

# Podstawy teoretyczne inteligentnych metod optymalizacji

Andrzej Jaskiewicz

# Algorytmy inspirowane biologicznie i nie tylko...

- **Evolution-based**

- Differential evolution, Evolution strategies, Genetic/evolutionary 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
- Mohapatra, Sarada, and Prabhujit Mohapatra. "American zebra optimization algorithm for global optimization problems." Scientific Reports 13.1 (2023).

# Problem języka (lingo)

- Rozwiązanie – osobnik, genotyp, mrówka, robak, kruk...
- Metafory (biologiczne, społeczne, fizyczne) ułatwiają zrozumienie/zapamiętanie poszczególnych algorytmów, ale ukrywają ich podobieństwa
- Czy da się przedstawić wspólny schemat wszystkim/większości inteligentnych metod optymalizacji?

# Iteracyjna poprawa

Wygeneruj i oceń populację początkową  $\mathbf{X}$

Powtarzaj

Na podstawie populacji  $\mathbf{X}$  oraz jej ocen wygeneruj i oceń populację  $\mathbf{X}'$

Z  $\mathbf{X} \cup \mathbf{X}'$  wybierz nową populację  $\mathbf{X}$

Do momentu spełnienia warunków stopu

# Iteracyjna poprawa

- Populacja może być:
  - Jednoelementowa – lokalne przeszukiwanie, iteracyjne przeszukiwanie lokalne, symulowane wyżarzanie, przeszukiwanie Tabu,...
  - Wieloelementowa – algorytm ewolucyjny, algorytmy kolonii mrówek, ogólnie algorytmy populacyjne
- Co z listą Tabu? Pamięcią długoterminową? Śladem feromonowym?

# Iteracyjna poprawa z pamięcią

Zainicjuj pamięć **M**

Wygeneruj i oceń populację początkową **X**

Uaktualnij **M** na podstawie **X**

Powtarzaj

Na podstawie populacji **X**, jej ocen oraz pamięci **M**

Wygeneruj i oceń nową populację **X'**

Uaktualnij **M** na podstawie **X'**

Z  $\mathbf{X} \cup \mathbf{X}'$  wybierz nową populację **X**

Do momentu spełnienia warunków stopu

# Pojęcia intensyfikacji i dywersyfikacji

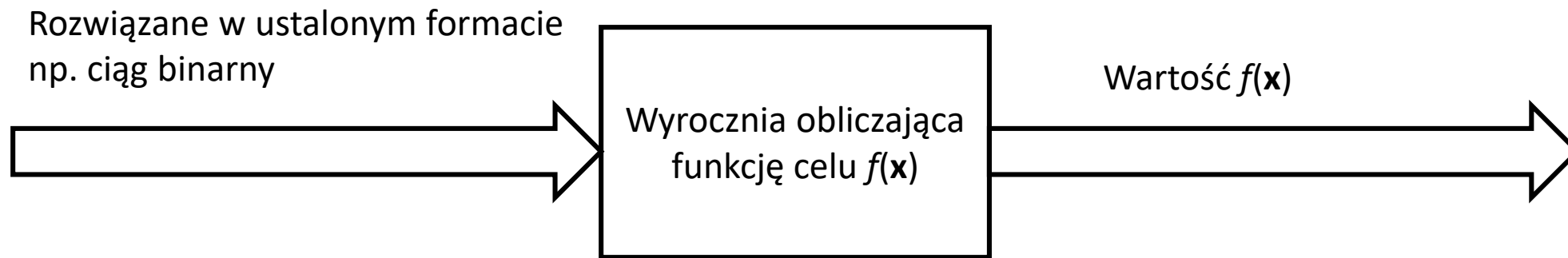
- Intensyfikacja
  - Poszukiwanie nowych dobrych rozwiązań we wcześniej zidentyfikowanych obiecujących obszarach przestrzeni rozwiązań
  - Z reguły powiązane z większym naciskiem na akceptowanie rozwiązań przynoszących poprawę
- Dywersyfikacja
  - Poszukiwanie nowych obiecujących obszarów odległych od poprzednio przeszukiwanych
  - Z reguły powiązane z akceptowaniem rozwiązań przynoszących pogorszenie
- Celem jest znalezienie dobrego balansu pomiędzy intensyfikacją a dywersyfikacją
- Balans ten może zmieniać się w czasie, np. w symulowanym wyżarzaniu stopniowo przechodzimy od dywersyfikacji do intensyfikacji



# Adaptacja inteligentnych metod optymalizacji do konkretnych problemów

- Wygeneruj populację początkową  $X$
- Wygeneruj nową populację  $X'$ 
  - Oznacza użycie specyficznych dla problemu operatorów (rekombinacji, sąsiedztwa, perturbacji...) do generowania nowych rozwiązań
- IMO to schematy algorytmów

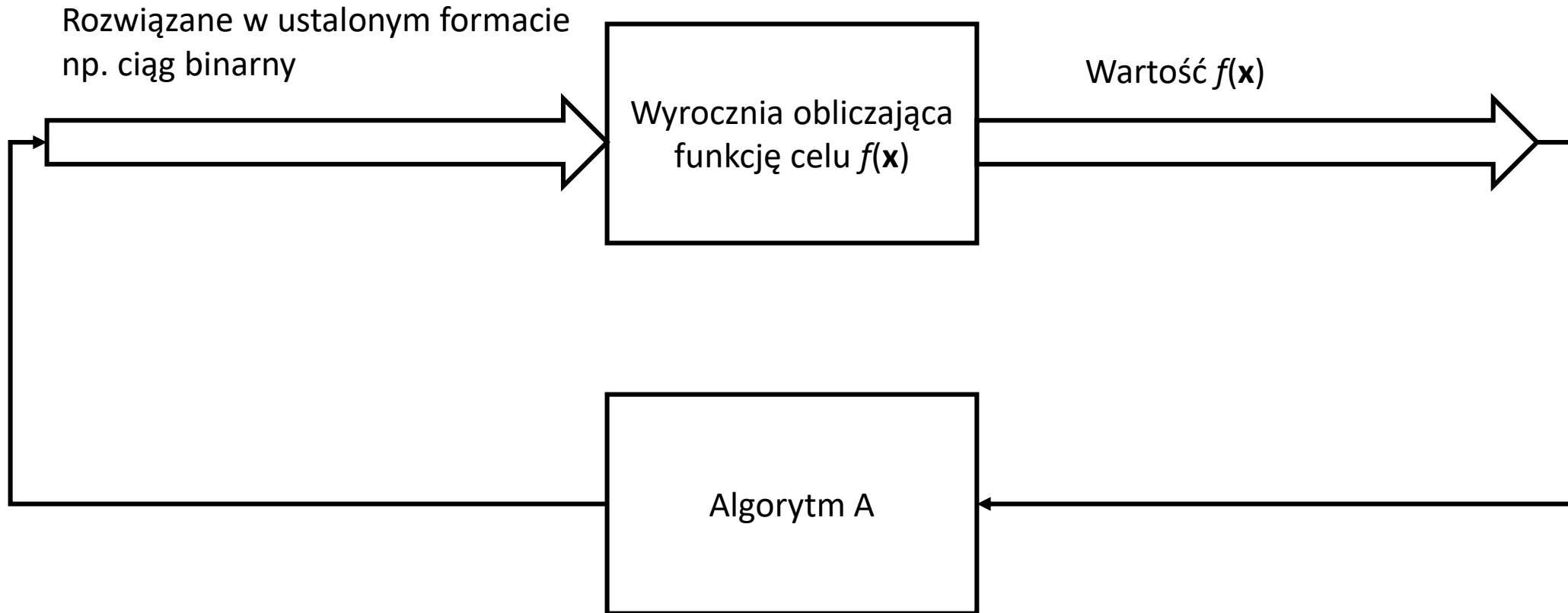
# Model wyroczni, czarnej skrzynki



# Wyrocznia – funkcja o nieznanej implementacji

```
double function (Vector<boolean>) {  
    // Nieznana implementacja  
    ...  
    return...  
}
```

# Optymalizacja funkcji zawartej w wyroczni, black-box optimization



# Pytania

- Czego możemy się spodziewać stosując jakiś algorytm A do optymalizacji funkcji zawartej w wyroczni?
- Czy np. iteracyjne przeszukiwanie lokalne będzie działało lepiej niż przeszukiwanie losowe?
- Czy w przeszukiwaniu lokalnym lepiej akceptować rozwiązania przynoszące poprawę niż pogorszenie? Wydaje się oczywiste.

# Twierdzenie „No free lunch” (nic za darmo)

- Założenia:

1. Przestrzeń rozwiązań jest skończona (np. wektor binarny o stałej długości).
2. Liczba możliwych wartości funkcji jest skończona.
3. Wyrocznia działa deterministycznie zwracając wynik pewnej funkcji.  
Z 1, 2 i 3 wynika, że liczba funkcji, które mogą znajdować się w wyroczni jest skończona.
4. Każda z możliwych funkcji ma równe prawdopodobieństwo wystąpienia w wyroczni (!)
5. Nie posiadamy, żadnej dodatkowej wiedzy na temat funkcji  $f(\mathbf{x})$ .
6. Algorytm A generuje rozwiązania bez powtórzeń.

# Twierdzenie „No free lunch”

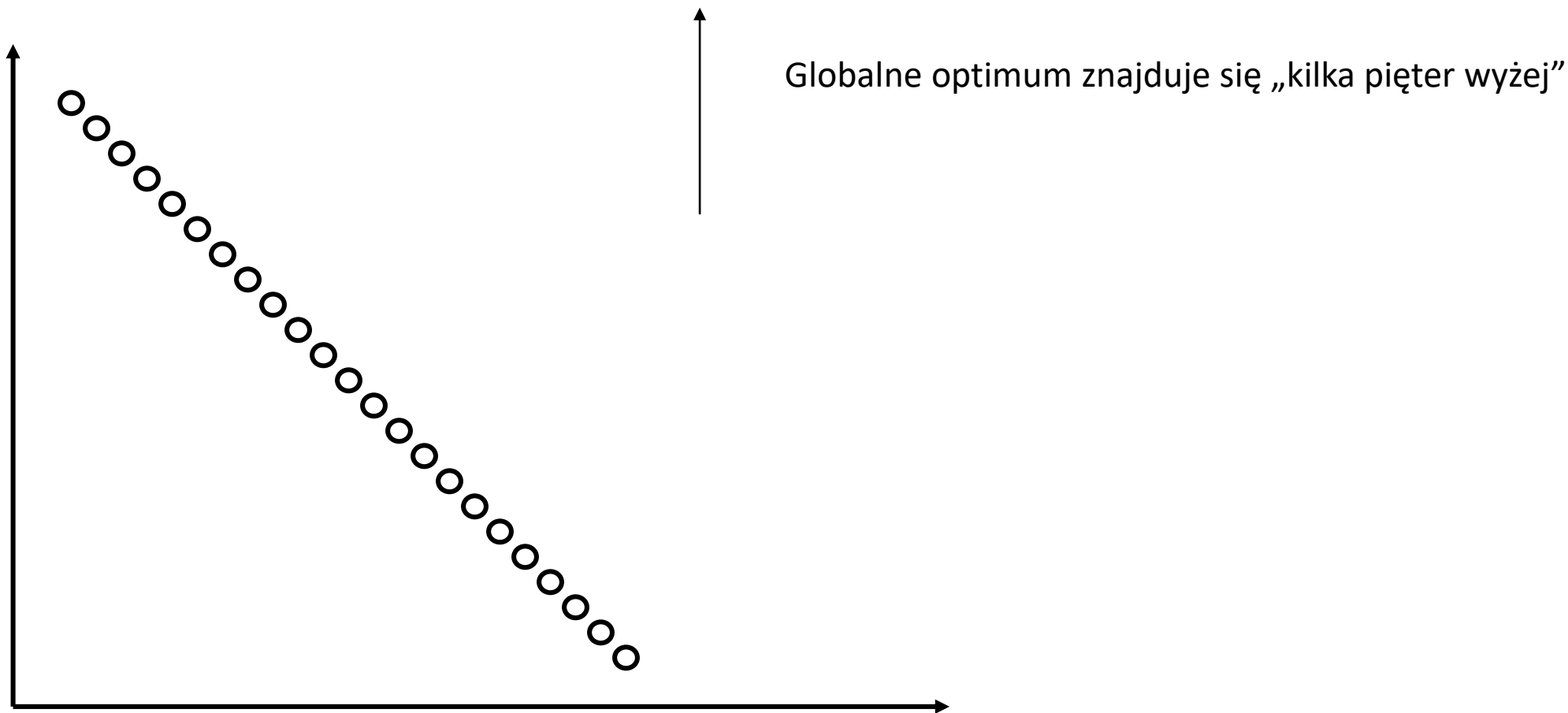
- **Przy powyższych założeniach, każdy algorytm A da w zadanej liczbie iteracji da tę samą wartość oczekiwaną dowolnej „sensownej” miary jakości uzyskanych rozwiązań.**
- „Sensowna” miara to miara bazując na rozkładzie wartości wszystkich rozwiązań wygenerowanych przez algorytm A.
- Np.:
  - najlepsze wygenerowane rozwiązanie
  - średnia wartość N najlepszych rozwiązań
  - średnia wartość wygenerowanych rozwiązań
- ale nie:
  - średnia wartość rozwiązań w aktualnej populacji
  - średnia wartość rozwiązań w końcowej populacji
  - Zwrócone przez algorytm rozwiązanie(ponieważ wartości różnych wygenerowanych rozwiązań mogą być uwzględniane różną liczbę razy)

# Twierdzenie „No free lunch”

- „Każdy” oznacza każdy
  - Np. lokalne przeszukiwanie z wyborem najlepszego rozwiązania z sąsiedztwa da tę samą wartość oczekiwaną każdej sensownej miary jakości jak lokalne przeszukiwanie z wyborem najgorszego rozwiązania z sąsiedztwa
- Jeżeli algorytm A działa lepiej niż algorytm B na pewnych funkcjach, to muszą być inne funkcje, na których będzie działał gorzej



# Przykład funkcji „zwodniczej” dla lokalnego przeszukiwania - maksymalizacja



# Algorytm A jako reguła generowania nowych rozwiązań

- Rozwiązanie generowane w  $i$ -tej iteracji:

$$\mathbf{x}_i = A(\mathbf{x}_1, f(\mathbf{x}_1), \mathbf{x}_2, f(\mathbf{x}_2), \dots, \mathbf{x}_{i-1}, f(\mathbf{x}_{i-1}))$$

- Mówiąc inaczej maksymalna możliwa pamięć  $M$  to lista wszystkich wygenerowanych wcześniej rozwiązań i ich wartości funkcji celu

# Zarys dowodu - intuicja

- Rozważmy dwie zmienne binarne  $x_1, x_2$ .
- Cztery możliwe rozwiązania:
  - 00
  - 01
  - 10
  - 11
- Dziedzina funkcji celu to  $\{0, 1\}$  – tylko dwie wartości

# Wszystkie możliwe funkcje celu

$x_1$	$x_2$		f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16
0	0		0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1		0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0		0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1		0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Na przykład:

$$f1 = 0$$

$$f2 = x_1 \wedge x_2$$

$$f4 = x_1$$

$$f5 = x_2$$

$$f8 = x_1 \vee x_2$$

$$f13 = \neg x_1$$

# Pierwszy krok algorytmu

$x_1$	$x_2$		f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16
0	0		0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1		0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0		0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1		0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Dla każdego z możliwych czterech rozwiązań mamy tę samą wartość oczekiwaną wartości funkcji celu

Generujemy rozwiązanie  $\mathbf{x} = [0,0]$

Otrzymujemy z wyroczni  $f(\mathbf{x}) = 0$

Co to oznacza?

# Wiedza zdobyta po pierwszym kroku

$x_1$	$x_2$		f1	f2	f3	f4	f5	f6	f7	f8
0	0		0	0	0	0	0	0	0	0
0	1		0	0	0	0	1	1	1	1
1	0		0	0	1	1	0	0	1	1
1	1		0	1	0	1	0	1	0	1

Generujemy rozwiązanie  $\mathbf{x} = [1, 0]$

Otrzymujemy z wyroczni  $f(\mathbf{x}) = 0$

Co to oznacza?

# Drugi krok

$x_1$	$x_2$		f1	f2	f3	f4	f5	f6	f7	f8
0	0		0	0	0	0	0	0	0	0
0	1		0	0	0	0	1	1	1	1
1	0		0	0	1	1	0	0	1	1
1	1		0	1	0	1	0	1	0	1

Dla każdego z możliwych trzech rozwiązań mamy tę samą wartość oczekiwaną wartości funkcji celu

Generujemy rozwiązanie  $\mathbf{x} = [1, 0]$

Otrzymujemy z wyroczni  $f(\mathbf{x}) = 0$

Co to oznacza?

