

Sztuczna Inteligencja

3.3 Szukanie heurystyczne II

Włodzisław Duch
Katedra Informatyki Stosowanej UMK
Google: Włodzisław Duch

B&B, rozgałęziaj i ograniczaj



Wybierz na każdym kroku m najlepszych węzłów,
rozwijaj dopóki wystarczy pamięci (beam search, szukanie wiązki).
Utwórz listę składającą się z korzenia

Dopóki lista nie jest pusta lub nie osiągnięto węzła celu:

- Znajdź m węzłów dla których $f(n_m)$ = minimalne

- rozwiń je tworząc wszystkie węzły potomne

- jeśli któryś z nich jest celem zakończ

- dla każdego węzła potomnego n' :

 - oceń węzeł funkcją $f(n')$

 - usuń powtórki

Wybierz m najlepszych węzłów.

Algorytmy minimalizacyjne

- Meta-heurystyki na przeszukiwanie globalne całej przestrzeni.
- Najłatwiej je stosować gdy każdy stan można ocenić niezależnie od drogi dojścia do tego stanu.
- Mogą być kosztowne ale dla dużych przestrzeni poszukiwań pełne przeszukiwanie jest znacznie bardziej kosztowne.
- Mogą nie być optymalne ale zwykle znajdują niezłe rozwiązania.
- Oparte są na metodach globalnej minimalizacji funkcji (heurystycznej) $f(s)$, gdzie s to zbiór parametrów.

Jest wiele metod globalnej minimalizacji, np. opisanych tu:

Duch, Korczak: [Optimization and global minimization methods suitable for neural networks.](#)

Algorytm wspinaczki

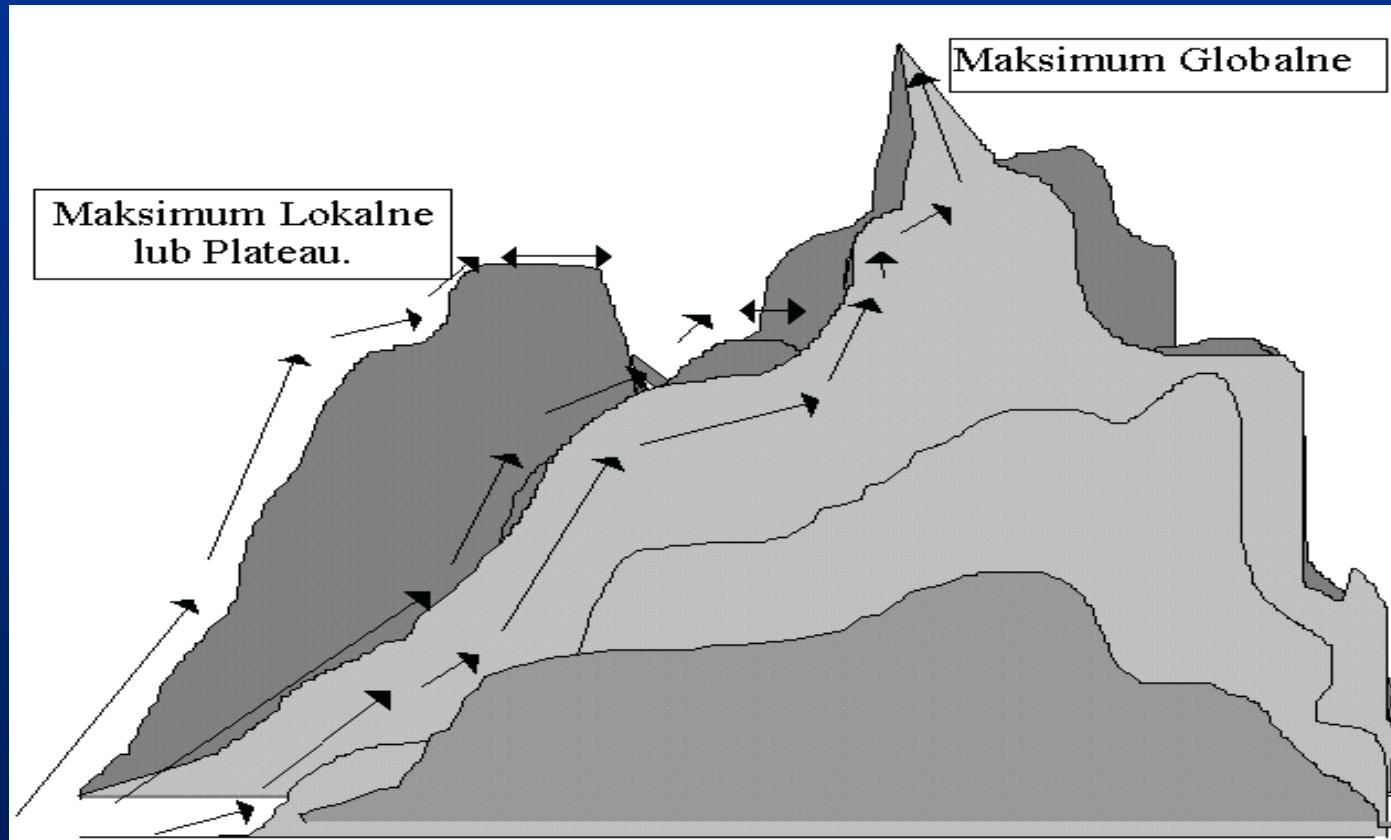
- Hill-climbing algorithm, czyli algorytm wspinaczki: idź w kierunku wzrastającej wartości funkcji heurystycznej.
- Równoważny szukaniu „najpierw najlepszy”, ale tu zastosowany do optymalizacji parametrów ciągłych, rozwija i pamięta tylko jedną ścieżkę.
- Przełamuj impasy (identyczne koszty kilku dróg) w sposób przypadkowy.

