METAHEURYSTYKI
(Metaheuristics)

Ogólne cechy i klasyfikacja metaheurystyk

1. Metaheurystyki wytyczające trajektorię
    - 2.1. Iterative Improvement (II)
    - 2.2. Simulated Annealing (SA)
    - 2.3. Tabu Search (TS)
    - 2.4. Greedy Randomized Adaptive Search Procedure (GRASP)
    - 2.5. Variable Neighborhood Search (VNS)
    - 2.6. Guided Local Search (GLS)
    - 2.7. Iterated Local Search (ILS)
2. Metaheurystyki oparte na populacji rozwiązań
    - 2.1. Evolutionary Computations (EC)
        - 2.1.1. Genetic Algorithms (GA)
        - 2.1.2. Genetic Programming (GP)
    - 22. Scatter Search & Path Relinking (SS, PR)
    - 2.3. Ant Colony Optimization (ACO)

Heurystyki (gr. odkrywać, znajdować) – metody przybliżone:

- **heurystyki „szyte” – specjalizowane metody rozwiązywania konkretnego problemu, o ściśle określonej złożoności obliczeniowej i jakości aproksymacji**
- **metaheurystyki – ogólne strategie przeszukiwania przestrzeni rozwiązań problemu kombinatorycznego**

Meta-heurystyki (meta-, gr. poza, nad):

- **schematy, szkielety metod, które należy dostosować do rozważanego problemu**
- **metody ogólne, nie są specjalizowane dla żadnego konkretnego problemu**

Metaheurystyki stwarzają szansę znalezienia dobrych rozwiązań dla trudnych obliczeniowo problemów kombinatorycznych, dla których metody dokładne nie są akceptowalne ze względów czasowych.

Jakość aproksymacji, zbieżność metaheurystyk jest często trudna do oszacowania.

Czas działania metaheurystyk zależy od przyjętego warunku stopu, często można jedynie określić złożoność pojedynczej iteracji metody.

OGÓLNE CECHY METAHEURYSTYK

- są to strategie określające sposób przeszukiwania przestrzeni problemu
- celem działania jest efektywne przeszukiwanie przestrzeni problemu w celu znalezienia optimum (lokalnego)
  - znajdowanie dobrych rozwiązań w określonym regionie (intensyfikacja)
  - przeglądanie możliwie najszerszego obszaru przestrzeni problemu (dywersyfikacja)
- są to metody przybliżone i przeważnie niedeterministyczne
- opierają się na różnych technikach od prostego przeszukiwania lokalnego do skomplikowanych procesów uczenia
- wykorzystują mechanizmy zapobiegające uwięzieniu metody w ograniczonym obszarze przestrzeni problemu
- idea metod może być opisana na wysokim poziomie abstrakcji
- nie są specjalizowane dla żadnego specyficznego problemu
- wykorzystują wiedzę o problemie i/lub doświadczenie zgromadzone w procesie przeszukiwania przestrzeni problemu

KLASYFIKACJA METAHEURYSTYK

Inspirowane przez naturę lub nie
inspirowane naturą (SA, GA, ACO) i oparte na innych koncepcjach (TS, ILS)

Bazujące na populacji, lub nie
oparte na populacji rozwiązań (GA, ACO) i na pojedynczych rozwiązaniach czyli wytyczające trajektorię w przestrzeni rozwiązań (TS, ILS, VNS)

Zmieniające dynamicznie funkcję kryterialną
metody wykorzystujące niezmienne sformułowanie funkcji celu będące elementem definicji problemu i modyfikujące funkcję kryterialną (GLS) w oparciu o przebieg przeszukiwań (np. w celu opuszczenia ekstremów lokalnych)

Sąsiedztwo
większość metod opiera się na jednej definicji sąsiedztwa, niektóre (VNS) wykorzystują kilka definicji (dywersyfikacja)

Pamięć (historia obliczeń
metody „bez pamięci” bazujące tylko na bieżącym stanie procesu przeszukiwania (SA), z pamięcią krótko- i długoterminową (zapamiętywanie podjętych decyzji i gromadzenie informacji o problemie)

Często konstruuje się **metody hybrydowe**.

METODY WYTYCZAJĄCE TRAJEKTORIĘ

- proces poszukiwania rozwiązania to ewolucja w (dyskretnym) czasie
- algorytm startuje z rozwiązania początkowego i wytycza trajektorię w przestrzeni rozwiązań problemu
- dynamika procesu zależy od przyjętej strategii
  – trajektoria składa się z dwóch fragmentów związanych z przeszukiwaniem przestrzeni problemu i zbieżnością do extremum lokalnego
  - zaawansowane metody konstruują bardziej złożone trajektorie wielokrotnie powtarzając obie fazy

dynamika procesu zależy od algorytmu, reprezentacji problemu i danych wejściowych (instancji)

PRZESZUKIWANIE LOKALNE
Iterative Improvement (II)

- podstawowa, najprostsza metoda przeszukiwania lokalnego (Local Search), metoda zstępująca dla zagadnienia minimalizacji

Iterative Improvement
s ← GenerujRozwiązaniePoczątkowe()
repeat
    s ← Popraw(s, N(s))
until dalsza poprawa nie jest możliwa

- dla rozwiązania s generowane jest jego sąsiedztwo N(s), z którego wybrane zostaje pierwsze rozwiązanie lepsze od bieżącego (**first improvement**) lub najlepsze rozwiązanie lepsze od bieżącego (**best improvement**)
- algorytm zatrzymuje się w ekstremum lokalnym
- jakość rozwiązania zależy od definicji sąsiedztwa, przyjętego kryterium i danych wejściowych
- metoda prosta w implementacji, szybka w działaniu, ale często generująca rozwiązania o niskiej jakości

METODA SYMULOWANEGO WYŻARZANIA
Simulated Annealing (SA)

- Jedna z najstarszych metaheurystyk.
- Metoda symuluje ewolucję stanu równowagi termodynamicznej ciała stałego podgrzanego do wysokiej temperatury, w którym w miarę ochładzania zachodzą swobodne procesy krystalizacji.

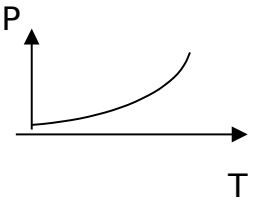
  Proces modelowany w mechanice statystycznej przez algorytm Metropolis (Metropolis i.in. 1953) generujący sekwencje stanów ciała stałego w trakcie jego ochładzania:

  Dla pewnego stanu ciała stałego następuje niewielka przypadkowa perturbacja położenia pewnej przypadkowej cząstki w strukturze.

