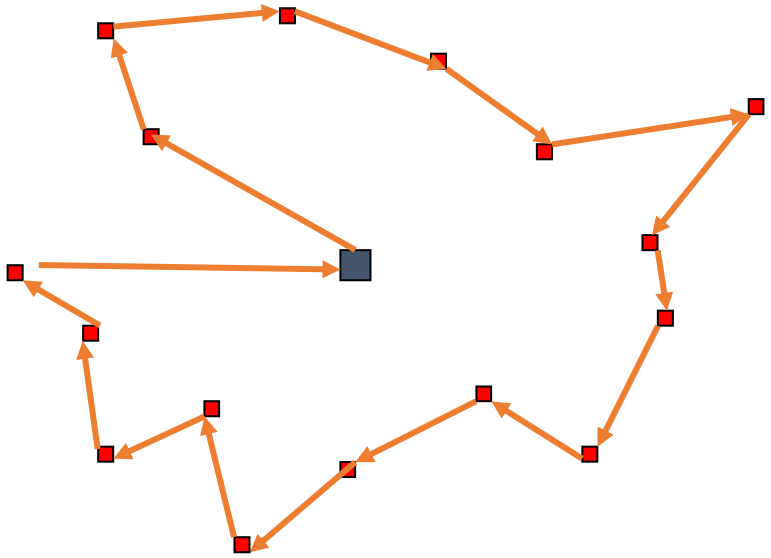
Układanie tras pojazdów. Modele i algorytmy

Radosław Jadczak

Instytut Informatyki i Logistyki
Katedra Badań Operacyjnych

WYDZIAŁ
EKONOMICZNO-SOCJOLOGICZNY
Uniwersytet Łódzki

Układanie trasy jednego pojazdu



1. wszystkie punkty trasy muszą zostać odwiedzone
2. każdy punkt tras może być odwiedzony tylko jeden raz
3. pojazd wraca do bazy po odwiedzeniu wszystkich punktów

Układanie trasy jednego pojazdu

$$min\ F(\mathbf{x}) = \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij}$$

$$\sum_{i=0}^{n} x_{ij} = 1 \qquad \text{dla } j = 0, 1, \ldots, n$$

$$\sum_{j=0}^{n} x_{ij} = 1 \qquad \text{dla } i = 0, 1, \ldots, n$$

$$z_i - z_j + (n+1)x_{ij} \leq (n+1) - 1 \qquad \text{dla } i, j = 1, \ldots, n \text{ oraz } i \neq j$$

$$x_{ij} = \begin{cases} 0 \\ 1 \end{cases} \qquad \text{dla } i, j = 1, \ldots, n$$

	0	1	2	3	4	5	6	7	8	9
0			**1**							
1						**1**				
2					**1**					
3		**1**								
4							**1**			
5								**1**		
6										**1**
7									**1**	
8	**1**									
9				**1**						

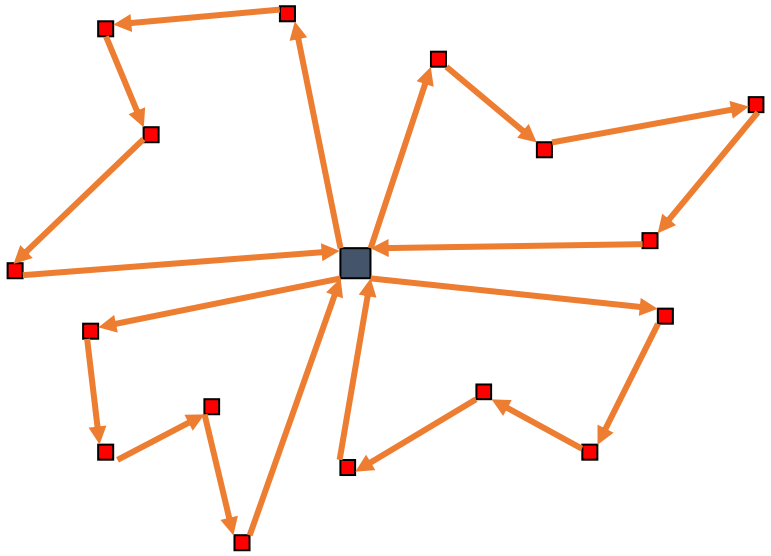
Trasa: 0 → 2 → 4 → 6 → 9 → 3 → 1 → 5 → 7 → 8 → 0

Układanie trasy jednego pojazdu

1. Liczba zmiennych decyzyjnych:
   $(n+1)^2$
2. Liczba ograniczeń:
   $(n+1) + (n+1) + (n^2-n)$
3. Liczba możliwych tras:
   $n!$

n	Liczba tras (symetryczny)	Liczba tras (niesymetryczny)	Liczba zmiennych	Liczba ograniczeń
1	1	1	4	4
2	2	1	9	8
3	6	2	16	14
4	24	12	25	22
5	120	60	36	32
6	720	360	49	44
7	5 040	2 520	64	58
8	40 320	20 160	81	74
9	362 880	181 440	100	92
10	3 628 800	1 814 400	121	112

Układanie tras wielu pojazdów



1. wszystkie punkty trasy muszą zostać odwiedzone
2. każdy punkt tras może być odwiedzony tylko jeden raz
3. każdy punkt tras może być odwiedzony tylko przez jeden pojazd
4. każdy pojazd wraca do bazy po odwiedzeniu wszystkich przydzielonych mu punktów

- podział zbioru wszystkich punktów na rejony dla poszczególnych pojazdów
- konstrukcja trasy pojazdu w ramach rejonu

Układanie tras wielu pojazdów

$$min\ F(\mathbf{x}) = \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{K} c_{ij} x_{ij}^{k}$$

$$x_{ij}^{k} = \begin{cases} 0 \\ 1 \end{cases} \quad \text{dla } i, j = 1, \ldots, n \text{ oraz } k = 1, \ldots, K$$

	0	1	2	3	4	5	6	7	8	9
0			**1**							
1										
2					**1**					
3										
4										**1**
5										
6										
7										
8										
9	**1**									

Trasa k_1: $0 \rightarrow 2 \rightarrow 4 \rightarrow 9 \rightarrow 0$

	0	1	2	3	4	5	6	7	8	9
0				**1**						
1						**1**				
2										
3		**1**								
4										
5	**1**									
6										
7										
8										
9										

Trasa k_2: $0 \rightarrow 3 \rightarrow 1 \rightarrow 5 \rightarrow 0$

	0	1	2	3	4	5	6	7	8	9
0								**1**		
1										
2										
3										
4										
5										
6	**1**									
7									**1**	
8							**1**			
9										

Trasa k_3: $0 \rightarrow 7 \rightarrow 8 \rightarrow 6 \rightarrow 0$

Układanie tras wielu pojazdów

$$min\ \ F(\mathbf{x}) = \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=1}^{K} d x_{ij}^{k}$$

$$\sum_{i=0}^{N} \sum_{k=1}^{K} x_{ij}^{k} = 1$$ dla $j = 0,1,\ldots,N$

$$\sum_{i=0}^{N} x_{ip}^{k} - \sum_{j=0}^{N} x_{pj}^{k} = 0$$ dla $p = 0,1,\ldots,N$ oraz $k = 1,\ldots,K$

$$\sum_{j=1}^{N} x_{0j}^{k} \leq 1$$ dla $k = 1,\ldots,K$

$$z_i - z_j + (N+1) \sum_{k=1}^{K} x_{ij}^{k} \leq (N+1) - 1$$ dla $i,j = 1,\ldots,N$ oraz $i \neq j$

$$x_{ij}^{k} = \begin{cases} 0 \\ 1 \end{cases}$$ dla $i, j = 1, \ldots, n$ oraz $k = 1, \ldots, K$

Warianty układania tras wielu pojazdów

CVRP (*Capacitated Vehicle Routing Problem*)

VRPTW (*Vehicle Routing Problem with Time Windows*)

SDVRP (Split Delivery *Vehicle Routing Problem*)

VRPPD (*Vehicle Routing Problem with Pick-Up and Delivering*)

VRPB (*Vehicle Routing Problem with Backhaul*)

DVRP (*Dynamic Vehicle Routing Problem*)

MDVRP (*Multi-Depot Vehicle Routing Problem*)

MVRP (*Multiobjective Vehicle Routing Problem*)

PVRP (Periodic *Vehicle Routing Problem*)

VRPSF (*Vehicle Routing Problem with Satellite Facilities*)

SVRP (Stochastic *Vehicle Routing Problem*)

CVRP (*Capacitated Vehicle Routing Problem*) – ograniczenia zasobowe

1. ograniczenie ładowności pojazdu $\boldsymbol{Q_k}$

$$\sum_{i=1}^{n} q_i \sum_{j=0}^{n} x_{ij}^k \leq Q_k \qquad \text{dla } k = 0, 1, \ldots, K$$

2. ograniczenie pojemności pojazdu $\boldsymbol{V_k}$

$$\sum_{i=1}^{n} q_i \sum_{j=0}^{n} v_{ij}^k \leq V_k \qquad \text{dla } k = 0, 1, \ldots, K$$

3. ograniczenie czasu pracy pojazdu $\boldsymbol{T_k}$

$$\sum_{i=1}^{n} t_i^k \sum_{j=0}^{n} x_{ij}^k + \sum_{i=0}^{n} \sum_{j=0}^{n} t_{ij}^k x_{ij}^k \leq T_k \qquad \text{dla } k = 0, 1, \ldots, K$$

4. ograniczenie długości trasy pojazdu $\boldsymbol{D_k}$

$$\sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij}^k x_{ij}^k \leq T_k \qquad \text{dla } k = 0, 1, \ldots, K$$

WYDZIAŁ
EKONOMICZNO-SOCJOLOGICZNY
Uniwersytet Łódzki

- flota pojazdów ograniczona lub nieograniczona
- pojazdów flota niejednorodna lub niejednorodna

VRPTW (*Vehicle Routing Problem with Time Windows*) – okna czasowe

1. przedział czasu w jakim każdy punkt może być odwiedziny [$\boldsymbol{a_i}$,$\boldsymbol{b_i}$]

2. przybycie pojazdu wcześniej niż $\boldsymbol{a_i}$ oznacza oczekiwanie na obsługę

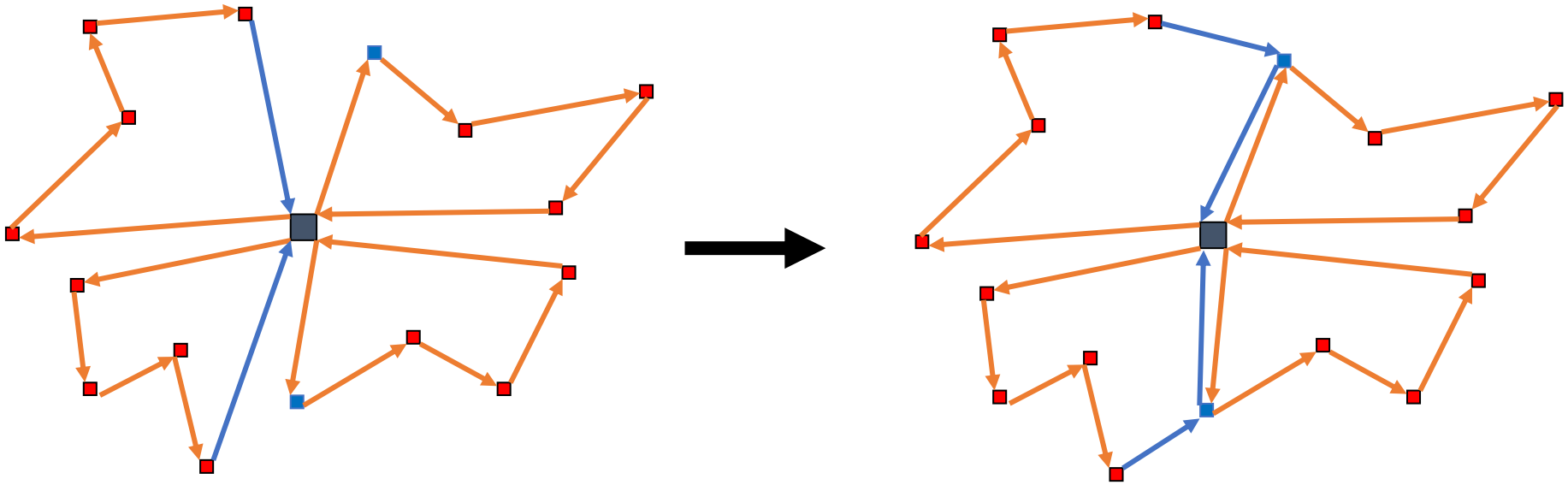
3. przybycie pojazdu później niż $\boldsymbol{b_i}$ oznacza