# Wiedza zdobyta po drugim kroku

$x_1$	$x_2$		f1	f2		f5	f6
0	0		0	0		0	0
0	1		0	0		1	1
1	0		0	0		0	0
1	1		0	1		0	1



# Dowód twierdzenia „No free lunch”

- Dowód polega na wykazaniu, że w każdym kroku algorytmu, niezależnie od wyboru poprzednich rozwiązań, mamy ten sam rozkład wartości funkcji celu dla wszystkich możliwych do wygenerowania rozwiązań

# Założenie twierdzenia można osłabić

- Wystarczające jest założenie, że podzbiór możliwych funkcji jest closed under permutation (c.u.p.), tj. dla dowolnej permutacji zmiennych zbiór funkcji pozostaje niezmienny

# Analogiczne twierdzenie „No free lunch” istnieje także dla maszynowego uczenia

- Związki pomiędzy ML a optymalizacją (ML to w zasadzie optymalizacja jakiejś miary błędu)
- Analogie
  - rozwiązania - obiekty, przypadki
  - wygenerowane rozwiązania - przypadki uczące
  - wartość funkcji celu - klasa decyzyjna
  - reguła/algorytm - mechanizm klasyfikacji
  - wiedza - wiedza
- Różnice
  - Cel: ML - klasyfikacja, optymalizacja - generowanie
  - ML: założenie opis dyskryminujący, optymalizacja chyba nieistotne
  - Optymalizacja - uporządkowanie wartości funkcji celu

# Twierdzenie NFL a „praktyka”

- Jak twierdzenie NFL ma się do wiedzy, że typowe metaheurystyki dobrze sprawdzają się w praktyce
- „W teorii nie ma różnicy pomiędzy teorią a praktyką, a w praktyce jest”

# Jakie założenia NFL mogą nie być spełnione w praktyce?

- Nie posiadamy, żadnej dodatkowej wiedzy na temat funkcji  $f(x)$ .
  - Wiedza na temat problemu może być wykorzystywana w operatorach (sąsiedztwa, perturbacji, rekombinacji) – np. operator może kierować się wartością funkcji celu – w efekcie dedykowana metoda
  - Np. w TSP wykorzystanie listy najbliższych wierzchołków do definiowania ruchów kandydackich w lokalnym przeszukiwaniu...
  - ... ale LP dla TSP działa dobrze także bez ruchów kandydackich

# Jakie założenia NFL mogą nie być spełnione w praktyce?

- Zbiór funkcji jest c.u.p. i każda z możliwych funkcji ma równe prawdopodobieństwo wystąpienia w wyroczni
  - Np. konkretny problem kombinatoryczny może nie dopuszczać wszystkich funkcji (wiele funkcji nie odpowiada żadnym instancjom tego problemu) i zbiór dopuszczalnych funkcji nie musi być c.u.p.
    - Udział podzbiorów c.u.p. wśród wszystkich możliwych podzbiorów szybko dąży do zera wraz ze wzrostem rozmiaru przestrzeni rozwiązań ...
    - ...czyli intuicyjnie jeżeli mamy do czynienia z jakimś podzbiorem wszystkich funkcji, to prawie na pewno nie jest on c.u.p. i nie obowiązuje NFL
  - Rozkład wartości parametrów problemu może przekładać się na nierównomierny rozkład prawdopodobieństwa poszczególnych instancji i funkcji

# Przykład prostego problemu kombinatorycznego

$$f(\mathbf{x}_1, \mathbf{x}_2) = w_1 \mathbf{x}_1 \vee (w_2 (\neg \mathbf{x}_2))$$

$$w_1, w_2 \in \{0,1\}$$

$$w_1 = 0 \\ w_2 = 0$$

$$w_1 = 1 \\ w_2 = 0$$

$$w_1 = 0 \\ w_2 = 1$$

$$w_1 = 1 \\ w_2 = 1$$

$x_1$	$x_2$	f1	f4	f11	f12
0	0	0	0	1	1
0	1	0	0	0	0
1	0	0	1	1	1
1	1	0	1	0	1

Poszczególne rozwiązania różnią się oczekiwanymi wartościami funkcji celu

Rozkład prawdopodobieństwa parametrów może wpływać na rozkład prawdopodobieństwa funkcji

- $P(w_1=1) = 0.7, P(w_2=1) = 0.7$
- $P(f = f_1) = 0.09$
- $P(f = f_4) = 0.21$
- $P(f = f_{11}) = 0.21$
- $P(f = f_{12}) = 0.49$



# W praktyce znaczenie może mieć też szybkość generowania rozwiązań

- Twierdzenie NFL odnosi się od liczby wygenerowanych rozwiązań, a nie do czasu
- Jeżeli dana metoda generuje nowe rozwiązania szybciej (bo np. mniej się różnią od poprzednich), to w praktyce będzie lepsza

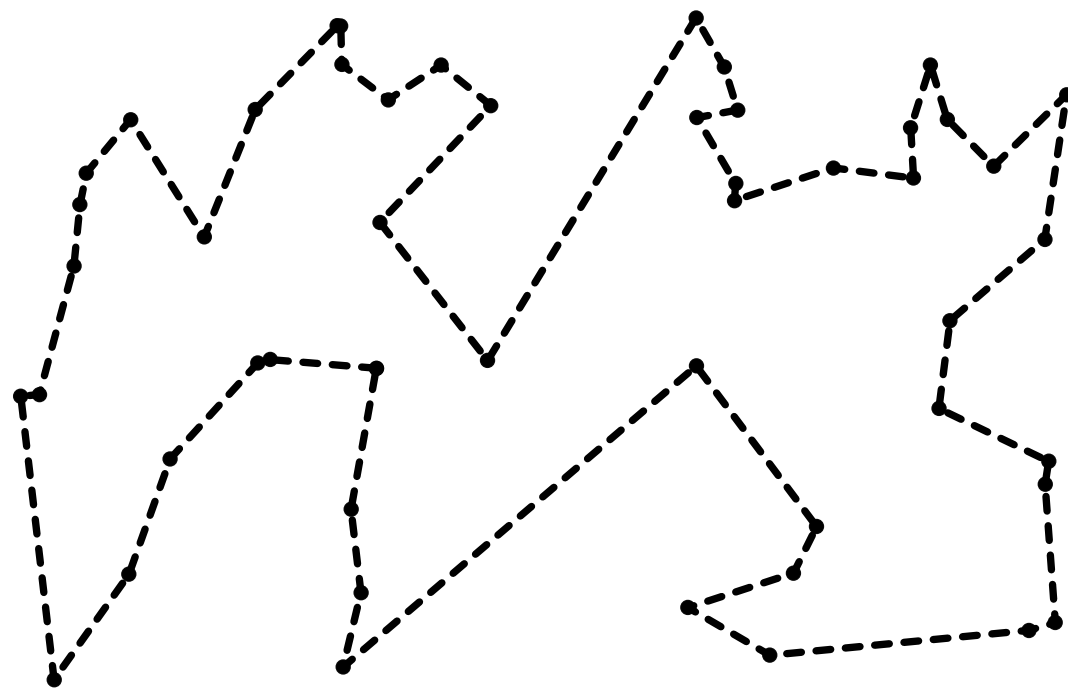
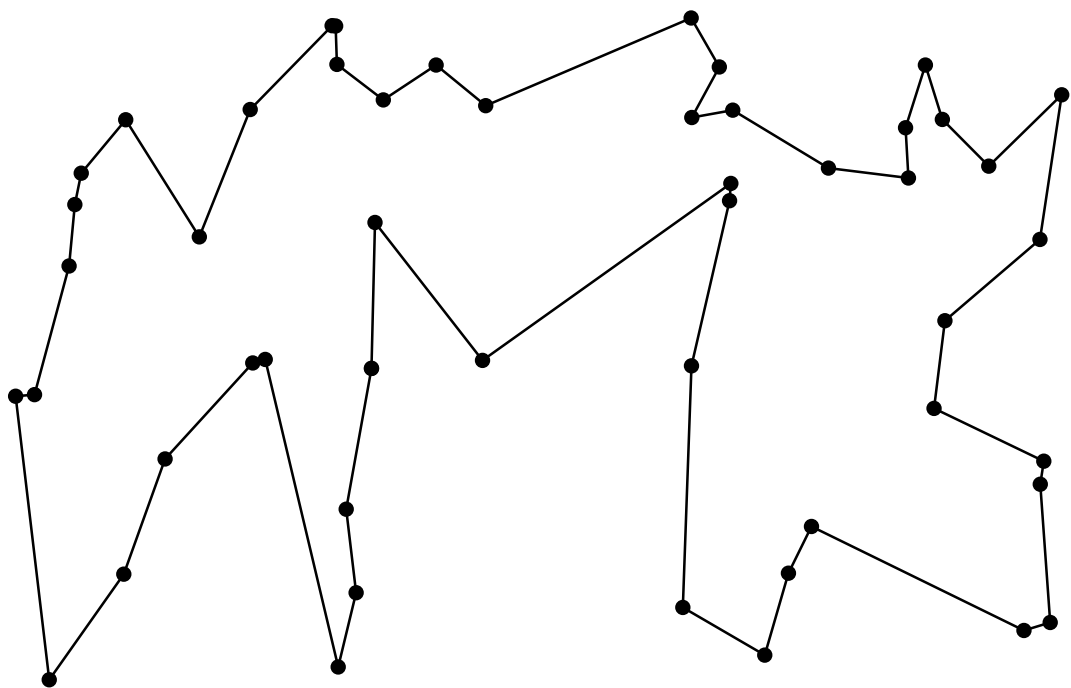
# Czym są inteligentne metody optymalizacji (metaheurystyki)

- Nie są to uniwersalne metody optymalizacji black-box, gdyż zgodnie z NFL takie metody nie istnieją
- IMO działają na pewnej klasie funkcji (problemów), które najwidoczniej często występują w „praktyce”
- Jakie cechy mają te funkcje (problemy)

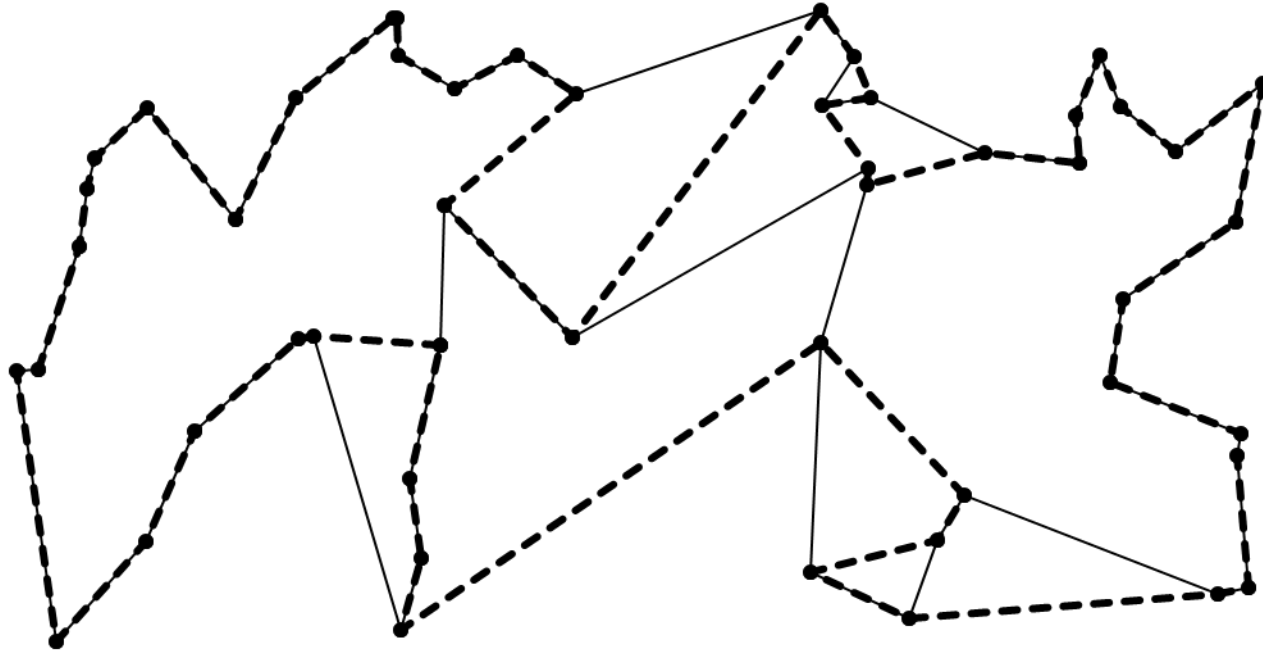
# Eksperyment Boese, Kahng i Muddu (1994) dla TSP

- Typowe IMO bardzo dobrze się sprawdzają na problemie komiwojażera
- Instancje o rozmiarach 100 i 500 miast
- 2500 „losowych” optimów lokalnych uzyskanych za pomocą lokalnego przeszukiwania w wersji zachłannej startującego z losowych rozwiązań początkowych
- Miara podobieństwa rozwiązań – liczba wspólnych krawędzi
- Badanie relacji pomiędzy wzajemnym podobieństwem optimów lokalnych (średnim do wszystkich pozostałych lub do najlepszego) a ich jakością
- Obserwacje
  - Bardzo silne korelacje pomiędzy podobieństwem rozwiązań a ich jakością – lepsze są bardziej podobne
  - Silniejsze korelacje dla średniego podobieństwa do wszystkich pozostałych niż do najlepszego

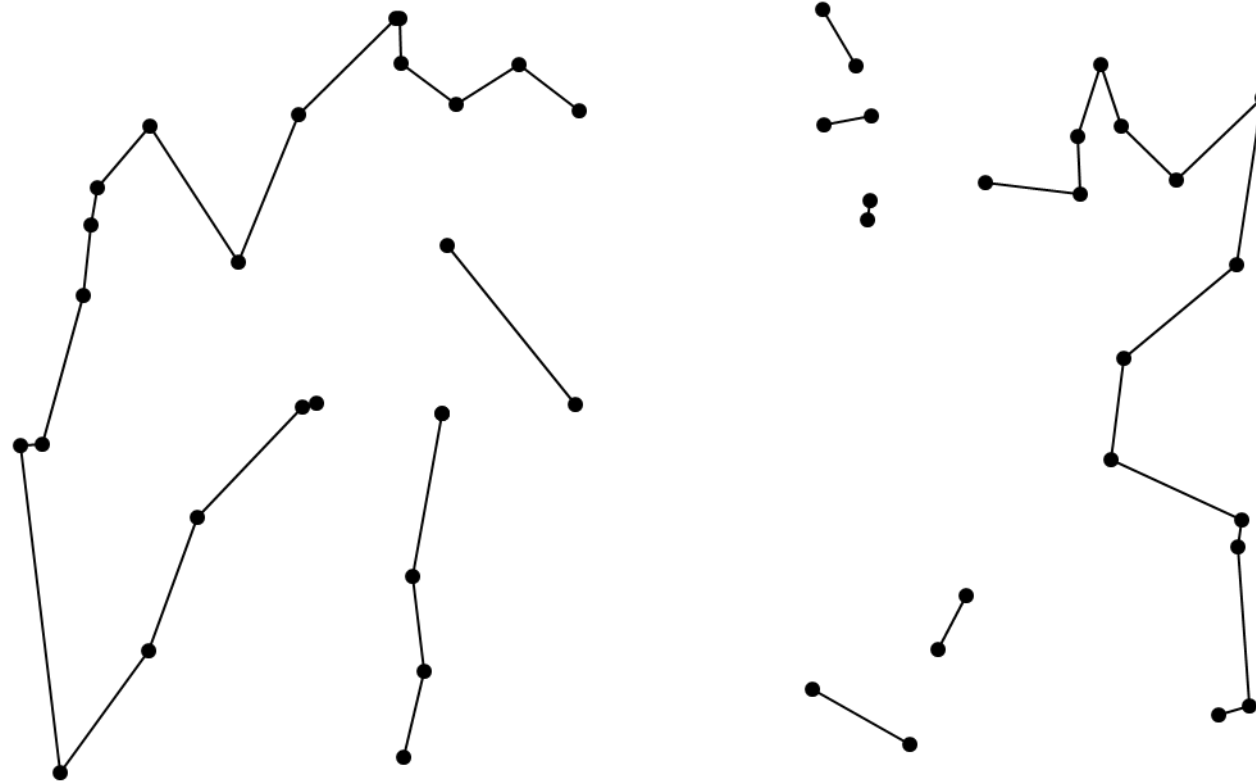
# Dwa przykładowe optima lokalne problemu komiwojażera



