Algorytmy ewolucyjne i populacyjne

Andrzej Jaszkiewicz

„Akademia Innowacyjnych Zastosowań Technologii Cyfrowych (AI Tech)”
projekt finansowany ze środków Programu Operacyjnego Polska Cyfrowa POPC.03.02.00-00-0001/20

Fundusze Europejskie Polska Cyfrowa
Rzeczpospolita Polska
Unia Europejska Europejski Fundusz Rozwoju Regionalnego

Ewolucja biologiczna

- Przekazywanie cech
- Krzyżowanie cech pomiędzy osobnikami i efektywnie w populacji
- Mutacje
- Dobór naturalny
- Dryft genetyczny

Ewolucja biologiczna a optymalizacja

- Osobnik - Rozwiązanie
- Populacja - Zbiór rozwiązań
- Genotyp - Reprezentacja rozwiązania
- Fenotyp - Wartość funkcji celu i inne parametry (np. wartości ograniczeń)
- Mutacja - Konstrukcja nowego rozwiązania poprzez niewielką modyfikację innego rozwiązania
- Krzyżowanie - Krzyżowanie/rekombinacja – konstrukcja nowego rozwiązania poprzez połączenie cech dwóch rozwiązań
- Dobór naturalny - Selekcja dobrych rozwiązań
- Przystosowanie (fitness) - Funkcja celu
- Poprawa przystosowania - Optymalizacja funkcji celu
- Dryft genetyczny - Zbieżność i losowe błądzenie populacji

Ogólny schemat algorytmu ewolucyjnego

Wygeneruj zrandomizowaną populację początkową X

powtarzaj

X_1 := Krzyżowanie (X)

X_1 := Mutacja (X_1)

X := Selekcja ($X \cup X_1$)

do spełnienia warunków stopu

Zwróć najlepsze wygenerowane rozwiązanie

Kodowanie genów w biologii

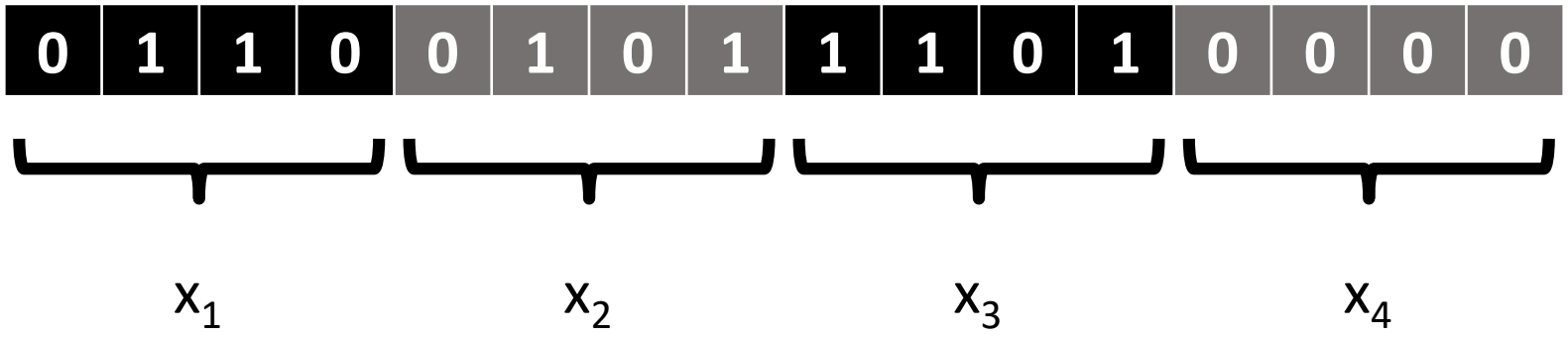
- DNA
- Alfabet czteroliterowy – cztery zasady A, G, T, C
- Trzy zasady kodują jeden aminokwas w sposób nadmiarowy np. kodony AAA i AAG kodują aminokwas lizynę

Algorytmy genetyczne

- Algorytmy ewolucyjne z kodowaniem binarnym
- Rozwiązanie reprezentowane jako ciąg zer i jedynek
- Zawsze możliwe (z pewną dokładnością), ale nie zawsze naturalne

Kodowanie binarne zmiennych liczbowych

Np.:



Przykład kodowania binarnego dla problemu plecakowego

- Pozycja dla każdego elementu
- 0 – element nie został wybrany
- 1 – element został wybrany

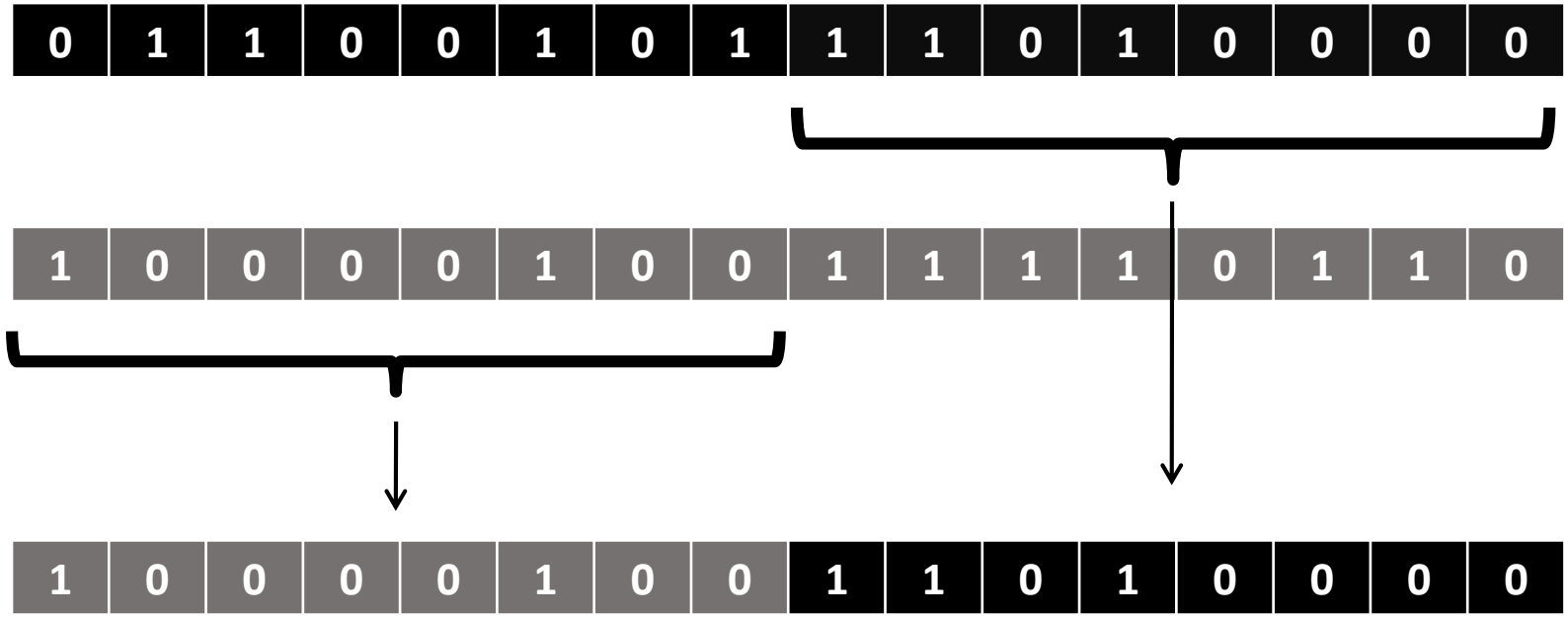
- Nie każdy ciąg binarny koduje dopuszczalne rozwiązanie – waga elementów może przekraczać pojemność

Przykład kodowania binarnego dla problemu komiwojażera

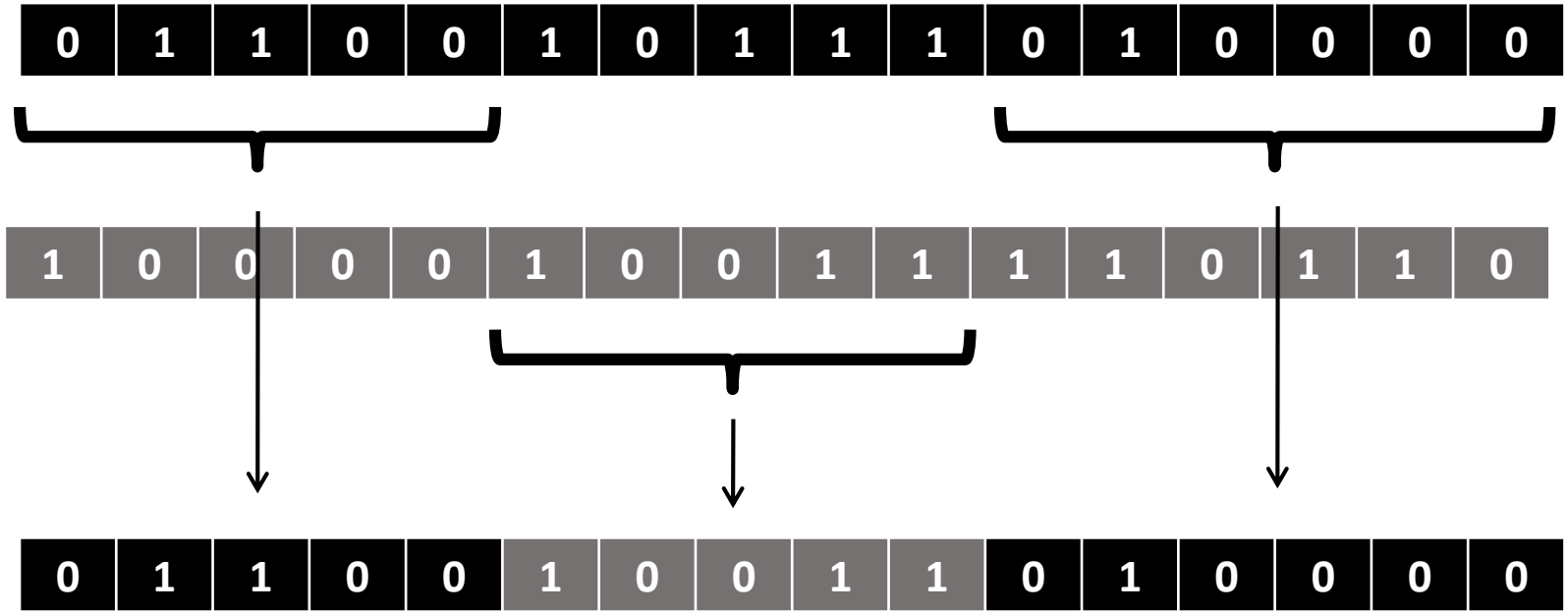
- Pozycja dla każdego łuku
- 0 krawędź nie znajduje się w rozwiązaniu
- 1 krawędź znajduje się w rozwiązaniu
- Mówiąc inaczej – macierz koincydencji rozwinięta w wektor

- Nie każdy ciąg binarny koduje dopuszczalne rozwiązanie – krawędzie mogą nie tworzyć cyklu Hamiltona
- Długość ciągu $O(n^2)$