Oceń stan początkowy; jeśli nie jest to cel wykonaj:

Utwórz nowy stan;

- Jeśli nowy stan jest stanem celu zakończ.
- Jeśli nowy stan jest bliżej stanu celu przyjmij go; w przeciwnym przypadku zignoruj go.
- Jeśli nie można utworzyć nowych stanów zatrzymaj się.

[Ilustracja w Java](#)



Funkcja heurystyczna może przyjmować wartości maksymalne (np. globalne zyski przy optymalizacji cen produktów), lub możemy ją minimalizować (np. straty).

Algorytm wspinaczki - wady

- Lokalne minima.
- Plateaux, czyli równiny na których nie ma kierunku wzrostu.
- Wąskie grzbiety (ridges).

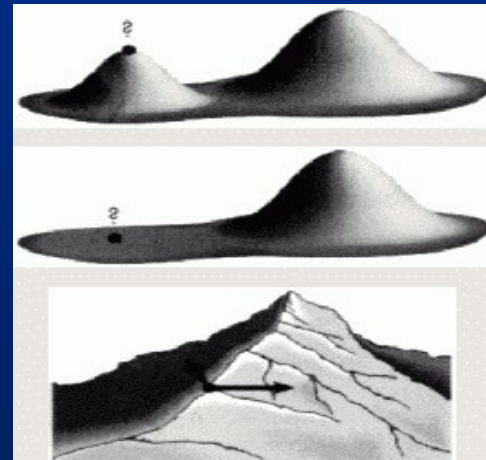
Wiele wariantów:

Generuj liczne przypadkowe ruchy, wybierz najlepszy.

Zastosuj wielokrotne starty z przypadkowych punktów początkowych.

Szukaj wiązkę.

Dopuszczaj gorsze wartości funkcji przez ograniczoną liczbę kroków.

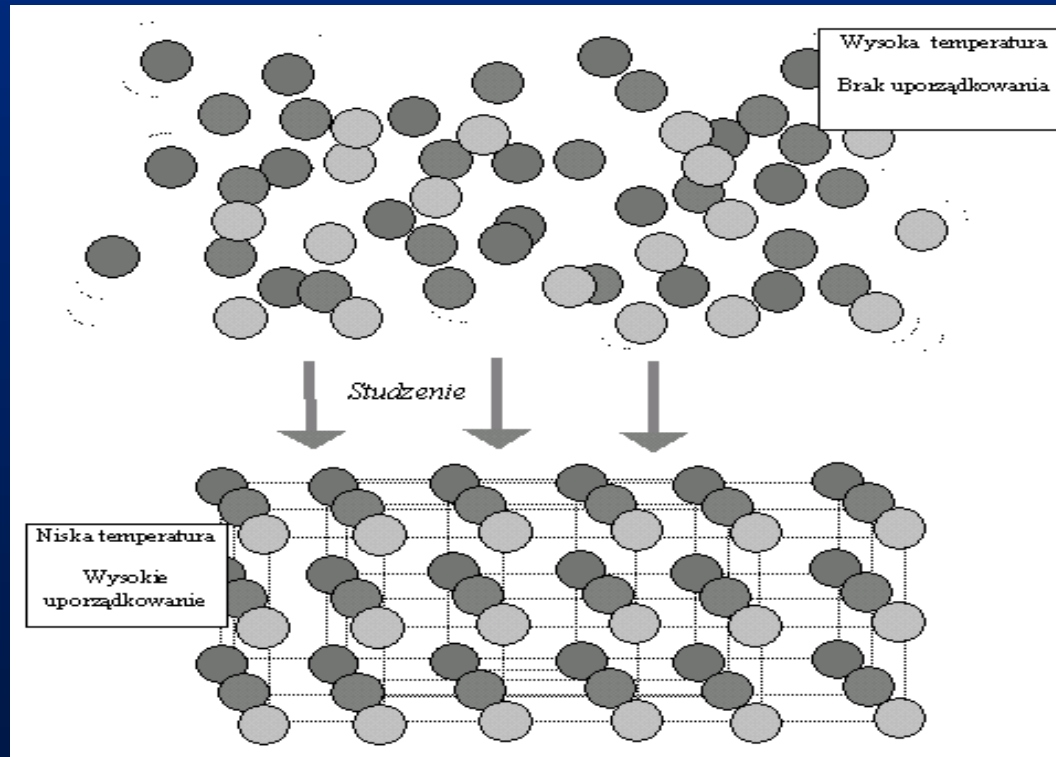


Monte Carlo i Symulowane Wyżarzanie

- Szukanie przypadkowe, nazywane algorytmem British Museum, wybiera przypadkowo węzły.
- W metodzie Monte Carlo (MC) rozwija się przypadkowo wybrany węzeł. Odpowiada to przypadkowemu tworzeniu różnych fragmentów grafu.
- Jeśli mamy funkcję heurystyczną to można zastosować metodę symulowanego wyżarzania (SA).