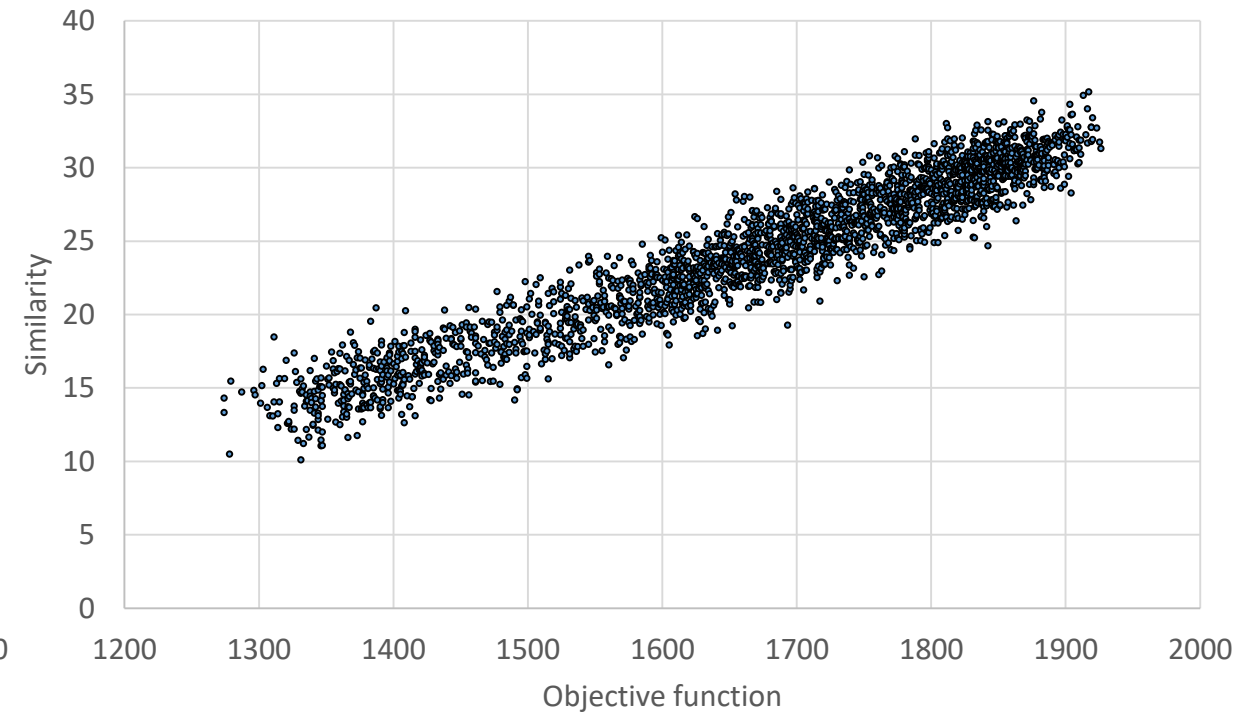
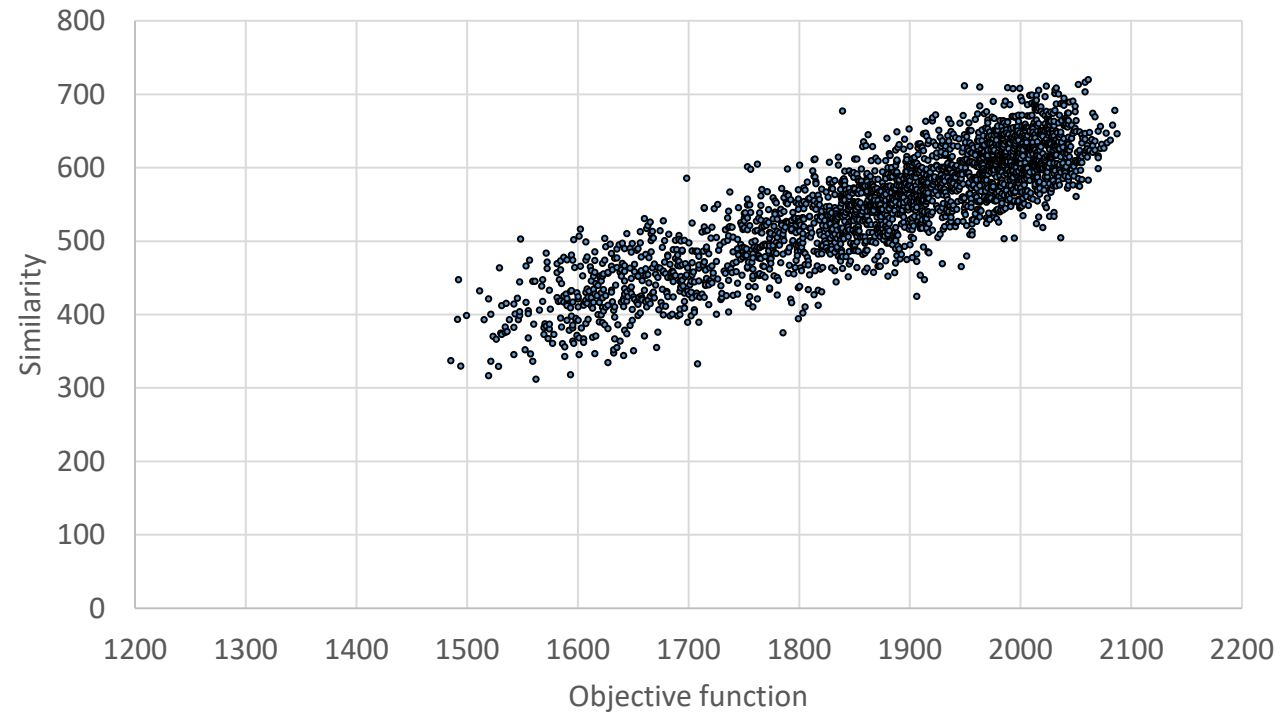
# Dwa przykładowe optima lokalne problemu komiwojażera



# Wspólne krawędzie (i ścieżki)



# Przykłady korelacji (inny problem i eksperyment) – maksymalizowana funkcja celu



# Inne obserwacje Boese, Kahng i Muddu (1994) dla TSP

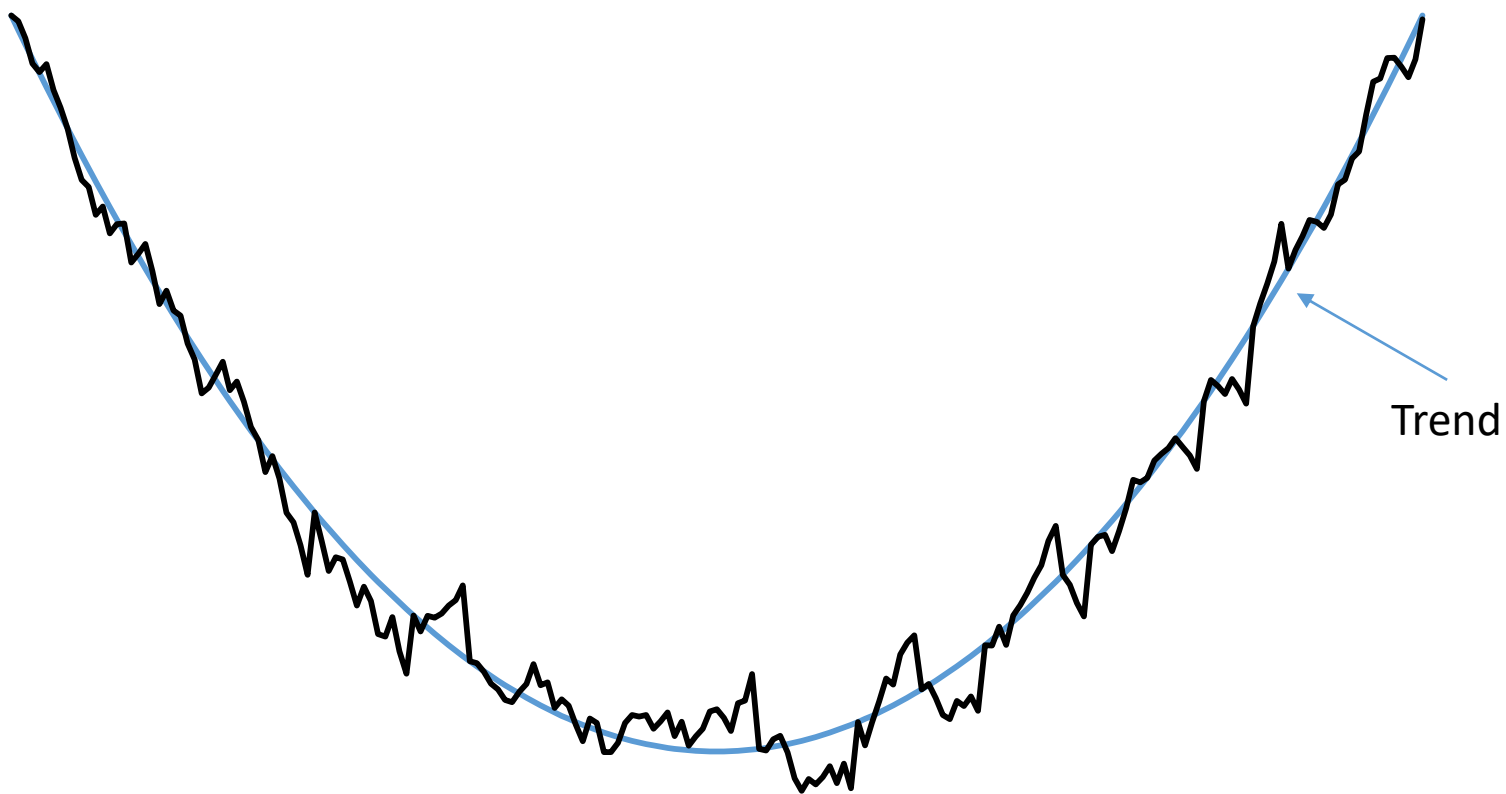
- Średnia spodziewana odległość dwóch losowych rozwiązań to  $N-2$  (dwie wspólne krawędzie)
- Dla stu miast maksymalna odległość to 48 (52 wspólne krawędzie)
- Wszystkie optima lokalne są zawarte w „kuli” o średnicy 48. Kula to obejmuje  $1/10^{59}$  rozwiązań
- Dla 500 miast: średnica „kuli” 243,  $1/10^{468}$  rozwiązań



# Globalna wypukłość funkcji (problemu)

- Autorzy nazwali zaobserwowaną cechę globalną wypukłością lub głęboką doliną (deep valley)
- W matematyce funkcja jest globalnie wypukła, jeżeli można ją sprowadzić do funkcji wypukłej dodając/odejmując do jej wartości dla poszczególnych rozwiązań pewną wartość  $\leq \varepsilon$
- Może być rozumiana jako złożenie trendu (funkcja wypukła) i (deterministycznego) szumu
- Rozwiązania leżące blisko optimum trendu są średnio lepsze i bardziej podobne do innych dobrych rozwiązań (które leżą wokół optimum trendu)

# Przykład globalnie wypukłej funkcji jednej zmiennej



# Uwagi

- Globalna wypukłość oznacza, że dobre rozwiązania są do siebie podobne, a nie że rozwiązania podobne do dobrych też są dobre (szum może być bardzo duży i małe różnice mogą powodować duże skoki wartości funkcji celu)
- Globalna wypukłość zależy od sposobu pomiaru podobieństwa rozwiązań

# Z czego wynika trend dla problemu komiwojażera?

- Dobre rozwiązania składają się z krótkich krawędzi (trend)...
- ... ale  $N$  najlepszych krawędzi raczej nie tworzy cyklu Hamiltona
- ... żeby utworzyć cykl Hamiltona musimy wymienić pewne krawędzie na dłuższe, czyli zwiększyć wartość funkcji celu – dodać deterministyczny szum

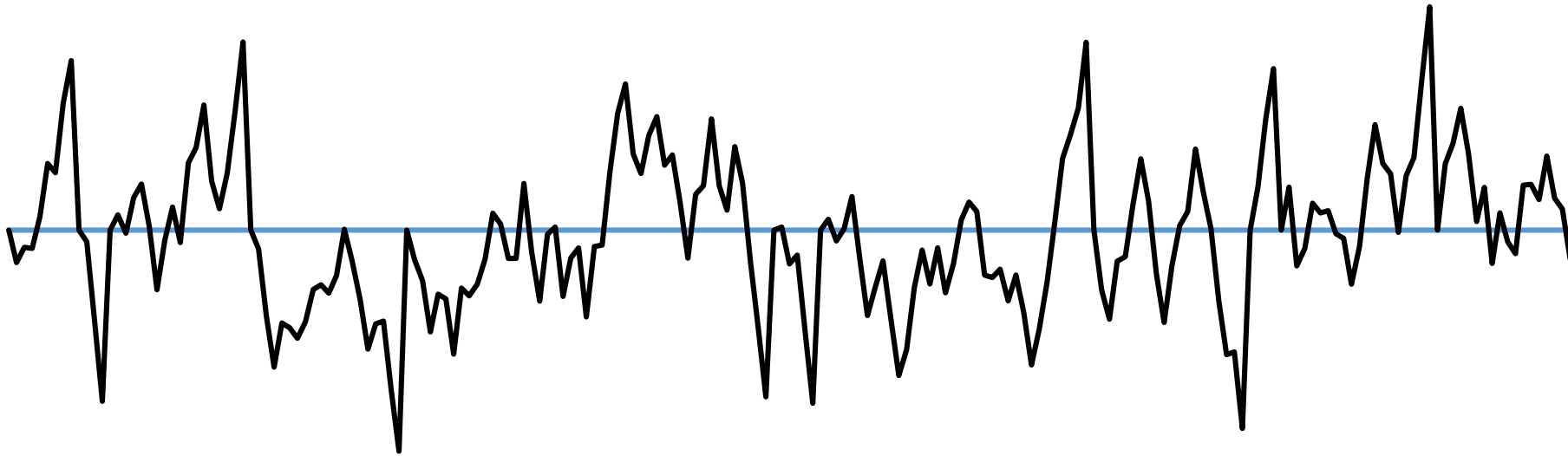
# Globalna wypukłość w innych problemach

- Globalna wypukłość została wykazana dla wielu innych problemów, np.:
  - Problem podziału grafu - Boese, Kahng i Muddu (1994), Problem kwadratowego przydziału - Merz, Freisleben (2000), Problem marszrutyzacji pojazdów - Jaskiewicz, Kominek (2000, 2003), Problem marszrutyzacji pojazdów - Kubiak (2004), Capacitated Vehicle Routing Problem (CVRPTW), Pickup and Delivery Problem (PDPTW), and Team Orienteering Problem (TOPTW) z oknami czasowymi - Cybula, Jaskiewicz, Pełka, Rogalski, Sielski (2021)
- Intuicyjne trendy dla wielu problemów
  - Np. w problemie VRP spodziewamy się występowania krótkich ale też „dobrze skierowanych” (do/z bazy) krawędzi. Krawędzie powinny też łączyć wierzchołki o dopasowanych oknach czasowych

# Jak globalna wypukłość tłumaczy działanie IMO

- Iteracyjne przeszukiwanie lokalne, wieloskalowe przeszukiwanie sąsiedztwa, symulowane wyżarzanie, przeszukiwanie tabu... – model podrzucania kulki w poszarpanej misce
- Operatory rekombinacji – konstrukcja nowych rozwiązań łączących cechy (a więc podobnych do) rodziców (przerzucanie lin nad doliną)
  - Podobieństwo do dobrych rozwiązań nie gwarantuje jednak wysokiej jakości, ale:
    - Średnia jakość będzie jednak wyższa
    - Lokalne przeszukiwanie może być w stanie bardzo szybko poprawić takie rozwiązanie (w pobliżu są inne bardzo dobre rozwiązania)

# Problemy globalnie „płaskie”



- Brak globalnego trendu
- Nadal może działać lokalne przeszukiwanie choć najlepszą metodą może być proste lokalne przeszukiwanie z wieloma punktami startowymi
- Z jednej strony trudne dla IMO, z drugiej nie możemy zbyt wiele stracić na wartości funkcji celu

# Inne badania

- Nie prowadzono zbyt wiele prac nad pytaniem „na jakich funkcjach działają IMO?”, za to wiele prac na temat „jakie funkcje są proste/trudna dla IMO?”



# Fitness distance correlation FDC (korelacja jakość przystosowanie

- Podejście podobne do testowania globalnej wypukłości – badanie korelacji jakość-podobieństwo/odległość
- Odległość od optimum globalnego
- Oryginalnie zakładano kodowanie 0-1

# Twierdzenie o schematach Hollanda

- Krótkie, niskiego rzędu i dobrze przystosowane schematy rozprzestrzeniają się w kolejnych pokoleniach zgodnie z wykładniczym prawem wzrostu
- Występowanie wspólnego schematu(ów) w krzyżowanych rozwiązaniach świadczy o ich podobieństwie

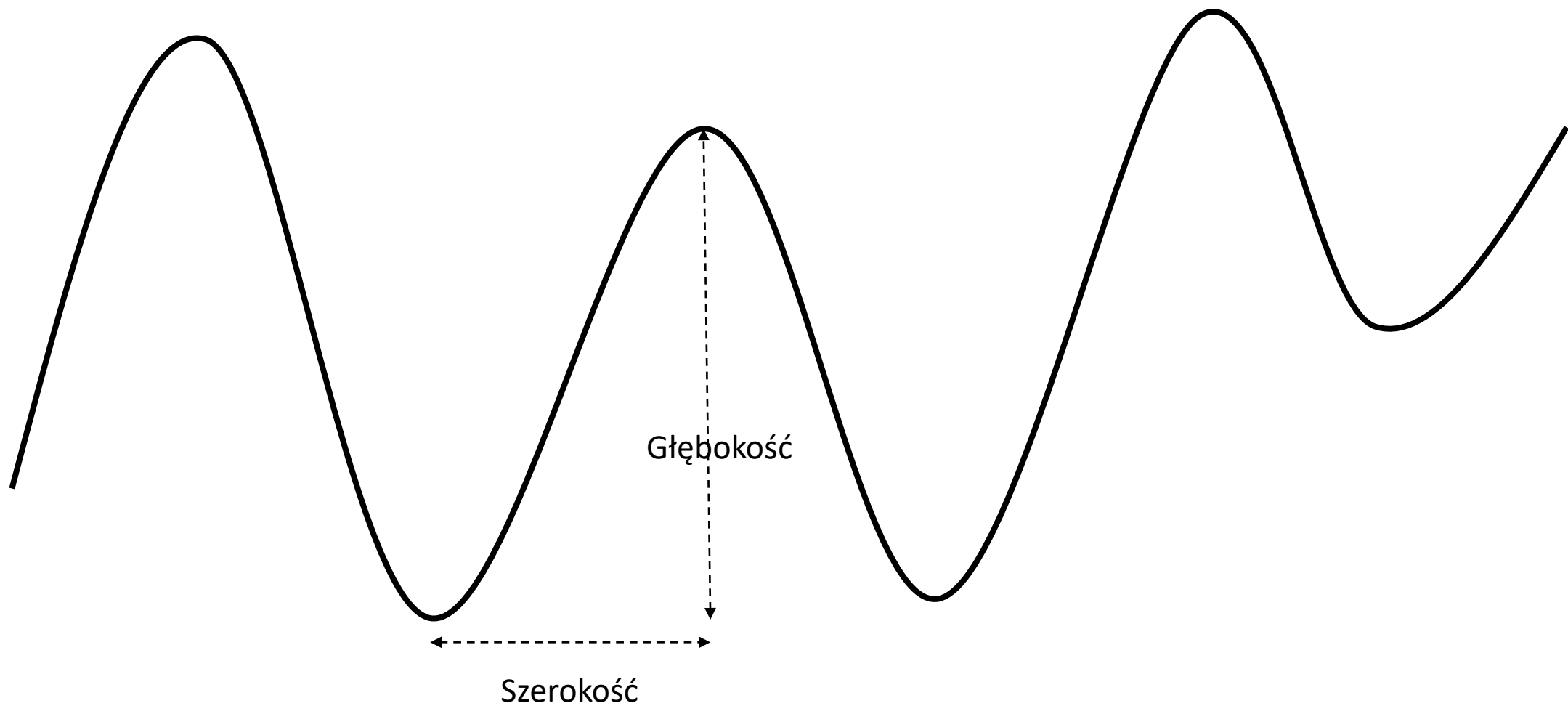
# Entropia w zbiorze optimów lokalnych

- Niska entropia – lokalne optima są do siebie podobne
  - Globalna wypukłość oznacza niską entropię
- Wysoka entropia – lokalne optima dowolnie porozrzucone w przestrzeni rozwiązań – brak trendu

