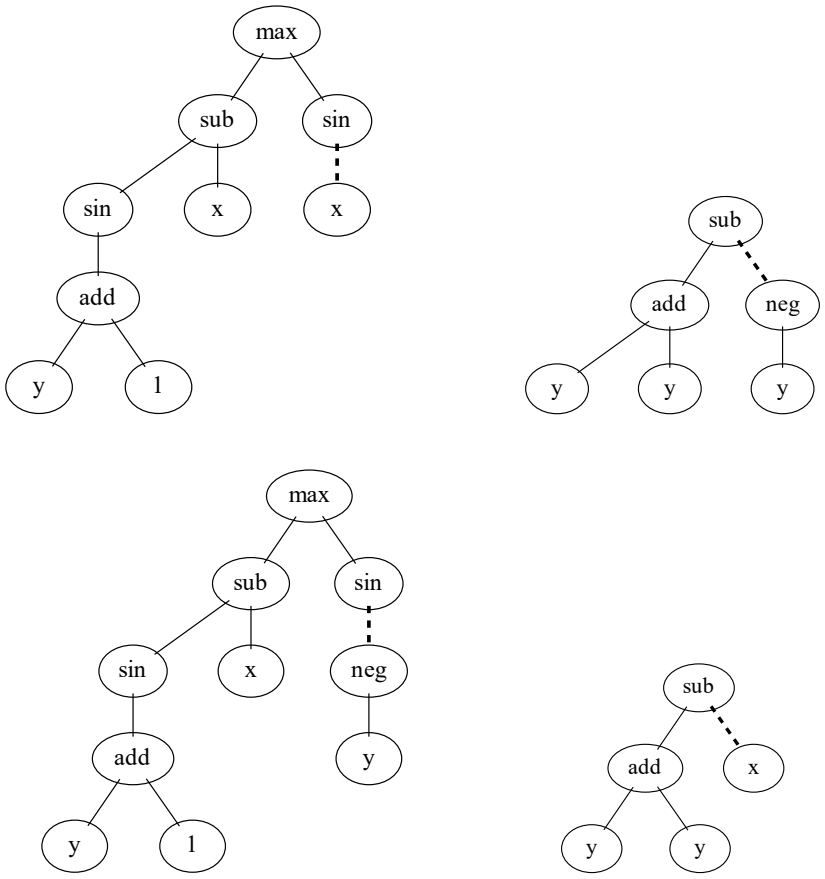
Rysunek 2.10: Pięć osobników wygenerowanych metodą *Full*, `gp.genFull(pset,1,3)` (DEAP wymaga dwóch parametrów, nie jednego) dla $\boldsymbol{T}$={x, y, 0, 1, 2, 3, 4, 5}.



Rysunek 2.11: Pięć osobników wygenerowanych metodą *Grow*, `gp.genGrow(pset,1,3)`, dla $\boldsymbol{T}$={x, y, 0, 1, 2, 3, 4, 5}. W metodzie `genGrow()` DEAP'a nie ma sensu ustawienie argumentów min_depth i max_depth na taką samą wartość, bo generowane drzewa będą miały wtedy wszystkie liście na tej samej głębokości – tak samo, jakby drzewa generowała metoda `genFull()`.

Krzyżowanie w GP to najczęściej wymiana losowo wybranych poddrzew u obu rodziców (Rys. 2.12).

```
toolbox.register("mate", gp.cxOnePoint)
```



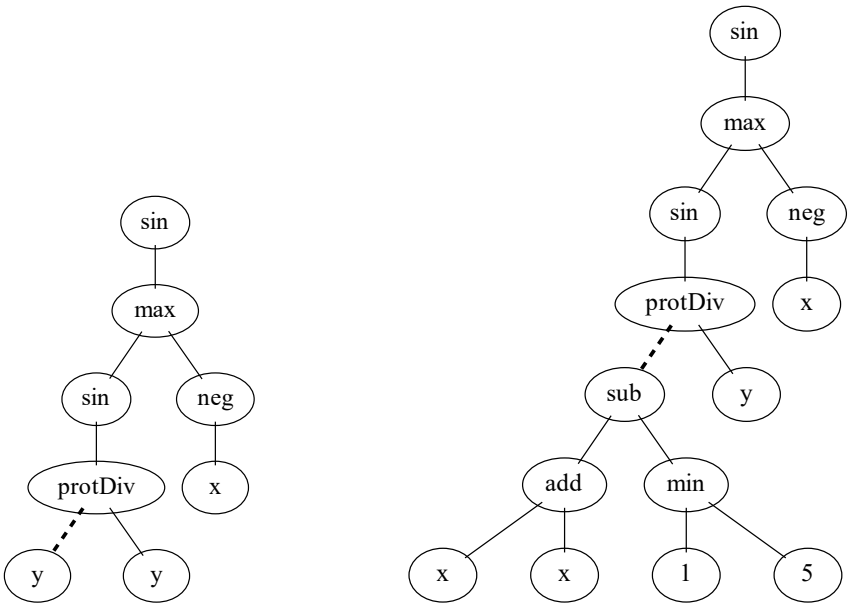
Rysunek 2.12: Krzyżowanie w GP. U góry rodzice wygenerowani metodą `gp.genGrow(pset,2,4)`. Na dole dzieci utworzone metodą `gp.cxOnePoint(parent1,parent2)`.

Typowa mutacja to wybranie losowego miejsca w oryginalnym drzewie i zastąpienie poddrzewa nowym, wygenerowanym jedną z powyżej opisanych metod (Rys. 2.13).

```
toolbox.register("expr_mut", gp.genFull, min_=0, max_=2)
toolbox.register("mutate", gp.mutUniform, expr=toolbox.expr_mut, pset=
   pset)
```

Żeby zabezpieczyć się przed niekontrolowanym „puchnięciem” wyrażeń (ang. *bloat*), można włączyć do oceny rozwiązań kary za rozmiar wyrażenia lub zastosować ograniczenia na głębokość drzewa.

```
toolbox.decorate("mate", gp.staticLimit(key=operator.attrgetter("height")
   , max_value=13))
```



Rysunek 2.13: Mutacja w GP. Po lewej przodek wygenerowany metodą `gp.genGrow(pset,2,5)`. Po prawej mutant utworzony metodą `gp.mutUniform(parent, toolbox.expr_mut, pset=pset)`, przy wcześniejszym `toolbox.register("expr_mut", gp.genFull, min_=0, max_=2)`.

```
toolbox.decorate("mutate", gp.staticLimit(key=operator.attrgetter("height
    "), max_value=11))
```

Ponieważ wyrażenia lub programy generowane przez GP mają przypadkowy charakter, trudno byłoby je uruchamiać bezpośrednio w systemie operacyjnym – bezpieczniej jest je interpretować lub oceniać w wirtualnym środowisku (np. w maszynie wirtualnej albo „sandboxie"). Ocena jakości rozwiązania wymaga najczęściej jego wyliczenia lub użycia w wielu sytuacjach (różne wartości argumentów, różne lokalizacje robota, itp.).

```
# Exception: MemoryError - Error in tree evaluation: Python cannot
    evaluate a tree higher than 90.
```

Można stosować omówione wcześniej, standardowe metody selekcji, ale często lepszych wyników dostarcza selekcja Lexicase. W tej metodzie nie agreguje się błędów każdego rozwiązania na wszystkich testach do jednej liczby. Zamiast tego, aby wybrać jednego osobnika z populacji, najpierw losujemy kolejność testów, a potem wybieramy te osobniki, które na pierwszym teście (z tej losowej kolejności) uzyskały najlepszy wynik w populacji. Jeśli takich osobników jest więcej niż jeden, rozważamy wśród nich tak samo drugi test, potem ewentualnie trzeci, itd. Dyskusja: jak takie podejście różni się od metod selekcji z ewolucyjnej optymalizacji wielokryterialnej takich jak NSGA czy SPEA?

Dyskusja: krajobraz przystosowania, globalna wypukłość i skuteczność optymalizacji w GP.

2.6.1 Regresja symboliczna

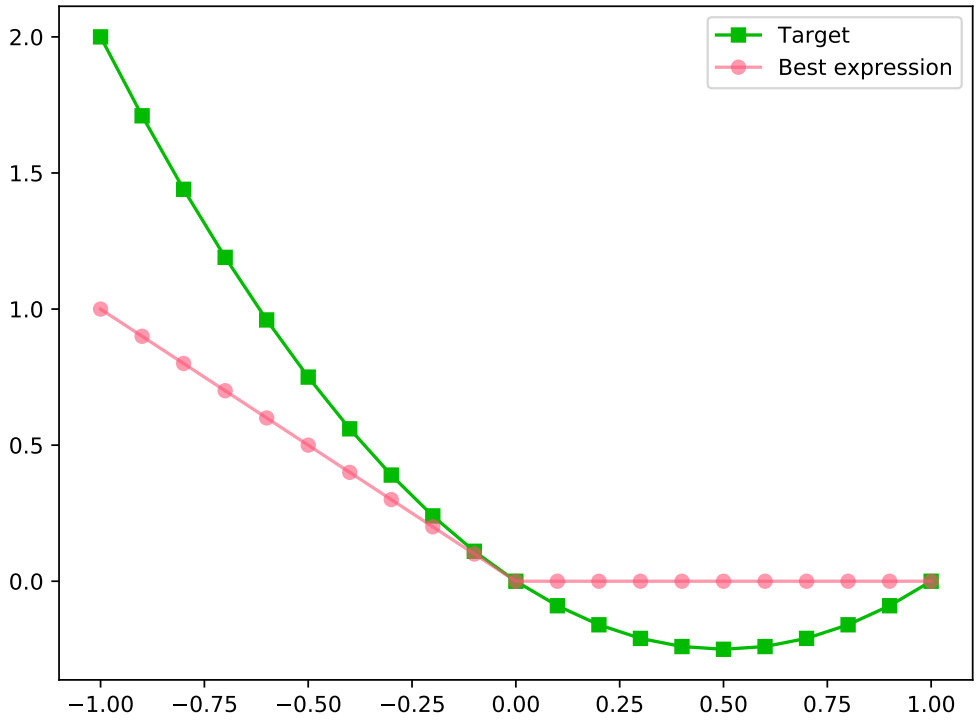
Regresja symboliczna to typowe zastosowanie GP, w którym szukamy funkcji opisującej możliwie dokładnie zadane punkty. O ile w tradycyjnych metodach regresji postać szukanej funkcji jest ustalona (poszukujemy tylko współczynników), o tyle w GP można łatwo manipulować postacią funkcji[19], a nawet szukać pewnych klas funkcji lub dowolnych funkcji (stąd ta metoda regresji nazywana jest *symboliczną*).

```
def target_function(x):
    return x**2 - x  # in a real application, this is what we look for!

def eval_expr(individual, points):
    # transform the tree expression into a callable function
    func = toolbox.compile(expr=individual)
    # evaluate the mean squared error between the expression and the
    target function
    sqerrors = ((func(x) - target_function(x))**2 for x in points)
    return math.fsum(sqerrors) / len(points),

toolbox.register("evaluate", eval_expr, points=[x/10. for x in range
    (-10,11)])
```

Sterowanie postacią szukanego wyrażenia odbywa się za pomocą odpowiedniego doboru elementów zbioru funkcji $\boldsymbol{F}$ i terminali $\boldsymbol{T}$, oraz narzucaniem ewentualnych ograniczeń na głębokość drzewa, liczbę wystąpień funkcji ze zbioru $\boldsymbol{F}$, itp.

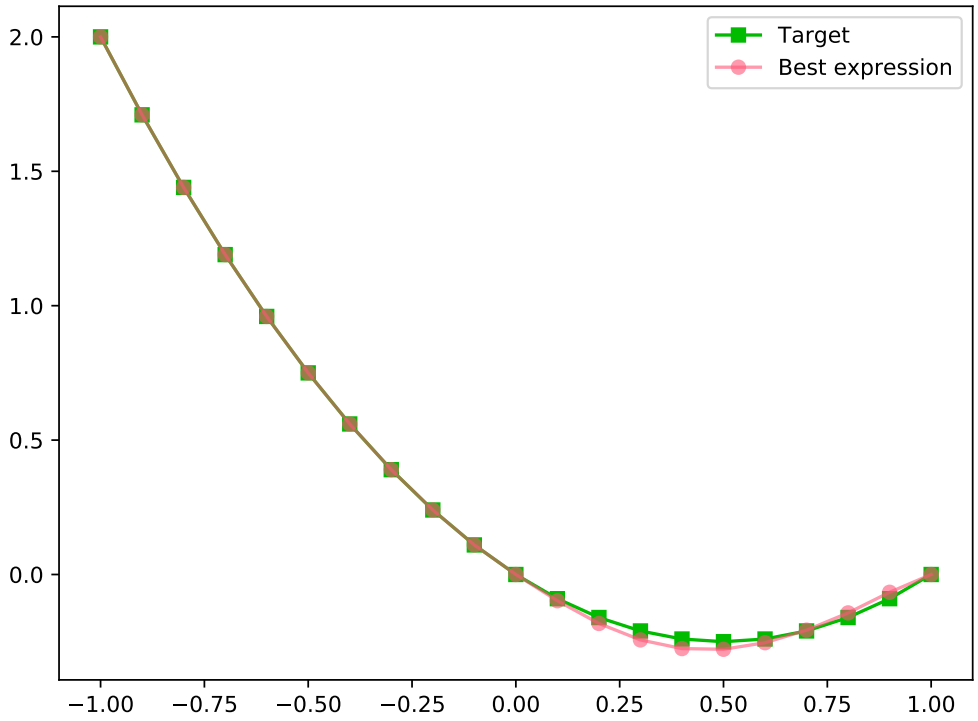
Przykładowy eksperyment #1: Znajdź wyrażenie najlepiej opisujące zbiór punktów należących do funkcji $f(x) = x^2 - x$. Pamiętajmy, że w praktyce funkcja ta jest nieznana i chcemy ją odkryć! Do dyspozycji GP dajemy to co w przykładowych źródłach powyżej, czyli oprócz x operatory neg, $+, -, *, /, \max, \min, \sin, \cos$, i dodatkowo też stałe $-1, 0, 1$.

[19] http://www.alife.pl/programowanie-genetyczne



Rysunek 2.14: Rozwiązanie najlepsze w pierwszym pokoleniu (czyli w losowo wygenerowanej populacji).

```
mul(min(0, x), neg(1))
```



Rysunek 2.15: Rozwiązanie najlepsze po zakończeniu ewolucji.

```
sub(x, add(min(min(min(0, x), mul(0, add(0, max(1, 0)))),
add(x, max(x, mul(add(0, x), neg(x))))), max(add(min(min(x, 0),
add(min(sin(x), x), max(sin(x), add(add(0, 0), sin(sin(sin(x)))))))),
max(sin(add(min(sin(x), sin(sin(sin(sin(x))))), max(sin(sin(x)), -1))),
x)), x)))
```

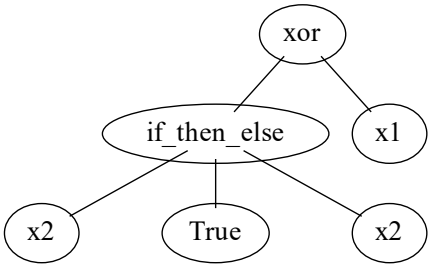
Po zwiększeniu rozmiaru populacji i liczby pokoleń: `mul(add(-1, x), min(x, x))`. Podobnie, po ograniczeniu złożoności wyrażeń (intensyfikacja przeszukiwania prostych wyrażeń): `mul(add(-1, x), protectedDiv(x, 1))`.

Przykładowy eksperyment #2: Znajdź układ logiczny realizujący funkcję XOR, czyli $\{x_1, x_2, y\} = \{(0,0,0); (0,1,1); (1,0,1); (1,1,0)\}$.

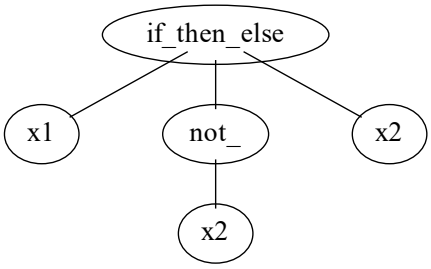
```
def nand(input1, input2):
    return not(input1 and input2)

def if_then_else(input, output1, output2):
    return output1 if input else output2

pset = gp.PrimitiveSetTyped("main", [bool, bool], bool) # let's use
   strongly-typed GP as an example
pset.addPrimitive(operator.xor, [bool, bool], bool)
pset.addPrimitive(operator.or_, [bool, bool], bool)
pset.addPrimitive(operator.and_, [bool, bool], bool)
pset.addPrimitive(operator.not_, [bool], bool)
pset.addPrimitive(nand, [bool, bool], bool) # custom
pset.addPrimitive(if_then_else, [bool, bool, bool], bool) # custom
pset.addTerminal(True, bool)

pset.renameArguments(ARG0="x1")
pset.renameArguments(ARG1="x2")

def eval_expr(individual):
    # transform the tree expression into a callable function
    func = toolbox.compile(expr=individual)
    # evaluate the error between the expression and the target function
    err = 0
    for x1 in (False,True):
        for x2 in (False,True):
            target = x1^x2
            actual = func(x1,x2)
            if target != actual:
               err += 1
    return err,
```

W tym eksperymencie GENERATIONS=100 oraz POPSIZE=150, a w przypadku niepowodzenia – kolejna próba z POPSIZE=1500.

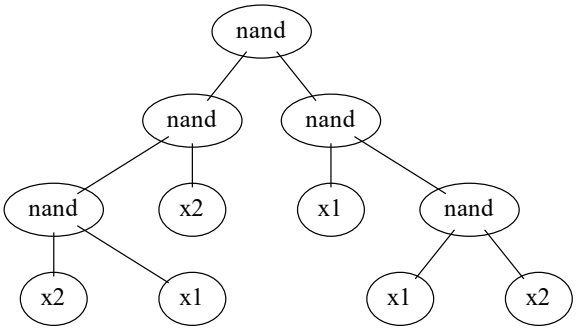
- Wszystkie operatory oraz stała `True` jak w kodzie powyżej:
  `xor(if_then_else(x2, True, x2), x1)`



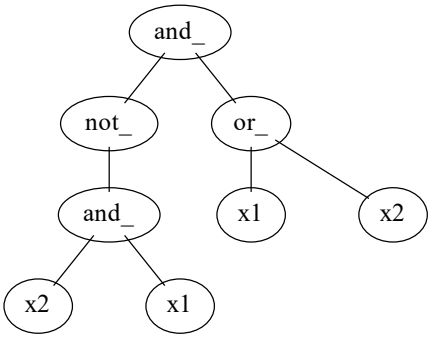
- Samo `if-then-else`: brak idealnego rozwiązania (najmniejszy błąd = 1)
- Tylko `if-then-else` oraz `not`:
  `if_then_else(x1, not_(x2), x2)`



- Tylko `not` oraz `and`: brak idealnego rozwiązania (najmniejszy błąd = 1)
- Samo `nand`:
  `nand(nand(nand(x2, x1), x2), nand(x1, nand(x1, x2)))`



- Trójka `and`, `or`, `not`:
  `and_(not_(and_(x2, x1)), or_(x1, x2))`



Dyskusja: czy warto stosować upraszczanie wyrażeń podczas ewolucji?

Dyskusja: w jakich dziedzinach GP ma szansę konkurować z człowiekiem, w jakich go przewyższyć, a w jakich nie ma szans? Dlaczego?

Polepszanie skuteczności: semantyczne GP (semantyka = zbiór efektów działania osobnika na zbiorze testów) oraz geometryczne semantyczne GP (operatory genetyczne uwzględniają topologię przestrzeni semantyk) [Bak+19].

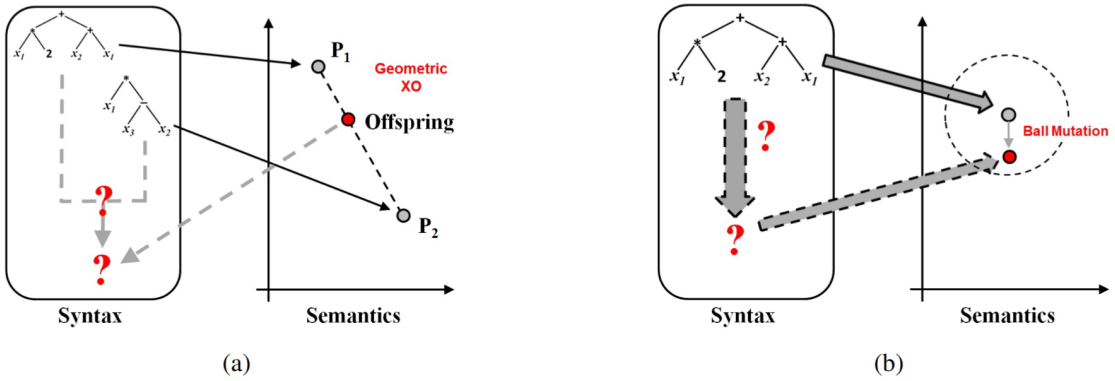


Figure 1: Geometric semantic crossover (plot (a)) (respectively geometric semantic mutation (plot (b))) performs a transformation on the syntax of the individual that corresponds to geometric crossover (respectively geometric mutation) on the semantic space. In this figure, the unrealistic case of a bidimensional semantic space is considered, for simplicity.

Por. wcześniejsze przypomnienie FDC, DPX, "Jak świadomie tworzyć (projektować) skuteczne operatory krzyżowania?" oraz embriogenezy/mapowania.

Przykładowy eksperyment #3: Znajdź algorytm uczenia sieci neuronowej...

Architektura ewolucyjna: „ewolucja regularyzowana" (Rys. 2) [Rea+20].
Odkrycia ewolucji – Rys. 6:

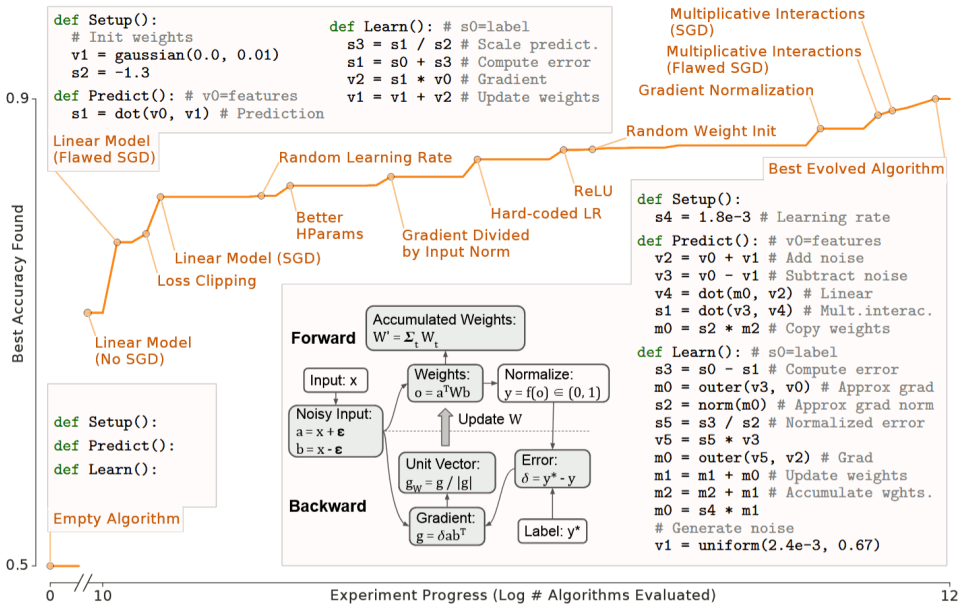


Figure 6: Progress of one evolution experiment on projected binary CIFAR-10. Callouts indicate some beneficial discoveries. We also print the code for the initial, an intermediate, and the final algorithm. The last is explained in the flow diagram. It outperforms a simple

---

Jeśli masz czas i lubisz SF, przeczytaj https://www.teamten.com/lawrence/writings/coding-machines/.

2.6.2 Hiperheurystyki i samo-programujące się algorytmy

Struktura algorytmu ewolucyjnego (rodzaj selekcji, krzyżowania, mutacji, ...) może podlegać kontroli GP (czyli ewolucyjnemu ulepszaniu) [Wąs97; BT96; OG03; Olt05]. GP pozwala na „zbudowanie" algorytmu optymalizacji z modułów, przy czym możliwe są nietypowe architektury: wiele rodzajów mutacji, nietypowe operatory działające na części populacji, wielokrotna selekcja w jednym kroku itp. – zależnie od stopni swobody GP.

Wyniki – lepsze od tradycyjnego, ustalonego AE, ale kosztem...

Porównaj: twierdzenie *No Free Lunch* oraz hiperheurystyki[20] przeszukujące przestrzeń heurystyk i ich kombinacji [Ros05; ÖBK08; Bur+10].

Przykładowe pytania

Przypomnij sobie warianty algorytmów ewolucyjnych przedstawione we wcześniejszych rozdziałach oraz ich skrótowe oznaczenia i upewnij się, że odróżniasz je od siebie – przypomnij sobie ich wyróżniające cechy.

[20] http://en.wikipedia.org/wiki/Hyper-heuristic

2.7 Mechanizmy inspirowane naturą

2.7.1 Reprezentacja

Powielanie genów

Dotychczas mówiliśmy o genotypach haploidalnych. W przyrodzie występują również genotypy diploidalne, składające się z jednej lub wielu par chromosomów (u człowieka są 23 pary), oraz poliploidalne. Taka reprezentacja wymaga specjalnych operacji, takich jak **segregacja** (wybór chromosomów z par), **translokacja** (przemieszczenie genów do innego chromosomu), **duplikacja** (podwojenie genu i umieszczenie kopii w chromosomie) i **delecja** (usunięcie zduplikowanego egzemplarza genu). Ponadto występują problemy związane z interpretacją osobnika (z powodu nadmiarowości informacji w genotypie). W naturze istnieje mechanizm **dominacji** rozstrzygający o tym, które geny uzewnętrznią swoje znaczenie; dominacja może też zachodzić nie tylko pomiędzy chromosomami, ale również wewnątrz nich.

Dyskusja: kiedy nadmiarowość informacji w genotypie może być zaletą i pomagać w optymalizacji?

x'_1	x'_2	x'_3	x'_4	...
x''_1	x''_2	x''_3	x''_4	...
...				

W AE nadmiarowość informacji to pamięć genetyczna: choć niektóre cechy nie objawiają się zewnętrznie, są przechowywane i przekazywane. Przyroda przechowuje rozwiązania, które kiedyś okazały się korzystne, w postaci ukrytych genów, które ulegają dominacji. Geny takie w przyszłości, przy odpowiednich warunkach, mogą zacząć same dominować.

Zastosowanie podobnego mechanizmu w AE ma sens, kiedy funkcja celu jest zmienna – niestacjonarna (odpowiada to zmiennym warunkom środowiska). Stwierdzono eksperymentalnie [SW15, Fig. 2, 3], że w takim przypadku algorytm z diploidalną reprezentacją osobników jest w stanie szybciej dostosować się do zmian środowiska (szczególnie okresowych) znajdując dobre rozwiązania.

Istnieje wiele mechanizmów dominacji: stałe, zmienne, losowe, deterministyczne, dominowanie lepszego chromosomu itd. W tym ostatnim przypadku przerobienie tradycyjnego, haploidalnego algorytmu na poliploidalny jest bardzo proste: to co było haploidalnym genotypem staje się chromosomem, genotyp składa się z wielu chromosomów i jego jakość (dla selekcji) jest jakością najlepszego chromosomu, selekcja działa na całych genotypach, a mutacja i krzyżowanie operują na chromosomach tak, jak dawniej na haploidalnych genotypach.

Relacja gen–fen

W naturze występują:

- geny nadmiarowe: nie mają wpływu na fenotyp osobnika,
- efekt plejotropowy: pojedynczy gen może wpływać na wiele cech fenotypowych,
- efekt poligeniczny: pojedyncza charakterystyka fenotypowa osobnika może być determinowana przez jednoczesne oddziaływanie wielu genów.

Tymczasem, w klasycznym AG: jeden gen — jedna cecha. Można zaimplementować nadmiarowość, plejotropowość i/lub poligeniczność o ile jest się dobrze świadomym konsekwencji oraz wad i zalet każdego z tych mechanizmów – przypomnij sobie pojęcie epistazy ze str. 26.

2.7.2 Operatory

Inwersja (odwrócenie kolejności losowego podciągu genotypu) pozwala na odkrycie korzystnych ustawień i połączeń genów (w sensie ich znaczenia). Chodzi tu o współpracę z operacją krzyżowania, która rozcina łańcuchy genów, rozcinając zarazem długie schematy (zob. twierdzenie o schematach, rozdział 2.2.7). Fragmenty genotypu związane ze sobą nieliniowo i leżące w dużej odległości od siebie będą zwykle rozdzielane przy krzyżowaniu, choć razem mogą stanowić istotne i korzystne połączenie. Operacja krzyżowania w obecności inwersji nie jest już jednak tak oczywista: ponieważ przodkowie posiadają różną kolejność znaczeń genów, po rozcięciu i zamianie odcinków potomkowie zwykle nie mają pełnego garnituru genów (przypomnij sobie, dlaczego taki problem nie występuje przy opisanej w rozdziale 2.2.4 operacji „tasowania", ale występuje w nieporządnym AG z rozdziału 2.2.9). Dlatego ogranicza się zbiór rodziców, którzy mogą być ze sobą skrzyżowani (do takich, którzy dadzą poprawne potomstwo), lub stosuje się różne złożone operacje krzyżowania.

Dopełnienie częściowe polega na zastąpieniu w wybranych osobnikach pewnej części genów wartościami przeciwnymi – dopełnieniami.[21] Celem jest zwiększenie różnorodności osobników i ochrona przed zbytnim ukierunkowaniem populacji. Choć cel zostaje osiągnięty, to odbywa się to kosztem spadku prędkości zbieżności algorytmu genetycznego.

Dodatkowe mechanizmy dotyczące operatorów:

[21] Pierwszy raz zaproponowane w [Fra72, str. 70], ta operacja została nazwana przez autora „migracją", ponieważ odpowiada napływowi osobników tego samego gatunku z zewnątrz demu (imigranci) do tego demu (tubylcy). https://pl.wikipedia.org/wiki/Dem_(biologia) Osobniki w demie i poza nim są zgodne genetycznie, ale są przystosowane do innych środowisk, zatem allele imigrantów nie są przystosowane do warunków demu, do którego napływają – czyli różnią się od alleli tubylców. To powoduje wzrost różnorodności genetycznej w demie po krzyżowaniu imigrantów z tubylcami.

- Zmienne (adaptacyjne) prawdopodobieństwa operatorów: szczególnie przydatne przy niestacjonarnych (zmiennych) funkcjach celu! [przykład stałej funkcji]
- Adaptacyjne operatory: są umieszczane w genotypach osobników i wraz z nimi podlegają ewolucji.
- Krzyżowanie z uwodzeniem: preferencje osobnika zapisane w jego genotypie i ewoluujące.

Pomysły testowane w [Kwa97, opis na str. 142]:

- Tranzycja: przeniesienie pojedynczej kopii genu z jednego genotypu do innego, gdzie umieszczany jest jako gen nadmiarowy.
- Transpozycja: gen nadmiarowy zamienia się miejscem z genem fenotypowym (tj. mającym wpływ na fenotyp).
- Rekrudescencja: u niewielkiej liczby osobników w każdym pokoleniu występuje zwiększone prawdopodobieństwo mutacji i transpozycji. Po rekrudescencji występują znaczne zmiany fenotypowe i większość osobników jest kiepska, ale czasem pojawiają się „*hopeful monsters*".
- Kryzys (katastrofa): w losowym momencie czasu w całej populacji zachodzi znaczna reorganizacja genotypów. Skutki – wymarcie populacji lub możliwość zasiedlenia nowego szczytu w krajobrazie przystosowania.
- Różna intensywność mutacji („makromutacje" i „mikromutacje").

Naśladując naturę w implementacjach należy pamiętać, że nie wszystkie jej mechanizmy zostały poznane i wyjaśnione. Istnieją liczne problemy w analizie ewolucji naturalnej – grupy naukowców posiadają własne, rozbieżne teorie: *punctuated equilibria*[22], jak powstają nowe gatunki, jak działa ewolucja i na jakim poziomie (poziomach?) – gen, osobnik, grupa osobników/stado, mem, populacja/gatunek.

2.7.3 Funkcja celu i logika algorytmu

Zmienna liczebność populacji (np. o rząd wielkości) – np. czas istnienia osobnika w populacji zależy od jego wartości funkcji oceny. Samodostrajanie wielkości populacji; „eksplozje demograficzne" – szukanie optimów.

Algorytmy kulturowe[23] – „ewolucja w ewolucji". Osobniki poprawiają swoje możliwości

[22] https://en.wikipedia.org/wiki/Punctuated_equilibrium
[23] https://en.wikipedia.org/wiki/Cultural_algorithm

uczenia się przez doświadczenie wyniesione z procesu ewolucyjnego (jak ewolucja kulturowa społeczeństw ludzkich[24], czego dowodem miałby być ponadwykładniczy wzrost poziomu rozwoju ludzkości).

Starzenie się osobników – np. usuwanie najstarszych osobników, żeby zrobić miejsce dla nowych. Dyskusja: pomyśl, jaki to będzie miało efekt dla procesu przeszukiwania w porównaniu do usuwania najgorszych osobników i do usuwania losowych osobników?
Nazwana przez autorów „ewolucją regularyzowaną" [Rea+19], może ograniczać problemy wynikające z niedeterministycznej (zaszumionej) oceny („szczęściarze" i „pechowcy" poddawani selekcji) zwiększając różnorodność i sprzyjając eksploracji [Yin+19].

Klonowanie – inna prosta metoda radzenia sobie z niedeterministyczną (zaszumioną) oceną. Żeby nie tracić zasobów obliczeniowych na wielokrotne ocenianie każdego osobnika i uśrednianie ocen, można powtarzać ocenianie proporcjonalnie do jakości osobnika: oprócz mutacji i krzyżowania – operator klonowania i uśrednianie oceny klonów, zobacz [KR01, rys. 12].

Specjalizacja i specjacja

Dyskusja: jaką znasz odpowiedź na pytanie „po co jest podział na płcie w naturze?"

Szczególnym przypadkiem specjalizacji jest w naturze **zróżnicowanie płciowe osobników**. Ponieważ dotychczas nie ma w biologii jednego, pewnego wyjaśnienia tego fenomenu[25] [26], rozważmy teoretycznie bardzo prosty model [Gol03, str. 196], aby zrozumieć wady i zalety zróżnicowania płciowego. W modelu tym zakłada się, że przeżycie potomków s zależy od tego, jaki procent czasu rodzice poświęcą na dwa rodzaje zadań (np. polowanie h i opiekę n): $s = n \cdot h$. Czas, którym dysponuje rodzic jest stały i może być dowolnie dzielony między te zadania. Wprowadzamy dodatkowo opcjonalny (regulowany parametrem a) koszt czasowy związany z „przełączaniem się" rodzica między zadaniami: anh. Podział czasu rodzica spełnia równanie $n + h + anh = 1$.

Gdy nie ma zróżnicowania płciowego, najlepiej jest rodzicowi podzielić czas po równo pomiędzy czynności, $n = h = \frac{1}{2}$. Wzrost a pogarsza s. Z kolei gdy występuje zróżnicowanie płciowe, para rodziców zajmuje się potomkiem i $a = 0$, nie ma znaczenia jak dokładnie oboje rodzice podzielą własne czasy – maksymalne s osiągną dając wspólnie potomstwu taki sam czas n i h. Gdy $a > 0$, natychmiast optymalne stają się tylko takie sytuacje, kiedy każdy z rodziców poświęca cały swój czas na inną czynność (a więc maksymalna specjalizacja). Dzięki eliminacji kosztu „przełączania się" między zadaniami, przypadek zróżnicowania

[24] https://pl.wikipedia.org/wiki/Koewolucja_genetyczno-kulturowa
[25] https://en.wikipedia.org/wiki/Evolution_of_sexual_reproduction
[26] https://web.archive.org/web/20200128050806/http://archiwum.wiz.pl/1999/99113000.asp

płciowego rodziców pozwala uzyskać wyższe s, niż odpowiadający mu przypadek braku zróżnicowania.[27]

Zróżnicowanie płciowe to specjalizacja dwóch rodzajów osobników. W przyrodzie specjalizacja występuje również pod postacią podziału organizmów na **gatunki** (**specjacja**), z których każdy ma swoje miejsce (**niszę**) i zadania [Gol03, str. 200]. Podobne zjawisko może być korzystne w AE: jeśli funkcja posiada kilka optimów, możemy oczekiwać, by pod koniec ewolucji osobniki nie gromadziły się w jednym z nich, a dzieliły się równo pomiędzy optima (albo proporcjonalnie do jakości lokalnych optimów). Osobniki podobne do siebie, zajmujące otoczenie pewnego ekstremum mogą być nazwane gatunkiem.

