Algorytmy genetyczne
dla problemu komiwojażera
(ang. traveling salesperson)

Wprowadzenie

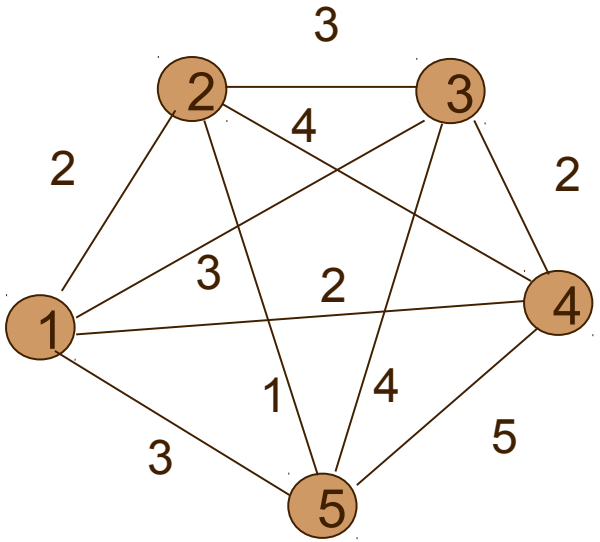
- Sztandarowy problem optymalizacji kombinatorycznej.
- Problem NP-trudny.
- Potrzeba poszukiwania heurystyk.
- Chętnie stosowaną heurystyką są algorytmy ewolucyjne (mnogość reprezentacji, operatorów etc.).

Przedstawienie problemu

- Dane jest n miast. Komiwojażer ma odwiedzić każde miasto dokładnie jeden raz wracając do punktu początkowego. Znamy koszty przejazdu pomiędzy każdą parą miast. Zaplanuj drogę komiwojażera tak, aby każde miasto zostało odwiedzone dokładnie jeden raz i całkowity koszt przejazdu był jak najmniejszy.
- Graf pełny G=(V,E), gdzie |V|=n, każdy element V to wierzchołek, a każdy element E to krawędź łącząca parę wierzchołków. W grafie pełnym każda para wierzchołków jest połączona.
- Macierz wag W=[w_{ij}]. Element w_{ij}>0 reprezentuje wagi (długości) krawędzi łączącej wierzchołki i oraz j.
- Droga to ciąg wierzchołków, przy czym sąsiednie wierzchołki muszą być połączone krawędziami. Droga jest cyklem jeżeli wierzchołek końcowy jest taki sam jak początkowy. Długość drogi jest sumą wag krawędzi łączących wierzchołki wchodzące w jej skład.
- Cykl Hamiltona to cykl w którym każdy wierzchołek jest odwiedzany dokładnie jeden raz.
- Problem komiwojażera możemy sformułować: ***Znaleźć cykl Hamiltona o minimalnej długości.***
- Macierz wag W jednoznacznie określa nam problem.

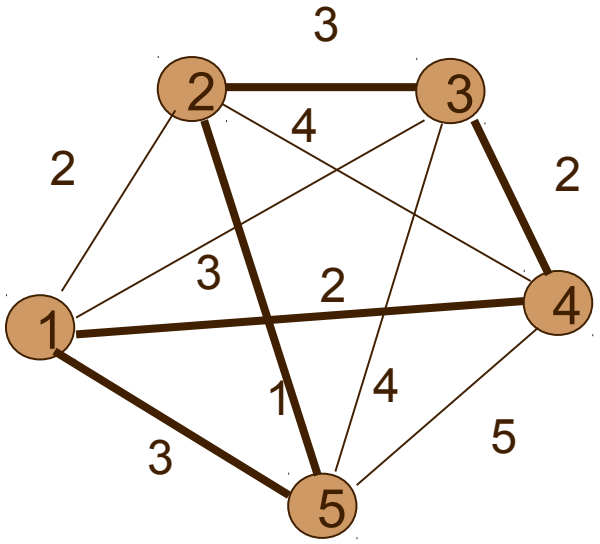
Przykład

- Przykładowy graf oraz macierz wag **W**: (na przekątnej wagi nieskończone, bo wierzchołek nie jest połączony krawędzią z samym sobą).

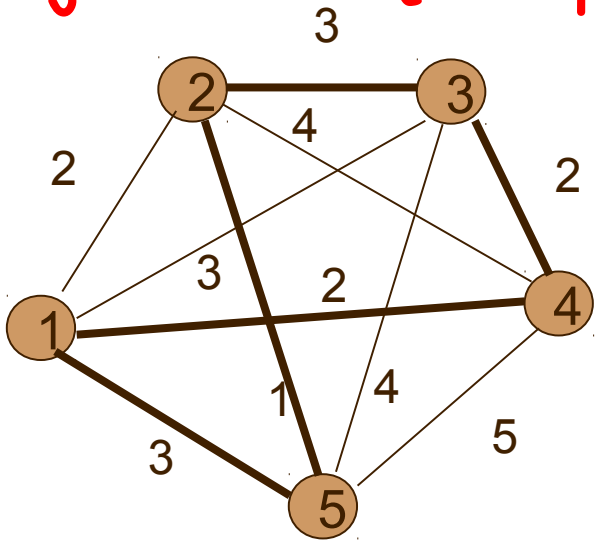


$$W = \begin{bmatrix} \infty & 2 & 3 & 2 & 3 \\ 2 & \infty & 3 & 4 & 1 \\ 3 & 3 & \infty & 2 & 4 \\ 2 & 4 & 2 & \infty & 5 \\ 3 & 1 & 4 & 5 & \infty \end{bmatrix}$$

- I rozwiązanie optymalne: (1-4-3-2-5-1) o długości 11.