# Cechy lokalne

- Efektywność lokalnego przeszukiwania w stosunku do losowego przeglądu
  - Dla TSP lokalne optimum z wymianą dwóch krawędzi/łuków osiąga się w średnio  $n$  iteracjach, gdzie  $n$  to liczba miast, czyli złożoność to  $O(n^3)$
  - Tymczasem dla  $n=100$  lokalne optima są zawarte w kuli stanowiącej  $1/10^{59}$  przestrzeni rozwiązań
  - Przyspieszenie  $10^{59} / 10^6 = 10^{53}$
- Głębokość i szerokość optimów lokalnych
  - Ile ruchów trzeba średnio wykonać lub o ile trzeba średnio pogorszyć funkcję celu, żeby przejść do innego lokalnego optimum

# Głębokość i szerokość optimów lokalnych



# Ruggedness (chropowatość) – współczynnik autokorelacji

$$\rho(d) = \frac{\overline{(f(\mathbf{x}) - f(\mathbf{y}))^2 |_{d(\mathbf{x}, \mathbf{y})=d}}}{\overline{(f(\mathbf{x}) - f(\mathbf{y}))^2}}$$

- $d(\mathbf{x}, \mathbf{y})$  – miara odległości rozwiązań, liczba kroków lokalnego przeszukiwania wymagana aby przejść z  $\mathbf{x}$  do  $\mathbf{y}$
- $d=1$  oznacza rozwiązania sąsiednie w liczniku
- Niezależność od skalowania funkcji celu
- Hipoteza – lokalne przeszukiwanie działa lepiej dla mniejszych współczynników autokorelacji

# Przykład operatory sąsiedztwa dla problemu komiwojażera

- Wymiana dwóch wierzchołków
  - Rozwiązania sąsiednie różnią się czterema krawędziami
- Wymiana dwóch krawędzi
  - Rozwiązania sąsiednie różnią się dwoma krawędziami
- Średnia różnica wartości sąsiednich rozwiązań jest więc większa w pierwszym przypadku
- Wielkość sąsiedztwa jest w obu wypadkach podobna  $O(n^2)$
- Wniosek wymiana dwóch krawędzi powinna działać lepiej – doskonale potwierdza się w praktyce
- Rozumowania nie można jednak łatwo przenieść na sąsiedztwa o różnej wielkości, np. wymiana 3 krawędzi będzie dawała lepsze wyniki (w dłuższym czasie) choć współczynnik autokorelacji będzie większy

# Fitness/objective function landscape analysis

- Ogólnie tego typu badania nazywa się analizą krajobrazu funkcji celu
- Krajobraz funkcji celu – trójka (zbiór rozwiązań, miara odległości/podobieństwa, funkcja celu)



# Podsumowując – na jakich funkcjach/problemach działają IMO

- Globalna struktura, trendy – dobre rozwiązania rozłożone nieprzypadkowo
- Lokalny charakter (gładkość) funkcji celu

# Co daje wiedza o powyższych cechach problemów/funkcji?

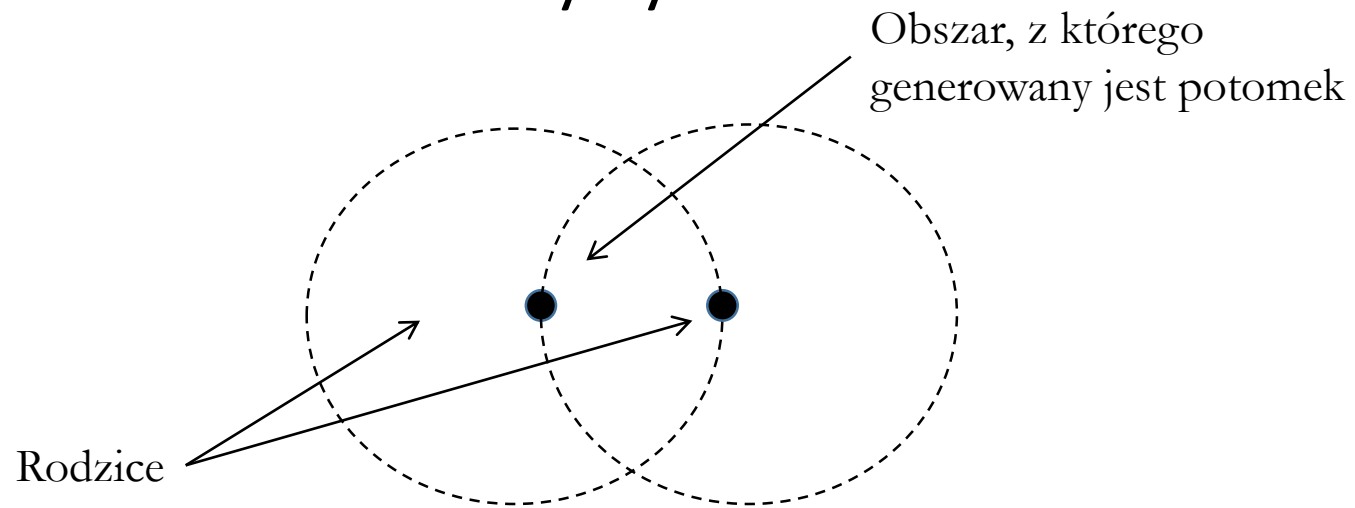
- Znaczenie kodowania, sąsiedztwa, rekombinacji
  - Np. globalna wypukłość/autokorelacja może zależeć od tych wyborów
  - Dobór sąsiedztwa o niskim współczynniku autokorelacji
- Metody motywowane globalną wypukłością:
  - Np. Adaptacyjne lokalne przeszukiwanie z wieloma punktami startowymi
- Operatory rekombinacji zachowujące dystans
- Operatory destroy-repair kierowane innym(i) rozwiązaniem(rozwiazaniami) – usuwanie ważnych różnych cech
  - Różnica pomiędzy operatorami destroy-repair a rekombinacji może się zacieć

# Znaczenie operatorów

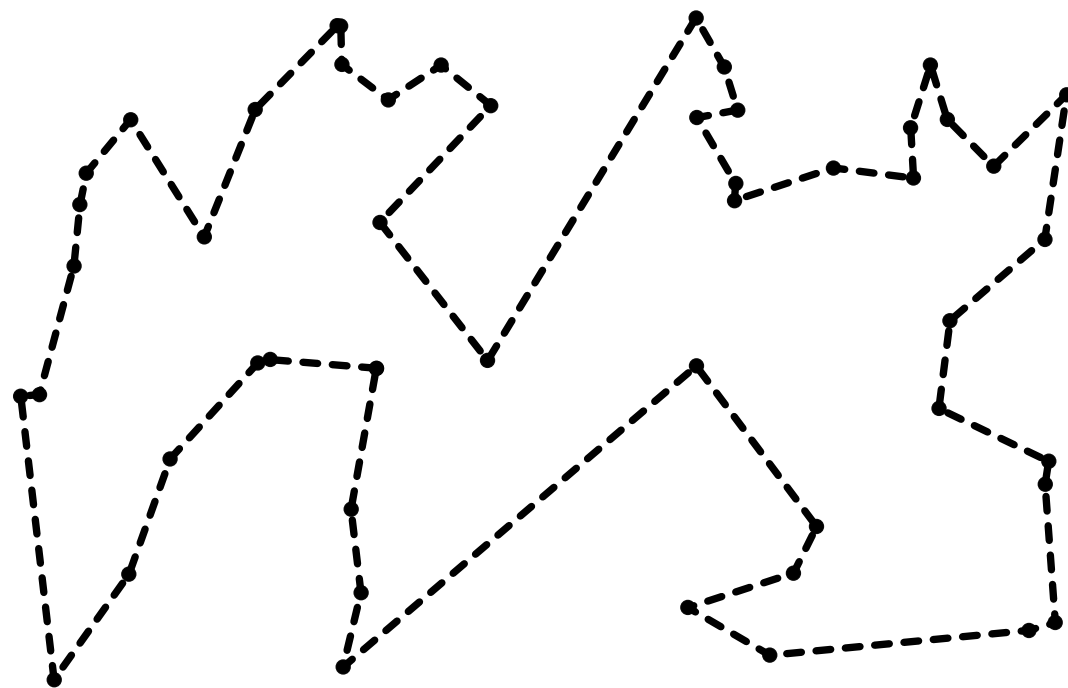
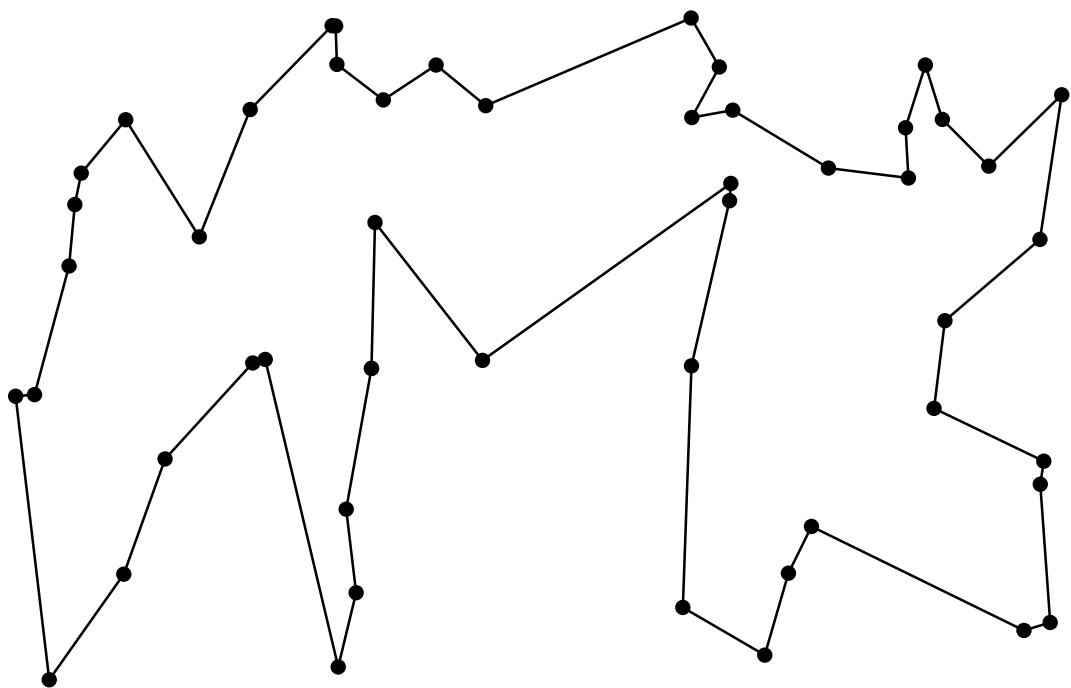
- IMO to szkielety algorytmów, które odwołują się do reprezentacji rozwiązań za pomocą operatorów
- Cechy problemów, które powodują, że nadają się one do stosowania IMO zależą więc od operatorów, a nie od samego problemu!
- Z drugiej strony pytanie, czy dla każdego problemu da się zdefiniować zestaw operatorów zapewniający występowanie pożądanych cech?
  - Na pewno da się „zepsuć”

# Operatory rekombinacji zachowujące dystans – distance preserving crossover

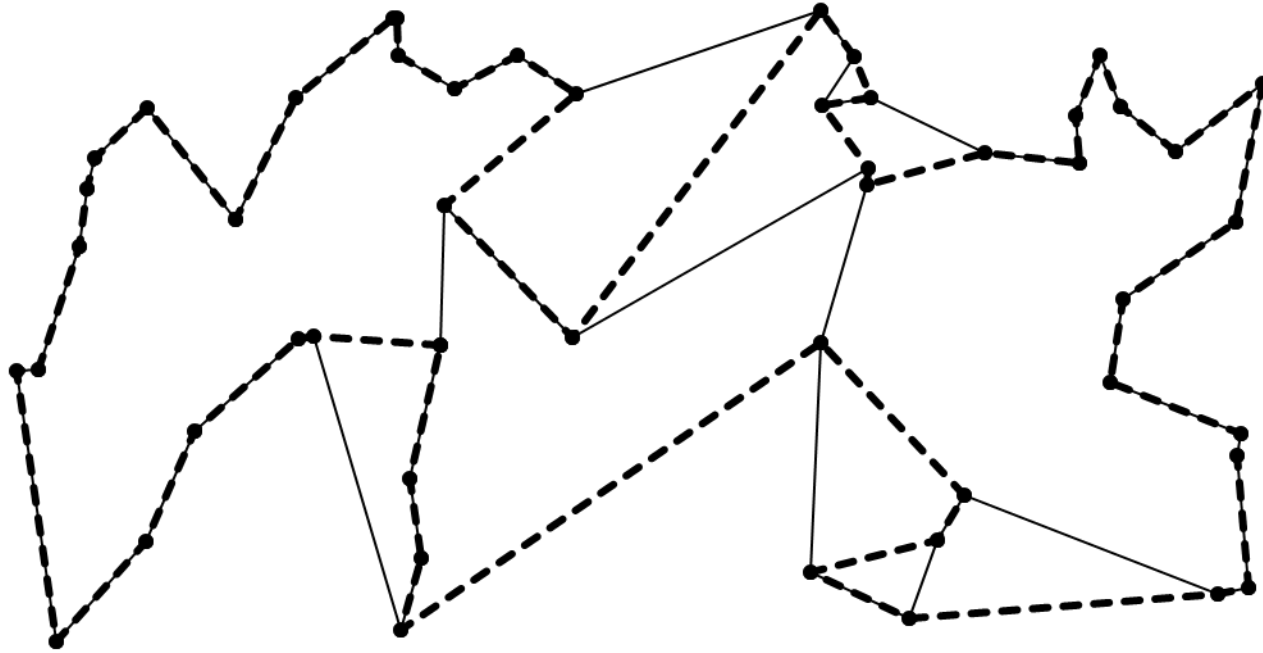
- Dystans – miara przeciwna do podobieństwa – liczba różnych cech
- Zachowywanie dystansu = zachowywanie wspólnych cech rodziców
- Potomek zawiera wszystkie cechy wspólne u obu rodziców. Reszta jest uzupełniana losowo lub heurystycznie.