Efekt powstawania i utrzymania gatunków można uzyskać na wiele sposobów, na przykład:

- stosując zastępowanie przez potomka najbardziej podobnego osobnika (opisana w rozdziale 2.2.2 selekcja według modelu ze współczynnikiem zatłoczenia),
- modyfikując wartość **przystosowania** w zależności od **podobieństwa** osobników.
  nowa ocena := pierwotna ocena/(suma podobieństw do pozostałych osobników)
  Popularna, prosta i skuteczna metoda! Więcej na temat podobnych metod w rozdziale o sterowaniu różnorodnością.

Utrzymywanie gatunków potrafi ostatecznie doprowadzić do odkrycia lepszego rozwiązania, niż w standardowej ewolucji bez modyfikacji wartości przystosowania. Ponadto jest ono korzystne przy optymalizacji wielokryterialnej, kiedy chcemy, by osobniki-rozwiązania pokryły równomiernie front rozwiązań niezdominowanych. Przy stosowaniu specjacji zalecane może być ograniczenie krzyżowania do osobników tego samego gatunku, jeśli krzyżowanie osobników bardzo odmiennych daje słabe rozwiązania.

Koewolucja

Naśladownictwo natury przejawia się w eksperymentach z systemami **współewoluującymi**, w których istnieje kilka populacji wzajemnie na siebie wpływających. Ocena rozwiązań jednej populacji może zależeć od rozwiązań znajdujących się w innej populacji. Koewolucja może posłużyć do wyznaczenia optymalnej strategii, która jest zależna od strategii innych populacji – wszystkie populacje dążą do przewyższenia pozostałych lub współpracują. Taka sytuacja prowadzi do powstania zachowań typowych dla pasożytów, drapieżników, a także wprowadza element konkurencji lub kooperacji (→ rozdział o algorytmach koewolucyjnych, 2.12).

[27] Szwecja: do 1 500 € premii za podział urlopu rodzicielskiego po równo między rodziców.

2.8 Systemy klasyfikatorowe (CFS/LCS/GBML)

LCS is a simple example of a *cognitive architecture*. A cognitive architecture can mean:

- a theory about the structure of the human mind,
- an implementation of such a theory (used in AI) – a cognitive system or agent.

Possible functional criteria of such architectures include flexible behavior, real-time operation, rationality, large knowledge base, learning, development, adaptation, modularity, linguistic abilities, and self-awareness. Competencies and behaviors demonstrated by such systems include – similarly to AGI – perception, memory, attention, actuation, social interaction, planning, motivation, emotion, development and using knowledge efficiently to perform new tasks. Components include memory storage, control components, data representation, and input/output devices [KT20]. More in Sect. 5.12.7.

John Holland envisioned a cognitive system [HR78] capable of classifying the goings on in its environment, and then reacting to these goings on appropriately[28]. To build such a system[29] (see Fig. 2.16) we need

(1) an environment;

(2) receptors/sensors that tell our system about the goings on;

(3) effectors/actuators that let our system manipulate its environment; and

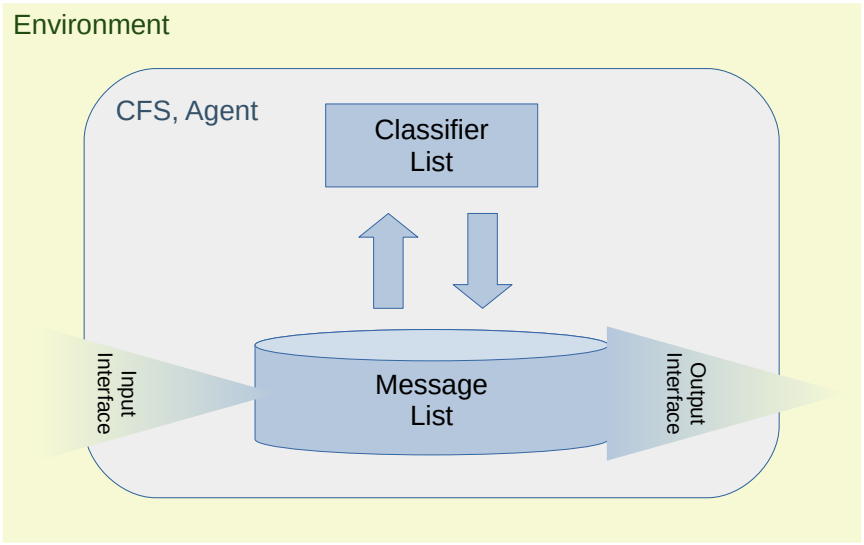
(4) the system itself that has (2) and (3) attached to it, and "lives" in (1).

CFS is quite a general and versatile architecture – consider the following three examples:

- (4) can be a real or simulated robot or creature: (1) is a world with "food" (something beneficial, reward) and "poison" (something detrimental, penalty), and a robot walking (3) across this environment and trying to learn to distinguish (2) between these two items, and to survive while maximizing reward.
- (4) can be a computer. It has inputs (2), outputs (3), and a message passing system in-between, converting certain input messages into output messages, according to a set of rules, usually called a *program*.
- (4) can be a machine learning (ML) algorithm. Inputs (2) provide values of conditional attributes, outputs (3) send out a value of the decision attribute, and a message passing

[28] With minor updates and corrections, from https://www.cs.cmu.edu/Groups/AI/html/faqs/ai/genetic/part2/faq-doc-5.html and from no longer available parts of online slides by Riccardo Poli.

[29] https://en.wikipedia.org/wiki/Learning_classifier_system



Rysunek 2.16: The architecture of the classifier system.

system in-between is the ML model that maps inputs to outputs (predicts outputs based in inputs).

The input interface (2) generates messages – strings of symbols that are written on the message list. Then these messages (internal and external) are matched against the condition-part of all classifiers ("if-then" rules), to find out which actions are to be triggered. The message list is then emptied, and the encoded actions, themselves just messages, are posted to the message list. Then, the output interface (3) checks the message list for messages concerning the effectors. Then the cycle restarts.

You may start from scratch (from tabula rasa – without any knowledge) using a randomly generated classifier population, and let the system learn its program by induction. The input stream are input patterns that must be repeated over and over again, to enable the agent to classify its current situation/context and react on the goings on appropriately, as in the example below:

```
IF small, flying object to the left THEN send @
IF small, flying object to the right THEN send %
IF small, flying object centered THEN send $
IF large, looming object THEN send !
IF no large, looming object THEN send *
IF * and @ THEN move head 15 degrees left
IF * and % THEN move head 15 degrees right
IF * and $ THEN move in direction head pointing
IF ! THEN move rapidly away from direction head pointing
```

Classifier list is a list of classifiers:

```
IF cond-1 AND cond-2 ... AND cond-N THEN action

cond-1, cond-2, ... cond-N / action
```

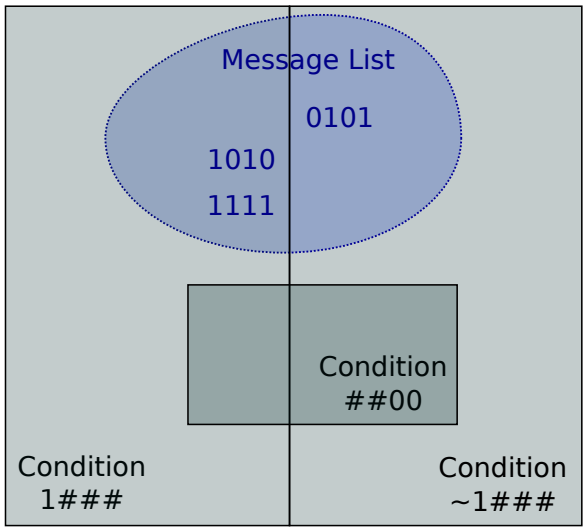
(we use a shorter notation, "," means "AND").

For now, let us assume the simplest language for messages and conditions – they will be encoded by {0, 1, #}. In a real application, we would use the natural language (set of symbols) – this is analogous to the GA/EP difference.

A message matches a condition if all its 0's and 1's are in the same positions as in the condition string. A negated condition is satisfied if no message in the message list matches it:

Message List

Label	Message
a	0101
b	1010
c	1111



Condition	Matched by	Satisfied	Negation satisfied
0101	a	Yes	No (~0101)
1101		No	Yes (~1101)
#101	a	Yes	No (~#101)
1###	b, c	Yes	No (~1###)
##00		No	Yes (~##00)
####	a, b, c	Yes	No (~####)

In CFSs actions are strings of fixed length built from characters in the alphabet {0, 1, #}; their length is usually the same as that of messages. Action strings can be interpreted as parameterized assertions (messages) that go into the message list. When a classifier is activated, a message is built using the following procedure:

- 0's and 1's in the action string are simply copied in the message
- #'s are substituted by the corresponding characters in the message that matches the first condition in the condition part. For this reason the # character is also called the pass-through operator.

Example:

Message List	
Label	**Message**
a	0101
b	1010
c	1111

Classifier List	
Label	**Classifier**
i	#11#, ∼#110 / 00##
ii	###1, ∼#110 / ###0
iii	##1#, ∼1110 / 0##0

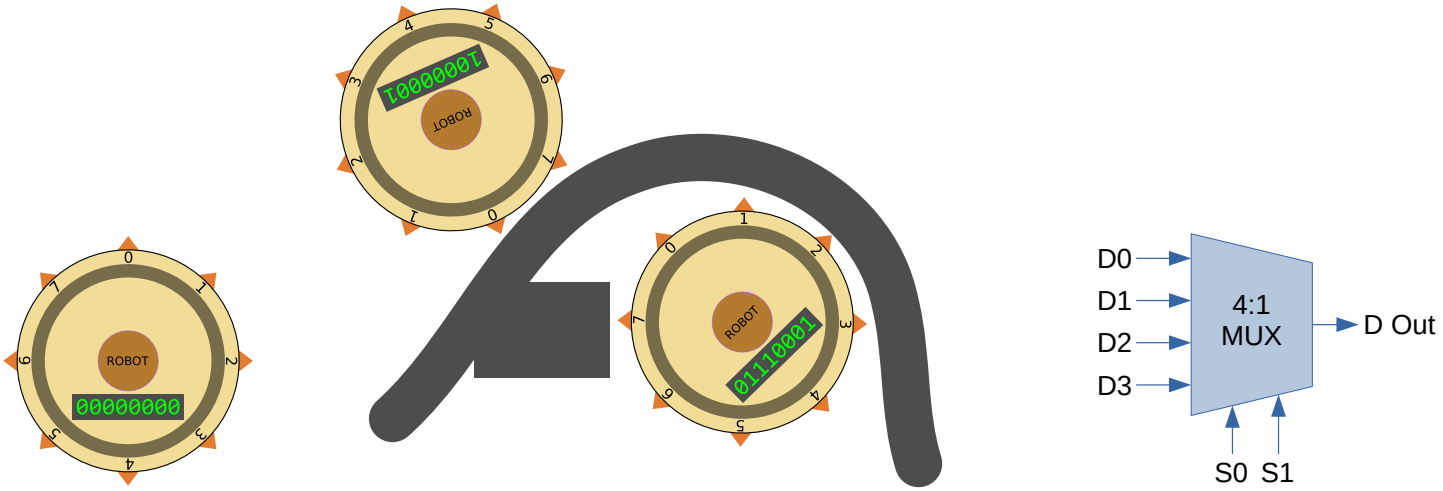
The following set of messages will be produced:

Message	Reason
0011	Posted by i, c matches cond-1
0100	Posted by ii, a matches cond-1
1110	Posted by ii, c matches cond-1
0010	Posted by iii, b matches cond-1
0110	Posted by iii, c matches cond-1

The only actions allowed in the basic CFS are assertions, so messages (facts) cannot be explicitly deleted. However, as the message list is of finite size, old messages can be overwritten. Many classifiers can be activated in parallel by the messages in the message list. A classifier can post as many messages as the number of messages matching its first condition. Conflict resolution is only necessary if the active classifiers can produce more messages than entries in the message list.

2.8.1 Input and output interfaces

The input interface can be thought of as a mechanism by which the CFS can obtain information about the environment. The messages posted by the input interface are often descriptions of the state of a set of (binary) detectors that can sense various features of the environment.



Rysunek 2.17: Simple examples of CFS control: a robot with eight proximity sensors and a 4:1 multiplexer.

The output interface can be realized by any procedure capable of selecting (deleting) and using some messages in the message list. Usually we imagine that the bits in a message picked up by the output interface represent (and control) the state of a set of effectors which act on the environment, for example to control the actions of a robot.

In any case the output interface must be able to recognize which messages are input messages posted by the input interface, which are internal messages posted by classifiers, and messages meant to be output messages. This is obtained by using tags usually consisting of two additional bits in the messages that are interpreted in a special way. An example convention:

Tag	Interpretation
01	Internal Message
11	Internal Message
00	Output Message
10	Input Message

Or:

bit 1: Input/Output (from/to)
bit 2: Inside/Outside

For example, a CFS which controls a robot (Fig. 2.17, left) might have some classifiers devoted to obstacle avoidance, like

```
10 1####### / 00 0100 0000
10 ##1##### / 00 0001 0000
```

```
10 ####1### / 00 1000 0000
10 ######1# / 00 0010 0000
```

The conditions of these simple classifiers just check whether there is an obstacle in front, on the right, on the back, on the left, respectively. The action-strings have the following interpretation:

- The first two bits indicate that the messages are for the effectors in the output interface
- The following four bits indicate which movement (forward, backward, right, left) is appropriate to avoid the obstacle
- The other bits are padding.

In the case of the 6-bit 4:1 multiplexer (Fig. 2.17, right), the following classifier list would provide the correct behavior:

```
10 00 0### / 00 0 00000
10 00 1### / 00 1 00000
10 01 #0## / 00 0 00000
10 01 #1## / 00 1 00000
10 10 ##0# / 00 0 00000
10 10 ##1# / 00 1 00000
10 11 ###0 / 00 0 00000
10 11 ###1 / 00 1 00000
```

- The first two bits of the conditions mean "input message"
- The next two bits of the conditions are interpreted as address (i.e., selector) bits
- The other characters of the conditions as checks for the state of the data lines
- The first two bits of the action are interpreted as the "output message" tags
- The third bit as the output of the multiplexer
- The remaining bits as padding.

2.8.2 Main cycle

1. Activate the input interface and post the input messages it generates to the message list.
2. Perform the matching of all the conditions of all classifiers against the message list.

3. Activate the fireable classifiers (those whose conditions are satisfied) and add the messages they generate to the message list.
4. Activate the output interface, i.e. remove the output messages from the message list and perform the actions they describe; go to 1.

In the previous two examples we have considered non-overlapping rulesets, i.e. sets of classifiers in which one and only one classifier is active in the presence of a given message in the message list. An alternative, more parsimonious way of using classifiers is to organize them to form a default hierarchy (pol. *hierarchia domniemań*), in which some very general classifiers provide the default behavior for the system. Other, more specific classifiers refine such a behavior in the presence of messages improperly handled by the default ones.

For example, for the 6-bit multiplexer we could use:

```
10 00 0### / 00 0 00000
10 01 #0## / 00 0 00000
10 10 ##0# / 00 0 00000
10 11 ###0 / 00 0 00000
10 ## #### / 00 1 00000
```

"Unless there is a 0 on the data line currently addressed, set the output to 1"

Default hierarchies are not only more parsimonious ways of programming CFSs, they also make the "search" for a good program much easier (for example we can successively refine the hierarchy). This is very important when we introduce learning and rule discovery in CFSs.

2.8.3 *Learning* Classifier Systems (LCS)

The real power of CFSs derives from the possibility of adding adaptation mechanisms to the basic architecture, so that they can learn to behave appropriately in the environment[30] or, more simply, to perform useful tasks. There are two ways in which we can adapt (i.e. improve the performance of) a CFS:

- Adaptation by **credit assignment**: changing the way existing classifiers are used.
- Adaptation by **rule discovery**: introducing new classifiers in the system.

[30] http://www.alife.pl/adaptacyjny-system-klasyfikatorowy

2.8.4 Good and bad classifiers

Not all the classifiers active at a given cycle will in general produce a message which will lead (directly or after additional processing) to a good action. For example, if a CFS controls an autonomous agent who has to find a source of energy (i.e. food) to survive, some classifiers will post actions which will lead to get some food; others will post actions which will delay the search for food. In summary, in a CFS there are usually some classifiers which are better than the others.

2.8.5 The need for competition

Unfortunately, the basic CFS is absolutely, blindly fair and gives to all the fireable classifiers the same chances of posting messages, and therefore of influencing the overall behavior of the system. To maximize the performance of a CFS it would be nice to give higher priority to the messages posted by good classifiers and low priority to the others. Even better – to prevent low quality classifiers from posting their messages at all if other, better classifiers are fireable. This behavior could be obtained if the classifiers had to compete to post their messages, basing on some measure of quality of classifiers.

2.8.6 Quality of classifiers

There may be several properties of classifiers on which a quality measure can be based. The two most important ones are:

- The usefulness of the classifier in determining the good performance of the whole system: *strength*.
- The relevance of a classifier in a particular situation: the *specificity* of the classifier = its (length – number_of_#'s) / length.

Strength and *specificity* are usually combined into a single measure, the *bid* a classifier makes in the auction (competition):

$$bid = k \cdot strength \cdot specificity \qquad (k \text{ is a constant} \approx 0.1)$$

- To maintain parallelism, in the auction there must be more than one winner.
- To avoid premature convergence, we use a noisy (probabilistic) auction, in which classifiers have a bid-proportionate winning probability.

- As a classifier's specificity is a constant, the strengths associated to classifiers are the only quantities that can be varied to influence the auction and therefore the behavior of a CFS.

2.8.7 Adaptation by credit assignment

A learning algorithm for CFS should be capable of modifying the *strengths* of classifiers to optimize the behavior of the system as a whole. To do that, the algorithm will need to have some kind of information about the quality of the behavior of the system. This information will come from the environment, e.g. from an external observer (a teacher), or from some other part of the system, e.g. from an internal variable representing the level of energy of the system. The simplest (more biologically plausible) form of behavioral-quality information would be a scalar value, termed *reward*, whose sign tells the learning algorithm whether the actions of the system are good (positive reward) or bad (negative reward or punishment) and whose magnitude may be fixed (e.g. $+1$, -1, 0) or variable. If rewards only are available, the learning algorithm will have to solve the so-called credit assignment problem: which classifiers are responsible (and to which extent) for the good or bad overall behavior of the system?

2.8.8 The Bucket Brigade algorithm

The bucket brigade (pol. *brygada kubełkowa/wiaderkowa*) algorithm is a parallel, domain-independent, local credit-assignment-based learning algorithm:

1. If there is a reward (or punishment), add it to the strength of all the classifiers active in the current major cycle.
2. Make each active classifier pay its bid to the classifiers that prepared the stage for it (i.e. posted messages matched by its conditions).

The stage-preparing classifiers had to pay (invest) their bids in the previous cycle (when they were active). In this cycle they get back their "money". In turn, the classifiers that prepared the stage for the stage-preparing classifier received some money two cycles earlier, and so on. Good classifiers are rewarded often so their strengths tend to grow. So, they will make bigger bids, and so they will pay more to their stage-preparing classifiers. In turn those classifiers will be able to pay more to their stage-preparing classifiers, and so on. During time, strength is propagated backwards, and each classifier receives the correct share of credit for the good (or bad) behavior of the system as a whole. With time, strengths reach a (nearly) constant equilibrium value.

Układ oceniający w zadaniu klasyfikowania

Układ oceniający spełnia podstawową rolę w procesie uczenia, gdyż od jego działania zależy szybkość i skuteczność tego procesu. Zadaniem układu oceniającego jest określenie jakości poszczególnych reguł w systemie. Z ocen tych korzysta AG przy tworzeniu następnej populacji reguł, a także użytkownik, przy wykorzystywaniu otrzymanych reguł. Ocena zależy od dwóch czynników: od tego, czy dana regułą dopasowała się do kolejnego przetwarzanego komunikatu i od „nagrody" jaką system otrzymał od środowiska w momencie gdy określona przez niego klasa decyzyjna pokrywała się z decyzją przewidzianą dla danego przykładu uczącego w zbiorze uczącym.

Algorytm oceniania zastosowany w systemie nosi nazwę *bucket brigade*. Algorytm ten można porównać do rynku usług informacyjnych. Reguły nabywają i sprzedają prawo do obrotu informacją. W takim układzie owe reguły tworzą „łańcuszek pośredników informacji" od detektorów (ze środowiska) do efektorów (do środowiska). Tłumaczy to nazwę algorytmu.

W algorytmie występują dwa zasadnicze elementy. Są to instytucja przetargu i instytucja izby rozrachunkowej. Jeżeli reguła dopasuje się do nadchodzącego komunikatu, oznacza to, że mogłaby zostać uaktywniona. Jednak tak się nie dzieje: reguła nabywa jedynie prawo do udziału w przetargu. Z każdą regułą związana jest informacja o jej sile (nabywczej) – *strength*. Każda reguła, która została dopasowana do komunikatu, składa swoją ofertę (*bid*), proporcjonalną do swojej siły. Reguły dobrze dostosowane (mające dużą siłę) składają odpowiednio wyższą ofertę. W instytucji przetargu wyłania się regułę (reguły), która będzie mogła wysłać swój komunikat (wykonać akcję, podać decyzję, zaklasyfikować – zależnie od interpretacji komunikatu).

Po wybraniu zwycięskiej reguły (reguł) musi nastąpić uregulowanie płatności w izbie skarbowej. Każda zwycięska reguła wpłaca do izby wielkość swojej oferty (*bid*). Odpowiednio maleje jej siła. Izba skarbowa natomiast rozdysponowuje tę ofertę pośród wszystkich *reguł, które nadały komunikaty uaktywniające regułę zwycięską*. Zapobiega to monopolizacji zbioru reguł przez jeden ich typ, co pozwala utrzymać pewną podpopulację reguł dobrych, choć nie podobnych do siebie. Analogią tego zjawiska w przyrodzie (i w AE) jest współżycie różnych gatunków w swoich niszach ekologicznych. Takie postępowanie jest także zgodne z metodologią uczenia maszynowego, gdzie nie próbuje się szukać jednej uniwersalnej reguły, lecz ich dobrego, użytecznego zbioru.

Dla opisanej ogólnej zasady potrzebne są konkretne metody wykorzystywane w systemie klasyfikującym. Pierwszą jest wypłacanie nagrody (przychodu) ze środowiska. Ma to miejsce wówczas, gdy decyzja jaką podjęła dana reguła dla nadchodzącego komunikatu jest zgodna z rzeczywistą decyzją (podaną w zbiorze uczącym) dla tego komunikatu. Odbywa się to poprzez zwiększenie siły danego klasyfikatora. Drugim elementem upodabniającym całość

układu oceniającego do rynku są podatki. Równanie opisujące siłę S reguły w kolejnym cyklu przetwarzania $t+1$ możemy zapisać jako funkcję jej dotychczasowej siły $S(t)$, wypłaty na rzecz innych reguł B, podatków T, oraz przychodów R:

$$S(t+1) = S(t) + R(t) - T(t) - B(t)$$

Wielkość R jest jednym z parametrów systemu, a początkowa wartość $S(0)$ jest cechą każdej z reguł które są początkowo umieszczone w zbiorze reguł. Wyjaśnienia wymagają wielkości B (oferta) oraz T (podatek). Oferta zależy od siły:

$$B = C_{bid} \cdot S$$

C_{bid} jest parametrem systemu i określa, jaka część siły staje się ofertą (np. 10%). Bardziej skomplikowanie wygląda sprawa podatku. Rolą podatków jest usuwanie z pamięci takich reguł, które są bezproduktywne. Nigdy się nie dopasowują, ale mając pewną siłę początkową cały czas istnieją w systemie. To, że nigdy się nie dopasowują do przykładów uczących może świadczyć o ich zbyteczności. Są dwa rodzaje podatków – obrotowy i stały (od istnienia). Ten pierwszy płacą tylko te reguły, które się dopasowały. Drugi podatek jest płacony przez wszystkie reguły. Jeżeli reguła płaci podatek przez cały czas, a nie ma możliwości zwiększenia swojej siły (poprzez otrzymanie części oferty zwycięskiego klasyfikatora lub nagrody od środowiska), wówczas jej siła zmaleje do 0. Podatki obrotowy i stały są pewną częścią siły reguły, S. Jaka to część, jest określone parametrem systemu.

Jeśli razem z CFS stosuje się AE, wtedy selekcja zależy od siły reguł, zatem reguły z zerową siłą nie zostaną wylosowane przy tworzeniu nowej populacji.

Chociaż widzimy wiele elementów w formule na siłę $S(t+1)$, udowodniono, że wszystkie są niezbędne – stanowią minimalny zestaw, który zapewnia skuteczne działanie mechanizmu ustalania sił reguł.

2.8.9 Adaptation by rule discovery

In addition to credit assignment, in order to learn we need a way to introduce new classifiers to the system. Evolutionary algorithm can be used to optimize and adapt a CFS in two ways:

- Considering the classifier list as a single individual whose chromosome is obtained by concatenating the conditions and actions of all classifiers (the "Pittsburgh" ("Pitt") approach, De Jong)
- Considering each classifier as a separate individual (the "Michigan" approach, Holland).

In the Pittsburgh approach, the fitness of each CFS is determined by observing the behavior of the system for a certain amount of time or on some test data. The EA optimizes the CFS by breeding and making *compete* different sets of classifiers. The Michigan approach requires a fitness measure for each classifier. If used with the bucket brigade algorithm, the strength of the classifier can be taken as its fitness. In this case, the EA optimizes the CFS by breeding and making *compete* and *co-operate* different classifiers.

Algorytm ewolucyjny (w podejściu Michigan)

LCS to CFS z mechanizmami ewolucyjnymi, które optymalizują zestaw reguł. Algorytm ewolucyjny stosowany w takim systemie jest dość typowy, jednak posiada pewne cechy potrzebne z punktu widzenia systemu uczącego się. Głównym zadaniem AE jest zasilanie zbioru reguł nowymi, lepszymi regułami.

Siła reguły pełni dla AE rolę oceny (*fitness*) reguły. Ocena ta jest wykorzystywana przy selekcji. Dodatkowo można stosować mechanizm „ścisku" (zob. rozdział 2.2.2) – eliminuje on z populacji reguły identyczne poprzez zastępowanie regułami nowymi istniejących już reguł im podobnych. Po dokonaniu selekcji reguł, wykonujemy na nich krzyżowanie oraz mutację. Zwykle nie zmieniamy całej populacji, a jedynie jej część. Ma to zapobiegać wymianie całego materiału genetycznego i służy utrzymaniu części reguł niezmienionych. Dzięki temu możemy zachować wysoki poziom bieżącej efektywności w czasie, gdy system uczy się sprawniejszych wzorców zachowania – nie interesuje nas rozwiązanie zupełnie nowe, a jedynie poprawa rozwiązania dotychczasowego. W systemach uczących się nie chodzi o znalezienie jednej reguły, lecz ich zbioru (podczas gdy typowy AE jest nastawiony na zbieżność do jednego, najlepszego rozwiązania). Mamy wpływ na to, jaka część populacji zostanie wymieniona: odpowiedni parametr nazywa się współczynnikiem wymiany i określa jaka część populacji pierwotnej zostanie zastąpiona nowymi chromosomami (np. 20%).

Kolejnym parametrem wpływającym na działanie AE jest okres jego wywoływania. Tradycyjnie, gdy AE jest częścią większego systemu i służy optymalizacji jakiejś funkcji, wywołujemy go w każdym cyklu działania tego systemu. W systemie uczącym, AE wywołujemy co kilka cykli działania (deterministycznie lub średnio) lub przy zajściu określonych zdarzeń (np. spadek trafności klasyfikowania). Jeden cykl działania systemu uczącego się oznacza przetworzenie jednego komunikatu nadchodzącego ze środowiska. Choć teoretycznie można by wywoływać AE w każdym kroku działania systemu uczącego, takie postępowanie nie przynosi nadzwyczajnych wyników – a dość istotnie zwiększa nakłady obliczeniowe, ponieważ AE w porównaniu do pozostałych elementów systemu uczącego bywa kosztowny obliczeniowo.

2.8.10 Podsumowanie

Powyżej opisana została klasyczna idea GBML (analogiczna w swojej prostocie do AG). W praktyce elementy systemu i całą jego architekturę dostosowuje się do konkretnego problemu. Modyfikacje i ulepszenia systemów LCS dotyczą języka przekazu (wiele symboli to duża siła wyrazu gramatyki), zasad reprezentacji (złożone reguły), operatorów genetycznych, mechanizmów nagradzania (np. integracja z algorytmami uczenia ze wzmocnieniem), itd. Zastosowania są bardzo różnorakie ze względu na uniwersalność samej idei. Przykładowo można tu wymienić eksplorację nieznanego środowiska przez robota, systemy sterowania, trudne gry (np. poker), budowę sieci semantycznych, wyuczenie się reguł postępowania w diagnostyce medycznej i leczeniu, i wiele innych.

Ogólnie, dla złożonych problemów, systemy LCS/GBML zachowują się lepiej od tradycyjnych systemów klasyfikujących i tradycyjnych algorytmów uczenia maszynowego. W ich działaniu widzimy analogie do głębokich i rekurencyjnych sieci neuronowych, jednak LCS oferują symboliczną, zatem w pewnych zastosowaniach łatwiejszą do interpretacji niż sieci neuronowe, formę wiedzy.

2.9 Inne techniki w AE

2.9.1 Wiele kryteriów

Multiple objective GA/EA (MOGA/MOEA):

- randomized weighting (utility function),
- Pareto optimality (dominance relation) and ranking. Memory ("archive") helps build better Pareto fronts,
- dedicated selection techniques [Kun05] such as SPEA (strength Pareto EA) and NSGA (non-dominated sorted GA), and the "crowding distance" operator; https://www.youtube.com/watch?v=Q8KvtVCTjEw.

2.9.2 Wzbogacanie wiedzą

You get some specific problem to optimize (e.g., optimize satellite trajectory to maximize image collection (Earth coverage) per day). How do you customize the algorithm taken from some evolutionary library/toolkit/framework? [discussion]

The problem turns out to be more difficult than expected. The algorithm seems to get stuck and does not produce satisfactory results. What do you do to help? [discussion]

Incorporating knowledge into EA:

- use many fitness functions (multiple objectives) for additional guidance,
- use knowledge-rich representations and complex genotype-phenotype mappings,
- use knowledge seeding (include good solutions in the initial population),
- to evaluate individuals, compare them with a database of good/bad cases (case-based knowledge).

Wzbogacanie wiedzą jest szczególnie potrzebne w skomplikowanych problemach optymalizacji – przykładem jest *evolutionary design* czyli projektowanie ewolucyjne omawiane w rozdziale 5.10.3.

2.9.3 Sterowanie różnorodnością i techniki jakość–różnorodność

We learned how to preserve diversity by using the crowding model or the convection selection (Sect. 2.2.2), and by speciation (Sect. 2.7.3). One recent example of optimization where maintaining diversity of solutions was important was the Michigan-style LCS (Sect. 2.8).

Controlling the diversity in fitness. In the order of complexity:

- FUSS (Fitness Uniform Selection Scheme) [HL06].
- FUDS (Fitness Uniform Deletion Scheme) [LH05].
- Convection selection – already discussed.
- HFC (Hierarchical Fair Competition) [Hu+05].

A more complex diversification mechanism (albeit requiring the introduction of additional evaluation criteria) is MAP-Elites [MC15]:

- Define your fitness as usually (for example: maximize velocity of a robot).
- Define other features of each solution (for example: robot size, weight, energy consumption).
- Discretize ranges of these other features, thus creating a multi-dimensional grid with "cells".
- During evolution, select an individual from a random cell, mutate/crossover it, evaluate its fitness and features, and put into the appropriate cell.
- In each cell keep only one solution with the best fitness found so far.

Variants:

- Keep more than one solution in each cell.
- Start with a coarse discretization of features and gradually increase the number of intervals (resolution).

Controlling the diversity in solution content.

- Niching/speciation (already discussed).
- Novelty search (i.e., niching but disregarding the fitness component).
- Two-criteria optimization of fitness and diversity (no aggregation like in niching).
- NSLC (Novelty Search with Local Competition).

2.9.4 Obsługa ograniczeń

Constraints (hard, soft): when solutions violate them,

- correct
- penalize (pressure)
- remove (prevention)

Think about advantages and disadvantages of each approach. When each approach is suitable and when it is not?

2.10 Równoległe algorytmy ewolucyjne

Parallel EA (PEA), Równoległe AE [Pod97]. Zaleta – przyspieszenie obliczeń, ponadto wpływ na zbieżność, eksplorację i eksploatację.

I. Algorytmy scentralizowane. Ocena osobników równolegle, ale podział zadań między procesami w schemacie master-slave (por. [KU17]). LAN, topologia gwiazdy.

II. Algorytmy o gruboziarnistej równoległości (alg. subpopulacyjne a.k.a. model wyspowy). Równoległe procesy przetwarzają autonomiczne subpopulacje. Wymiana najlepszych osobników. Duża moc komputerów, niewielka komunikacja.

   (a) Subpopulacje są początkowo losowane z równym rozkładem z całej dziedziny.

   (b) Subpopulacje dzielą dziedzinę.

(c) Subpopulacje dzielą zakres wartości funkcji celu (rozpraszanie konwekcyjne z rodziału 2.2.2).

III. Algorytmy o drobnoziarnistej równoległości (alg. komórkowe). Każdy osobnik przetwarzany przez osobny proces połączony z procesami sąsiednimi (operacje genetyczne). Duża komunikacja!

PEA można też podzielić ze względu na komunikację – synchroniczną lub asynchroniczną.

ad. II.

Migracja osobników:

- model migracji: emigracja (osobnik opuszcza swoją subpopulację) i imigracja (kopia osobnika rozsyłana)
- topologia sieci połączeń komunikacyjnych: pierścień, torus, krata, hipersześcian, drabina... (zależnie od architektury sprzętowej)
- zasięg migracji
  - model globalny (*island model*): osobniki mogą migrować do dowolnej subpopulacji
  - model lokalny (*stepping stone model*): osobniki mogą migrować tylko do sąsiednich subpopulacji (wymaga zdefiniowania sąsiedztwa)

ad. III.

Maszyny o masowej równoległości. Każdy osobnik przetwarzany przez osobny proces, ewaluacja nie wymaga komunikacji, ale selekcja i rekombinacja – tak. Dlatego te dwie ostatnie operacje przeprowadza się na pewnym otoczeniu osobnika (by zmniejszyć nakłady na komunikację).

Topologia sieci ma duży wpływ na dynamikę i zbieżność. Najpopularniejsze topologie: krata, torus, hipersześcian, drabina.

Zaleta PEA (II i III) – utrzymanie większej różnorodności osobników, niż GA z pojedynczą populacją. Szybciej znajduje optima, nie dochodzi do przedwczesnej zbieżności całej populacji.

Problemy i kierunki badań:

ad. II.

- Lepsze wykorzystanie właściwości, poznanie, zrozumienie szczególnych cech tego typu PEA.
- Wyznaczenie optymalnych wartości parametrów opisujących architekturę tego typu PEA – np. na ile subpopulacji dzielić, jaka część osobników ma być wymieniana pomiędzy subpopulacjami itp.

ad. III.

- Opracowanie operatorów które nie wymagają tak dużych nakładów na komunikację.
- Przeciwdziałanie osłabieniu presji selekcyjnej, spowodowanej dużą decentralizacją.

2.11 Jak zrobić skuteczny AE? Na co zwrócić uwagę?

- Dobre kodowanie i operatory (wiedza dziedzinowa) wynikające z identyfikacji cech rozwiązania ujawniających globalną wypukłość.
  Jeśli AE jest mocno dopasowany do problemu, nazywa się go silnie typowanym (*strongly typed*).
- Funkcja oceny – łagodne, ciągłe zmiany, dobrze dobrana. Lepsza „średnia odległość od źródła" niż „czy znaleziono źródło". Należy uwzględnić spodziewany krajobraz ewolucyjny, myśleć kategoriami algorytmu. Mała zmiana w genotypie powinna powodować małą zmianę w fenotypie (małą zmianę wartości funkcji celu).
- Dobrze zdefiniowane środowisko, nie za trudne na początek – przykład: robot, symulacje, ...; możliwość koewolucji rozwiązania i środowiska.
- Populacja początkowa: czasem opłaca się zacząć od dobrych rozwiązań. Podczas działania warto rozważyć wykorzystanie prostej, lokalnej optymalizacji (wtedy powstanie „*genetic local search*" nazywany też czasem algorytmem memetycznym, por. rozdział 5.5.2).
- Dokładna ocena i analiza działania algorytmu: musimy wiedzieć co się dzieje, czy wystarczyło czasu na działanie optymalizacji, czy nie za silna zbieżność, czy nie marnujemy czasu, itd. Powinniśmy obserwować to na odpowiednich wykresach, w przeciwnym razie uruchomienie „czarnej skrzynki AE" nie ma sensu – z powodu źle dobranego pojedynczego parametru algorytmu możemy uzyskać beznadziejne wyniki nawet po długim czasie (podobnie jest z każdym sparametryzowanym algorytmem optymalizacji). Właściwe, świadome wykorzystanie algorytmu jest szczególnie ważne gdy nie

znamy optimum (czyli w większości praktycznych sytuacji) – wtedy widzimy że „algorytm działa" i odkrywa coraz lepsze rozwiązania, ale nie mamy pojęcia na ile lepiej mógłby działać.