   - rozwiązanie niedopuszczalne (*hard windows*)
   - rozwiązanie dopuszczalne obarczone karą (*soft windows*)

4. Time Dependent Vehicle Routing Problem - TDVRP

SDVRP (Split Delivery *Vehicle Routing Problem*) – obsługa rozdzielona

1. każdy punkt tras może być odwiedzony przez więcej niż jeden pojazd
2. możliwość lepszego wykorzystania ładowności pojazdów, w szczególności gdy wielkość dostawy do punktu stanowi istotną jej część



VRPPD (*Vehicle Routing Problem with Pick-Up and Delivering*)
VRPB (*Vehicle Routing Problem with Backhauls*) – dostawy i/lub odbiór

1. każdy punkt (obok bazy) może być zarówno dostawcą oraz odbiorcą „*wielu – do – wielu*”

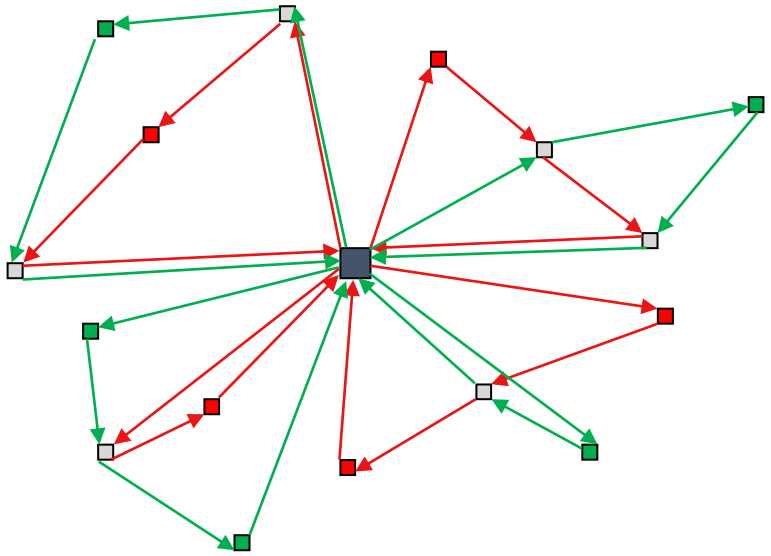
2. każdy punkt (obok bazy) może być zarówno dostawcą oraz odbiorcą, ale każdy punkt odbioru ma tylko jednego skojarzonego z nim dostawcę „ *jeden – do – jeden*”

3. dostawcą i odbiorcą jest baza „*jeden – do – wielu – do – jeden*”

   - do każdego punktu tylko dostarczany jest produkt lub tylko od niego odbierany
   - do każdego punktu zarówno dostarczany jest produkt, jak i od niego odbierany

   - kolejność odwiedzin punktów przez pojazd nie ma znaczenia
   - najpierw dostarczany jest produkt do punktów, a dopiero potem realizowane są odbiory

PVRP (Periodic *Vehicle Routing Problem*) – obsługa okresowa



1. ustalony okres planowania ***T*** (np. tydzień) jako zbiór kolejnych dni
2. ustalona częstotliwość f_i odwiedzania każdego punktu
3. Zbiór ***H*** alternatywnych harmonogramów odwiedzania każdego punktu

T	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}
1	1	1	1	1						
2	1				1	1	1			
3		1			1			1	1	
4			1			1		1		1
5				1			1		1	1

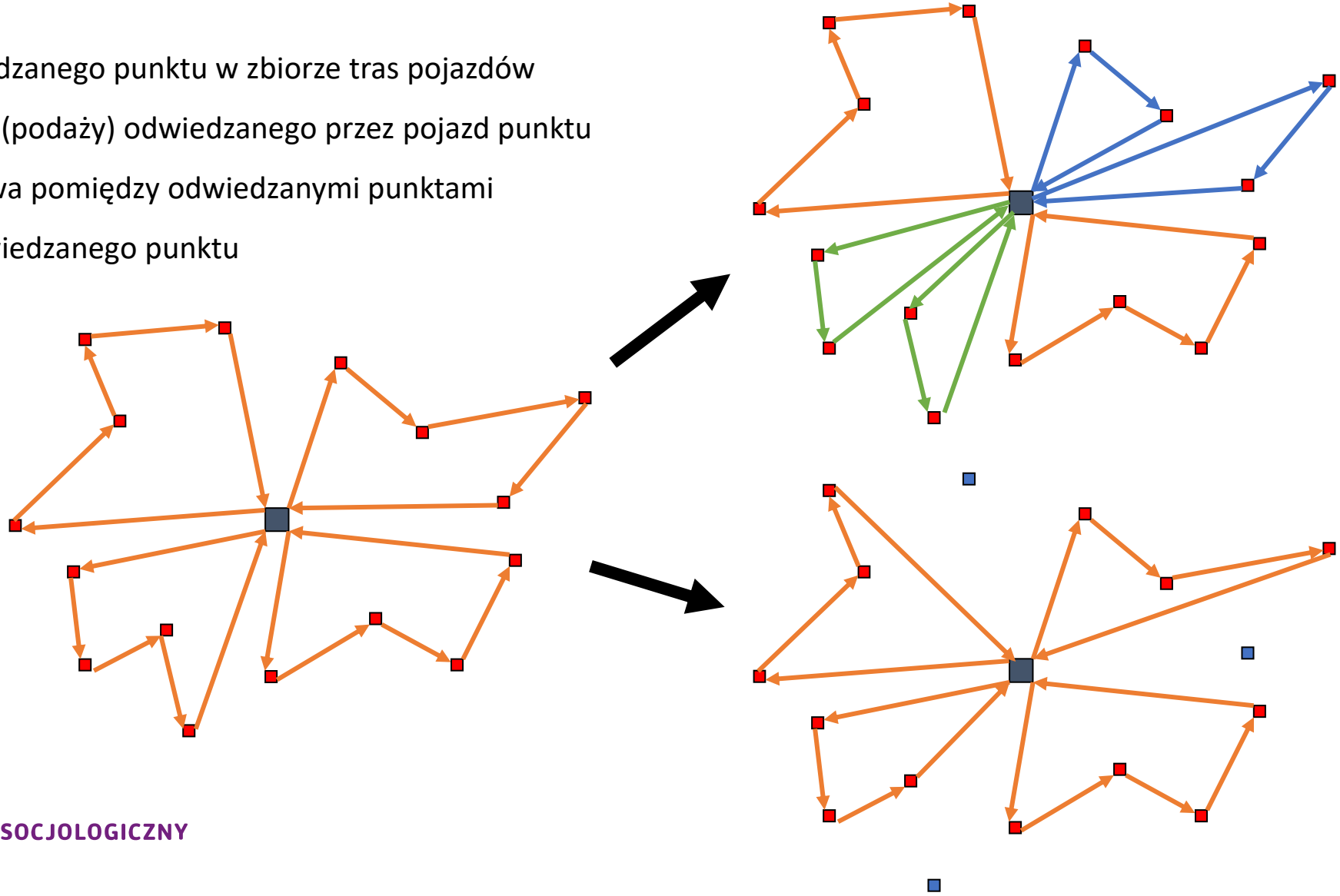
T = 5

f_i = 2

- podział zbioru wszystkich punktów na rejony dla poszczególnych pojazdów
- konstrukcja trasy pojazdu w ramach rejonu
- wybór dopuszczalnego harmonogramu odwiedzania punktu

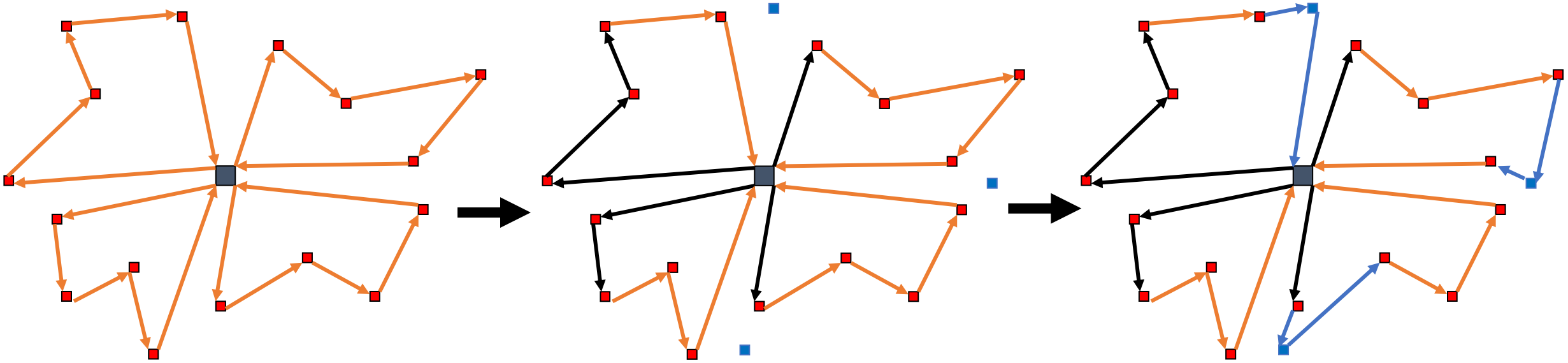
SVRP (*Stochastic Vehicle Routing Problem*) – losowość parametrów problemu

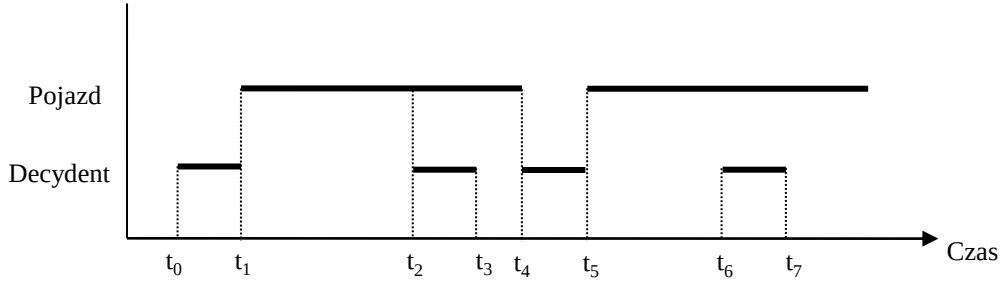
1. obecność odwiedzanego punktu w zbiorze tras pojazdów
2. wielkość popytu (podaży) odwiedzanego przez pojazd punktu
3. odległość czasowa pomiędzy odwiedzanymi punktami
4. czas obsługi odwiedzanego punktu



DVRP (*Dynamic Vehicle Routing Problem*) – zmienność problemu w czasie

1. dane wejściowe do problemu są zmienne i zależne od czasu (ciągła aktualizacja)
2. rozwiązanie (układ tras pojazdów) musi być dostosowywany na bieżąco (korekty tras)
3. charakter danych nie zmienia się
4. znany jest sposób zmiany danych (możliwość tworzenia scenariuszy zmian bez ponownej optymalizacji)