Symulowane Wyżarzanie

- Powoli studzone kryształy osiągają konfiguracje o minimalnej energii, ruchy termiczne są coraz słabsze => to wskazówka by w MC stosować operatory dające stopniowo malejące zmiany - coraz mniejsze skoki parametrów wokół minimum.



Symulowane Wyżarzanie

- Cel: szukanie globalnie optymalnych rozwiązań.
- Trzeba czasami iść pod górkę, nie wystarczą algorytmy zachłanne ...
- Akceptuj nowe węzły nawet jeśli ich ocena jest gorsza z prawdopodobieństwem:

$$p(\Delta E) \sim e^{-\Delta E / T}$$

gdzie $\Delta E = f(O) - f(O')$ to ocena zmiany jakości rozwiązania w wyniku zmiany stanu z O na O' . Pamiętaj najlepsze dotychczasowe wyniki.

Jest to rozkład energii Boltzmana w układach termodynamicznych
- właśnie taki rozkład energii znajdujemy w przyrodzie.

Parametr T pełni rolę temperatury. Duże T - wszystkie zmiany są akceptowalne, czyli to Monte Carlo; małe T - przyjmujemy zmiany tylko jeśli dają poprawę.

ASA = Adaptive Simulated Annealing, program do optymalizacji metodą SA z możliwością regulacji T niezależnie dla wielu parametrów.

Algorytm SA

- Określ parametry problemu, koszty operacji itp.
- Zdefiniuj funkcję energii (heurystyczną).
- Wybierz początkową temperaturę i schemat chłodzenia.
Np. $T=100$ i co 100 iteracji będziemy obniżać o 10.
- Utwórz przypadkowo początkowy węzeł S (np. jakieś nieoptymalne rozwiązanie) lub użyj dany węzeł.
- Twórz przypadkowo nowe węzły modyfikując lub rozwijając istniejące - duże zmiany są początkowo dopuszczalne.
- Obniżaj temperaturę: początkowo akceptowane są wszystkie zmiany, potem tylko drobne zmiany korzystne.

Dobrym zastosowaniem takich algorytmów są problemy optymalizacyjne typu TSP, problemu komiwojażera.

Podróż dookoła świata.

Przypadkowa permutacja miast, zaczynając od startu:

$S_1 = [\text{Wwa}, \text{Dub}, \text{Bei}, \text{Tok}, \text{Los}, \text{NY}, \text{Sai}, \text{Par}, \text{Chi}, \text{Bom}, \text{Cai}]$

$T = 100$, Koszt = 370

Kolejne przypadkowe rozwiązanie ($T = 90$; Koszt: 340)

$S_2 = [\text{Wwa}, \text{Los}, \text{Bei}, \text{Tok}, \text{Dub}, \text{NY}, \text{Sai}, \text{Par}, \text{Chi}, \text{Bom}, \text{Cai}]$

(zamiana Dublin : Los Angeles)

Kolejne przypadkowe rozwiązanie ($T = 90$; Koszt: 377)

$S_3 = [\text{Wwa}, \text{Los}, \text{Bei}, \text{Tok}, \text{Dub}, \text{NY}, \text{Par}, \text{Chi}, \text{Sai}, \text{Bom}, \text{Cai}]$

(zamiana Saigon : Chicago, przyjęta chociaż gorzej)

.....

Kolejne przypadkowe rozwiązanie ($T = 10$; Koszt: 188)

(zamiana tylko jeśli koszt się obniża).

- Wady: wiele iteracji.

Podróż dookoła świata.

Przypadkowa permutacja miast, zaczynając od startu:

$S_1 = [\text{Wwa, Dub, Bei, Tok, Los, NY, Sai, Par, Chi, Bom, Cai}]$

$T = 100$, Koszt = 370

Kolejne przypadkowe rozwiązanie ($T = 90$, Koszt = 240)

$S_2 = [\text{Wwa, Los, Bei, Tok, Dub}]$

(zamiana Dublin : Los Angeles)