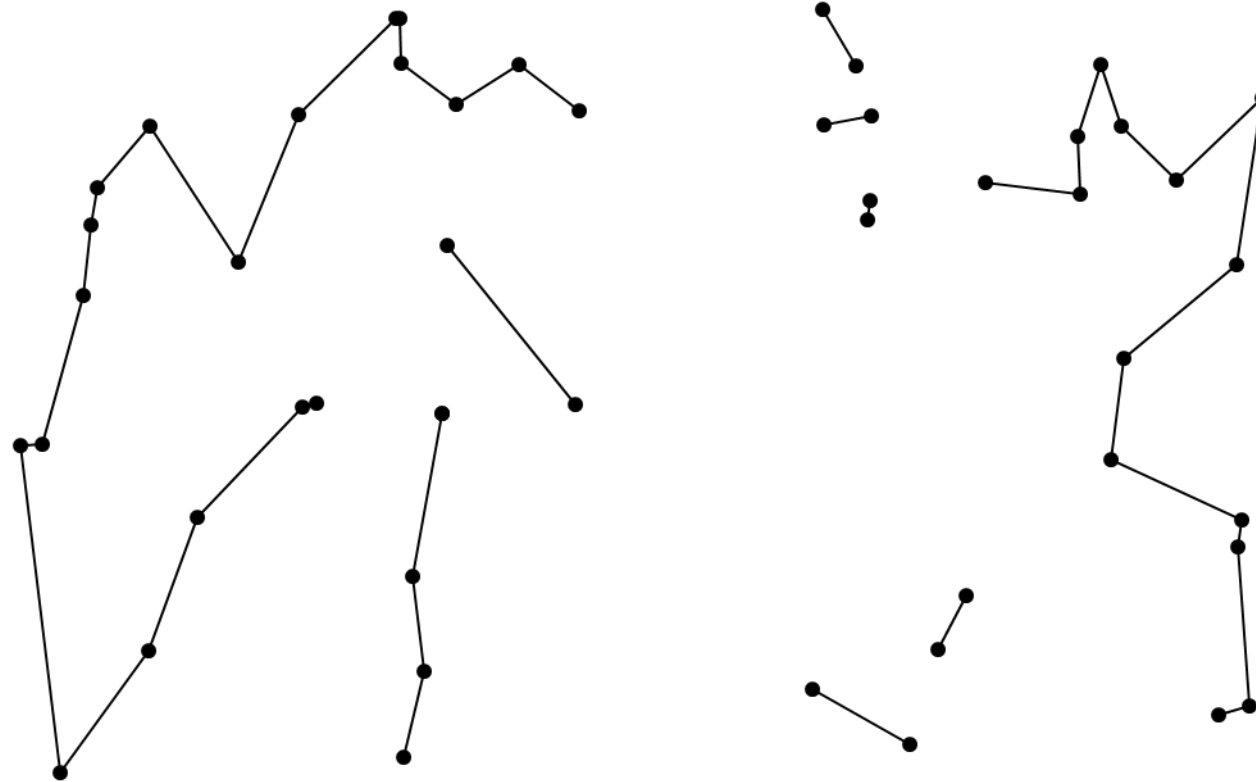
# Dwa przykładowe optima lokalne problemu komiwojażera



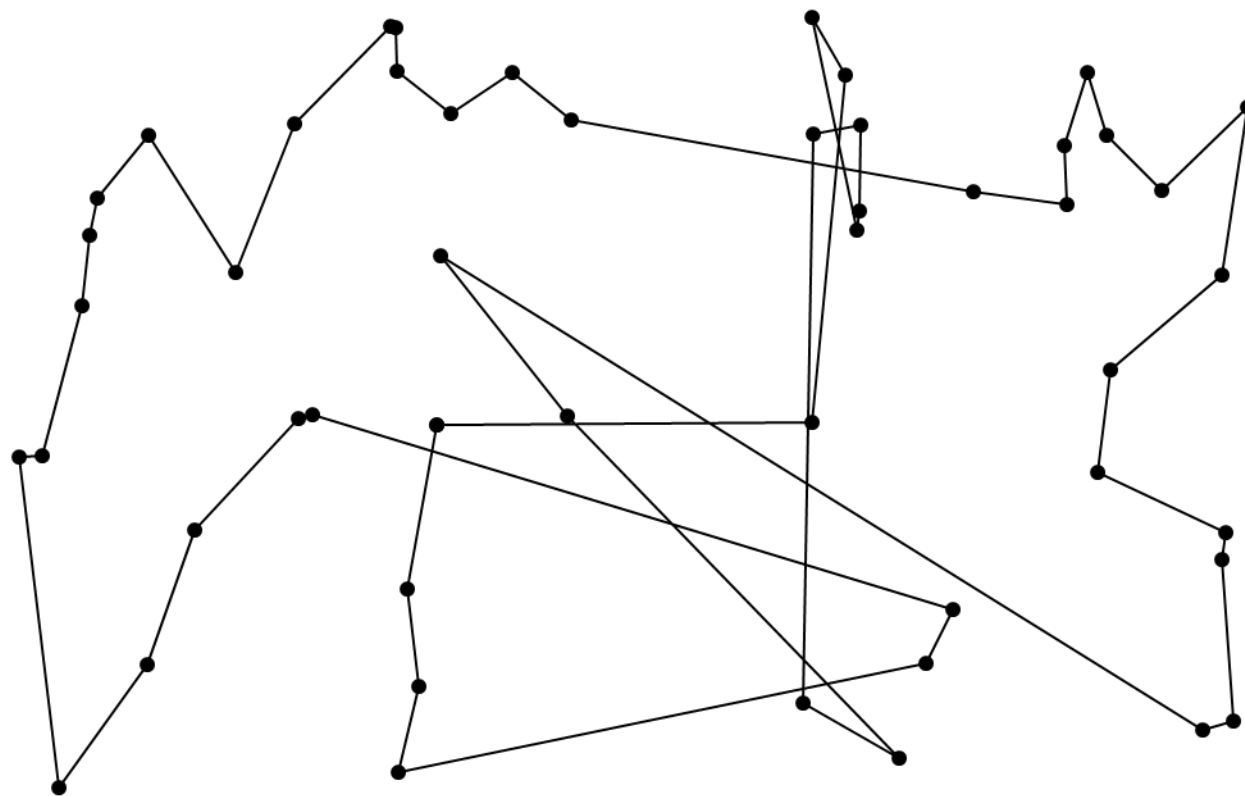
# Dwa przykładowe optima lokalne problemu komiwojażera



# Wspólne krawędzie (i ścieżki)



# Losowe połączenie wspólnych ścieżek

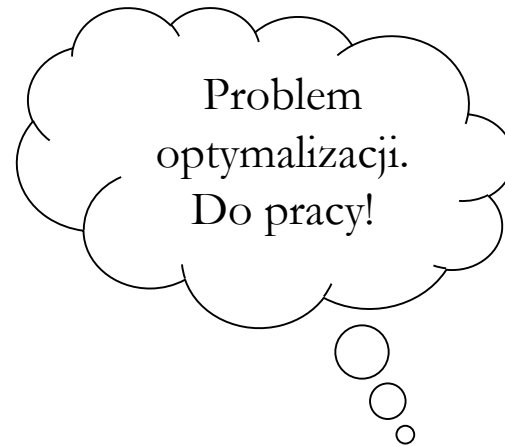
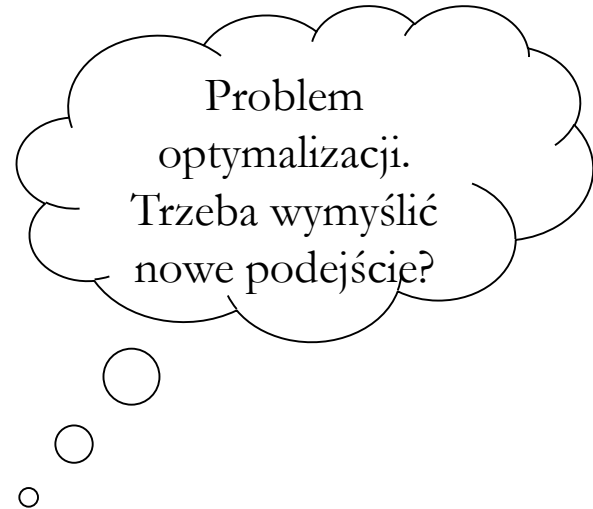




# Projektowanie IMO dla praktycznych problemów

- Uzyskanie w praktyce dobrych wyników przy zastosowaniu IMO nie jest ani oczywiste, ani łatwe
  - Najlepsze adaptacje IMO dla klasycznych (prostych w definicji problemów) bazują na wynikach nawet kilkudziesięciu lat badań (np. TSP)
    - Czas „raczej nieakceptowalny w praktyce”
  - Algorytmy bazujące na IMO muszą być konkurencyjne dla innych metod np. solverów programowania matematycznego

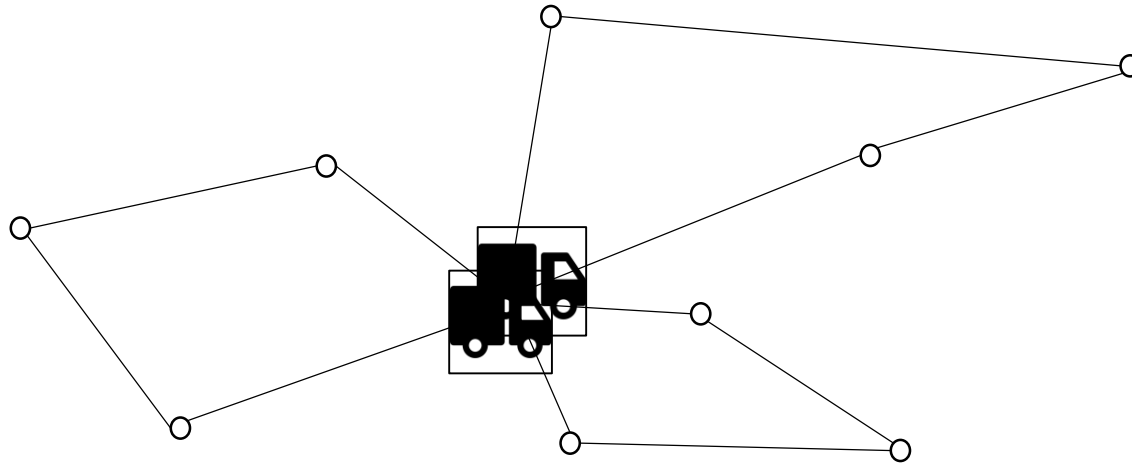
# „Naukowe/artystyczne” vs inżynierskie podejście do adaptacji IMO do konkretnego problemu



# Systematyczna/inżynierska konstrukcji efektywnych algorytmów optymalizacji

- Skonstruuj efektywną metodę lokalnego przeszukiwania
  - Heurystyczne tworzenie dobrych rozwiązań początkowych
  - Wybierz sąsiedztwo o niskim współczynniku autokorelacji
  - Wykorzystaj takie techniki jak delty, wykorzystywanie ocen ruchów z poprzednich iteracji, ruchy kandydackie, ogólne techniki poprawy efektywności kodu
- Sformułuj hipotezy co do istotnych cech rozwiązań
- Wykonaj testy korelacji jakość-odległość dla różnych miar podobieństwa – różnych cech rozwiązań
- Jeżeli obserwujesz korelację dla pewnych cech, to zaprojektuj operator(y) rekombinacji/destroy-repair wykorzystujące te cechy i zastosuj hybrydowy algorytm populacyjny
- Jeżeli nie ma korelacji, zastosuj Lokalne przeszukiwanie z wieloma punktami startowymi

# Przykład – marszrutyzacja pojazdów VRP – różne wersje



# Testowane cechy

- Liczba wspólnych par dowolnych wierzchołków – przedzielonych w obu rozwiązaniach do tej samej trasy
- Liczba wspólnych krawędzi
- Liczba wspólnych łuków
- Liczba wspólnych par wierzchołków nie połączonych łukami/krawędziami
- Liczba wspólnych par krawędzi
- Liczba wspólnych uporządkowanych par wierzchołków
- Różnica względnych pozycji wierzchołków w trasach

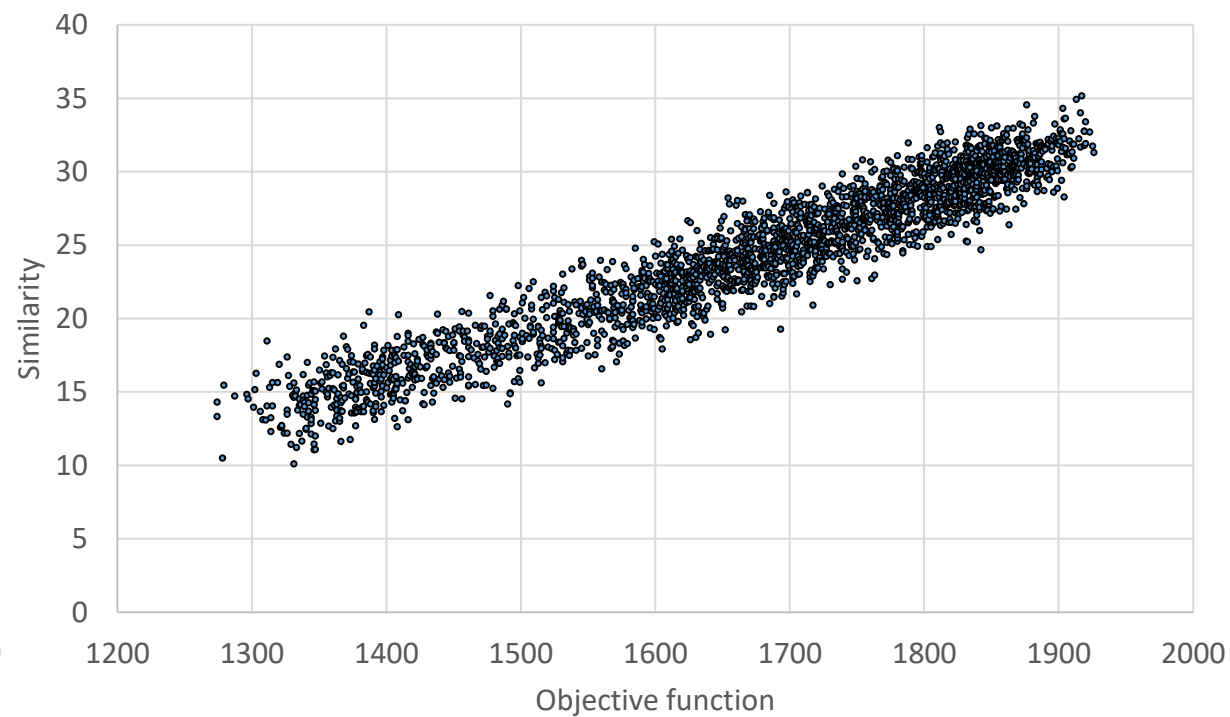
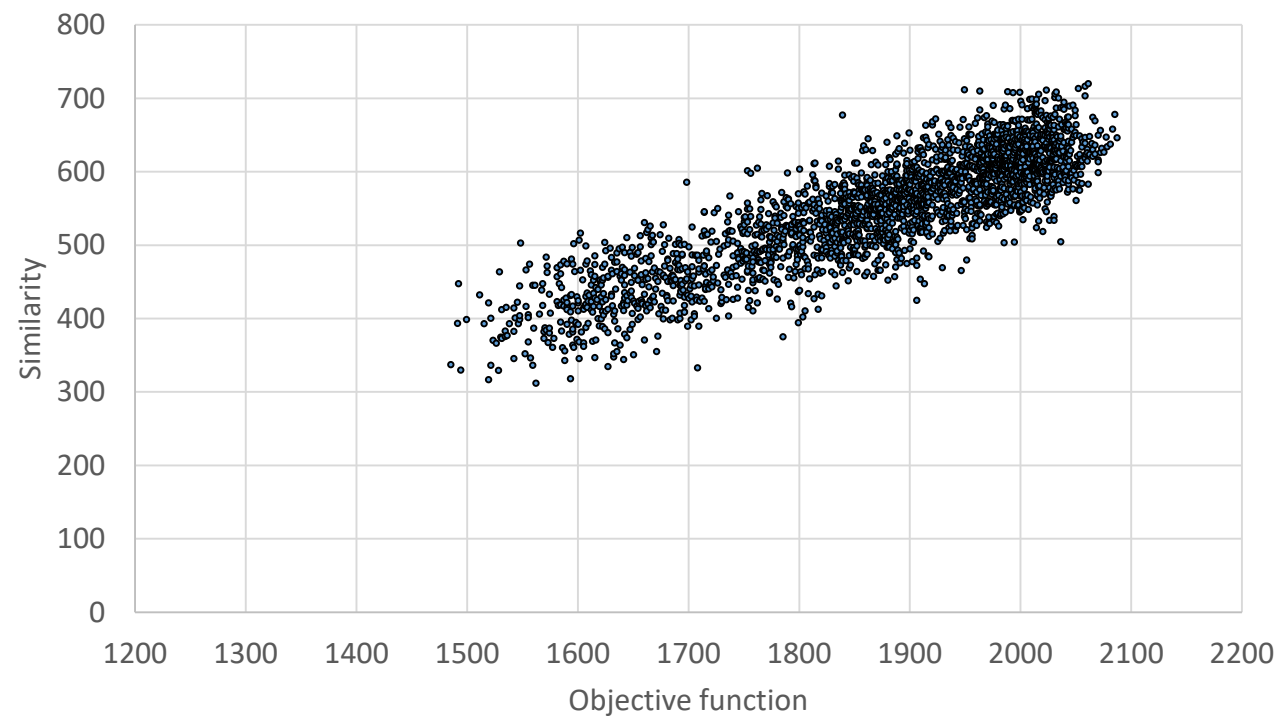
# Wielkości korelacji

	Miara podobieństwa						
Instancja	1	2	3	4	5	6	7
CVRPTW							
rc_2_4_1	-0.79	-0.4	-0.34	-0.78	-0.67	-0.44	0.02
rc_2_8_1	-0.7	-0.62	-0.59	-0.69	-0.66	-0.56	-0.32
rc_1_4_10	-0.42	-0.21	-0.28	-0.4	-0.28	-0.25	-0.2
rc_1_8_10	-0.33	-0.13	-0.23	-0.33	-0.21	-0.37	-0.2
c_1_10_10	-0.81	-0.65	-0.68	-0.81	-0.74	-0.76	-0.48
rc_2_10_2	-0.79	-0.15	-0.16	-0.79	-0.5	-0.42	-0.13
PDPTW							
lc1_4_4	-0.75	-0.52	-0.56	-0.72	-0.62	-0.4	-0.43
lc2_10_4	-0.51	-0.29	-0.39	-0.5	-0.43	-0.4	-0.34
lr1_6_3	-0.62	-0.6	-0.56	-0.59	-0.55	-0.16	-0.17
lr2_10_8	-0.5	-0.46	-0.39	-0.5	-0.55	-0.28	-0.18
lr2_6_10	-0.49	-0.46	-0.42	-0.49	-0.52	0.05	-0.32
lrc2_10_3	-0.57	-0.36	-0.29	-0.56	-0.35	-0.1	-0.07
lrc2_4_10	-0.52	-0.45	-0.39	-0.52	-0.51	-0.08	-0.26
lrc2_8_10	-0.51	-0.43	-0.38	-0.5	-0.47	-0.02	-0.29
TOPTW							
pr05	0.8	0.92	0.89	0.78	0.87	0.49	0.84
pr10	0.89	0.95	0.93	0.88	0.91	0.6	0.9
pr15	0.83	0.94	0.89	0.81	0.86	0.62	0.85
pr16	0.83	0.93	0.89	0.81	0.87	0.57	0.87
pr19	0.83	0.93	0.88	0.82	0.86	0.51	0.79
pr20	0.86	0.93	0.9	0.84	0.89	0.54	0.81