Reprezentacja rozwiązań przez permutacje



- Cykl odwiedza wierzchołki w kolejności (1-4-3-2-5-1).
- Ponieważ ostatni wierzchołek jest taki sam jak pierwszy możemy go pominąć. Otrzymujemy 1-4-3-2-5 (jedna z 5! permutacji numerów wierzchołków 1 2 3 4 5).
- Czy liczba cykli Hamiltona jest równa n! ? Oczywiście nie, ponieważ za wierzchołek startowy możemy przyjąć dowolny wierzchołek grafu (np. 4-3-2-5-1 albo 3-2-5-1-4). Prowadzi to do liczby cykli równej n!/n=(n-1)!.
- Dodatkowo jeżeli macierz W jest symetryczna, to odwrócenie kolejności wierzchołków nie zmienia długości cyklu (Np. 1-4-3-2-5 → 5-2-3-4-1).
- Zatem mamy (n-1)!/2 rozwiązań problemu. Uwaga: to jest bardzo duża liczba ! Np. dla n=50 liczba cykli jest równa $49!/2=3.041*10^{62}$.

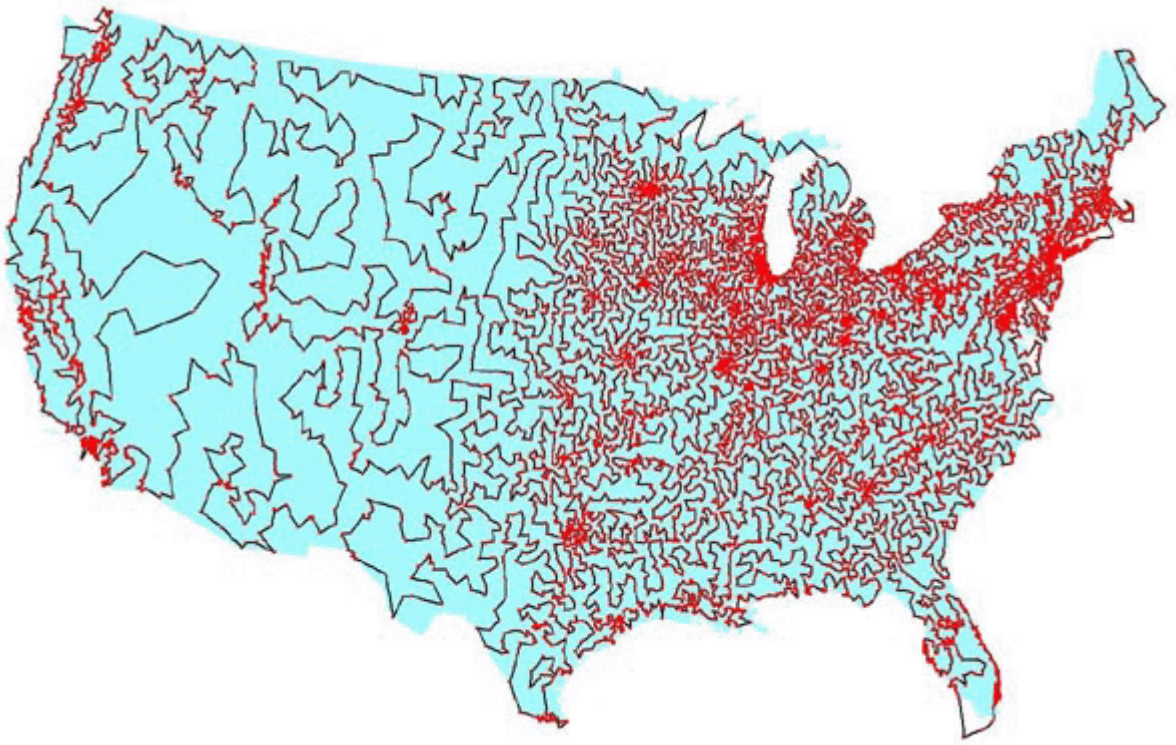
Problem komiwojażera jest NP-trudny

- Oznacza to, że obecnie istniejące algorytmy rozwiązujące ten problem mają pesymistyczną wykładniczą złożoność obliczeniową.

- Może zatem skonstruujmy algorytm który znajdzie rozwiązanie co najwyżej k razy dłuższe, niż optymalne, gdzie k jest pewną stałą.

