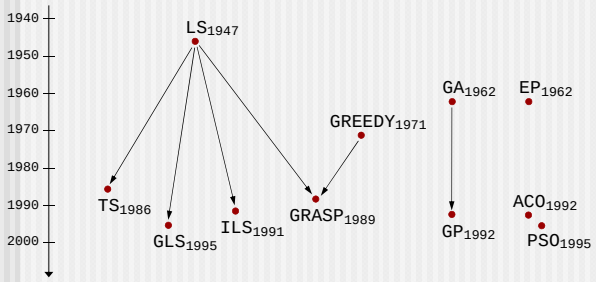
Zaawansowane programowanie

wykład 4: jeszcze o metaheurystykach

prof. dr hab. inż. **Marta Kasprzak**
Instytut Informatyki, Politechnika Poznańska

Genealogia metaheurystyk

- Genealogia wg [El-Ghazali Talbi, *Metaheuristics: From Design to Implementation*, 2009] — wybór



Genealogia metaheurystyk

- Genealogia wg [El-Ghazali Talbi, *Metaheuristics: From Design to Implementation,* 2009] — legenda
  - LS — *local search*, jako pierwszy algorytm tego typu zaliczona metoda *simplex* [Dantzig 1947], brak wskazania na pierwszą typową metodę LS
  - GA — *genetic algorithms*, pierwszy zarys metody [Holland, 1962]
  - EP — *evolutionary programming*, pierwszy zarys metody [Fogel, 1962]
  - GREEDY — *greedy algorithm* [Edmonds, 1971]
  - TS — *tabu search*, pierwszy zarys metody [Glover, 1986]
  - GRASP — *greedy randomized adaptive search procedures* [Feo, Resende, 1989]
  - ILS — *iterated local search* [Martin, Otto, Felten, 1991]; także [Baum, 1986]
  - GP — *genetic programming* [Koza, 1992]
  - ACO — *ant colony optimization* [Dorigo, 1992]
  - GLS — *guided local search* [Voudouris, Tsang, 1995]
  - PSO — *particle swarm optimization* [Eberhart, Kennedy, 1995]

Klasyfikacja metaheurystyk

- Klasyfikacja wg [F. Glover, M. Laguna, *Tabu Search*, 1997] — uzupełniona

Tabu search	A / N / 1
Scatter search	M / N / P
Genetic algorithms	M / S / P
GRASP	M / N / 1
Ant colony optimization	A / S / P
Particle swarm optimization	M / S / P

Legenda: A — *adaptive memory*, M — *memoryless*
N — *neighborhood search*, S — *random sampling*
P — *population-based*, 1 — *single solution*

GRASP

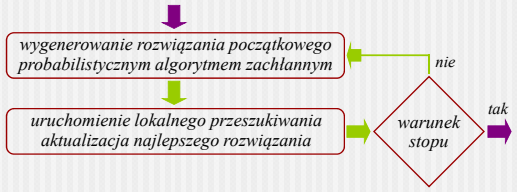
[T.A. Feo, M.G.C. Resende, *Operations Research Letters* 8, 1989, 67–71; *Journal of Global Optimization* 6, 1995, 109–133]

- Metaheurystyka GRASP (ang. *greedy randomized adaptive search procedures*) przypomina podejście *multistart local search*. W kolejnych iteracjach najpierw konstruowane jest rozwiązanie początkowe heurystyką zachłanną, następnie jest ono poprawiane metodą przeszukiwania lokalnego
- Heurystyka zachłanna użyta w GRASP różni się od zwyczajowego podejścia, gdyż zamiast wybierać w każdym kroku element najbardziej poprawiający w danym momencie funkcję celu wybiera jeden z najbardziej obiecujących elementów

GRASP

- Wybór jednego z najbardziej obiecujących elementów dokonywany jest losowo, przez co w każdej iteracji tej metaheurystyki jest szansa na wygenerowanie innego rozwiązania początkowego
- Parametry metaheurystyki:
  - liczba iteracji
  - liczność podzbioru najbardziej obiecujących elementów (może zmieniać się w kolejnych iteracjach)
  - ew. inne

Ogólny schemat metaheurystyki GRASP



Evolutionary programming

[L.J. Fogel, *Proceedings of IFIPC,* Munich 1962, 395–399; L.J. Fogel, A.J. Owens, M.J. Walsh, *Artificial Intelligence through Simulated Evolution*, Wiley, New York 1966]