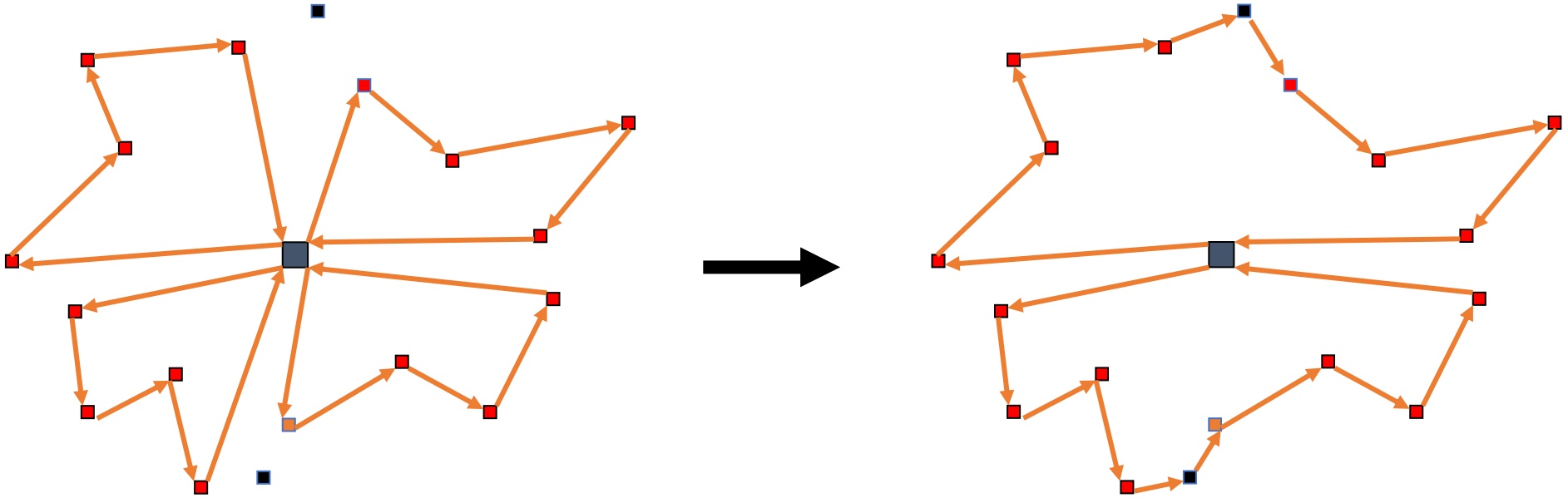




WYDZIAŁ
EKONOMICZNO-SOCJOLOGICZNY
Uniwersytet Łódzki

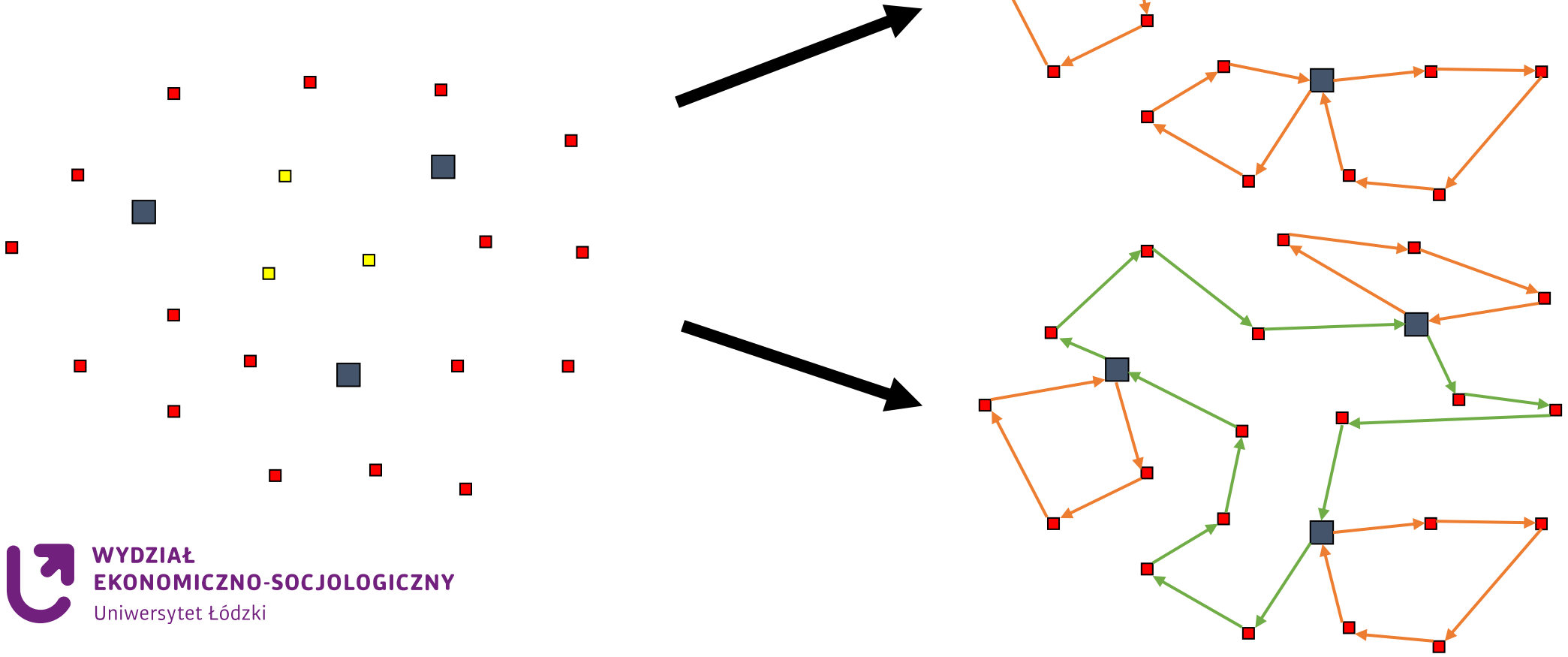
VRPSF (*Vehicle Routing Problem with Satellite Facilities*) – uzupełnienia ładunku

1. możliwość uzupełnienia ładunku przez pojazd w wyznaczonych lokalizacjach
2. trasa pojazdu zaczyna i kończy się w bazie



MDVRP (*Multi-Depot Vehicle Routing Problem*) – wiele baz

1. pojazd rozpoczyna i kończy trasę w tej samej bazie
2. pojazd może zakończyć trasę w innej bazie
3. rejonizacja baz – rejonizacja pojazdów – trasy pojazdów



WYDZIAŁ
EKONOMICZNO-SOCJOLOGICZNY
Uniwersytet Łódzki

MVRP (*Multiobjective Vehicle Routing Problem*) – wiele kryteriów oceny tras

1. przykłady kryteriów oceny rozwiązań:

   - łączna długość (czas, koszt) trasy pojazdów
   - liczba użytych pojazdów
   - równomierność pracy pojazdów
   - maksymalna długość trasy pojazdu
   - wykorzystanie ładowności pojazdu
   - czas oczekiwania pojazdów na obsługę (*VRPTW*)
   - czas opóźnień pojazdów (*VRPTW*)
   - przyporządkowanie pojazdu do tego samego klienta (*PVRP*)

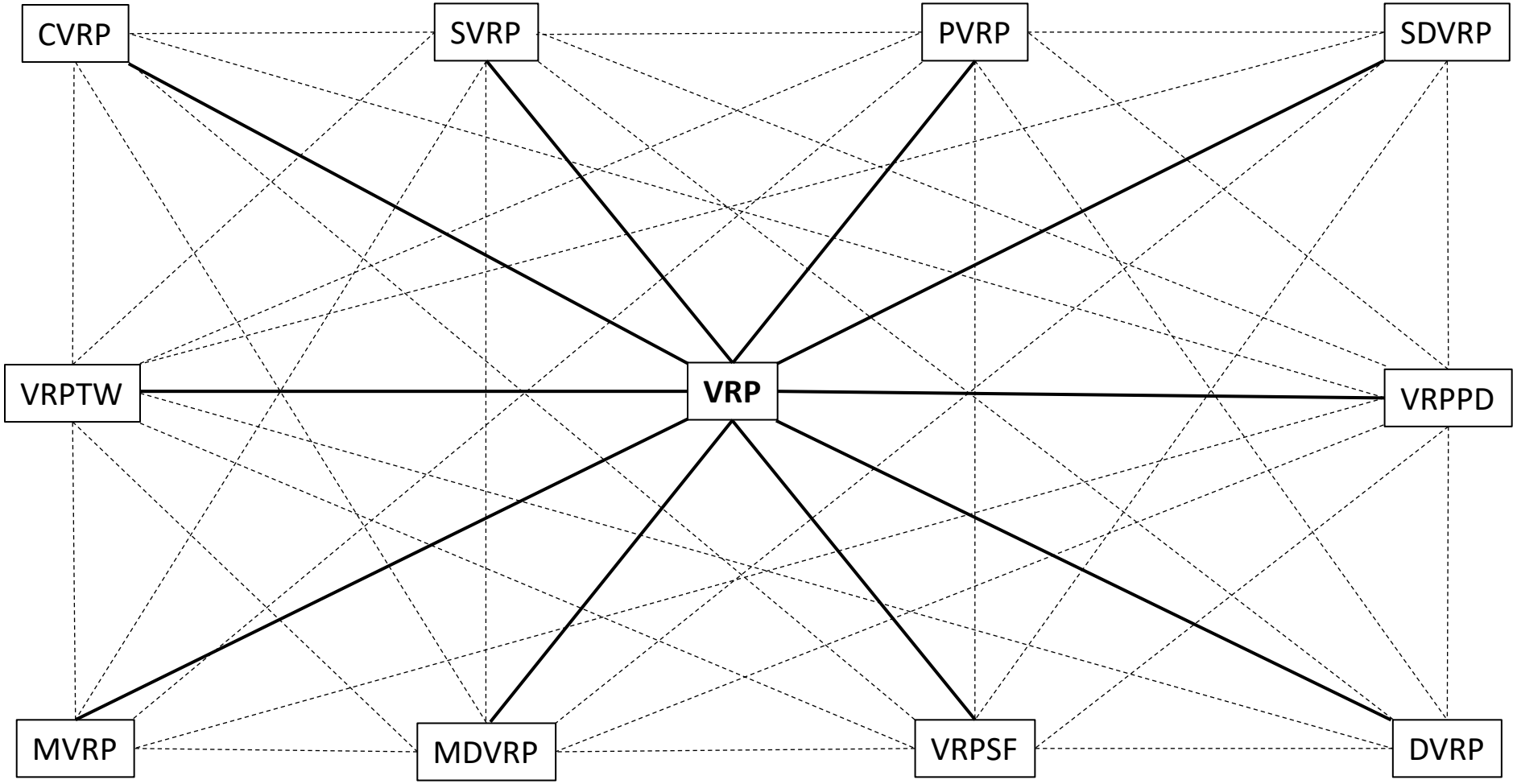
MVRP (*Multiobjective Vehicle Routing Problem*) – wiele kryteriów oceny tras

2. Preferencje:

   - bez określenia preferencji
   - a’priori
   - a’posteriori

   - jedna funkcja jako suma ważona
     - ✓ konwersja do tej samej jednostki (np. ujęcie wartościowe)
     - ✓ skalaryzacja (konwersja na wielkości bez jednostek)
   - wybrane cele jako ograniczenia

Hybrydyzacja problemów VRP



Wybrane zastosowania modeli VRP

1. Usługi kurierskie (Tebaldi i in. 2020) – CVRPTW
   - kilkudziesięciu klientów, w tym obsługa większości części z nich w godz. 10-18, a części w godz. 10-13
   - ograniczenia dotyczące ładowności pojazdów i czasu ich pracy

2. Dystrybucja (Kramera i in. 2019) – MDCVRPPTW (RVRP – *Rich* VRP)
   - dostawy produktów farmaceutycznych do szpitali i placówek opieki zdrowotnych (kilkaset punktów odbioru)
   - problem wielobazowy (magazyny główne i pomocnicze)
   - niejednorodna flota pojazdów
   - okna czasowe
   - dostawy okresowe (na ustalony okres 6 dni)
   - ograniczenia dotyczące możliwości wykorzystania pojazdu u danego odbiorcy
   - maksymalny czas trwania trasy i liczby klientów w trasie

Wybrane zastosowania modeli VRP

3. Organizacja domowej opieki zdrowotnej (Moussavi i in. 2019) – PVRP
   - liczba pacjentów: 10-30
   - liczba pracowników: 4-10
   - liczba planowanych dni: 2-14, dzień podzielony na okresy godzinne (czas trwania wizyty 1 h)
   - minimalizacja łącznej długości wszystkich tras pracowników w całym okresie planowania
   - minimalizacja najdłuższej trasy dziennej pracownika
   - minimalizacja najdłuższej trasy pracownika w całym okresie planowania

4. Zarządzanie pracą patroli policyjnych (Saint-Guillain 2021) – SVRP
   - budowa tras patroli dla wyznaczonych wcześniej miejsc „oczekiwania” na zgłoszenie
   - obsługa przez patrol pojawiających się losowo zdarzeń
   - minimalizacja oczekiwanego czasu odpowiedzi na zgłoszenia (od momentu pojawienia się zgłoszenia w systemie do przybycia patrolu na miejsce)

Wybrane zastosowania modeli VRP

5. Usługi komunalne (odbiór odpadów) (Angelleli i Speranza 2002) – SFPVRP
   - więcej niż 1 punkt odbioru odpadów (nie są bazami)
   - możliwość kontynuowania trasy po odwiezieniu odpadów do punktu odbioru
   - ograniczenia dotyczące czasu trwania trasy i ładowności pojazdu
   - ilość odbieranych śmieci jako zmienna losowa (Markovic 2019)

6. Usługi komunalne (odśnieżanie ulic) (Perier i in 2007) – CVRPTW
   - miasto przedstawione jako graf 462 węzłów i 1234 krawędzi reprezentujących ulice (3 kategorie ulic)
   - trzy typy pojazdów o różnych wydajnościach pracy (różna prędkość, ładowność)
   - dodatkowe ograniczenia dotyczące m.in. priorytetów odśnieżanych ulic, ulic jednokierunkowych (brak możliwości skrętów)