- Niedeterminizm oceny może mieć wiele źródeł. Rozważ następujące sytuacje, kiedy optymalizujemy:

  1. Model UM oceniany C-V: bardzo czasochłonna ocena, naturalny niedeterminizm, nie da się go usunać (chyba, że olbrzymim kosztem obliczeniowym – wszystkie możliwe podziały C-V).
  2. Robota/konstrukcję 3D w symulacji: bardzo czasochłonna ocena, determinizm wyidealizowanej symulacji – wiemy, że on nie odpowiada rzeczywistości.
  3. Bota do gry wieloosobowej. Jak dla przykładu robota powyżej – można łatwo zapewnić determinizm przeciwników, ale co to spowoduje...
  4. Trasę TSP: niedeterminizm czasów przejazdu w rzeczywistości, ale jeśli mamy średnie czasy w macierzy, to ocena permutacji jest błyskawiczna.

     A jeśli nie mamy średnich czasów, tylko wyidealizowany symulator jeżdżącego agenta?

  Kiedy w rzeczywistym środowisku występuje niepewność, niedeterminizm lub bardzo/nieskończenie wiele stanów wpływających na ocenę rozwiązania: podczas jego oceniania dodaje się w symulowanym środowisku losowy szum[31] – w ten sposób rozwiązania stają się odporniejsze (*robust*), choć ich optymalizacja jest trudniejsza, bo ocena nie jest już deterministyczna. Takie podejście często stosuje się przy optymalizacji robotów i w projektowaniu ewolucyjnym.

- Jeśli ocena jest niedeterministyczna, to pomimo tego, że AE (bez elitaryzmu) są stosunkowo odporne na jej niedokładności („niepewność"), żeby wartość oceny była stabilniejsza stosuje się uśrednianie – wielokrotną ewaluację każdego rozwiązania. Selekcja promuje niestety „szczęściarzy" i wielokrotne ocenianie każdego osobnika, radykalnie zwiększając koszt obliczeniowy, jedynie zmniejsza skalę tego problemu. Ilukrotnie przy 100-krotnym powtórzeniu oceny osobnika? Przypomnienie: [KR01, rys. 12] oraz metody starzenia i klonowania z rozdziału 2.7.3.

[31] Nawiązując do dyskusji o roli i rodzajach losowości z rozdziału 2.2.2, rozważ i porównaj sytuacje: stałe kombinacje wartości parametrów oceny (regularna siatka), stałe kombinacje wartości parametrów oceny (nieregularne, chaotyczne), niedeterministyczne kombinacje (zmieniające się przy każdej ocenie).

2.12 Architektury koewolucyjne

Koewolucja: wiele gatunków (grup organizmów) – co najmniej dwie – wpływa nawzajem na swoje procesy ewolucyjne.

2.12.1 Kooperatywne

Osobna prezentacja bazująca na publikacji [PD00], por. https://deap.readthedocs.io/en/master/examples/coev_coop.html.

2.12.2 Konkurencyjne

Typowy przykład: koewolucja (optymalizacja) strategii [Elf+21]. Osobnik reprezentuje wiedzę zawartą w strategii (np. mogą być to wagi kryteriów używanych do oceny sytuacji na planszy w grze). Ocenę osobnika uzyskuje się np. przez rozegranie wielu partii z pozostałymi osobnikami (każdy gra według własnej strategii).

Dyskusja: czy włączając taki proces, jeśli odpowiednio długo poczekamy, otrzymamy mistrzowską strategię? Jeśli nie, to dlaczego? (podaj przyczyny – listę ewentualnych problemów).

Trudności: chcemy *arms race* (ciągła konkurencja), lecz możemy zostać w *MSS – Mediocre Stable State* (stagnacji – kiepskim, stabilnym stanie). Zbyt silny przeciwnik nie pozwoli rozróżnić przeciętnego i złego rozwiązania; zbyt słaby – przeciętnego i dobrego. Ocena każdego zależy od innych (zewnętrzny, obiektywny „nauczyciel" rozwiązuje ten problem równocześnie eliminując zalety koewolucji). Ocena strategii może być nieprzechodnia.

- Dyskusja przykładowych sytuacji: GP (populacja wyrażeń i populacja testów), strategie gry w szachy, piłka nożna, tenis i nieprzechodniość „lepszości", papier–kamień–nożyce, lokalna szkoła szermierki i różnorodność (por. *exploiter agents* w AlphaStar), natura.
- Pojęcia: arms race, Red Queen[32], MSS.
- Problemy: brak lub utrata gradientu, zapętlenie (cykle, nieprzechodniość relacji porównania wynikającej z oceniania), brak monotoniczności (postępu) [Mic09].
- Rady: *competitive fitness sharing* (zwiększanie oceny tych rozwiązań, które wygrywają z testami (przeciwnikami) trudnymi dla pozostałych rozwiązań [RB95]), specjalny

[32] https://en.wikipedia.org/wiki/Red_Queen_hypothesis

dobór zbioru testów, utrzymywanie *hall of fame* lub zbiorów Pareto-niezdominowanych rozwiązań i testów.

Rozdział 3

Inne techniki optymalizacji inspirowane naturą

3.1 Algorytmy mrówkowe (AS, ACO) i inteligencja grupowa

The behavior of social insects in general, and of ant colonies in particular, has since long time fascinated researchers in ethology and animal behavior, who have proposed many models to explain their capabilities. Ant algorithms have been proposed as a novel computational model that replaces the traditional emphasis on control, preprogramming, and centralization with designs featuring autonomy, emergence, and distributed functioning. These designs are proving flexible and robust, able to adapt quickly to changing environments and to continue functioning even when individual elements fail.

Ant algorithms are a part of Swarm Intelligence (pol. *inteligencja grupowa/zbiorowa/roju*). A particularly successful research direction in ant algorithms is known as "ant colony optimization"[1] (ACO). ACO has been applied successfully to a large number of difficult combinatorial problems like the quadratic assignment (QAP) and the traveling salesman (TSP) problems, to routing in telecommunications networks, to scheduling problems. In ACO, the discrete optimization problem is mapped onto a graph called *construction graph* in such a way that feasible solutions to the original problem correspond to paths in the construction graph.

An "Ant System" (AS) – a particular ant colony optimization algorithm – was introduced in

[1] https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms

1992, and was inspired by the observation of the behavior of ant colonies: emergence[2] of global properties following the mutual interaction among many elementary agents performing simple actions. The algorithm is easy to parallelize. How do ants find the shortest route between two points, being almost blind and having very simple individual capacities? By pheromone deposition[3]: each ant moves at random unless they feel the pheromone – then they follow the marked path and leave new pheromone. The collective effect is a positive feedback. Two aspects are present: exploration and exploitation. Pheromone is simulated by a global knowledge structure, which is updated by "ants". This knowledge is used in the construction of solutions during the iterative optimization process.

Swarm intelligence $\supset$ ant algorithms $\supset$ ACO $\supset$ {AS, Ant-Q, Max-Min-AS, Ant Colony System, ...}

Do ACO należą m.in.: AS (prosty Ant System), jego ulepszenie Ant-Q, Max-Min-AS, ANTS (Approximate Non-deterministic Tree Search), oraz ACS (popularny Ant Colony System będący uproszczeniem Ant-Q).

Zastosowanie AS do TSP: w danym momencie czasu każde miasto ma pewną liczbę mrówek, które wybierają kolejne nieodwiedzone jeszcze miasto z prawdopodobieństwem będącym funkcją **odległości** do tego miasta (im mniejsza tym wyższe prawdopodobieństwo) i **ilości feromonu** pomiędzy miastami (im wyższa, tym wyższe prawdopodobieństwo). Istnieje kilka mechanizmów modyfikacji ilości feromonu [DCG99]: *ant-density* – w trakcie tworzenia rozwiązania stała ilość, *ant-quantity* – w trakcie tworzenia rozwiązania zmienna ilość w zależności od jakości dokładanych fragmentów, oraz najskuteczniejszy: *ant-cycle* – zmienna ilość, po skompletowaniu całego rozwiązania.

Przykład zastosowania AS w problemie nie przypominającym TSP – problem selekcji atrybutów [SB06]: np. miasta to atrybuty, a ścieżki to podzbiory atrybutów; w tym podejściu nie musimy odwiedzić wszystkich miast, a ich kolejność nie gra roli. Zatem w ogólności nie mówimy o odległości do jakiegoś miasta, a o atrakcyjności dołączenia jakiegoś fragmentu rozwiązania.

Some parameters in ACO:

- m: number of ants (agents) – population size,
- pheromone persistence $\rho < 1$ (evaporation is $1 - \rho$),
- α: pheromone importance,
- ...

[2] https://en.wikipedia.org/wiki/Emergence

[3] This is a form of **stigmergy**: https://en.wikipedia.org/wiki/Stigmergy

3.2 Algorytm roju cząstek (PSO)

Particle Swarm Optimization – PSO (1995). Dosyć podobne do AE. Grupa („populacja" albo „stado") rozwiązań („cząstek") przemieszcza się („leci") w przestrzeni rozwiązań dążąc do coraz lepszych obszarów.[4] Nie ma selekcji i krzyżowania. Każda cząstka pamięta najlepsze rozwiązanie jakie odkryła, oraz zna najlepsze rozwiązanie znalezione przez sąsiadów (albo całe stado). Ruch każdej cząstki w każdym kroku zależy od jej prędkości, której wektor jest zmieniany w kierunku (losowo ważonych) najlepszych pamiętanych rozwiązań[5]. PSO jest zwykle szybciej zbieżny niż AE.

3.3 Sztuczne systemy odpornościowe (AIS)

Artificial Immune Systems – AIS (1994). Inspiracja systemem odpornościowym kręgowców (dyskryminacja pomiędzy klasą „swoich" i resztą, mechanizmy adaptacji i zapamiętywania). Antygeny i przeciwciała mogą być modelowane np. jako lista atrybutów. Ocenianie podobieństwa antygenów i przeciwciał następuje za pomocą pewnej miary (Euklidesowa, Hamminga itp.). Stosowana jest selekcja negatywna i klonalna oraz mutacja[6]. AIS mają wiele zastosowań związanych z bezpieczeństwem systemów komputerowych (np. *anomaly detection* i *intrusion detection systems*, IDS) oraz z uczeniem maszynowym. Stosowane są również w optymalizacji i *często łączone z AE*: przeciwciała to rozwiązania a antygeny to ograniczenia, funkcja celu lub dobre/dopuszczalne rozwiązania. AIS pomagają tu utrzymać różnorodną populację rozwiązań.

3.4 Algorytmy pszczele (ABC)

Artificial Bee Colony (ABC) algorithms (2005). Analogie biologiczne: rodzaje pszczół to sposoby przeszukiwania, a kierunek do pyłku to gradient. Trzy rodzaje pszczół: *exploration* – pszczoły losowo próbkujące przestrzeń w celu zasilenia zbioru rozwiązań nowymi (zwiadowcy: *scout bee*) oraz *exploitation* – pszczoły przeszukujące sąsiedztwo deterministycznie (robotnice: *employed bee*) lub losowo (obserwatorki: *onlooker bee*). Algorytm jest stosunkowo prosty, ma tylko trzy parametry. Pozycje „pożywienia" (miejsca w przestrzeni rozwiązań, gdzie jest w danym momencie najlepsza wartość funkcji celu) są modyfikowane przez pszczoły-pracownice oraz pszczoły-obserwatorki. Kolonia (populacja) może być na przykład podzielona w takich proporcjach: 50% pracownic, 50% obserwatorek, jeden zwiadowca. Na

[4] https://en.wikipedia.org/wiki/Particle_swarm_optimization

[5] http://www.alife.pl/optymalizacja-rojem-czastek

[6] http://www.alife.pl/sztuczny_system_odpornosciowy

jedno miejsce „pożywienia” przypada jedna pracownica. Jeśli ilość „nektaru” (wartość funkcji celu) jest lepsza w danym miejscu, to pszczoła zapamiętuje to miejsce, zapominając poprzednie; por. artykuł [ABC09].

3.5 Algorytmy grawitacyjne (GSA) i elektrostatyczne (CSS)

Gravitational search algorithms – GSA (2009). Rozwiązania to punkty masy przemieszczające się w przestrzeni rozwiązań. Im rozwiązanie jest lepsze, tym ma większą masę. W GSA wykorzystuje się klasyczne zasady oddziaływania mas – każde rozwiązanie wpływa na prędkość i kierunek poruszania się pozostałych. Algorytmem zbliżonym do GSA jest CSS (Charged System Search, 2010): ruch rozwiązań jest determinowany przez ładunek cząstek (zakładamy, że wszystkie się przyciągają), przy czym rozwiązanie–cząstka o lepszej wartości funkcji celu przyciąga cząstki gorsze (*exploitation*), a przyciągnie w drugą stronę zachodzi z pewnym prawdopodobieństwem (*exploration*).

3.6 Świetliki, kukułki i szczętki (GSO, FA, CS, KH)

Proponowane są kolejne algorytmy inspirowane naturą (i ciągle są niewykorzystane jeszcze zwierzątka):

- algorytm robaczków świętojańskich (*Glowworm swarm optimization* – GSO, 2005)
- algorytm świetlikowy (*Firefly algorithm* – FA, 2008)
- algorytm kukułczy (*Cuckoo search* – CS, 2009)
- algorytm ławicy szczętek (*Krill herd algorithm* – KH, 2012)
- i inne[7]...

reprezentujące nurt *Swarm Intelligence*. Jednak większość z nich po usunięciu biologicznej interpretacji stosowanych mechanizmów staje się podobna, a w każdym algorytmie można znaleźć dwa kluczowe aspekty: eksploracji i eksploatacji.

[7] https://en.wikipedia.org/wiki/List_of_metaphor-based_metaheuristics

Przykładowe pytania

- Co to jest emergencja? Podaj przykłady naturalne (fizyka, biologia) i sztuczne (technika).
- Co to jest stygmergia? Podaj przykłady naturalne (biologia) i sztuczne (technika).

Rozdział 4

Niekonwencjonalne środowiska obliczeniowe

Obliczenia nie-tradycyjne, z których kilka pokrótce przedstawiono w kolejnych podrozdziałach, określa się czasem nazwą *unconventional computing*[1].

4.1 Obliczenia molekularne

(*molecular computing, DNA computation*)

Idea obliczeń molekularnych została przedstawiona już przez Richarda Feynmana. W przypadku obliczeń DNA nośnikami informacji są cząsteczki chemiczne DNA. Na nich dokonują się procesy obliczeniowe. Obecnie implementacje algorytmów na DNA są w fazie wczesnego rozwoju. Faktyczne eksperymenty laboratoryjne są czasochłonne (obliczenia trwają z przyczyn technicznych kilka dni, tygodni) i kosztowne.

Koncepcja obliczeń molekularnych wykorzystuje zjawisko samoskładania oligonukleotydów (cząsteczek DNA) w czasie hybrydyzacji. Cechą tego podejścia jest masowość obliczeń (cząsteczka lub ich grupa pełnią rolę procesora, liczba cząsteczek jest olbrzymia). Ponieważ cząsteczki reagują selektywnie, występuje duża równoległość obliczeń. Łatwo jest zrealizować pamięć asocjacyjną (kojarzenie cząsteczek na zasadzie klucz-zamek).

Niektóre operacje inżynierii genetycznej [KAEiOG99, str. 211 i 227]:

[1] https://en.wikipedia.org/wiki/Unconventional_computing

- powielanie PCR: powielanie łańcuchów DNA o znanych sekwencjach krańcowych (primerach). Początkowo w roztworze są fragmenty które chcemy powielić i dużo więcej primerów. PCR wykonuje się cyklicznie, 3 fazy (trwają kilka minut):
  - denaturacja: wysoka temperatura, dwuniciowe formy rozpadają się
  - hybrydyzacja: komplementarne fragmenty nici tworzą formy dwuniciowe. Primery przyłączają się
  - polimeryzacja: białko polimeraza „reperuje DNA" wydłużając primer przez dołączanie komplementarnych do wzorca nukleotydów. Liczba fragmentów DNA podwaja się.
- elektroforeza: pokazanie, jakiej długości fragmenty DNA są w mieszaninie. DNA migruje w polu elektrycznym z szybkością odwrotnie proporcjonalną do masy (i zależną od kształtu). Elektrody, żel, wybarwianie, pomiar przesunięcia DNA. Dokładność pomiaru – 1 nm (1 nukleotyd).

Zastosowania – optymalizacja (obliczanie funkcji boole'owskich, szukanie cyklu Eulera w grafie, ...), konstrukcja systemów eksperckich, pamięci asocjacyjnych, łamanie szyfrów...

Komputer:	**półprzewodnikowy**	**DNA**
Rozmiar bitu informacji [nm^3]	10^4	1
Operacji / Dżul	10^9 (teoret. do 10^{20})	10^{19}
Równoległość [procesorów]	10^3	10^{20}
Szybkość [operacji/s]	10^6	10^{-3}

Motywacja: tradycyjne komputery kiepsko radzą sobie z problemami NP-zupełnymi. W komputerach DNA liczba procesorów może wzrastać wykładniczo! (ale dla dużych problemów potrzeba by było np. kilograma DNA, co oznacza duże koszty).

4.2 Obliczenia kwantowe

Kwantowe przetwarzanie informacji (*quantum information processing*)[2].

Obecne komputery – ciągle zasada z XIX wieku (Charles Babbage, potem Alan Turing): jeden stan stabilny odpowiada jednej liczbie. Dotyczy to nawet obliczeń DNA. Tymczasem mechanika kwantowa[3] pozwala na zapamiętywanie informacji w bitach kwantowych (qubits, kubity). Qubit może przechowywać naraz 0 i 1 (przypomnij sobie podstawową

[2] https://en.wikipedia.org/wiki/Quantum_information_science

[3] https://www.youtube.com/watch?v=0dXNmbiGPS4

różnicę między mechaniką klasyczną i kwantową, oraz przestudiuj działanie kwantowego wykrywacza bomb[4]). Rejestr kwantowy np. 64 kubitów może „zapamiętać" naraz 2^{64} liczb (każda kombinacja 0 i 1). Komputer kwantowy może dokonywać obliczeń na tych wszystkich superpozycjach bitów równocześnie, jednak poznanie rezultatu (stanu) jest specyficzne: pomiar niszczy zawartość. Zatem można tylko raz zapytać o wynik. Równoległość – w klasycznych komputerach rośnie liniowo wraz ze wzrostem liczby procesorów, w kwantowych – wykładniczo (*quantum parallelism*). Zastosowania: potęgowanie, optymalizacja, uczenie maszynowe, kryptografia, bezpieczne kody (wiadomo, czy informacja była podsłuchana pomiędzy nadawcą a odbiorcą).

Popularny algorytm szyfrowania RSA opiera się na tym, że rozkład liczb na czynniki (faktoryzacja) na klasycznych komputerach jest czasochłonna (najlepszy algorytm ma złożoność ponadwielomianową). Tymczasem Peter Shor w 1994 r. wymyślił taki algorytm (na komputer kwantowy) o złożoności kwadratowej[5], co oznacza, że kiedy tylko technika kwantowa pozwoli na zbudowanie odpowiedniego komputera, RSA będzie łatwo złamać! Zatem od momentu odkrycia tego algorytmu, w pewnych zastosowaniach (wojsko, rząd) RSA **przestał być uznawany** za algorytm bezpieczny dla informacji, które muszą pozostać tajne przez wiele lat[6]. Aby utrudnić tego typu ataki, w 2023 r. komunikator Signal wprowadził protokół PQXDH (PQ to skrót od *Post-Quantum*), a Google wprowadziło do przeglądarki Chrome protokół X25519Kyber768. W tym samym celu Apple wprowadziło rok później protokół PQ3 do swojego komunikatora iMessage.

Z kolei Lov Grover wymyślił na komputer kwantowy algorytm *unstructured search* (przeszukiwanie zbioru elementów bez dodatkowej informacji o jakichkolwiek zależnościach między nimi) kwadratowo szybszy od klasycznych (liniowych), lecz trzeba wiedzieć, kiedy go zatrzymać (rozwiązanie może zacząć się pogarszać). Kwadratowy przyrost prędkości to jednak za mało, by poradzić sobie z wykładniczą złożnością problemów NP.

Styczeń 2000 – największe kwantowe komputery: 100 operacji logicznych na dwóch kubitach, 10 operacji na 7 kubitach. 2002 – powstaje firma produkująca moduł detekcji pojedynczych fotonów (zawsze wiadomo czy ktoś podsłuchał przekaz[7], więc klucza nie można ukraść) za \$88k. 2011 – chip 6x6mm, 9 bramek kwantowych, 4 kubity. Maj 2011 – powstaje D-Wave, komercyjny 128-kubitowy komputer do kwantowego wyżarzania,[8] i choć zakres jego stosowalności jest ograniczony,[9] to wydajność już teraz dla wybranych problemów staje się

[4] http://en.wikipedia.org/wiki/Elitzur%E2%80%93Vaidman_bomb_tester
[5] https://en.wikipedia.org/wiki/Shor%27s_algorithm
[6] https://www.youtube.com/watch?v=6H_9l9N3IXU
[7] https://www.youtube.com/watch?v=fLJ9mvTS68Y&t=221
[8] http://en.wikipedia.org/wiki/Quantum_annealing
[9] https://www.youtube.com/watch?v=Yy93LMGQbpo

przydatna[10]. Od 2015r. największe firmy informatyczne prowadzą własne badania nad obliczeniami kwantowymi, testując różne realizacje kubitów (por. podział na kubity 'fizyczne' i 'logiczne': logiczne powstają z wielu niedoskonałych fizycznych). 2019, 2020 – każda duża firma chce pierwsza pochwalić się osiągnięciem *quantum supremacy*.

W organizmach biologicznych efekty kwantowe są wykorzystywane w wielu różnych mechanizmach i procesach. Ich badaniem zajmuje się *biologia kwantowa*[11].

Efekty kwantowe wydają się (i są przedstawiane zwykle jako) bardzo nieintuicyjne i sprzeczne z naszym rozumieniem makro-świata – tymczasem nawet w makro-świecie występują zjawiska przypominające efekty kwantowe, np. cząsteczka będąca w ciągłej interakcji ze „swoją" falą[12].

4.3 Obliczenia na błonach/membranach

Membrane computing: model obliczeniowy[13] (zaproponowany w 1998 roku przez Gheorghe Păun) charakteryzujący się równoległością i rozproszeniem. Błony nie mogą się przecinać i są zagnieżdżone (rekurencyjnie) w sobie. W błonach umieszczone są obiekty, które mogą się zmieniać, przemieszczać i rozpuszczać/dzielić błony w których występują. Obiekty zmieniają się w czasie dzięki uporządkowanemu według priorytetów zbiorowi reguł. W ten sposób powstaje specyficzny komputer („P system"[14]), który rozpoczyna obliczenia od określonej liczby błon, obiektów i reguł. Można więc znaleźć tu pewne analogie do CFS (rozdział 2.8), Artificial Chemistry (rozdział 5.10.4) i L-systemów (rozdział 5.6).

Membrane computing jest jednym z paradygmatów obliczeniowych w ramach *natural computing*. Natura bowiem dokonuje „obliczeń" nie tylko na poziomie sieci neuronowych czy genetyki/ewolucji, ale również na poziomie komórkowym (ciągły przepływ informacji, materii i energii). W naturze występują różne błony – w tym selektywnie przepuszczające tylko określone substancje, jednokierunkowe i dwukierunkowe, itp.

Konkretny P system wymaga ustalenia składni obiektów (często traktowanych jako ciągi znaków), składni reguł i możliwości reguł (czy dotyczą pojedynczych obiektów, czy ich grup, i jakie mają działanie). W ten sposób powstaje kilka klas P systemów. P systemy są równoważne maszynom Turinga, a dzięki równoległemu podziałowi błon (prowadzącemu do wykładniczego wzrostu ich liczby) mogą rozwiązywać problemy NP-zupełne (np. SAT)

[10] http://googleresearch.blogspot.ca/2015/12/when-can-quantum-annealing-win.html
[11] https://www.youtube.com/watch?v=Z_KI9sjyVIQ
[12] https://www.youtube.com/watch?v=WIyTZDHuarQ, https://www.youtube.com/watch?v=RlXdsyctD50
[13] https://en.wikipedia.org/wiki/Membrane_computing
[14] https://en.wikipedia.org/wiki/P_system

w wielomianowym czasie. Niektóre klasy P systemów są możliwe do realizacji w medium biologicznym.

P systemy są zdefiniowane bardzo formalnie i mają wiele wspólnego z gramatykami i językami formalnymi. Przykład [Pău00] P systemu, który rozstrzyga, czy liczba n jest wielokrotnością k (czy k dzieli n). Jeśli tak, w błonie oznaczonej jako wyjściowa zostanie jeden obiekt; jeśli nie – dwa. Po prawej pokazano strukturę Π_2.

$$\Pi_2 = (V, \mu, \lambda, a^n c^k d, a, (R_1, \varnothing), (R_2, \rho_2), (\varnothing, \varnothing), 3),$$

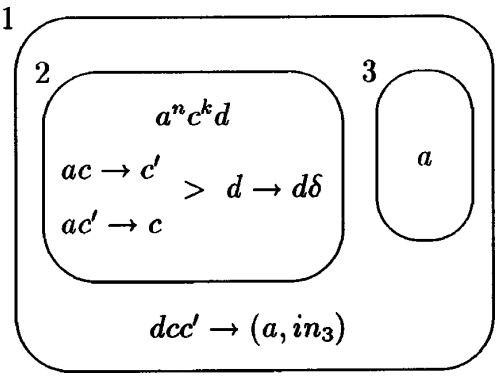
$$V = \{a, c, c', d\},$$

$$\mu = [_1 [_2]_2 [_3]_3]_1,$$

$$R_1 = \{dcc' \to (a, in_3)\},$$

$$R_2 = \{r_1 : ac \to c', r_2 : ac' \to c, r_3 : d \to d\delta\},$$

$$\rho_2 = \{r_1 > r_3, r_2 > r_3\}.$$



In membrane 2 we subtract k from n, repeatedly (by the rules $ac \to c'$, $ac' \to c$: at each step, k copies of a disappear, while c is reproduced, primed or not primed, alternating the priming from one step to another).

The rules $ac \to c'$, $ac' \to c$ have priority over the rule $d \to d\delta$; therefore we can dissolve membrane 2 only after exhausting the n occurrences of a. If n is a multiple of k—and only in this case—then we never have both occurrences of c and of c' simultaneously present in membrane 2 (or in membrane 1, after dissolving membrane 2). Therefore, the rule $dcc' \to (a, in_3)$ is used in membrane 1 if and only if n is not a multiple of k.

Note that this rule can be used at most once, because we have only one occurrence of d and that the computation stops after using the rule $dcc' \to (a, in_3)$.

In conclusion, the computation always stops and the output membrane contains two objects if and only if n is not a multiple of k (in the opposite case, we have here only one object).

Rozdział 5

Sztuczne życie

Nagranie do tego rozdziału: https://youtu.be/YiMWJpBObrw

5.1 Definicja, metodyka, cele

Discussion: "What is life?"

Artificial Life[1][2] (AL, ALife) [Sip95]:

- is an interdisciplinary research enterprise aimed at understanding ***life-as-it-is*** (***life-as-we-know-it***) on Earth and ***life-as-it-could-be*** (larger domain of "bio-logic" of possible life)
- synthesizes life-like phenomena (embodiment and physical constraints for the self-organization) in chemical (wetware), electronic (hardware) [AK09], software [KA09] (cf. movie "Her", 2013; movie "Transcendence", 2014), and other artificial media
- is devoted to understanding life by attempting to abstract the fundamental dynamical principles underlying biological phenomena, and recreating these dynamics in other physical media, such as computers, making them accessible to new kinds of experimental manipulation and testing [Lan97]
- redefines the concepts of artificial and natural, blurring the borders between traditional disciplines and providing new insights into the origin and principles of life.

Complementary research methods (Fig. 5.1):

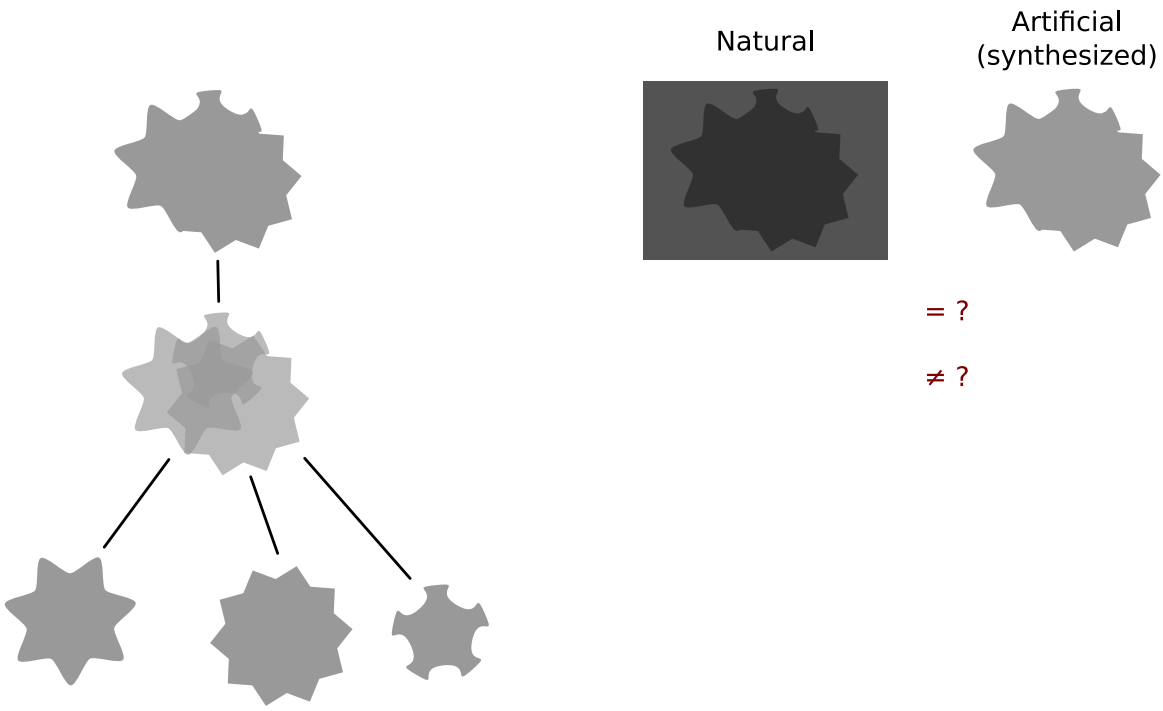
- most research (biology and AI, too) is essentially *analytic*, breaking down complex phenomena into their basic components (which is not always possible),
- ALife is *synthetic*, attempting to construct phenomena from their elemental units – this is inevitable when trying to understand emergent phenomena.

Main goals of ALife:

- Increasing our understanding of nature by studying existing biological phenomena. Examples are provided in Sect. 5.15.
- Enhancing our insight into applicable artificial models (this requires studying complex systems) in order to improve their performance. Examples are software development

[1] http://www.alife.pl/czym-jest-sztuczne-zycie

[2] http://www.alife.pl/zycie-w-komputerze-symulacja-czy-rzeczywistosc



Rysunek 5.1: Left: analytic/synthetic research methods. Right: synthesis as a way of the inference about how complex systems are built and how they work. Consider two examples: neurons, brain, thoughts and companies, market, stock prices. Another use case: consider the vertical axis to be time (top=*present*) – sometimes time obscures the past and we cannot know what happened (e.g. how life emerged, how species evolved, how atypical supernovae were created [JMK20]), so to learn what might have happened we have to build models of changes in time, simulate them and compare the outcomes to the present state.

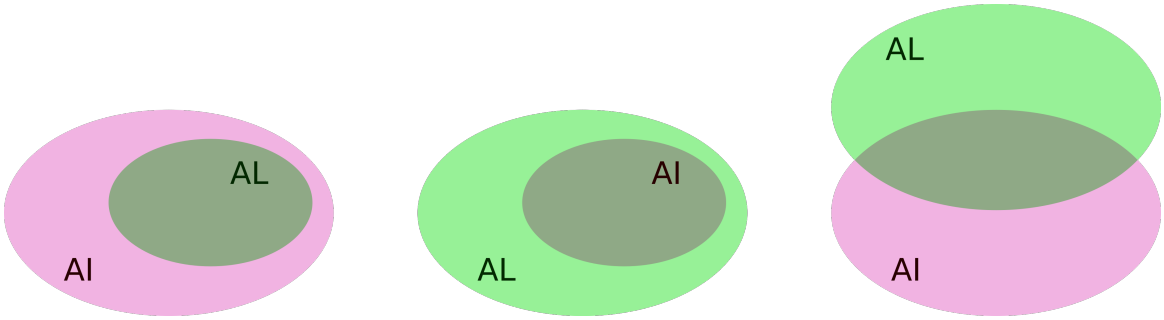
through evolution (Genetic Programming, GP) or devising biologically-inspired optimization algorithms. This is where we will focus during this course.

Sample questions for ALife:

- Can a machine reproduce?[3] (John von Neumann, early 1950's – CA, Sect. 5.8)
- Can software be evolved? (John Koza – GP, Sect. 2.6)
- How are sophisticated robots built to function in a human environment?
- Can an ecological system be created within a computer?
- How do flocks of birds fly?

[3]The ability to repair is a part of reproduction, and repair may concern the https://en.wikipedia.org/wiki/Trolley_problem.

5.2 Sztuczne życie a sztuczna inteligencja



Rysunek 5.2: Three possible relationships between AL and AI. Left: AI-centric/traditional. Middle: common sense. Right: right.

A remark on notation: sometimes a difference between "Artificial Life" and "artificial life" is emphasized (same as with "Artificial Intelligence" and other domains/disciplines/fields) and such a distinction is useful, but capitalization rules say not to do it[4].

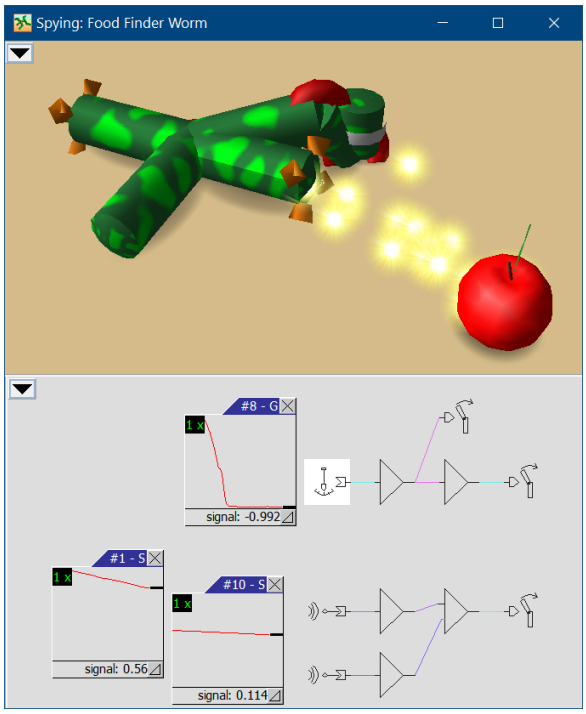
Nowy paradygmat badania inteligencji mówi, że nie jest ona abstrakcyjnym procesem, lecz wymaga osadzenia (*situatedness*) w środowisku i wcielenia (*embodiment*). Pozwala to agentowi wpływać na dane, które otrzymuje; powstaje oddziaływanie agent-środowisko, wymagana jest koordynacja – *sensory-motor coordination*. Niekiedy zachodzi wymóg, by agent reagował różnie na takie same bodźce (w robotyce jest to *perceptual aliasing problem*[5]: należy zachować się w różny sposób w pozornie takich samych sytuacjach – sytuacjach, które są postrzegane jako identyczne, por. Rys. 5.3). Zwykle rozwiązaniem jest wtedy zdobycie innych informacji lub przekształcenie/przetworzenie już posiadanych. Dzięki „wcieleniu" agent może tego dokonać (*active perception*), np. przez zmianę swojego sposobu postrzegania obiektów, dotarcie do dodatkowych, dyskryminujących decyzję danych, lub nawet obserwację swojego zachowania i analizę swojej interakcji ze środowiskiem [NP99]. Przykład 1 – idealnie symetryczne boisko piłkarskie i identyczni zawodnicy: po czym odróżnić własną bramkę od bramki przeciwnika? Przykład 2 – mała bakteria i chemotaksja. Przykład 3 – wypadki samochodów autonomicznych. Przykład 4 – brak widzenia.

5.3 Czym życie jest, a czym nie jest: definicje życia

From [Ada98], for an extended discussion see [Life10]:

[4] https://english.stackexchange.com/questions/6246/capitalize-fields-of-study

[5] Percepcja – odbieranie zmysłami/czujnikami informacji z otoczenia (i z samego agenta). Procesy poznawcze (kognitywne) – odbieranie, przetwarzanie i przechowywanie informacji w celu kierowania zachowaniem agenta. Więcej w rozdziale 5.12.7.