Krzyżowanie jednopunktowe

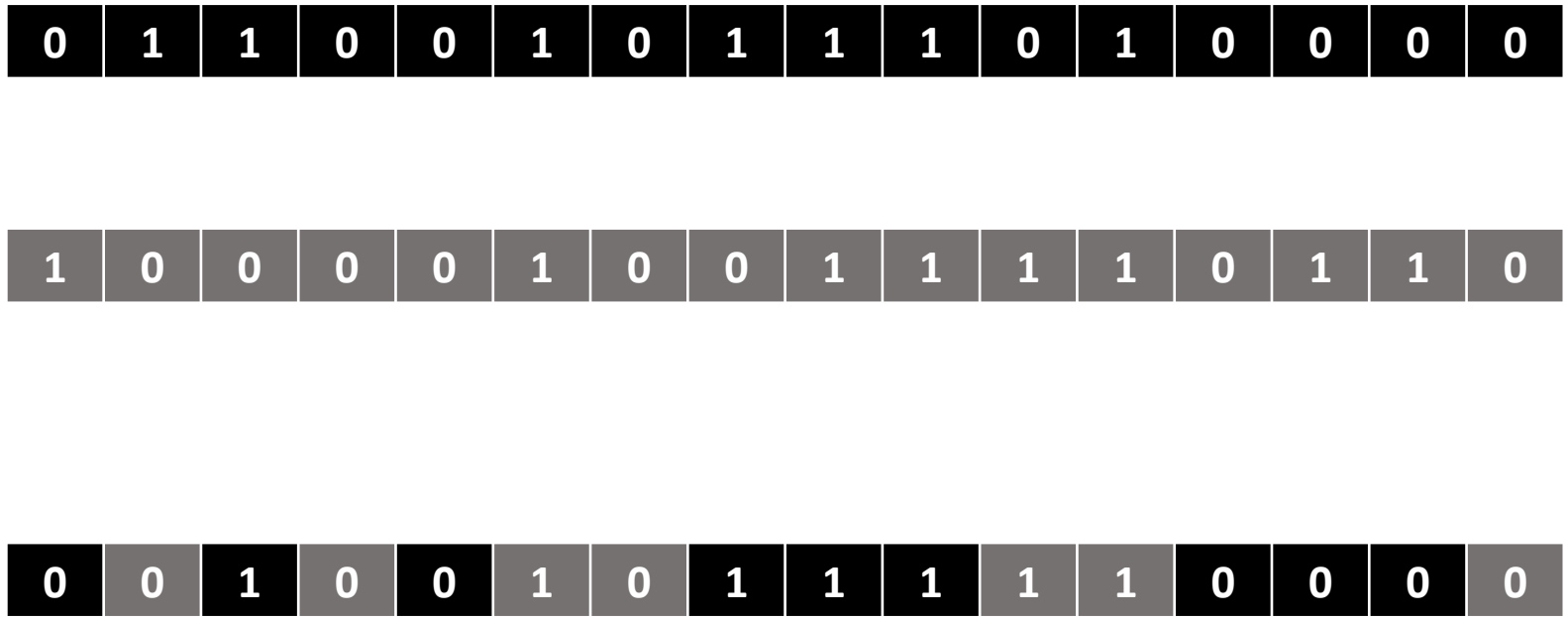


Krzyżowanie wielopunktowe



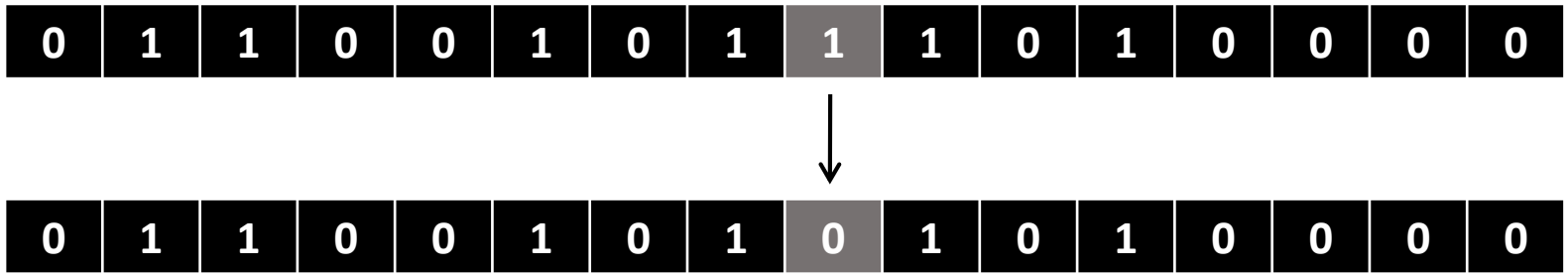
Krzyżowanie równomierne – uniform crossover

- Każda pozycja niezależnie losowa z jednego z rodziców



Mutacja – losowa zmiana rozwiązania

- Mutacja jednego bitu



- Prawdopodobieństwo mutacji – prawdopodobieństwo zmiany jednego bitu
  - Np. prawdopodobieństwo 0,01 oznacza średnio zmianę jednego bitu na 100

Fitness - dostosowanie

- Przeskalowana (w ogólności) funkcja celu $\omega(\mathbf{x})$
- Z reguły wartość maksymalizowana
- Często skalowanie w zakresie [0, Max]

Selekcja ruletkowa

- Oblicz całkowite dostosowanie
  - $F_{tot} = \sum_{i=1}^{pop\_size} \omega(\mathbf{x})$
- Oblicz prawdopodobieństwo wyboru każdego rozwiązania
  - $p_i = \frac{\omega(\mathbf{x})}{F_{tot}}$

Algorytm selekcji ruletkowej

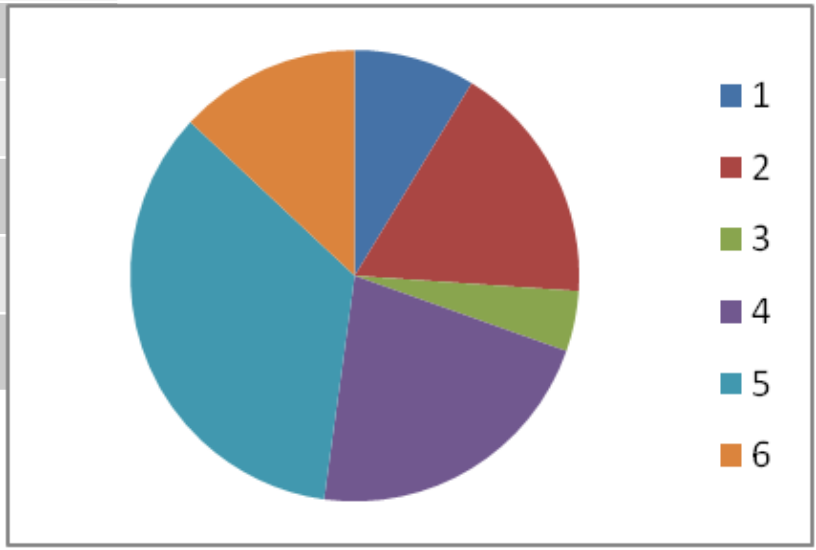
Powtarzaj *pop_size* razy

wygeneruj liczbę losową *r* z przedziału [0,1) z rozkładem równomiernym

wybierz rozwiązanie *j* dla którego $\sum_{i=1}^{j-1} p_i \leq r \leq \sum_{i=1}^{j} p_i$ (dopuszczalne są powtórzenia)

Selekcja ruletkowa przykład

Rozwiązanie	Dostosowanie rozwiązania	P_i
1	2	0,087
2	4	0,174
3	1	0,043
4	5	0,217
5	8	0,348
6	3	0,130
Całkowite dostosowanie	23	



Selekcja turniejowa - algorytm

Powtarzaj *pop_size* razy

wybierz losowo *K* rozwiązań do turnieju – rozkład równomierny

wybierz najlepsze rozwiązanie z grupy turniejowej (dopuszczalne powtórzenia)

- Bardzo często *K*=2
- Im większe *K* tym większa presja selekcyjna (presja na wybór lepszych rozwiązań)

Selekcja elitarna

- Wybór *pop_size* najlepszych rozwiązań
- Możliwe unikanie powtarzających się rozwiązań – kopii
- Możliwość hybrydyzacji z innymi mechanizmami (ruletkowa, turniejowa) – częściowa elitarność
  - Wybór L < P najlepszych rozwiązań. Pozostałe w inny sposób
- Często prowadzi do (zbyt) szybkiej zbieżności

Przedwczesna zbieżność i dryft genetyczny

- Przedwczesna zbieżność – zbieżność do rozwiązań innych niż optimum globalne
- Dryft genetyczny
  - Iteracyjnie powtarzane krzyżowanie/rekombinacja prowadzi do upodabniania się rozwiązań w populacji połączonego z losowym błądzeniem
  - Jedna z kluczowych sił napędowych ewolucji

Zapobieganie przedwczesnej zbieżności – zwiększanie różnorodności populacji

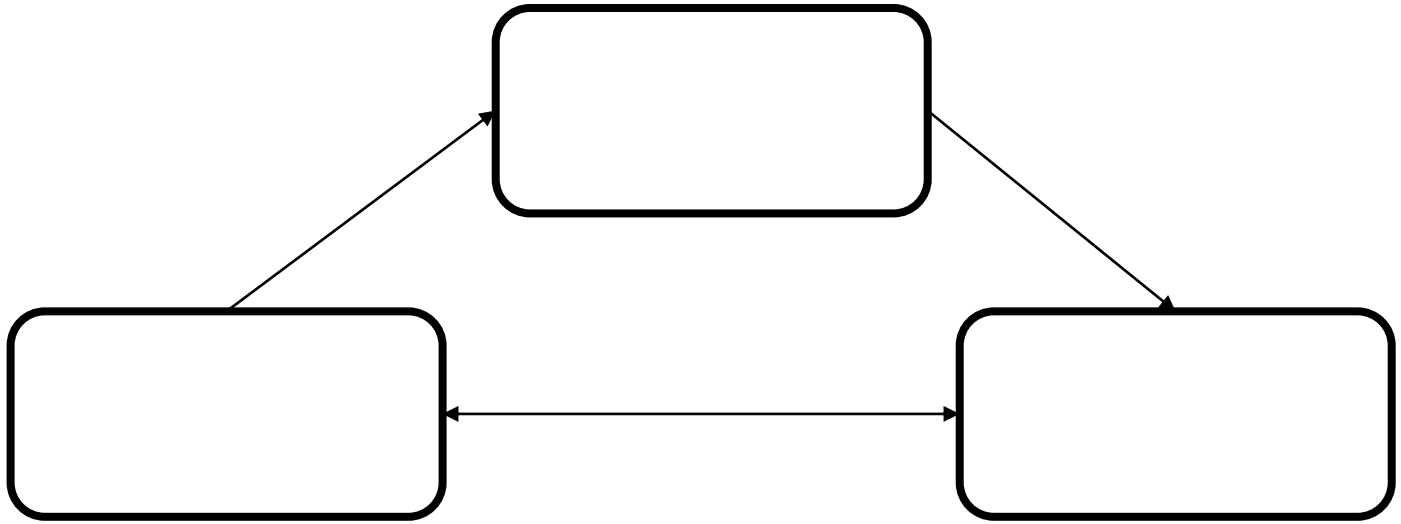
- Zwiększanie wielkości populacji
  - Wydłuża czas obliczeń
  - Szybkie nasycenie – powyżej pewnej wielkości populacji brak zauważalnych efektów przy dalszym zwiększaniu
- Mechanizmy zwiększania różnorodności - modyfikacja mechanizmów selekcji
- Modele wyspowe populacji

Mechanizmy zwiększania różnorodności

- Eliminowanie (nie akceptowanie) kopii
  - Takich samych rozwiązań lub takich samych wartości funkcji celu
- Crowding (zapobieganie tłumowi?) – potomek konkuruje ze swoimi rodzicami
- Ograniczona selekcja turniejowa (Restricted Tournament Selection) – potomek konkuruje z najbliższym (w sensie jakiejś miary odległości, np. Hamminga) rozwiązaniem w populacji
- Współdzielenie przystosowania (fitness sharing) – podobne rozwiązania współdzielą „zasoby” i ich przystosowanie jest obniżane
- Clearing – potomek konkuruje ze wszystkimi rozwiązaniami w granicach pewnego promienia, przeżywa tylko jeden zwycięzca
  - Problem ustawiania promienia – radius problem

Modele wyspowe (Island models)

- Populacja podzielona jest na kilka (prawie) rozłącznych populacji ewoluujących niezależnie
- Co jakiś czas pewne rozwiązania (np. najlepsze z każdej populacji) migrują do innej populacji (raczej kopiowanie) – na inną wyspę