Wybrane zastosowania modeli VRP

7. Usługi serwisowe (Tricoire 2007) – PVRPTW
   - sieć dystrybucji i uzdatniania wody
   - określone terminy obsługi punktu (część musi być wykonana w danym dniu, część ma ustalony okres ważności dni)
   - niektóre żądania obsługi mogą być odroczone (dodatkowa kara);
   - pojazdy nie muszą rozpoczynać tras z bazy – punkty początkowe mogą być inne związane z danym technikiem;
   - minimalizacja odległości wszystkich tras techników;
   - minimalizacja liczby zgłoszeń, które nie mogą być wykonane w określonym terminie

8. Planowanie pracy dronów (Cannioto, 2013) – CVRP
   - ocena stanu technicznego obiektów uszkodzonych w wyniku trzęsienia ziemi
   - ograniczony zasięg dronów
   - minimalizacja łącznej długości ścieżek dronów
   - określenie najlepszej lokalizacji startu dronów

Wybrane zastosowania modeli VRP

1. logistyka dystrybucji oraz zaopatrzenia
2. logistyka zwrotna (np. zużytych produktów)
3. przewozy pocztowe i kurierskie
4. transport publiczny
5. transport wewnętrzny (np. magazynie)
6. planowanie ruchu robotów w procesie produkcyjnym
7. służba zdrowia
8. usługi serwisowe
9. planowanie tras w zwalczaniu pasożytów

Algorytmy heurystyczne układania tras pojazdów

- konstrukcyjne

- wieloetapowe (dla wielu pojazdów)
  - rejony $\rightarrow$ trasy
  - trasa $\rightarrow$ rejony

- poprawy (wzrostu)
  - jednoagentowe
  - wieloagentowe

Algorytmy konstrukcyjne

krok 1: inicjalizacja

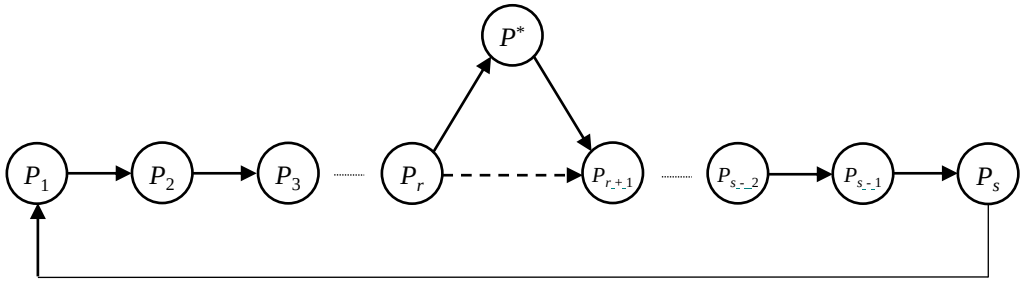
repeat

krok 2: wybór punktu obsługi

krok 3: wstawienie punktu do trasy

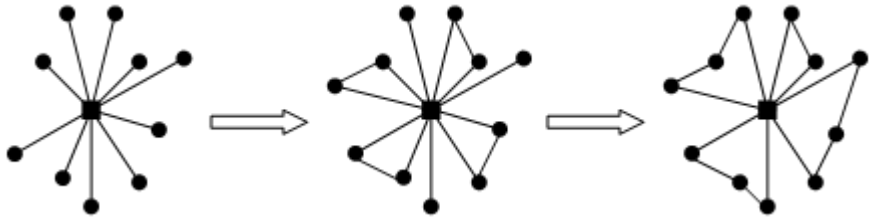
until trasa jest dopuszczalna

- algorytmy wstawiania (drogi do najbliższego sąsiada, sukcesywne dołączanie)

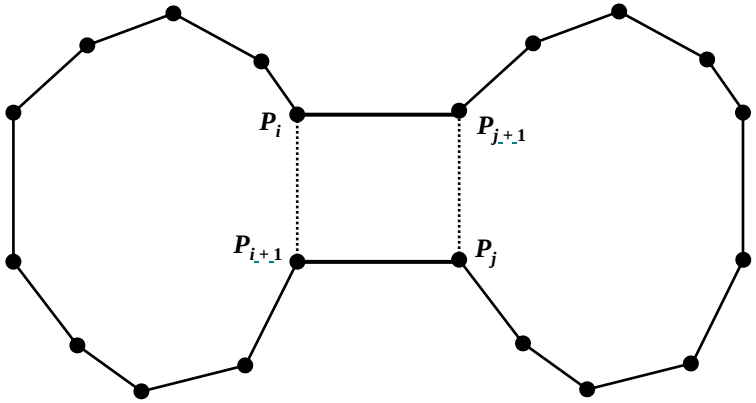


Algorytmy konstrukcyjne

- algorytm oszczędnościowego łączenia tras (*savings*)

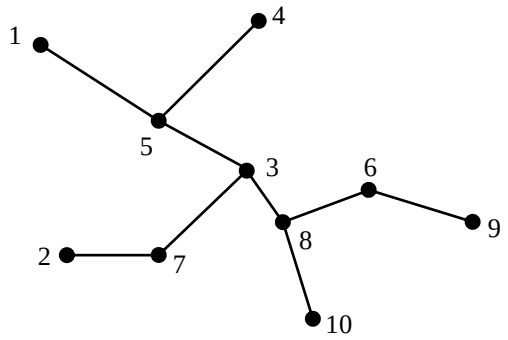


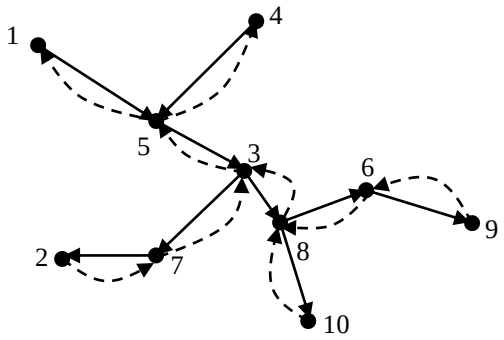
- problem przydziału (łączenie podcykli)

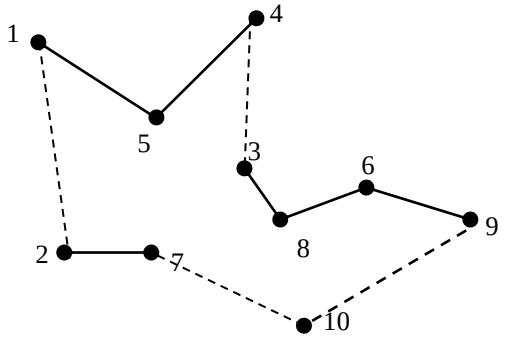


Algorytmy konstrukcyjne

- minimalne drzewo rozpinające







Algorytmy dekompozycyjne

krok 1: inicjalizacja algorytmu

repeat

krok 2: inicjalizacja trasy

repeat

krok 3: przyporządkowanie punktu do pojazdu

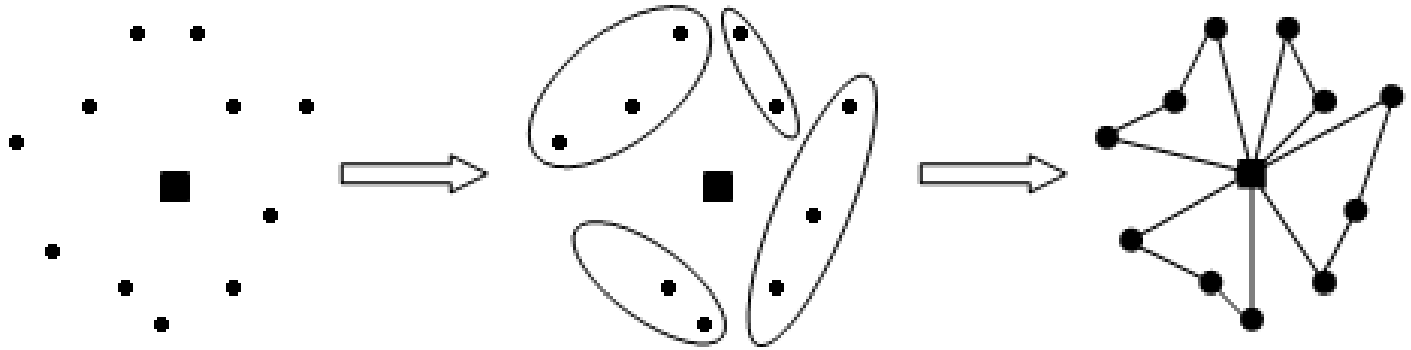
until trasa jest dopuszczalna

until wszystkie punkty obsługi są umieszczone w trasach

repeat

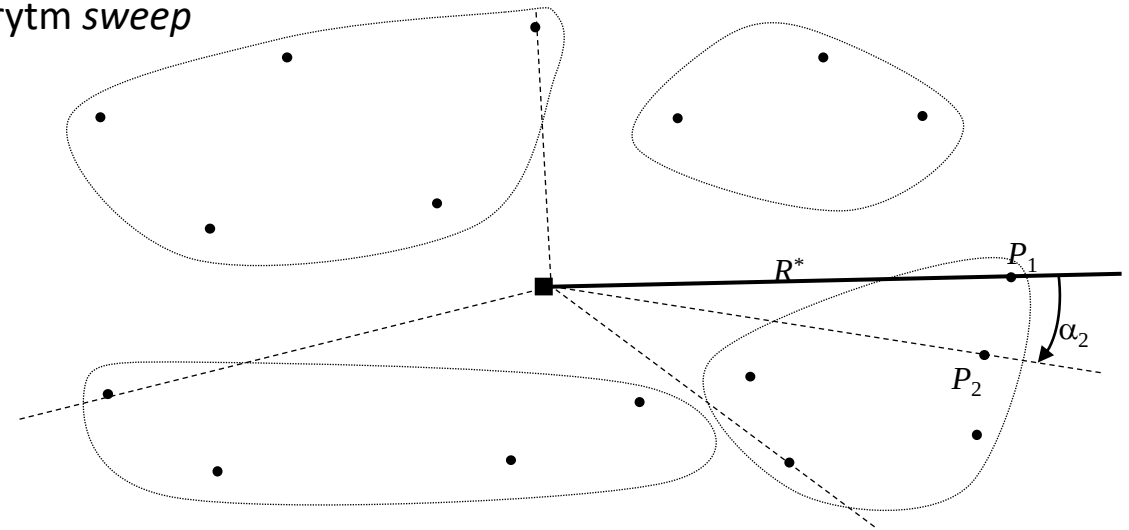
krok 4: rozwiązanie problemu komiwojażera