Minimalizowana f. celu

Maksymalizowana f. celu

# Przykłady korelacji– maksymalizowana funkcja celu



# Rekombinacja trasowa (route-based) dla VRP

Rozwiązanie potomne  $O \leftarrow$  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  $O$

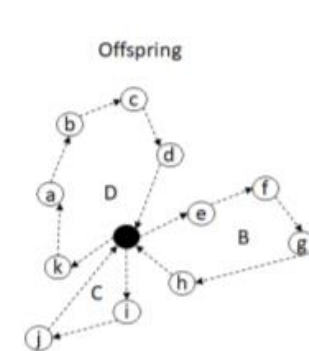
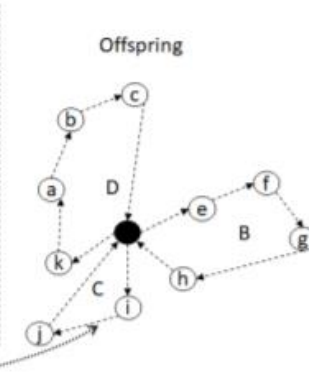
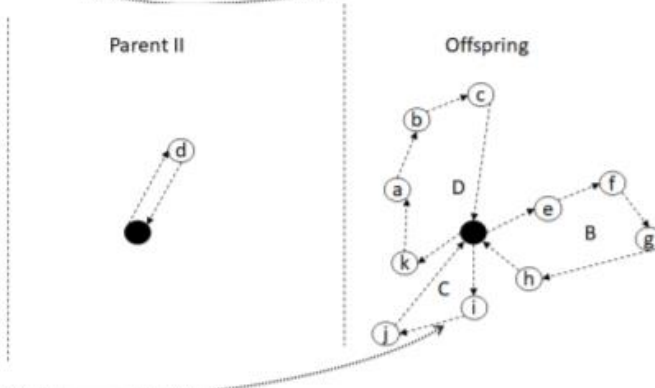
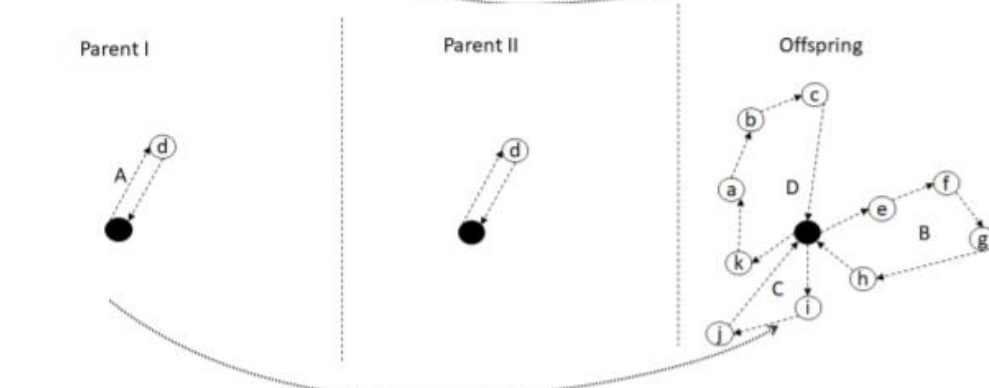
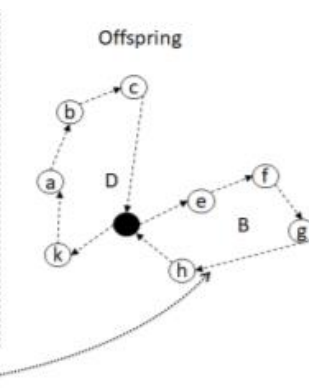
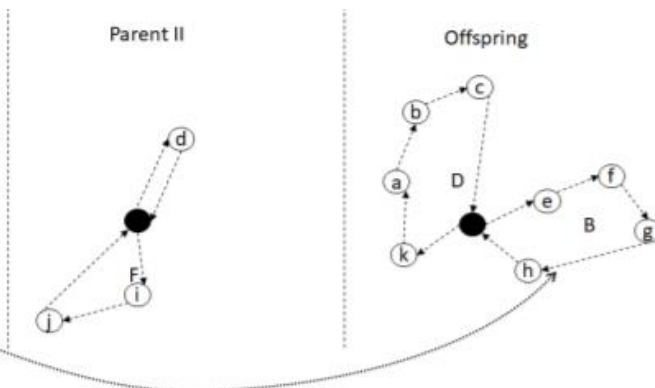
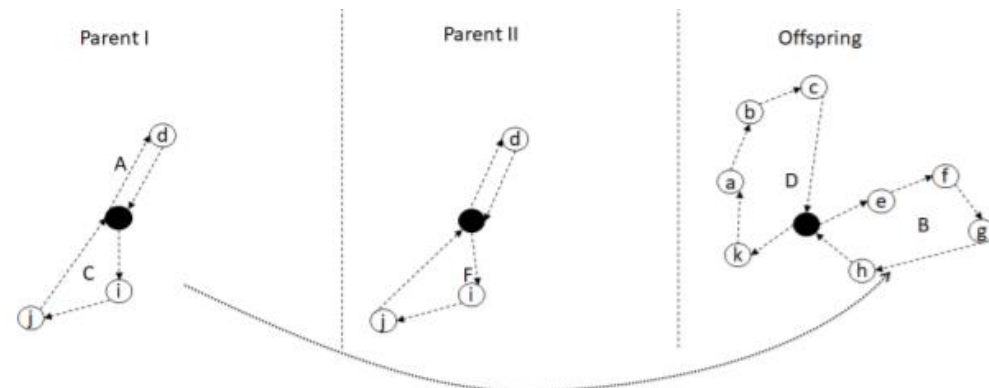
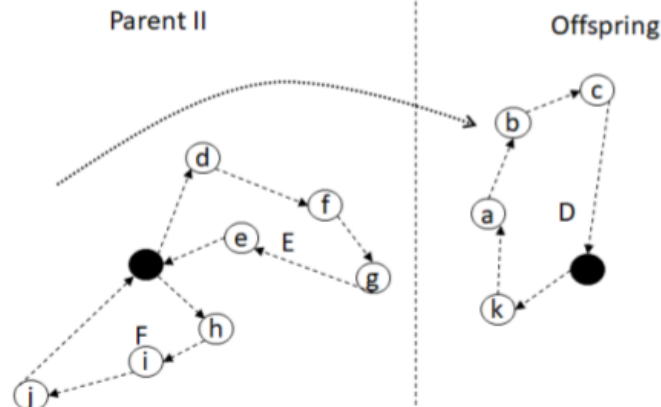
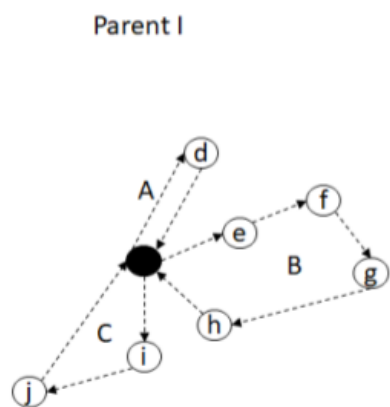
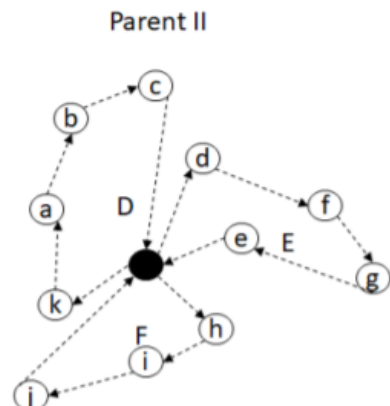
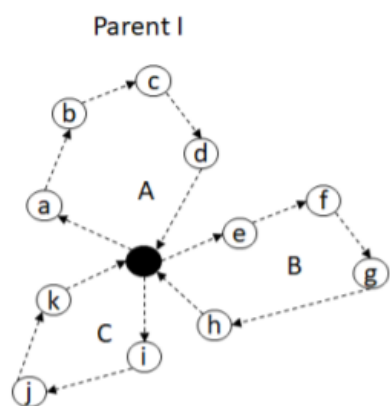
**dopóki** Liczba tras w potomku  $O$  nie jest równa liczbie tras w  $p_A$

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

# Order-based recombination

Znajdź względne pozycje wierzchołków w rodzicach A i B

Znajdź w A trasy, które zawierają co najmniej jedną parę wierzchołków o względnych pozycjach innych niż w B

Posortuj te trasy wg. względnych pozycji w B

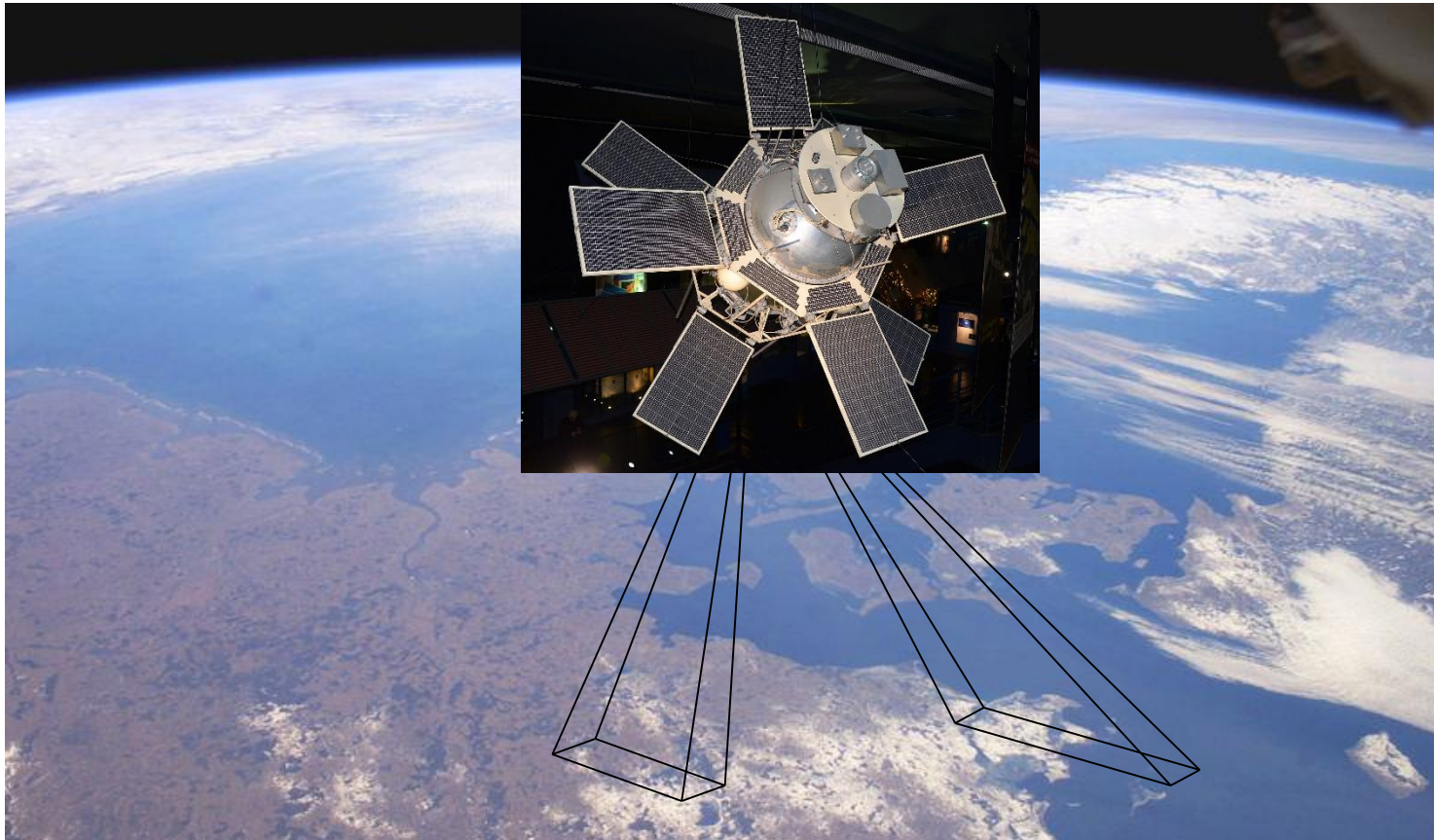
Dla każdej kombinacji 1, 2 lub 3 tras

Jeżeli taka kombinacja nie była jeszcze rozważana, zamień te trasy w potomku

Jeżeli zostały już przejrzone wszystkie powyższe kombinacje

Wykonaj losową liczbę inwersji par wierzchołków

# Case study – zarządzanie satelitami obserwacyjnymi



# Opis problemu

- Zamówienia – zbiory zdjęć
- Zdjęcia zwykłe – wymagane jedno ujęcie
- Zdjęcia 3D – wymagane dwa ujęcia
- Dane zdjęcie można wykonać na dwa sposoby (ujęcia) w zależności od kierunku ruchu kamery
- Zysk z realizacji zlecenia zależy w sposób nieliniowy od zrealizowanej powierzchni zlecenia
- Dla każdego ujęcia istnieje najwcześniejszy i najpóźniejszy czas rozpoczęcia wynikający z pola widzenia satelity – okno czasowe
- Każdego ujęcie ma czas realizacji
- Przełączanie się pomiędzy ujęciami zajmuje znaczący czas zależny od ich położenia
- **Cel: Wybrać zdjęcia (ujęcia) i ustalić plan ich realizacji maksymalizujący zysk**

# Analogie z klasycznymi problemami

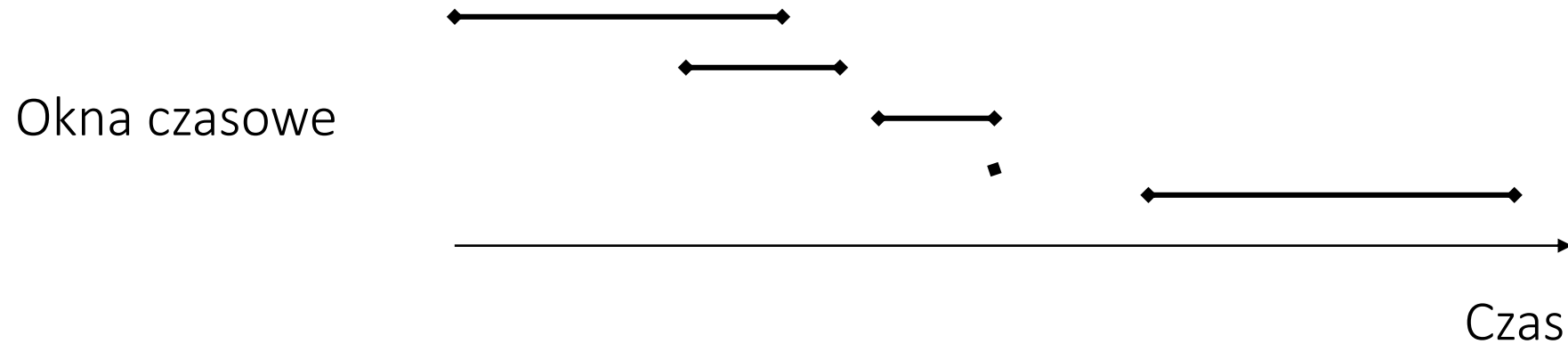
- Problem plecakowy – wybór zdjęć
  - Bardziej skomplikowane ograniczenia i obliczanie funkcji celu
- Szeregowanie zadań – szeregowanie zadań na jednej maszynie z czasami przełączania i oknami czasowymi
  - Celem (bezpośrednim) nie jest minimalizacja czasu realizacji
  - W szeregowaniu z reguły wszystkie zadania muszą być zrealizowane, tu większość nie jest
- Problem komiwojażera – zadania to miasta, czasy przełączania to czasy przejazdu pomiędzy miastami
  - Nie wszystkie zadania muszą być zrealizowane, większość nie jest
  - Dochodzą czasy realizacji zdjęć
- Marszrutyzacja pojazdów (*vehicle routing*) – zadania to miasta, czasy przełączania to czasy przejazdu pomiędzy miastami, czasy realizacji zdjęć do czasu załadunku/rozładunku, okna czasowe załadunku/rozładunku
  - Nie wszystkie zadania muszą być zrealizowane, większość nie jest

# Analogie z innymi problemami

- Problem planowania wycieczki - zadania to odwiedzane miejsca, czasy przełączania to czasy przejazdu, czasy realizacji zdjęć do czasy pobytu, okna czasowe – dla atrakcji czasowych
  - Niewiele prac nt. tego problemu

# Reprezentacja rozwiązania

- Uporządkowana lista (sekwencja) wybranych ujęć
- Wynikają z tego okna czasowe czasu rozpoczęcia