Pojęcie schematu

Wspólny schemat

Do ciągu o długości n pasuje 2^n schematów

Rząd schematu

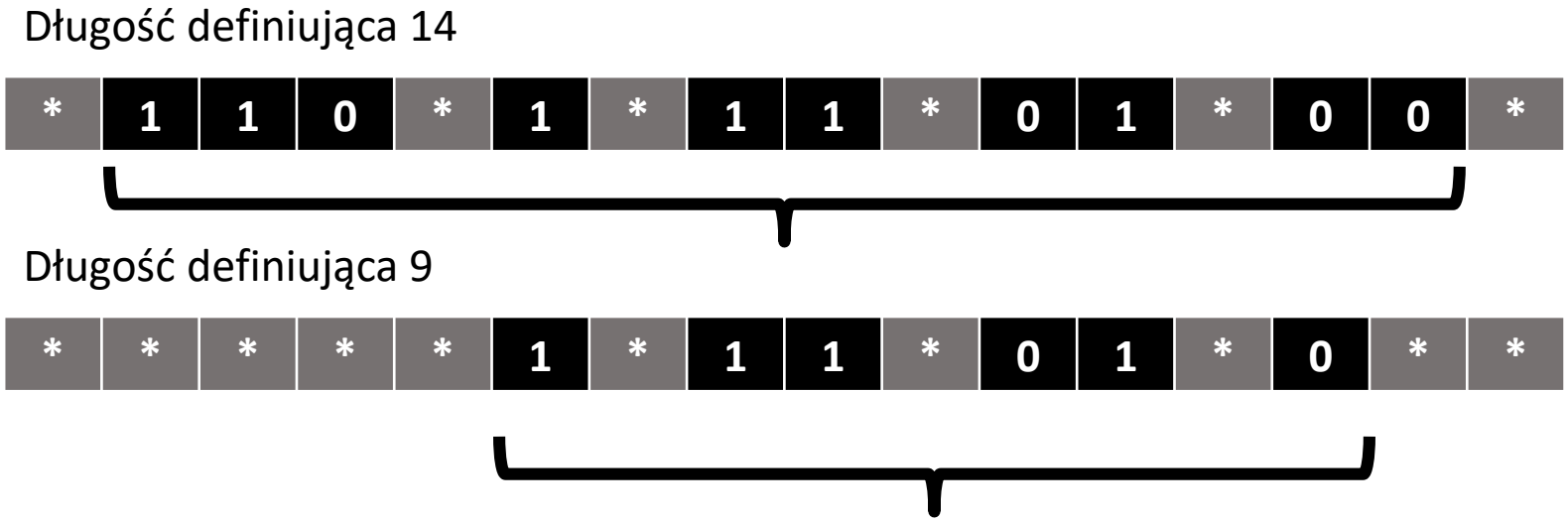
- Liczba ustalonych pozycji - 0 lub 1

Rząd 12

Rząd 8

Długość definiująca schematu

- Odległość pomiędzy pierwszą a ostatnią pozycją ustaloną schematu



Ewolucja schematów

- Algorytm ewolucyjny:
  - Ewolucja rozwiązań
  - Także ewolucja schematów, których jest znacznie więcej niż rozwiązań
- Selekcja
  - Dla selekcji ruletkowej średnia liczba rozwiązań pasujących do schematu S zmienia się w populacji tak, jak stosunek dopasowania schematu do średniego dopasowania populacji
- Krzyżowanie
  - Większe szanse „przeżycia” mają krótsze schematy. Krzyżowanie wielopunktowe, a szczególnie równomierne, w większym stopniu niszczy schematy
- Mutacja
  - Większe szanse „przeżycia” maję schematy o mniejszym rzędzie
- Algorytm ewolucyjny ma sens, jeżeli można wyróżnić złe i dobre schematy

Twierdzenie o schematach Hollanda

- Krótkie, niskiego rzędu i dobrze przystosowane schematy rozprzestrzeniają się w kolejnych pokoleniach zgodnie z wykładniczym prawem wzrostu

Algorytmy ewolucyjne – kodowanie naturalne

- Więcej liter alfabetu
- Liczby naturalne i rzeczywiste
- Listy, sekwencje, permutacje
- Macierze
- Drzewa
- Zbiory
- … dowolne struktury danych

Zalety kodowania naturalnego

- Łatwiejsze krzyżowanie – rekombinacja
  - Utworzenie nowego (lub kilku) nowego rozwiązania (potomka) na podstawie dwóch rodziców łączącego (rekombinującego) cechy rodziców
- Bardziej „sensowne” krzyżowanie – zachowywanie ważnych cech rozwiązań
- Łatwiejsze (często gwarantowane) zachowywanie ograniczeń

Rekombinacja porządkowa dla list - Order crossover (OX)

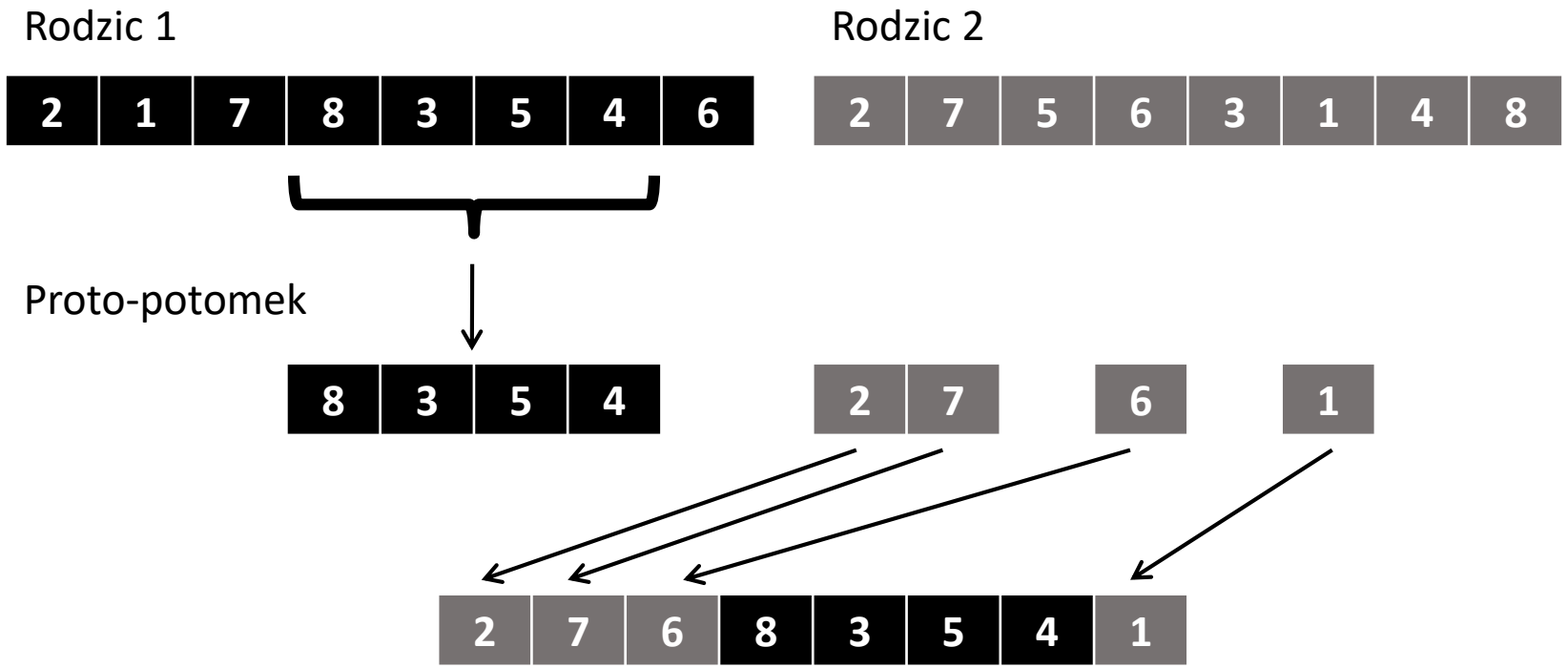
Wybierz podciąg z rodzica 1

Skopiuj wybrany podciąg do potomka

Usuń umieszczone już w potomku elementy z rodzica 2

Dodawaj kolejne elementy z rodzica 2 do potomka w kolejności występowania w rodzicu 2