until wszystkie rejony mają ustaloną trasę

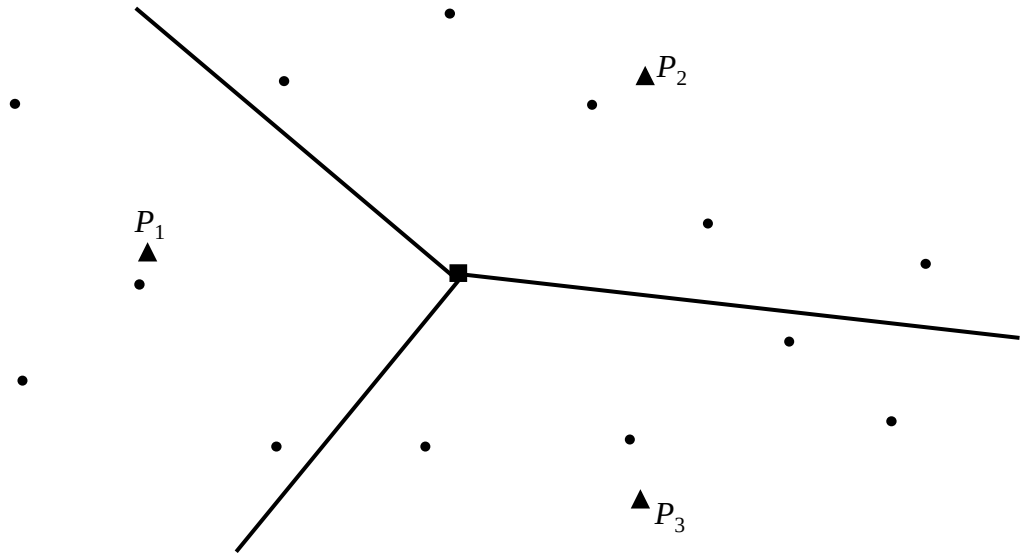


Algorytmy dekompozycyjne

- algorytm *sweep*

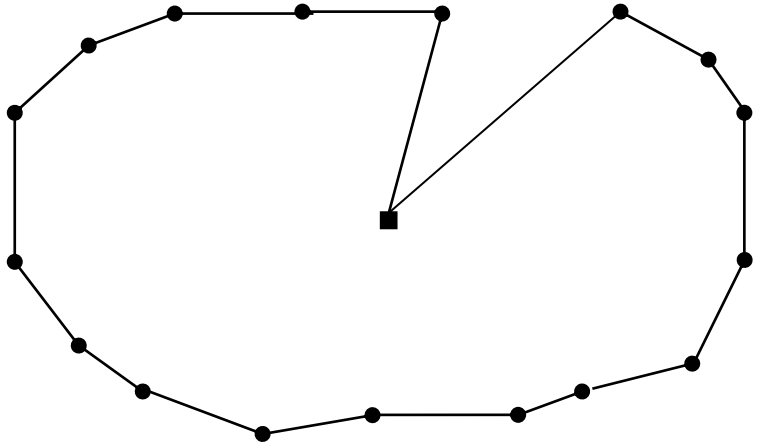


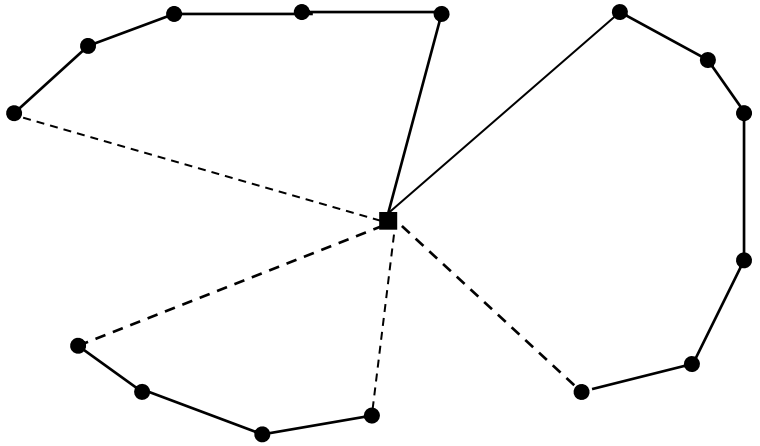
- algorytm baz fikcyjnych



Algorytmy dekompozycyjne

- algorytm *„supertrasy”* komiwojażera





Przeszukiwanie lokalne

I. Ustalić początkowy zbiór tras ***T*** i określić ***D***(*T*)

II. Przyjąć *T* jako najlepsze dotychczas znalezione rozwiązanie ***T****

repeat

III. Wygenerować rozwiązanie / zbiór rozwiązań sąsiednich ***N*(*T*)**

IV. Znaleźć lepsze / najlepsze rozwiązanie ***T'*** ze zbioru *N*(*T*)

V. Jeżeli wartość *D*(*T'*) < *D*(*T**) przyjąć *T** = *T'*

until brak poprawy

Przeszukiwanie lokalne

- algorytmy przeszukiwania lokalnego najpopularniejszej klasy heurystyk

- punkt początkowy (startu)
  - najczęściej wybierany jest losowo
  - można wykorzystać wiedzę eksperta (jeżeli jest dostępna)

- przeszukiwanie lokalne może mieć charakter:
  - dyskretny (sąsiedztwo jest niewielkie – można przejrzeć całe)
  - zrandomizowany (sąsiedztwo jest duże – w celu wybrania nowego punktu rozpatrywany jest losowo jego fragment)

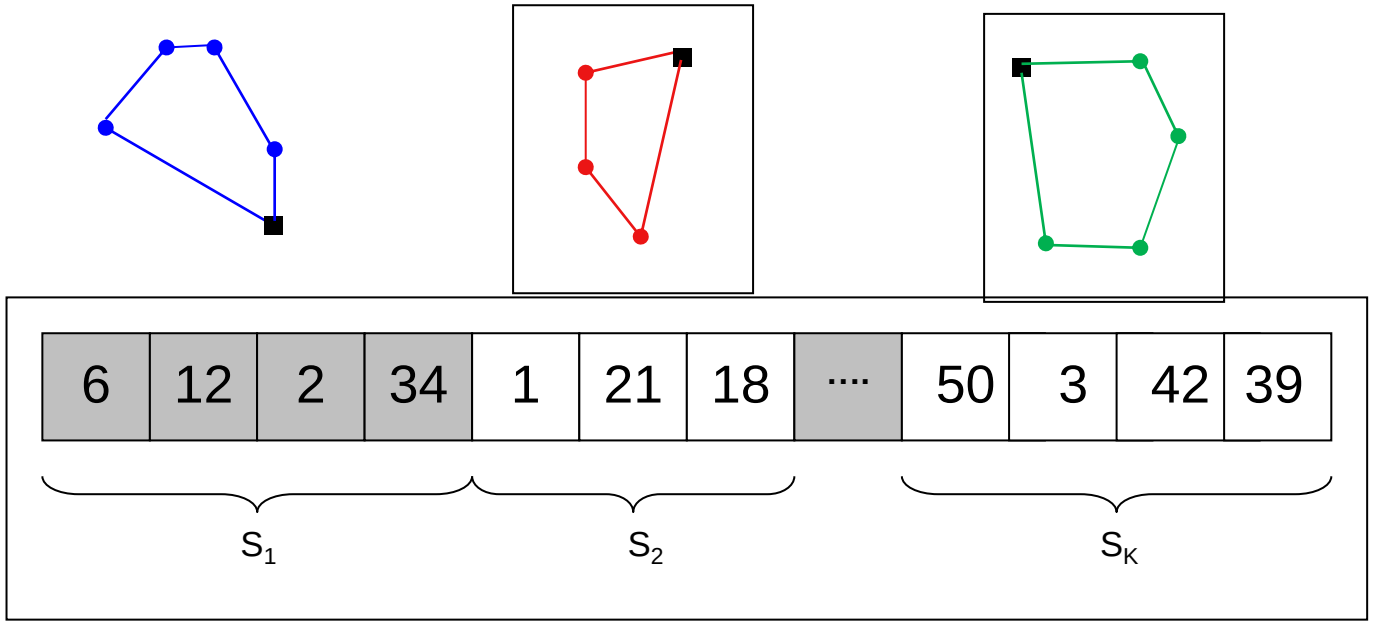
Przeszukiwanie lokalne

- zalety przeszukiwania lokalnego:
  - elastyczność
  - znajdują zastosowania w każdym problemie należącym do klasy zagadnień optymalizacji kombinatorycznej
  - są skuteczne dla zadań, których funkcja celu jest jednomodalna (posiada jedno optimum w przeszukiwanej przestrzeni rozwiązań)

- wady przeszukiwania lokalnego:
  - najczęściej znajdują najlepsze rozwiązanie, które jest optimum lokalnym (w przypadku zadań z wielomodalną funkcją celu)
  - jakość uzyskanego rozwiązania uzależniona jest od wyboru punktu startowego

- strategie eliminacji wad przeszukiwania lokalnego :
  - wielostartu – wielokrotne uruchamianie algorytmu rozpoczynając za każdym razem od innego losowo wybranego punktu przestrzeni rozwiązań
  - wielostartu ze zmodyfikowanego bieżącego położenia – wielokrotne uruchamianie algorytmu rozpoczynając za każdym razem od punktu przestrzeni rozwiązań, którego współrzędne powstają poprzez modyfikację współrzędnych punktu ostatniego położenia (unikanie lokalnych optimów)

Przeszukiwanie lokalne – reprezentacja rozwiązania

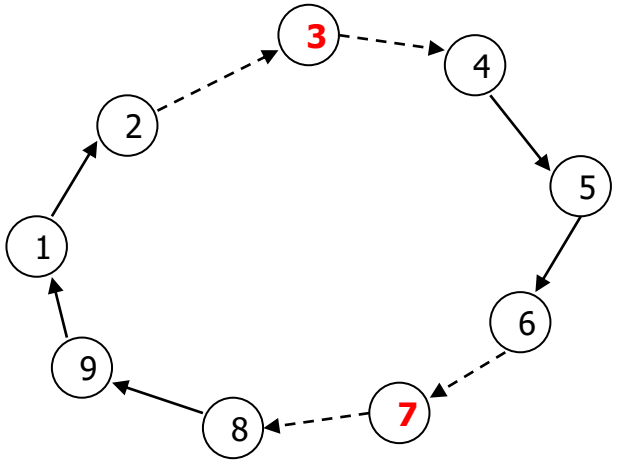


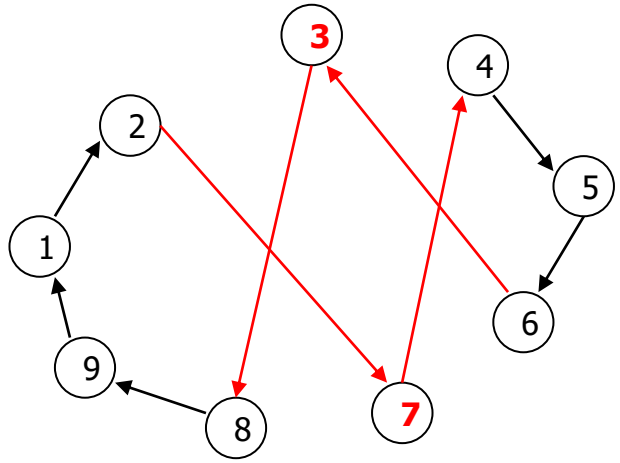
Przeszukiwanie lokalne - sąsiedztwo

1. NS1: zamiana dwóch punktów miejscami

   T = [1, 2, **3**, 4, 5, 6, **7**, 8, 9]

   T' = [1, 2, **7**, 4, 5, 6, **3**, 8, 9]



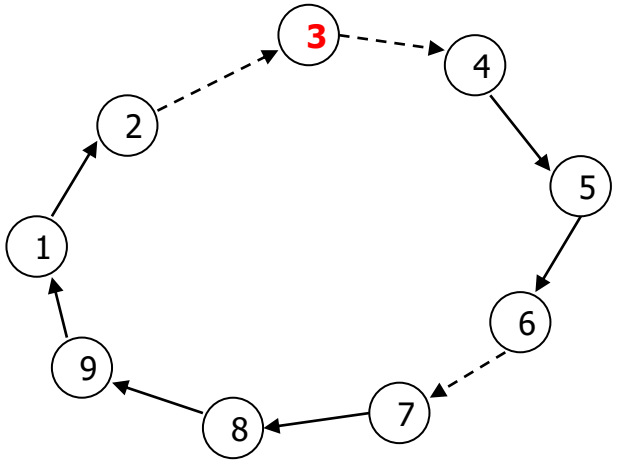


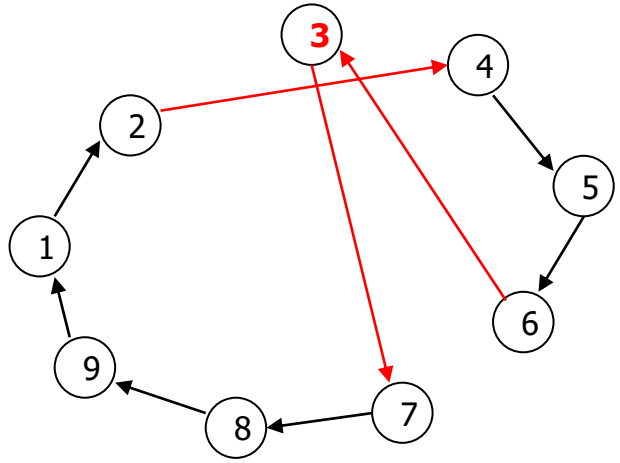
Przeszukiwanie lokalne - sąsiedztwo

2. NS2: zmiana pozycji punktu (3-opt)

T = [1, 2, **3**, 4, 5, 6, 7, 8, 9]

T' = [1, 2, 4, 5, 6, **3**, 7, 8, 9]





Przeszukiwanie lokalne - sąsiedztwo

3. NS3: zamiana trzech punktów miejscami

T = [1, 2, **3**, 4, **5**, 6, **7**, 8, 9]

T_1' = [1, 2, **7**, 4, **3**, 6, **5**, 8, 9]

T_2' = [1, 2, **5**, 4, **7**, 6, **3**, 8, 9]

4. NS4: zamiana dwóch lub trzech punktów miejscami

T = [1, 2, **3**, 4, **5**, 6, **7**, 8, 9]

T_1' = [1, 2, **7**, 4, **3**, 6, **5**, 8, 9]

T_2' = [1, 2, **5**, 4, **7**, 6, **3**, 8, 9]

T_3' = [1, 2, **5**, 4, **3**, 6, 3, 8, 9]

T_4' = [1, 2, 3, 4, **7**, 6, **5**, 8, 9]