  Jeśli zmiana ta powoduje obniżenie energii układu ($\Delta E < 0$), to jest akceptowana, a proces jest kontynuowany z nowego stanu.

  Jeśli zmiana zwiększa energię układu ($\Delta E \geq 0$), to jest akceptowana z prawdopodobieństwem

$$P = \frac{1}{Z(T)} e^{-\frac{\Delta E}{k_B T}}$$



gdzie Z(T) współczynnik normalizacyjny, k_B stała Boltzmanna

T – temperatura.

Koncepcję algorytmu Metropolisa zastosowano w latach 80'tych do rozwiązywania problemów optymalizacji kombinatorycznej (Kirkpatrick i.in. 1983)

stan ciała stałego s $\leftrightarrow$ rozwiązanie dopuszczalne s

energia układu E w stanie s $\leftrightarrow$ wartość funkcji kryterialne f(s)

temperatura układu T $\leftrightarrow$ parametr kontrolny T

Idea metody symulowanego wyżarzania:

- metoda startuje z **rozwiązania początkowego** s_0 z pewną **temperaturą początkową** T_0
- w n-tej iteracji metody, dla temperatury T_n, w oparciu o funkcję generacji sąsiedztwa dokonywana jest elementarna zmiana rozwiązania bieżącego s w celu otrzymania rozwiązania sąsiedniego ŝ
- jeśli $\Delta f = f(\hat{s}) - f(s) < 0$, to ŝ jest bezwarunkowo akceptowane jako nowe rozwiązanie bieżące; jeśli $\Delta f \geq 0$, to ŝ jest akceptowane z $P = e^{-\frac{\Delta f}{T_n}}$
- po obniżeniu temperatury zgodnie z przyjętym **schematem chłodzenia** następuje przejście do kolejnej iteracji
- metoda zatrzymuje się po osiągnięciu **warunku stopu**

Jest to metoda pozbawiona pamięci (proces przeszukiwania jest łańcuchem Markowa) – kolejny stan zależy wyłącznie od stanu poprzedniego.

Simulated Annealing

$s_0 \leftarrow$ GenerujRozwiązaniePoczątkowe()

$s \leftarrow s_0$

$n \leftarrow 0$

while nie osiągnięto warunku stopu **do**

wybierz losowo $\hat{s} \in N(s)$

if $f(\hat{s}) - f(s) < 0$ **then** $s \leftarrow \hat{s}$

else zaakceptuj $\hat{s}$ z prawdopodobieństwem $P(\Delta f, T_n)$

zaktualizuj T_n zgodnie ze schematem chłodzenia $Q(T_n, n)$

zwiększ licznik iteracji $n \leftarrow n+1$

rozwiązanie początkowe

- losowe lub heurystyczne

temperatura początkowa

- dobierana eksperymentalnie w wyniku prób wstępnych lub w oparciu o specjalne procedury
- kompromis między zbieżnością metody, a liczba zanalizowanych rozwiązań

warunek stopu

- limit czasu procesora
- maksymalna liczba iteracji
- maksymalna liczba iteracji bez poprawy
- osiągnięcie rozwiązania s o f(s) poniżej założonego progu

schemat chłodzenia (cooling schedule)

- ponieważ proces przeszukiwania jest łańcuchem Markowa istnieje schemat chłodzenia gwarantujący zbieżność metody z prawdopodobieństwem 1 do optimum globalnego, tzn.:

$$\exists_{\Gamma \in R} \lim_{k \to \infty} P[optimum\ \textbf{globalne znalezione w}\ \ \textbf{k}\ \ \textbf{iteracjach}] = \mathbf{1}$$

$$\sum_{\mathbf{k=1}}^{\infty} \mathbf{e}^{\frac{\Gamma}{\mathbf{T_k}}} = \infty$$

taką własności posiada np. ciąg logarytmiczny $T_{k+1} = \frac{\Gamma}{\log(k+k_0)}$

schematy chłodzenia gwarantujące zbieżność do optimum wymagają wykładniczego czasu obliczeń

- w praktyce stosuje się schematy chłodzenia nie gwarantujące zbieżności np. ciąg geometryczny, $T_{k+1} = \alpha T_k$ gdzie $\alpha \in (0, 1)$, modelujący wykładniczy spadek temperatury

schemat chłodzenia (cd.)

- zmiana schematu chłodzenia w trakcie działania metody w celu zachowania równowagi między intensyfikacją i dywersyfikacją:
  - stała/liniowo zmniejszana temperatura na początku działania metody i spadek geometryczny pod koniec
  - niemonotoniczne schematy chłodzenia powodujące naprzemienne ochładzanie i ogrzewanie systemu
- dla stałej temperatury T_n metoda wykonuje L_n iteracji:
  - $L_n = 1$ – (jednorodny łańcuch Markowa) - zmiana temperatury po każdej iteracji
  - $L_n > 1$ – (seria jednorodnych łańcuchów Markowa) - zmiana temperatury po pewnej liczbie iteracji
    - zakłada się, że w danej temperaturze konieczne jest osiągnięcie stanu równowagi termodynamicznej w wyniku pewnej liczby perturbacji stanów (zmian rozwiązania) zmniejszających energię układu (poprawiających rozwiązanie)
    - w praktyce wprowadza się górne ograniczenie liczby iteracji L_n dla temperatury T_n
    - wartość L_n może ulegać zmianie w trakcie działania metody (np. wzrastać do pewnego poziomu maksymalnego ze spadkiem temperatury)

efektywność metody uzależniona od:

- sposobu reprezentacji rozwiązania i rozmiaru przestrzeni rozwiązań
- definicji funkcji kryterialnej
- jakości rozwiązania początkowego
- początkowej wartości temperatury
- sposobu generacji rozwiązania sąsiedniego
- schematu chłodzenia
- warunku stopu

Większość metod metaheurystycznych wymaga wstępnej fazy strojenia w celu odpowiedniego doboru wartości parametrów sterujących.

METODA PRZESZUKIWANIA TABU
Tabu Search (TS)

- Zaproponowana przez Freda Glovera w 1986 roku jako połączenie metody przeszukiwania lokalnego z mechanizmem zapobiegającym uwięzieniu w optimum lokalnym i powstawaniu cykli.