Rekombinacja porządkowa dla list - OX



Inny przykład rekombinacji dla list

powtarzaj dla każdej pozycji

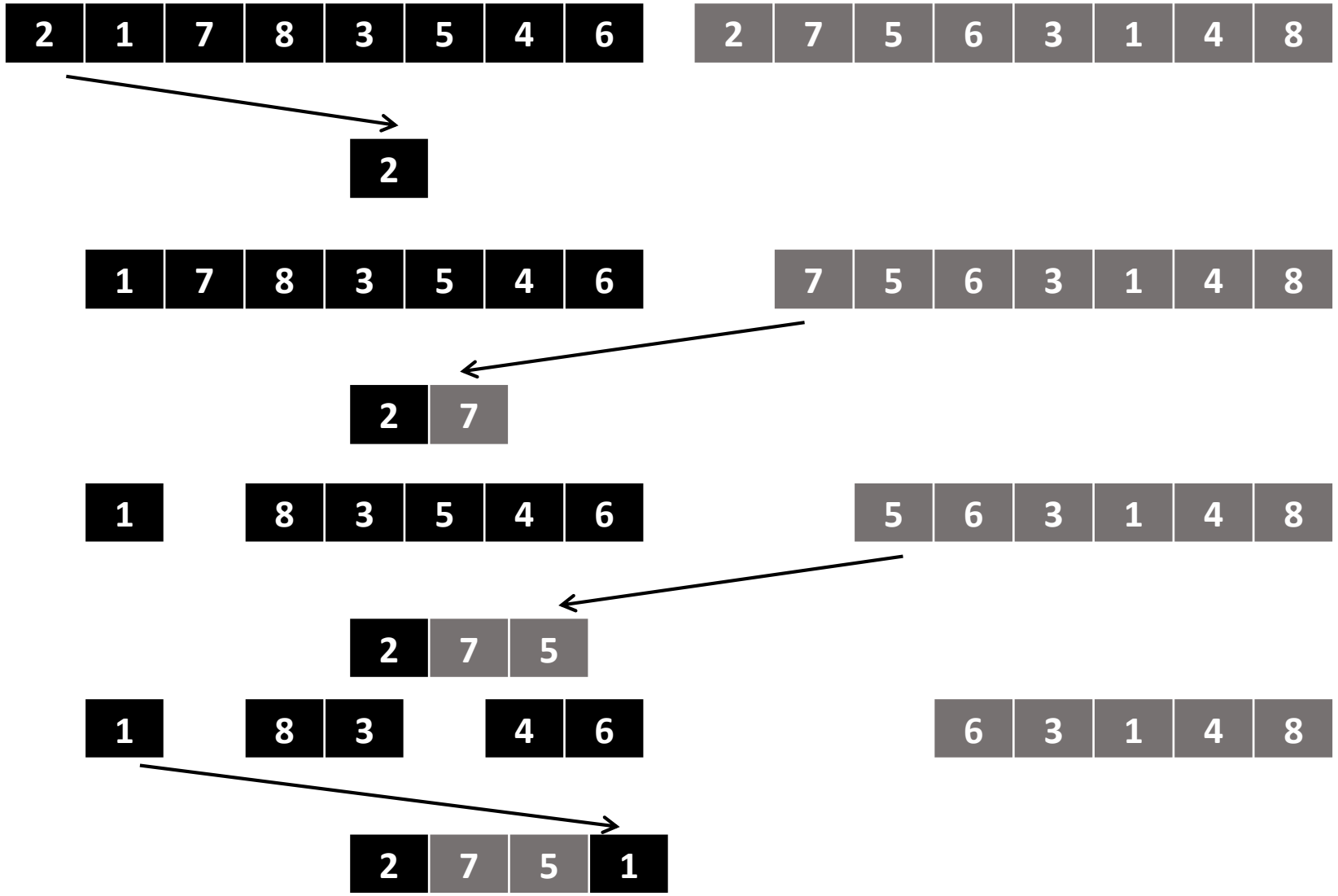
wylosuj rodzica 1 lub 2

wybierz z wylosowanego rodzica pierwszy element

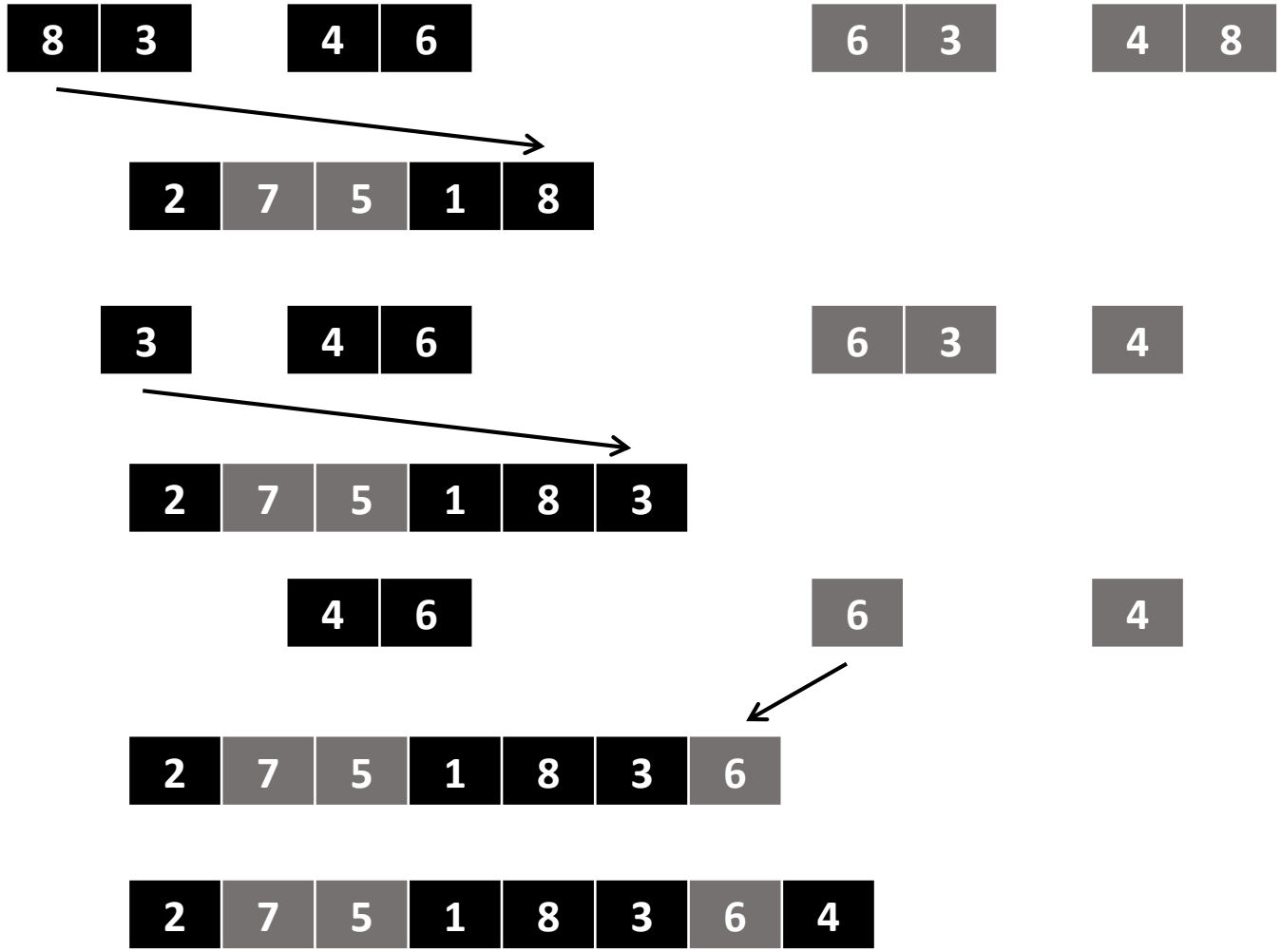
usuń wybrany element z obu rodziców

dodaj wybrany element na koniec potomka

Przykład


2 1 7 8 3 5 4 6
2 7 5 6 3 1 4 8
2
1 7 8 3 5 4 6
7 5 6 3 1 4 8
2 7
1 8 3 5 4 6
5 6 3 1 4 8
2 7 5
1 8 3 4 6
6 3 1 4 8
2 7 5 1

Przykład c.d.



Inne operatory rekombinacji dla list

- Partially mapped crossover (PMX)
  - Jak w OX zaczynamy od skopiowana podciągu z Rodzica 1, elementy z Rodzica 2 wstawiamy starając się zachowywać ich bezwzględne pozycje (najpierw wstawiamy te, które da się wstawić na wolne pozycje
- Edge recombination crossover (ERX)
  - Wybieramy dowolny element. Kolejny element jest kolejnym wierzchołkiem z jednego (losowo wybranego) z rodziców. Jeżeli oba kolejne elementy są już wybrane, wierzchołek wybieramy losowo
- Cycle crossover (CX)
  - Wybieramy pierwszy element z Rodzica 1 i wstawiamy go na pozycję jaką w Rodzicu 2. Potem wybieramy element z Rodzica 2, który był na pozycji ostatniego elementu wybranego z Rodzica 1, itd..

Operator rekombinacji może być dość skomplikowanym algorytmem- np. rekombinacja trasowa (route-based) dla VRP

Rozwiązanie potomne *O* ← puste rozwiązanie

Wybierz losowo rodzica *p*

powtarzaj

- Wybierz z *p* trasę *r* o największej liczbie wspólnych, nieprzydzielonych jeszcze wierzchołków
- Usuń z *r* wierzchołki, które zostały już dodane do *O*
- Dodaj trasę *r* do *O*

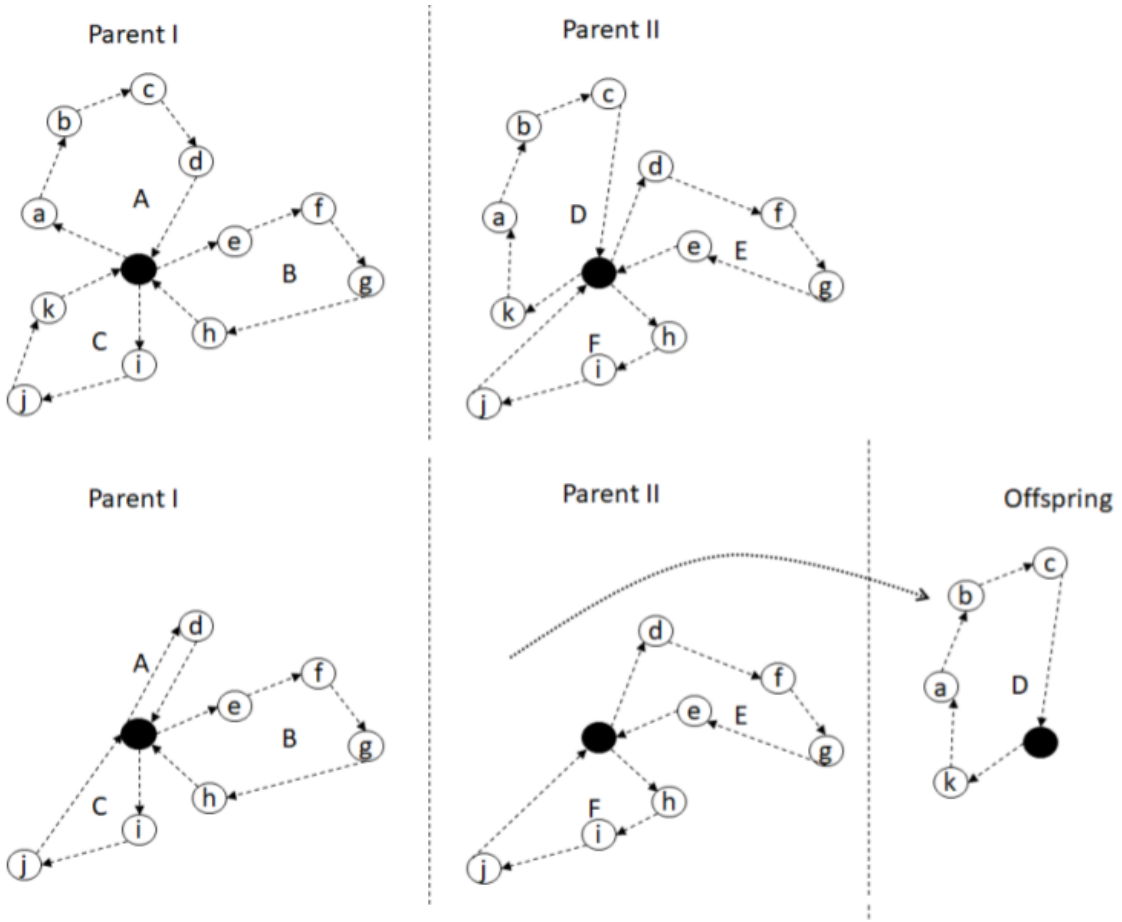
dopóki Liczba tras w potomku *O* nie jest równa liczbie tras w p_A

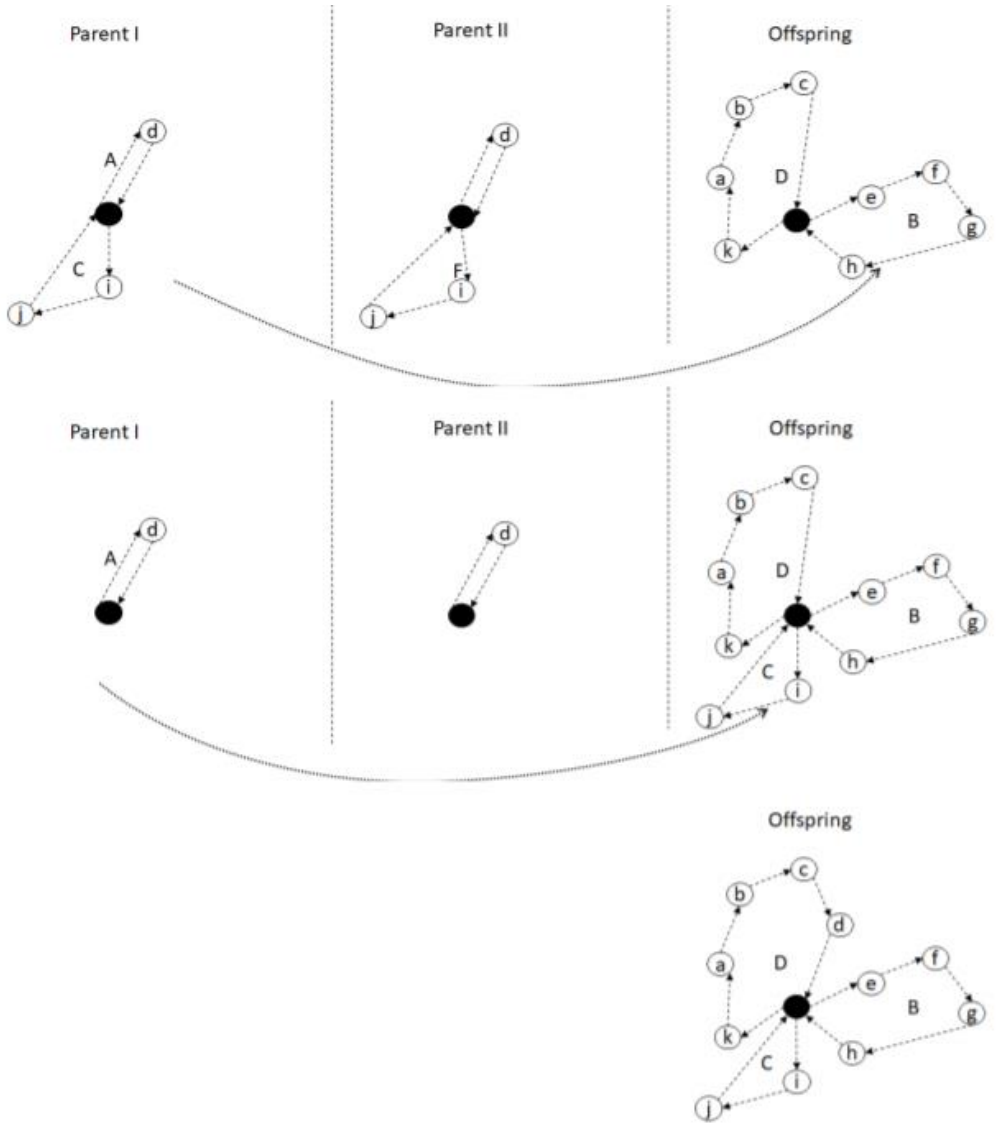
Zablokuj wszystkie wspólne łuki i krawędzie w potomku