- Niestety powyżej sformułowany problem jest również NP trudny. Jeżeli jednak założymy, że macierz wag **W** jest symetryczna oraz wagi krawędzi spełniają warunek trójkąta ($w_{ij} \leq w_{ik} + w_{kj}$ dla każdej trójki i,k,j) (***metryczny problem komiwojażera***) to istnieją algorytmy przybliżone o wielomianowej złożoności (np. algorytm Christofidesa, 1973, dla którego k=1.5).
  - Z taką sytuacją mamy do czynienia jeżeli wagi krawędzi są odległościami (np. Euklidesowymi pomiędzy punktami na płaszczyźnie)

Rozwiązania dokładne



- Wykorzystując najnowsze zdobycze nauki i techniki (programowanie matematyczne, obliczenia równoległe, potężne komputery i klastry) naukowcy byli w stanie wyznaczyć rozwiązania dokładne dla zdumiewająco dużego n.
- Rysunek przedstawia rozwiązanie dla 13509 miast w USA. (D. Applegate, W. Cook, V. Chvatal, 1998)
- Program liczy ponad 1000 stron kodu.
- Obliczenia zajęły 3 miesiące na mieszanym klastrze 3 x Digital Alphaserver 4100 (12 CPU) + (32xPentium II PC)

Rozwiązania heurystyczne

- Algorytmy nie gwarantujące znalezienia rozwiązania optymalnego, ale odnajdujące, w rozsądnym czasie, rozwiązanie leżące dość blisko optimum. Można podzielić je na dwie grupy:
- Algorytmy włączania (ang. insertion heuristics) startują od jednego (np. losowo wybranego) wierzchołka s. Następnie dodają do niego nowy wierzchołek q, tworząc cykl s-q-s.
  - Następnie dodają nowy wierzchołek p tworząc cykl s-p-q-s albo s-q-p-s, w zależności od tego który cykl będzie krótszy.
  - Kontynuują rozszerzanie rozwiązania częściowego za każdym razem wykonując jeden z dwóch kroków
    - wybierz (jak o tym za chwilę) nowy wierzchołek.
    - dodaj go do istniejącego cyklu na takiej pozycji, aby nowy cykl był jak najkrótszy.
- Algorytmy lokalnych poszukiwań biorą rozwiązanie (cykl Hamiltona) i starają się je ulepszyć znajdując lepsze rozwiązanie w sąsiedztwie bieżącego rozwiązania.
  - Algorytm 2-optymalny.
  - Algorytm 3-optymalny.
  - Algorytm Lina i Kernighana

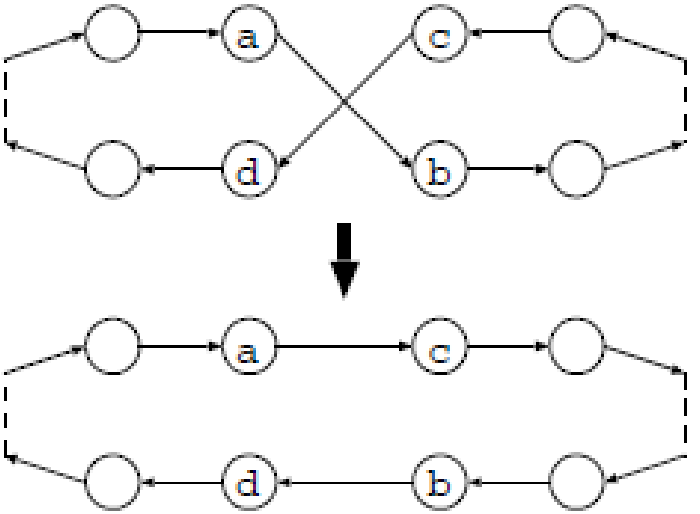
Algorytm włączania - kryteria wyboru nowego wierzchołka

- Nowy wierzchołek może być:
  - wybrany losowo (ang. random insertion)
  - położony najbliżej cyklu (ang. nearest insertion)
  - położony najdalej cyklu (ang. farthest insertion).
- przy czym odległość wierzchołka od cyklu jest definiowana jako najmniejsza waga krawędzi łączących wierzchołek z wierzchołkami wchodzącymi w skład cyklu.
- Sysło i współpracownicy (1999) twierdzą, że technika farthest insertion działa najlepiej.
  - procedura FITSP w języku Pascal w dodatku do książki
  - uwaga: tablice indeksowane od 1 !

Algorytm 2-optymalny (ang. 2-opt).

1. Wygeneruj początkowy cykl Hamiltona (np. heurystyką fathest insertion).
2. Rozważamy wszystkie cykle Hamiltona, które możemy uzyskać przez 2-wymianę krawędzi z cyklu bieżącego. 2-wymiana polega na

   - Usunięciu z cyklu dwóch dowolnych krawędzi. Krawędzie nie mogą być sąsiednie.
   - Zastąpieniu ich przez zestaw dwóch innych krawędzi tworzących inny cykl Hamiltona.