Idea podstawowej metody przeszukiwania tabu:

- metoda startuje od **rozwiązania początkowego** s_0
- dla rozwiązania bieżącego s generowane jest **sąsiedztwo** N(s) na drodze elementarnej modyfikacji tzw. **ruchu**, a następnie ze zbioru kandydatów wybierane jest najlepszego rozwiązanie jako nowe rozwiązanie bieżące
- metoda zatrzymuje się po osiągnięciu **warunku stopu**
- metoda z pamięcią krótkoterminową – modyfikacje rozwiązania zapamiętywane są na **liście tabu,** a rozwiązania posiadające **status tabu** są wykluczane ze zbioru kandydatów – nie mogą być wybrane jako nowe rozwiązanie bieżące

Simple Tabu Search

$s_0 \leftarrow$ GenerujRozwiązaniePoczątkowe()
$s \leftarrow s_0$
ListaTabu $\leftarrow \varnothing$
while nie osiągnięto warunku stopu **do**
    s $\leftarrow$ NajlepszeRozwiązanie(s, N(s), ListaTabu)
    Aktualizuj(ListaTabu)

rozwiązanie początkowe

- losowe lub heurystyczne

sąsiedztwo

- zbiór rozwiązań otrzymywanych z s na drodze elementarnej modyfikacji (ruchu)
- rozmiar sąsiedztwa wpływa na efektywność metody - kompromis między czasem obliczeń a jakością (różnorodnością) rozwiązań

lista tabu

- metoda akceptuje również rozwiązania pogarszające funkcję celu co grozi wpadnięciem w cykl
- lista tabu - pamięć krótkoterminowa (**short term memory**) – służy zapobieganiu cyklom zapamiętując historię przeszukiwań przestrzeni problemu
- struktura listy tabu – elementami listy mogą być:
  - rozwiązania – (**Ideal Tabu Search**) nieefektywne ze względów pamięciowych i czasowych
  - atrybuty – cechy ruchu lub rozwiązania, różnice między rozwiązaniami (utrzymywanie wielu list tabu związanych z różnymi atrybutami)
- wybór nowego rozwiązania bieżącego ograniczony jest do zbioru rozwiązań nie posiadających **statusu tabu** (status tabu posiadają elementy występujące na liście tabu lub na co najmniej jednej liście tabu, na wszystkich listach tabu...)

lista tabu (cd.)

- długość listy tabu (**tabu tenure**) jest:
  - skończona
  - warunkuje efektywność przeszukiwania (kompromis między zbieżnością, a eksploracją przestrzeni rozwiązań)

  długość listy tabu może być:
  - stała (dobierana eksperymentalnie)
  - zmienna (**dynamiczna lista tabu**):
    - modyfikowana losowo co pewną liczbę iteracji
    - modyfikowana w celu sterowania procesem przeszukiwania w oparciu o jego historię (**Reactive Tabu Search**)
- obsługa listy wg zasady FIFO (kolejka)
- możliwość powstania cyklu

Skończona lista tabu oraz umieszczania na niej atrybutów, a nie rozwiązań powoduje, że status tabu mogą uzyskać rozwiązania dotychczas nie analizowane.

poziom aspiracji

- znosi status tabu związany z określonym atrybutem, jeśli jakość nowego rozwiązania przekracza założony próg

warunek stopu

- limit czasu procesora
- maksymalna liczba iteracji
- maksymalna liczba iteracji bez poprawy
- osiągnięcie rozwiązania s o f(s) poniżej założonego progu
- brak rozwiązań w sąsiedztwie bez statusu tabu

Tabu Search

$s_0 \leftarrow$ GenerujRozwiązaniePoczątkowe()

$s \leftarrow s_0$

$n \leftarrow 0$

UtwórzListyTabu(TL_1, ...TL_r)

while nie osiągnięto warunku stopu **do**

$C \leftarrow \{z \in N(s)$: *co najmniej* jeden warunek tabu nie jest spełniony lub przynajmniej jeden poziom aspiracji został osiągnięty$\}$

$s \leftarrow$ NajlepszeRozwiązanie(s, C)

AktualizujListyTabu_i_PoziomyAspiracji()

$n \leftarrow n+1$

W przedstawionej formie metoda jest algorytmem deterministycznym. Często wprowadza się mechanizm **dywersyfikacji** – znacznej modyfikacji rozwiązania w celu przeniesienia przeszukiwań w inny region.

dywersyfikacja

- losowa
- wykorzystująca pamięć długoterminową (**long term memory**) związaną z atrybutami (ruchami) opartą na :
  - czasie ostatniego wystąpienia (**recency**)
  - **-**częstości występowania (**frequency**)
  - jakości rozwiązań (**quality**)
  - wpływie na strukturę rozwiązania (**influence**)
- liczba dywersyfikacji może być elementem warunku stopu

Pamięć długoterminowa może być również wykorzystana do konstrukcji dobrych rozwiązań początkowych i ukierunkowywania procesu przeszukiwań.

efektywność metody uzależniona od:

- sposobu reprezentacji rozwiązania i rozmiaru przestrzeni rozwiązań
- definicji funkcji kryterialnej
- jakości rozwiązania początkowego
- definicji i rozmiaru sąsiedztwa
- definicji i długości listy tabu
- definicji poziomu aspiracji
- warunku stopu

GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURE (GRASP)

- prosta metoda (bez pamięci) stanowiąca połączenie heurystyki konstruującej rozwiązanie z lokalnym przeszukiwaniem (od II do TS)

GRASP
while nie osiągnięto warunku stopu **do**
    s ← KonstrukcjaRozwiązaniaZachłannego()
    LokalnePrzeszukiwanie(s)
    ZapamiętanieNajlepszegoRozwiązania()

Heurystyka zachłanna jest oparta na dynamicznej konstrukcji rozwiązania i randomizacji

KonstrukcjaRozwiązaniaZachłannego()
s ← ∅ // *rozwiązanie częściowe* $\{x_1, .., x_n\}$
while rozwiązanie nie jest kompletne **do**
    RCL ← OgraniczonaListaKandydatów()
    x ← LosowyElement(RCL)
    s ← s ∪ {x}
    AktualizacjaWartościFunkcjiPrzystosowania(s)

- RCL jest listą elementów spoza s, uporządkowanych wg pewnej funkcji przydatności i ograniczoną do k najlepszych elementów:
  - k =1 – klasyczna metoda zachłanna
  - k =|CL| - metoda w pełni losowa

VARIABLE NEIGHBOURHOOD SEARCH (VNS)