Rysunek 5.3: A sensory input space (here, 3D) is all a creature can sense. Contrary to what we, its observers, can sense...

- **Physiological Definition**: Focuses on physiological functions such as breathing, moving, digesting, etc, to construct a list of requirements that will distinguish living from non-living. *Outdated*.
- **Metabolic Definition**: Centers on the exchange of materials between the organism and its surroundings as the only requirement for it to be alive. *Too narrow?* or *Too general?*
- **Biochemical Definition**: Classifies living systems by their capability to store hereditary information in nuclear acid molecules. *Focuses on DNA/RNA. Too narrow.*
- **Genetic Definition**: Focuses on the process of *evolution* as the central defining characteristic of living systems, without regard to *how* the information is coded (i.e., independently of substrate).
- **Thermodynamic Definition**: Describes systems in terms of their ability to maintain low levels of *entropy* (i.e., disorder) despite a noisy environment [Sch44]. *Too general?*
- **Physics-based Definition**: Life is a property of an *ensemble* of units that *share information* coded in a physical substrate and which, in the presence of noise, manages to keep its entropy significantly lower than the maximal entropy of the ensemble, on

timescales exceeding the "natural" timescale of *decay* of the (information-bearing) substrate by many orders of magnitude.

According to one theory, life is a perfect dissipator of energy[6]. The universe tends toward disorder, decay, and equilibrium (entropy grows). But life maintains low entropy due to a non-random, specific structure; the complexity of life on Earth increased during evolution, as if entropy decreased. However, Earth is not a closed system – Sun provides energy. Life decreases its own entropy by increasing the entropy of its surroundings (it absorbs order and ejects disorder), and so energy gradients (energy flow) facilitate the emergence of life[7]. The original source of the lowest entropy was the Big Bang, and life (as a local phenomenon of order) arises naturally as the energy is redistributed into the most random possible state.

Why these are alive?

- **Viruses:** which definitions support it, and which are against it?
- **Sand Dunes**: as they get blown through the desert, end up growing, and splitting off smaller dunes. Many think of this as a form of self-replication.
- **An Organized Religion**: a meme of sorts, with no physical representation. Certainly an 'idea' which can spread like wildfire as a parasite through a host population of 'people'. It can also be argued to be the people who are part of the religion, and then it would have a genetic basis.
- **Chain Letter**: This is another meme, only this one *does* certainly have a physical representation. And I know many people who wish Chain Letters would just die already...
- **Prions**: These proteins will infect a living organism and bend its proteins around to become more prions (e.g., Mad Cow and Creutzfeld-Jacob disease). It requires a very select environment to work (so do we...)
- A **Robot** which moves around collecting resources from the environment and returns them to a robotic 'hive' to build more robots just like it.

Epistemology (epistemologia – relacje między poznawaniem, poznaniem a rzeczywistością). Co potocznie uważamy za żywe? Jak rozpoznać życie? Jakie cechy występują u istot żywych? [Dom99]

[6] https://www.youtube.com/watch?v=GcfLZSL7YGw

[7] https://www.sciencefocus.com/science/the-origin-of-life-a-new-theory-suggests-physics-holds-the-answer/

Wedle Farmer i Belin [FB90]: Life is a pattern in spacetime, self-reproduction, storage of a self-representation, a metabolism, functional interaction with the environment, interdependence of parts, stability under perturbations, the ability to evolve.

Z kolei Mark Bedau jako cechę (wyznacznik) życia zaproponował ciągłe dopasowywanie się, „elastyczną adaptację" (*supple adaptation*) [Bed96]. Wydawało się bowiem, że do określenia czy obiekty są żywe nie wystarczą ich cechy, ale że trzeba na nie także spojrzeć z perspektywy systemu, jaki tworzą – jego zachowania i dynamiki. Aby stwierdzić czy zachodzi ciągła adaptacja, porównuje się ewolucję organizmów w badanym modelu i dodatkowo stworzony „*null model*", w którym mutacje zachodzą równie często jak w modelu badanym. Jednak w *null modelu* genotyp nie ma wpływu na przeżywalność. Bada się *traits* (charakterystyczne właściwości, cechy) organizmów: jeśli w badanym modelu występują one statystycznie znacząco częściej niż w *null modelu*, oznacza to, że badany system wykazuje ewolucję adaptacyjną, co jest wedle Bedau wyznacznikiem życia.

W początkowych modelach sztucznego życia albo nie zauważamy takich cech symulowanych organizmów, albo pojawiają się tak długo, dopóki nie zostanie „rozwiązane" „zadanie" stawiane przez środowisko (potem ewolucja ustaje), a zatem statystycznie nie trwają dłużej, niż w *null modelu*. Z drugiej strony stosując cechę elastycznej adaptacji jako definicję życia dochodzimy do wniosku, że pojedynczy ludzie są mniej żywi, niż biosfera jako całość, a także niż niektóre systemy chemiczne czy ekonomiczne [Dom99].

Cechy (wyznaczniki) zaproponowane przez Farmer i Belin oraz cecha elastycznej adaptacji były krytykowane; Paul Domjan wymyślił „Romance Novel System" – system, w którym pisarze romansów są zarazem czytelnikami, wszystkie powieści są publikowane, a autorzy mogą je czytać i swobodnie zapożyczać od siebie pomysły [Dom99]. Domjan uważa, że jego system w pewnym sensie i przy odpowiedniej interpretacji ma wszystkie zaproponowane powyżej cechy życia, a ani ten system, ani jego składniki nie zostałyby przecież zdroworozsądkowo lub intuicyjnie uznane za żywe.

Więcej o roli *null modelu*: jeśli bada się model ewolucyjny pod kątem adaptacji (np. szukając częstotliwości występowania genów, cech itp.), warto stworzyć odpowiadający mu *null model* („*neutral shadow*") [RB99]. Jest on „cieniem" modelu badanego: parametry ewolucyjne, zasady egzystencji, rozmnażania itd. są identyczne, jednak genotypy nie mają znaczenia adaptacyjnego. Selekcja w *null model*'u jest czysto przypadkowa, podczas gdy w modelu badanym zwykle powoduje usuwanie słabych genotypów. *Null model* pozwala zminimalizować wpływ zjawisk ewolucyjnych takich jak **przypadek** (*chance*) i **prawidłowość** (*necessity*)[8]. Przy porównaniu modelu badanego z jego „cieniem" odznacza się wyraźnie efekt

[8]"Everything existing in the universe is the fruit of chance and necessity" is attributed to Democritus, an ancient Greek pre-Socratic philosopher (400 BC). "Chance and Necessity: Essay on the Natural Philosophy of Modern Biology" is a 1970 book

adaptacji, a pozostałe efekty ewolucyjne (potencjalnie nakładające się na proces adaptacji i utrudniające analizę) mogą być odfiltrowane.

5.4 Zakres badań i zastosowania

- Self-organization
- Chemical origins of life, Autocatalytic systems, Prebiotic evolution, RNA systems, Evolutionary/artificial chemistry
- Fitness landscapes
- Natural selection
- Artificial evolution
- Ecosystem evolution
- Multicellular development
- Natural and artificial morphogenesis
- Learning and development
- Bio-morphic and neuro-morphic engineering
- Artificial / Virtual worlds
- Simulation tools
- Artificial organisms
- Synthetic actors
- Artificial (virtual and robotic) humanoids
- Intelligent autonomous robots
- Evolutionary Robotics / Design
- Life detectors
- Self-repairing hardware

by a French biochemist, Nobel Prize winner (1965, for discoveries concerning genetic control of enzyme and virus synthesis) Jacques Monod. He interpreted the processes of evolution to show that life is only the result of natural processes by "pure chance" – https://en.wikipedia.org/wiki/Chance_and_Necessity. More discussion in [Küp91].

- Evolvable hardware (EHW)
- Emergent collective behaviors
- Swarm intelligence
- Evolution of social behaviors
- Evolution of communication
- Epistemology
- Artificial Life in Art. Evolutionary art (example); evolutionary music; creative evolutionary design; conceptual evolutionary design; strategy evolution; collaborative evolutionary systems; interactive evolutionary systems; evolutionary sculpture; evolutionary architecture.

Prace nad sztucznym życiem nie ograniczają się do systemów ewoluujących. Czasem celem jest jedynie symulacja życia zapewniająca realizm zachowań. Powstały w tym zakresie realistyczne makro-symulacje – jednymi z pierwszych były Artificial Fishes [TTG94] oraz projekty Humanoid i Humanoid-2 [Tha+95]. Do takich celów wykorzystywane są techniki wirtualnej rzeczywistości (VR), interaktywnych „aktorów" (*avatars*), czy też systemy Lindenmayera (L-systems, rozdział 5.6).

Wybrane zastosowania AL [KC06]: robotyka, projektowanie i konstruowanie trójwymiarowych obiektów; roboty adaptujące się do zadań, środowisk i swoich awarii; animacje komputerowe (filmy, reklamy, gry, symulacje); medycyna, terapia i fizykoterapia; badanie zachowań grupowych, społecznych, tłumów, ławic (ryby, ptaki), ekosystemów (lasy, bakterie, wirusy); badanie zachowania złożonych systemów adaptacyjnych[9] (biologia, ekonomia, modele rynku i konsumentów) oraz procesów biologicznych (np. budowanie sieci pajęczej, budowa i zasada działania oka); badania rozproszonej wiedzy i informacji, inteligencji, zdolności komunikacji i ewolucji języków; tworzenie odpornych algorytmów i protokołów dla zmiennych w czasie-/mobilnych sieci komputerowych; układy scalone adaptujące się do obliczeń.

Przykładowe pytania

Czym jest aliasing percepcyjny? Co jest przyczyną i jak można mu zapobiegać?

[9] https://en.wikipedia.org/wiki/Complex_adaptive_system

5.5 Evolution

5.5.1 Fundamental mechanisms

If you're looking for a paradigm of adaptability, look no further than biology. Living things, based on a set of simple underlying chemical principles, have shown remarkable flexibility and adaptability throughout billions of years of changing environments. Implementing biological concepts creates software that evolves solutions. In some cases, a biological algorithm might find solutions that its programmer never envisioned – and that concept allows software to go beyond the human creator's vision.

In the mid-nineteenth century, Charles Darwin reasoned that immutable species would become increasingly incompatible with their restless environment. The resemblance of offspring to their parents suggested to him that traits pass from one generation to the next; he also noticed slight differences between siblings, which provide a species with a pool of unique individuals who compete for food and mates.

From those observations, Darwin concluded that, as the environment changed, organisms best-suited to the new conditions would bear offspring reflecting their successful traits. Darwin named this process "natural selection", and he believed that it was the central mechanism by which species evolved.

Modern science recognizes evolution as the mechanism that creates biological organization. While evolutionary theory has been refined in the century since Charles Darwin's death, the core concepts remain intact. We can see evolution operating today and in the fossil record of past species; we can see how organisms change to survive in an ever-changing world.

A tiny English moth provides a classic example of natural selection in action. Before the Industrial Revolution, light-colored pepper moths blended with the white lichen on trees, hiding them from predaceous birds; dark-colored moths contrasted with the lichen and often became avian meals. But when smoke from England's new coal-fired factories killed the lichens and coated the trees with soot, the light-colored moths became visible targets for birds, while dark-colored moths blended into the new environment. Within a few years the pepper moth population was nearly all dark-colored, having adapted to its new environment through natural selection.[10]

While the survival of individuals determines the characteristics of the next generation, it is the reproductive success of a population as a whole that determines the evolution of a species. Natural selection is limited by the characteristics of a population; while it is often called survival of the fittest, natural selection really operates through the survival of the

[10] https://en.wikipedia.org/wiki/Peppered_moth_evolution

best available organisms. An organism's "fitness" is relative to a changing ecosystem, other species, and other members of its population. What is "best" today (light-colored moths on lichen) may not be "best" tomorrow (dark-colored moths on soot).

Darwin didn't know how characteristics were passed from parent to offspring; he simply saw it happening. The missing element was beyond the science of his time, and nearly a century passed before someone identified the mysterious agents behind evolution. In 1951, biologists Francis Crick and James Watson first described the deoxyribonucleic acid (DNA). DNA encodes the chemical recipes for life's proteins and enzymes, and it packs an amazing amount of information into an incredibly tiny space; if the DNA in a single human cell were straightened out, it would be nearly two feet long.

Biologists are still exploring this most fundamental piece of life's mystery. Each tightly coiled strand of DNA contains genes that define individual parts of an organism's blueprint. Human DNA includes more than 200,000 genes that are responsible for controlling everything from eye color to the potential for developing certain illnesses.

Offspring inherit characteristics through genes received from a parent. Simple organisms such as fungi and bacteria reproduce asexually by duplicating themselves. A single-celled amoeba, for instance, creates offspring by splitting into two new organisms that contain identical DNA. Thus, asexual reproduction produces new organisms that differ little from each other or their progenitor.

Most complex organisms reproduce sexually by combining genes from two parents in their offspring. By mixing and matching DNA from two organisms, sexual reproduction increases the variation within a species. The possibilities are almost endless; for example, a human couple can produce more than seven trillion different blueprints for a person.

The collective genetic information in a population constitutes a gene pool. Large gene pools are healthier than small ones by allowing a greater number of genetic combinations. Greater variability means that a larger gene pool is more adaptable and less prone to recessive genetic disorders. A small gene pool leads to inbreeding, increasing the chance that recessive genes will manifest themselves in the offspring of closely-related organisms.

Natural selection changes the frequencies of genes in a population, but it doesn't produce new genes. The first life forms began as self-replicating chemicals that bound to each other in mutual cooperation. The first complete organisms resembled an amoeba – and an amoeba clearly does not contain the genes required to evolve it into a human being. New characteristics must somehow arise; otherwise, the simple original life forms would never have evolved into the millions of species on Earth today.

A mutation is a random change in an organism's genes. It is highly unlikely that a random ge-

netic change will improve a complex organism that is well-adapted to its environment; most manifest mutations disappear from the population through natural selection. Fortunately, the vast majority of mutations have practical influence, while some "random" characteristics provide new material for evolutionary development. Studies of human DNA have found long sequences of "junk genes" that serve no explicit purpose; mutations in junk genes are likely to be meaningless. And sometimes, cells can repair damaged DNA, eliminating many mutations before they are passed along to new cells or offspring.

Natural selection mixes and sifts gene pools, acting on variations produced by reproduction and mutation. Sometimes a gene pool evolves in a straight line, carrying a species from one form to another, as in the earlier example of the peppered moth. In other cases, forces act to divide a gene pool, and natural selection works on the now-separate populations to produce new species.

If a species encounters several open niches, it may quickly diversify in a process known as adaptive radiation[11] (pol. *radiacja adaptacyjna*). When the dinosaurs vanished 65 million years ago, they left unoccupied niches that were exploited by mammals. Through adaptive radiation, a few shrew-like species blossomed into thousands of types ranging from bears to people and whales.

In the 1980s, biologists Niles Eldredge and Stephen Jay Gould introduced a substantial modification in evolutionary theory. They postulated that evolution was not a steady process, moving from species to species continuously. Their review of the fossil record suggested that species remain static until environmental factors force rapid evolution into new forms. Known as punctuated equilibrium, this new idea has been hotly debated. Recent computer simulations suggest that punctuated equilibrium (pol. *równowaga przestankowa*) is present in many complex systems – not just in evolution, but also in weather, economy, etc.

Przykładowe pytania

- Co oznacza termin *traits*?
- Jakie są konsekwencje małej (w porównaniu do dużej) puli genów?
- Co oznacza termin *radiacja adaptacyjna*?

Mechanisms needed for evolution to occur:

- transmission (reproduction and inheritance)
- variation

[11] https://en.wikipedia.org/wiki/Adaptive_radiation

- selection

Przykładowe pytania o ewolucję, które można badać używając technik sztucznego życia:

- czy ewolucja jest powtarzalna?
- czy przebiega liniowo, czy skokowo? (por. saltacjonizm[12])
- czy przy niezmiennych warunkach zewnętrznych osiąga stabilizację?
- czy komplikuje, czy upraszcza organizmy?
- czy organizmy późniejsze są lepsze od wcześniejszych?
- czy powstają organizmy odporne na zmiany warunków zewnętrznych?

Wydaje się oczywiste, że dobór naturalny premiuje osobniki lepiej przystosowane. Okazuje się, że może to się odbywać dwojako: dobór może być dośrodkowy-stabilizujący (usuwa osobniki skrajne, gorzej przystosowane) i odśrodkowy-różnicujący (sprzyja skrajnym, usuwa średnie – działa z reguły w przypadku zmiany środowiska i może powodować podział na rasy i nowe gatunki). Niestety nie jest łatwo stwierdzić, kiedy jeden sposób działania przechodzi w drugi i czym to jest spowodowane.

Bardzo istotna rola selekcji grupowej: przykład ewolucji umiaru – prostego analogu bezinteresowności i altruizmu. Osobniki w symulacji zawsze podążają do najbliższego pożywienia i mają „gen żarłoczności", który podlega ewolucji (zmienia się podczas krzyżowania i mutacji oraz jest dziedziczony). Gen decyduje o tym, jak wiele z napotkanego pożywienia organizm pochłania. Osobniki rodzą dzieci (oddając im połowę swojej energii) po przekroczeniu stałego dla wszystkich osobników poziomu energii, tak samo jak w `reproduction.expdef`.
▷ `frams/frams_evol_of_restraint.mp4`
Analiza i dyskusja: dwa przypadki miejsca narodzin potomków – w losowym miejscu świata albo tuż przy rodzicu. Co się stanie po dłuższym okresie symulacji w tych przypadkach?

5.5.2 Teorie: ewolucja, uczenie, symbioza

- Lamarckian evolution (pol. *lamarkizm*, adaptation by the intention of individuals to survive), Darwinian (pol. *darwinizm*, adaptation by natural selection) and the Baldwin effect[13]

[12] https://en.wikipedia.org/wiki/Saltation_(biology)
[13] https://en.wikipedia.org/wiki/Baldwin_effect

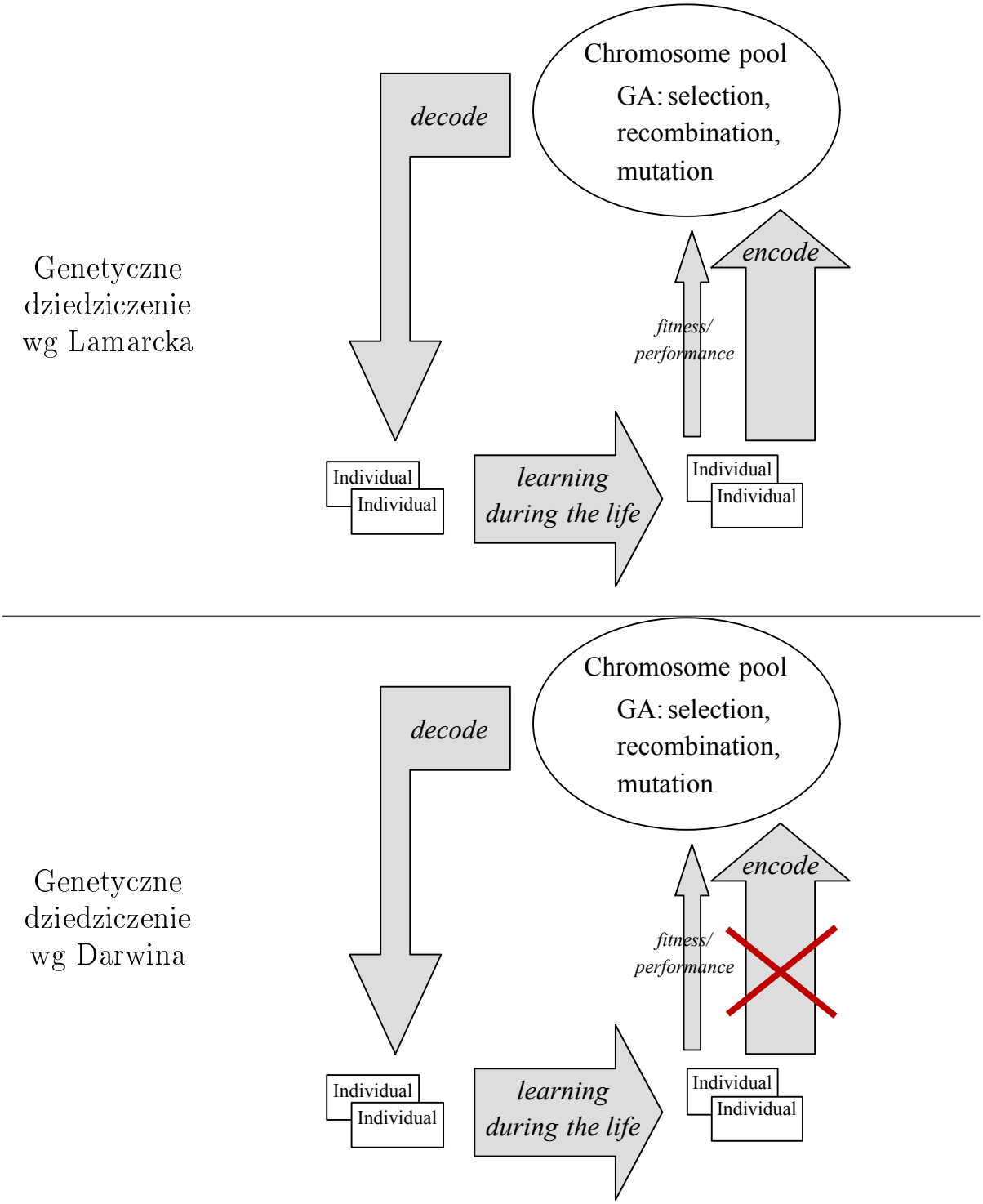
- gradualism (pol. *gradualizm*), saltationism (pol. *saltacjonizm*), punctuated equilibria[14] (pol. *punktualizm*) and many other theories
- Red Queen effect; instability of two-population coevolution – Sect. 2.12.2
- competitive arms races and, on the contrary, mediocre stable-states (MSS)

Lamarck/Baldwin: analogia do hybrydowych algorytmów optymalizacji (EA + Local Search). Uczenie, adaptacja w czasie życia, LS – zmieniają krajobraz przystosowania z punktu widzenia algorytmu nadrzędnego (ocena osobnika, ewolucja, EA).

Sam Darwin akceptował możliwość dziedziczenia nabytych cech, natomiast wykluczał to Weismann, którego poglądy wpłynęły znacząco[15] na teorię Neo-Darwinizmu (obecnie „Teoria Darwina" głosi brak bezpośredniego dziedziczenia cech nabytych podczas życia[16]; zresztą w wyniku postępu nauk genetycznych[17] kwestie te stały się oczywiste). W ewolucji Darwinowskiej występuje efekt Baldwina odzwierciedlający fakt, że istotniejszy wpływ na dopasowanie organizmu ma przebieg jego życia, niż cechy opisane jego genami. Jednak w pewnych warunkach (np. kiedy wszystkie osobniki „nabędą" już zdolność uczenia się w trakcie życia), szczególnie istotna staje się zawartość genotypu, zatem występuje wtedy silniejsza tendencja do umieszczania w genotypie korzystnych cech (a przynajmniej tendencja usuwania cech niekorzystnych) – dotyczy to oczywiście tych cech, dla których jest miejsce w genotypie.

Rozważ np. zgrubienia skóry na stopach: cecha nabywana czy dziedziczona? [dyskusja]. Zaprojektuj eksperyment, który bez odwoływania się do współczesnej wiedzy genetycznej i współczesnych technik laboratoryjnych odpowiedziałby na pytanie, która z teorii (Lamarck/Darwin) jest słuszna.

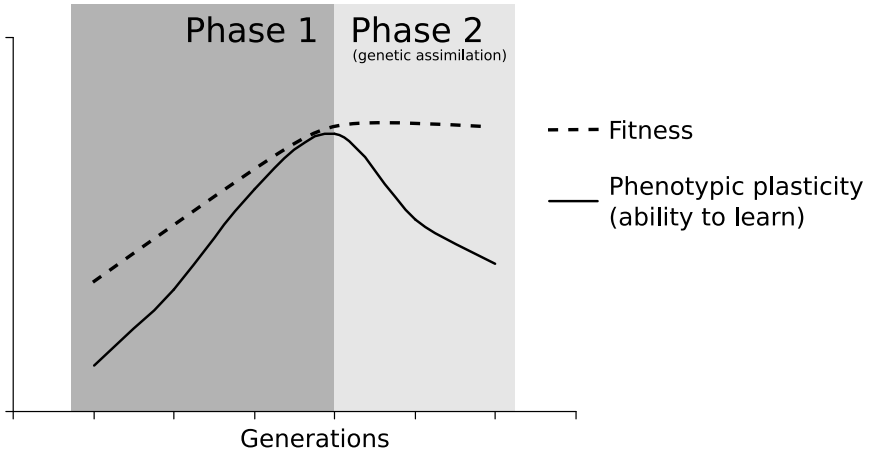
[14] https://en.wikipedia.org/wiki/Punctuated_equilibrium
[15] https://en.wikipedia.org/wiki/The_eclipse_of_Darwinism
[16] Zobacz jednak http://en.wikipedia.org/wiki/Epigenetics.
[17] Por. film „Gattaca"



Rysunek 5.4: Uproszczona koncepcja genetycznego dziedziczenia według Lamarcka i Darwina. Dyskusja: które z tych podejść jest skuteczniejsze w optymalizacji (ew. w jakich warunkach jest skuteczniejsze)? Czy (ew. dla jakich zadań optymalizacji?) warto również zaimplementować asymilację genetyczną Baldwina?

Oddziaływanie pomiędzy uczeniem i ewolucją [SA04]: uczenie się podczas życia osobnika może mieć wpływ na przebieg ewolucji nawet bez mechanizmu lamarckowskiego. Efekt Baldwina zwraca uwagę na koszty i zyski uczenia się. Można wyróżnić dwie fazy: w pierwszej uczenie się pozwala osobnikom na zmianę swoich fenotypów. Jeśli nabyte cechy są korzystne, zwiększają przystosowanie osobnika i będą zwielokrotnione w przyszłych pokoleniach. W drugiej fazie, o ile warunki środowiskowe są wystarczająco stabilne, ewolucja odkrywa cechy wrodzone, które zastępują cechy nabyte, by wyeliminować koszty uczenia się (ta faza to *genetic assimilation*). Zatem uczenie się może przyspieszyć ewolucję i ulepszanie genotypów bez mechanizmu lamarckowskiego. Z punktu widzenia optymalizacji/skuteczności, każde z trzech podejść (Darwin/Lamarck/Baldwin) może okazać się najlepsze w określonych warunkach środowiskowych (czyli dla określonego zachowania się funkcji celu, potrzeby eksploracji i eksploatacji).

"The Baldwin effect has two aspects. First, lifetime learning in individuals can, in some situations, accelerate evolution. Second, learning is expensive. Therefore, in relatively stable environments, there is a selective pressure for the evolution of instinctive behaviors." [Tur02].



Doświadczenie [YST99]: porównanie ewolucji Lamarcka i Darwina przy stacjonarnej (stałej) funkcji celu, zmiennej (powtarzającej się) funkcji celu i nieznanej funkcji celu. Przy stałej funkcji celu: ewolucja Lamarcka szybciej niż darwinowska się zbiega, znajduje lepsze rozwiązania (kolejne osobniki mogą kontynuować uczenie się swoich przodków!). Przy zmiennej: dopasowanie osobników ewoluowanych wg podejścia Lamarcka oscyluje, wg Darwina – polepsza się pomimo zmian. Przy nieznanej funkcji (po długim czasie ewolucji zmieniają się znacząco sposoby przeżycia): osobniki Lamarcka w ogóle nie potrafią się przystosować (nie potrafią porzucić uzyskanego wcześniej przystosowania), Darwina – w pewnym stopniu potrafią, szczególnie jeśli wcześniej były ewoluowane przy zmiennej funkcji celu.

Podczas ewolucji darwinowskiej (zmienna funkcja celu) może zajść ciekawe zjawisko – ich

wrodzona jakość (ta wynikająca bezpośrednio z genów) się pogarsza, chociaż ich dopasowanie do środowiska (radzenie sobie podczas życia) jest coraz lepsze.

Dwa sposoby na adaptację i dobre przystosowanie: *ability to learn* i *(innate) ability to perform. Innate* (ang.) – wrodzony. Podczas życia osobnik może przy pewnych założeniach przezwyciężyć negatywne cechy zapisane w genotypie („*ability to learn*"). Może mieć też dobre cechy wpisane w genotypie, ale wtedy zwykle ma słabsze zdolności adaptacji do nowych warunków.

Symbioza Uczenie zmienia krajobraz przystosowawczy (→ efekt Baldwina), ale podobnie jest z symbiozą. W dodatku oba te zjawiska ułatwiają ewolucję i prowadzą (przy odpowiednich warunkach i o ile jest taka możliwość) do umieszczenia odpowiednich genów w genotypie [WP99, p. 33].

Prosty eksperyment: organizmy mogą posiadać w genotypie wartości genów: „dobry", „zły", „neutralny". Nie ma krzyżowania. Dopasowanie osobnika bierze pod uwagę również osobniki znajdujące się blisko – tak, że wszystkie geny „neutralne" są wypełniane „dobrymi" albo „złymi" (pochodzącymi z pobliskich osobników). W ten sposób pojawia się dodatkowy czynnik (źródło zmienności), który ułatwia proces ewolucji (krajobraz funkcji dopasowania staje się łatwiejszy, mutacja – i ew. krzyżowanie – nie jest jedynym elementem wprowadzającym zmienność).

Ważny aspekt modelu: genetyczna zmienność jest wolniejsza niż ta, którą daje okres życia. Ocena dopasowania: dla każdego osobnika wybiera się losowo 1000 grup sąsiadów („okres życia"). Dla każdego wyboru z tych 1000 wyborów: jeśli osobnik + grupa dają razem wszystkie geny „dobre", ocena osobnika jest zwiększana o 1. Zatem krajobraz dopasowania jest bardzo wymagający. Inne parametry: mutacja 5%, selekcja ruletkowa, 1000 osobników w populacji.

Jeśli osobniki były gęsto rozmieszczone, ewolucja kończyła się na stanie symbiozy, gdy osobniki nawzajem uzupełniały się do genów „dobrych" (osobniki miały geny „dobre" i „neutralne", a „złe" wyginęły). Mniejsza dostępność osobników w środowisku wymusiła konieczność posiadania dobrych genów przez każdego osobnika (wyginęły geny „złe"), a symbioza stała się zbędna na pewnym etapie ewolucji (wyginęły też geny „neutralne").

Uzyskanie maksymalnego dopasowania – obserwowane w tym eksperymencie – nie miało miejsca, gdy nie wprowadzono mechanizmu symbiozy (tzn. gdy nie było wypełniania genów „neutralnych"). Być może w końcu dopasowanie by wystąpiło, lecz stałoby się to dużo wolniej (trudniejszy krajobraz dopasowania, sama mutacja). Wyłączenie krzyżowania miało na celu ułatwienie obserwacji zjawiska symbiozy. Gdy krzyżowanie włączono, symbioza również występowała, lecz było ją trudniej zaobserwować (krzyżowanie wprowadza też korzystny

gradient podczas ewolucji, podobnie jak uczenie i symbioza). Opisywany efekt nazwano *[symbiotic] scaffolding* (dosł. wznoszenie rusztowań, czy też „symbiotyczne wspieranie się").

Proces wolnej zmienności (ewolucja) jest prowadzona przez proces szybkiej zmienności (symbioza/współpraca/uczenie/lokalne przeszukiwanie). Można to wykorzystać w optymalizacji – szybkie, nietrwałe przeglądanie rozwiązań (czas życia, *lookahead*) oraz wolne przeglądanie (z pewnym zapisem informacji) – przykładem jest zachowanie algorytmu Steepest. Zauważmy, że kiedy osobniki wejdą już w stan symbiozy, nie ma presji do genetycznej asymilacji (jak w efekcie Baldwina) – nie ma nic złego w posiadaniu genów „neutralnych". Gdyby jednak warunki się zmieniły (np. sąsiedzi staliby się niewiarygodni, albo byłoby ich coraz mniej, albo zapożyczanie „dobrych" genów nie działałoby tak skutecznie), powstałaby presja, by każdy osobnik miał własne „dobre" geny – i uzyskanie wysokiego dopasowania stałoby się trudniejsze.

Symbioza a krzyżowanie: symbioza – rodzice odmienni, potomek sumą rodziców; krzyżowanie – rodzice podobni (ten sam gatunek), potomek z części rodziców.

W naturze najsilniejszą formą symbiozy jest symbiogeneza[18] (geneza nowych gatunków dzięki genetycznemu połączeniu symbiontów). Twierdzi się, że tak powstały komórki eukariotów, z których wywodzą się wszystkie rośliny i zwierzęta.

Przykładowe pytania

Scharakteryzuj teorie i mechanizmy ewolucyjne odwołując się do tego, co wnieśli Lamarck, Darwin i Baldwin.

Rodzaje uczenia Traktując uczenie ogólnie, możemy o tym procesie mówić na trzech (przenikających się!) poziomach:

- filogenetyczne: ewolucja jako proces uczenia (medium: genotyp)
- ontogenetyczne: podczas życia (medium: fenotyp)
- socjogenetyczne: rozwój w grupie, społeczeństwie (medium: fenotyp, wymagana komunikacja/interakcja)

5.5.3 Ukierunkowana a nieukierunkowana, ograniczona a nieograniczona

Podział procesów ewolucyjnych pod względem pochodzenia funkcji oceny:

[18] https://pl.wikipedia.org/wiki/Symbiogeneza

	Directed evolution	Non-directed evolution
Also known as (1)	evolutionary optimization	spontaneous evolution
Also known as (2)	genetic algorithm	coevolution
Fitness	exogenous (external)	endogenous (internal)
Fitness depends on	individual	individual and other individuals
Fitness formulation	known	unknown
Fitness landscape	static	dynamic
Positive and neg. selection	explicit "schemes"	implicit
Population size	constant	variable
Lifespan	irrelevant	usually crucial
Hardness	well-defined (a function)	fitness = environment...
Goal	well-defined (optimum)	none (implicit, variable)
Framsticks experiment	`standard.expdef`	`reproduction.expdef`

- *directed/guided evolution*: ukierunkowana (*exogenous fitness* – definiujemy kryterium dopasowania): Virtual Creatures (rozdział 5.10.1), Framsticks (`standard.expdef`), ...
  - *human-guided evolution* (sztuczny dobór: preferencje człowieka/użytkownika ukierunkowują) – np. `biomorph.show`. Stosowane szczególnie wtedy, kiedy nie można określić/sformalizować obiektywnej funkcji celu. Przykłady zastosowań to tworzenie portretów pamięciowych, sztuka (grafika[19], muzyka[20], rzeźbiarstwo, ...[21]), hodowanie według uznania stworzeń, robotów itp.
- *undirected/"spontaneous" evolution*: nieukierunkowana (*endogenous fitness* – bez odgórnego kryterium oceny): Tierra (rozdział 5.9), Avida, Framsticks (`reproduction.expdef/show` ...

Podział procesów ewolucyjnych pod względem ich potencjału i ograniczeń:

- *open-ended evolution*: nieograniczona (model osobnika na którym działa ewolucja nie posiada ograniczeń, np. złożoności): Tierra, Avida, Framsticks, ...
- *closed-ended evolution*: ograniczona – wiadomo co można, a czego nie można uzyskać.

[19] http://www.alife.pl/art_painter/p, http://www.alife.pl/art_painter/p/colorful.html
[20] http://www.alife.pl/musicgen/p/wstep.html
[21] http://www.alife.pl/dummies/p/program

5.5.4 Krytycznie...

„Ewolucja darwinowska jest nieuniknioną konsekwencją istnienia współzawodniczących systemów powielających informację, które znajdują się w skończonym środowisku w świecie o dodatniej entropii" [Atm92a; Atm92b].

Czy ewolucja wyjaśnia wszystko? Jest bardzo wiele niepewności i luk w teorii neodarwinizmu, które niełatwo będzie badać i dowodzić – nie możemy cofać czasu. Niekiedy propagatorzy ewolucjonizmu używają naciąganych dowodów i nadużywają autorytetu, przedstawiając hipotetyczne konsekwencje tej teorii na równi np. z istnieniem grawitacji i prawem powszechnego ciążenia – przez co dają łatwe argumenty przeciwnikom[22]. Tymczasem formalnie patrząc, brakuje wyjaśnień lub bezpośrednich dowodów na wiele z obserwowanych zjawisk[23]. Brak bezpośrednich dowodów nie jest zaskakujący – nie możemy przecież cofnąć się w czasie i bezpośrednio obserwować zachodzących wtedy, bardzo powolnych procesów. Stąd duża rola symulacji jako metody pozwalającej na analizy typu „jak mogło być" oraz „na ile to jest prawdopodobne".

Dyskusja: czy procesy ewolucyjne wystarczyły (por. [Küp91]) do rozwinięcia się życia w takim stanie, jaki obserwujemy obecnie? Tak / Nie jestem pewien / Nie.