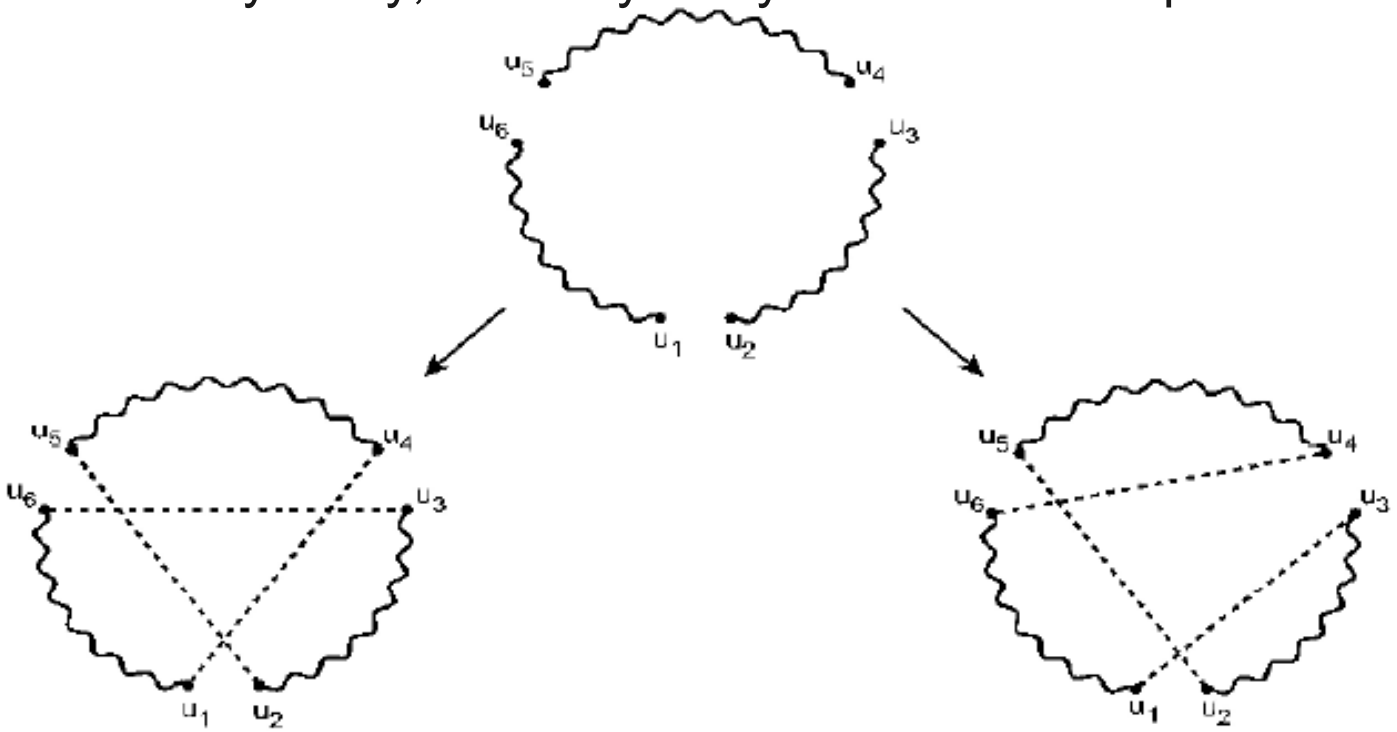


3. Spośród wszystkich cykli 1/2*n(n-3) cykli Hamiltona otrzymanych przez 2-wymianę z cyklu bieżącego wybierz ten który daje największą redukcję długości drogi komiwojażera. Ten krok wymaga $O(n^2)$ operacji.
4. Jeżeli przez 2-wymianę nie udało się zredukować długości drogi zakończ algorytm w przeciwnym razie skocz do punktu 2.

- procedura TWOOPT w dodatku do książki Sysły i współpracowników (1999).

Algorytm 3-optymalny (ang. 3-opt)