- *Evolutionary programming* jest metaheurystyką o schemacie zbliżonym do algorytmów genetycznych. Zmienność osobników opiera się tu jedynie na mutacjach i nie ma rekombinacji
- W typowym podejściu każdy rodzic produkuje jednego potomka poprzez losową mutację, po czym następną populację ustala się wybierając połowę najlepszych osobników spośród połączonej puli rodziców i potomków. Selekcja najlepszych osobników odbywa się z użyciem metody turniejowej (ang. *tournament selection*)

Evolutionary programming

- Brak operatora krzyżowania pozwala na dowolny sposób reprezentacji osobnika, nie musi już być liniowa
- Parametry metaheurystyki:
  - liczba iteracji lub inny warunek stopu
  - rozmiar populacji
  - stopień mutacji
  - rozmiar „drużyny" turniejowej
  - ew. inne

Ogólny schemat metaheurystyki EP

Genetic programming

[J. Koza, *Genetic Programming,* MIT Press, Cambridge, MA, 1992]

- *Genetic programming* jest metaheurystyką przeznaczoną do automatycznego generowania programów rozwiązujących zadany problem. Jej schemat przypomina algorytmy genetyczne, z tym że populacja składa się tutaj z zapisanych w postaci drzew programów komputerowych (podejście używane w *AI*)
- Zapis drzewiasty programów został zaczerpnięty ze struktury języka programowania LISP. Charakteryzuje się on jednolitą formą zapisu struktur danych i funkcji w notacji polskiej nawiasowej (tzw. *S-wyrażenia*). Funkcja jest listą z nazwą funkcji jako pierwszym elementem i jej argumentami w dalszej części listy (ciało funkcji definiowane osobną funkcją). Oprócz list/funkcji podstawowym elementem składni języka są atomy (stałe, zmienne)

Genetic programming

- Przykładowy fragment kodu w języku LISP:

```
(define (make-counter x)
  (let ((count x))
    (define (counter)
      (set! count (+ count 1))
      count)
    counter))

(define count-form-10 (make-counter 10))

(display (count-form-10))
(newline)
(display (count-form-10))
(newline)
;Powyższy kod wyświetli 11 i 12.
```

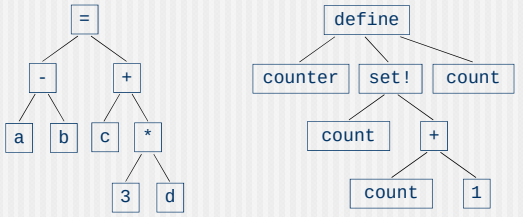
wikipedia.org

Genetic programming

- Przełożenie operacji z programu na postać drzewiastą:

```
(= (- a b)
(+ c (* 3 d)))
```

```
(define (counter) (set!
count (+ count 1)) count)
```



Genetic programming

- Programy składane są z elementów zamieszczonych w predefiniowanych zbiorach dostępnych funkcji i atomów. Należy zapewnić kompletną z punktu widzenia rozwiązywanego problemu zawartość tych zbiorów
- Populacja początkowa generowana jest w dużej mierze losowo, zawiera drzewa o różnym kształcie. Wszystkie wygenerowane programy muszą być wykonywalne (poprawne syntaktycznie). Liczność populacji powinna być bardzo duża (tysiące osobników)
- Parametry tej metaheurystyki i jej schemat są takie same, jak w przypadku algorytmów genetycznych, z dodatkowymi predefiniowanymi zbiorami dostępnych składników kodu źródłowego programu oraz z dopuszczalną głębokością drzewa

Genetic programming

- Należy w szczególny sposób zadbać o poprawność programu, utrzymującą się niezależnie od zastosowanych rekombinacji i mutacji, czyli:
  - wszystkie funkcje i argumenty powinny być tego samego typu (np. liczby całkowite) lub ewentualnie należy zapewnić konwersję typów
  - funkcje powinny być dostosowane do przyjęcia dowolnych argumentów i nie powinny zawieść w żadnym przypadku (np. w dzieleniu przez 0) — należy w razie potrzeby zapewnić zastępczą wartość funkcji
  - dopuszczalna głębokość drzewa lub dopuszczalny rozmiar programu nie powinny zostać przekroczone