- metaheurystyka polegająca na dynamicznej modyfikacji sposobu generacji sąsiedztwa wykorzystująca listę metod generacji sąsiedztwa N_1, N_2, ..., N_K uporządkowanych tak, że $N_1 \subset N_2 \subset ... \subset N_K$

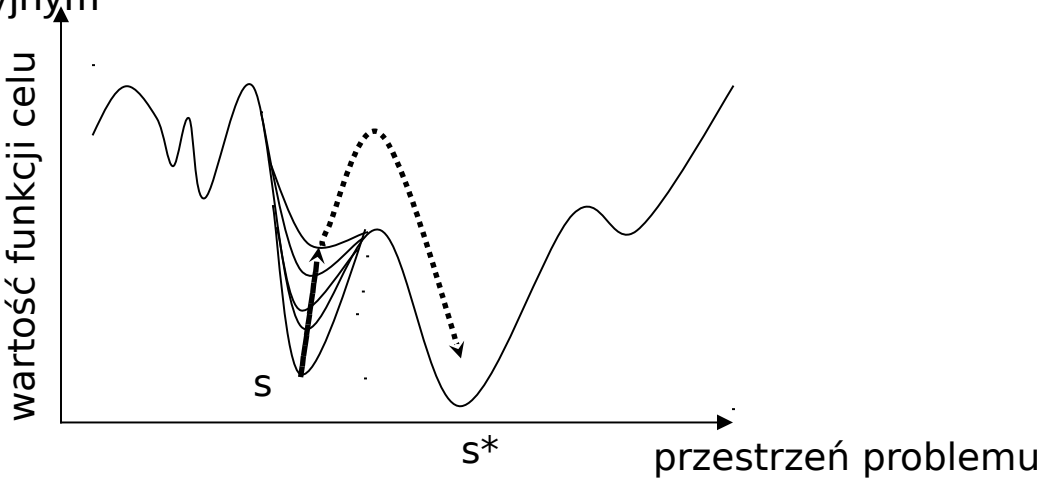
Variable Neighbourhood Search

s ← GenerujRozwiązaniePoczątkowe()
while nie osiągnięto warunku stopu **do**
    k ← 1
    while k < K **do**
        s' ← LosoweRozwiązanie(N_k(s))
        s" ← LokalnePrzeszukiwanie(s') *// z dowolną definicją sąsiedztwa*
        if f(s")<f(s) **then** s ←s"
                                  k = 1
                          else k = k+1

- zmiana sąsiedztwa pozwala na wybór różnych koncepcji lokalnego przeszukiwania (dywersyfikacja)
- sąsiedztwo jest zawsze generowane wokół „dobrego" rozwiązania
- rozwiązanie niekorzystne dla jednej metody generacji sąsiedztwa może być korzystne dla innej
- optimum lokalne dla jednej metody generacji nie musi nim być dla innej

GUIDED LOCAL SEARCH (GLS)

- modyfikacja funkcji celu (zamiast metody generacji sąsiedztwa) tak aby uczynić minimum lokalne „mniej atrakcyjnym”



- wyodrębnienie **elementów rozwiązania** – różnicujących rozwiązania między sobą

$$I_i(s) = \begin{cases} 1, & \text{jesli element } i \text{ jest w rozwiazaniu } s \\ 0, & \text{w przeciwnym wypadku} \end{cases}$$

- p_i – współczynnik kary, koszt przynależności i do rozwiązania
- c_i – zysk z przynależności elementu i do rozwiązania
- U(i,s) - użyteczność elementu i w rozwiązaniu s, gdzie $U(i,s) = I_i(s)\frac{c_i}{1+p_i}$
- zmodyfikowana funkcja celu (λ - współczynnik normalizacyjny):

$$f'(s) = f(s) + \lambda \sum_{i=1}^{m} p_i I_i(s)$$

Guided Local Search

s ← GenerujRozwiązaniePoczątkowe()
while nie osiągnięto warunku stopu **do**
    s ← LokalnePrzeszukiwanie(s, f')
    for wszystkich elementów i z maksymalną użyteczności U **do**
        $p_i \leftarrow p_i+1$
    Aktualizuj(f',[p_i])

- ze wzrostem p_i spada użyteczność elementów, które były dotychczas korzystne dla jakości bieżącego rozwiązania
- spadek „atrakcyjności" pewnych elementów eliminuje je z rozwiązania i pozwala na opuszczenie minimum lokalnego
- dzięki współczynnikom kary wartość funkcji celu zależy nie tylko od parametrów problemu, ale i od historii przeszukiwań
- co pewną liczbę iteracji (np. co kilkaset iteracji) można zastosować inną regułę modyfikacji p_i w celu zrównania współczynników kary:

$$p_i = \alpha p_i \text{ dla } \alpha \in (0,1)$$

ITERATED LOCAL SEARCH (ILS)

- ogólna koncepcja metody wytyczającej trajektorię
  - opisuje inne metaheurystyki
  - inne metaheurystyki mogą być jej komponentami

- metoda startuje z pewnego optimum lokalnego i do osiągnięcia warunku stopu:
  - zaburza rozwiązanie bieżące w oparciu o historię procesu przeszukiwania
  - stosuje przeszukiwanie lokalne do rozwiązania bieżącego otrzymując optimum lokalne
  - akceptuje nowe optimum lokalne, jeśli spełnione jest kryterium akceptacji

Iterated Local Search

$s_0 \leftarrow$ GenerujRozwiązaniePoczątkowe()

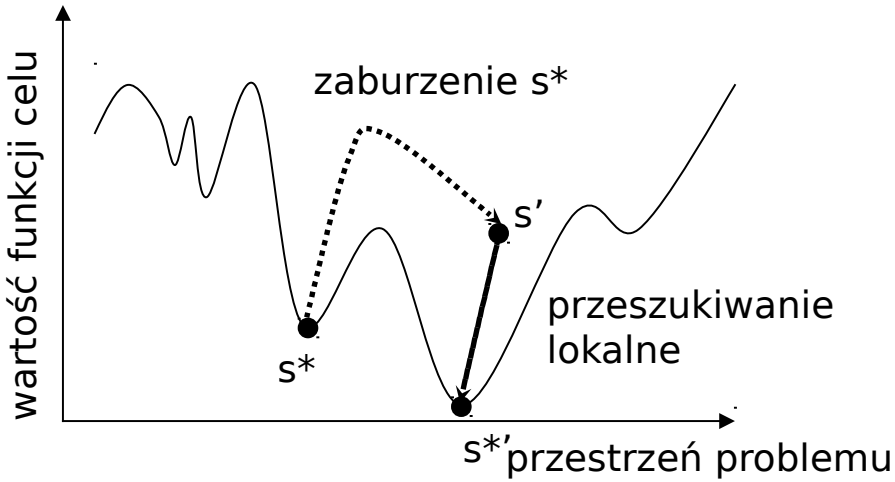
$s^* \leftarrow$ LokalnePrzeszukiwanie(s_0)

while nie osiągnięto warunku stopu **do**

  $s' \leftarrow$ Zaburzenie(s^*, H)