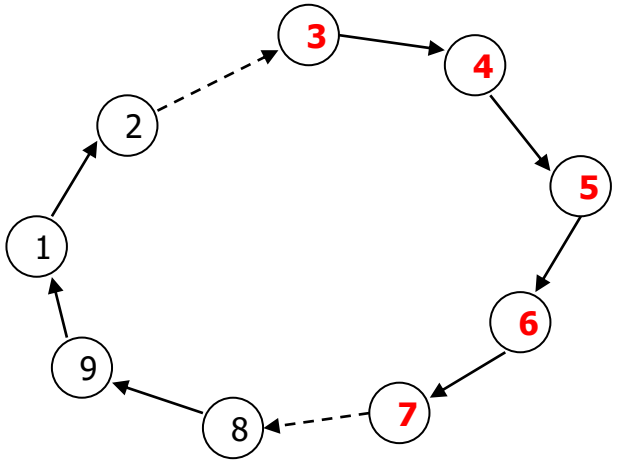
T_5' = [1, 2, **7**, 4, 5, 6, **3**, 8, 9]

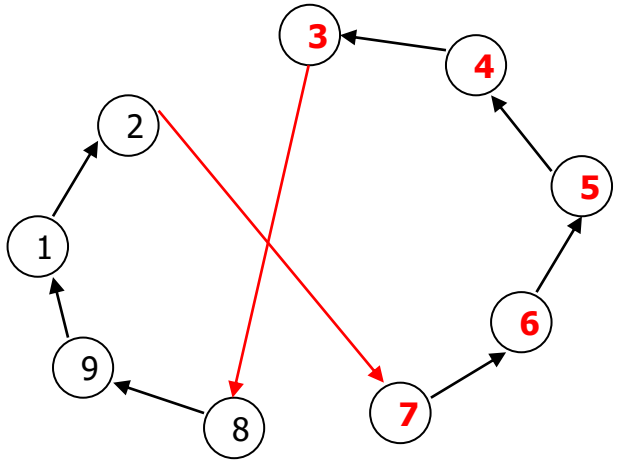
Przeszukiwanie lokalne - sąsiedztwo

5. NS5: wymiana dwóch połączeń (2-opt)

T = [1, 2, **3, 4, 5, 6, 7**, 8, 9]

T' = [1, 2, **7, 6, 5, 4, 3**, 8, 9]





Przeszukiwanie lokalne – *tabu*

I. Ustalić początkowy zbiór tras ***T*** i określić ***D***(*T*)

II. Przyjąć *T* jako najlepsze dotychczas znalezione rozwiązanie ***T****

repeat

III. Wygenerować rozwiązanie / zbiór rozwiązań sąsiednich ***N*(*T*)**

IV. Znaleźć lepsze / najlepsze rozwiązanie ***T'*** ze zbioru *N*(*T*) z zastosowaniem reguły ***tabu***

V. Jeżeli wartość *D*(*T'*) < *D*(*T**) przyjąć *T** = *T'*

VI. Uaktualnić pamięć ***tabu***

until brak poprawy

Przeszukiwanie lokalne – *tabu*

- algorytm TS jest strategią przeszukiwania lokalnego, wykorzystującą mechanizm ograniczeń nakładanych na zbiór rozwiązań sąsiednich

- celem nakładanych ograniczeń jest:
  - zminimalizowanie ryzyka zapętlenia się algorytmu
  - skierowanie procesu poszukiwań rozwiązania optymalnego w inne rejony przestrzeni rozwiązań

- ograniczenia nakładane na zbiór rozwiązań sąsiednich mogą mieć formę:
  - zakazu
  - restrykcji

Przeszukiwanie lokalne – *tabu*

- w przypadku rozwiązania objętego zakazem nie jest w ogóle możliwe przejście do niego z rozwiązania aktualnie rozpatrywanego

- w przypadku restrykcji, można przejść do rozwiązania sąsiedniego objętego tym ograniczeniem, jednak w przypadku spełnienia określonych warunków (np. w całym procesie działania algorytmu przyniesie to określoną korzyść)

- podstawowymi elementami algorytmu TS są:
  - bufor przechowujący najlepsze dotychczas znalezione rozwiązanie
  - struktura pamięci - gromadzone są informacje o przeszukiwanym obszarze przestrzeni rozwiązań (o zrealizowanych dotychczas przejściach)
  - operator przejścia do rozwiązania sąsiedniego
  - zbiór reguł ograniczających wybór rozwiązania sąsiedniego (reguły tabu) – budowany na podstawie: aktualności zapisanych w pamięci danych, częstotliwości zapisu w pamięci określonych danych, wpływu danych zgromadzonych w pamięci na jakość uzyskiwanych wyników

Przeszukiwanie lokalne – *tabu*

Załóżmy, że w pamięci przechowujemy 5 ostatnio wykonanych przejść polegających na zamianie dwóch punktów w trasie.

Trasę: $0 \rightarrow 3 \rightarrow 9 \rightarrow 5 \rightarrow 4 \rightarrow 1 \rightarrow 7 \rightarrow 2 \rightarrow 6 \rightarrow 8$

otrzymano w wyniku dokonanych przekształceń w kolejności odpowiednio:

5 iteracji wcześniej:	zamiana punktów 2 i 4
4 iteracje wcześniej:	zamiana punktów 1 i 3
3 iteracje wcześniej:	zamiana punktów 2 i 9
2 iteracje wcześniej:	zamiana punktów 8 i 6
1 iteracja wcześniej:	zamiana punktów 5 i 7

Przeszukiwanie lokalne – *tabu*

	1	2	3	4	5	6	7	8	9
1			4						
2				5					3
3	4								
4		5							
5							1		
6								2	
7					1				
8						2			
9		3							

$t_{tabu} = 5$

	1	2	3	4	5	6	7	8	9
1			4						
2				5					3
3									
4									
5							1		
6								2	
7									
8									
9									

Przeszukiwanie lokalne – *tabu*

Przyjmijmy, że dla aktualnie rozpatrywanego rozwiązania:

$0 \rightarrow 3 \rightarrow 9 \rightarrow 5 \rightarrow 4 \rightarrow 1 \rightarrow 7 \rightarrow 2 \rightarrow 6 \rightarrow 8$

wygenerowano rozwiązania sąsiednie i posortowano je wg korzyści zmian w wartości funkcji celu:

x'_1: zamiana punktów 3 i 1	**ΔF**: = -20	**tabu**
x'_2: zamiana punktów 2 i 6	**ΔF**: = -14	
x'_3: zamiana punktów 3 i 9	**ΔF**: = -10	
x'_4: zamiana punktów 6 i 8	**ΔF**: = -3	**tabu**
x'_5: zamiana punktów 4 i 2	**ΔF**: = +5	
x'_6: zamiana punktów 9 i 7	**ΔF**: = +10	

Przeszukiwanie lokalne – *tabu*

nowa trasa (nowe przejście) i modyfikacja struktury pamięci:

$0 \rightarrow 3 \rightarrow 9 \rightarrow 5 \rightarrow 4 \rightarrow 1 \rightarrow 7 \rightarrow \mathbf{6} \rightarrow \mathbf{2} \rightarrow 8$

	1	2	3	4	5	6	7	8	9
1			3						
2				4		**5**			2
3	3								
4		4							
5									
6		**5**						1	
7									
8						1			
9		2							

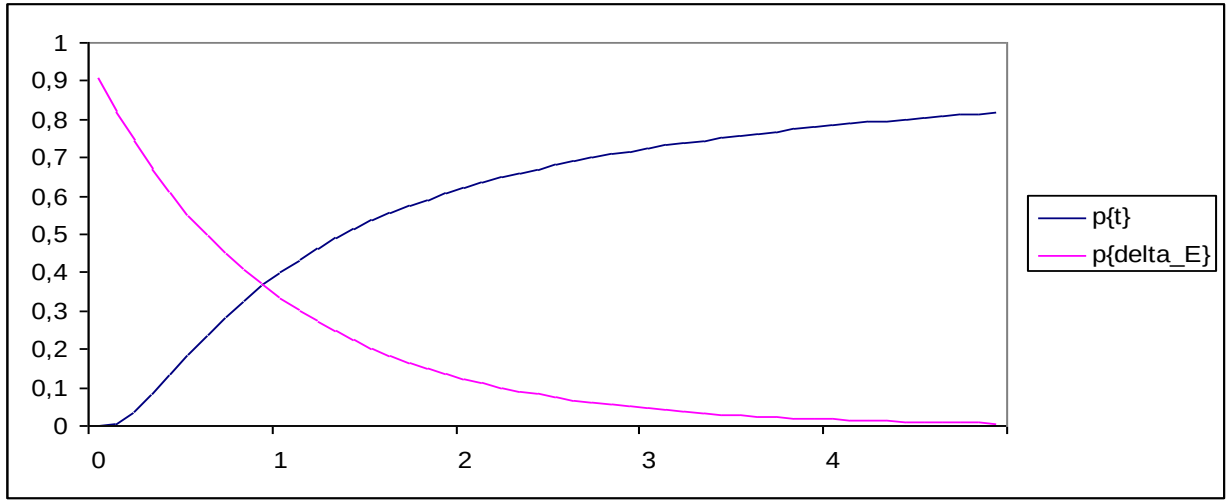
$t_{tabu} = 5$

	1	2	3	4	5	6	7	8	9
1			3						
2				4		**5**			2
3									
4									
5									
6								1	
7									
8									
9									

Przeszukiwanie lokalne – symulowane wyżarzanie

- zastosowanie prawa termodynamiki → przy ustalonej temperaturze, prawdopodobieństwo wzrostu energii cząsteczki o δE jest określone wzorem:

$$p(\delta E) = e^{-\delta E / kt}$$



Przeszukiwanie lokalne – symulowane wyżarzanie

I. Ustalić początkowy zbiór tras ***T*** i określić ***D***(*T*)

II. Przyjąć *T* jako najlepsze dotychczas znalezione rozwiązanie ***T****

III. Przyjąć ***Tmax***, ***Tmin***, ***LicznikPróbMax***, ustalić schemat chłodzenia (obniżania aktualnej temperatury ***Temp***)

repeat

IV. Wygenerować rozwiązanie / zbiór rozwiązań sąsiednich ***N*(*T*)**

V. Znaleźć lepsze / najlepsze rozwiązanie ***T'*** ze zbioru *N*(*T*) z zastosowaniem reguły ***tabu***

VI. Jeżeli wartość *D*(*T'*) < *D*(*T**) przyjąć *T** = *T'*

VII. Jeżeli wartość *D*(*T'*) ≥ *D*(*T**) przyjąć *T* = *T'* z prawdopodobieństwem $\boldsymbol{Prob} = e^{[D(T)-D(T')]/Temp}$

VIII. Uaktualnić ***LicznikPrób*** oraz jeżeli *LicznikPrób* > *LicznikPróbMax* obniżyć ***Temp***

until *Temp* < *Tmin*

Przeszukiwanie lokalne – symulowane wyżarzanie

- energia cząsteczki ***E*** zastąpiona jest funkcją celu
- wybór temperatury początkowej ***T*** oraz określenie funkcji spadku temperatury
- określenie wielkości limitu prób (generowanych rozwiązań sąsiednich przy aktualnej temperaturze *T*)

- obliczenie „przyrostu energii”

  $\delta E = f(x') - f(x)$ dla problemów minimalizacji

  $\delta E = f(x) - f(x')$ dla problemów maksymalizacji

- jeżeli $\delta E < 0$ to $x = x'$ oraz $x^* = x$ jeżeli x jest lepsze od x^*

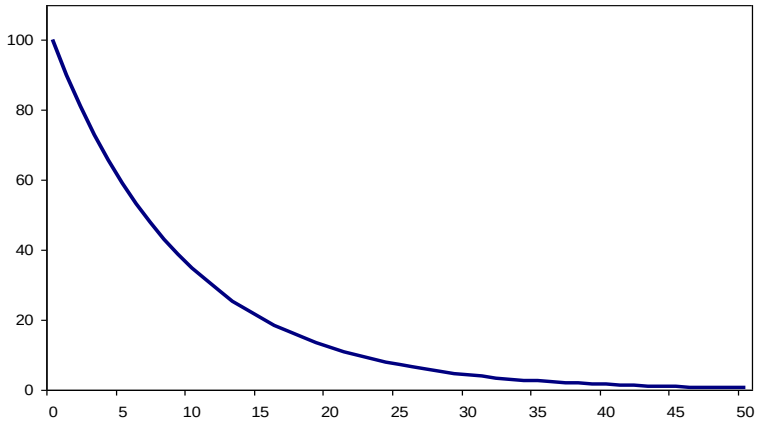
- jeżeli $\delta E \geq 0$ to $x = x'$ z prawdopodobieństwem $p = e^{-\delta E/t}$ oraz $x^* = x$ jeżeli x jest lepsze od x^*