Genetic programming

- Selekcja w GP odbywa się na podstawie wartości funkcji przystosowania. Funkcję tę oblicza się uruchamiając programy i porównując wynik z oczekiwanym dla badanych instancji
- Krzyżowanie to zamiana miejscami losowo wybranych poddrzew u dwóch osobników w populacji (dwóch programów)
- Mutacja to losowa zmiana w drzewie osobnika
- Prawdopodobieństwo krzyżowania jest często ustawiane na wartość mniejszą niż 1 (czyli część osobników przechodzi bez zmian do następnego pokolenia)
- Obecnie implementacje wychodzą poza tradycyjny język LISP, także dopuszcza się zamianę struktury drzewiastej na grafową lub liniową

Scatter search

[F. Glover, *Decision Sciences* 8, 1977, 156–166; *New Ideas in Optimization*, McGraw-Hill, New York, 1999, 297–316]

- *Scatter search* jest metaheurystyką opartą na populacji lecz — w przeciwieństwie do innych tego typu metod — w dużej mierze wykorzystującą deterministyczne wybory
- Populacja tworzona jest z najlepszych rozwiązań lub ich fragmentów, osiągniętych uprzednio przez zastosowanie innej heurystyki/metaheurystyki. Powinna charakteryzować się dużym zróżnicowaniem osobników. Osobniki nie muszą tworzyć rozwiązań dopuszczalnych
- Dodatkowo tworzony jest *zbiór referencyjny* (ang. *reference set*), który zawiera wybrane z populacji najlepsze rozwiązania
- Rozmiar zbioru referencyjnego jest niewielki (nie większy niż 20), populacja jest zazwyczaj ok. 10 razy liczniejsza

Scatter search

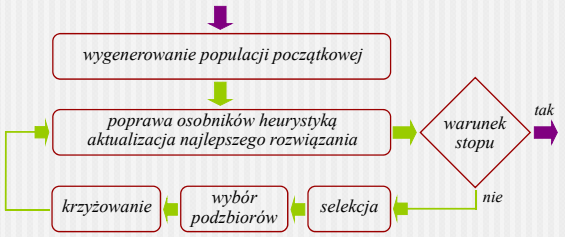
- W „rekombinacji” uczestniczą jedynie osobniki ze zbioru referencyjnego. Wewnątrz tego zbioru wybór zazwyczaj nie opiera się już na ich wartości funkcji celu i wszystkie osobniki biorą udział w tworzeniu nowych
- Powyższe może odbywać się na zasadzie doboru wszystkich możliwych par osobników. Liczba tak stworzonych nowych osobników może przekroczyć liczność populacji, wtedy wybierane są jedynie najlepsze z nich, z dodatkowym wymogiem na zachowanie różnorodności populacji
- Więcej niż dwa osobniki mogą zostać wybrane do rekombinacji. Rekombinacja może wyprodukować jednego lub większą liczbę nowych osobników
- Nowe rozwiązania są następnie poprawiane heurystyką i zastępują te ze zbioru referencyjnego, jeśli są od nich lepsze

Scatter search

- *Scatter search* jest tradycyjnie łączony z metaheurystyką *tabu search* i z inną metodą tego samego autora, *path relinking*. Wszystkie preferują determinizm nad losowością
- Parametry metaheurystyki:
  - liczba iteracji lub inny warunek stopu
  - rozmiar populacji
  - rozmiar zbioru referencyjnego
  - parametry heurystyki/metaheurystyki używanej do poprawy osobników z populacji
  - ew. inne

19

Ogólny schemat metaheurystyki SS



20

Ant colony optimization

[M. Dorigo, *Optimization, learning and natural algorithms*, PhD thesis, Politecnico di Milano, 1992]

- *Ant colony optimization* jest najbardziej znaną metaheurystyką z grupy metod opartych na inteligencji zbiorowej (inteligencji roju, ang. *swarm intelligence*). Odzwierciedla zachowanie kolonii mrówek w procesie zdobywania pożywienia. Inne metody z tej grupy to np. *bee colony optimization* i *particle swarm optimization*
- Zaobserwowano, że połączone działanie prostych obiektów przekazujących sobie informacje może zostać z sukcesem wykorzystane do rozwiązywania problemów optymalizacyjnych. Ruch obiektów w naturze odpowiada w algorytmie przemieszczaniu się w przestrzeni rozwiązań