  $s^{*'} \leftarrow$ LokalnePrzeszukiwanie(s')

  $s^* \leftarrow$ KryteriumAkceptacji(s^*,$s^{*'}$,H)



rozwiązanie początkowe

- generowane w krótkim czasie
- od metod losowych do heurystyk „szytych”

zaburzanie rozwiązania

- generacja punktu startowego dla przeszukiwania lokalnego, które zakończy się rozwiązaniem lepszym (i bliższym poprzedniemu optimum lokalnemu) niż rozwiązanie osiągnięte dla rozwiązania wygenerowanego losowo
- niedeterministyczne (w celu uniknięcia cykli)
- siła zaburzenia – „odległość” zaburzonego rozwiązania od rozwiązania oryginalnego (liczba modyfikacji):
  - kompromis między przeszukiwaniem losowym, a zagrożeniem pozostania w minimum lokalnym
  - od stałej przez zależną od rozmiaru problemu do wyznaczanej przez osobny algorytm w oparciu o historię procesu przeszukiwania

kryterium akceptacji

- zapobiega akceptacji zbyt podobnych rozwiązań
- od akceptacji wszystkich rozwiązań do akceptacji rozwiązań lepszych od dotychczasowych przez koncepcje pośrednie tj. schematy chłodzenia

historia procesu przeszukiwania

- pamięć krótko- lub długoterminowa

METODY OPARTE NA POPULACJI ROZWIĄZAŃ

- w każdej iteracji analizowany jest zbiór rozwiązań (populacja), a nie pojedyncze rozwiązanie

ALGORYTMY EWOLUCYJNE
Evolutionary Algorithms (EA)

- model obliczeniowy procesów ewolucyjnych, oparty na teorii ewolucji Karola Darwina zakładającej, że podstawowymi czynnikami kształtującymi życie na Ziemi są:
  - dziedziczenie
  - zmienność organizmów
  - dobór naturalny

Idea algorytmów ewolucyjnych:

- metoda startuje z początkowym zbiorem rozwiązań – **chromosomów** (genotypów) składających się na **populacją** początkową
- w każdej iteracji następuje zastosowanie **operatorów genetycznych** do populacji w celu dokonania **rekombinacji** (recombination)
  - **krzyżowanie** - połączenie 2 lub więcej chromosomów w celu otrzymania nowych
  - **mutacji** – modyfikacja pojedynczego chromosomu
- chromosomy należące do populacji poddawane są **wartościowaniu** w oparciu o **funkcję przystosowania**
- wartość funkcji przystosowania decyduje o wyborze chromosomu do nowej populacji w procesie **selekcji**

3 niezależne kierunki rozwoju algorytmów ewolucyjnych:

- programowanie ewolucyjne (**Evolutionary Programming**) – Fogel i.in. 1966 – koncepcja rozwinięta w ramach badań nad inteligencją maszynową, zastosowana do dyskretnej reprezentacji stanów automatów skończonych, wykorzystywana obecnie w optymalizacji ciągłej
- strategie ewolucyjne (**Evolutionary Strategies**) – Rechenberg 1973 – rozwijane w ramach optymalizacji ciągłej

- **ALGORYTMY GENETYCZNE** (**GENETIC ALGORITHMS**) – Holland 1975, i.in. – rozwijane w ramach optymalizacji dyskretnej

Evolutionary Algorithms

P ← GenerujPopulacjęPoczątkową()
Wartościowanie(P)
while nie osiągnięto warunku stopu **do**
    P' ← Krzyżowanie(P)
    P" ← Mutacja(P')
    Wartościowanie(P")
    P ← Selekcja(P"∪P)

chromosomy

- elementy populacji: rozwiązania problemu lub komponenty rozwiązań (rozwiązania częściowe lub dowolne obiekty, które można przekształcić w rozwiązania problemu)
- reprezentacja chromosomów – ciąg bitów lub liczb całkowitych, reprezentacje złożone np. drzewiaste

populacja początkowa

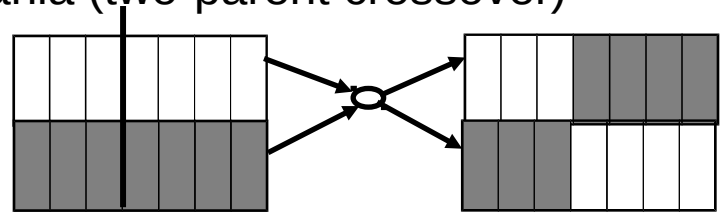
- generowana losowo lub na drodze losowego zaburzenia rozwiązań otrzymanych przez inne metody

krzyżowanie (crossover)

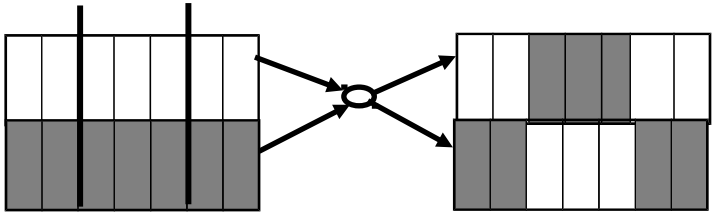
- ograniczone lub nieograniczone krzyżowanie - możliwość krzyżowania chromosomu z wybranymi lub wszystkimi chromosomami populacji
- połączenie 2 (two-parent crossover) lub więcej (multi-parent crossover) chromosomów w celu otrzymania nowych łączących cechy rodziców
- modeluje dobór naturalny i dziedziczenie
- realizacja mechanizmu **intensyfikacji:**
  – generacja nowych rozwiązań z połączenia dobrych dotychczas znalezionych rozwiązań zamiast generacji losowej
  - dodatkowe mechanizmy intensyfikacji – lokalne przeszukiwanie dla elementów populacji (metody hybrydowe)

krzyżowanie (cd.)