Kolejne przypadkowe rozwiązanie

$S_3 = [\text{Wwa, Los, Bei, Tok, Dub}]$

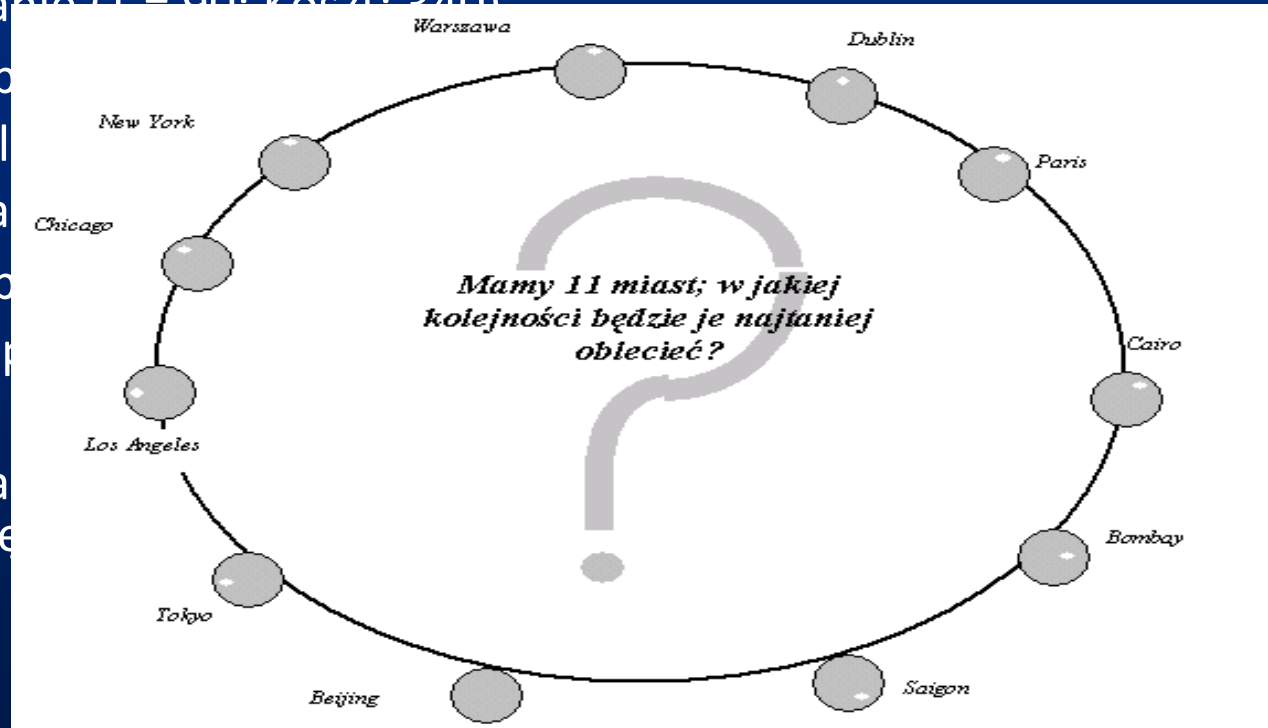
(zamiana Saigon : Chicago, p)

.....

Kolejne przypadkowe rozwiązanie

(zamiana tylko jeśli koszt się zmniejszy)

- Wady: wiele iteracji.



Algorytmy genetyczne

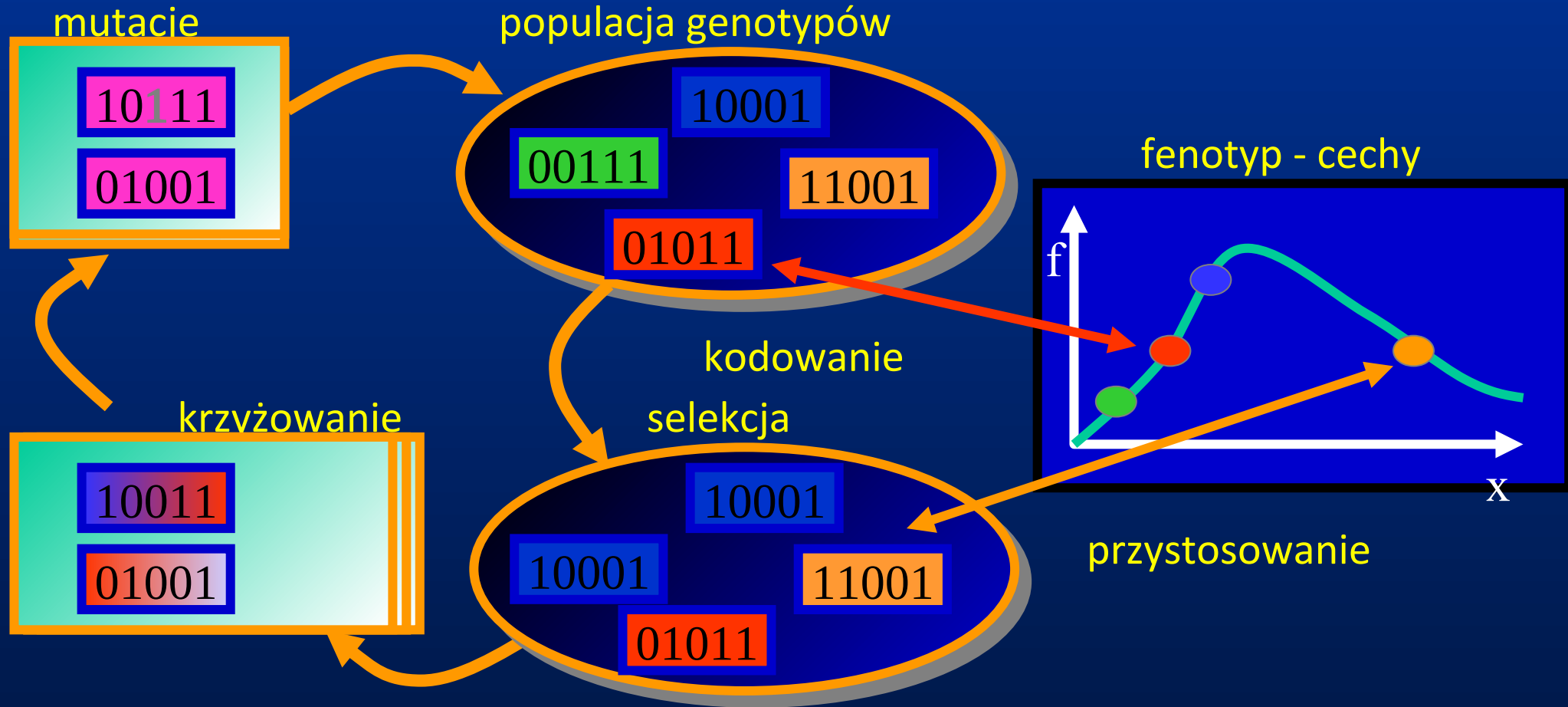
- GA - bardziej wyrafinowane niż SA, startują z populacją a nie jednym węzłem. Inspirowane przez ewolucyjne procesy „przetrwania osobników przystosowanych” z całej populacji = zbioru parametrów.
- GA maksymalizuje funkcje kosztu stopniowo, w kolejnych generacjach, podobnie jak SA ze skokową zmianą temperatury.
- W każdym pokoleniu oceniany jest każdy nowy węzeł za pomocą „funkcji przystosowania”, czyli funkcji heurystycznej oceny zbioru parametrów danego stanu problemu, czyli jego „chromosomu”.
- Najlepsze węzły biorą udział w krzyżowaniu i mutacji zostawiając „potomstwo”, czyli tworząc nowe węzły grafu.
- Najślabiej przystosowane węzły są usuwane.