Wstaw nieprzydzielone jeszcze wierzchołki za pomocą procedury zachłannej

zwróć *O*

Rekombinacja trasowa dla VRP





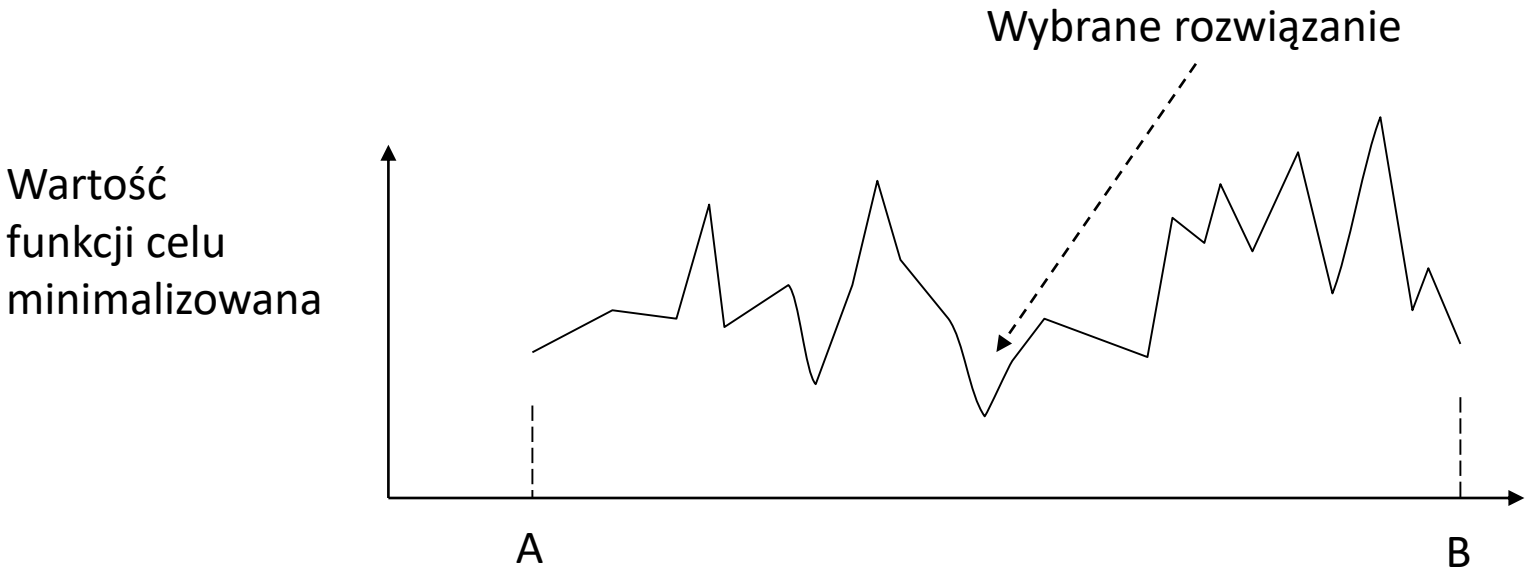
Inny przykład dla VRP - Modified Selective Route Exchange Crossover

- Potraktuj trasy wchodzące w skład obu rodziców jako zbiory wierzchołków
- Zdefiniuj problem maksymalnego pokrycia – wybór zbioru tras o liczności takiej jak w jednym z rodziców z kryterium leksykograficznym:
  - Maksymalizacja pokrytych wierzchołków, potem suma długości wybranych tras
- Rozwiąż problem maksymalnego pokrycia za pomocą prostego hybrydowego algorytmu ewolucyjnego z lokalną i globalną listą tabu (zakazanych rozwiązań) (HAE wewnątrz rekombinacji)
- Dla nadmiarowo przydzielonych wierzchołków (do dwóch tras) wybierz losowo jedną z nich (usuń z drugiej)
- Wstaw nieprzydzielone jeszcze wierzchołki za pomocą procedury zachłannej

Path relinking

- Rodzaj rekombinacji
- Przejdź z rodzica A do B wykonując proste ruchy
  - Może to być lokalne przeszukiwanie startujące z A, gdzie funkcją celu jest odległość do B
  - Leksykograficznie można tez brać pod uwagę wartość funkcji celu, jeżeli dwa lub więcej ruchów dają tę samą odległość do B, wybieramy najlepszy pod względem wartości f. celu
- Zwróć najlepsze rozwiązanie na tej ścieżce
- Może prowadzić do (zbyt) szybkiej zbieżności. Warto stosować dodatkowe mechanizmy dywersyfikacji populacji

Path relinking



Kodowanie vs rekombinacja

- Ten sam efekt można często uzyskać stosując:
  - Prostsze kodowanie i bardziej złożoną rekombinację
    - lub
  - Bardziej złożone kodowanie i prostszą rekombinację

Hybrydowe algorytmy ewolucyjne (HAE)

- Inne nazwy
  - Algorytmy memetyczne – memetic algorithms
  - Genetyczne przeszukiwanie lokalne – genetic local search
  - …
- Hybrydyzacja algorytmów ewolucyjnych i lokalnego przeszukiwania
  - lub ogólniej heurystyk lokalnych
- Celem jest połączenie zalet:
  - Szybkość i możliwość lokalnej poprawy rozwiązań przez przeszukiwanie lokalne
  - Globalność algorytmów ewolucyjnych

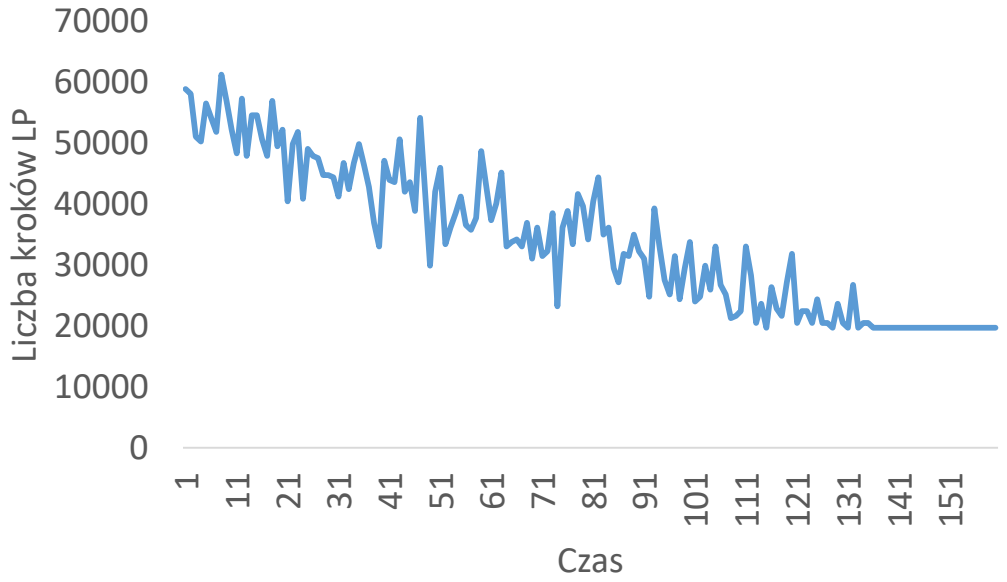
Rożne sposoby hybrydyzacji

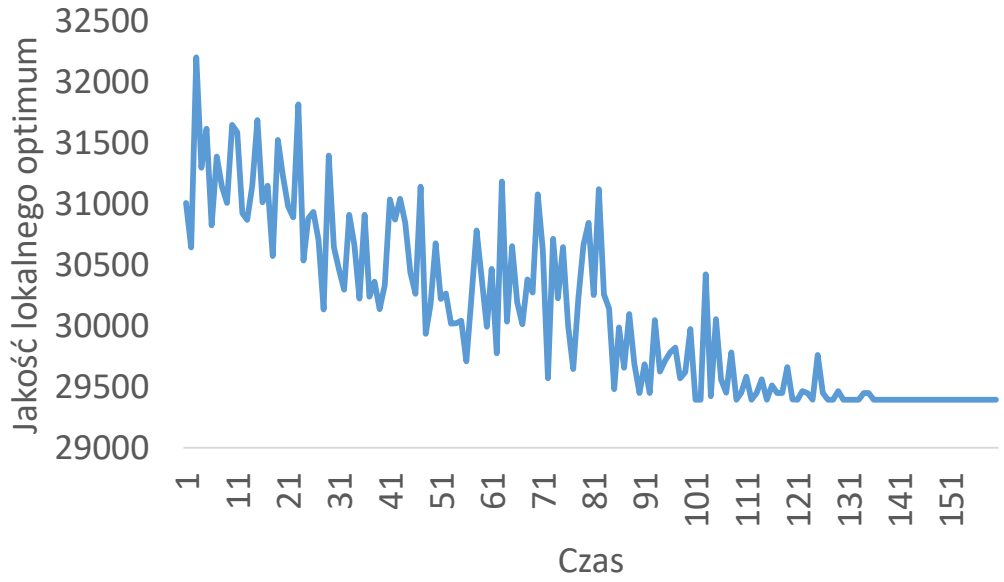
- Najpierw PL potem AE
  - AE startuje z bardzo dobrą populacją złożoną z optimów lokalnych
- Najpierw AE potem PL
  - AE znajduje globalnie dobry region eksplorowany dokładniej przez PL
- **PL w trakcie AE**
  - **Np. PL po każdej (co którejś) rekombinacji i mutacji**

Dwa punkty widzenia (uzasadnienia efektywności) HAE

- Punkt widzenia AE
  - HAE to AE pracujący na ograniczonym zbiorze rozwiązań – tylko zbiorze optimów lokalnych
  - Mniejsza przestrzeń przeszukiwań – bardziej efektywne działanie
  - Przeszukiwanie lokalne zapewnia efektywne ograniczanie przestrzeni przeszukiwań AE do bardzo dobrych rozwiązań
- Punkt widzenia PL
  - Rekombinacja dobrych rozwiązań zapewnia dobry punkt startowy dla przeszukiwania lokalnego, co z kolei daje:
    - Szybszą zbieżność do lokalnego optimum
    - Zbieżność do lepszych rozwiązań

Przykład działania przeszukiwania lokalnego podczas pracy HAE dla TSP – win-win pod względem czasu i jakości





W HAE często stosowana jest selekcja elitarna

- W przypadku zwykłych AE selekcja elitarna prowadzi do (zbyt) szybkiej zbieżności.
- Dodanie przeszukiwania lokalnego często eliminuje ten problem – PL wprowadza dodatkową dywersyfikację rozwiązań uzyskanych po rekombinacji
  - Często nie jest potrzebna jawna mutacja, która może też być wbudowana w zrandomizowany operator rekombinacji
  - Znaczenie ma też fakt, że ze względu na dość długi czas PL, wykonywanych jest mniej iteracji AE (mniej rekombinacji/mutacji)

Steady State evolutionary algorithms

- Brak generacji (pokoleń)
- Potomek może być dodany do populacji (jeżeli spełnia odpowiednie warunki) natychmiast po utworzeniu

HAE z selekcją elitarną i steady state - przykład

Wygeneruj populację początkową **X**

powtarzaj

Wylosuj dwa różne rozwiązania (rodziców) stosując rozkład równomierny

Skonstruuj rozwiązanie potomne **y** poprzez rekombinację rodziców

y := Lokalne przeszukiwanie (**y**)

jeżeli y jest lepsze od najgorszego rozwiązania w populacji i (wystarczająco) różne od wszystkich rozwiązań w populacji

Dodaj **y** do populacji i usuń najgorsze rozwiązanie

dopóki nie są spełnione warunki stopu

Ogólniejsze ujęcie – algorytm populacyjny uogólniający metody typu HAE, ILS, LNS

Wygeneruj populację początkową **X**

powtarzaj

Wylosuj dwa różne rozwiązania (rodziców) stosując rozkład równomierny

Skonstruuj nowe rozwiązanie **y** poprzez zastosowanie wybranego operatora – np. rekombinacja, perturbacja, destroy-repair

y := Lokalne przeszukiwanie (**y**)

jeżeli y jest lepsze od najgorszego rozwiązania w populacji i (wystarczająco) różnie od wszystkich rozwiązań w populacji

Dodaj **y** do populacji i usuń najgorsze rozwiązanie

dopóki nie są spełnione warunki stopu

Genetic Programming Hyper-Heuristic

- Ewolucji podlegają reguły/heurystyki/kod tworzenia rozwiązań (a nie same rozwiązania)
- Heurystyki te są oceniane na zbiorze instancji uczących, oceniane na zbiorze instancji testowych i stosowane do nowych instancji

Parametry w algorytmach ewolucyjnych

- Liczba generacji – czas obliczeń
- Wielkość populacji
  - Mniejsza populacja – szybsza zbieżność
  - Większa populacja – lepsza jakość w dłuższym czasie
- Rodzaj selekcji i aktualizacji populacji
  - Siła presji selekcyjnej – intensyfikacja vs dywersyfikacja