Kilka problemów:

- Trudności w dowodzeniu eksperymentalnym i gromadzeniu obserwacji
- Zbyt duże przedziały czasowe i liczby, zbyt małe prawdopodobieństwa, zbyt duże przedziały niepewności – zgadywanie (niepewność o wiele rzędów wielkości)
- Wiele różnych teorii, przekonujących hipotez i ich odmian; dyskusje/walki naukowców i radykalnie różne poglądy

Popularne wątpliwości:

- Powstanie DNA: koduje ono powstawanie białek, ale samo nim jest – co było więc pierwsze?
- Płciowość: czemu służy, jak uzasadnić?
- Nadmiarowość ludzkiego mózgu: niepotrzebny wydatek? „Ewolucja by na takie coś nie pozwoliła!"

[22] https://www.youtube.com/watch?v=6XIEJdrbW9M

[23] https://www.youtube.com/watch?v=C3mqw4oxb2w, https://www.youtube.com/watch?v=xxh9o32m5c0, https://www.youtube.com/watch?v=2CnZ3n8I5b8, https://www.youtube.com/watch?v=e8MzPmkNsgU, https://www.youtube.com/watch?v=3LGm0iWPC80

- Jak mogły na drodze ewolucji powstać doskonałe lub ściśle zależne od siebie elementy organizmów, dla których trudno sobie wyobrazić „drogę małych ulepszeń”?

Ponadto, poza teorią ewolucji – pytanie o powstanie życia we wszechświecie:

- Ogrom świata: układy słoneczne w galaktykach, te w gromadach. Ok. 400 000 000 000 gwiazd w naszej galaktyce, możemy zobaczyć teleskopami około 2 000 000 000 000 *galaktyk*.

Czy na to pytanie da się kiedykolwiek odpowiedzieć stricte naukowo? Jakie są granice ludzkiego poznania? Czy ludzkość przy istniejących ograniczeniach może poznać więcej, niż zaniedbywalnie niewielki fragment całości?[24]

5.5.5 Artificial life helps understand evolution

From [Sip95]:

> Darwin's fundamental theory, while still sound today, is in need of expansion. For example, one well-known principle is that of natural selection, usually regarded as an omnipotent force capable of molding organisms into perfectly adapted creatures. Other factors can influence evolution besides natural selection. Certain complex systems tend to self-organize; that is, order can arise spontaneously. A major conclusion is that such order constrains evolution to the point where natural selection cannot divert its course.
>
> Another principle of Darwin's theory is that of gradualism – small phenotypic changes accumulate slowly in a species. Paleontological findings discovered over the years have revealed a different picture – long periods of relative phenotypic stasis, interrupted by short bursts of rapid changes. This phenomenon has been named *punctuated equilibria* by biologist Stephen Jay Gould. While a full explanation does not yet exist, the phenomenon has been recently observed in a number of ALife works, suggesting that it may be inherent in certain evolutionary systems.
>
> ALife offers opportunities for conducting experiments that are extremely complicated in traditional biology or not feasible at all. ALife complements biological research, raising the possibility of joint ventures leading to valuable new scientific discoveries. ALife also holds potential for developing new technologies: software evolution, sophisticated robots, ecological monitoring tools, and so on.

[24] Por. film „The Man from Earth”.

5.6 Modeling plants using *L-systems*

An L-system (a Lindenmayer system) is a type of a formal grammar, where all possible rules are applied in each step of development. L-systems can be deterministic or stochastic, context-insensitive or sensitive, and parametric or not.

- Basic information: https://en.wikipedia.org/wiki/L-system
- Comprehensive book [PL96] – first published in 1990, Lindenmayer[25] & Prusinkiewicz[26]
- Basic 2D demo: http://www.alife.pl/lsyst/p
- Basic 3D demo: http://www.alife.pl/lsyst/p/ls_3d.html
- More advanced: modeling development [JPM00] and climbing [Knu09]; can also be used to model differences in development in response to various environmental conditions: temperature, humidity, sunlight, fertilization...
- Used not just for modeling plants [Bou+12], but also for robot morphologies, architectural design (buildings, cities) [PM01], games [EE17], generating music [WS05], and in other applications, e.g. [Bie+18]. Can be evolved – available as 'fL' genetic encoding in Framsticks[27].

5.7 Emergence in *Boids*

Boids are a simple example of emergent phenomena. From [Sip95]:

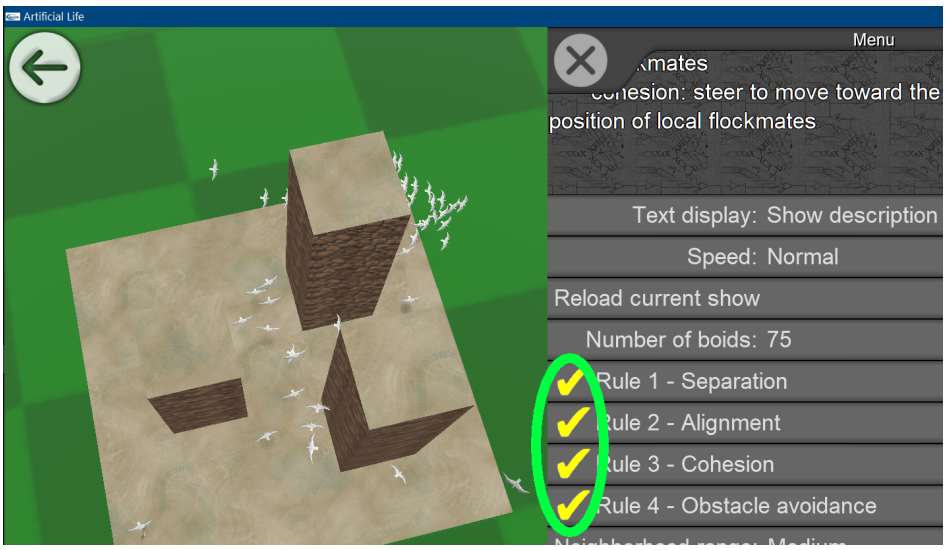
> Another process predominating ALife systems is that of emergence (pol. *emergencja*), where phenomena at a certain level arise from interactions at lower levels. In physical systems, temperature and pressure are examples of emergent phenomena. They occur in large ensembles of molecules and are due to interactions at the molecular level. An individual molecule possesses neither temperature nor pressure, which are higher-level, emergent phenomena.
>
> ALife systems consist of a large collection of simple, basic units whose interesting properties are those that emerge at higher levels (with no central controller). One example is von Neumann's model, where the basic units are grid cells

[25] https://en.wikipedia.org/wiki/Aristid_Lindenmayer
[26] https://en.wikipedia.org/wiki/Przemys%C5%82aw_Prusinkiewicz
[27] http://www.framsticks.com/a/al_genotype

(a CA, cellular automaton, Sect. 5.8) and the observed phenomena involve composite objects consisting of several cells (for example, the universal constructing machine). Another example is Craig Reynolds' work on flocking behavior.

Reynolds wished to investigate how flocks of birds fly, without central direction (that is, a leader). He created a virtual bird with basic flight capability, called a "boid". The computerized world was populated with a collection of boids, flying in accordance with the following three rules:

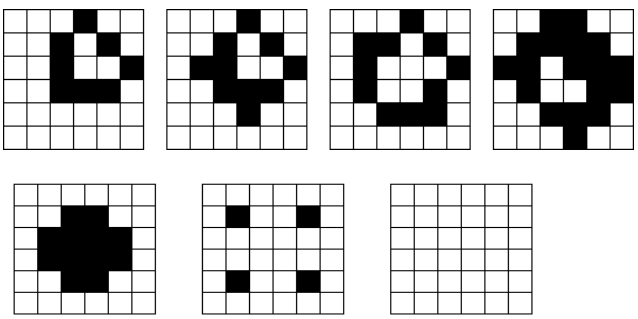
- Collision Avoidance: Avoid collisions with nearby flock-mates.
- Velocity Matching: Attempt to match velocity with nearby flock-mates.
- Flock Centering: Attempt to stay close to nearby flock-mates.

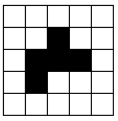


Each boid comprises a basic unit that "sees" only its nearby flock-mates. The three rules served as sufficient basis for the emergence of flocking behavior. The boids flew as a cohesive group, and when obstacles appeared in their way they spontaneously split into two subgroups, without any central guidance, rejoining again after clearing the obstruction (as observed in nature). The boids algorithm has been used to produce photorealistic imagery of bat swarms for the classical motion pictures *Batman Returns* and *Cliffhanger*.

5.8 Spatio-temporal dynamics in *Cellular Automata*

Eksperymenty lub prezentacje na laboratoriach lub ćwiczeniach: *Golly*, *DDLab*, *VoC*.





- szybowiec – przesuwa się
- działo – „produkuje” szybowce
- zderzenie szybowców – powstają inne poruszające się elementy
- odpowiednie zderzenie 13 szybowców – daje działo
- pożeracze – rozbijają i pochłaniają inne obiekty; odbudowują się

Istnieje rozbudowana systematyka obiektów dla reguł „Life”, w której dla każdej liczby aktywnych pól (złożoności automatu) opisane są wszystkie możliwe do zbudowania stany stabilne, oscylatory o różnych okresach, konfiguracje ruchome, konfiguracje powodujące nieograniczony rozwój, itd.

Współwynalazcą modelu CA (obok Johna von Neumanna) jest Stanisław Ulam (lata 40-te XX wieku).

Von Neumann próbował zbudować „uniwersalny konstruktor” który tworzyłby wskazane obiekty. A co gdyby kazać mu stworzyć siebie? (konstrukcję i program – nie tak jak z szybowcami, samą konstrukcję). Czy przypadkiem nie zapętliłby się? Gdyby wyłączyć program (potraktować jako dane) w odpowiednim momencie – nie zapętliłby się. Analogia do biologii: transkrypcja i translacja.

Uniwersalny konstruktor – dowiedziono, że da się zrobić, wymagane więcej niż 200 000 komórek, każda ma minimum 21 stanów.

Można w CA zrealizować bramki logiczne (np. bit to szybowiec). Stąd w CA można symulować każdy (uniwersalny) komputer. Czyli: komputer symuluje CA, które symuluje komputer! CA może nawet symulować komputer który symuluje CA, itd.!

Istnieje odpowiedniość takiego modelu CA i maszyny Turinga. Niektóre układy w CA będą „nieobliczalne” (o ile będą na tyle złożone, że wystąpi autoreferencja – odwołanie do siebie, np. przy zadaniu symulacji samego siebie).

W opisywanym aspekcie badań nad sztucznym życiem nie chodzi o podobieństwo modeli do rzeczywistości, lecz o nietrywialny sposób autoreplikacji – wówczas choć sztuczne twory nie są oparte na białku i nie żyją w naszej rzeczywistości – są bardziej żywe niż wirusy. Czy kiedyś posiądą świadomość?

W historii CA miały miejsce dwa szczególnie istotne wydarzenia. Pierwszym było wynalezienie przez von Neumanna (lata 50-te XX wieku) uniwersalnego konstruktora, który mógł stworzyć dowolny automat (w tym samego siebie) i wykonać dowolny program (uniwersalna maszyna licząca), ale był bardzo skomplikowany. Drugim było wynalezienie ponad 30 lat później przez Christophera Langtona „pętli Langtona" – prostego automatu, który umiał tylko powielać się. Udało się ostatnio skonstruować proste automaty, które potrafią się reprodukować i wykonywać proste operacje arytmetyczne. W dodatku umieją one dzielić się i rozmnażać do takiej ilości, by dostosować się do zadanych operacji i zrównoleglić ich wykonanie (wiele prostych, identycznych części rozwiązuje złożony problem). Pamiętajmy, że CA można łatwo i naturalnie zaimplementować w układach VLSI.

Przykładowe zastosowania CA: models and simulations of fluid dynamics, molecular dynamics, excitable media, biology, chemistry, environment, landslide, cryptography, VLSI, automobile traffic flow, financial markets, social behavior. Generating data sequences for reliability tests. Image processing and pattern recognition. Na przykład symulacja przepływu lawy, pożaru lasu, bioremediacji (bio-naprawy skażonej ziemi), trzęsienia ziemi, komórek organizmu, rozprzestrzeniania infekcji, itd.

Nie podaję tu literatury do każdego z powyższych zagadnień, bo bibliografia by się znacznie rozrosła, ale bardzo łatwo znaleźć szczegółowe artykuły.

> Niektóre automaty komórkowe demonstrują samoorganizację i własności emergentne. Te cechy, niezależnie czy występujące w automatach komórkowych, czy w innych modelach, mają swoje zastosowania: "Structure formation and self-organization can be applied to chemistry, material science, plasma physics, hydrodynamics, biological sciences, nanotechnology, nanorobotics, collective robotics, distributed building, fabrication of smart matter" [AHM99]. Przykładowe wykorzystanie: równoległe sortowanie rozproszonych w przestrzeni obiektów przez roboty, wyznaczanie parametrów i reguł zachowania się wielowymiarowej substancji wzbudnej w celu uzyskania jej określonej czasowo-przestrzennej dynamiki, projektowanie „*swarm algorithms*" (algorytmów „grupowych") działających w masowo równoległych środowiskach.

From [Sip95]:

> A *machine* in the cellular automata model is a collection of cells that can be

> regarded as operating in unison. For example, if a square configuration of four black cells exists, that appears at each time step one cell to the right, then we say that the square acts as a machine moving right.
>
> Von Neumann used this simple model to describe a universal constructing machine, which can read assembly instructions of any given machine, and construct that machine accordingly. These instructions are a collection of cells of various colors, as is the new machine after being assembled – indeed, any compound element on the grid is simply a collection of cells.
>
> Von Neumann's universal constructor can build any machine when given the appropriate assembly instructions. If these consist of instructions for building a universal constructor, then the machine can create a duplicate of itself; that is, reproduce. Should we want the offspring to reproduce as well, we must *copy* the assembly instructions and attach them to it. Von Neumann showed that a reproductive process is possible in artificial machines. (...)
>
> One of von Neumann's main conclusions was that the reproductive process uses the assembly instructions in two manners: as interpreted code (during assembly), and as uninterpreted data (copying of assembly instructions to offspring). During the following decade, it became clear that nature had "adopted" von Neumann's conclusions. The process by which assembly instructions (that is, DNA) are used to create a working machine (that is, proteins), indeed makes dual use of information: as interpreted code (translation) and as uninterpreted data (transcription).

Based on this video: https://www.youtube.com/watch?v=xP5-iIeKXE8, what can you tell about the CA and the "game of life" rules?

Complexity of self-replicating systems estimated in [bits] despite different environments [Mer97]:

- 800, C-program: more...
  ```
  main(){char q=34,n=10,*a="main()
  {char q=34,n=10,*a=%c%s%c;printf(a,q,a,q,n);}%c"; printf(a,q,a,q,n);}
  ```
- 500 000, Von Neumann's universal constructor about...
- 500 000, Internet worm (1988) about...
- 8 000 000, *Mycoplasma capricolum* about...
- 100 000 000, Drexler's assembler about...
- 6 400 000 000, Human
- 100 000 000 000, NASA Lunar Manufacturing Facility

Reguły dla CA można też odkrywać ewolucyjnie, tak by osiągnąć zadany cel, czyli aby automat wykonywał określone zadanie. Przykład [KAEiOG97, str. 271]:

one-dimensional, binary-state CA

rule table (radius = 1):

otoczenie (wejście)	000	001	010	011	100	101	110	111
środkowa komórka (wyjście)	0	0	0	0	0	0	0	1

Do wizualizacji działania takiego zespołu reguł używa się wykresu przestrzenno-czasowego (*space-time diagram*).

Przykładowe zadanie – CA ma zadecydować, czy jest na nim więcej niż połowa jedynek. Jeśli tak, to ma po kolejnych iteracjach osiągnąć same jedynki, jeśli nie – same zera. Aby nie próbować ręcznie konstruować zestawu reguł (*rule table*), optymalizujemy go np. ewolucyjnie, aby osiągnąć zadany cel.

Related topics:

- https://en.wikipedia.org/wiki/Garden_of_Eden_(cellular_automaton)
- https://en.wikipedia.org/wiki/Boolean_network
- https://en.wikipedia.org/wiki/Gene_regulatory_network#Boolean_network

5.9 Spontaneous (and open-ended) evolution in *Tierra*

Related: core wars[28].

From [Sip95]:

> Can open-ended evolution be constructed within a computer, proceeding without any human guidance? This issue was addressed by Thomas Ray who devised a virtual world called Tierra[29], consisting of computer programs that can undergo evolution [Ray92]. In contrast to genetic programming where fitness

[28] https://en.wikipedia.org/wiki/Core_War

[29] https://en.wikipedia.org/wiki/Tierra_(computer_simulation)

is defined by users, the Tierra creatures (programs) receive no such direction. Rather, they compete for the natural resources of their computerized environment, namely CPU time and memory. Since only a finite amount of these are available, the virtual world's natural resources are limited, as in nature, serving as the basis for competition between creatures.

Thomas Ray modeled his system on a relatively late period of earth's evolution known as the Cambrian era, roughly 600 million years ago. The beginning of this period is characterized by the existence of simple, self-replicating organisms, marking the onset of evolution that resulted in the astounding diversity of species found today. For this reason, the era is also referred to as the Cambrian explosion. Ray did not wish to investigate how self-replication is attained, but rather wanted to discover what happens after its appearance on the scene. He inoculated his system with a single, self-replicating organism, called the "Ancestor", which is the only engineered (human-made) creature in Tierra. He then set his system loose, and the results obtained were quite provocative: An entire ecosystem had formed within the Tierra world, including organisms of various sizes, parasites, hyper-parasites, and so on. The parasites, for example, that had evolved are small creatures that use the replication code of larger organisms (such as the ancestor) to self-replicate. In this manner, they proliferate rapidly without the need for the excess reproduction code.

Tierra inspired the development of another software platform, Avida[30].

5.10 Directed (guided) evolution and coevolution

5.10.1 Karl Sims – virtual creatures

▷ sims_evolved_virtual_creatures.mp4 (same at https://www.youtube.com/watch?v=JBgG_VSP7f8)

- Reprezentacja: grafy skierowane, mogą zawierać cykle, rekurencję i kopiowanie
- Typy połączeń elementów: sztywne, obrotowe, skręcające, uniwersalne, zgięcieobrót, obrótzgięcie, sferyczne
- Atrybuty położenia kolejnego elementu: pozycja, orientacja, skala, odbicie
- Receptory – czujniki: zgięcia, dotyku, kierunku oświetlenia

[30] https://en.wikipedia.org/wiki/Avida

- Efektory – mięśnie
- „Neurony": suma, iloczyn, dzielenie, suma z progiem, $>$, sgn, min, max, abs, jeżeli, interpolacja, sin, cos, arctg, log, expt, f. sigmoidalna, całka, różniczka, wygładzanie, pamięć, oscylacja-sin, oscylacja-piłokształtna.
- Sieć „neuronowa" – połączenia tylko pomiędzy elementami sąsiednimi fizycznie
- Funkcja oceny: stopień zawładnięcia „zasobem" (sześcianem)
- POPSIZE: 300
- Survival-ratio: 20%
- Model oceniania: all vs. best (patrz diagramy poniżej)

First published in [Sim94]:

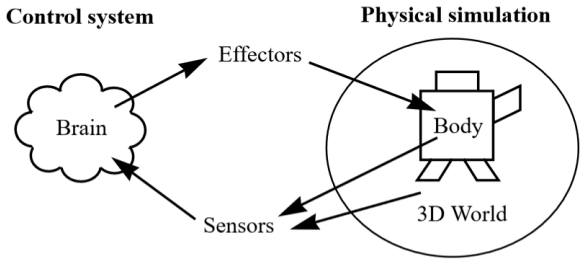
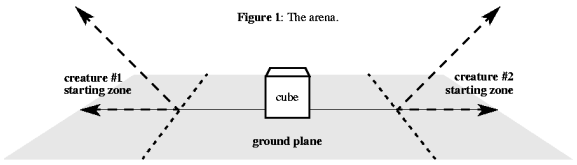


Figure 2: The cycle of effects between brain, body and world.



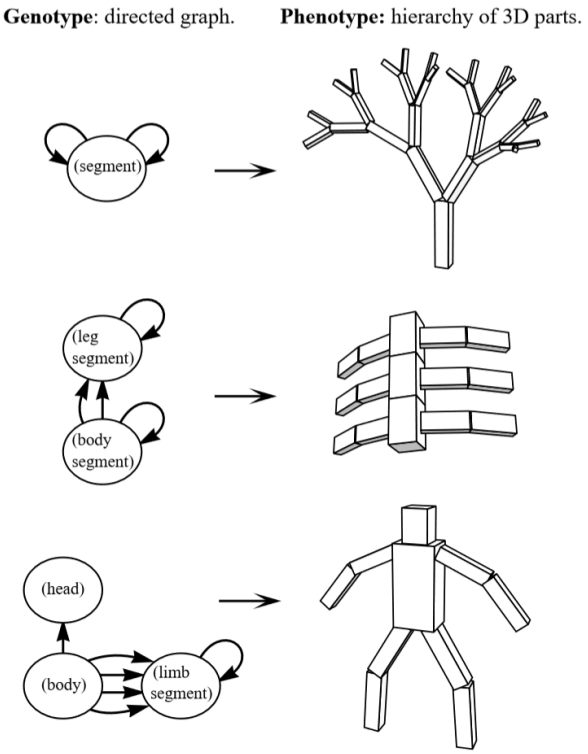


Figure 1: Designed examples of genotype graphs and corresponding creature morphologies.

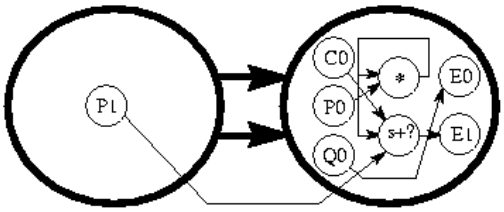


Figure 5: Example evolved nested graph genotype. The outer graph in bold describes a creature's morphology. The inner graph describes its neural circuitry. C0, P0, P1, and Q0 are contact and photosensors, E0 and E1 are effector outputs, and those labeled "*" and "s+?" are neural nodes that perform *product* and *sum-threshold* functions.

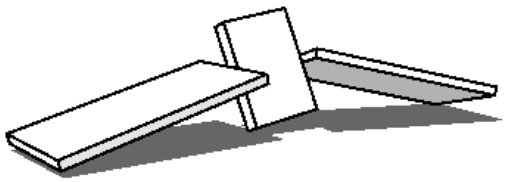
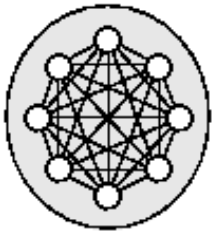
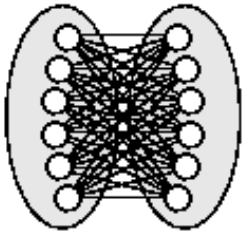


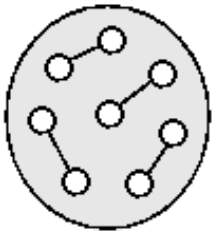
Figure 6a: The phenotype morphology generated from the evolved genotype shown in figure 5.



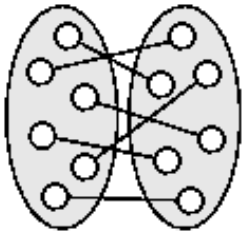
a. All vs. all, within species.



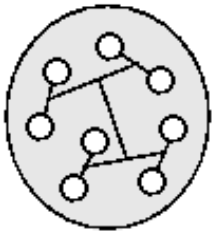
e. All vs. all, between species.



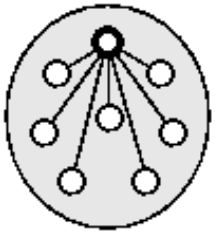
b. Random, within species.



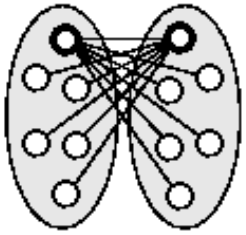
f. Random, between species.



c. Tournament, within species.



d. All vs. best, within species.



g. All vs. best, between species.

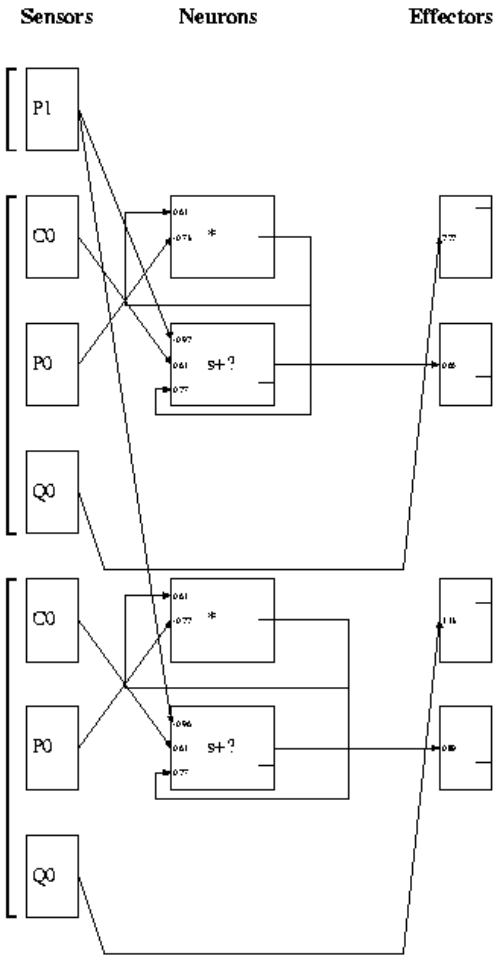


Figure 6b: The phenotype "brain" generated from the evolved genotype shown in figure 5. The effector outputs of this control system cause the morphology above to roll forward in tumbling motions.

20 years later, evolution (no coevolution): rigid bodies https://www.youtube.com/watch?v=fyVr7gdGEPE and soft bodies https://www.youtube.com/watch?v=HgWQ-gPIvt4.

5.10.2 Coevolution of pursuit and evasion

The experiment that illustrates such coevolution was described in [CM96].

▷ pe0.mp4 ▷ pe200.mp4 ▷ pe999.mp4 ▷ pe999200.mp4

- Symulacja: 2-D
- Osobnik: rekurencyjna sieć neuronowa („mózg"), fotoreceptory („oczy") w dowolnym

miejscu okręgu, patrzące w dowolnym kierunku, efektory (napęd) – „silniki"

- Genotyp opisujący osobnika: stałej długości, podzielony na pola. Pola informują o położeniu neuronów na okręgu, oraz o położeniu fraktalnego drzewa wyjścia tego neuronu i jego wejścia. Gdy wyjście i wejście dwóch neuronów się nakładają – powstaje połączenie. Gdy nie – wyjścia odpowiadają sterowaniu silników, wejścia – fotoreceptorom. Możliwe neurony „ukryte" – nie działają, lecz są przechowywane.

Algorytm ewolucyjny

- Dwie równoliczne populacje: uciekających i goniących.
- Krzyżowanie – tylko pomiędzy podobnymi osobnikami z danej populacji.
- Ocenianie:
  - Pierwsze pokolenie: każdy osobnik przeciwko losowo wybranym przeciwnikom
  - Następne: każdy osobnik przeciwko najlepszemu przeciwnikowi z poprzedniego pokolenia (LEO: Last Elite Opponent)

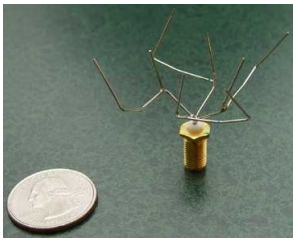
Ocena:

- uciekający: jak długo pozostał nie złapany
- goniący: ocena tym wyższa im bliżej dotarł do uciekającego; premia za dotknięcie (tym większa im szybciej)

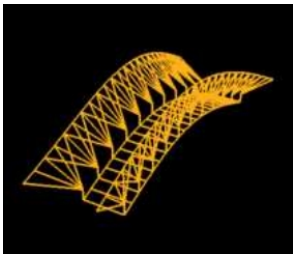
Nowe osobniki:

- krzyżowanie genotypów dwóch rodziców, wielopunktowe
- powielanie (zapobiegało zbytnim losowym zmianom genotypów przy krzyżowaniu prowadzącym do bezużyteczności potomków)
- mutacje

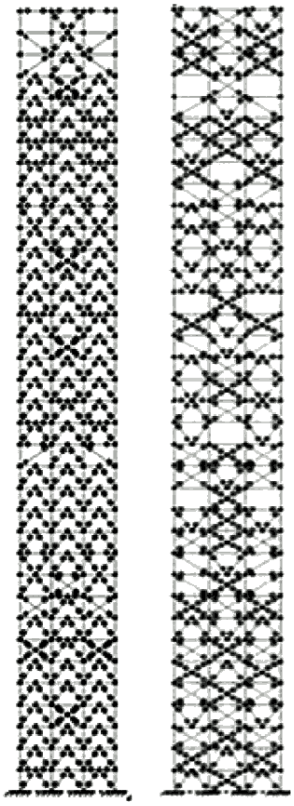
5.10.3 Optymalizacja konstrukcji (*Evolutionary design*)



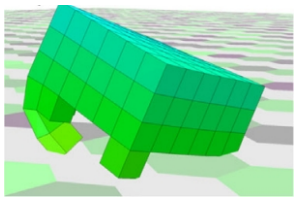
Automated Antenna Design with Evolutionary Algorithms, G. Hornby et al., 2006



Combining Structural Analysis and Multi-Objective Criteria for Evolutionary Architectural Design, J. Byrne et al., 2011

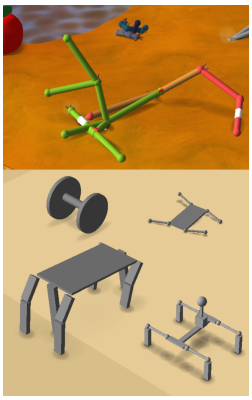


Evolutionary Design of Steel Structures in Tall Buildings, R. Kicinger et al., 2005

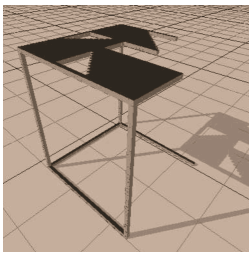


Evolutionary Developmental Soft Robotics As a Framework to Study Intelligence and Adaptive Behavior in Animals and Plants, F. Corucci, 2017

Evolving virtual creatures, K. Sims [Sim94]



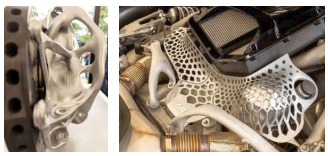
Framsticks [KU24]



Generative representations, G. Hornby [Hor03]

Inne przykłady:

- Wczesne ewoluowane roboty
  https://youtu.be/tgJcYx-yewA?t=237
- Optymalizacja turbin wiatrowych
  https://www.youtube.com/watch?v=cNaFhhwTpS8
- ...i wodnych (na co zwracać uwagę, co jest celem)
  https://youtu.be/fcE6HV1g2kk?t=2477
- Optymalizacja aerodynamiki samochodu
  https://www.youtube.com/watch?v=sw7_XWdd56c&t=25
- Optymalizacja struktury samochodu



> Czinger 21C: "Using supercomputing and AI (...) the chassis structure is generatively designed. Every component of the structure is pareto optimized for its precise function, not a single gram of material goes to waste." https://youtu.be/Pppne2jcgok?t=1541

Projektowanie ewolucyjne jest szczególnym przypadkiem automatyzacji projektowania.

Optymalizowane konstrukcje mogą być pasywne (statyczne) albo aktywne (wyposażone w siłowniki–efektory, a czasem również sensory). Jednym z przykładów ED jest zatem robotyka ewolucyjna.

Ćwiczenie z dyskusją: wymyśl kilka reprezentacji genetycznych do optymalizacji mostu.

Porównajmy złożoność klasycznego permutacyjnego problemu optymalizacji i problemu projektowania konstrukcji:

Własność	QAP/TSP	Optymalizacja konstrukcji
Skończony zbiór rozwiązań	+	−
Przestrzeń dyskretno-ciągła	−	+
Genotyp o stałej długości	+	−
Oczywista, naturalna reprezentacja	+	−
Prosta koncepcja sąsiedztwa	+	−
Wiele optimów lokalnych	+	++
Silne związki między fragmentami rozwiązania	+	++
Liczne ograniczenia	−	+
Wiele kryteriów oceniających	−	+
Kryteria trudno sformalizować	−	+
Determinizm oceny	+	−
Ocena obejmuje aspekt czasowy	−	+
Czasochłonna ocena	−	+
Przewidywalny czas oceny	+	−
Proste szacowanie podobieństwa	+	−

Depending on the level of granularity, evolutionary design can be:

- Conceptual ED: production of high-level conceptual frameworks for designs. New design concepts can be evolved, but building blocks are provided by the designer. Example: a hydropower system as a combination of locations, dam types, tunnel lengths and modes of operation.
- Generative ED: generation of the form of design directly. No pre-defined high-level concepts, no conventions, no imposed knowledge (the Einstellung effect). Low-level

building blocks defined. Complex representations. Examples: tables, heatsinks, optical prisms, aerodynamic and hydrodynamic forms, bridges, cranes, EHW, analogue circuits.

In evolutionary design, phenotypes are usually much more different from their genotypic representations, than in typical optimization problems. That means that mapping from genotype to phenotype ("embryogeny", pol. *embriogeneza*) is needed and may be complex – see Sect. 2.5.3. The goal is good **scalability** (the ability to scale up and create more sophisticated designs [Hor08]) and **evolvability** (the ability to produce offspring that are diverse/more fit [Gaj+19]) – consider the example of optimizing a *toothbrush* [discussion].

The same desirable properties of the genotype-phenotype mapping that we identified in the toothbrush example also apply when optimizing bridges, turbines, robots, cars, etc.

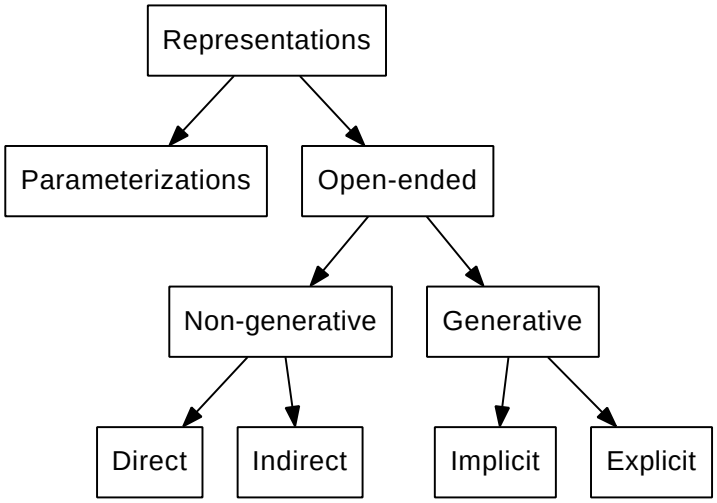
In nature embryogeny is defined by interactions between genes, their phenotypic effects and the environment in which the embryo develops. There are chains of interacting "rules"; the flow of activation is not completely predetermined and preprogrammed; it is dynamic, parallel and adaptive.[31]

In optimization (e.g., in evolutionary design) embryogenies can be [Ben99]:

- External (non-evolved). Fixed, static rules, which specify how phenotypes are constructed from the genotypes. E.g. *f0*, *f1*, *fH*, *f7* and *f9* in Framsticks [KU21].
- Explicit (evolved). Genotype and embryogeny are evolved simultaneously, but embryogeny is made of pre-defined blocks/features – like iteration, recursion, etc., as in GP (genetic programming). Specialized operators and representations are often needed. E.g. *f4* in Framsticks.
- Implicit (evolved). The same genes can be activated and suppressed many times; the same genes can specify *different* functions. Conditional iteration, subroutines, parallel interpretation of genes are allowed. However, it is very difficult to design a good implicit representation. E.g. *fB*, *f6* and *fL* in Framsticks.

Another, similar classification of embryogenies [Hor03]:

[31] https://nautil.us/the-strange-inevitability-of-evolution-235189/



In non-generative representations, each gene is activated once. In Direct and Explicit, the meaning of genes is fixed (not subject to evolution).

Question: to which category belongs: permutation in TSP, DNA in nature, *f9* in ED?

The development of an efficient embryogeny/mapping may be itself posed as an optimization or machine learning problem ("find an encoding that results in a smooth fitness landscape: maximize FDC" or "find an encoding that makes similar phenotypes genetic neighbors"), and may be addressed using techniques similar to word embeddings[32] or (neural) autoencoders[33] [KKM21].

Ontogeny (pol. *ontogeneza*) is the development during the life span (often means learning). It can be useful in evolutionary design, for example in evolvable hardware (EHW): designs can have self-repair mechanisms so that they can automatically correct faults within themselves and survive injuries.