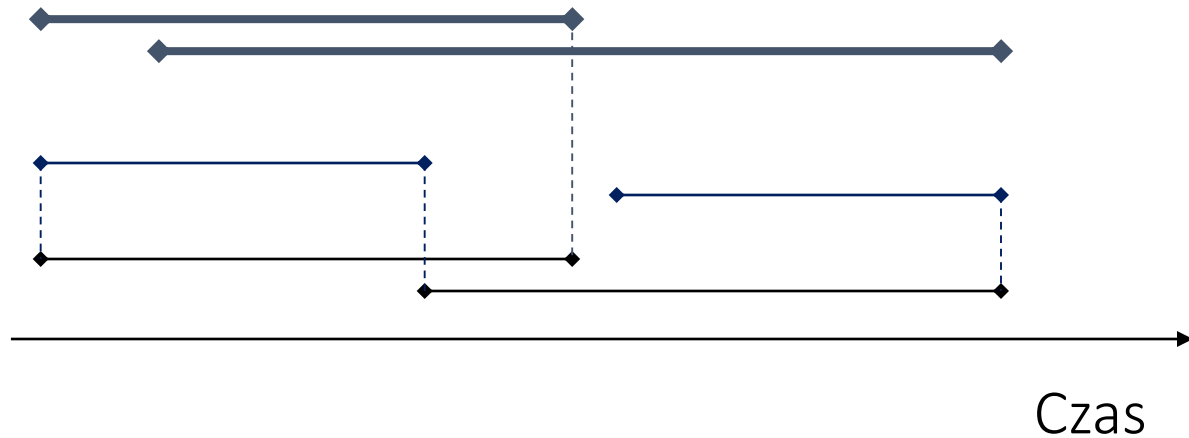




# Propagacja ograniczeń czasowych

Oryginalne okna  
czasowe

Czas realizacji  
i przełączenia



# Konstrukcja rozwiązania początkowego

- Heurystyka zachłanna motywowana heurystykami dla problemu plecakowego
- W danym kroku wstawiane jest jedno ujęcie
- Wybierane jest ujęcie i miejsce o najlepszym stosunku wzrostu zysku do zajętego czasu
- Zajęty czas uwzględnia realizację ujęcia i czasy przełączania
- Przybliżone (liniowe) obliczanie zysku
- Okna czasowe pozwalają zredukować zakres przeszukiwań
- Randomizacja - pewna liczba początkowych ujęć jest wstawiana losowo

# Lokalna optymalizacja

- Ruch – wstawienie ujęcia w najlepszym, tj. dającym największy wzrost zysku, miejscu połączone z usunięciem niezbędnych ujęć
- Zysk jest obliczany dokładnie
- Okna czasowe pozwalają zredukować zakres przeszukiwań

# Problem jest trudny!

- Proste rozszerzenia lokalnego przeszukiwania – przeszukiwanie tabu i iteracyjne lokalne przeszukiwanie nie dają zauważalnej poprawy wyników
  - ... lub słabe sąsiedztwo

# Hipotezy co do istotnych cech

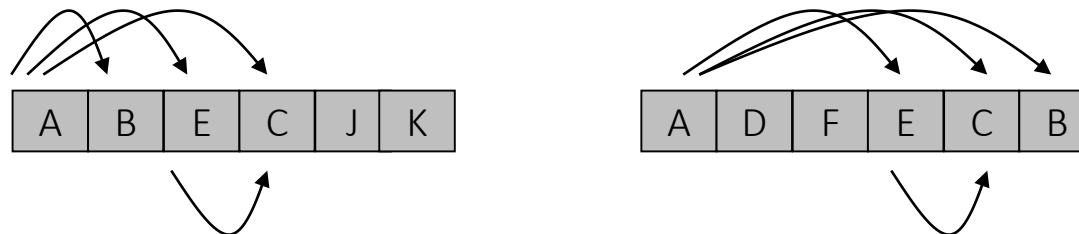
- Wybrane ujęcia



- Zamówienia wybrane do realizacji – zrealizowana powierzchnia



- Kolejność par ujęć

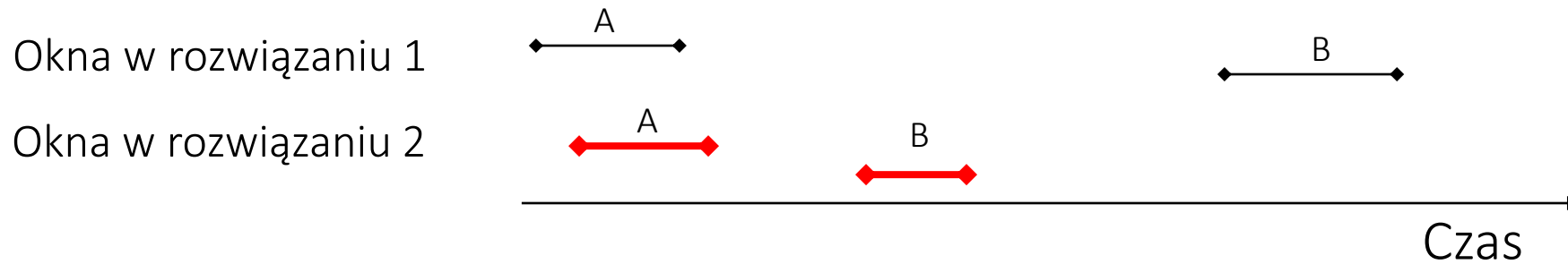


# Hipotezy co do istotnych cech

- Pary sąsiednich ujęć – odpowiednik łuków w problemie komiwojażera

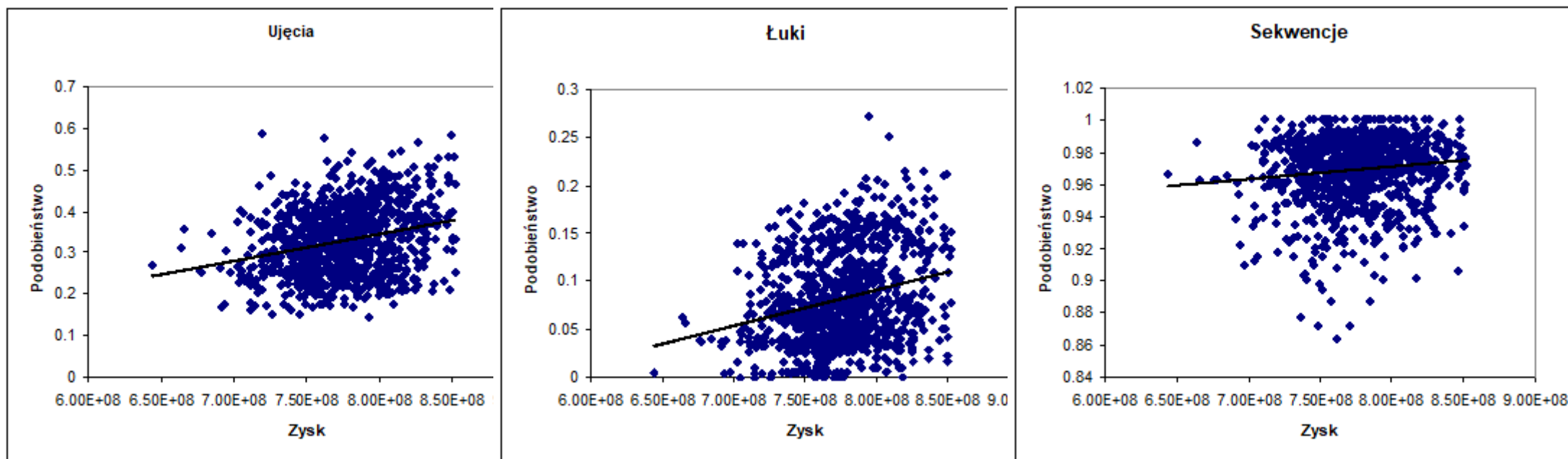


- Okna czasowe rozpoczęcia – odległość pomiędzy punktami centralnymi



# Korelacje jakość-odległość

- Dla danego rozwiązania średnia odległość od wszystkich lepszych rozwiązań

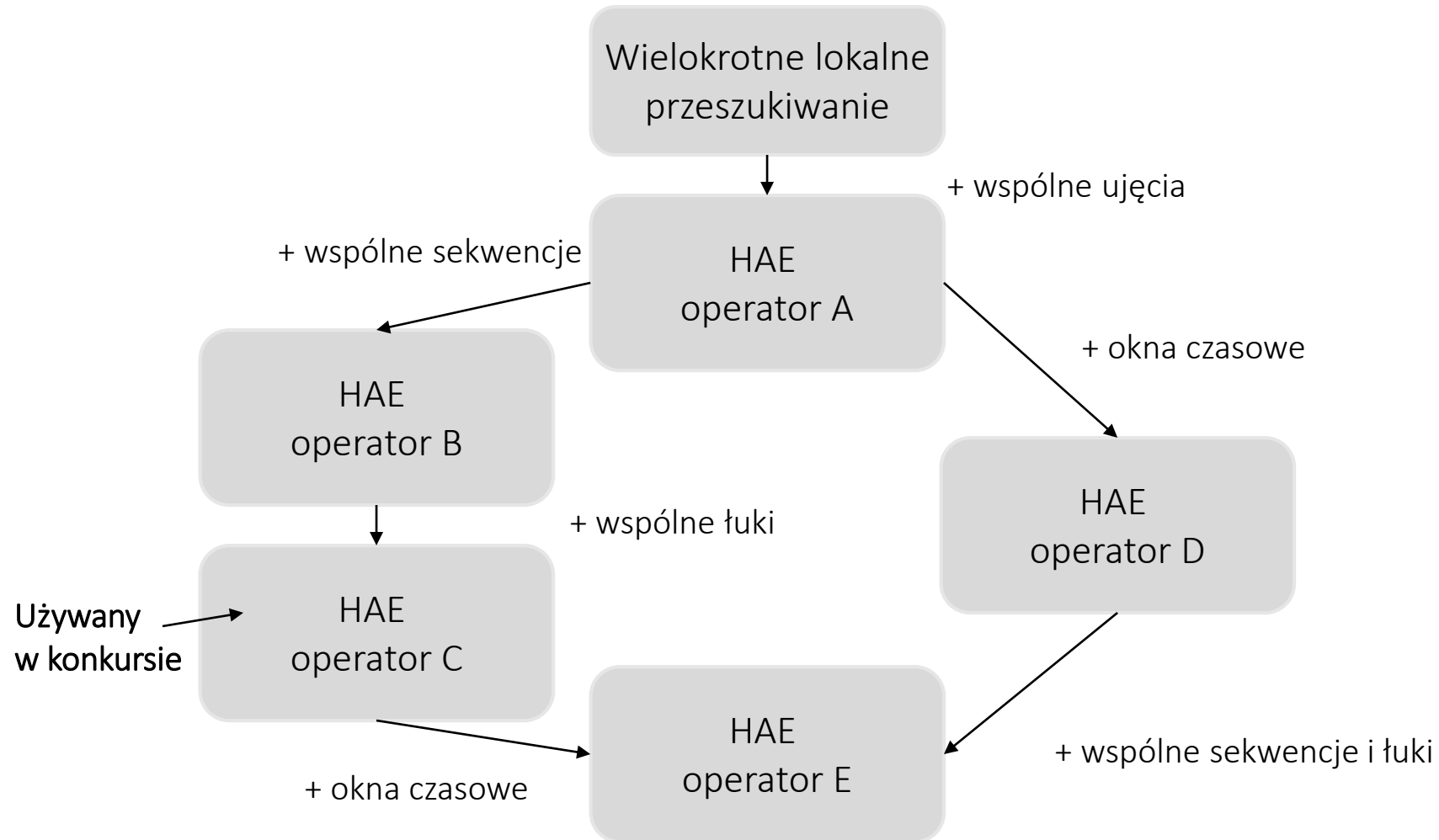


# Wnioski

- Operator rekombinacji powinien zachowywać wspólne
  - Ujęcia (zapewnia zachowanie zamówień)
  - Sekwencje
  - Łuki
  - Okna czasowe (tj. ograniczać je)



# Seria operatorów rekombinacji na bazie heurystyki dla rozwiązań początkowych



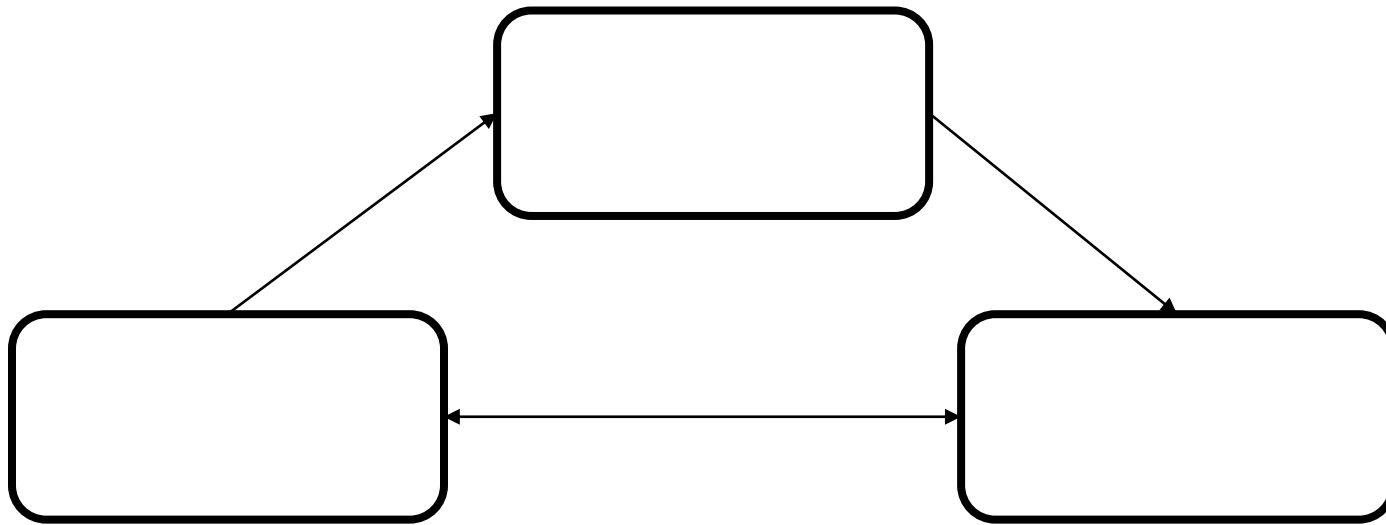
# Operator rekombinacji i hybrydowy algorytm ewolucyjny

- Rozwiązanie jest uzupełniane poprzez heurystykę wstawiania ujęć
- Lokalna optymalizacja jest uruchamiana z prawdopodobieństwem 2.5%
- Wspólne łuki nie mogą być rozrywane
- Algorytm z pełną elitarnością
- Obserwacje:
  - Znacząca poprawa wyników w stosunku do lokalnego przeszukiwania mimo małego prawdopodobieństwa poprawy rozwiązania w danym kroku
  - Długi czas obliczeń dla dużych instancji
  - Duży rozrzut wyników dla dużych instancji

# Usprawnienie lokalnego przeszukiwania

- Ograniczona liczba przeglądanych ruchów
- Rozważane jest wstawienie wszystkich ujęć, które występowały w przynajmniej jednym rodzicu
- Wstawienie pozostałych ujęć jest rozważane z prawdopodobieństwem 10%
- Obserwacje
  - Zadowalający czas obliczeń
  - Duży rozrzut wyników dla dużych instancji
  - Niewielka poprawa rozrzutu wyników przy zwiększaniu rozmiaru populacji


# Model wyspowej populacji



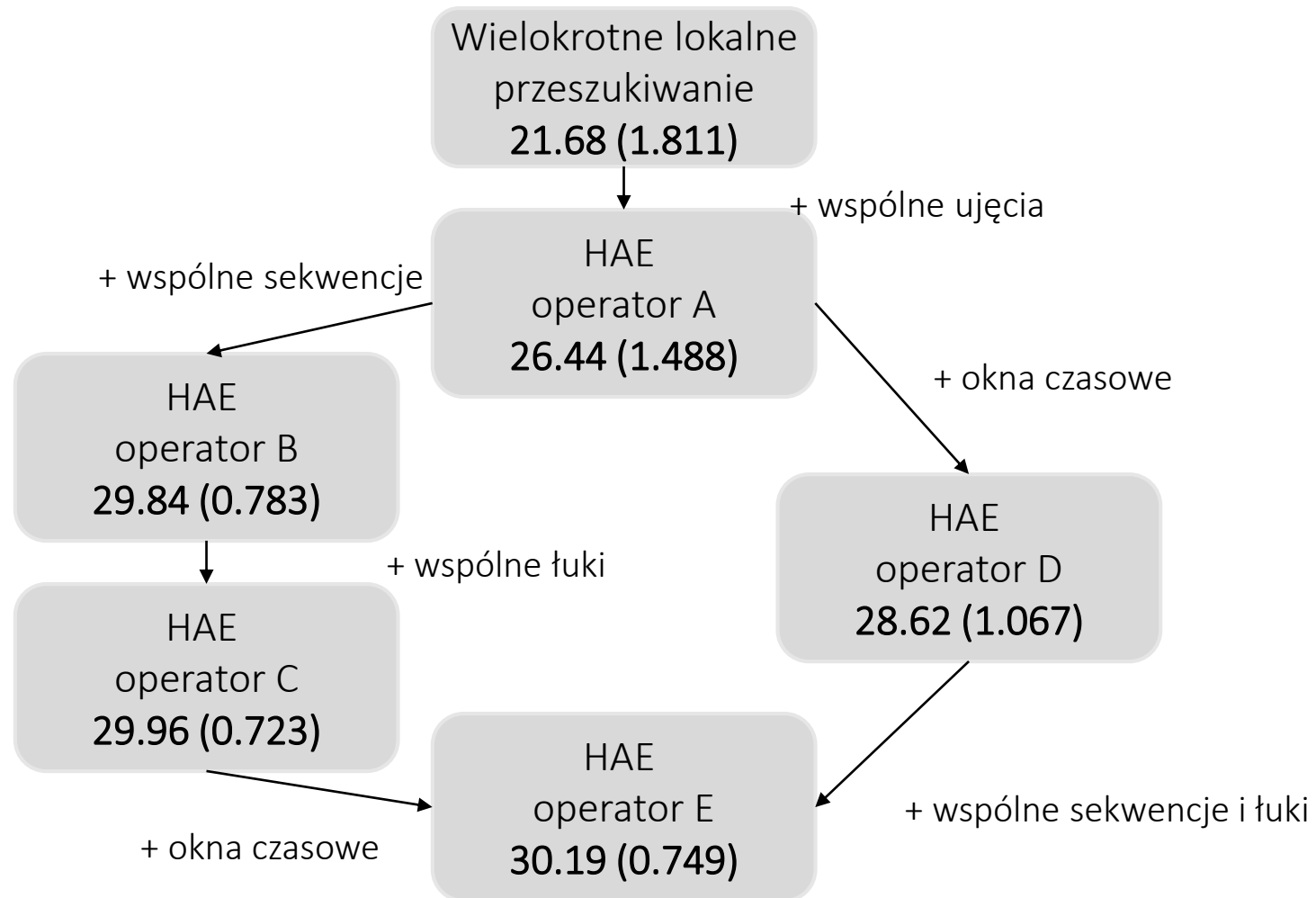
- Obserwacje
  - Zadowalający czas obliczeń
  - Mniejszy rozrzut wyników przy braku poprawy najlepszych wyników