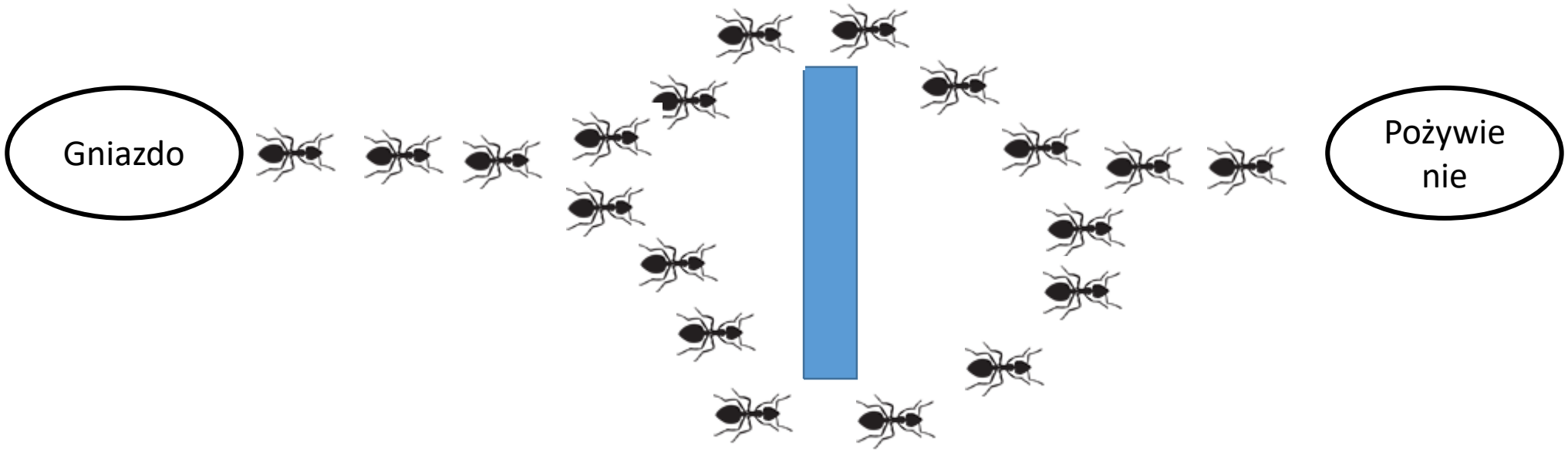
Algorytmy inspirowane biologicznie i nie tylko…

- **Evolution-based**
  - Differential evolution, Evolution strategies, Genetic/evolutinary algorithms, Genetic programming…
- **Swarm-based**
  - Ant colony optimization, Artificial Bee Colony, Bat Algorithm, Crow search algorithm, Cuckoo search, Firefly Algorithm, Flower Pollination Algorithm, Grey Wolf Optimizer, Krill Herd Algorithm, Moth-Flame Optimization Algorithm, Particle Swarm Optimization, Social Spider Optimization, Whale Optimization Algorithm…
- **Physics-based**
  - Electromagnetism-like mechanism, Gravitational Search Algorithm, Simulated annealing, Sine Cosine Algorithm, States of matter search…
- **Human-based**
  - Fireworks Algorithm, Harmony Search, Imperialist Competitive Algorithm, Tabu search…

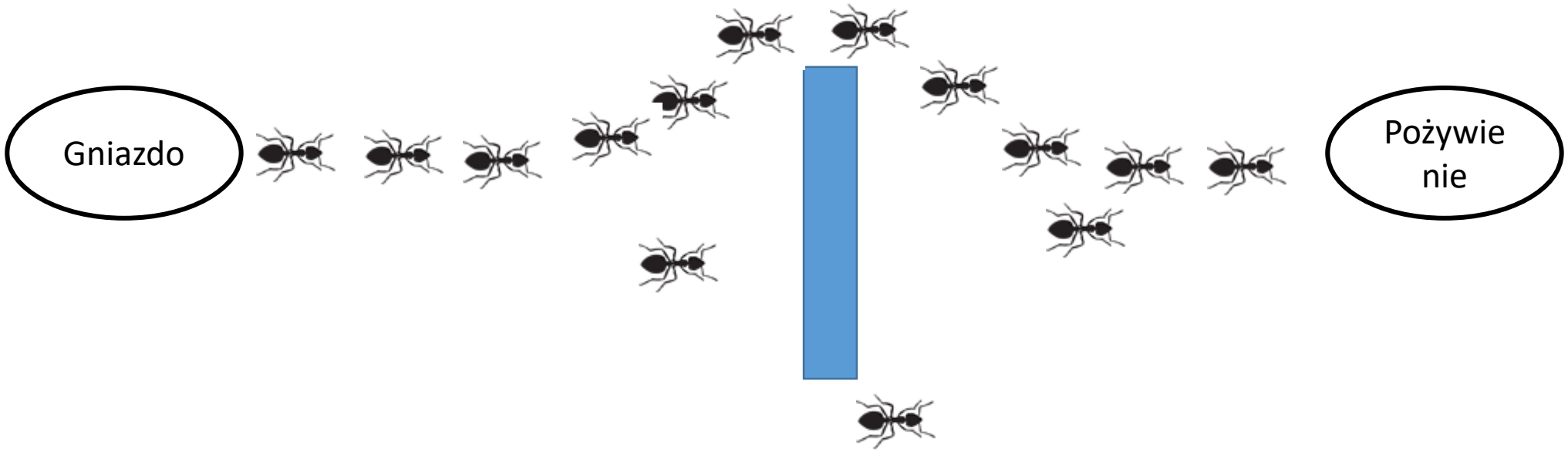
The ZOO of (nature-inspired) metaheuristics

- Fausto, Fernando; Reyna-Orta, Adolfo; Cuevas, Erik; et al., From ants to whales: metaheuristics for all tastes, ARTIFICIAL INTELLIGENCE REVIEW Volume: 53 Issue: 1 Pages: 753-810 Published: JAN 2020
- Fathollahi-Fard, Amir Mohammad; Hajiaghaei-Keshteli, Mostafa; Tavakkoli-Moghaddam, Reza, Red deer algorithm (RDA): a new nature-inspired meta-heuristic, SOFT COMPUTING Volume: 24 Issue: 19 Pages: 14637-14665 Published: OCT 2020
- Torabi, Shadi; Safi-Esfahani, Faramarz, Improved Raven Roosting Optimization algorithm (IRRO), SWARM AND EVOLUTIONARY COMPUTATION Volume: 40 Pages: 144-154 Published: JUN 2018

Algorytm kolonii mrówek – ant colony



Po pewnym czasie większość mrówek wybiera najkrótszą trasę



Ślad feromonowy

- Mrówki poruszając się pozostawiają ślad feromonowy wyczuwalny przez inne mrówki
- Feromony stopniowo parują - ślad zanika
- Mrówki pozostawiają silniejszy ślad na ścieżkach prowadzących do pożywienia
- Na krótszych ścieżkach parowanie jest wolniejsze
- Silny ślad feromonowy przyciąga inne mrówki

Idea algorytmu kolonii mrówek dla TSP

- Sztuczna mrówka – agent, który porusza się z miasta do miasta
- Mrówki preferują miasta połączone łukami z dużą ilością feromonu
- Mrówki startują z losowo wybranych miast
- Przemieszczają się do nowych miast, modyfikując smugę feromonową na przemierzanych krawędziach (***local trail updating***)
- Po ukończeniu wszystkich tras mrówka, której trasa była najkrótsza modyfikuje krawędzie należące do jej trasy przez dodanie ilości feromonu odwrotnie proporcjonalnej do długości trasy (***global trail updating***)
- Ślad feromonowy stopniowo zanika

Hybrydyzacja algorytmów mrówkowych z lokalnym przeszukiwaniem

- Najlepsze wyniki dają metody hybrydowe
- Po zbudowaniu rozwiązania przez mrówkę uruchamiane jest lokalne przeszukiwanie

Ograniczenia w inteligentnych metodach optymalizacji

- Większość IMO jest definiowana jako metody dla problemów bez ograniczeń
- W praktyce operatory np. sąsiedztwa, krzyżowania, mutacji, mogą prowadzić do powstania rozwiązań niedopuszczalnych
- Np. dla problemu plecakowego – przekroczenie pojemności plecaka
- Np. dla problemu komiwojażera – zbiór łuków nie tworzący cyklu Hamiltona

Możliwe sposoby uwzględniania ograniczeń

- Przeformułowanie problemu/operatorów do wersji bez ograniczeń
- Odrzucanie rozwiązań niedopuszczalnych
- Naprawa rozwiązań niedopuszczalnych
- Kodowanie pośrednie i dekodowanie
- Funkcje kary
- Podejście wielokryterialne

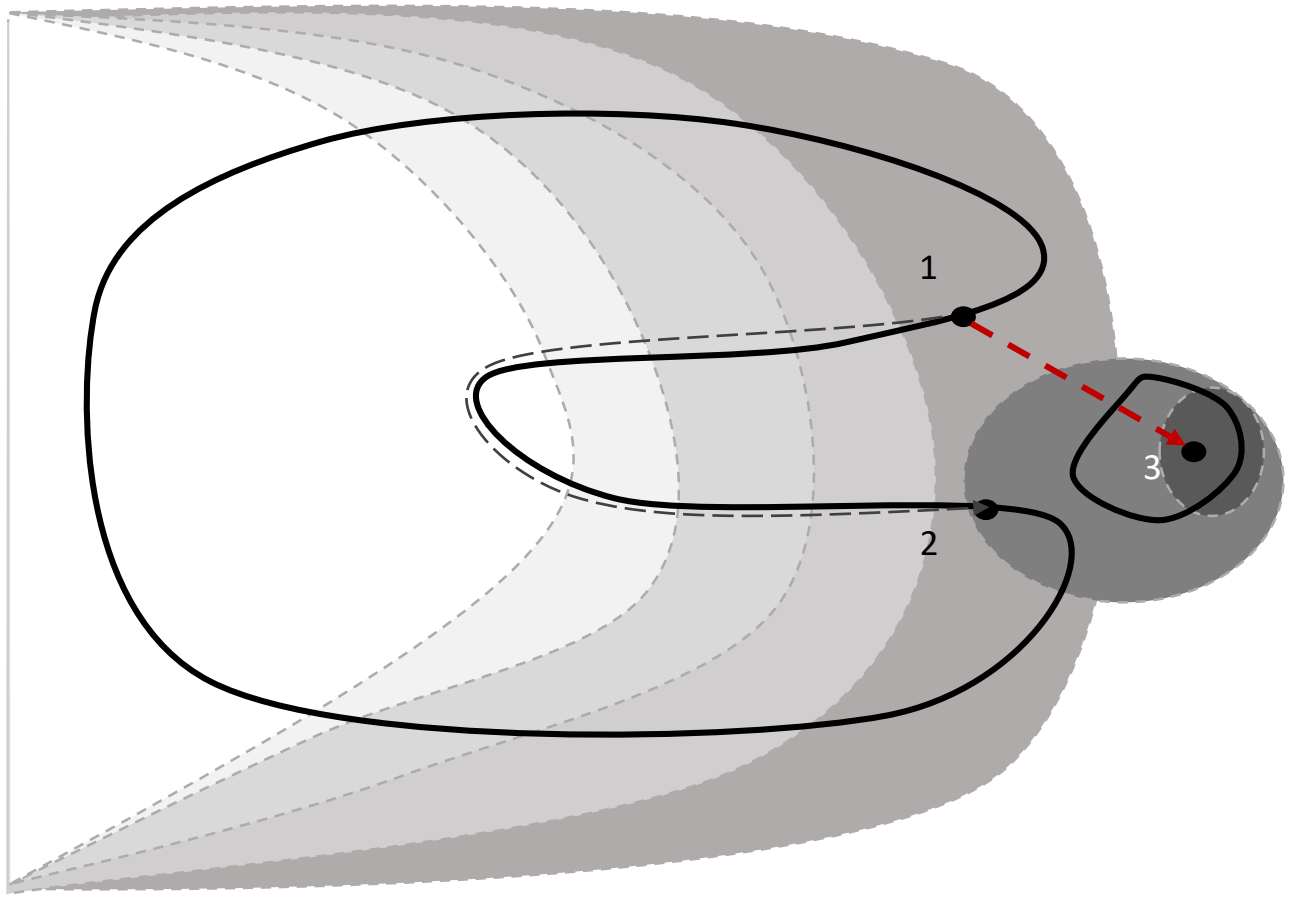
Przeformułowanie problemu/operatorów

- Np. problem komiwojażera ma ograniczenia, jeżeli rozwiązanie jest kodowane jako zbiór łuków/krawędzi, ale jeżeli rozwiązanie jest kodowane jako permutacja wszystkich wierzchołków, to dowolna permutacja definiuje dopuszczalne rozwiązanie
- Łatwo skonstruować operatory działające na permutacjach i tworzące nowe permutacje
- Podejście możliwe w przypadku stosunkowo prostych ograniczeń (np. trudne do stosowania w VRP)

Odrzucanie niedopuszczalnych rozwiązań

- Niedopuszczalne rozwiązania są natychmiast odrzucane
- Nieefektywne dla problemów silnie ograniczonych, gdzie trudne jest uzyskanie rozwiązania dopuszczalnego
- Zmienia krajobraz funkcji celu, np. przejście z pewnego rozwiązania dopuszczalnego do lepszego rozwiązania dopuszczalnego może być niemożliwe lub bardzo trudne