Przyroda nie działa optymalnie, ale często wystarczająco dobrze.

Nie można teoretycznie udowodnić zbieżności.



Struktura algorytmów ewolucyjnych



Reprezentacja GA

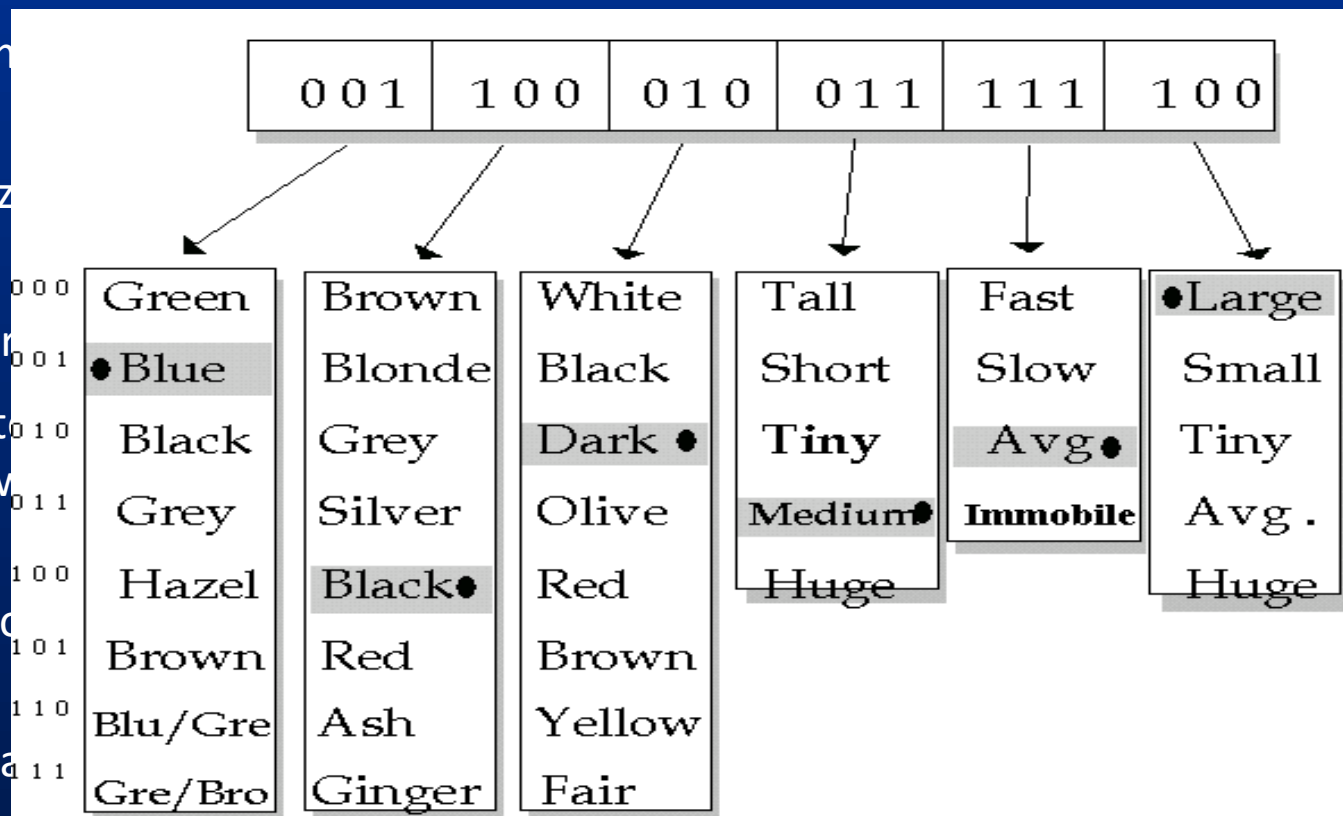
- Populacja tworzy się dzięki chromosomom: są to listy elementów (genów) pozwalające w pełni określić stan.
- Genotyp: możliwe typy chromosomów.
- Fenotyp: konkretny genotyp (instancja).
- Gen: element chromosomu, jego konkretne wartości to allele.
- Genotyp człowieka:
[Kolor oczu, kolor włosów, kolor skóry, wzrost, waga]
- Fenotyp M.M.
[zielony, blond, jasny, niski, piórkowa]
- Gen „kolor oczu” ma allele: zielony, niebieski, brązowy ...