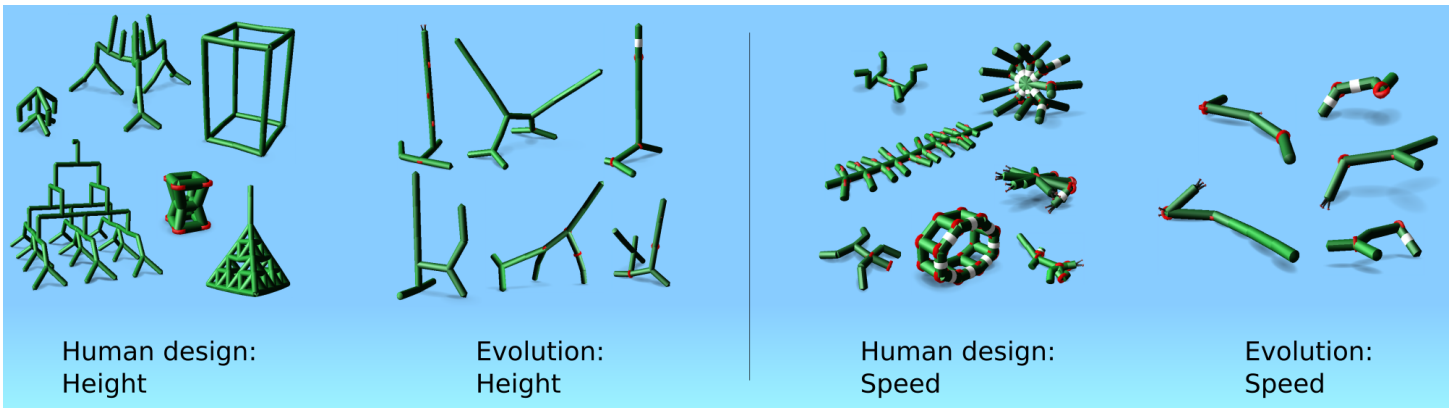
Evolutionary art. When using evolutionary design for aesthetic purposes, usually no crossover is used (no convergence wanted) and evolution is guided by a human. Output is attractive, but often does not have to be functional.

Human vs. natural design. From 7th Int. Conf. on Swarm Intelligence, session on Morphogenetic Engineering:

Engineered products are generally made of a number of unique, heterogeneous components assembled in a precise and complicated way, and work deterministically following the speci-

[32] https://en.wikipedia.org/wiki/Word_embedding
[33] https://en.wikipedia.org/wiki/Autoencoder

fications given by the designers. By contrast (compare Fig. 5.5), self-organization in natural systems (physical, biological, ecological, social) often relies on the repetition of identical agents and stochastic dynamics. Nontrivial behavior can emerge from relatively simple agent rules. However, most natural patterns (spots, stripes, waves, trails, clusters, hubs, etc.) can be described with a small number of statistical variables. They are either random or shaped by boundary conditions, but never exhibit an intrinsic architecture like engineered products do.



Rysunek 5.5: Human design vs. evolutionary design. Source: genotypes from the Framsticks distribution.

One monumental exception is biological development. Morphogenetic processes demonstrate the possibility of combining pure self-organization and elaborate structures. Multicellular organisms are composed of segments and parts arranged in specific ways that might resemble the devices of human inventiveness. Yet, they entirely self-assemble in a decentralized fashion, under the guidance of genetic and epigenetic information spontaneously evolved over millions of years and stored in every cell. In other words, they are examples of programmable self-organization – a concept not sufficiently explored so far, neither in complex systems science (for the "programmable" part), nor in traditional engineering (for the "self-organization" part). How do biological organisms achieve morphogenetic tasks so reliably? Can we export their self-formation capabilities to engineered systems? What would be the principles and best practices to create such morphogenetic systems?

The field of *Morphogenetic Engineering* explores the artificial design and implementation of autonomous systems capable of developing complex, heterogeneous morphologies without central planning or external drive. Particular emphasis is set on the mutual relationship between programmability/controllability and self-organization. Inspiration from evo-devo; applications in nanotechnologies, reconfigurable robots, swarm robotics, techno-social networks, etc.

Przykładowe pytania

Co to jest embriogeneza i ontogeneza (*embryogeny, ontogeny*)?

5.10.4 Building brains

CAM-Brain Project: Dr. Hugo de Garis, head of the Brain Builder Group at ATR, a research lab in Kyoto, Japan. CAM stands for cellular automata machine. The original goal formulated in 1993:

...to build an artificial brain with a billion artificial neurons, with evolved CA-based NN circuits, by the year 2001, with the help of international collaborators from 9 countries. In practice, in 1999, that number will be maximum 40 million neurons.

NN circuit modules with maximum 1000 neurons each, will be evolved at electronic speeds (a complete run of a GA with tens of thousands of circuit growths and fitness evaluations in less than a second), using a special evolvable hardware device (a CAM-Brain Machine – CBM). These CA-based neural circuit modules[34] (each with its own evolved user specified function) will be downloaded into user specified brain architectures embedded in a RAM based space of half a billion CA cells. The same CBM (programmable) hardware will then update the whole RAM CA space frequently enough (at over 150 BILLION CA cell updates a second) for real time control of robotic devices.

By the first quarter of 1999, the Brain Builder Group expects to see the completion of the design and fabrication of the CBM. During 1998 and 1999, attempts will be made to create an artificial brain architecture (with maximum 32000 modules) to control a robot kitten's many behaviors.

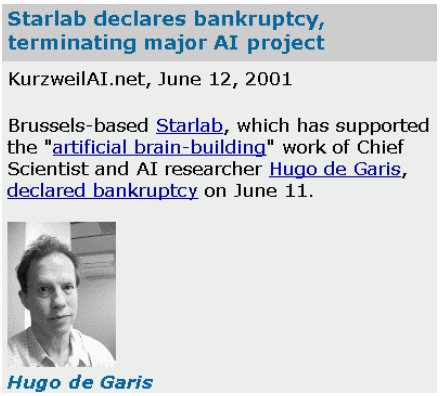
Starting in February 1998, the mechanical design of a robot kitten called "ROBOKONEKO" was begun, whose motions will be simulated and evolved (in combination with the CBM) using a simulation software. The evolution of maximum 32,000 modules for the artificial brain will be attempted in 1999 with the CBM, and placed in the large RAM memory. The I/O between the RAM brain will link via radio antenna with the (life sized) kitten robot.

If all the above is successful in the next year or two, then possibly after 2000, the BBG hopes to work on more ambitious projects, such as household cleaner robots, and to collaborate with substantially more brain builder teams around the world.

/* Mózg ludzki: co najmniej $2 \cdot 10^{10}$ neuronów, każdy ma 10 000 synaps na każdym drzewie dendrytowym i 10 000 synaps na aksonie. */

[34] https://en.wikipedia.org/wiki/CoDi

Bardzo duże fundusze, również komercyjne, znaczny udział wybitnych naukowców z całego świata, także ochotników – uczniów, studentów, młodzieży, silna propaganda, udział mediów. Nie udało się. Por. [Bul98], [Pas00].
Dyskusja: dlaczego? jakie były słabe punkty tego pomysłu?



Wet Artificial Brains; Chemical Computation: Systemy sterujące, w których medium są substancje chemiczne. Pewna analogia do automatów komórkowych i *Artificial Chemistry*[35]. Zalety:

- masywna równoległość obliczeń
- odporność
- związek z obliczeniami z wykorzystaniem cząsteczek biologicznych (rozdział 4.1)

Dwa rodzaje obliczeń chemicznych:

- brak zróżnicowania przestrzennego (np. dzięki mieszaniu): obliczenia polegają na pojawieniu się i zachodzeniu połączonych reakcji chemicznych; wynikiem są produkty reakcji określonych typów o określonych parametrach,
- lokalne różnice i przestrzenne zróżnicowanie (np. przez brak mieszania): nadają się bezpośrednio do przetwarzania danych 2D i 3D. Np. matryca komórek wypełnionych substancją wzbudną (*excitable medium*), która może posiadać jeden z kilku stanów. Zmiana stanu zależy od stanów sąsiednich komórek.

Zastosowania: budowa oscylatorów chemicznych, bramek logicznych, przetwarzanie obrazów. Nawet śledzenie celów, omijanie przeszkód, szukanie optymalnej drogi. Por. program demonstracyjny i edukacyjny *Organic Builder*[36] [Hut09].

5.10.5 Building brains and bodies

Szerokie zastosowania w robotyce, a zarazem wyzwanie dla robotyki.

[35] https://en.wikipedia.org/wiki/Artificial_chemistry
[36] https://sourceforge.net/projects/organicbuilder/

Obecnie ewolucja systemu sterującego („mózgu”) nie sprawia problemu. Nie jest też bardzo trudno wyewoluować samą strukturę (morfologię, „ciało”); używa się ewolucji ukierunkowanej. Największe zalety ma ewolucja całego robota: równocześnie struktury mechanicznej i systemu sterującego, lecz napotykamy na wiele trudności:

- Kodowanie – jak razem zakodować dwa tak odmienne aspekty?
- Bardzo silne współzależności, wysoka epistaza.
- Zmiany (mutacje, krzyżowanie) mogą powodować nieprzewidywalne skutki.
- Krajobraz przystosowania – bardzo trudny (niezliczone optima lokalne).

Prace są prowadzone m.in. w LEGO Lab, Aarhus, Dania.

Inspiracją może być natura i DNA, ale czy jest to najlepsze rozwiązanie? Co warto wykorzystać, z czego zrezygnować? Czy lepiej opisywać wzrost organizmu, czy jego samego? Jak ma wzrastać „mózg” w trakcie rozwoju robota? Jak konkretnie rozwiązać zadanie interpretacji genotypu i fazę wzrostu (rozwoju) organizmu do postaci końcowej? Skuteczność różnych podejść można ocenić i porównać dopiero realizując je, niestety trudno o systematyczne podejście i analizy teoretyczne; na razie to bardziej sztuka i doświadczenie niż nauka i gotowe zasady.

Prowadzone są też prace nad robotami, które w obliczu własnej awarii zbudują (zmierzą i „wyobrażą”) sobie model uszkodzonych siebie samych, a następnie same sobie wyewoluują (symulując siebie uszkodzonego) nowy, odpowiedni dla danego uszkodzenia system sterowania, i zaczną go używać. Podobnie, prowadzi się prace nad ewolucją „ciał” (konstrukcji), które będą odporne na uszkodzenia zarówno ciała, jak i systemu sterującego. Takie konstrukcje – w efekcie zbudowane tak, że są pozbawione słabych, pojedynczych elementów – w obliczu uszkodzeń będą dalej w stanie wykonać postawione zadanie – co prawda gorzej lub wolniej, ale z (być może niepełnym) sukcesem.

Pierwszy przykład robotów prawie samo-wytwarzających się na podstawie optymalizacji w symulowanym środowisku:

▷ robots/406974ai1.mp4 ▷ robots/406974ai2.mp4

▷ robots/406974ai8.mp4 ▷ robots/406974ai9.mp4

Rekonfigurujące się roboty: ▷ robots/reconf-robot-4x4ht4a.mp4

Sprężystość pomagająca w poruszaniu się: ▷ robots/dumbo2.mp4

Bezwładność pomagająca w poruszaniu się: ▷ robots/MonkeyBestLoco.mp4

5.11 Współczesne „symulatory fizyki”

Symulacje 3D. Sama symulacja mechaniki – środowiska Bullet https://pybullet.org/ oraz ODE https://www.ode.org/. Symulacja mechaniki + sieci neuronowych + ewolucja – Framsticks.

5.11.1 Framsticks

Więcej o tym środowisku w ramach zajęć laboratoryjnych; por. rozdział 6.

Cel i zakres badań

Cel: zbadanie możliwości i właściwości ewolucji, której poddane są stworzenia w warunkach symulowanych, możliwie zbliżonych do warunków życia na Ziemi: trójwymiarowe środowisko, opis organizmów przez genotyp determinujący budowę struktury fizycznej i sieci neuronowej, operacje zmieniające genotypy, wymagania energetyczne i pojęcie sprawności energetycznej, specjalizacja na poziomie osobniczym (gatunki) i w obrębie organizmu (specjalizacja kończyn), możliwość wykształcenia różnych sposobów przetrwania itd.

Najistotniejszym elementem badań jest poznanie i ocena możliwości ewolucji ukierunkowanej na złożone zachowania pojedynczych osobników i ich grup, oraz eksperymenty z ewolucją nieukierunkowaną (co dotychczas nie było badane w tak złożonych warunkach środowiskowych i symulacyjnych).

Możliwości symulatora

- Trójwymiarowa, mechaniczna symulacja sztucznego świata:
  - Stworzeń z wykorzystaniem metody elementów skończonych
  - Specjalizacji ich kończyn – tarcia, wytrzymałości, zdolności zdobywania energii przez asymilację, wchłanianie itd.
  - Płaskiego podłoża, terenu składającego się z bloków/wzgórz o różnej wysokości i środowiska wodnego
  - Kolizji niedestrukcyjnych i destrukcyjnych
  - Możliwość interakcji użytkownika w symulowany świat (przenoszenie stworzeń, umieszczanie porcji energii, ożywianie i uśmiercanie osobników)
  - Dwa tryby: MechaStick (szybka symulacja, uproszczone kolizje, sprężystość) i ODE (wolna symulacja, dokładne kolizje, bryła sztywna)

- Symulacja sterowania („mózgu”) stworzeń:
  - Sieci neuronowej o dowolnej topologii (każdy neuron może mieć różną charakterystykę)
  - Interakcji ze środowiskiem: receptorów (zmysłu dotyku, równowagi i lokalizacji energii) i efektorów (mięśni sterowanych siecią neuronową)
  - Ewoluowalnej komunikacji (sygnałów, nadajników, odbiorników)
  - Nowych typów neuronów definiowanych przez użytkownika
- Symulacja ewolucji:
  - Utrzymywanie zbioru genotypów pogrupowanych w pule genów
  - Utrzymywanie zbioru osobników pogrupowanych w populacje
  - Modyfikacja opisów stworzeń przy pomocy krzyżowania i mutacji
  - Ocenianie organizmów pod kątem wielu kryteriów (czasu życia, prędkości itd.)
  - Utrzymywanie bilansu energetycznego stworzeń (zysków i zużycia energii na poszczególne czynności)
- Parametryzacja większości operacji, środowiska, zasad symulacji i ewolucji, sposobu działania systemu, funkcji celu, neuronów – język skryptowy

5.12 Agent i środowisko

5.12.1 Complex Adaptive Systems (CAS), Multi-Agent Systems (MAS)

CAS: https://en.wikipedia.org/wiki/Complex_adaptive_system

MAS: https://en.wikipedia.org/wiki/Multi-agent_system

Prezentacje lub eksperymenty na laboratoriach lub ćwiczeniach: *StarLogo*[37], *NetLogo*[38], *Repast*[39].

[37] https://education.mit.edu/project/starlogo-tng/
[38] https://ccl.northwestern.edu/netlogo/
[39] https://repast.github.io/

5.12.2 Model Lotki–Volterry

Jest to popularny model oddziaływania dwóch populacji w relacji drapieżnik–ofiara, gdzie dwoma podstawowymi zmiennymi są liczba ofiar i liczba drapieżników. Równania i symulację znajdziesz tu: http://www.alife.pl/drapiezniki-i-ofiary-model-Lotki-Volterry. Porównaj zalety i wady modelowania równaniami różniczkowymi i wieloagentowego [dyskusja].

5.12.3 Sensor evolution

Creating life-like behavior in simulation or robots has increased our understanding of design and evolution of controllers for artificial systems. Despite the interrelationship between behavior, sensors, and other morphological characteristics of animal systems, the evolution of sensors is rarely the primary aim of scientific investigations. The choice of sensors for robots is often limited by practical or financial constraints, and sensors in simulation are often modeled without strong reference to biological sensors.

In natural evolution one finds impressive examples of the principle of exploiting new sensory channels and information they carry. Olfactory, tactile, auditory and visual, but also e.g. electrical and even magnetic senses have evolved in a multitude of variants, often utilizing organs not originally "intended" for the purpose they serve at present. Many biological sensors reach a degree of structural and functional complexity and of efficiency which is envied by engineers creating man-made sensors. Sensors enable animals to survive in dynamic and unstructured environments, to perceive and react appropriately to features in the biotic and abiotic environment, including members of the own species as well as predators and prey.

Synthesizing artificial sensors for hardware or software systems suggests a similar approach taken for generating life-like behavior, namely using evolutionary techniques in order to explore design spaces and generate sensors which are specifically adapted with respect to environmental and other fitness related constraints.

5.12.4 Osadzenie jako miara współzależności

Agenci (programy, roboty, organizmy) działają zwykle w pewnym środowisku. Ich zachowanie zależy od dynamiki agent-środowisko oraz wzajemnych zależności i możliwości interakcji (np. efektory, receptory). Aby móc porównywać różne agenty niezależnie od szczegółów ich budowy i natury, wprowadza się pojęcie *embodiment* (wcielenie, ucieleśnienie, upostaciowienie, osadzenie w środowisku).

Agent (system) **X** jest wcielony w środowisku **E** jeśli istnieją pomiędzy nimi kanały za-

kłócające. **X** jest wcielony w **E** jeśli dla każdego momentu czasu **t**, w którym istnieją **E** i **X**, podzbiór możliwych stanów **E** ma możliwość zmiany stanu **X**, oraz podzbiór możliwych stanów **X** ma możliwość zmiany stanu **E**.

Ta relacyjna definicja [QD99] pozwala ilościowo mierzyć *embodiment*, niezależnie od ontologii, ale z uwzględnieniem zachowania (*behavior*). Wyznaczona wartość zależy m.in. od przepustowości kanałów zakłócających. Miara ta nadaje się do oceniania organizmów żywych (np. bakteria *E. coli*), robotów, wirusów komputerowych, automatów komórkowych, itd. Jest to pierwszy krok w kierunku opracowania ogólnych miar porównujących ilościowo i obiektywnie wybrane właściwości tworów naturalnych i sztucznych, co jest o tyle istotne, że oba te światy coraz bardziej się przenikają, i np. uniwersalna miara inteligencji (nie działająca przez porównanie do inteligencji ludzkiej) byłaby bardzo pożądana.

5.12.5 Robotyka: hierarchia warstw sterowania

Podczas projektowania robotów działających w (zaszumionym) środowisku, np. chodzących po budynku i zbierających śmieci, wykorzystuje się hierarchiczną budowę ich mózgu (systemu sterującego). Składa się on z „warstw" (*layers*). Każda warstwa wykonuje funkcje bardziej złożone od warstwy niższej. Na przykład: pierwsza odpowiada za unikanie przeszkód (*obstacle avoidance*). Druga – za poruszanie się (losowe krążenie w środowisku – pokoju, budynku, itp.), i nie zajmuje się już unikaniem przeszkód. Wyższe warstwy mogą przekryć funkcje warstw niższych poprzez zastąpienie („nadpisanie") ich działania, chociaż niższe warstwy nadal działają kiedy dodajemy wyższe. Taka architektura przypomina w przybliżeniu budowę ludzkiego mózgu, w którym prymitywne warstwy odpowiadają podstawowym funkcjom (np. oddychaniu), a wyższe – bardziej złożonym (np. myśleniu abstrakcyjnemu). Podejście hierarchiczne pozwala na inkrementacyjne tworzenie robotów i ich inkrementacyjną optymalizację (przez sukcesywne dodawanie warstw).

Każda warstwa składa się z modułów zachowania, które komunikują się asynchronicznie, bez centralnego sterownika.[40] Np. warstwa wykrywania kolizji posiada moduły czujników, moduły wykrywania niebezpieczeństwa, i system silników – i te moduły komunikują się ze sobą uzgadniając decyzję, która wpływa na zachowanie.

Taka metoda demonstruje podejście typowe dla sztucznego życia i umiejscowionej, ucieleśnionej AI: *bottom-up*, zaczynając od prostych, elementarnych modułów, stopniowo budując w górę dzięki ewolucji, emergencji, i rozwojowi.[41] Tradycyjna AI używa metodologii *top-down*: złożone zachowanie (np. gra w szachy) jest analizowane i dzielone na części w celu zbudowania systemu, który ostatecznie odzwierciedli szczegóły tego zachowania.

[40] Por. architektura sterowania Eliry z opowiadania [Kom19].

[41] https://en.wikipedia.org/wiki/Artificial_intelligence,_situated_approach

5.12.6 Poziom autonomii

W wielu zastosowaniach, w których występują agenci (Internet, symulacja, robotyka) optymalny jest różny poziom ich autonomii [MKT99] – zależny od zastosowania i wymagań: prędkość działania, niski koszt, realizm, bezpieczeństwo, itp. Na autonomię wpływa samodzielne zachowanie, zakres dostępnych akcji, percepcja, pamięć, wnioskowanie, samokontrola, itd.

Poziom autonomii	Agent podąża w określone miejsce	Agent wykonuje czynność
Guided	Agent potrzebuje szczegółowych informacji o drodze (zewnętrznego sterowania)	Agent potrzebuje szczegółowych informacji o czynności (zewnętrznego sterowania)
Programmed	Agent jest zaprogramowany, by podążał po drodze unikając kolizji	Agent jest zaprogramowany tak, by wykonać czynność w odpowiednich okolicznościach
Autonomous$_1$	Agent decyduje o wyborze drogi	Agent decyduje o tym, jak wykonać czynność
Autonomous$_2$	Agent decyduje, czy...	Agent decyduje, czy...
Autonomous$_3$	Agent decyduje, co robić	Agent decyduje, co robić

Uznaje się, że do pewnych zastosowaniań nie potrafimy jeszcze skonstruować robotów o wystarczająco wysokim poziomie autonomii, i postęp jest tu powolny. Ale czy chcielibyśmy? „To, że możesz, nie oznacza, że powinieneś".

5.12.7 Cognitive architectures and artificial general intelligence

So far we encountered two very simple examples of cognitive architectures: a recurrent neural network in an agent (Fig. 5.3) when we discussed situatedness and embodiment, and a set of rules – LCS (Sect. 2.8).

Now let's broaden our view by briefly reviewing the properties of more elaborate cognitive architectures [KT20]. [youtube lecture video: pay particular attention to and learn about the concepts marked in yellow]

5.13 Formalne opisy systemu ewoluującego

5.13.1 Maszyny von Neumanna

1940-1950: von Neumann szuka formalnej definicji maszyny, która może się reprodukować i przekazywać mutacje swoim „potomkom" (a więc taka maszyna może uczestniczyć w ewo-

lucji). Taki model obejmuje 3 części [Tay99]:

1. Maszyna konstruująca **A**, czytająca opis φ innej maszyny **X**, interpretująca go i tworząca **X**:

$\mathbf{A} + \varphi(\mathbf{X}) \rightarrow \mathbf{X}$

+ : odpowiednie połączenie maszyn

$\rightarrow$: proces konstrukcji

2. Maszyna kopiująca **B** (kopiująca taśmę z opisem/instrukcjami):

$\mathbf{B} + \varphi(\mathbf{X}) \rightarrow \varphi(\mathbf{X})$

3. Maszyna sterująca **C**, która po połączeniu z **A** i **B** najpierw odpala **B**, potem **A**, potem podłącza **X** do $\varphi(\mathbf{X})$ i oddziela wynik od (**A**+**B**+**C**).

$\mathbf{A} + \mathbf{B} + \mathbf{C} + \varphi(\mathbf{X}) \rightarrow \mathbf{X} + \varphi(\mathbf{X})$

Niech **X** będzie (**A** + **B** + **C**):

$\mathbf{A} + \mathbf{B} + \mathbf{C} + \varphi(\mathbf{A} + \mathbf{B} + \mathbf{C}) \rightarrow \mathbf{A} + \mathbf{B} + \mathbf{C} + \varphi(\mathbf{A} + \mathbf{B} + \mathbf{C})$

A zatem mamy równanie reprodukcji maszyny i jej taśmy (opisu). Z punktu widzenia ewolucyjności układu ważne jest, że do taśmy wejściowej możemy dodać opis dodatkowego automatu **D**:

$\mathbf{A} + \mathbf{B} + \mathbf{C} + \varphi(\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D}) \rightarrow \mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D} + \varphi(\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D})$

Jeśli taśma wejściowa $\varphi(\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D})$ będzie ulegała takim mutacjom, że zmieni się jedynie część **D** (na **D'**), to wtedy konstrukcja będzie wyglądała tak:

$\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D} + \varphi(\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D}) \xrightarrow{\text{mutacja}} \mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D'} + \varphi(\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D'})$

W takiej sytuacji zdolności reprodukcyjne są odporne na mutacje **D**, a maszyna może ewoluować w stronę dowolnie złożonych potomków dzięki temu, że **B** (o stałej wielkości) umie

kopiować dowolny opis. Nie działa tu generalna zasada, że maszyna jest większa i bardziej złożona od strumienia wyjściowego, który generuje.

Dlaczego w ogóle potrzebny jest opis, by zaszło rozmnażanie? Jest on „*quasi-quiescent*", czyli jest bierny, nieaktywny podczas kopiowania. Kopiowanie go nie niszczy, co mogłoby mieć miejsce gdyby podmiot kopiowany działał (CA) / żył (biologia).

Można stwierdzić, w jakim stopniu proces reprodukcji jest zakodowany w rozwiązaniu, które podlega reprodukcji, a w jakim – w regułach środowiska. W CA wszystkie części **A**, **B**, **C**, **D** mogą być zakodowane *explicite* na taśmie $\varphi(\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D})$. Np. w systemie Tierra tylko **B** i **D** są zakodowane *explicite* na $\varphi(\mathbf{B} + \mathbf{D})$, a **A** i **C** są *implicite* w środowisku (systemie, algorytmie) i **nie mogą** podlegać ewolucji. Wtedy ewolucja nie może być całkowicie *open-ended*, a przekształcenie genotyp → fenotyp jest stałe (bo **A** się nie zmienia). Z kolei jeśli **B** jest zakodowany na taśmie, jego mutacja może uniemożliwić reprodukcję. W AE tylko **D** może się zmieniać.

5.13.2 Elementy algorytmu ewolucyjnego jako funkcje

Schemat działania AE można wyrazić za pomocą funkcji:

Przestrzeń genotypów: **G**

Przestrzeń fenotypów: **P**

Środowisko: **I** (zbiór sekwencji środowiskowych)

epigeneza	f_1: $\mathbf{I} \times \mathbf{G} \to \mathbf{P}$
selekcja	f_2: $\mathbf{P} \to \mathbf{P}$
przeżycie genotypu	f_3: $\mathbf{P} \to \mathbf{G}$
mutacja	f_4: $\mathbf{G} \to \mathbf{G}$
krzyżowanie	f_5: $\mathbf{G} \times \mathbf{G} \to \mathbf{G}$

5.14 Elementy teorii gier i gry ewolucyjne

W tym rozdziale spojrzymy na proces ewolucji (w sensie koewolucji, por. rozdział 2.12) agentów realizujących różne strategie z perspektywy teorii gier. Opiszemy relacje między agentami i ich zyski za pomocą „macierzy wypłat" oraz nauczymy się analizować dynamikę takiego systemu. Modele omawiane w ramach (ewolucyjnej) teorii gier stosuje się w biologii, ekonomii, socjologii, psychologii, polityce, czy też w obronności.

5.14.1 Pojęcia podstawowe

Kombinatoryczna teoria gier To matematyczna teoria gier, część teorii gier. Zajmuje się grami, które mogą być reprezentowane przez struktury drzewiaste (tj. w postaci ekstensywnej) – np. szachy, warcaby, go. Istotna jest kolejność zdarzeń podczas gry, w drzewie występują węzły decyzyjne, a drzewo zawiera wszystkie możliwe stany gry. Koniec gry następuje gdy gracz (przegrywający) nie może wykonać żadnego dopuszczalnego ruchu. Twórcy (1960): Elwyn Berlekamp, John Conway i Richard Guy.

Reprezentacja gry w postaci normalnej Inaczej niż w postaci ekstensywnej, w postaci normalnej[42] gra jest reprezentowana jako macierz definiująca wypłaty dla wszystkich graczy, którzy realizowaliby wszystkie możliwe kombinacje swoich strategii. Wyobraź sobie przykład – 3 graczy mających odpowiednio 4, 2 i 3 różne strategie postępowania. Minimalny przykład – 2 graczy, każdy 2 strategie.[43]

Ewolucyjna teoria gier To zastosowanie teorii gier w biologii ewolucyjnej. ETG bada dynamikę i równowagi w grach, w których występują populacje graczy. Strategie przez nich stosowane wyznaczają ich zyski (*fitness*), które zależą od strategii innych osobników. W przeciwieństwie do tradycyjnych założeń teorii gier, osobniki często postępują nie racjonalnie (nie świadomie), lecz według strategii zakodowanych na stałe w ich genomach.

Gra o sumie zerowej i niezerowej Suma zerowa – kiedy zysk gracza jest równy stracie przeciwników. Podział zasobu jest grą o sumie zerowej (więcej dla mnie to mniej dla innych). Podobnie szachy czy poker. Obopólna korzystna wymiana nie jest grą o sumie zerowej (z reguły wymiany handlowe są uważane przez wszystkich uczestników jako korzystne dla nich). Ekonomia też nią nie jest (zależnie od strategii, może doprowadzić do wzbogacenia lub zubożenia ogółu).

Każdy wygrywa / każdy przegrywa Sytuacja bez wygranej (*no-win*), w przeciwieństwie do sytuacji *win-win*, oznacza sytuację w której nikt nie wygrywa, albo ktoś (lub wszyscy) przegrywają. Bywa tak z różnych powodów, także z powodu tego, że najlepsza decyzja dla jednostki nie oznacza najlepszej decyzji dla ogółu (→ dylemat więźnia). Wojna jest często uznawana za sytuację bez wygranej („nie zaczynaj wojny, której już nie wygrałeś") także dlatego, że zwycięstwo militarne nie musi oznaczać polepszenia sytuacji („pyrrusowe zwycięstwo").

[42] https://pl.wikipedia.org/wiki/Gra_w_postaci_normalnej

[43] Zobacz proste przykłady w https://en.wikipedia.org/wiki/Normal-form_game

Pareto-optymalność Pareto-zysk: zmiana, która polepsza komuś sytuację, nie pogarszając sytuacji innych. Jeśli niemożliwe są takie zmiany, sytuacja jest Pareto-optymalna (narysuj przykład analizy Pareto zysków graczy).

W pewnych matematycznych wyidealizowanych modelach można pokazać, że wolny rynek osiągnie stan Pareto-optymalny. Nie zawsze sytuacja Pareto-optymalna jest dla wszystkich korzystna: przykładowo, totalitarna dyktatura jest taką sytuacją, bowiem każde polepszenie sytuacji poddanych pogarsza sytuację dyktatora.

Hicks-optymalność Sytuacja jest Hicks-optymalna (John Hicks), jeśli nie istnieje sytuacja, która dałaby większą sumę zysków graczy. Sytuacja Hicks-optymalna jest zawsze Pareto-optymalna.

Równowaga Nasha i jej stabilność To sytuacja w teorii gier (nazwana od nazwiska Johna Nasha[44]), kiedy każdy z graczy przyjął pewną strategię i żadnemu nie opłaca się jej indywidualnie zmieniać – chyba, że zrobią to też inni. Zbiór takich strategii (po jednej dla każdego gracza) tworzy równowagę Nasha (1950). Równowaga taka nie musi być Pareto-optymalna – jeśli nie jest, wtedy zyski wszystkich graczy mogą być zwiększone.

Dla modeli gier, w których gracze mogą zmieniać strategie (i znane są prawdopodobieństwa ich wyboru), zamiast zysku rozważa się oczekiwany zysk lub średni zysk.

Dana gra może mieć 0 lub więcej równowag Nasha[45]. Jeśli gracze mogą przyjmować różne strategie (tzw. strategie mieszane), wtedy gra ma przynajmniej jedną równowagę Nasha. Jeśli gra ma równowagę Nasha i gracze postępują całkowicie racjonalnie, to wybiorą strategie tworzące taką równowagę.

Przykład #1. Dwaj gracze wybierają liczbę od 0 do 10. Potem obaj wygrywają tyle złotych ile wynosi minimum z obu liczb, a dodatkowo, o ile liczby nie są równe, gracz podający wyższą z nich płaci 2 zł drugiemu. Równowaga Nasha – obaj gracze podają liczbę 0 (bowiem każda inna strategia może być polepszona, gdy gracz poda liczbę o 1 mniejszą od liczby przeciwnika).

Przykład #2. Dwie firmy mogą zainstalować droższe (lepsze) lub tańsze (gorsze) urządzenia do komunikacji ze sobą. Lepsze urządzenia opłacają się wtedy, gdy druga strona również ich używa. Dwie równowagi Nasha.

Przykład #3. Wybór strony drogi po której jedzie dwóch kierowców w przeciwną stronę, po jednej drodze. Dwie równowagi Nasha (obaj pojadą po lewej, obaj pojadą po prawej).

[44] Por. film https://en.wikipedia.org/wiki/A_Beautiful_Mind_(film).

[45] http://www.alife.pl/teoria-gier

Przykład #4. Dylemat więźnia. Jedna równowaga Nasha – obopólna zdrada. Lepsza jest strategia obopólnej współpracy, lecz taka sytuacja nie jest stabilna (lepiej zdradzić gdy współwięzień współpracuje).

Przykład #5. Urząd skarbowy (kontrolowanie podatników kosztuje) i podatnik (oszukiwać – zysk jeśli nie ma kontroli, nie oszukiwać – zysk gdy jest kontrola). Brak równowagi Nasha: jeśli nie kontrolują, będziemy oszukiwać. Jeśli oszukują, będziemy kontrolować. Jeśli kontrolują, będziemy uczciwi. Jeśli są uczciwi, nie opłaca się kontrolować.

Przykład #6. Ucieczka z zajęć. 

Przykład #7. Parytety, samochody elektryczne, ekonomia skali, ceny ekologicznej żywności, żarówki energooszczędne, sterowniki dla urządzeń (linux...), członkostwo w UE, ...

Aby równowaga Nasha była stabilna, musi zachodzić sytuacja, że po (nieskończenie) niewielkiej zmianie strategii gracza: (1) inni nie mają lepszej strategii, oraz (2) ten gracz pogorszył swoją sytuację.

Strategie i równowagi mieszane

Rozważmy sytuację racjonalnych (czyli, w rozumieniu teorii gier, egoistycznych) pasażerów. Decydują oni, czy opłaca się kupować bilety, czy płacić kary, znając cenę biletu, wartość kary, oraz częstotliwość kontroli. Przeanalizuj konkretny scenariusz z konkretnymi wartościami oraz skrajne zachowania pasażerów i konsekwencje dla przewoźnika i kontrolerów. Zauważ, że racjonalni pasażerowie, dzięki swojemu racjonalnemu zachowaniu, determinują dobór przez przewoźnika ceny biletu, wartości kary, oraz częstotliwość kontroli.

Dalej rozważmy przykład #5 (opisany powyżej).

US \ Podatnik	Płacę	Oszukuję
Nie kontrolujemy	0 0	10 −10
Kontrolujemy	0 −1	−90 −6

Strategia typu *max-min* to taka, która maksymalizuje najmniejszy zysk po wszystkich decyzjach przeciwnika. Dla US jest to kontrola (−6), dla podatnika – płacenie podatków (0). Nie jest to stabilna sytuacja.

Termin „mieszane” oznacza możliwość wyboru strategii z pewnym prawdopodobieństwem. Równowaga mieszana obejmuje wtedy rozkłady prawdopodobieństwa strategii (decyzji).

W rozważanym przykładzie występuje mieszana równowaga Nasha, w której *prawdopodobieństwo* kontroli jest dokładnie takie, by podatnik nie widział różnicy w zysku pomiędzy oszustwem i uczciwością. Obaj gracze stosują wtedy mieszane strategie, podatnik również. „Optymalne" prawdopodobieństwa kontroli i oszustw wynikają z tabeli zysków obu stron.[46]

Podatnik nie widzi różnicy pomiędzy oszustwem a uczciwością, gdy prawdopodobieństwo kontroli wynosi 0.1 (bowiem jego *oczekiwany* zysk z oszustwa wynosi $0.9 \cdot 10 + 0.1 \cdot (-90) = 0$, co jest równe zyskowi z uczciwości). US z kolei nie widzi różnicy między kontrolowaniem a nie, gdy podatnik oszukuje z prawdopodobieństwem 0.2 – bo $0.8 \cdot 0 + 0.2 \cdot (-10) = -2$, co jest równe $0.8 \cdot (-1) + 0.2 \cdot (-6)$. Te dwa prawdopodobieństwa tworzą mieszaną równowagę. Należy zwrócić uwagę, że dla danego gracza nie zależą one od jego zysków, tylko od zysków przeciwnika.

W **populacji** graczy, **prawdopodobieństwa strategii odpowiadają liczbie osobników** obierających daną strategię, a przebieg gry determinuje (koewolucyjną) dynamikę populacji.