Problemy przy odrzucaniu rozwiązań niedopuszczalnych



Procedury naprawy

- Niedopuszczalne rozwiązanie powstałe w wyniku zastosowania operatora jest naprawiane przez dedykowaną heurystykę
- Procedura naprawy nie powinna zbyt mocno zmieniać rozwiązania
- Procedura naprawy może być kierowana funkcją celu
- Np. problem plecakowy
  - Usuwanie elementów aż do osiągnięcia dopuszczalności
  - Opcja – usuwanie elementów o najgorszym stosunku wartości do wagi
- Np. problem komiwojażera
  - Odbudowa cyklu Hamiltona przy zachowaniu jak największej liczby łuków
  - Opcja – preferowanie dodawania do rozwiązania jak najkrótszych łuków

Kodowanie (reprezentacja) pośrednie

Dekodowanie rozwiązania
Heurystyczna lub dokładna transformacja

Kodowanie pośrednie dla problemu plecakowego

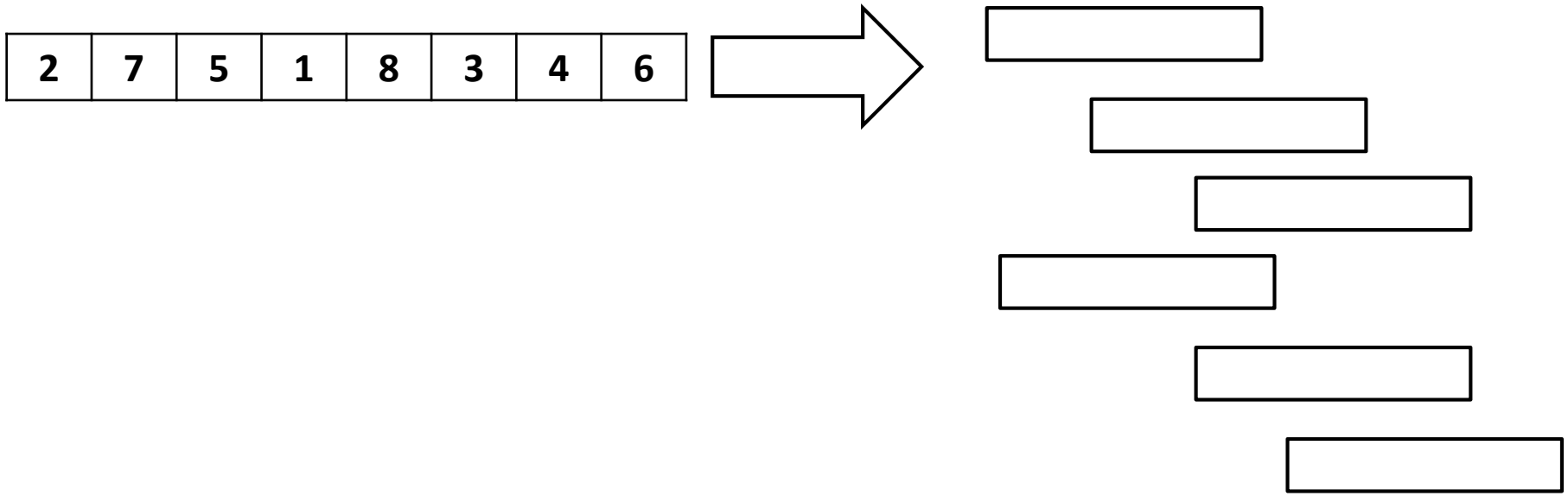
- Reprezentacja pośrednia uporządkowana lista elementów – operatory działające na listach
- Dekodowanie – wybór po kolei elementów z listy, dopóki plecak nie jest zapełniony

Kodowanie pośrednie dla szeregowania zadań

Kodowanie pośrednie –
lista priorytetowa zadań

Heurystyka
priorytetowa

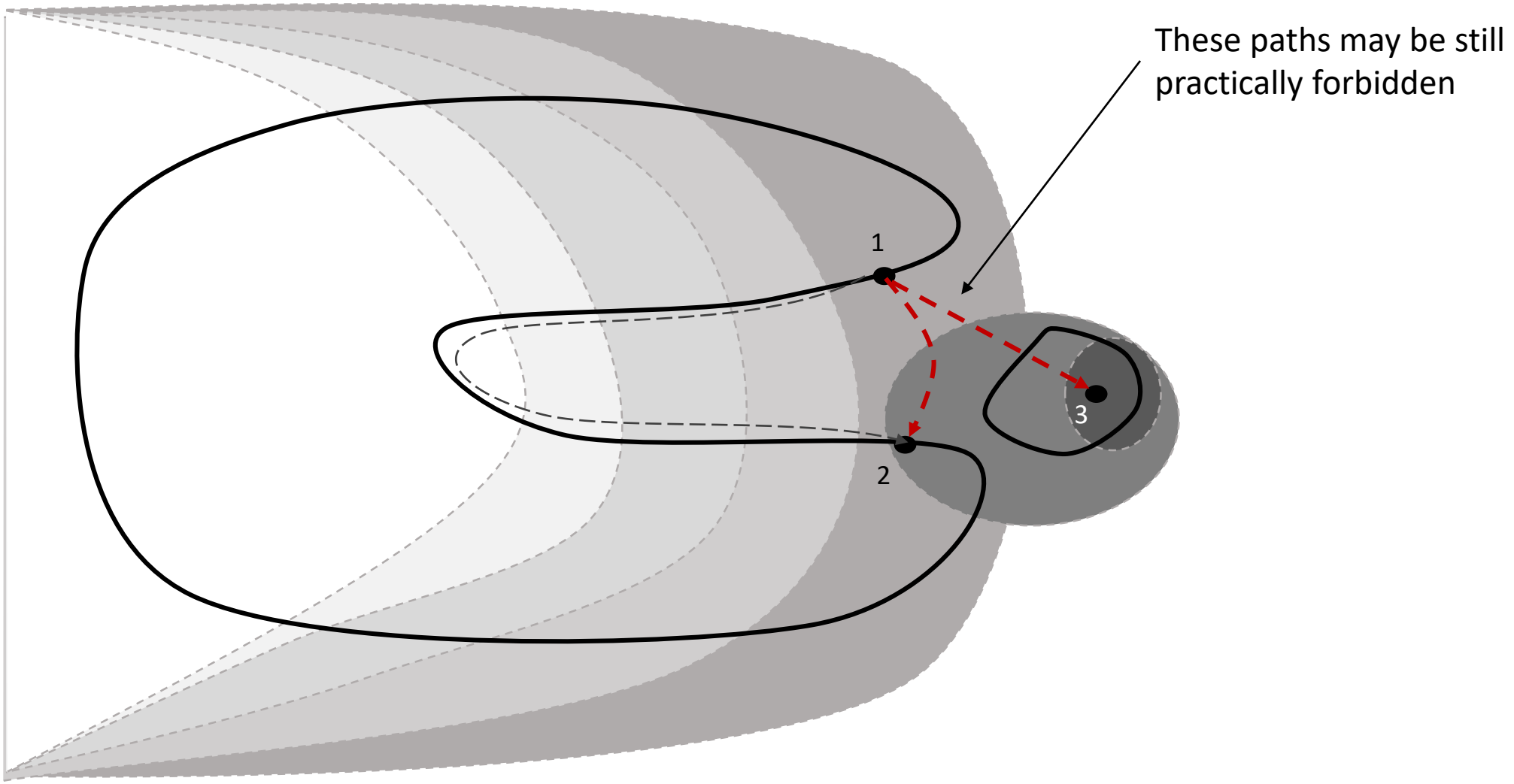
Harmonogram w czasie



Funkcja kary

- Dla każdego ograniczenia (typu ograniczeń) miara wielkości przekroczenia ograniczeń (0 jeżeli ograniczenie jest spełnione), np. o przekroczenie wielkości plecaka - kara
- Rozwiązania niedopuszczalne są traktowane jak dopuszczalne, ale wartość funkcji celu zostaje pogorszona o wartość kar(y)
- Problemy
  - Zbyt mała kara może spowodować ewolucję w kierunku rozwiązań niedopuszczalnych
  - Zbyt duża kara może utrudniać dojście do dobrych rozwiązań – de facto zmienia krajobraz funkcji celu – rozwiązania mocno przekraczające ograniczenia są w praktyce nieosiągalne
- Odrzucanie może być interpretowane jako zastosowanie nieskończonej wagi przekroczenia ograniczeń