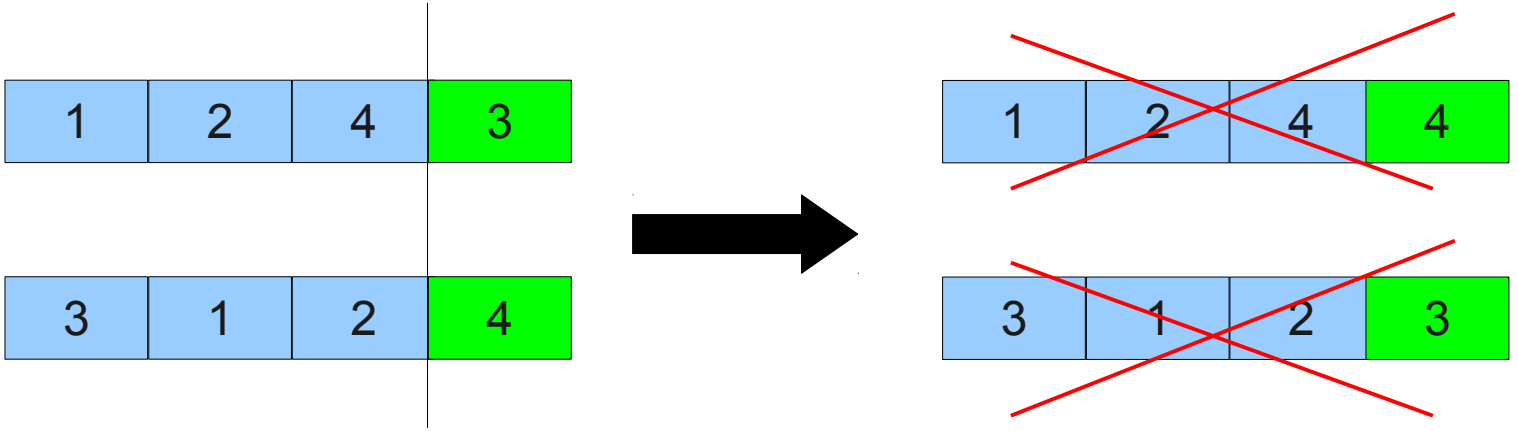
- W przypadku tego algorytmu rozważane są wszystkie cykle Hamiltona, które możemy otrzymać poprzez 3-wymianę krawędzi z cyklu aktualnego.
  - 3 – wymiana polega na usunięciu 3 krawędzi z cyklu, i ich ponownym połączeniu 3 powstałych dróg w cykl Hamiltona, przy czym w odróżnieniu od 2-wymiany, możemy to wykonać na kilka sposobów.



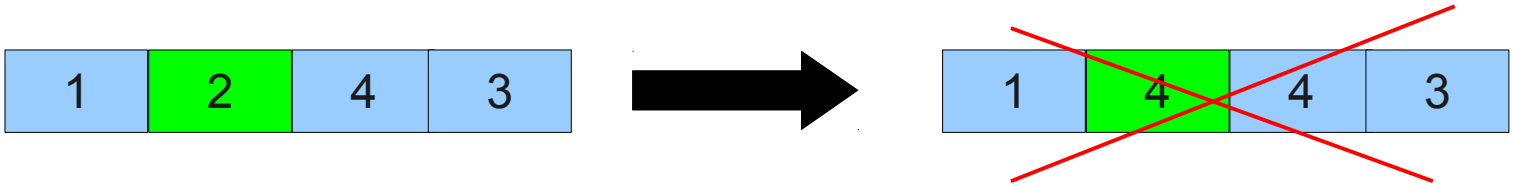
  - Liczba 3 – wymian jest $O(n^3)$.
- Procedura TREEOPT w dodatku do książki Sysły i współpracowników (1999).
- Algorytm 3-optymalny jest wyraźnie lepszy niż 2-optymalny, ale 4-optymalny nie opłaca się.

Problem komiwojażera, a algorytmy genetyczne

- Możemy funkcję dopasowania oprzeć na długości cyklu, a chromosomy o długości L=n niech będą permutacjami wierzchołków.
- Jednakże standardowe krzyżowanie:



oraz mutacja

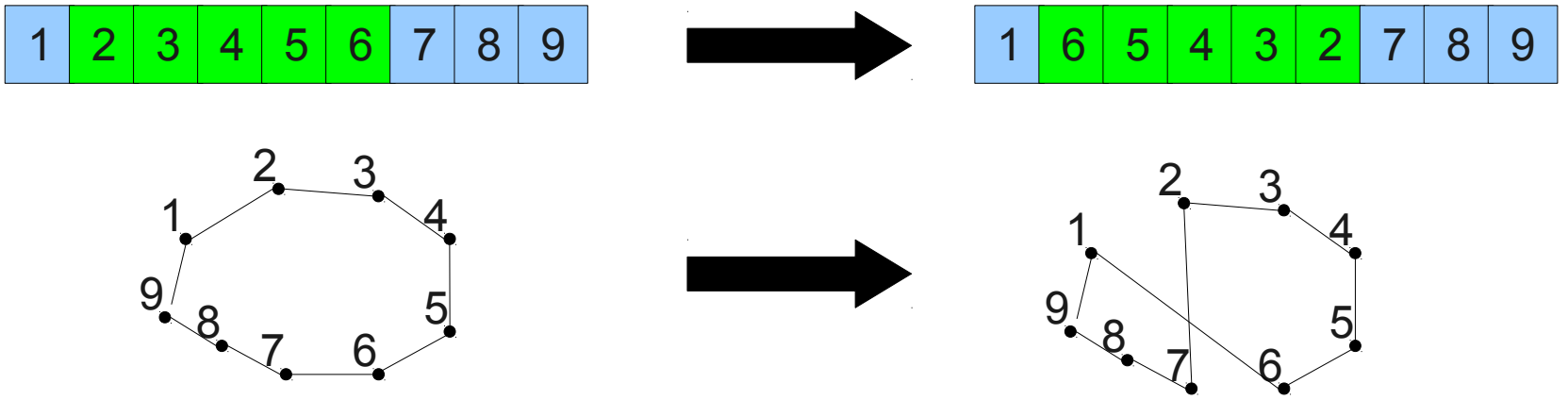


nie działają (ich wynikiem na ogół nie jest permutacja).