Krzyżowanie

- Para osobników może się mieszać tworząc nowy chromosom składający się z mieszanych genów.
- Krzyżowanie: weź K pierwszych genów z Chromosomu 1 i pozostałe od $K+1$ do N_2 z Chromosomu 2.
- Mutacja: zamień wartość przypadkowo wybranego genu na inną (rzadziej – kilku genów).
- Geny o symbolicznych wartościach dają zbyt duże mutacje, lepiej więc stosować reprezentację binarną, używając kilku bitów do określenia wartości cechy (lub parametry ciągłe).
- Krzyżowanie i mutacje robione są na sub-symbolicznym poziomie, nie patrząc na granice genów.
- Dzięki temu pojawiają się całkiem nowe warianty (allele), których nie mieli rodzice.

Krzyżowanie

- Para osobników może się m...
- genów.
- Krzyżowanie: weź K pierwsz...
- Chromosomu 2.
- Mutacja: zamień wartość p...
- Geny o symbolicznych wart...
- reprezentację binarną, używ...
- ciągłe).
- Krzyżowanie i mutacje robic...
- genów.
- Dzięki temu pojawiają się ca...



Algorytm GA

- Zdefiniuj chromosomy: listy elementów (genów) pozwalające w pełni określić stan/obiekt, określ binarne reprezentacje.
- Zdefiniuj funkcję przystosowania – funkcję heurystycznej oceny jakości stanu dla stosowanego chromosomu.
- Wprowadź operatory genetyczne tworzenia nowych stanów na poziomie krzyżowania i mutacji chromosomów przez zmianę ich genów (możliwych jest tu wiele strategii, np. ruletki).
- Utwórz grupę nowych chromosomów c_i stosując operatory genetyczne.
- Oblicz prawdopodobieństwa oceniając nowe chromosomy za pomocą funkcji przystosowania.
- Zostaw tylko najbardziej obiecujące chromosomy (np. $p_i > 0.6$) i powtórz procedurę.

$$p_i = h(c_i) / \sum_j h(c_j)$$

Możliwych jest wiele wariantów takich algorytmów, łatwo je programować i umożliwiają dużo manipulacji parametrami, stąd wiele publikacji.

J.R. Koza: Algorytmy genetyczne to drugie najlepsze rozwiązanie dla każdego problemu.

Biomorfy

W krótkim czasie dzięki selekcji kierowanej przez człowieka powstały setki gatunków zwierząt: psów, kotów, zwierząt hodowlanych. Wystarczy ukierunkowana selekcja hodowców.

Biomorfy:

- wybierzmy pożądane rozwiązanie docelowe, czyli najlepiej przystosowanego osobnika,
- mutujemy geny pozostałych zostawiając potomstwo podobne do celu.

[Program GAVen](#) + [Opis programu](#)

8 genów koduje biomorf: długość/kierunek rozgałęzienia, każdy ma 4 bity+znak, czyli mamy 40 bitów/biomorf, 8.8 biliona form!
Wartości -15..+15, 9 gen to stopień rozgałęzienia 2-9.