Przeszukiwanie lokalne – symulowane wyżarzanie

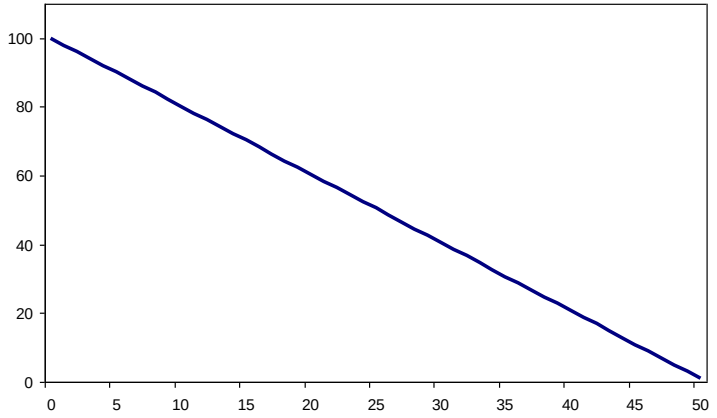
- parametrami SA są:
  - wartość górnego limitu temperatury T_0
  - wartość dolnego limitu temperatury T_N
  - formuła określania aktualnej wartości temperatury T_N – schemat chłodzenia
- prędkość spadku temperatury:
  - ma wpływ na szybkość działania algorytmu
  - zbyt powolny spadek temperatury oznacza dużą swobodę przeszukiwania przestrzeni rozwiązań, przy jednoczesnej niskiej zbieżności do określonego jej obszaru
  - zbyt szybki spadek temperatury oznacza znacznie ograniczoną swobodę w przeszukiwaniu przestrzeni rozwiązań, przy jednoczesnej szybkiej zbieżności do określonego jej obszaru (ryzyko utknięcia w optimum lokalnym)
- nowa wartość temperatury może być funkcją:
  - aktualnej wartości temperatury
  - górnego limitu temperatury
  - dolnego limitu temperatury
  - numeru kolejnej zmiany temperatury
  - maksymalnej liczby zmian temperatury

Przeszukiwanie lokalne – symulowane wyżarzanie

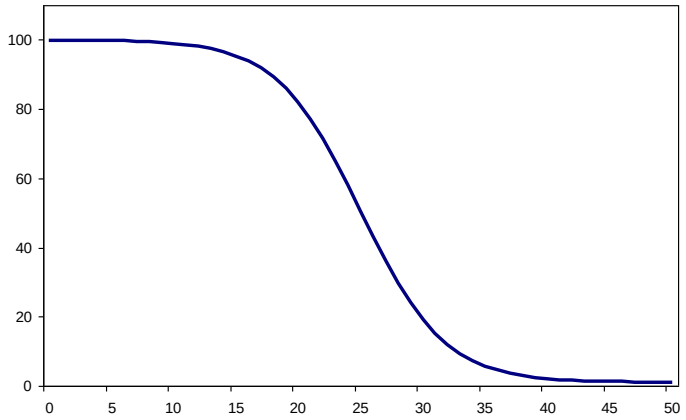
$T_i = aT_i$ gdzie a$\in$(0,8;0,99)



$$T_i = T_0 - i\frac{T_0 - T_N}{N}$$



$$T_i = \frac{T_0 - T_N}{1 + e^{0,3(i - N/2)}} + T_N$$



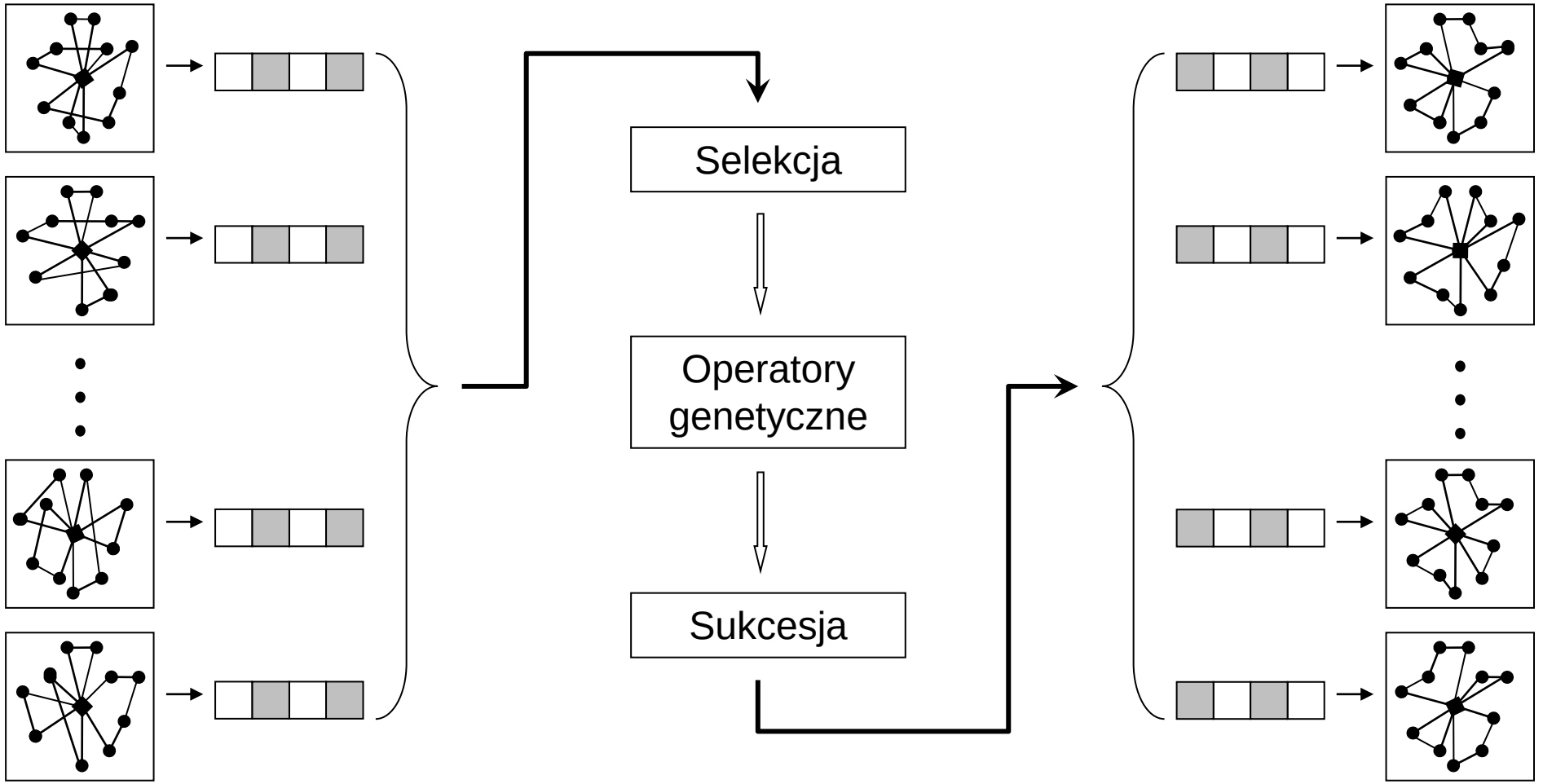
Algorytmy ewolucyjne

- reprezentacja rozwiązania problemu w postaci zakodowanej
- implementacja algorytmu symulującego proces ewolucji zachodzący na chromosomach żywych organizmów
- poszukiwanie „dobrych” chromosomów bez wiedzy o rodzaju optymalizowanego problemu
- niezbędna informacja to sposób oceny chromosomów – określenie „przystosowania”
- przetwarzanie parametrów zadania (potencjalnych rozwiązań) w postaci zakodowanej
- przeszukiwanie przestrzeni rozwiązań optymalizowanego problemu począwszy nie od jednego punktu, ale ich zbioru (populacja)
- najczęściej probabilistyczne reguły mające na celu znalezienie coraz to lepszych rozwiązań

Algorytmy ewolucyjne

- ***chromosom*** – łańcuch lub ciąg kodowy – uporządkowany ciąg genów
- ***gen*** – cecha, znak, pojedynczy element chromosomu
- ***osobnik*** – zakodowany w postaci jednego lub kilku chromosomów parametr zadania – rozwiązanie.
- ***populacja*** – zbiór osobników o określonej liczebności
- ***funkcja przystosowania*** – funkcja oceny osobnika (w szczególności jednego chromosomu) w populacji, pozwala na ocenę osobnika populacji względem innych i wskazać najlepszych (mających największe szanse na „przetrwanie”)