- Należy zaprojektować zatem operatory krzyżowania i mutacji operujące na permutacjach.

Operatory mutacji dla TSP – mutacja poprzez inwersję (ang. inversion mutation)

- Dla reprezentacji w postaci permutacji niemożliwe jest rozważanie poszczególnych genów niezależnie. W zwi
- Mutacja poprzez inwersję wybiera losowo dwie pozycje w chromosomie i odwraca kolejność wierzchołków pomiędzy nimi:



- Inwersja w permutacji jest równoważna zmianie jedynie dwóch krawędzi w cyklu (zakładamy problem symetryczny). Jest to najmniejsza zmiana, jaka może być wprowadzona przez operator mutacji.
- Czy ta mutacja coś Państwu przypomina ?

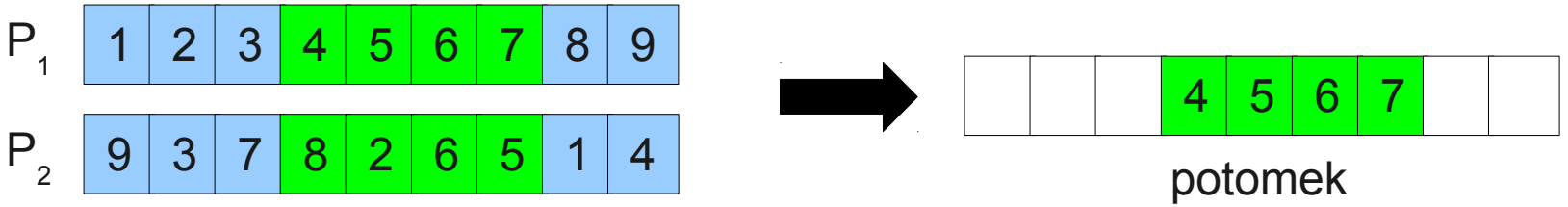
Operator Scramble Mutation

- Wybierz losowo podzbiór k pozycji w chromosomie.
- Przestaw losowo wierzchołki przechowywane na tych pozycjach.

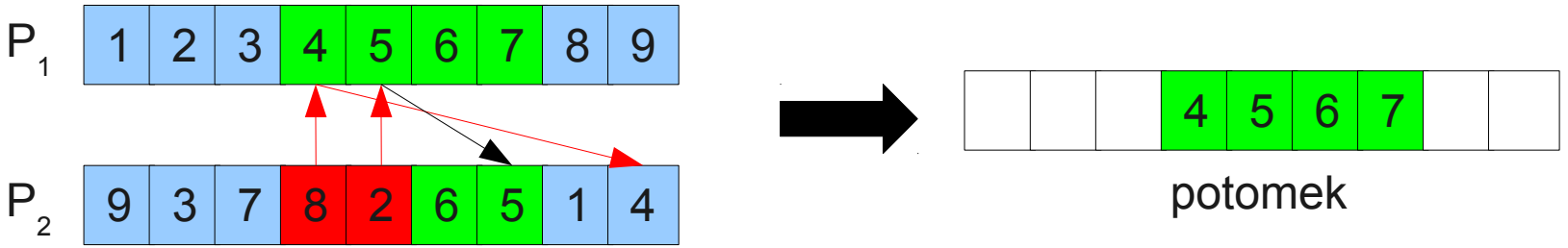
- Implementacja
  - Wylosuj numery pozycji $i_1,i_2,...,i_k$
  - Dla każdego wylosowanego numeru i_j zamień wierzchołek z innym losowo wybranym spośród pozycji $i_1,i_2,...,i_k$.
- Liczba k wylosowanych pozycji może być stała, albo zmniejszana wraz z postępem algorytmu.

Operator krzyżowania PMX (ang. partially matched crossover, Goldberg i Lingle, 1985)

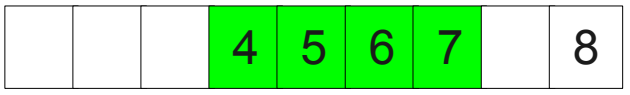
- Wybierz losowo dwa punkty krzyżowania. Skopiuj wybrany segment z rodzica P_1.