# Wyniki konkursu – operator C

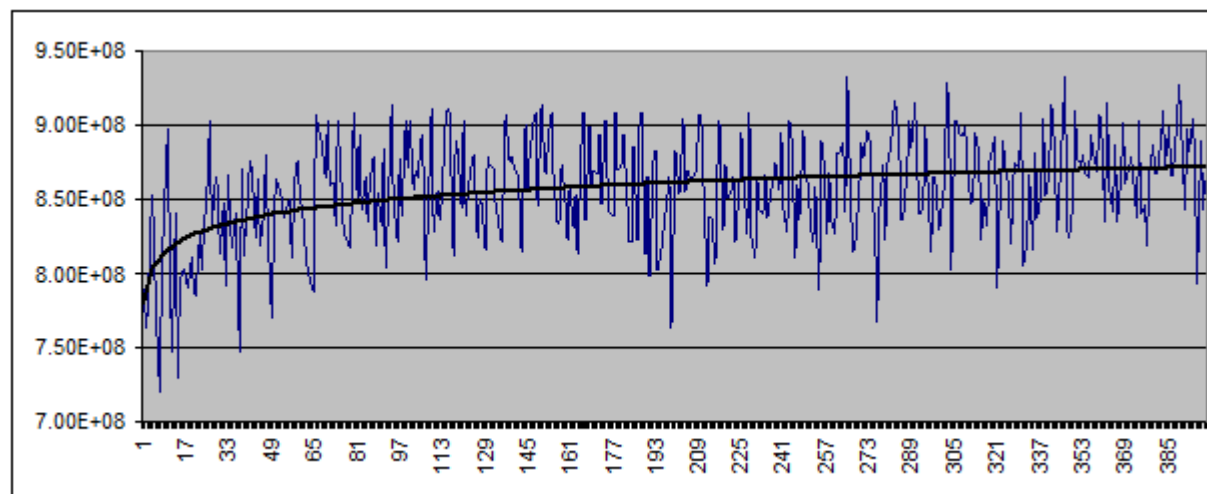
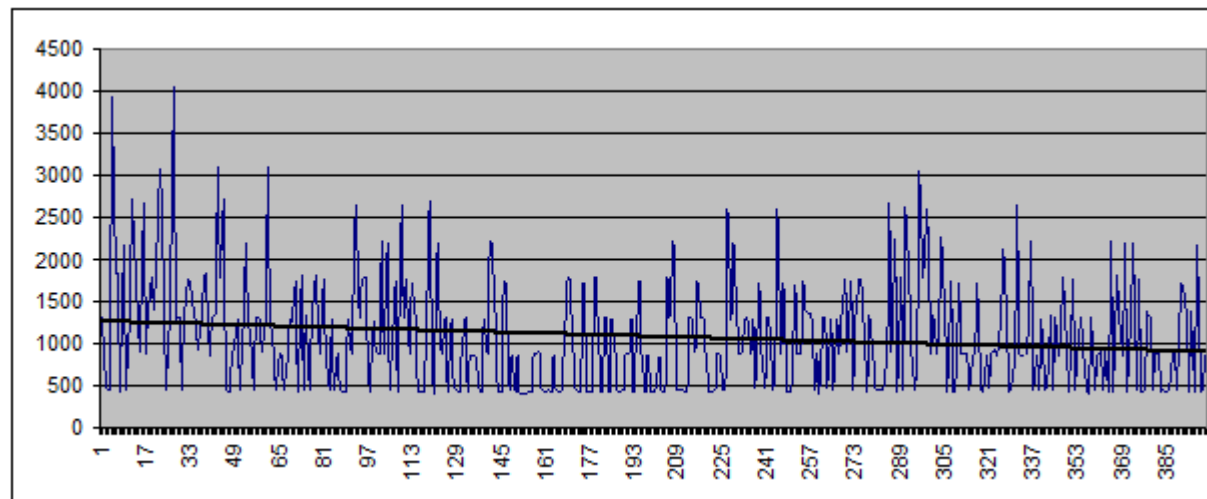
Team	Evaluation
1 (best)	31.76
2	30.84
3	30.28
4	30.1
5 (Author)	29.91
6	29.88
7	29.84
8	29.69
9	29.22
10	29.08
11	23.56



# Porównanie operatorów



# Czas i jakość wyników lokalnego przeszukiwania



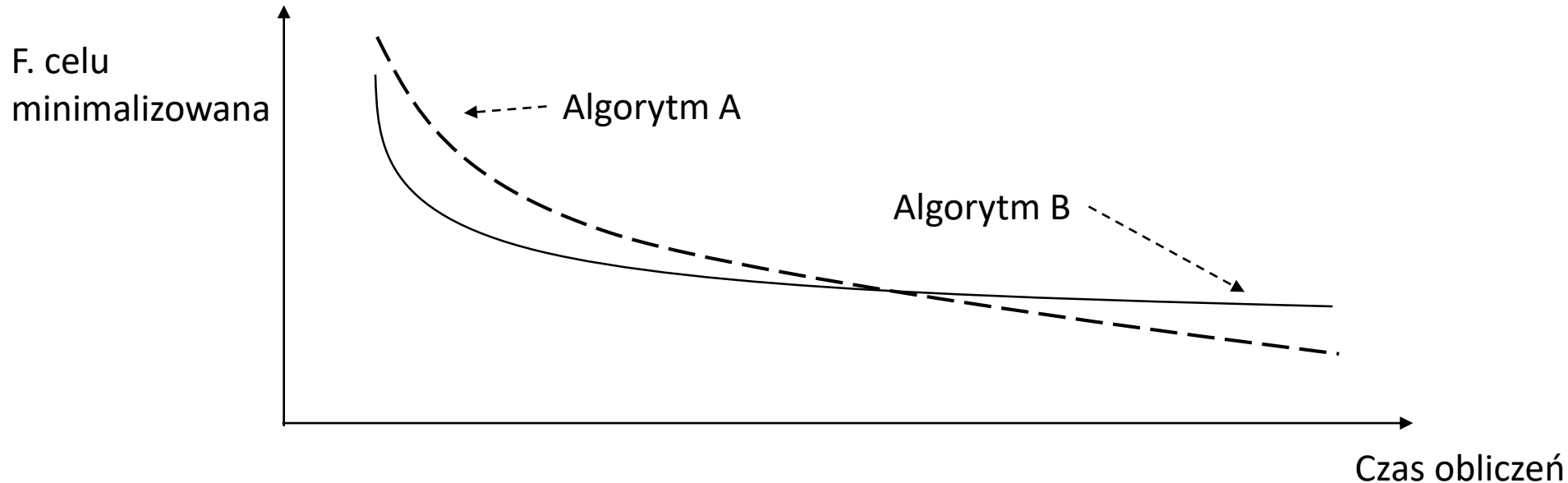
# Podsumowanie – systematycznego podejścia do projektowania IMO

- Operatory sąsiedztwa, rekombinacji, destroy-repair można projektować systematycznie, a nie metodą prób i błędów
  - Oznacza to mniej, krótsze i łatwiejsze w implementacji eksperymenty obliczeniowe
  - Ponadto rozumiemy dlaczego nasz algorytm działa (lub nie działa)



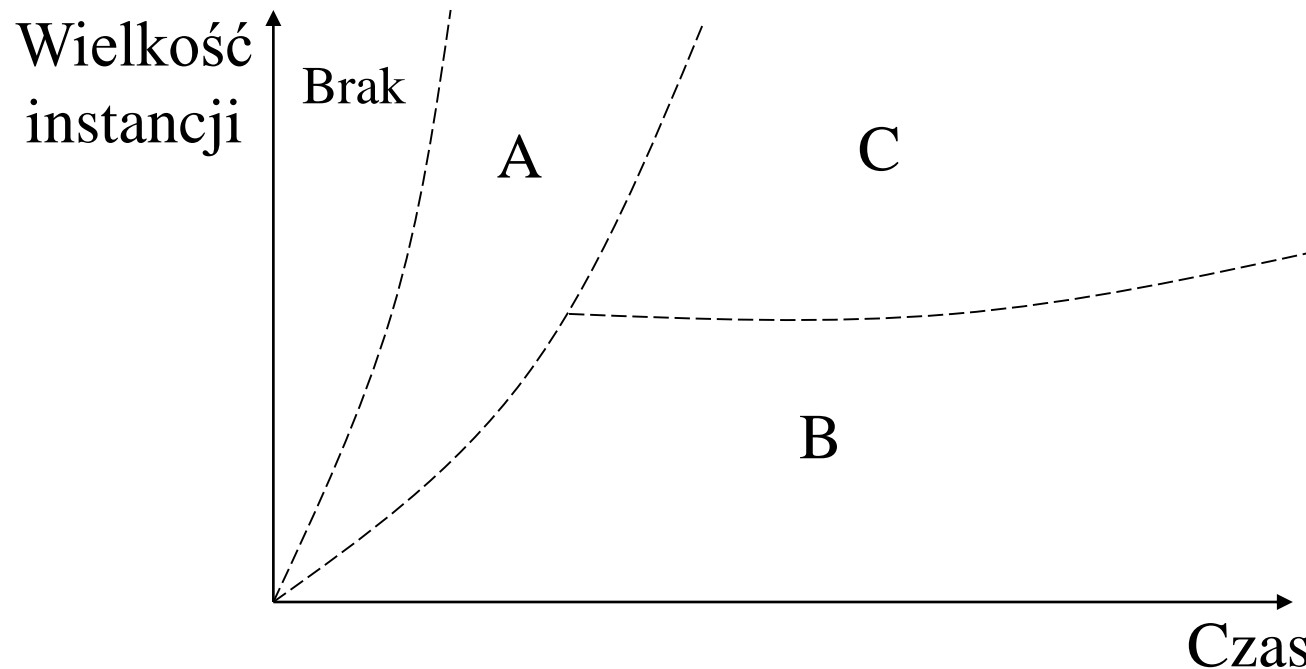
# Eksperymentalna ocena algorytmów heurystycznych

- Dwa główne kryteria:
  - Jakość – wartość funkcji celu
  - Czas obliczeń
- Względna jakość może być różna w zależności od dostępnego czasu obliczeń



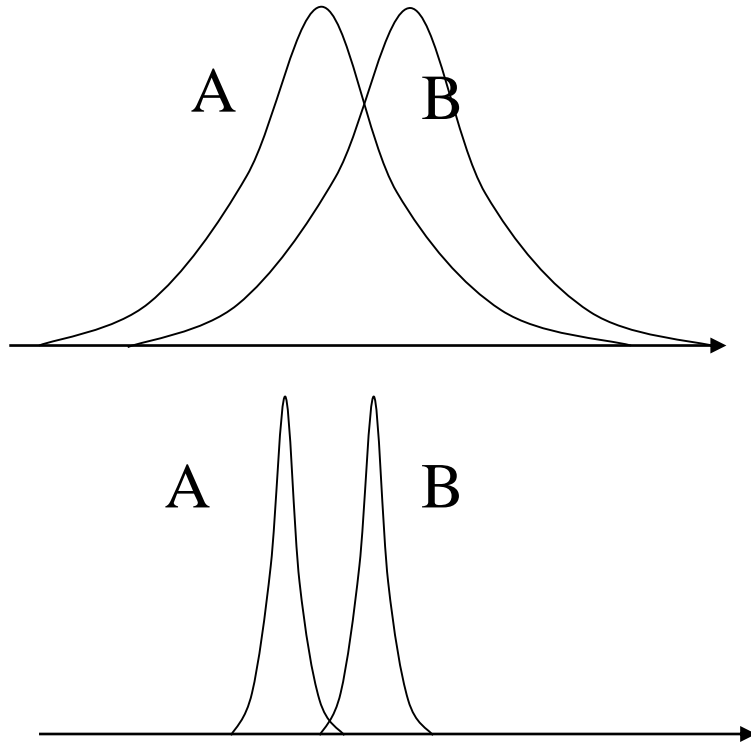
# Ranking na płaszczyźnie czas-wielkość instancji

- Najlepszy algorytm z A, B i C dla danego czasu i wielkości



Względny rozrzut wyników (wartości f. celu) często maleje wraz ze wzrostem wielkości instancji

- Porównanie algorytmów staje się bardziej jednoznaczne



Przykładowe treny – nowe ciekawe pomysły

# Matheuristics - Wykorzystanie IMO w solverach

- Solvery programowania matematycznego - postęp w ostatnich latach
  - Preprocessing – usuwanie zbędnych (zawsze nieaktywnych) ograniczeń i zmiennych
  - Branch and price – połączenie branch and bound z generowaniem kolumn
  - Branch and cut – połączenie branch and bound z cutting planes
- IMO mogą służyć do:
  - ustalania kolejności wyboru gałęzi
  - Heurystycznego wyszukiwania nieaktywnych zmiennych i ograniczeń
  - Heurystycznego generowania kolumn i cutting planes (płaszczyzn odcinających)

# Matheuristics – wykorzystanie solverów w IMO

- Np. operatory rekombinacji wykorzystujące dokładne solvery – np. wybór tras dla problemu VRP spośród większego zbioru tras (problem typu set covering), np. pochodzących z wielu rozwiązań
- Metoda Probe

# AI, ML, NN w IMO

- Np. projektowanie operatorów, heurystyk za pomocą NN
  - Dynamic Partial Removal: A Neural Network Heuristic for Large Neighborhood Search, Mingxiang Chen, Lei Gao, Qichang Chen, Zhixin Liu, arXiv:2005.09330
  - Evolution of Heuristics: Towards Efficient Automatic Algorithm Design Using Large Language Model, ICML 2024.
- Autoencodery jako operatory perturbacji/rekombinacji
- Bezpośrednie generowanie rozwiązań, instancja -> rozwiązanie za pomocą NN
  - Lu Duan, Yang Zhan, Haoyuan Hu, Yu Gong, Jiangwen Wei, Xiaodong Zhang, and Yinghui Xu. 2020. Efficiently Solving the Practical Vehicle Routing Problem: A Novel Joint Learning Approach. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20). 3054–3063.
- Prace przeglądowe
  - Analytics and Machine Learning in Vehicle Routing Research, Ruibin Bai et al., arXiv:2102.10012.
  - Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d’horizon. European Journal of Operational Research, 290(2):405–421, 2021.
  - Quentin Cappart, Didier Chételat, Elias Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks, 2021, arXiv:2102.09544.

# Obliczenia kwantowe

- Quantum adiabatic algorithm (QAA) (quantum annealing) na urządzeniach noisy intermediate-scale quantum (NISQ)
- Quantum approximate optimization algorithm (QAOA)
- Variational quantum algorithms (VQA) – kwantowo-klasyczne

Science

[Current Issue](#) [First release papers](#) [Archive](#) [About](#) [Submit manuscript](#)

RESEARCH ARTICLE



## Quantum optimization of maximum independent set using Rydberg atom arrays

S. EBADI, A. KEESLING, M. CAIRN, T. T. WANG, H. LEVINE, D. BLUVSTEIN, G. SEMEGHINI, A. OMRAN, J.-G. LIU, [...] M. D. LUKIN

+15 authors [Authors Info & Affiliations](#)

SCIENCE • 5 May 2022 • First Release • DOI:10.1126/science.abc6587

131



### Abstract

Realizing quantum speedup for practically relevant, computationally hard problems is a central challenge in quantum information science. Using Rydberg atom arrays with up to 289 qubits in two spatial dimensions, we experimentally investigate quantum algorithms for solving the Maximum Independent Set problem. We use a hardware-efficient encoding associated with Rydberg blockade, realize closed-loop optimization to test several variational algorithms, and subsequently apply them to systematically explore a class of graphs with programmable connectivity. We find the problem hardness is controlled by the solution degeneracy and number of local minima, and experimentally benchmark the quantum algorithm's performance against classical simulated annealing. On the hardest graphs, we observe a superlinear quantum speedup in finding exact solutions in the deep circuit regime and analyze its origins.





# Podsumowanie. Co działa?

- Zachłanne heurystyki konstrukcyjne
- Randomizacja
- Lokalne przeszukiwanie
- Poprawa efektywności lokalnego przeszukiwania
- Perturbacje, mniejsze losowe i większe destroy-repair
- Populacje i rekombinacja
- Zapewnianie odpowiedniej dywersyfikacji
- Projektowanie operatorów na bazie współczynników autokorelacji i globalnej wypukłości
- Zmiany/oscylacje intensyfikacji/dywersyfikacji, poziomu niedopuszczalności