- dwu-argumentowy operator krzyżowania (two-parent crossover)
  - jednopunktowe krzyżowanie (one-point crossover)



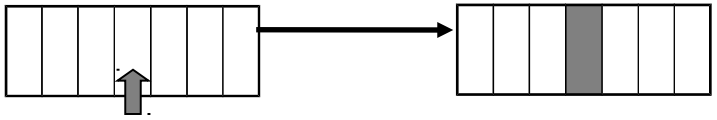
  - dwupunktowe krzyżowanie (two-point crossover)



  - losowy wybór miejsca (miejsc) krzyżowania
  - wybór rodziców:
    - losowy - dwukrotne losowanie pary chromosomów i wybór lepszego chromosomu z pary (wg funkcji przystosowania) jako rodzica
    - wg rankingu – wybór par chromosomów z prawdopodobieństwem zależnym od wartości funkcji przystosowania
- problem niedopuszczalności rozwiązań powstałych w efekcie krzyżowania:
  - odrzucenie rozwiązań niedopuszczalnych (reject)
  - wprowadzenie funkcji kary (penalization)
  - próba przywrócenia dopuszczalności rozwiązania (repair)
- **prawo krzyżowania** – liczba chromosomów podlegających krzyżowaniu w każdej iteracji

mutacja (mutation)

- jedno-argumentowy operator genetyczny



- losowa (niewielka) zmiana chromosomu w wyniku zmiany pojedynczych bitów (symboli) kodujących rozwiązanie

- modeluje zmienność organizmów i przystosowanie do warunków środowiska

- realizacja mechanizmu **dywersyfikacji**:
  - losowa modyfikacja rozwiązania
  - dodatkowe mechanizmy dywersyfikacji - wprowadzenie do populacji nowych chromosomów z nie odwiedzonych regionów przestrzeni rozwiązań w oparciu o pamięć długoterminową

- **prawo mutacji** – liczba chromosomów podlegających mutacji w każdej iteracji

selekcja (selection)

- modeluje proces ewolucyjny
- wybór chromosomów do nowej populacji w każdej iteracji:
  - wymiana pokoleń (**generation replacement**) – zastąpienie w całości starej populacji nowymi chromosomami otrzymanymi w efekcie rekombinacji, czyli krzyżowania i mutacji
  - stabilny proces ewolucyjny (**steady state evolution process**) – pozostawienie w nowej populacji części chromosomów z populacji pierwotnej (np. najlepszych)
- selekcja odbywa się w oparciu o wartość funkcji przystosowania (**fitness function**)
- prawdopodobieństwo selekcji rośnie z wartością funkcji przystosowania
  - modelowanie prawa przetrwania najlepszych
  - nawet słabe chromosomy mają szansę pozostania w populacji
- stała lub zmienna liczność populacji we wszystkich iteracjach

warunek stopu

- limit czasu procesora
- maksymalna liczba iteracji
- maksymalna liczba iteracji/populacji, w których najlepszy chromosom nie uległ zmianie
- osiągnięcie rozwiązania s o f(s) poniżej założonego progu
- rozmiar populacji równy 1

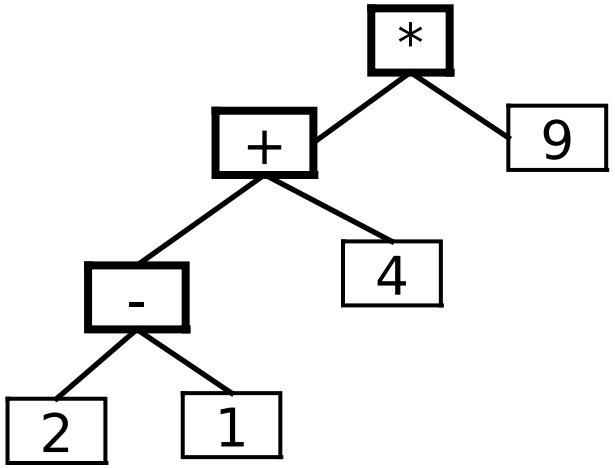
efektywność metody uzależniona od:

- definicji przestrzeni problemu i sposobu reprezentacji rozwiązania
- definicji funkcji przystosowania
- definicji operatorów genetycznych (mutacji i krzyżowania)
- prawa mutacji i prawa krzyżowania
- reguły selekcji
- rozmiaru populacji
- warunku stopu

PROGRAMOWANIE GENETYCZNE
Genetic Programming (GP)

- koncepcja analogiczna do algorytmów genetycznych
- elementami populacji nie są rozwiązania, ale programy generujące rozwiązania
- metoda realizuje proces ewolucji programów, dokonuje ich rekombinacji (krzyżowania, mutacji) i selekcji w celu otrzymania programów najlepiej realizujących założony cel
- chromosomy - programy – są przeważnie reprezentowane w postaci drzew zbudowanych z obiektów będącymi danymi, stałymi, instrukcjami, operatorami, ...

9*((2-1)+4)



SCATTER SEARCH & PATH RELINKING (SS & PR)

- uzupełnienie obliczeń ewolucyjnych o ścisłą zasadę rekombinacji rozwiązań przez zastosowanie idei pochodzących z metody przeszukiwania tabu

Idea algorytmów:

- metoda generuje nowe rozwiązania w oparciu o zbiór **rozwiązań referencyjnych** (rozwiązań dopuszczalnych problemu)
- nowy zbiór **rozwiązań próbnych** tworzony jest na drodze rekombinacji podzbioru zbioru referencyjnego
- rozwiązania próbne poddawane są optymalizacji (ewentualnie procedurze naprawczej) w celu otrzymania **rozwiązań rozproszonych** (w sąsiedztwie zbioru referencyjnego)
- selekcja rozwiązań rozproszonych i referencyjnych do nowego zbioru referencyjnego
- **Scatter Search** – przeszukiwanie przestrzeni Euklidesowej na drodze generacji kombinacji liniowej punktów referencyjnych (reprezentacji rozwiązań referencyjnych w przestrzeni Euklidesowej)

Path Relinking – uogólnienie metody dla dowolnej przestrzeni problemu, budowa ścieżek między rozwiązaniami przez dodawanie do pewnego rozwiązanie początkowego (initial solution) atrybutów rozwiązań sterujących (guiding solutions) (rozwiązania te tworzą zbiór referencyjny)