Gry ewolucyjne

Rozważmy przykład #2 (powyżej), kiedy firm jest wiele.

gracz 1 \ gracz 2	Szybka	Wolna
Szybka	5 5	1 0
Wolna	0 1	1 1

[symetria względem przekątnej → gra symetryczna, gracze nierozróżnialni]

W takiej sytuacji dynamiczny proces uwzględniający możliwość zmiany decyzji graczy w populacji doprowadzi do ustalenia się równowagi – albo wszyscy będą mieli wolne łącza, albo szybkie. To, która równowaga (atraktor) ostatecznie wygra, zależy od początkowych proporcji w populacji; zastawszy określony atraktor w populacji, możemy wnioskować, co działo się z tą populacją wcześniej. Przy powyższej macierzy zysków, graniczną proporcją szybkie:wolne jest 1:4 (oblicz samodzielnie!). Wtedy nowy użytkownik nie wie co wybrać (oczekiwane zyski są równe), lecz podejmując jakąkolwiek decyzję i dołączając do populacji przeważa szalę i determinuje decyzje innych użytkowników (por. niechętne wdrażanie nowego, bezpiecznego standardu w telefonii).

[46] https://www.youtube.com/watch?v=7jBf5fzGBlk

Strategia ewolucyjnie stabilna (ESS), równowaga stabilna To taka strategia, że jeśli (prawie) wszyscy gracze ją stosują, to żadnemu z nich nie opłaca się zmiana strategii na inną. Strategia ta wcale nie musi być globalnie optymalna. ESS jest analogiem równowagi Nasha. W strategii ewolucyjnie stabilnej sporadyczne inwazje osobników realizujących odmienne strategie nie powodują zmiany postępowania większości osobników. Natomiast strategia nie-do-pobicia (*unbeatable*) jest odporna nawet na masowe inwazje.

Np. strategia bezinteresownej pomocy innym nie jest ewolucyjnie stabilna, ponieważ jeśli któryś gracz zmieni strategię na wykorzystywanie pomagających, uzyska nad nimi przewagę. Gracz po graczu, w końcu *prawie* wszyscy odeszliby od strategii pomocy (porównaj: gry ewolucyjne i strategie mieszane).

Często strategia postępowania grupy osobników nie jest stabilna: jakaś nowa, obca strategia wygrywa, więc oryginalne osobniki zmieniają swoje postępowanie na tę nową strategię, przy czym w pewnym momencie zmiana przestaje być opłacalna i ostatecznie obie strategie współistnieją w populacji (grupie graczy).

Nadracjonalność Jest to założenie (wpływające na własne decyzje), że wszyscy gracze wybierają swoje strategie prowadząc taką samą analizę, by maksymalizować swoje zyski.[47] Przykład 1: w dylemacie więźnia nadracjonalni gracze wybiorą współpracę. Przykład 2: w sytuacji, gdy n graczy ma bez możliwości komunikacji ze sobą wybrać jednego (sytuacja, w której wszystko dostaje jeden dowolny gracz, a jeśli będzie więcej chętnych to nikt nic nie dostanie), każdy z nadracjonalych graczy rzuci „monetą" o prawdopodobieństwie sukcesu $1/n$. Przykład 3: jeśli n osób zamierza przyjść na konsultacje w godzinach 13:00–16:00 i nie mają możliwości komunikacji, to aby uniknąć czekania w kolejce powinni wybrać godzinę swojej wizyty w następujący sposób: (wymyśl samodzielnie).

Niepotrzebne wyścigi zbrojeń mogą być eliminowane przez nadracjonalność – przykłady: niejawne przetargi, (względne) bezpieczeństwo państw, drzewa konkurujące o dostęp do światła, itd.

Różne motywacje graczy (różne podejścia do przyjmowania określonych rodzajów strategii lub zachowań w grze) skutkują różnymi przebiegami gry[48].

Dwa popularne przykłady takich podejść (których popularność wynika z ich skuteczności w wyjaśnianiu zachowań rzeczywistych) omówiliśmy powyżej – to równowaga Nasha i ESS. Innym przykładem jest równowaga „drżącej ręki"[49], gdzie gracze mogą sporadycznie wybierać

[47] https://en.wikipedia.org/wiki/Superrationality

[48] http://en.wikipedia.org/wiki/Solution_concept

[49] https://en.wikipedia.org/wiki/Trembling_hand_perfect_equilibrium

nieracjonalne strategie.

5.14.2 Modele

Ten podrozdział przedstawia popularne w teorii gier gry, które stały się znane dlatego, że 1) modelują sytuacje często występujące wśród ludzi, 2) z ważnymi konsekwencjami postępowania wedle podjętych decyzji, 3) których ostateczny rezultat bywa daleki od ideału. Ich nazwy są najczęściej charakterystyczne[50] i odwołują się do konkretnej, wzorcowej sytuacji („dylemat więźnia", „jastrząb–gołąb", itp.).

Problem rendezvous Dwoje ludzi umawia się na spotkanie w (jak się okazuje, dużym) parku[51]. Przychodzą oddzielnie. Mogą czekać lub szukać siebie. Co mają zrobić, by zmaksymalizować prawdopodobieństwo spotkania? Przeanalizuj cztery różne sytuacje.

Gra w cykora To jeden z najważniejszych modeli rozpatrywanych w teorii gier (*chicken game*[52]). Dwie osoby wsiadają w samochody, rozpędzają się i z dużą prędkościa jadą na siebie – ten kto pierwszy zahamuje lub zjedzie z trasy jest „cykorem" i przegrywa [Ray55]. Jest to gra o sumie niezerowej (bo niezbalansowana jest sytuacja kraksy – obaj gracze przegrywają. Inne sytuacje mogą być zbalansowane).

Nadracjonalność: przeciwnik jest racjonalny więc nie doprowadzi do kraksy, zatem mogę jechać do końca prosto? [przemyśl to]

Łatwo można zapewnić przegraną przeciwnika [jak?]

Istnieją tylko dwie niezdominowane strategie proste: jedziemy do końca lub skręcamy w ostatniej chwili. Skręcając wcześniej nic nie możemy zyskać w porównaniu z czekaniem na ostatnią chwilę, za to jeśli przeciwnik planował skręcić chwilę później – możemy stracić. Gra ma dwie równowagi Nasha – pierwszy gracz jedzie, drugi skręca oraz drugi gracz jedzie, pierwszy skręca.

W przeciwieństwie do dylematu więźnia najgorsza nie jest sytuacja asymetryczna (jeden jedzie, drugi ucieka), ale symetryczna (obu jedzie prosto do końca). Natomiast jeśli koszty „honorowe" byłyby większe od kosztów wypadku, gra odpowiadałaby typowemu dylematowi więźnia.

Sytuacja „gry w cykora" występuje w rzeczywistości, gdy postępowanie stron jest podyktowane dumą, lub któraś ze stron nie ma nic do stracenia.

[50] https://en.wikipedia.org/wiki/List_of_games_in_game_theory

[51] https://en.wikipedia.org/wiki/Rendezvous_problem

[52] https://en.wikipedia.org/wiki/Chicken_(game)

Analogiczna sytuacja biologiczna – podział pożywienia (zasobu) pomiędzy dwa osobniki o równej sile. Możliwe nastawienie pokojowe lub agresywne (walka, zadawanie ran).

Dylemat ochotników Wersja gry w cykora dla wielu graczy. Przykład – następuje awaria i ktoś musi zadzwonić po specjalistów (np. awaria prądu, wody itp.), przez co ponosi pewien koszt. Jest tyle równowag, ilu graczy, i wszystkie są Pareto-optymalne i Hicks-optymalne. Każdy gracz preferuje inną równowagę. Przykład macierzy wypłat:

	Ktoś inny zadzwonił	Nikt inny nie zadzwonił
Ty dzwonisz	4	4
Ty nie dzwonisz	5	0

Dylemat więźnia To jeden z najważniejszych problemów teorii gier. Dwóch zamieszanych w duże przestępstwo przestępców złapano za małe przewinienie. Policja wie, że ktoś z nich jest winien, lecz nie ma dowodów. Jeśli:

- będą współpracować ze sobą, odsiedzą niewielką karę za małe przewinienie (określenie *współpraca* dotyczy współpracy między przestępcami, nie współpracy z policją i oznacza, że obaj nie będą zeznawać),
- jeden zerwie współpracę i będzie zeznawał, a drugi nie, pierwszy zostanie uwolniony, drugi natomiast pójdzie siedzieć za poważne przestępstwo,
- obaj będą zeznawać, to obaj pójdą siedzieć, przy czym wyrok będzie z tego względu minimalnie złagodzony.

Zwykle każdy gracz chce maksymalizować swój zysk bez zwracania uwagi na zysk drugiego gracza. Problem jest następujący: niezależnie od postępowania drugiego, opłaca się zeznawać (a najlepiej jeszcze obiecać współwięźniowi, że będziemy współpracować, by on nie zeznawał). Jeśli natomiast żadna ze stron by nie zeznawała, wynik obu graczy byłby o wiele lepszy od oczekiwanego wyniku w pozostałych sytuacjach. Jednak gracz boi się współpracować w obliczu możliwości zdrady (zeznań) współwięźnia. Nawet jeśli mamy możliwość komunikacji, to czy możemy zaufać temu, co mówi współwięzień?

Tak więc często każdy więzień realizuje egoistyczny cel zeznając, i skazując siebie i współwięźnia na wyrok.

Pewnym rozwiązaniem dylematu jest *iterowany* dylemat więźnia (IPD) – wielokrotne rozgrywanie między dwoma graczami dylematu więźnia. Wtedy zysk z zerwania współpracy jest o wiele niższy od straty spowodowanej brakiem współpracy w następnych turach. PD występuje też w wersji wieloosobowej.

5.14.3 Strategie zachowań społecznych i dylematy społeczne

Dylematy społeczne (*social dilemmas*) – sytuacje współzależności społecznej, w których długoterminowy interes społeczny jest sprzeczny z doraźnym interesem jednostki. Jeżeli większość członków grupy wybierze własne korzyści, to w konsekwencji cała grupa traci[53]. Dylematy społeczne rozważa się w kontekście prób utworzenia i utrzymania współpracy w społeczeństwie; przedstawia je klasyczna opowieść z 1968 roku, *The Tragedy of the Commons* („Tragedia wspólnego pastwiska").

Najbardziej popularny dylemat społeczny to (iterowany) dylemat więźnia – *(Iterated) Prisoner's Dilemma*. Trochę inny jest dylemat drwali (*Lumberjacks' Dilemma*), w którym rozważa się nie dwóch agentów, a wielu drwali i wiele drzew. Kooperacja: drwale czekają, aż drzewa urosną, by je razem ściąć i podzielić się drzewem. Ale ścinając małe drzewo samemu można zyskać więcej (kosztem pozostałych). IPD: gra o niezerowej sumie (pewien zysk jednego gracza niekoniecznie oznacza taką samą stratę przeciwnika) i nie kooperacyjna (graczom nie wolno się przed grą umawiać).

[53] https://pl.wikipedia.org/wiki/Dylemat_spo%C5%82eczny

Kiedy macierz zysków (*payoff matrix*) jest symetryczna, wystarczy ją przedstawić dla jednego gracza (w tabeli są nazwy z perspektywy agenta 2):

agent 2 \ agent 1	Cooperate	Defect
Cooperate	Reward	Sucker
Defect	Temptation	Punishment

T > P > R > S : Deadlock

T > R > S > P : Chicken (hawk–dove) game[54] (lepsza jednostronna zdrada, niż obopólna współpraca)

R > T > P > S : Assurance (stag hunt) game[55] (lepsza obopólna zdrada, niż jednostronna współpraca)

T > R > P > S : Prisoner's dilemma[56] (np. dla Agenta2: R=3, S=0, T=5, P=1)

Dobierając parametry R, S, T, P można ustalać proporcje opłacalności poszczególnych zachowań. Warunek oznaczony „Prisoner's Dilemma" motywuje graczy do braku współdziałania; często rozważa się też warunek 2R>S+T (zabezpieczający przed wspólną kooperacją, np. w IPD). Z kolei w *Assurance game* najbardziej opłacalna sytuacja współpracy jest atrakcyjna dla obu graczy, ale ci mogą nie zdawać sobie z tego sprawy i stosować strategie optymalne dla PD lub dla *Chicken game*. Podczas analizy bardziej praktycznych, rzeczywistych scenariuszy opisanych powyższymi relacjami bada się dynamikę zachowań, przestrzeń strategii, optymalność, Pareto-optymalność i stabilność rozwiązań, a także ich odporność na zakłócenia środowiska i prowokacje – por. np. [SPL06].

Macierz PD dla obu graczy można też przedstawić jako (agent 2—agent 1):

agent 2 \ agent 1	Cooperate	Defect
Cooperate	win—win	lose much—win much
Defect	win much—lose much	lose—lose

Wiele sytuacji w życiu ma własności podobne do dylematu więźnia! Na przykład wymiana towaru za pieniądze, w zamkniętych torbach. Zawsze opłaca się wtedy przekazać pustą torbę (w wersji nieiterowanej ⌣̈). W polityce istnieją analogiczne sytuacje – na przykład kraje, które mogą powiększać swoje armie lub podpisać rozejm. Prowadzi to do wyścigu zbrojeń, a „racjonalna" decyzja każdej ze stron daje kompletnie nieracjonalny efekt globalny. Inny przykład to wyścigi rowerowe: dwaj rowerzyści na czele, aby utrzymać się przed peletonem muszą zdobyć się na szybszą jazdę – a zatem współpracować: jeden bierze na siebie trudną

[54] https://en.wikipedia.org/wiki/Chicken_(game)
[55] https://en.wikipedia.org/wiki/Stag_hunt
[56] https://en.wikipedia.org/wiki/Prisoner%27s_dilemma

rolę prowadzącego, drugi jedzie za nim bez oporów powietrza. Jeśli się zmieniają, dużo zyskują. Często bywa tak, że jeden z nich jest bardziej zmęczony, a przed metą drugi „zdradza" i wygrywa wyścig.

Jednym z popularnych sposobów uniknięcia dylematu jest dobrowolne przyjęcie na siebie kary w przypadku jeśli zerwie się współpracę, a drugi gracz będzie współpracował. Tak działają różne systemy honorowe, w tym świat przestępczy. Jeśli obie strony uczestniczą w tego typu systemie honorowym i są świadome tego u przeciwnika, mogą zaryzykować współpracę, na czym obie zyskają.

Wnioski z PD powodują, że w niektórych krajach zakazano darowania lub łagodzenia kar w zamian za zeznania oskarżonych. Często bowiem podejrzani zeznają przeciw sobie nawzajem, chociaż obaj są niewinni. Jeszcze gorzej jest gdy tylko jeden z nich jest winny – wtedy niewinny zwykle nie będzie obciążał winnego (może nawet nie wiedzieć o jego winie), a winny zezna przeciw niewinnemu by złagodzić sobie karę i skomplikować sprawę.

W IPD pokusa zdrady może być przewyższona obawą przed karą za zdradę w kolejnych iteracjach, stąd większe prawdopodobieństwo uzyskania współpracy. Ogólnie (średnio) optymalna jest strategia Tit-for-Tat[57] (wet za wet, oznaczę tutaj TfT). Istnieją lepsze strategie, ale przy założeniu konfrontacji z konkretnymi przeciwnikami.

Inne, bardziej skomplikowane strategie są trudno przewidywalne, więc przeciwnik może zdecydować, że najlepiej ciągle zdradzać w rozgrywce z nimi. Grając z TfT wiadomo, kiedy oczekiwać współpracy. Czasem opłacalne są modyfikacje polegające na okazjonalnej współpracy pomimo wcześniejszej zdrady przeciwnika – to okazja do „przebaczenia", przydatna na przykład gdy informacja o działaniu przeciwnika może być zaszumiona.

TfT nie jest jednak ewolucyjnie stabilna względem różnych innych strategii. Wyobraźmy sobie ewolucję strategii IPD, gdzie zaczynamy od strategii losowych. Jeśli gramy z przypadkowymi, nieznanymi graczami, opłaca się zawsze zdradzać. Jeśli gramy z zawsze zdradzającymi, wśród których pojawi się choć trochę TfT–podobnych: opłaca się zawsze TfT. Jeśli gramy w większości z TfT, opłaca się „wielkoduszne" (przypadkowo wybaczające zdradę, ang. *generous*) TfT. [Zastanów się: dlaczego wybaczanie musi być przypadkowe, a nie np. „wet za dwa wety"?] Grając z wielkodusznym TfT, opłaca się cały czas współpracować. A grając z zawsze współpracującymi, opłaca się zawsze zdradzać – i dalej zaczynamy od czwartego zdania tego akapitu.

Jeśli wiadomo, że IPD będzie trwał n iteracji, przy racjonalnych graczach równowagą Nasha jest zdradzanie w każdej iteracji [wymyśl dlaczego?] Dlatego warunkiem sensowności IPD jest niewiedza o tym, kiedy się skończy.

[57] https://en.wikipedia.org/wiki/Tit_for_tat

Kooperacja populacji może być modelowana przez IPD dla wielu graczy. Interesujące jest pytanie, czy zachowanie altruistyczne (kooperacja) mogło wyewoluować z czysto egoistycznych zachowań dzięki mechanizmom selekcji naturalnej (por. omówiony pod koniec rozdziału 5.5.1 eksperyment z selekcją grupową we Framsticks i ewolucją umiaru:
▷ `frams/frams_evol_of_restraint.mp4` oraz analizę wpływu moralności i zabezpieczeń na wybór zachowań [KŻ20]).

Wykorzystanie publicznych zasobów Dwa niebezpieczeństwa:

- maksymalizacja własnego zysku przez racjonalnych, działających niezależnie graczy powoduje zniszczenie zasobu (*tragedy of the commons* – „Tragedia wspólnego pastwiska”[58]). Gracz nie ponosi całych kosztów swojej działalności – przynajmniej krótkoterminowo (opóźnione konsekwencje), albo dopóki liczba takich zachowań nie przekracza pewnego progu. Przykłady: wypasanie nadmiernej ilości bydła zamieniające pastwisko w nieużytek, skażenie wody przez zanieczyszczenia fabryczne, wycinka lasów, zbyt intensywny połów ryb w oceanach, zaśmiecanie środowiska, spamowanie, jazda na „długich” światłach, jazda zbyt dużym samochodem, wybór najkrótszej trasy[59], globalne ocieplenie, kradzież worków z piaskiem podczas powodzi. Analogie do dylematu więźnia (ciągłe „zdradzanie”) i ekonomii (własność prywatna, publiczna, socjalizm liberalny, itd.)
- zablokowanie dostępu do zasobu przez uprawnionych do tego, racjonalnych i działających niezależnie graczy powoduje jego niewykorzystanie (*tragedy of the anticommons*).

Ćwiczenie: opisz i przeanalizuj (Hicks- i Pareto-optymalność) poniższe sytuacje w kategoriach teorii gier.

- przepuszczanie (lub nie) pieszych czekających na przejście przez jezdnię – ciągły sznur samochodów (w momencie zauważenia pieszego jest już dosyć późno na hamowanie)
- wydeptywanie scieżek–skrótów na narożnikach trawników („tylko raz przejdę”). Możesz rozważyć dwa rodzaje ludzi jako graczy: tych, którzy lubią niewydeptane trawniki, oraz tych, którym to obojętne
- oszukiwanie podzielników ciepła na kaloryferach (zasłanianie, by wskazywały mniejszą wartość)
- oszukiwanie w rankingach (oszust trafia na szczyt rakingu, wszyscy inni spadają tylko o jeden)

[58] William Forster Lloyd, 1833; Garrett Hardin, 1968, https://science.sciencemag.org/content/162/3859/1243
[59] https://pl.wikipedia.org/wiki/Paradoks_Braessa

- argument nieuchronnosci i swobody wyboru zła: „jesli nie ja, to ktoś inny i tak by to zrobił” (zaopatrzenie w broń, udostępnienie narkotyków, udostępnienie nielegalnego oprogramowania) – por. film „Lord of War”
- jeden uzbrojony „terrorysta” i duża liczba słabiej uzbrojonych osób – rozważ różne strategie słabszych oraz racjonalną strategię terrorysty w przypadku kiedy jest i kiedy nie jest świadomy strategii słabszych
- powszechny dostęp do broni jako gwarancja bezpieczeństwa (powracające dyskusje w USA)
- głosowanie na znajomych, a nie merytoryczne (przykład: Mazury jako Cud Natury[60])
- przebudowa śmietników, by były zamykane na klucz – tylko dla pobliskich mieszkańców
- obrona przez powodzią (umacnianie wału vs. ochrona własnego domu) 
- obowiązkowe używanie świateł (nawet w dzień) podczas jazdy samochodem
- obowiązkowe szczepienia (a przecież każdy chce być bezpiecznym bez kosztów i ryzykowania efektów ubocznych szczepionki)
- udział w sankcjach przeciwko jakiemuś krajowi: im więcej krajów bierze w nich udział, tym większa pokusa, żeby się wyłamać (lepszy biznes!)
- opór przeciw opresyjnej władzy („Mój protest nic nie da”)
- montowanie drzwi antywłamaniowych, inwestowanie w zaawansowane i kosztowne systemy bezpieczeństwa
- społeczność, w której każda osoba może być uczciwa lub nie (typowa sytuacja); oszuści więcej zyskują wchodząc w relacje z uczciwymi niż uczciwi z uczciwymi, ale życie wśród (prawie) samych oszustów jest koszmarem i dla uczciwych, i dla oszustów (por. analogiczny, ale bardziej złożony model [KŻ20]).

Zastanów sie, jakie są równowagi Nasha i jak zachowaliby się gracze nadracjonalni w każdej z tych sytuacji. Uwaga: zwykle w teorii gier najważniejsze nie są techniczne tricki z liczbami w tabelce oraz obliczenia, ale sformalizowanie rzeczywistej sytuacji (zidentyfikowanie graczy, ich punktów widzenia i strategii). Pomyśl, jakie środki będą skuteczne, żeby – uwzględniając niedoskonałość człowieka, jego nieufność i skłonność do egoizmu – przeciwdziałać

[60] *New Seven Wonders* – akcja w Polsce: „Jak podkreślają organizatorzy konkursu, głosowanie ma 3 główne zalety: jest bardzo szybkie, głosuje się tylko i wyłącznie na Mazury, oraz można głosować bez ograniczeń, nawet z jednego telefonu.”

niekorzystnym globalnie sytuacjom, które indywidualnie wydają się ludziom najlepsze. Czy znasz jakieś osoby, instytucje lub dzieła, które promują takie środki?

5.15 Modele życia biologicznego – wybrane przykłady

Na zajęciach z tego przedmiotu skupiliśmy się na jednym z dwóch głównych celów sztucznego życia wymienionych w rozdziale 5.1 – na „lepszym zrozumieniu praktycznych, stosowanych modeli sztucznego życia w celu poprawy ich skuteczności i wydajności", jako że ten cel jest bardziej istotny dla informatyki. Nie rozmawialiśmy zbyt wiele o drugim celu – o „lepszym zrozumieniu natury dzięki badaniom istniejących zjawisk biologicznych", stąd informacje z tego rozdziału (5.15) nie obowiązują na egzaminie.

W niniejszym rozdziale poznamy kilka przykładów eksperymentów prowadzonych z użyciem technik sztucznego życia (i podejścia *bottom-up*), pozwalających na zamodelowanie wybranych aspektów życia biologicznego. Celem tych eksperymentów było przede wszystkim odzwierciedlenie i zbadanie zjawisk i procesów, których nauki biologiczne i rozważania teoretyczne nie potrafią obecnie do końca wyjaśnić.

5.15.1 Badanie prawa Herrnsteina

Badanie prawa odpowiedniości (ang. *matching law*) Herrnsteina [ECAL99, str. 225]. W 1961 Herrnstein stwierdził, że u wielu stworzeń częstotliwość odpowiedzi na różne bodźce odpowiada pozytywnemu pobudzeniu, jakiego te bodźce dostarczają. Czy taka strategia w populacji różnych stworzeń jest stabilna? Inni twierdzą, że stworzenia wybierają zawsze najlepszą możliwość (reguła zero-jeden). Czy taka z kolei strategia jest stabilna?

Hipoteza Herrnsteina bardziej formalnie: „*the animal allocates its behavior in proportion to the rewards it has obtained from them*":

$$\log \frac{F_A}{F_B} = \log k + b \log \frac{r_a}{r_b}$$

A, B – alternatywy

F_A, F_B – częstotliwości ich wybrania

r_a, r_b – nagroda za wybór A i B

k, b – współczynniki skalujące

Po symulacji ewolucji okazało się, że w środowisku ze współzawodnictwem wyłania się zachowanie spełniające *matching law*, a w środowisku bez współzawodnictwa – spełniające regułę zero-jeden. Co istotne, obie te sytuacje są stabilne ewolucyjnie.

5.15.2 Badanie wczesnych etapów ewolucji systemów nerwowych

Wczesna ewolucja systemów nerwowych grup *Cnidaria* i *Porifera* [ECAL99, str. 236]. W eksperymencie zamodelowano prymitywne organizmy (o najprostszych systemach nerwowych) w prostym środowisku i sprawdzono, jakie wyniki da ich ewolucja ukierunkowana na zdobywanie pożywienia. Celem doświadczenia było sprawdzenie, jakie okoliczności i elementy są wymagane do wykształcenia się (genezy) najprostszego działającego systemu nerwowego. Organizmy miały postać tuby 3D oraz posiadały trzy rodzaje komórek wpływających na zachowanie: nabłonkowe (odbierają pobudzenia i pasywnie je przekazują, mogą kontrolować mięśnie), mięśniowe i nerwowe. Genotyp opisywał ogólny sposób rozwoju połączeń pomiędzy komórkami (*developmental encoding*), a nie konkretne połączenia.

W eksperymentach w różnych środowiskach stwierdzono ciekawe zjawisko: w początkach organizacji systemów nerwowych samo zwiększanie ilości komórek mogło być najprostszą drogą zwiększenia efektywności organizmu. Nie trzeba na tym etapie ewolucji przechowywać w genotypie szczegółów organizacji systemu nerwowego; w rzeczywistości nie miało to miejsca (genomy prostych stworzeń nie mogą opisać konstrukcji i zasad działania złożonych systemów nerwowych).

5.15.3 Badanie powstawania specyficznych struktur w systemach nerwowych

W podobnym eksperymencie (ewolucja ukierunkowana, RNN) stwierdzono [ECAL99, Str. 246], że w sieciach neuronowych osobników pojawiają się pewne specyficzne struktury, jak:

- neurony sterujące (*command neurons*): sterują częściami sieci neuronowej, włączając/wyłączając jej fragmenty i powodując różne zachowania organizmu,
- komórki miejsca (*place cells*): są to neurony sterujące, które odzwierciedlają mapę (układ) środowiska,
- mechanizmy pamięci krótkoterminowej (*short-term memory*). Jak można wykryć, że rekurencyjna NN posiada takie mechanizmy? Wystarczy porównać jej skuteczność z optymalnym algorytmem bez pamięci: jeśli RNN wypada (dużo) lepiej, to posiada zdolność zapamiętywania.

Można sprowokować powstanie powyższych struktur odpowiednio konstruując środowisko (np. gdy do sukcesu wymagane są różne zachowania, lub jeśli zbyt niska zdolność percepcji środowiska daje się zrównoważyć przechowywaniem wiedzy o wcześniejszych zdarzeniach).

Dzięki neurobiologii i neurologii wiemy, że podobne struktury występują np. w hipokampie mózgu szczura i *Aplysia*.

***

Inspiracja naturą przy konstrukcji bezzałogowych maszyn latających nisko nad ziemią – *UAVs for NOE flight: Unmanned Air Vehicles for Nap-of-the-Earth flight* [ECAL99, str. 334]. Muchy i osy są świetnie przystosowane do takiego lotu (omijanie przeszkód, nawigacja) dzięki złożonym oczom zapewniającym szeroki kąt widzenia. W ich sieciach neuronowych odkryto specyficzne połączenie sygnałów ze zmysłów wzrokowych, bezwładności, i aerodynamicznych. Dlatego przy konstrukcji UAVs, wiedza z zakresu robotyki i aeronautyki wzbogacana jest obecnie wiedzą z zakresu neurobiologii.

5.15.4 Badanie altruizmu i reguły Hamiltona

- badanie ewolucji altruizmu [ECAL99, str. 499]. Zachowanie altruistyczne jest korzystne dla innych kosztem altruisty. Czy takie zachowanie może powstać w wyniku ewolucji, która wspiera osobniki najlepiej przystosowane? Wnioski z teorii gier: altruizm wzajemny (*reciprocal altruism*) może być stabilny, o ile w dłuższym okresie przynosi korzyść altruiście. A co z altruizmem „samodzielnym" (nie-wzajemnym)? Na jakim poziomie działa selekcja: genów, osobników, czy grup?[61]
- altruizm [ECAL99, str. 504] to szczególny przypadek współpracy (*cooperation*). Kiedy współpraca jest opłacalna? Zachowanie niekorzystne dla osobnika, lecz korzystne w wystarczającym stopniu dla jego krewnych, może być przez selekcję wspierane.

Reguła Hamiltona:

$$W^i_{Inc} = W^i_{Ind} + \sum_{j,j\neq i} r_{ij} \cdot W^j_{Ind}$$

Selekcja wspiera nie zachowania/cechy, które mają dodatni efekt w przystosowaniu pojedynczego osobnika i (W^i_{Ind}), ale te, które mają dodatni efekt całościowy, W^i_{Inc}. r_{ij} jest stopniem pokrewieństwa osobnika i i innego osobnika j, na którego przystosowanie wpływa o W^j_{Ind} zachowanie osobnika i. W szczególności, w genotypach haploidalnych i diploidalnych, prawdopodobieństwo znalezienia tego samego genu u rodzeństwa jest równe $\frac{1}{2}$. Zatem zachowanie

[61] Wiele na ten temat pisał Richard Dawkins w książce „Samolubny gen"; do tematu nawiązują też „Ślepy zegarmistrz" i „Wspinaczka na szczyt nieprawdopodobieństwa".

niekorzystne dla osobnika ale o tyle samo korzystne dla dwóch lub więcej osobników z jego rodzeństwa będzie przez selekcję wspierane.

Podobne poglądy prezentuje Richard Dawkins. Teoria ta wyjaśnia też związki z rozproszeniem przestrzennym rozwijających się populacji: im większe rozproszenie (wymieszanie), tym mniej w otoczeniu osobnika i (gdzie znajduje on partnerów do rozmnażania i rodzą się dzieci) osobników o wspólnych z nim genach, a zatem $W^i_{Inc} \approx W^i_{Ind}$ i trudniej o ewolucję kooperacji. Zjawisko wpływu osobników spokrewnionych z pewnym osobnikiem na jego przystosowanie i selekcję nazywa się *kin selection*. Niektórzy naukowcy krytykują wyjaśnienie tego rodzaju przytaczając kontrargumenty formalne i eksperymentalne [ECAL99, str. 504].

5.15.5 Badanie ewolucji sygnałów seksualnych i zasada upośledzenia

Sygnalizowanie jakości osobników i zasada upośledzenia [ECAL99, str. 644]. W naturze organizmy, by móc się rozmnażać, muszą dożyć dojrzałości i (najczęściej) znaleźć partnera. Bardzo wiele gatunków posiada niewyobrażalnie rozwinięte mechanizmy nadawania sygnałów seksualnych, a wiele z nich pochłania zdumiewająco dużo czasu i energii. Darwin uważał, że selekcja atrakcyjności seksualnej wywiera odwrotną presję selekcyjną, niż selekcja najlepszego dopasowania do środowiska. Przykład – ogon pawia: sygnał dla płci przeciwnej, ale również sygnał dla groźnych drapieżników, oraz spory wydatek i utrudnienie dla organizmu. Jak więc powstały mechanizmy sygnalizacji swojej „jakości" płci przeciwnej?

- przypadek skojarzył geny odpowiedzialne za preferencję i za pewną cechę, a potem proces wzmacniania spowodował wyolbrzymienie tego połączenia, albo,
- jest to celowy mechanizm sygnalizacji, który mógł wyewoluować.

Dlaczego zatem gorsze osobniki miałyby uczciwie gorzej się „reklamować"? Jeśli „reklama" nic nie kosztuje, nie ma to uzasadnienia. Jednak jeśli kosztuje, to im lepszy osobnik, tym może sobie pozwolić na większe koszty i nadal utrzymać się przy życiu. Jest to zasada upośledzenia (*handicap principle*). Są trzy sposoby na jej dokładniejsze wytłumaczenie:

- upośledzenie wynikające wyłącznie z epistazy w genach. Osobniki o bardziej rzucających się w oczy sygnałach krócej żyją (uwarunkowane genetycznie).
- upośledzenie warunkowe. Osobniki gorsze (słabsze) nie będą w stanie pozwolić sobie w trakcie budowy organizmu na wytworzenie lepszych sygnałów (np. wielkiego ogona).
- upośledzenie ujawniające prawdę. Osobnikom gorszym środowisko nie pozwoli na przeżycie z bardziej widocznymi sygnałami, nawet jeśli mają je opisane w genach.

Eksperyment ujawnił, że wszystkie te sytuacje mogą zaistnieć i są ewolucyjnie stabilne w różnych środowiskach (opisanych wieloma różnymi parametrami); w każdej sytuacji występowała korelacja pomiędzy intensywnością sygnałów seksualnych i faktyczną jakością osobników.

5.15.6 Analiza społecznego uczenia się

Ewolucja społecznego uczenia się o jedzeniu u szczurów wędrownych [ECAL99, str. 514].

Istnieją trzy źródła strategii postępowania: instynkt, uczenie się, społeczne uczenie się. U szczura powstanie preferencja do nowego rodzaju pożywienia, kiedy poczuje jego zapach w oddechu (i tylko w oddechu) innego szczura. Paradoks: szczury uzyskują preferencję co do nowego jedzenia, niezależnie od tego, czy jest ono trujące (i inny szczur choruje), czy dobre. Wiadomo, że szczury skutecznie rozróżniają osobniki zdrowe od chorych (zapach, zachowanie) – ale w przypadku preferencji co do pożywienia akurat ignorują tak jawne przesłanki. Dlaczego? Inne gatunki (kury, kosy) potrafią nauczyć się zarówno preferencji, jak i awersji do nowego pożywienia oceniając stan sąsiada, który to pożywienie zjadł.

W wyniku symulacji pojawiło się możliwe wytłumaczenie paradoksu u szczurów:

- w środowisku, gdzie jest niewiele pożywienia, zbytnia ostrożność nie opłaca się i rezygnowanie z podejrzanego pożywienia może być nieopłacalne,
- analizowanie przydatności/niebezpieczeństwa pożywienia kosztuje czas i energię (trzeba posiadać wykształcony system sensoryczny i mechanizm decyzyjny),
- zatrute szczury pojawiają się rzadziej od zdrowych (bo umierają), więc presja ewolucyjna, by zwracać na nie uwagę, jest niewielka.

Dodatkowo symulacja ujawniła, że zależnie od różnych parametrów środowiska (stosunek pożywienia dobrego i zatrutego, umieralność, współczynnik błędnego rozpoznania, zdolność do uczenia się i inne) optymalne/stabilne mogą być różne rodzaje strategii. Dlatego być może różne gatunki zwierząt postępują w różny sposób.

5.15.7 Badanie dynamiki ekspresji genów

Badanie dynamiki ekspresji genów w sztucznych genomach [ECAL99, str. 457]. Dla niektórych organizmów znany jest już pełen genom, jednak nie znamy zależności pomiędzy genami. Stąd badania właściwości sztucznych genomów, takich jak cykliczna aktywność genów, sekwencje i sieci regulujące, różnicowanie na różne typy komórek, odporność, rola genów nadmiarowych.

Poszukuje się również zależności pomiędzy genami [ECAL99, str. 467, 472, 477] w prawdziwych genomach (np. u muszki owocowej *Drosophila melanogaster*). W zarodku muszki można zmierzyć aktywność poszczególnych genów wzdłuż osi jaja oraz zmiany ich natężenia w czasie. To pozwala nie tylko na aproksymację rozkładu aktywności genów funkcjami analitycznymi, ale również (dzięki informacji o zmianach w czasie) na odkrycie (potencjalnej) topologii sieci regulacyjnej genów (*gene regulatory network*). Zatem po „objawach" współdziałania genów możemy przewidywać („*reverse engineering*") ich wzajemne zależności. Dopiero taka wiedza umożliwi lepsze poznanie mechanizmów budowy i rozwoju organizmów.

5.15.8 Analiza heurystyk karmienia młodych

Analiza heurystyk ptaków karmiących swoje młode [ECAL99, str. 535]. Wiele młodych oraz niedostatek pożywienia powodują, że rodzice muszą dokonać wyboru alokacji zasobów. Młode różnią się pod wieloma względami i mają różne wymagania. Różne gatunki ptaków posiadają różne „algorytmy" karmienia młodych: łyski karmią w pierwszym rzędzie najmłodsze, gołębie najgłodniejsze, jerzyki największe/najstarsze, kwiczoły – losowo (po równo: egalitaryzm). Dotychczas nie ma jasnego wyjaśnienia i uzasadnienia tych różnic. Symulacja ujawniła, że w różnych warunkach środowiskowych (przede wszystkim chodzi o dostępność pożywienia), różne strategie są optymalne i stabilne ewolucyjnie. W dodatku pewne gatunki ptaków zachowują jedną strategię karmienia w różnych warunkach, podczas gdy inne ją zmieniają (muchołówki pstrokate karmią zwykle najmniejsze dzieci, a największe, kiedy pożywienia brakuje; krogulce dzielą pożywienie po równo, a kiedy go brakuje – karmią największe).