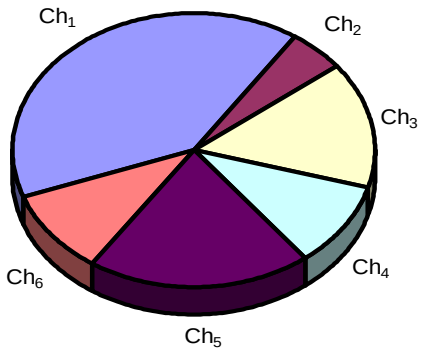
Algorytmy ewolucyjne



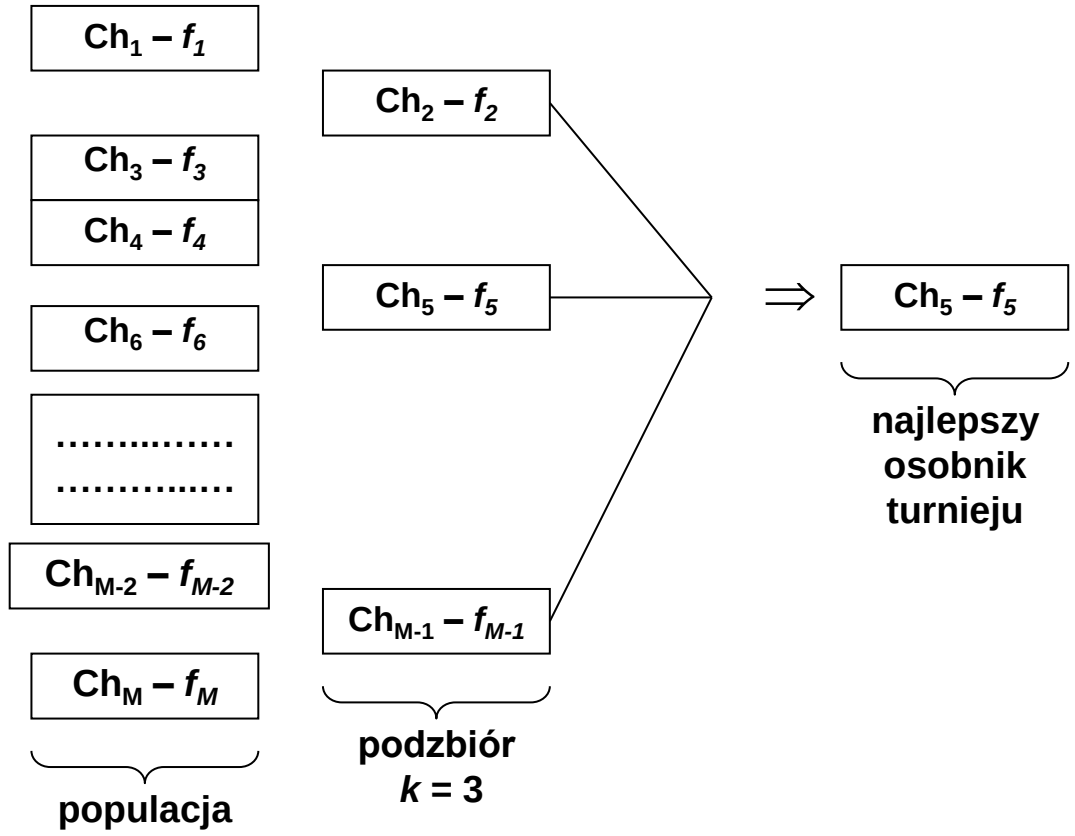
Algorytmy ewolucyjne – selekcja

- wskazanie zbioru osobników, które przeznaczone zostaną do reprodukcji

- selekcja ruletkowa



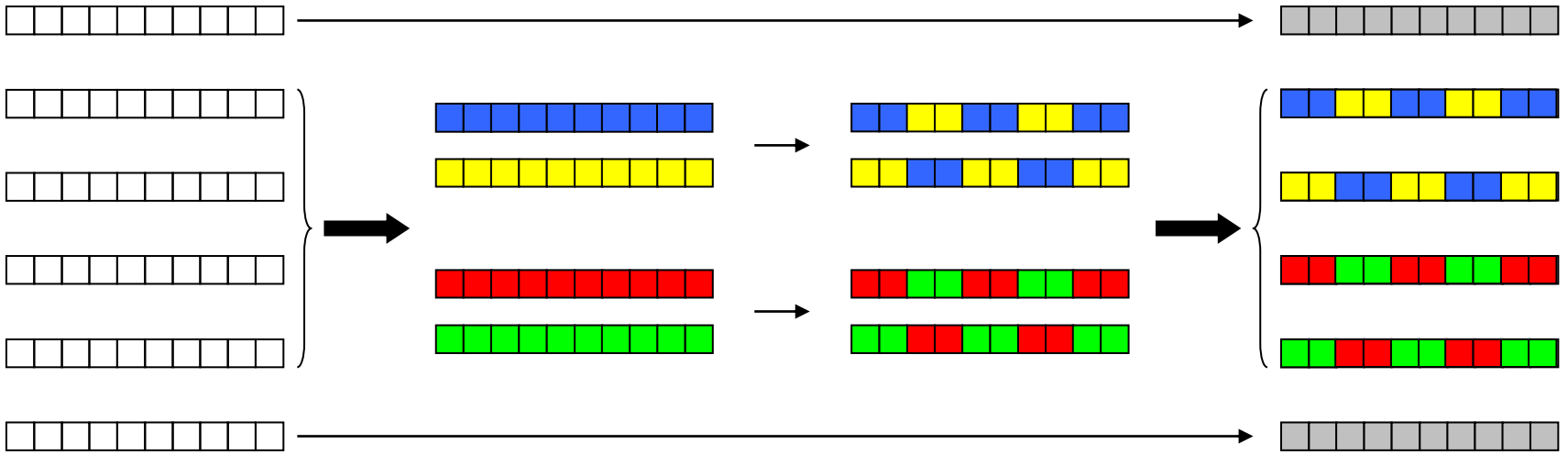
- selekcja turniejowa



WYDZIAŁ
EKONOMICZNO-SOCJOLOGICZNY
Uniwersytet Łódzki

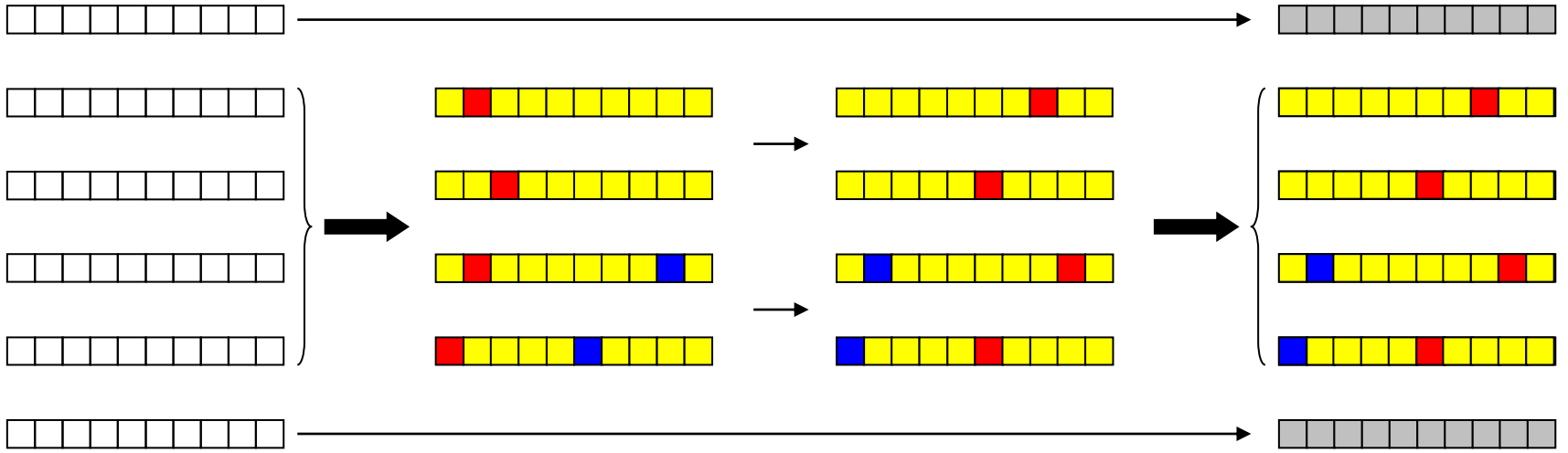
Algorytmy ewolucyjne – krzyżowanie

- krzyżowanie z częściowym odwzorowaniem (PMX)
- krzyżowanie z cykliczne (CX)
- krzyżowanie z porządkowe (OX)



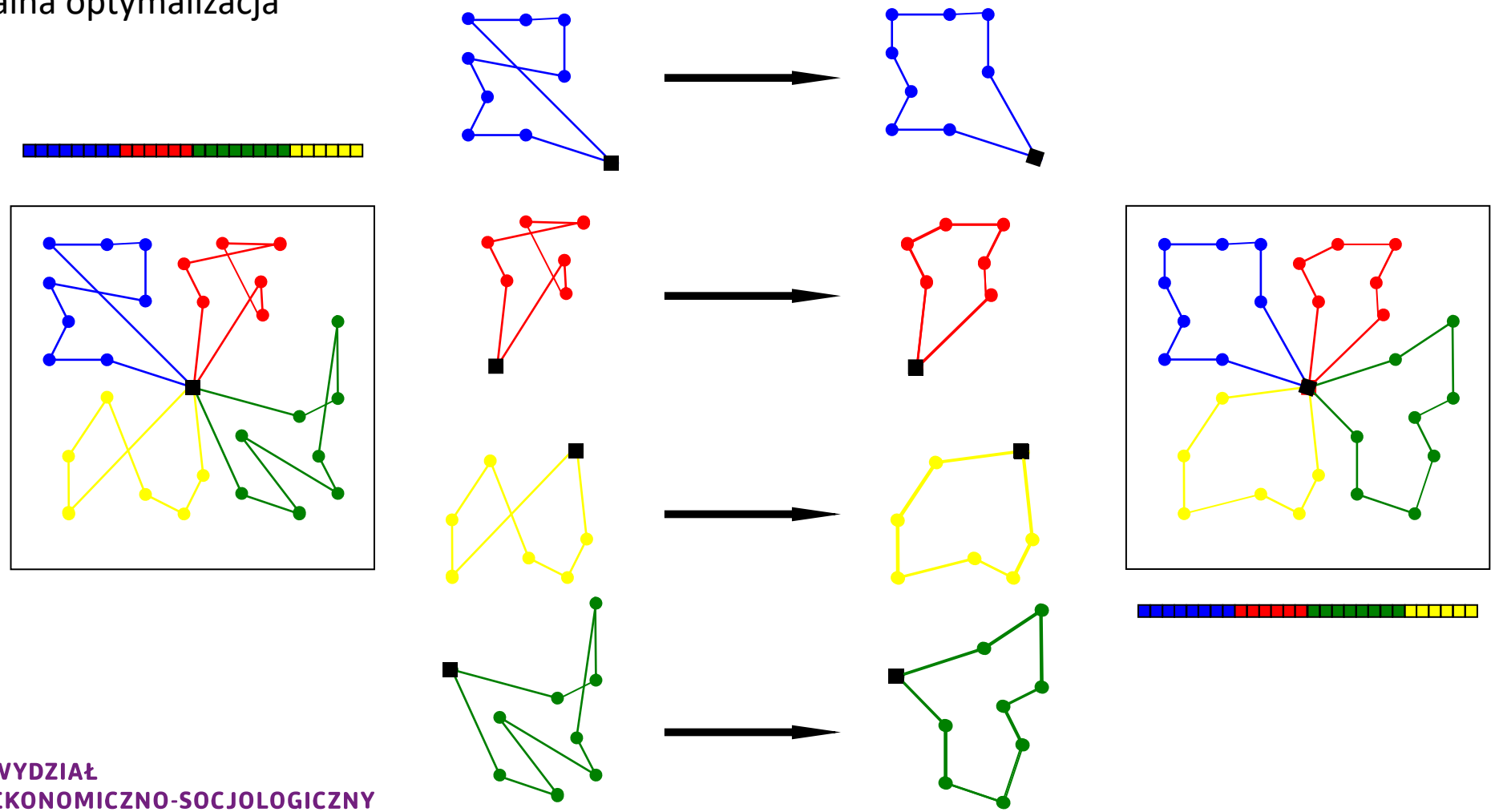
Algorytmy ewolucyjne – mutacja

- przemieszczenie genu
- wymiana pozycji genów
- inwersja



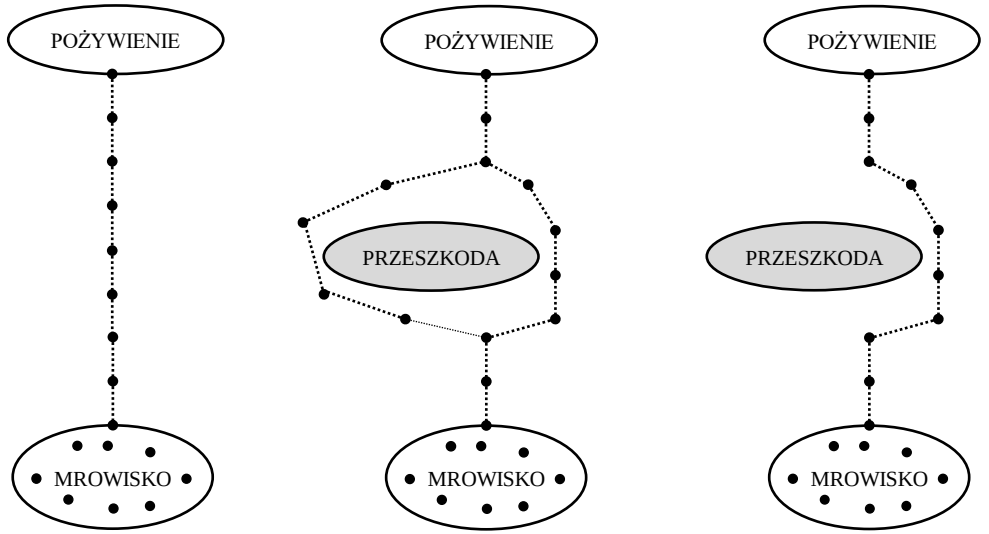
Algorytmy ewolucyjne – hybrydyzacja

- lokalna optymalizacja



Algorytmy mrówkowe

- podstawowym zadaniem mrowiska jest znalezienie pożywienia i jego transport do kolonii
- mrówki (tzw. agenci) to osobniki są zdolne tylko do bardzo prostych zachowań
- mrówki komunikują się pomiędzy sobą przy pomocy pozostawianej przez siebie substancji chemicznej – *feromonu*
- mrówki początkowo poruszają się losowo, aby następnie z biegiem czasu wybierać trasę najkrótszą (o największej intensywności feromonu)



Algorytmy mrówkowe

ustal początkową ilość feromonu na każdej krawędzi grafu

ustal liczebność kolonii mrówek i punkt początkowy dla każdej z nich

repeat

repeat

wybierz następny wierzchołek dla k-tej mrówki;

zaktualizuj ilość feromonu na połączeniu pomiędzy punktami (aktualizacja lokalna)

until wszystkie mrówki zbudują drogę w grafie

zaktualizuj ilość feromonu na trasie mrówki (aktualizacja globalna)

until warunek końca

Algorytmy mrówkowe

- aktualizacja lokalna feromonu

$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t)$, gdzie np. $\Delta\tau_{ij}(t) = \frac{\delta}{d_{ij}}$

d_{ij} – długość połączenia pomiędzy punktami, gdzie przeszła mrówka

- aktualizacja globalna feromonu

$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{K}\tau_{ij}^{k}(t)$, gdzie np. $\Delta\tau_{ij}(t) = \frac{\delta}{D_{ij}}$

D_{ij} – długość drogi którą przeszła mrówka, jeśli przeszła przez połączenie (i,j)

Algorytmy mrówkowe

- prawdopodobieństwo wyboru kolejnego punktu j w trasie

$$p_{ij}(t) = \frac{\tau_{ij}(t)^{\alpha}\eta_{ij}^{\beta}}{\sum_{u\in\Im_i}\tau_{ij}(t)^{\alpha}\eta_{ij}^{\beta}}$$

η_{ij} – „atrakcyjność" połączenia pomiędzy punktami ***i*** oraz ***j***, najczęściej: $1/d_{ij}$

α, β – wagi różnicujące wybór pomiędzy ilością feromonu a „atrakcyjnością" połączenia pomiędzy punktami

$\Im_i$ - zbiór punktów, które można jeszcze odwiedzić z punktu ***i***

Algorytmy mrówkowe

- liczba mrówek
- współczynnik parowania feromonu ρ, współczynnik poświaty feromonu $(1\text{-}\rho)$
- minimalna ilość feromonu na połączeniach pomiędzy punktami
- wagi α oraz β

Dziękuję za uwagę

Radosław Jadczak
radoslaw.jadczak@uni.lodz.pl

WYDZIAŁ
EKONOMICZNO-SOCJOLOGICZNY
Uniwersytet Łódzki