21

Ant colony optimization

- Mrówki poprzez komunikację pomiędzy sobą odnajdują najkrótszą drogę do pożywienia, dotyczy to także gatunków o słabym wzroku. Idąc, pozostawiają ślad feromonowy, który wskazuje kolejnym mrówkom trasę. Im większe stężenie feromonów na trasie, tym większe prawdopodobieństwo, że kolejna mrówka ją wybierze. Z czasem ślad słabnie
- Dla dwóch tras pomiędzy dwoma punktami, dłuższej i krótszej, prawdopodobieństwo ich wyboru przez mrówki jest początkowo takie same. Po dojściu do celu mrówki wracają wybraną wcześniej trasą. Przejście trasy krótszej wymaga mniej czasu, w dłuższym okresie więcej mrówek zdąży więc zostawić na niej swój ślad. Dodatkowo na krótszej trasie mniejsza ilość feromonu zdąży się ulotnić

22

Ant colony optimization

- Zachowanie mrówek w algorytmie imituje losowa heurystyka zachłanna. Przestrzeń rozwiązań problemu jest modelowana w postaci grafu, w którym ścieżka odpowiada trasie mrówki. Wierzchołkami są elementy składowe rozwiązania
- Mrówka konstruuje rozwiązanie krok po kroku, wybierając następny wierzchołek ścieżki w oparciu o ślad krawędzi wychodzących z bieżącego wierzchołka. Przykładowo, można zastosować prawdopodobieństwo wyboru opisane wzorem:

$$p_{ij} = \tau_{ij} / (\Sigma_{k \in S} \tau_{ik}), \quad \forall j \in S$$

gdzie i oraz j to odpowiednio wierzchołek bieżący i rozważany, τ_{ij} jest feromonowym śladem krawędzi (i,j), a S jest zbiorem wierzchołków spoza bieżącej ścieżki. Stosując np. metodę ruletki losuje się następnika bieżącego wierzchołka. Okresowo można dopuścić zupełnie losowy wybór

23

Ant colony optimization

- Ścieżka feromonowa jest rodzajem pamięci w algorytmie, magazynuje zdobytą przez mrówki wiedzę
- Ulatnianie się feromonu jest realizowane poprzez zastosowanie w każdej iteracji wzoru:

$$\tau_{ij} = (1-\rho)\tau_{ij}, \quad \forall i,j \in [1,n]$$

gdzie $\rho \in [0,1]$ jest współczynnikiem redukcji feromonu, a τ_{ij} jest śladem krawędzi pomiędzy wierzchołkami i oraz j

- Wzmocnienie feromonu może zostać zrealizowane przez zastosowanie różnych strategii, np. po przejściu całej ścieżki przez dodanie wartości proporcjonalnej do jakości rozwiązania, albo po przejściu wszystkich ścieżek w danej iteracji przez uaktualnienie wartości tylko k najlepszych ścieżek

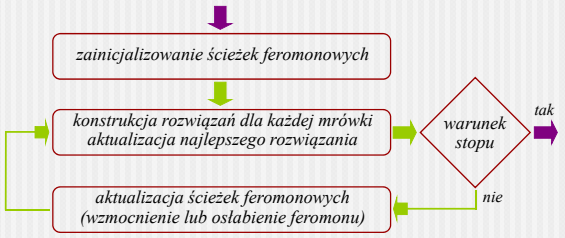
24

Ant colony optimization

- Parametry metaheurystyki:
  - liczba iteracji lub inny warunek stopu
  - liczba mrówek
  - współczynniki redukcji i wzmocnienia feromonu, ew. liczba *k* najlepszych ścieżek
  - ew. inne

25

Ogólny schemat metaheurystyki ACO



26

Particle swarm optimization

[J. Kennedy, R.C. Eberhart, *Proceedings of IEEE International Conference of Neural Networks*, Perth, 1995, 1942–1948]

- *Particle swarm optimization* jest kolejną metaheurystyką z grupy *swarm intelligence*. Odzwierciedla sposób poruszania się obiektów w grupie (zwierząt w stadzie) w sposób skoordynowany i bez centralnego sterowania. Przykładami tego typu ruchów może być lot ptaków w stadzie lub ruch ryb w ławicy w poszukiwaniu pożywienia
- Potencjalne rozwiązania są w PSO nazywane cząsteczkami (ang. *particles*). Trajektoria najlepszych w danym momencie cząsteczek wyznacza trasę pozostałym. Wartość cząsteczki mierzona jest funkcją przystosowania