5.15.9 Analiza ewolucji komunikacji i języków

U zwierząt [ECAL99, str. 654, 679, 695] nigdzie nie spotkano języka podobnego do złożonych języków ludzkich (ze składnią i kombinacjami sygnałów). Zwierzęta posiadają proste skojarzenia sygnał-obiekt. Podejrzewa się, że przyczyną jest różny mechanizm zdobywania wiedzy: u ludzi sygnały są powiązane (kojarzone) dodatkowo między sobą, u zwierząt – nie.

W ramach studiów nad ewolucją komunikacji prowadzi się badania początków języka, emergencji wspólnych systemów dźwięków, samo-organizacji leksykonów, początków gramatyki, i przypisywania znaczeń. Język traktowany jest jako twór emergentny, dynamiczny, powstający dzięki uwarunkowaniom środowiska i lokalnym oddziaływaniom osobników. Do badania powstawania, przekazywania, i ewolucji języków wykorzystuje się m.in. grę w nazywanie (*naming game* [ECAL99, str. 720]), polegającą na wybieraniu losowo dwóch agentów i konfrontowaniu ich przekonań co do nazwy określonego obiektu. W ten sposób modeluje się też uczenie (ustalanie) języka.

5.15.10 Robotyka zbiorowa inspirowana biologią

Robotyka zbiorowa inspirowana biologią (*biologically inspired collective robotics*) zajmuje się agentami rozproszonymi [ECAL99, str. 575 i 585] tzn. takimi, które nie są centralnie sterowane, ani nie przekazują swojej wiedzy do centrum, które potem rozpraszałoby ją do wszystkich agentów. Robotyka zbiorowa inspirowana biologią wykorzystuje tzw. zbiorową inteligencję obserwowaną u uspołecznionych insektów, a zatem studiuje się tu przede wszystkim zależności agent-agent i agent-środowisko. Dlaczego agenci *autonomiczni*? Prowadzi to do uzyskania odpornych na zakłócenia, przystosowanych do konkretnego problemu, najczęściej emergentnych zachowań i strategii. W szczególności może to być w pełni rozproszone sterowanie i minimalne zdolności robota (bez możliwości komunikacji, planowania, ciągłej adaptacji). Takie ograniczenia występują np. w mili- i mikrorobotyce (związane z komunikacją, mobilnością, postrzeganiem środowiska, mocą obliczeniową). Podobny model obserwujemy również u wielu zwierząt; tam ma też zwykle miejsce samoorganizacja, nadmiarowość i pośrednia komunikacja przez środowisko (**stygmergia**, ang. *stigmergy*). Dzięki takiej komunikacji wiele gatunków może się porozumiewać podczas pracy (np. termity, mrówki) nie komunikując się ze sobą nawzajem, lecz za pośrednictwem efektów pracy (kształtu budowli, miejsc pozostawienia elementów, itp.). Tworzy to niezwykle skomplikowaną sieć zależności pomiędzy środowiskiem i osobnikami; inteligencja jest rozproszona, a nośnikiem wiedzy staje się chwilowy stan środowiska i osobników.

5.16 Granice poznawalności

Rozważania filozoficzne (Paul Davies[62]): Czy otaczające nas obiekty mogą być procesami obliczeniowymi? [Dav96] Tak, zależnie od doboru warunków początkowych (danych) zachowują się w określony sposób (co odpowiada wynikowi). Np. okres obiegu planety wokół gwiazdy może odpowiadać kolejnym cyfrom liczby π...

Zatem wszystko może być obliczalne (o ile nie ma zmiennych ciągłych), my też. Czy więc jesteśmy symulowani?[63] Czy istnieje sposób, by się o tym przekonać? Nie. Nie możemy nawet stwierdzić, czy czas i przestrzeń są dyskretne (najmniejszy mierzalny kwant czasu: 10^{-26} s).

To, czy życie jest we wszechświecie zjawiskiem powszechnym czy unikatowym, pozostaje otwarte. Jedna z koncepcji rozwoju życia we wszechświecie odwołuje się do mechanizmów ewolucyjnych omówionych na wykładzie, bowiem zakłada działanie doboru naturalnego (a przynajmniej istnienie zróżnicowania) na poziomie światów. W tej koncepcji, w czarnych dziurach następuje silne ugięcie czasoprzestrzeni, tak silne, że może dojść do jej rozerwania

[62] https://pl.wikipedia.org/wiki/Paul_Davies_(fizyk)

[63] Kolejne krótkie opowiadanie SF: http://www.framsticks.com/zagniezdzenie

– powstaje wówczas świat-dziecko, w którym obowiązują inne prawa. Jeśli sprzyjają one powstawaniu gwiazd, tym samym sprzyjają powstaniu planet i życia, a także – kolejnych czarnych dziur! Zatem światy, które dobrze się „rozmnażają", służą też powstaniu życia. Wedle tej koncepcji, życie nie jest zjawiskiem rzadkim, jak to jest w większości innych teorii.

Rozdział 6

Środowisko eksperymentalne – Framsticks (program na lab.)

Prezentacje odpowiadające kolejnym podrozdziałom znajdziesz w http://www.framsticks.com/presentations. Te informacje nie obowiązują na egzaminie.

6.1 Informacje podstawowe

6.2 Wizualizacja

6.3 Fizyka i symulacja

6.4 Model i genetyka

6.5 Definicje eksperymentu

6.6 Pisanie skryptów

6.7 Przykładowe eksperymenty

6.7.1 Porównanie reprezentacji genetycznych

6.7.2 Mierzenie podobieństwa

6.7.3 Ocenianie symetrii

6.7.4 Sterowanie rozmyte

6.7.5 Ewolucja czujników, oko wektorowe i koordynacja wizyjno-ruchowa

6.7.6 Umysły: semantyka sieci neuronowej i reprezentacje

6.7.7 Syntetyczna psychologia ewolucyjna, zaawansowane eksperymenty

6.7.8 Emergencja i samoorganizacja

Bibliografia

[ABC09] Dervis Karaboga i Bahriye Akay. "A comparative study of artificial bee colony algorithm". W: *Applied mathematics and computation* 214.1 (2009), s. 108–132. DOI: 10.1016/j.amc.2009.03.090. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.463.3920&rep=rep1&type=pdf.

[Ada98] Christoph Adami. *Introduction to Artificial Life*. Springer, 1998. ISBN: 9780387946467. URL: https://books.google.pl/books?id=2wouAc-WOnYC.

[AHM99] Andrew Adamatzky, Owen Holland i Chris Melhuish. "Laziness + Sensitivity + Mobility = Structure". W: *European Conference on Artificial Life*. Springer, 1999, s. 432–441.

[AK09] Andrew Adamatzky i Maciej Komosinski, red. *Artificial Life Models in Hardware*. London: Springer, 2009, s. 270. ISBN: 978-1-84882-529-1. DOI: 10.1007/978-1-84882-530-7. URL: http://www.springer.com/978-1-84882-529-1.

[Atm92a] W. Atmar. "On the rules and nature of simulated evolutionary programming". W: *Proc. First Annual Conf. on Evolutionary Programming*. 1992, s. 17–26.

[Atm92b] W. Atmar. "The philosophical errors that plague both evolutionary theory and simulated evolutionary programming". W: *Proc. First Annual Conf. on Evolutionary Programming*. 1992, s. 26–34.

[Bak+19] Illya Bakurov i in. "A regression-like classification system for geometric semantic genetic programming". W: *Proceedings of the 11th International Joint Conference on Computational Intelligence (IJCCI)*. T. 1. 2019, s. 40–48. URL: https://run.unl.pt/bitstream/10362/87064/1/Regression_like_Classification_System_Geometric_Semantic_Genetic.pdf.

[Bed96] Mark A. Bedau. "The nature of life". W: *The philosophy of artificial life* (1996), s. 332–357.

[Ben99] Peter Bentley. *Evolutionary design by computers*. Morgan Kaufmann, 1999.

[Bie+18] Dongyang Bie i in. "Parametric L-systems-based modeling self-reconfiguration of modular robots in obstacle environments". W: *International Journal of Advanced Robotic Systems* 15.1 (2018). URL: https://journals.sagepub.com/doi/full/10.1177/1729881418754477.

[Bou+12] Frédéric Boudon i in. "L-Py: an L-system simulation framework for modeling plant architecture development based on a dynamic language". W: *Frontiers in plant science* 3 (2012). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3362793/.

[BT96] Andreas Bölte i Ulrich Wilhelm Thonemann. "Optimizing simulated annealing schedules with genetic programming". W: *European Journal of Operational Research* 92.2 (1996), s. 402–416. ISSN: 0377-2217. DOI: 10.1016/0377-2217(94)00350-5.

[Bul01] Seth Bullock. "Smooth operator? Understanding and visualising mutation bias". W: *European Conference on Artificial Life*. Springer. 2001, s. 602–612. DOI: 10.1007/3-540-44811-X_68.

[Bul98] Andrzej Buller. *Sztuczny mózg. To już nie fantazje*. Prószyński i S-ka, 1998.

[Bul99] Seth Bullock. "Are artificial mutation biases unnatural?" W: *European Conference on Artificial Life*. Springer. 1999, s. 64–73. DOI: 10.1007/3-540-48304-7_11. URL: https://eprints.soton.ac.uk/261452/1/10.1.1.40.2753.pdf.

[Bur+10] E. K. Burker i in. "A classification of hyper-heuristic approaches". W: *Handbook of Metaheuristics* (2010), s. 449–468.

[CG97] Andrzej Chorążyczewski i Roman Galar. "Wizualizacja dynamiki ewolucji fenotypowej". W: *Proceedings of the 2nd National Conference on Evolutionary Computation and Global Optimization*. Rytro: Oficyna Wydawnicza Politechniki Warszawskiej, wrz. 1997, s. 41–48. URL: https://troja.uksw.edu.pl/pdf/kaeiog/KAEiOG1997.41-48.pdf.

[CM96] Dave Cliff i Geoffrey F. Miller. "Co-evolution of pursuit and evasion II: Simulation methods and results". W: *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*. Red. Pattie Maes i in. MIT Press, 1996, s. 506–515. URL: https://pdfs.semanticscholar.org/a7a4/2a58007d2b059870c28cc9apdf.

[Dav90] Yuval Davidor. "Epistasis variance: Suitability of a representation to genetic algorithms". W: *Complex Systems* 4.4 (1990), s. 369–383. URL: http://www.complex-systems.com/pdf/04-4-1.pdf.

[Dav96] Paul Davies. *Plan Stwórcy. Naukowe podstawy racjonalnej wizji świata*. ZNAK, 1996. URL: https://en.wikipedia.org/wiki/The_Mind_of_God.

[DCG99] Marco Dorigo, Gianni Di Caro i Luca M. Gambardella. "Ant algorithms for discrete optimization". W: *Artificial life* 5.2 (1999), s. 137–172. URL: https://web2.qatar.cmu.edu/~gdicaro/Papers/ArtificialLife-original.pdf.

[DJ75] Kenneth Alan De Jong. "Analysis of the behavior of a class of genetic adaptive systems". Prac. dokt. University of Michigan, 1975. URL: https://deepblue.lib.umich.edu/bitstream/handle/2027.42/4507/bab6360.0001.001.pdf.

[Dom99] Paul Domjan. "Are Romance Novels Really Alive? A Discussion of the Supple Adaptation View of Life". W: *Advances in Artificial Life*. Red. Dario Floreano, Jean-Daniel Nicoud i Francesco Mondada. Springer, 1999, s. 21–25. DOI: 10.1007/3-540-48304-7_6.

[ECAL99] D. Floreano, J. D. Nicoud i F. Mondada. *Advances in artificial life: 5th European Conference, ECAL'99, Lausanne, Switzerland, September 13-17, 1999: proceedings*. Lecture notes in computer science. Springer, 1999. ISBN: 9783540664529. URL: http://books.google.pl/books?id=UTpRAAAAMAAJ.

[ECS89] Larry J. Eshelman, Rich Caruana i J. David Schaffer. "Biases in the crossover landscape". W: *Proceedings of the 3rd International Conference on Genetic Algorithms*. Red. J. David Schaffer. Morgan Kaufmann, 1989, s. 10–19.

[EE17] Gustavo Santarsiere Etchebehere i Maria Amelia Eliseo. "L-Systems and Procedural Generation of Virtual Game Maze Sceneries". W: *Proceedings of Brazilian Symposium on Computer Games and Digital Entertainment*. 2017, s. 602–605. URL: https://www.sbgames.org/sbgames2017/papers/ComputacaoShort/174978.pdf.

[Elf+21] Ehab Z. Elfeky i in. "A systematic review of coevolution in real-time strategy games". W: *IEEE Access* 9 (2021), s. 136647–136665. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9548932.

[ENS99] Ewa Niewiadomska-Szynkiewicz. "Przegląd metod optymalizacji globalnej". W: *Proceedings of the 3rd National Conference on Evolutionary Computation and Global Optimization*. Potok Złoty: Oficyna Wydawnicza Politechniki Warszawskiej, 1999, s. 363–371. URL: https://troja.uksw.edu.pl/pdf/kaeiog/KAEiOG1999.363-371.pdf.

[FB90] J. Doyne Farmer i Alletta Belin. *Artificial life: The coming evolution*. Spraw. tech. SFI working paper 1990–003. Santa Fe Institute, 1990. URL: https://www.santafe.edu/research/results/working-papers/artificial-life-the-coming-evolution.

[Fog00] David B. Fogel. *Evolutionary computation: toward a new philosophy of machine intelligence*. 2 wyd. IEEE Press, 2000.

[Fra72] Daniel Raymond Frantz. "Non-linearities in genetic adaptive search". Prac. dokt. University of Michigan, 1972. URL: https://deepblue.lib.umich.edu/bitstream/handle/2027.42/4950/bac2356.0001.001.pdf.

[Gaj+19] Alexander Gajewski i in. "Evolvability ES: scalable and direct optimization of evolvability". W: *Proceedings of the Genetic and Evolutionary Computation Conference*. 2019, s. 107–115. URL: https://arxiv.org/pdf/1907.06077.pdf.

[Gea+02] Nicholas Geard i in. "A comparison of neutral landscapes – NK, NKp and NKq". W: *Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02*. T. 1. IEEE. 2002, s. 205–210. URL: https://eprints.soton.ac.uk/264208/1/cec1-comp.pdf.

[Gol+93] David E. Goldberg i in. *Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms*. Spraw. tech. 93004. 1993, s. 1–16. URL: http://repository.ias.ac.in/81677/1/110-a.pdf.

[Gol03] David Edward Goldberg. *Algorytmy genetyczne i ich zastosowania*. Wydawnictwa Naukowo-Techniczne, 2003.

[GT12] Brian W. Goldman i Daniel R. Tauritz. "Linkage tree genetic algorithms: variants and analysis". W: *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. 2012, s. 625–632. URL: http://www.cmap.polytechnique.fr/~nikolaus.hansen/proceedings/2012/GECCO/proceedings/p625.pdf.

[Gut94] Marek W. Gutowski. "Smooth genetic algorithm". W: *Journal of Physics A: Mathematical and General* 27.23 (1994), s. 7893.

[Gut97] Marek Gutowski. "Zdolność rozdzielcza gładkiego algorytmu ewolucyjnego". W: *Proceedings of the 2nd National Conference on Evolutionary Computation and Global Optimization*. Rytro: Oficyna Wydawnicza Politechniki Warszawskiej, wrz. 1997, s. 73–79.

[Gwi07a] Tomasz Dominik Gwiazda. *Algorytmy genetyczne – kompendium. Tom I. Operator krzyżowania dla problemów numerycznych*. PWN, 2007.

[Gwi07b] Tomasz Dominik Gwiazda. *Algorytmy genetyczne – kompendium. Tom II. Operator mutacji dla problemów numerycznych*. PWN, 2007.

[HL06] Marcus Hutter i Shane Legg. "Fitness uniform optimization". W: *IEEE Transactions on Evolutionary Computation* 10.5 (2006), s. 568–589. URL: https://arxiv.org/pdf/cs/0610126.pdf.

[Hor03] Gregory S. Hornby. "Creating complex building blocks through generative representations". W: *Proceedings of the 2003 AAAI Spring Symposium: Computational Synthesis: From Basic Building Blocks to High Level Functionality*. 2003, s. 98–105. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.323.8779&rep=rep1&type=pdf.

[Hor08] Gregory S. Hornby. "Improving the scalability of generative representations for openended design". W: *Genetic Programming Theory and Practice V* (2008), s. 125–142.

[HR78] John H. Holland i Judith S. Reitman. "Cognitive systems based on adaptive algorithms". W: *Pattern-directed inference systems*. Elsevier, 1978, s. 313–329.

[Hu+05] Jianjun Hu i in. "The Hierarchical Fair Competition (HFC) framework for sustainable evolutionary algorithms". W: *Evolutionary Computation* 13.2 (2005), s. 241–277. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=9a9418ae55bdff2f1a988effe8554e8ec0050df0.

[Hut09] Tim J. Hutton. "The organic builder: A public experiment in artificial chemistries and self-replication". W: *Artificial life* 15.1 (2009), s. 21–28. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.1690&rep=rep1&type=pdf.

[JMK20] Anders Jerkstrand, Keiichi Maeda i Koji S. Kawabata. "A type Ia supernova at the heart of superluminous transient SN 2006gy". W: *Science* 367.6476 (2020), s. 415–418. DOI: 10.1126/science.aaw1469.

[JPM00] Catherine Jirasek, Przemyslaw Prusinkiewicz i Bruno Moulia. "Integrating biomechanics into developmental plant models expressed using L-systems". W: *Plant biomechanics* (2000), s. 615–624. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.364.585&rep=rep1&type=pdf.

[JTW04] E. D. de Jong, D. Thierens i R. A. Watson. "Hierarchical genetic algorithms". W: *Lecture notes in computer science* (2004), s. 232–241. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=20bde9fe796a5e15c9007c7dc770b35aa731751d.

[KA09] Maciej Komosinski i Andrew Adamatzky, red. *Artificial Life Models in Software*. 2nd. London: Springer, 2009, s. 442. ISBN: 978-1-84882-284-9. DOI: 10.1007/978-1-84882-285-6. URL: http://www.springer.com/978-1-84882-284-9.

[KAEiOG97] *Proceedings of the 2nd National Conference on Evolutionary Computation and Global Optimization*. Rytro: Oficyna Wydawnicza Politechniki Warszawskiej, wrz. 1997.

[KAEiOG99] *Proceedings of the 3rd National Conference on Evolutionary Computation and Global Optimization*. Potok Złoty: Oficyna Wydawnicza Politechniki Warszawskiej, maj 1999.

[KC06] Kyung-Joong Kim i Sung-Bae Cho. "A comprehensive overview of the applications of artificial life". W: *Artificial Life* 12.1 (2006), s. 153–182.

[KKM21] Piotr Kaszuba, Maciej Komosinski i Agnieszka Mensfelt. "Automated development of latent representations for optimization of sequences using autoencoders". W: *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2021, s. 1123–1130. DOI: 10.1109/CEC45853.2021.9504910. URL: http://www.framsticks.com/files/common/LatentRepresentationsForSequencesOptimization.pdf.

[KM18] Maciej Komosinski i Konrad Miazga. "Comparison of the tournament-based convection selection with the island model in evolutionary algorithms". W: *Journal of Computational Science* 32 (2018), s. 106–114. ISSN: 1877-7503. DOI: 10.1016/j.jocs.2018.10.001. URL: http://www.framsticks.com/files/common/ConvectionSelectionVsIslandModel.pdf.

[Knu09] Johan Knutzen. "Generating climbing plants using L-systems". Prac. mag. 2009. URL: http://www.cse.chalmers.se/~uffe/xjobb/climbingplants.pdf.

[Kom19] Maciej Komosinski. *Człowieczeństw0*. 2019. URL: http://www.mooncoder.com/czlowieczenstw

[KR01] Maciej Komosinski i Adam Rotaru-Varga. "Comparison of different genotype encodings for simulated 3D agents". W: *Artificial Life Journal* 7.4 (Fall 2001), s. 395–418. DOI: 10.1162/106454601317297022. URL: http://www.framsticks.com/files/common/ComparisonGeneticEncodings3DAgents.pdf.

[KT20] Iuliia Kotseruba i John K. Tsotsos. "40 years of cognitive architectures: core cognitive abilities and practical applications". W: *Artificial Intelligence Review* 53.1 (2020), s. 17–94. DOI: 10.1007/s10462-018-9646-y.

[KU17] Maciej Komosinski i Szymon Ulatowski. "Multithreaded computing in evolutionary design and in artificial life simulations". W: *The Journal of Supercomputing* 73.5 (2017), s. 2214–2228. ISSN: 1573-0484. DOI: 10.1007/s11227-016-1923-4. URL: http://www.framsticks.com/files/common/MultithreadedEvolutionaryDesign.pdf.

[KU21] Maciej Komosinski i Szymon Ulatowski. *Genetic representations in Framsticks*. http://www.framsticks.com/a/al_genotype. 2021.

[KU24] Maciej Komosinski i Szymon Ulatowski. *Framsticks website*. 2024. URL: http://www.framsticks.com.

[Kub04] Marek Kubiak. "Systematic construction of recombination operators for the vehicle routing problem". W: *Foundations of Computing and Decision Sciences* 29.3 (2004). URL: http://www.cs.put.poznan.pl/mkubiak/publications/Kubiak_SystematicConstruction_FCDS2004.pdf.

[Kun05] Daniel Kunkle. *A summary and comparison of MOEA algorithms*. Spraw. tech. 2005. URL: https://www.ccs.neu.edu/home/kunkle/papers/techreports/moeaComparison.pdf.

[Küp91] Bernd-Olaf Küppers. *Geneza informacji biologicznej. Filozoficzne problemy powstania życia*. http://bazhum.muzhp.pl/media//files/Studia_Philosophiae_Christianae/Studia_Philosophiae_Christianae-r1992-t28-n1/Studia_Philosophiae_Christianae-r1992-t28-n1-s183-186/Studia_Philosophiae_Christianae-r1992-t28-n1-s183-186.pdf. PWN, 1991. ISBN: 9788301102715. URL: http://www.obi.opoka.org/zfn/015/zfn01514Wolak.pdf.

[Kwa97] Halina Kwaśnicka. "Nadmiarowość genetyczna, poligeniczność i plejotropowość w algorytmach ewolucyjnych". W: *Proceedings of the 2nd National Conference on Evolutionary Computation and Global Optimization*. Rytro: Oficyna Wydawnicza Politechniki Warszawskiej, wrz. 1997, s. 137–144.

[KŻ20] Maciej Komosinski i Tomasz Żok. "Morality, protection, security and gain: lessons from a minimalistic, economically inspired multi-agent model". W: *Foundations of Computing and Decision Sciences* 45.1 (2020), s. 17–33. DOI: 10.2478/fcds-2020-0002.

[Lan97] Christopher G. Langton. *Artificial life: An overview*. MIT Press, 1997.

[LH05] Shane Legg i Marcus Hutter. "Fitness uniform deletion: A simple way to preserve diversity". W: *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. 2005, s. 1271–1278. URL: https://arxiv.org/pdf/cs/0504035.pdf.

[Life10] Jean Gayon i in., red. *Defining life*. T. 40. Origins of Life and Evolution of Biospheres 2. Springer, 2010, s. 119–244. URL: https://cache.media.eduscol.education.fr/file/Formation_continue_enseignants/52/2/Jean_Gayon_2_292522.pdf.

[Mah92] Samir W. Mahfoud. "Crowding and preselection revisited". W: *Parallel problem solving from nature*. Red. R. Männer i B. Manderick. T. 2. Elsevier, 1992, s. 27–36. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.52.3943&rep=rep1&type=pdf.

[MC15] Jean-Baptiste Mouret i Jeff Clune. "Illuminating search spaces by mapping elites". W: *arXiv preprint arXiv:1504.04909* (2015). URL: https://arxiv.org/pdf/1504.04909.pdf.

[Mer97] Ralph C. Merkle. "It's a small, small, small, small world". W: *MIT Technology Review* (1997). URL: https://www.researchgate.net/profile/Ralph_Merkle/publication/228378235_It's_a_small_small_small_small_world/links/0a85e53a224a7d10bd000000.pdf.

[Mic09] Thomas Miconi. "Why coevolution doesn't "work": superiority and progress in coevolution". W: *12th European Conference on Genetic Programming – EuroGP*. Springer, 2009, s. 49–60. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=c51f7914019c020d2386b4880cdc59b22dbbd7c4.

[Mic96] Z. Michalewicz. *Algorytmy genetyczne + struktury danych = programy ewolucyjne*. Wydawnictwa Naukowo-Techniczne, 1996.

[MKT99] Soraia Raupp Musse, Marcelo Kallmann i Daniel Thalmann. "Level of autonomy for virtual human agents". W: *European Conference on Artificial Life*. Springer, 1999, s. 345–349.

[NK00] B. Naudts i L. Kallel. "A comparison of predictive measures of problem difficulty in evolutionary algorithms". W: *IEEE Transactions on Evolutionary Computation* 4.1 (2000), s. 1–15. DOI: 10.1109/4235.843491.

[NP99] Stefano Nolfi i Domenico Parisi. "Exploiting the power of sensory-motor coordination". W: *European Conference on Artificial Life*. Springer, 1999, s. 173–182.

[ÖBK08] E. Özcan, B. Bilgin i E. E. Korkmaz. "A comprehensive analysis of hyper-heuristics". W: *Intelligent Data Analysis* 12.1 (2008), s. 3–23.

[OG03] Mihai Oltean i Crina Groşan. "Evolving evolutionary algorithms using multi expression programming". W: *European Conference on Artificial Life*. Springer. 2003, s. 651–658. URL: https://www.researchgate.net/profile/Mihai_Oltean2/publication/226167912_Evolving_Evolutionary_Algorithms_Using_Multi_Expression_Programming/links/55dac32308aed6a199aaf916.pdf.

[Olt05] Mihai Oltean. "Evolving evolutionary algorithms using linear genetic programming". W: *Evolutionary Computation* 13.3 (2005), s. 387–410. URL: https://mihaioltean.github.io/oltean_mit_draft_2005.pdf.

[Pas00] Andrzej Pastusiak. "Ewoluujące automaty komórkowe jako metoda wzrostu sztucznych sieci neuronowych". Prac. mag. 2000. URL: http://docus.no-ip.com/codi/mgr/mgr.htm.

[Pău00] Gheorghe Păun. "Computing with membranes". W: *Journal of Computer and System Sciences* 61.1 (2000), s. 108–143. URL: https://www.sciencedirect.com/science/article/pii/S0022000099916938.

[PD00] Mitchell A. Potter i Kenneth A. De Jong. "Cooperative coevolution: an architecture for evolving coadapted subcomponents". W: *Evolutionary computation* 8.1 (mar. 2000), s. 1–29. DOI: 10.1162/106365600568086. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.35.5861&rep=rep1&type=pdf.

[PKF21] Michal W. Przewozniczek, Marcin M. Komarnicki i Bartosz Frej. "Direct linkage discovery with empirical linkage learning". W: *Proceedings of the Genetic and Evolutionary Computation Conference*. 2021, s. 609–617. URL: https://www.cs.put.poznan.pl/mkomosinski/lectures/optimization/extras/DLED-epistasis-GECCO2021.pdf.

[PL96] Przemyslaw Prusinkiewicz i Aristid Lindenmayer. *The algorithmic beauty of plants*. Springer, 1996. URL: http://algorithmicbotany.org/papers/abop/abop.pdf.

[PM01] Yoav I. H. Parish i Pascal Müller. "Procedural modeling of cities". W: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 2001, s. 301–308. URL: https://dl.acm.org/doi/abs/10.1145/383259.383292.

[Pod97] Mariusz Podsiadło. "Równoległe algorytmy genetyczne – przegląd problematyki". W: *Proceedings of the 2nd National Conference on Evolutionary Computation and Global Optimization*. Rytro: Oficyna Wydawnicza Politechniki Warszawskiej, wrz. 1997, s. 219–227.

[PTK23] Michal W. Przewozniczek, Renato Tinós i Marcin M. Komarnicki. "First Improvement Hill Climber with Linkage Learning – on Introducing Dark Gray-Box Optimization into Statistical Linkage Learning Genetic Algorithms". W: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '23*. ACM, 2023, s. 946–954. DOI: 10.1145/3583131.3590495.

[QD99] Tom Quick i Kerstin Dautenhahn. "Making embodiment measurable". W: *Proceedings of '4. Fachtagung der Gesellschaft für Kognitionswissenschaft' (KogWis'99)*. 1999. URL: http://supergoodtech.com/tomquick/phd/kogwis/webtext.html.

[Ray55] Nicholas Ray. *Rebel Without a Cause (film)*. 1955.

[Ray92] Thomas S. Ray. *Evolution, ecology and optimization of digital organisms*. Spraw. tech. Technical Report 92-08-042, Santa Fe Institute, Santa Fe, NM, 1992. URL: https://pdfs.semanticscholar.org/66f6/de851f131cc78a3279623a96b086e3150483.pdf.

[RB95] Christopher Rosin i Richard Belew. "Methods for competitive co-evolution: finding opponents worth beating". W: *Proceedings of the Sixth International Conference on Genetic Algorithms*. Morgan Kaufmann, 1995, s. 373–380. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.52.9359&rep=rep1&type=pdf.

[RB99] Andreas Rechtsteiner i Mark A. Bedau. "A generic neutral model for quantitative comparison of genotypic evolutionary activity". W: *European Conference on Artificial Life*. Springer, 1999, s. 109–118.

[Rea+19] Esteban Real i in. "Regularized evolution for image classifier architecture search". W: *Proceedings of the AAAI conference on artificial intelligence*. T. 33. 1. 2019, s. 4780–4789. URL: https://ojs.aaai.org/index.php/AAAI/article/download/4405/4283.

[Rea+20] Esteban Real i in. "AutoML-Zero: Evolving machine learning algorithms from scratch". W: *International Conference on Machine Learning*. PMLR. 2020, s. 8007–8019. URL: https://proceedings.mlr.press/v119/real20a/real20a.pdf.

[Rec84] Ingo Rechenberg. "The Evolution Strategy. A Mathematical Model of Darwinian Evolution". W: *Synergetics – From Microscopic to Macroscopic Order*. Red. Eckart Frehland. Berlin, Heidelberg: Springer Berlin Heidelberg, 1984, s. 122–132. DOI: 10.1007/978-3-642-69540-7_13.

[Ros05] P. Ross. "Hyper-heuristics". W: *Search Methodologies* (2005), s. 529–556.

[Rot06] Franz Rothlauf. *Representations for genetic and evolutionary algorithms*. Springer, 2006. DOI: 10.1007/3-540-32444-5.

[RW95] Colin R. Reeves i Christine C. Wright. "Epistasis in Genetic Algorithms: An Experimental Design Perspective". W: *ICGA*. 1995, s. 217–224. URL: https://www.researchgate.net/profile/Christine_Wright5/publication/2504851_Epistasis_in_Genetic_Algorithms_An_Experimental_Design_Perspective/links/0912f50bf2d3d75799000000.pdf.

[SA04] Reiji Suzuki i Takaya Arita. "Interactions between learning and evolution: The outstanding strategy generated by the Baldwin effect". W: *Biosystems* 77.1-3 (2004), s. 57–71. DOI: 10.1016/j.biosystems.2004.04.002.

[SB06] Christine Solnon i Derek Bridge. "An ant colony optimization meta-heuristic for subset selection problems". W: *System engineering using particle swarm optimization* (2006), s. 7–29. URL: https://liris.cnrs.fr/Documents/Liris-2279.pdf.

[Sch44] Erwin Schrödinger. *What is life? The physical aspect of the living cell and mind*. https://en.wikipedia.org/wiki/What_Is_Life%3F. Cambridge University Press, 1944. URL: http://old.biovip.com/UpLoadFiles/Aaron/Files/2005051204.pdf.

[Sim94] Karl Sims. "Evolving virtual creatures". W: *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM. 1994, s. 15–22. URL: https://www.cs.drexel.edu/~david/Classes/Papers/p15-sims.pdf.

[Sip95] Moshe Sipper. "An introduction to artificial life". W: *Explorations in Artificial Life (special issue of AI Expert)* (1995), s. 4–8.

[SP97] Rainer Storn i Kenneth Price. "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces". W: *Journal of Global Optimization* 11.4 (1997), s. 341–359. ISSN: 1573-2916. DOI: 10.1023/A:1008202821328.

[SPL06] Francisco C. Santos, Jorge M. Pacheco i Tom Lenaerts. "Cooperation prevails when individuals adjust their social ties". W: *PLoS computational biology* 2.10 (2006), e140. DOI: 10.1371/journal.pcbi.0020140. URL: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.0020140.

[SW15] Boris Shabash i Kay C. Wiese. "Diploidy in evolutionary algorithms for dynamic optimization problems: A best-chromosome-wins dominance mechanism". W: *International Journal of Intelligent Computing and Cybernetics* 8.4 (2015), s. 312–329. URL: https://www.researchgate.net/profile/Kay-Wiese/publication/283648490_Diploidy_in_evolutionary_algorithms_for_dynamic_optimization_problems/links/5eea7a3d92851ce9Diploidy-in-evolutionary-algorithms-for-dynamic-optimization-problems.pdf.

[Tay99] Tim Taylor. "On self-reproduction and evolvability". W: *European Conference on Artificial Life*. Springer, 1999, s. 94–103.

[TB13] Dirk Thierens i Peter A. N. Bosman. "Hierarchical problem solving with the linkage tree genetic algorithm". W: *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. 2013, s. 877–884. URL: https://homepages.cwi.nl/~bosman/publications/2013_hierarchicalproblemsolving.pdf.

[Tha+95] Daniel Thalmann i in. "Virtual and real humans interacting in the virtual world". W: *Proc. International Conference on Virtual Systems and Multimedia95*. 1995, s. 48–57. URL: https://infoscience.epfl.ch/record/102037/files/Thalmann_and_al_VSMM_95.pdf.

[Thi18] Dirk Thierens. *Model-Based Evolutionary Algorithms, Part 2: Linkage Tree Genetic Algorithm*. 2018. URL: https://ics-websites.science.uu.nl/docs/vakken/ea/slides/LTGA_GOMEA.pdf.

[TTG94] Demetri Terzopoulos, Xiaoyuan Tu i Radek Grzeszczuk. "Artificial fishes: Autonomous locomotion, perception, behavior, and learning in a simulated physical world". W: *Artificial Life* 1.4 (1994), s. 327–351. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.33.8131&rep=rep1&type=pdf.

[Tur02] Peter D. Turney. "Myths and Legends of the Baldwin Effect". W: *CoRR* cs.LG/0212036 (2002). URL: http://arxiv.org/abs/cs.LG/0212036.

[TYH99] Shigeyoshi Tsutsui, Masayuki Yamamura i Takahide Higuchi. "Multi-parent recombination with simplex crossover in real coded genetic algorithms". W: *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation – Volume 1*. 1999, s. 657–664. URL: https://www.researchgate.net/profile/Shigeyoshi-Tsutsui/publication/243776468_Multi-parent_recombination_with_simplex_crossover_in_real-coded_genetic_algorithms/links/00463531fa92bd4738000000/Multi-parent-recombination-with-simplex-crossover-in-real-coded-genetic-algorithms.pdf.

[Wąs97] Piotr Wąsiewicz. "Self-programming of Algorithms". W: *Proceedings of the 2nd National Conference on Evolutionary Computation and Global Optimization*. Rytro: Oficyna Wydawnicza Politechniki Warszawskiej, wrz. 1997, s. 293–297.

[Wei09] Thomas Weise. *Global Optimization Algorithms – Theory and Application*. Second. Self-published, 2009. URL: http://www.it-weise.de/projects/book.pdf.

[WP99] Richard A. Watson i Jordan B. Pollack. "How symbiosis can guide evolution". W: *European Conference on Artificial Life*. Springer, 1999, s. 29–38.

[WS05] Peter Worth i Susan Stepney. "Growing music: musical interpretations of L-systems". W: *Workshops on Applications of Evolutionary Computation*. Springer. 2005, s. 545–550. URL: https://www-users.cs.york.ac.uk/susan/bib/ss/nonstd/eurogp05.pdf.

[Yin+19] Chris Ying i in. "NAS-Bench-101: Towards reproducible neural architecture search". W: *International Conference on Machine Learning*. PMLR. 2019, s. 7105–7114. URL: https://proceedings.mlr.press/v97/ying19a/ying19a.pdf.

[YST99] Yuka Yamamoto, Takahiro Sasaki i Mario Tokoro. "Adaptability of Darwinian and Lamarckian populations toward an unknown new world". W: *European Conference on Artificial Life*. Springer, 1999, s. 39–48.

Cytowanie tego skryptu:

```
@booklet{MK-MiOIBskrypt,
  title  = {Metaheurystyki, Obliczenia Inspirowane Biologicznie i Sztuczne Życie},
  author = {Maciej Komosiński},
  year   = {2025},
  note   = {Skrypt do zajęć},
  url    = {http://www.cs.put.poznan.pl/mkomosinski/lectures/mioib/MK\_MiOIB.pdf}
}
```