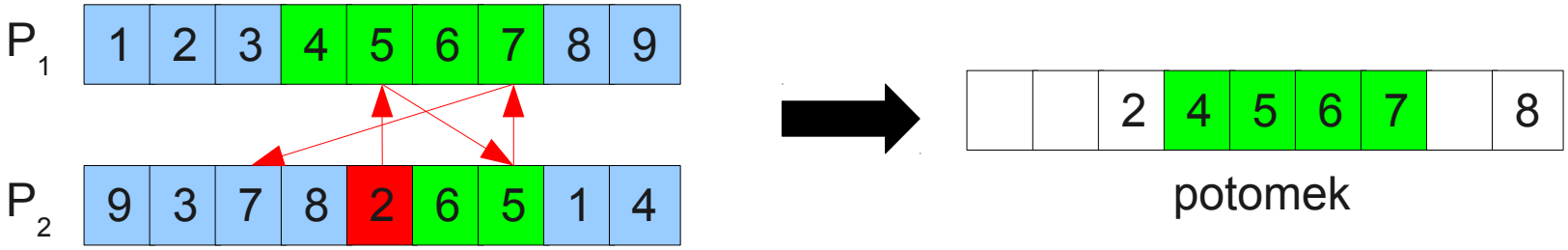
- Patrząc na analogiczny segment w P2 znajdź te elementy, które nie zostały skopiowane. (musimy je umieścić poza skopiowanym segmentem) Dla każdego z tych elementów i określ jaki element j został skopiowany z P1 na jego miejsce (na miejsce 8: 4, na miejsce 2: 5).



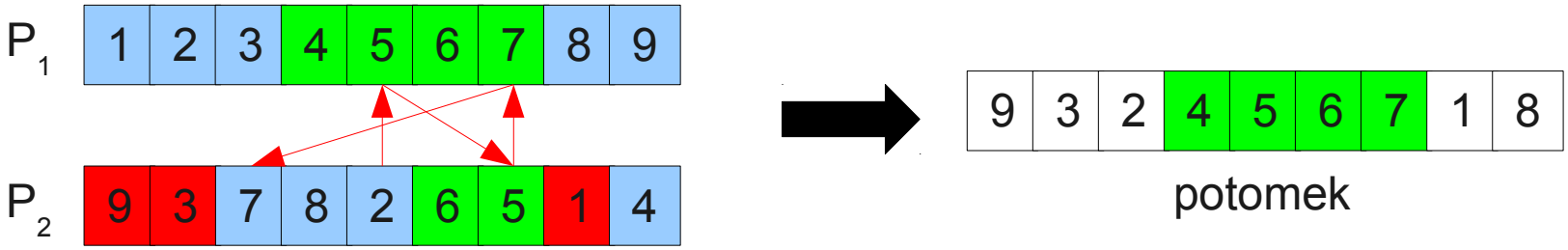
- A zatem mamy pary (i,j): (8,4) oraz (2,5). Dla każdej pary spróbuj umieścić i na pozycji zajmowanej przez j w rodzicu P_2. Dla 8 jest to możliwe, dla 2 nie, ponieważ 5 będąca na miejscu 2 został już skopiowany..



Operator PMX (2)



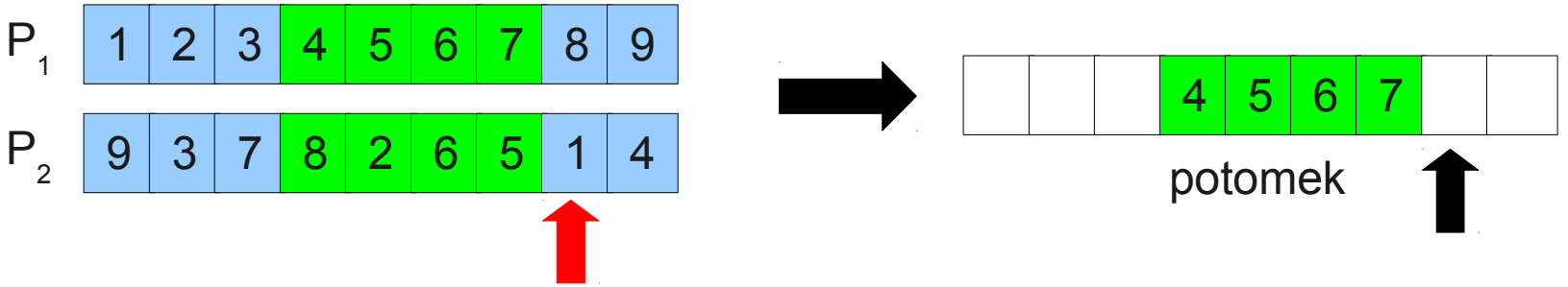
- W takiej sytuacji sprawdzamy który wierzchołek został skopiowany ma miejsce 5. Jest to 7, który nie został skopiowany jescze do potomka. Umieszczamy 2 na miejsce zajmowane w P_2 przez 7.
- Pozostałe wierzchołki kopiujemy z P_2.