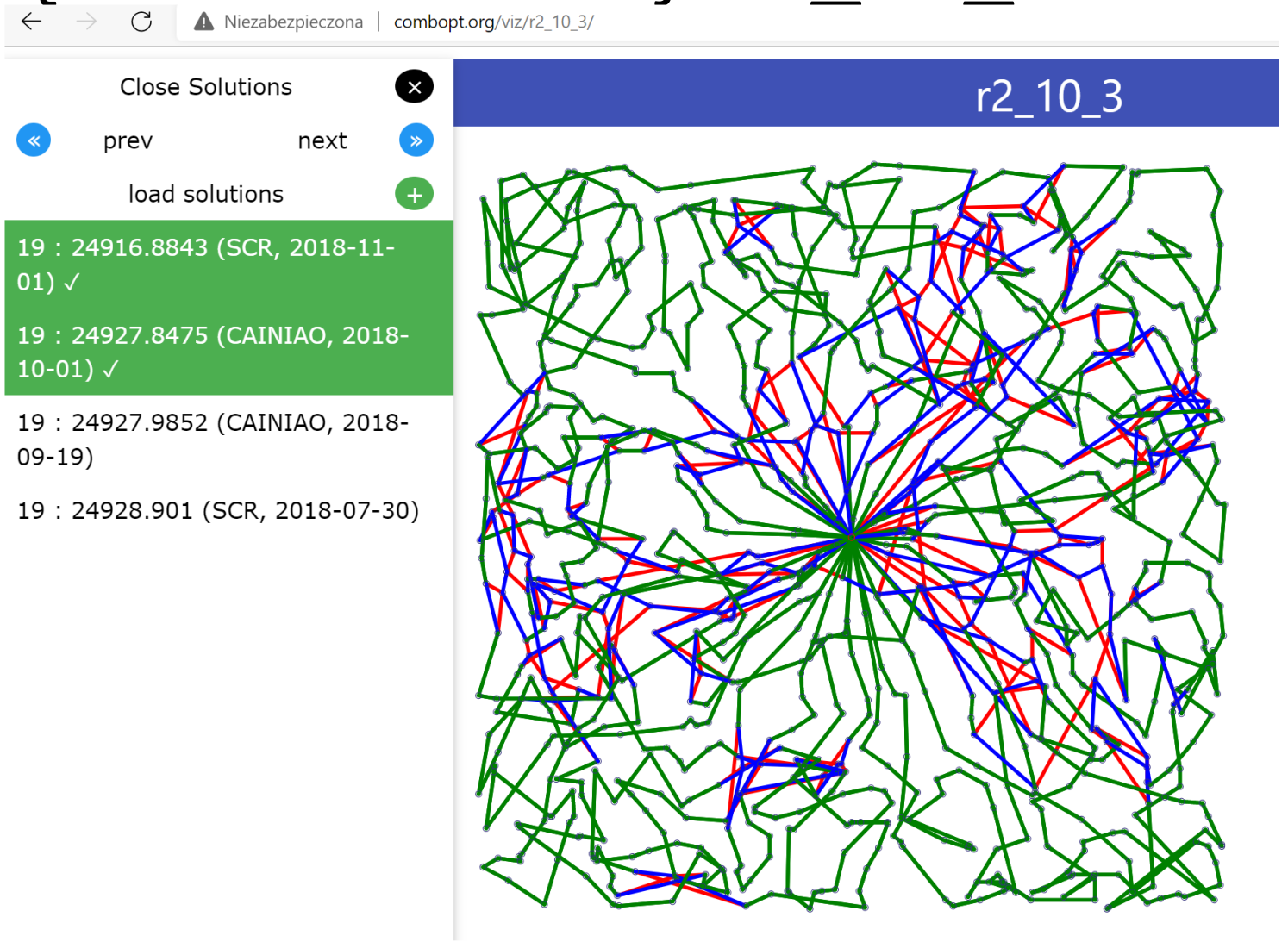
Problemy przy stosowaniu funkcji kary



Nowa, eksperymentalna metoda ustalania wag funkcji kary

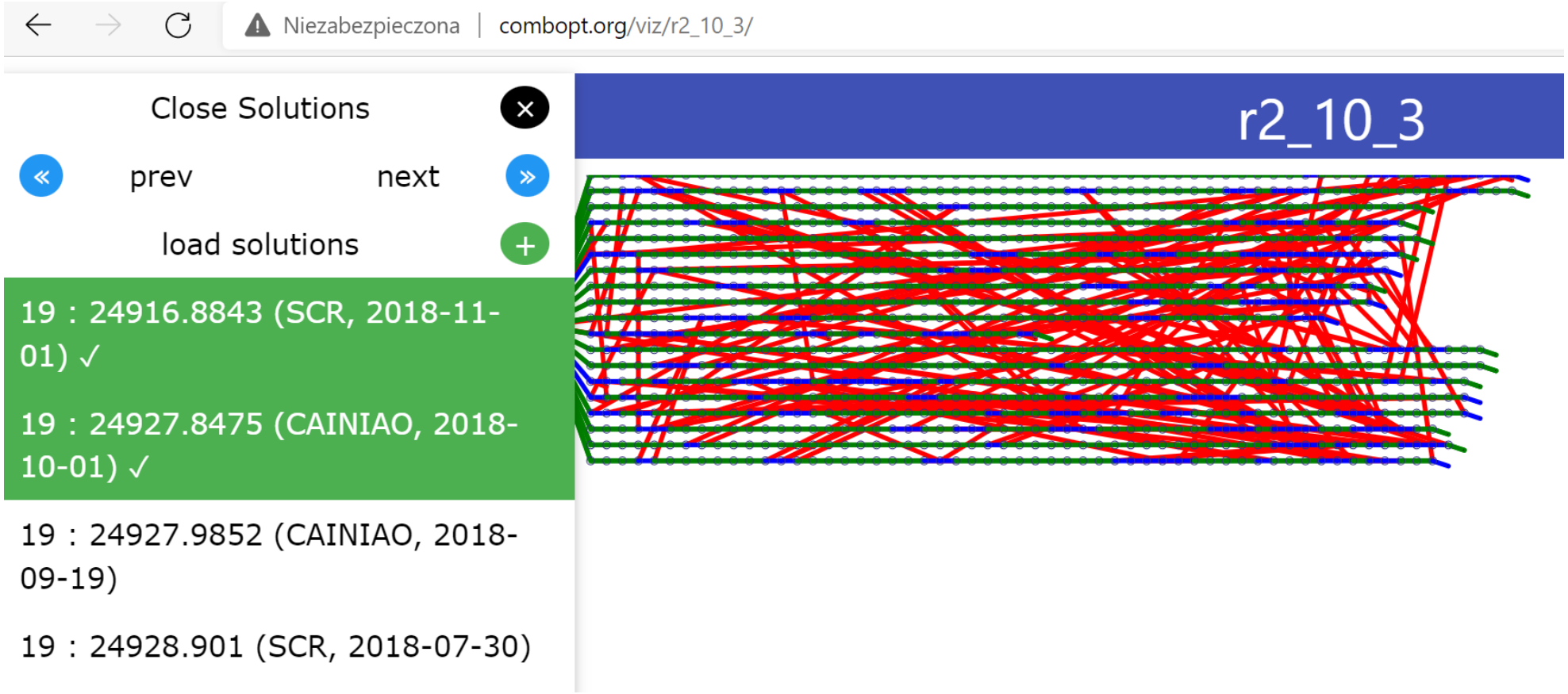
- Beling, Cybula, Jaszkiewicz, Rogalski, Sielski – ponad 100 rekordów dla Capacitated Vehicle Routing with time windows (www.sintef.no/vrptw), podobnie PDPTW, TOPTW
- Główna idea: aby poprawić najlepsze rozwiązanie dopuszczalne warto wejść głębiej w obszar rozwiązań niedopuszczalnych
- Fazy poprawy i dopuszczalności
- Faza poprawy – wagi kar zmniejszane, aż do poprawy najlepszego rozwiązania kosztem dopuszczalności
- Faza dopuszczalności – wagi zwiększane w celu uzyskania dopuszczalnego rozwiązania przy jak najmniejszym pogorszeniu wartości funkcji celu

Porównanie dwóch najlepszych znanych rozwiązań dla instancji r2_10_3 VRP

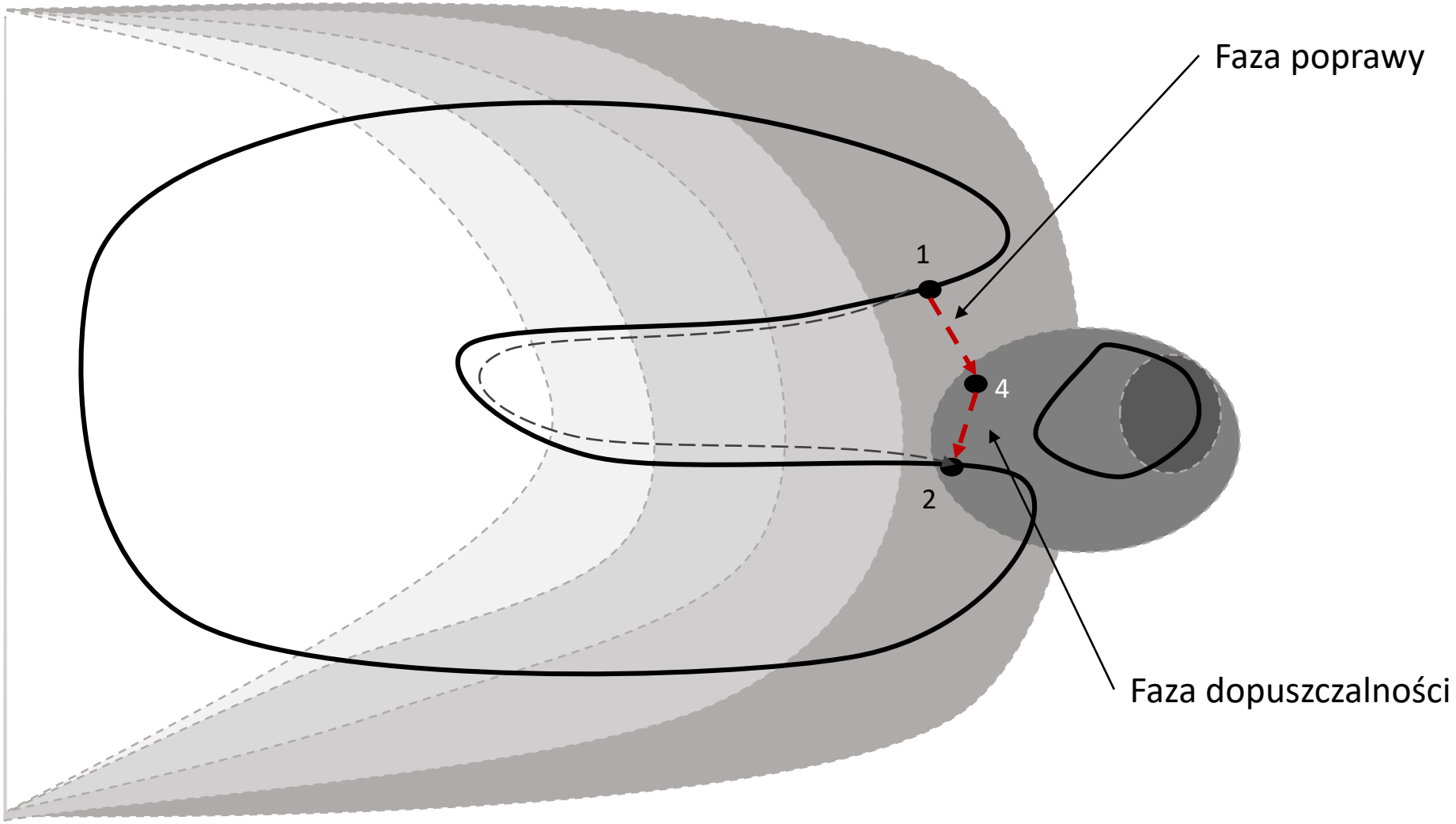


http://combopt.org

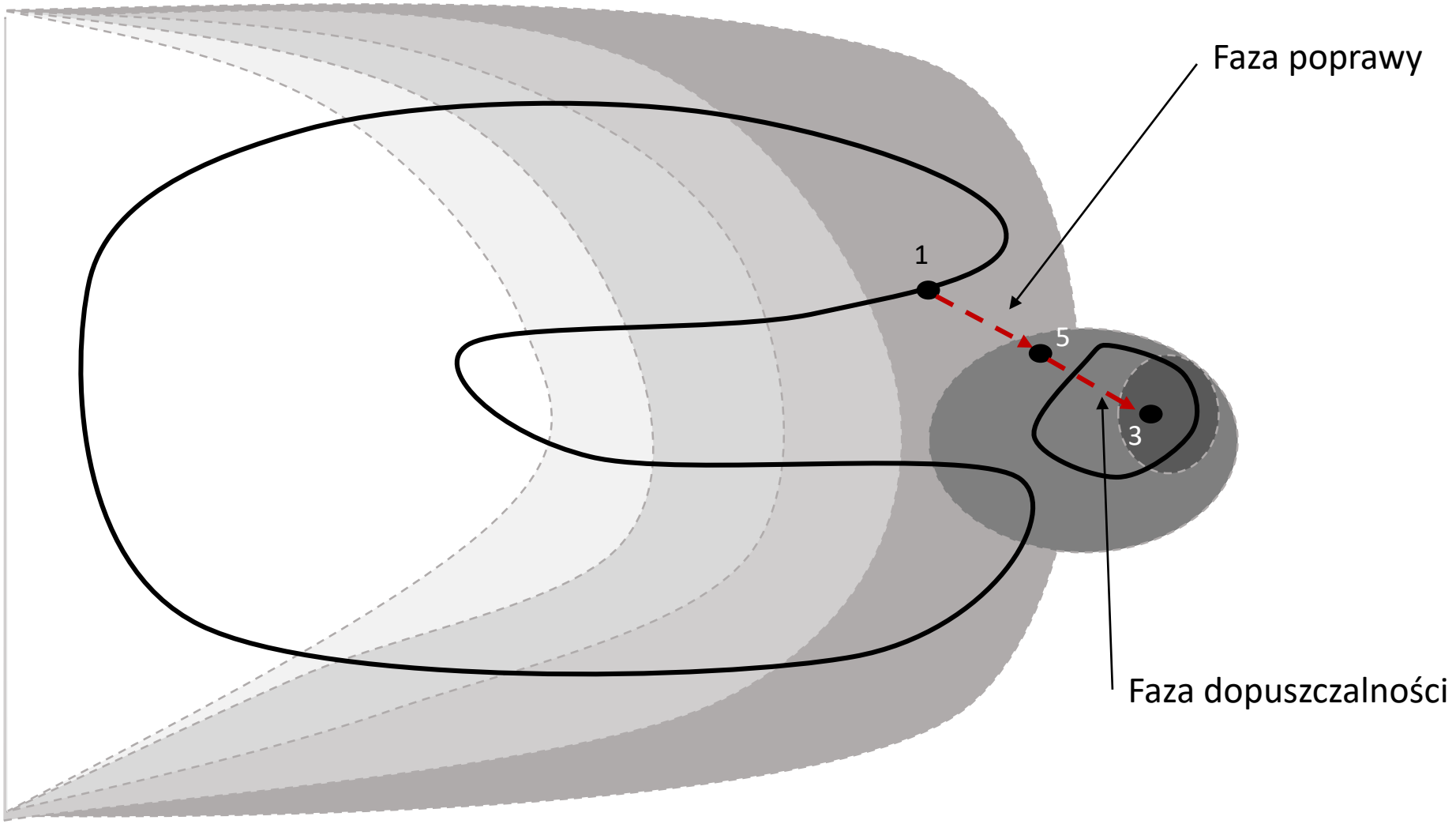
Porównanie dwóch najlepszych znanych rozwiązań dla instancji r2_10_3 VRP



Dwie naprzemienne fazy



Dwie naprzemienne fazy



Podejście wielokryterialne

- Funkcja celu i przekroczenie ograniczeń są traktowane jako dwa odrębne kryteria optymalizacji
  - Typowa funkcja kary może być interpretowana jako zastosowanie sumy ważonej funkcji celu i przekroczenia ograniczeń
- Stosowane są metaheurystyki wielokryterialne np. generowanie zbioru rozwiązań Pareto-optymalnych dla tych kryteriów