Initial Phase

GenerujRozwiązaniaPoczątkowe()
repeat
    DywersyfikacjaWybranegoRozwiązania()
    PoprawaRozwiązań()
    AktualizacjaZbioruReferencyjnego()
until liczność zbioru referencyjnego wynosi n

Scatter Search / Path Relinking

repeat
    GeneracjaPodzbiorów ()
    Rekombinacja()
    PoprawaRozwiązań()
    AktualizacjaZbioruReferencyjnego()
until nie osiągnięto warunku stopu

Faza wstępna:

- z pojedynczego rozwiązania początkowego generowany jest zbiór rozwiązań sąsiednich, poddawany optymalizacji i ewentualnej naprawie; najlepsze uzyskane rozwiązanie staje się rozwiązaniem referencyjnym
- problem zapewnienia odpowiedniej liczności zbioru początkowego, a tym samym zbioru referencyjnego

Proces optymalizacji:

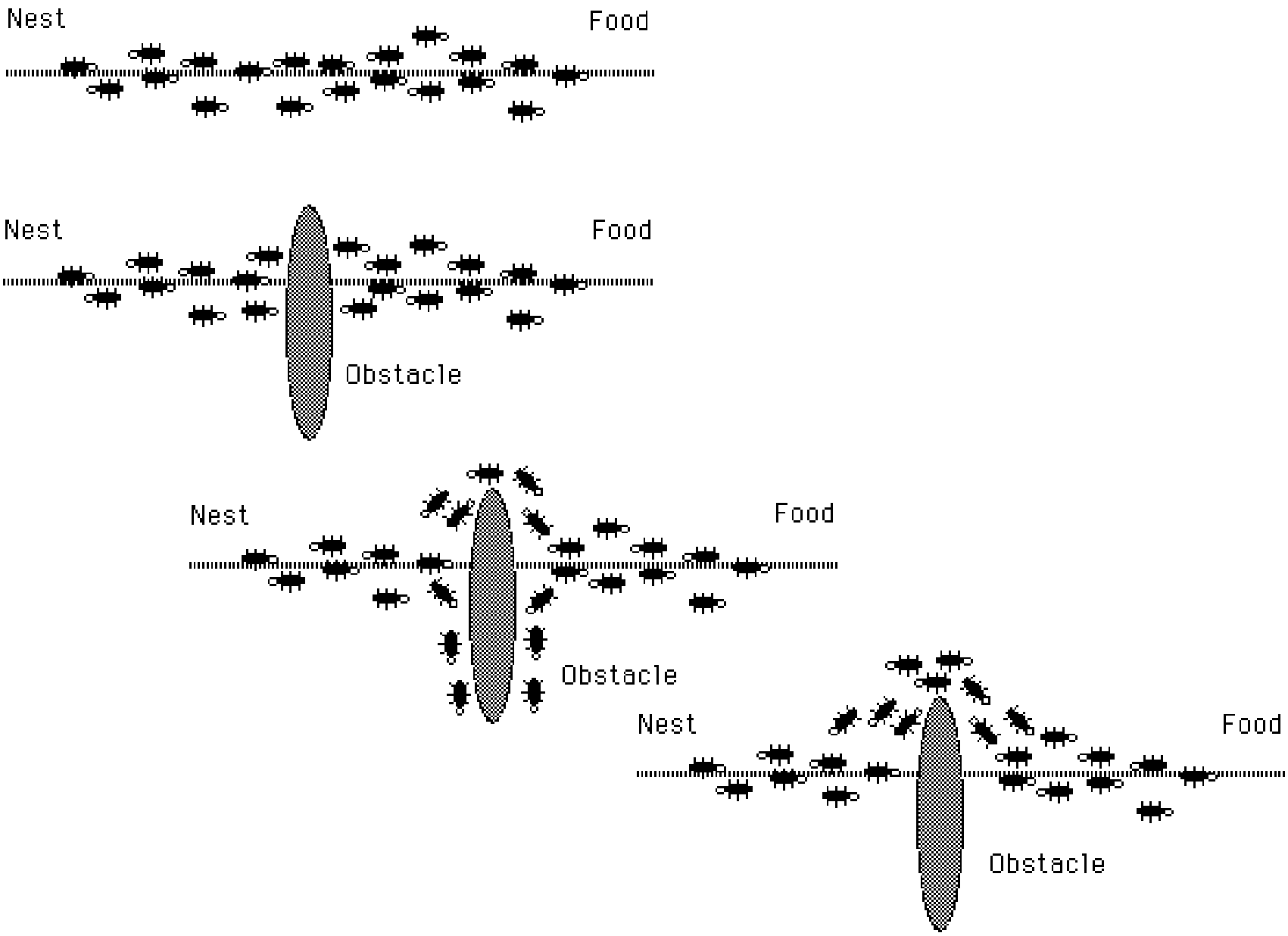
- w każdej iteracji (dłuższej niż w GA) ze zbioru referencyjnego wybierane są podzbiory poddawane rekombinacji (np. rekombinacja wszystkich par rozwiązań ze zbioru referencyjnego)
- w nowym zbiorze referencyjnym (mniej licznym niż populacja w GA) umieszczane są „najlepsze” rozwiązania tzn. o wysokiej jakości i dużej różnorodności
- mniej parametrów sterujących niż w GA

ALGORYTMY MRÓWKOWE
Ant Colony Optimization (ACO)

- metoda zaproponowana przez Marco Dorigo w 1992 roku w oparciu o obserwacje zachowania kolonii mrówek, które odnajdują najkrótszą drogę między mrowiskiem a pożywieniem w oparciu o ślad feromonowy (zapachowy)

Idea metody:

- w algorytmach mrówkowych **sztuczne mrówki** budują stopniowo rozwiązanie problemu przez dodawanie komponentów do rozwiązania częściowego (metoda może być stosowana jedynie dla problemów, w których rozwiązanie może być budowane w sposób stopniowy)
- mrówki poruszają się w sposób losowy po **grafie problemu** G = (C, L) gdzie C oznacza zbiór **komponentów** rozwiązania, a L **połączeń** między nimi
- procedura sterująca ruchem mrówki może (ale nie musi) zapewniać dopuszczalność generowanych rozwiązań
- z komponentami $c_i \in C$ lub połączeniami $l_{ij} \in L$ związany jest **ślad feromonowy** (τ_i lub τ_{ij}) oraz **współczynnik heurystyczny** (η_i lub η_{ij}) reprezentujące historię przeszukiwań i zgromadzoną wiedzę o problemie (znaczeniu komponentu)
- w oparciu o wartości τ_i i η_i (τ_{ij} i η_{ij}) mrówka podejmuje probabilistyczną decyzję co do ścieżki po jakiej porusza się po grafie problemu (w oparciu o **prawdopodobieństwo przejścia**)