<http://www.rennard.org/alife/english/gavgb.html>

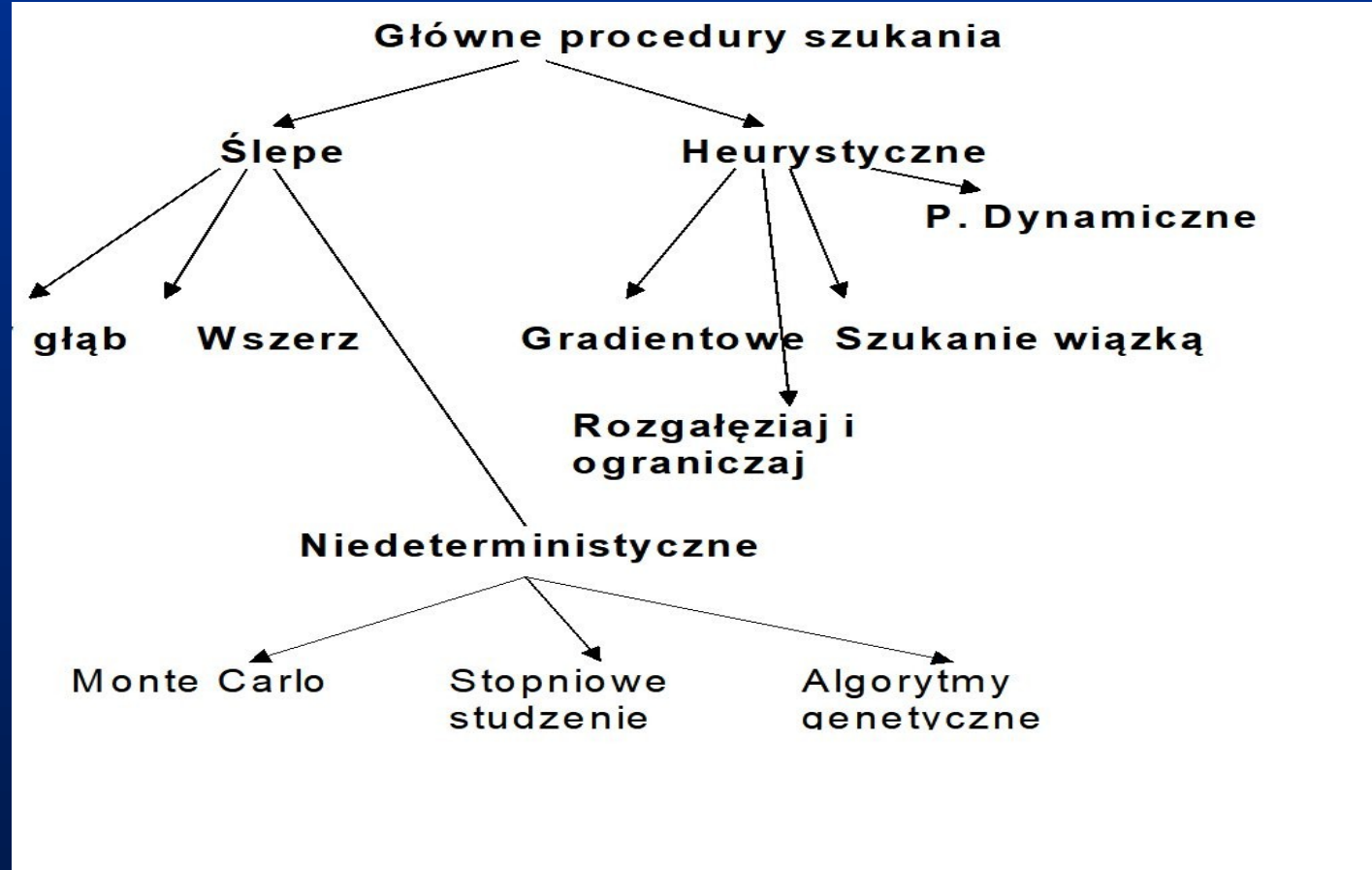
Zastosowania AG

Zastosowania:

- szukanie ciekawych rozwiązań maksymalizujących funkcje heurystyczne; algorytmy genetyczne są uniwersalne i zwykle nie odbiegają daleko od najlepszych, chociaż są kosztowne;
- optymalizacja parametrów = uczenie systemów zdolnych do adaptacji – szczególny przypadek;
- sztuka genetyczna: kombinacja genomów najładniejszych obrazów daje jeszcze ładniejsze, np. [Dr Mutatis](http://www.jhlabs.com/java/art.html) lub <http://www.jhlabs.com/java/art.html>
- Kto nie biega ten nie żyje ... czyli jak nauczyć się poruszać?

Robot evolution, evolving baby robots, filmy ilustrujące naukę poruszania się robotów.

Główne procedury szukania



AI Search Algorithms With Examples

Porównania dla 8-ki.

Rozwiązywanie 8-ki dla 20 i 100 ruchów						
Algorytm	Mem 20	Czas 20	L. ruch 20	Pamięć 100	Czas 100	L. ruch 100
W głąb	-	-	-	-	-	-
W głąb do 5 poziomów	11	144	4	-	-	-
Wszerz	104	60	4	> 4500	> 2000	> 8
Wspinanie do góry	1	5	4	-	-	-
Najpierw najlepszy	6	5	4	148	148	26
A*	6	5	4	86	86	18

Porównania dla minimalnej drogi.

Szukanie minimalnej drogi pociągiem dla bliskich i dalekich miast						
Algorytm	Pamięć, blisko	Czas, blisko	D.trasy, blisko	Pamięć, daleko	Czas, daleko	D.trasy, daleko
W głąb	-	-	-	-	-	-
W głąb do 5 poziomów	14	23	2990	-	-	-
Wszerz	103	31	1860	> 9999	> 3000	> 6000
Wspinanie do góry	1	4	2023	-	-	-
Najpierw najlepszy	7	4	2023	29	12	3592
A*	7	5	1860	15	23	2859

Szukanie z ograniczeniami

Szukanie z ograniczeniami (Constrained Satisfaction Problem, **CSP**) lub problemy z więzami wymagają:

1. Zbioru stanów, którym odpowiadają jakieś wartości numeryczne $X_1, X_2, \dots, X_n$.
2. Celów zawierających testy specyfikujące ograniczenia, ścisłe (muszą być spełnione) i preferowane (mogą być łamane): $C_1, C_2, \dots, C_m$

Stan określony jest przez przypisanie wartości podzbiorowi zmiennych, np.

$$\{X_i = v_i, X_j = v_j, \dots\}$$

Stan zgodny to stan nie naruszający żadnego ograniczenia.

Stan pełny określa wartości wszystkich zmiennych.

Celem jest znalezienie stanu zgodnego i pełnego.

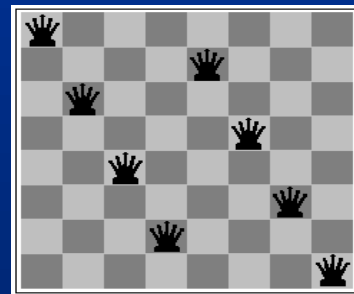
Dodatkowo, możliwa jest **maksymalizacja** funkcji celu.

Przykłady szukania z ograniczeniami:

8-hetmanów, kryptoarytmetyka, projekty VLSI, kolejkowanie, planowanie.