27

Particle swarm optimization

- W podstawowym modelu każda z N cząsteczek (potencjalnych rozwiązań) porusza się w D-wymiarowej przestrzeni rozwiązań. Ruch cząsteczek opisany jest ich położeniem (wektor x_i) i prędkością (wektor v_i wskazujący także kierunek/zwrot)
- Sąsiedztwo jest definiowane dla każdej cząsteczki w celu określenia wpływu otoczenia. W zależności od zastosowanego podejścia, może obejmować cały zbiór, dużą część zbioru lub jedynie najbliższych sąsiadów
- Cząsteczki zmieniają położenie w kierunku globalnego optimum na podstawie informacji o najlepszym swoim własnym położeniu p_i oraz najlepszym położeniu p_g spośród osiągniętych przez wszystkie cząsteczki z sąsiedztwa

28

Particle swarm optimization

- Prędkość cząsteczki v_i jest uaktualniana następującym wzorem:

$$v_i = v_i' + \rho_1 C_1 \cdot (p_i - x_i') + \rho_2 C_2 \cdot (p_g - x_i')$$

gdzie v_i' oznacza poprzednią prędkość, x_i' poprzednie położenie, ρ_1 i ρ_2 losowe wartości z przedziału [0,1], a C_1 i C_2 to stałe współczynniki uczenia się (ang. *learning factors*; wskazują, czy cząsteczka podąży raczej za własnym sukcesem, czy sukcesem swoich sąsiadów)

- Prędkość obliczana powyższym wzorem podlega korekcie, jeśli przekroczy zdefiniowany wcześniej dopuszczalny zakres wartości. Także można zwiększyć wpływ poprzedniej wartości v_i' przez umieszczenie we wzorze tzw. współczynnika inercji w (wv_i')

29

Particle swarm optimization

- Położenie cząsteczki x_i jest uaktualniane następującym wzorem:

$$x_i = x_i' + v_i$$

W każdej iteracji uaktualniane są także wartości p_i oraz p_g

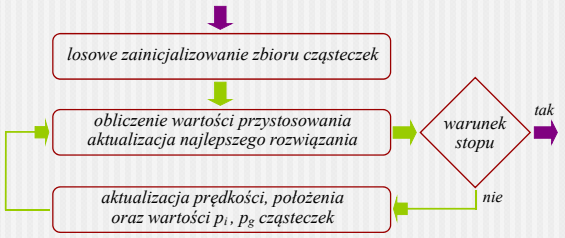
- PSO jest metodą dedykowaną głównie problemom optymalizacji ciągłej. W przypadku problemów kombinatorycznych należy zadbać o to, żeby nowe położenie odpowiadało wartościom akceptowalnym w problemie. Nie zawsze jest to łatwe — jeśli N-wymiarowym wektorem opisującym położenie cząsteczki w przestrzeni rozwiązań byłaby permutacja N elementów, nie wystarczy zaokrąglić wartości w tym wektorze do najbliższych całkowitych

30

Particle swarm optimization

- Aby rozwiązać problem kombinatoryczny, można zastosować odmianę metaheurystyki, w której aktualizacja wektorów odbywa się na zasadzie krzyżowania jak w algorytmach genetycznych. Można także zastąpić wektor prędkości listą dopuszczalnych ruchów, stosowanych po kolei do wektora położenia
- Parametry metaheurystyki:
  - liczba iteracji lub inny warunek stopu
  - liczba cząsteczek
  - rozmiar sąsiedztwa
  - współczynniki uczenia się, współczynnik inercji
  - dopuszczalny zakres wartości prędkości
  - ew. inne

Ogólny schemat metaheurystyki PSO



Heurystyki hybrydowe

- Mianem heurystyki hybrydowej określa się algorytm złożony z połączenia dwóch lub więcej klasycznych podejść heurystycznych (także z użyciem algorytmów dokładnych)
- Często stosowanymi połączeniami są np.:
  - algorytm genetyczny z heurystyką zachłanną
  - algorytm genetyczny z lokalnym przeszukiwaniem
  - *tabu search* i *scatter search*
  - PSO z GA
  - ACO z PSO
  - algorytmy populacyjne z losową heurystyką zachłanną
  - heurystyki z algorytmami dokładnymi

Literatura — cd.

- Fred Glover, Gary A. Kochenberger (ed.), *Handbook of Metaheuristics*, Kluwer Academic Publishers, Boston, 2003.