Nest
Food
Nest
Food
Obstacle
Nest
Food
Obstacle
Nest
Food
Obstacle

- najprostszy algorytm mrówkowy

Ant System
UstawPoczątkoweWartości(τ)
while nie osiągnięto warunku stopu **do**
  for wszystkich mrówek $a \in A$ **do**
    $s_a \leftarrow$ KonstruujRozwiązanie(τ, η)
  ZaktualizujŚladFeromonowy()

- mrówki podejmują decyzję o dodaniu komponentu do rozwiązania w oparciu o **probabilistyczna funkcję przejść**

$$p(c_r \mid s_a[c_q]) = \begin{cases} \dfrac{[\eta_r]^{\alpha}[\tau_r]^{\beta}}{\sum_{c_u \in J(s_a[c_q])} [\eta_u]^{\alpha}[\tau_u]^{\beta}} & \text{, jesli } c_r \in J(s_a[c_q]) \\ 0 & \text{, w przeciwnym wypadku} \end{cases}$$

$p(c_r|s_a[c_q])$ - prawdopodobieństwo dodania komponentu c_r do rozwiązania s_a budowanego przez mrówkę a przy założeniu, że ostatnim dołączonym komponentem był komponent c_q

$J(s_a[c_q])$ – zbiór komponentów, które mogą być dołączone do rozwiązania $s_a[c_q]$

α, β - współczynniki normalizujące wpływ śladu feromonowego i współczynnika heurystycznego

- modyfikacja śladu feromonowego

$$\tau_i = (1-\rho)\tau_i + \sum_{a \in A} \Delta\tau_i^{s_a}$$

$$\Delta\tau_i^{s_a} = \begin{cases} \frac{Q}{f(s_a)} & , \text{jesli } c_i \text{ bylowykorzystanew } s_a \\ 0 & , \text{w przeciwnymwypadku} \end{cases}$$

$f(s_a)$ – jakość rozwiązania s_a
ρ - współczynnik parowania feromonu, $0 < \rho < 1$
Q – parametr kontrolny (zazwyczaj Q = 1)
początkowa wartość τ_i jest ustalana jako mała liczba dodatnia

- ślad feromonowy ulega osłabieniu w czasie działania metody (mechanizm dywersyfikacji)

- ślad feromonowy τ_i jest wzmacniany dla komponentów wprowadzanych do rozwiązania:
  - w momencie dołączanie komponentu c_i w oparciu o jakość rozwiązania częściowego (**step-by-step pheromone update**)
  - po zakończeniu konstrukcji rozwiązania w oparciu o jakość rozwiązania dopuszczalnego (**online delayed pheromone update**)

Rozszerzenia podstawowego algorytmu (**Ant Colony System**):

- rozszerzenie algorytmy o nadrzędną procedurę sterującą:
  - lokalne przeszukiwanie dla wybranych rozwiązań skonstruowanych przez mrówki
  - dodatkowe wzmacnianie śladu feromonowego dla komponentów występujących w dobrych rozwiązaniach (**offline pheromone update**)
- różne definicje funkcji przejść:
  - probabilistyczna
  - zachłanna

Warunki stopu

- limit czasu procesora
- maksymalna liczba iteracji / wygenerowanych rozwiązań (mrówek)
- maksymalna liczba iteracji, w których najlepsze rozwiązanie nie uległo zmianie
- osiągnięcie rozwiązania s_a o $f(s_a)$ poniżej założonego progu

TESTOWANIE I OCENA METAHEURYSTYK

W przypadku metaheurystyk często brak ścisłych informacji dotyczących **oszacowania**:

- złożoności obliczeniowej,
- najgorszego przypadku,
- zbieżności metody.

Testowanie metaheurystyk to:

- badanie wpływu:
  - parametrów problemu (rozmiaru problemu, rozkładu danych wejściowych)
  - parametrów metody (strojenie, wartości parametrów zależne lub niezależne od parametrów problemu)
- porównanie metaheurystyki z innymi metodami rozwiązującymi ten sam problem
- rezygnacja z długich uruchomień na rzecz licznych testów
- wykonanie pojedynczych testów dla bardzo dużych instancji

Ocena efektywności metaheurystyki dokonywana jest w oparciu o:

- jakość rozwiązań - porównanie z:
  - rozwiązaniem optymalnym
  - ograniczeniem dolnym (górnym)
  - rozwiązaniem losowym
  - rozwiązaniami generowanymi przez inne metody

- nakład obliczeniowy:
  - całkowity czas (liczba iteracji) działania metody
    - user time, system time
    - jednoznaczne określenie, jakie etapy metody są uwzględnione w pomiarze czasu
  - czas (liczba iteracji) do znalezienia najlepszego rozwiązania
  - czas trwania pojedynczej iteracji metody

- wrażliwość na rozkład danych wejściowych:
  - zależność jakości rozwiązania od czasu działania metody
  - odchylenie standardowe jakości rozwiązań

Dane testowe:
- dane rzeczywiste
- dane syntetyczne:
  - nierealistyczne
  - trudniejsze od danych rzeczywistych
  - modelujące różne typy instancji
- dane referencyjne (benchmarks)

Wnioski z eksperymentu obliczeniowego powinny dotyczyć m.in. wpływu:
- czasu obliczeń na jakość rozwiązań
- rozmiaru problemu na czas obliczeń
- wartości parametrów sterujących na czas obliczeń i jakość rozwiązań

Wyniki eksperymentów muszą być uzupełnione danymi dotyczącymi **środowiska obliczeniowego** tzn.:
- sprzętu (typu komputera, rozmiaru pamięci operacyjnej, rodzaju i liczby procesorów, ...)
- oprogramowania (systemu operacyjnego, języka programowania, kompilatora i jego konfiguracji)

Nowa propozycja meteheurystyki jest sukcesem, jeśli jest:

- szybsza
- dokładniejsza
- szybsza i dokładniejsza
- mniej wrażliwa na rozkład danych wejściowych
- prostsza w implementacji
- ogólniejsza
- innowacyjna
- dostarcza ogólnej wiedzy o problemie lub metaheurystyce
- znacząca pod względem teoretycznym