Problemy: 8-hetmanów i mapy

- **Stany:** $\langle V_1, V_2, \dots, V_8 \rangle$, gdzie V_i jest rzędem zajęтым przez i -tego hetmana.
- **Przestrzeń:** $\{1, 2, 3, 4, 5, 6, 7, 8\}$
- **Ograniczenia:** *nie atakuj*
 $\{ \langle 1,3 \rangle, \langle 1,4 \rangle, \langle 1,5 \rangle, \dots \langle 2,4 \rangle, \langle 2,5 \rangle, \dots \}$
gdzie każdy element specyfikuje parę dopuszczalnych wartości zmiennych V_i oraz V_j .

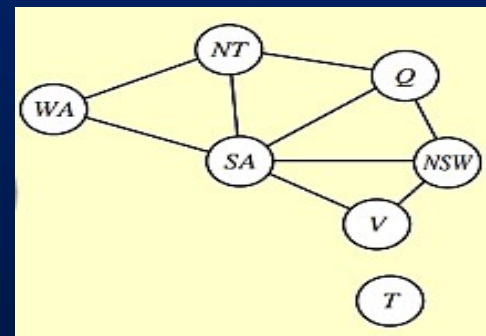


Cel: wszystkie ograniczenia muszą być spełnione.

Inne przykłady: problem 4 barw, kolorowanie mapy – Australia jest prosta, ale Europa?

Przydatne są tu metody optymalizacji np. symulowane wyżarzanie.

Inne przykłady



Optymalizacja ograniczeń

Rodzaje więzów

- Dyskretne (np. dla 8-hetmanów) albo ciągłe np. dla planu podróży.
- Ścisłe (np. przejść wszystkie miasta) oraz preferencje (np. najkrótszy czas przesiadki, najniższa cena).

Strategie:

- Jeśli są liniowe ograniczenia i maksymalizacja funkcji celu to stosujemy programowanie liniowe. Np.

$$a_1x_1 + a_2x_2 + \dots + a_nx_n \geq \alpha$$

$$f(x_1, \dots, x_n) = \alpha + c_1x_1 + c_2x_2 + \dots + c_nx_n$$

- Programowanie nieliniowe jest znacznie trudniejsze.
Warianty: programowanie całkowitoliczbowe i kwadratowe, analiza wypukła (convex analysis) i wiele innych specjalnych przypadków.
- Lista programów do optymalizacji jest długa.
- Python Jupiter CSP notebooks.

Szukanie z więzami

- Kolejność zmiennych jest bez znaczenia, problemy z ograniczeniami są **przemienne**.
- **Stopień zmiennej** to liczba ograniczeń, w których zmienna występuje z zmiennymi jeszcze nie ograniczanymi.
- Szukanie inkrementacyjne: kolejno wiążemy **jedną zmienną** w danym węźle drzewa.
- Heurystyka: rozważyć kolejność zmiennych i ich możliwe wartości, najpierw określić dopuszczalną wartość zmiennych o maksymalnym stopniu ograniczeń.
- Heurystyka **MRV** (*minimum remaining values*) wybiera zmienną z **minimalną liczbą dozwolonych wartości**.
- Heurystyka **najmniej ograniczającej wartości** (*least-constraining value*) – najpierw wartości, które eliminują najmniej możliwości innym zmiennym.
- Szukanie z nawrotami to szukanie **w głąb** z kontrolą ograniczeń.

Analiza celów i środków

Metoda stosowana w złożonych procesach szukania, gdy mamy operatory różnych typów; dla prostych problemów typ może być np. rodzajem i kierunkiem ruchu pionka.

- Oceń różnicę pomiędzy obecnym stanem a celem.
- Wybierz operator, który może tego typu różnice najłatwiej zlikwidować.
- Jeśli najlepszy operator nie da się zastosować do danego stanu, wybierz kolejny.

Metoda prowadzi do rekursywnie definiowanych podproblemów:
jak zastosować dany operator?

Te podproblemy stają się nowymi problemami do rozwiązania.

Przykładowa [ilustracja w Java](#)

[Przykłady z Wikipedii.](#)

Przykładowe pytania

- Opisz algorytm szukania XX.
- Jaka jest złożoność algorytmu XX?
- Podać zalety iteracyjnego pogłębianego szukania w głąb (IDDF).
- Przesuwanka potrzebuje 10 ruchów by ją uporządkować; oszacować pamięć potrzebną dla algorytmu XX.
- W jakich warunkach szukanie przy XX jest równoważne szukaniu YY?
- Jakie są wady i zalety algorytmu XX?
- Jakie są podstawowe elementy algorytmu genetycznego?
- Jakie znasz zastosowania algorytmów genetycznych?
- Jakie są główne procedury szukania?
- Jaką metodę zastosujesz do rozwiązania problemu ZZ?
- Jaką funkcję heurystyczną dla algorytmu A* zastosujesz do ZZ?
- Co to są biomorfy?
- Na czym polega analiza celów i środków? Jakie ma zalety?

Algorytmy szukania - implementacje

Na poniższych stronach jest sporo przykładów różnych algorytmów szukania w zastosowaniu do gier.

AI Search Algorithms Implementations. Popular search algorithms in AI explained and implemented.

- Game programming
<http://theory.stanford.edu/~amitp/GameProgramming/>
- Towards data science
- AI-Search-Algorithms-Implementations
- Variants of A* [Stanford Game Programming](#)
- Prolog notebooks – [repozytorium SWISH](#)
- [Single-Agent Search Demo Page](#)
- [Metody szukania i sztuczna inteligencja w grach](#) - slajdy Krzysztofa Grąbczewskiego