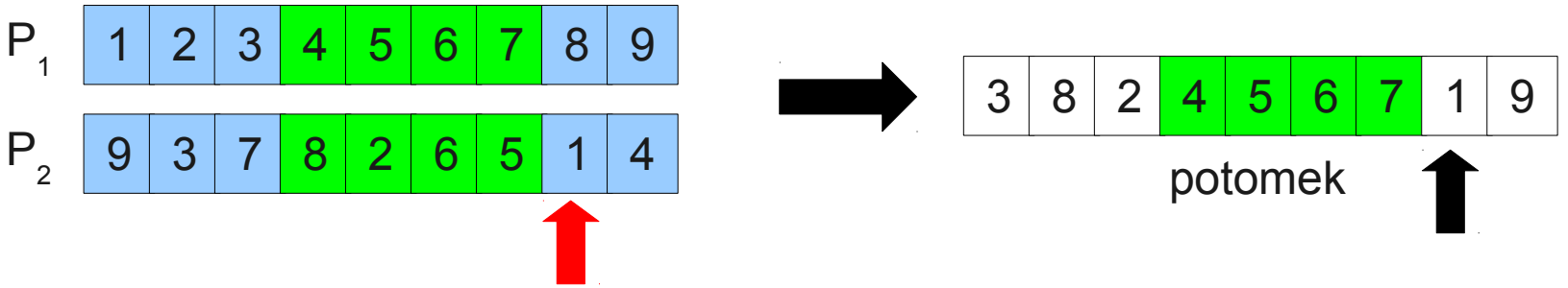
- Drugiego potomka generujemy zamieniając rodziców P_2 oraz P_1.
- Kod w PASCALu na stronie 186 książki Goldberga.

Order crossover (OX, Davis, 1985)

- Główna idea: zachowaj względną kolejność miast w cyklu. Na początku skopiuj segment od rodzica P_1 do potomka.



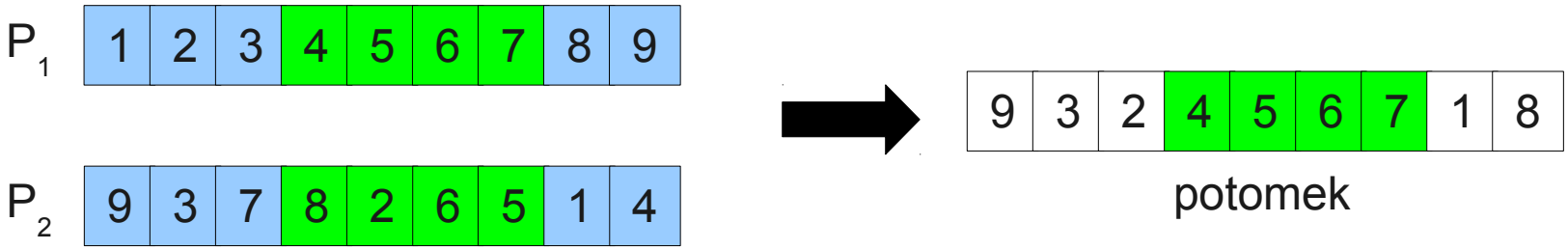
- Następnie rozpocznij (u potomka i rodzica P_2) zaraz za skopiowanym segmentem. Kopiuj wierzchołki w porządku w jakim występują w P_2, (czyli 1-4-9-3-7-8-2-6-5, z zawinięciem na końcu łańcucha). Pomijaj te wierzchołki, które zostały już skopiowane z rodzica P_1. (zaznaczone czerwono). Daje to kolejność wierzchołków 1-9-3-8-2). A zatem:



- Drugiego potomka generujemy zamieniając rodziców P_2 oraz P_1.

Jak zaprojektować lepszy operator krzyżowania ?

- W przypadku problemu komiwojażera informacja zawarta jest w krawędziach, to jest parach wierzchołków sąsiednich w permutacji. Tymczasem w operatorze PMX w naszym przykładzie (w OX jest jeszcze gorzej):



  - jedynie 6 z 9 krawędzi u potomka jest obecnych w jednym z rodziców.
  - dwie krawędzie 5-6 oraz 7-8 są obecne u ***obu*** rodziców, ale krawędzi 7-8 nie ma u potomka.

- Pożądana własność operatora krzyżowania: krawędź obecna u obu rodziców powinna także znaleźć się u potomka.
- Pożądana własność operatora krzyżowania: wszystkie krawędzie u potomka powinny pochodzić od jednego z rodziców.
- Badania naukowe zmierzają do opracowania operatorów krzyżowania spełniających pierwszą i możliwie jak najbardziej długą własność.

Operator EX (Edge Crossover).

- Opis dotyczy wersji edge-3 (Whitley, 2000).
- Algorytm rozpoczyna się poprzez konstrukcję tablicy krawędzi (ang. edge table). W tej tablicy dla każdego wierzchołka wylicza wierzchołki osiągane z tego wierzchołka u obydwu rodziców. Jeżeli ten sam wierzchołek docelowy jest osiągalny u obydwu rodziców, po jego numerze występuje znak „+”.
- Przykład: dla osobników:

P_1	1	2	3	4	5	6	7	8	9

P_2	9	3	7	8	2	6	5	1	4

mamy następującą tablicę krawędzi:

Element	Edges	Element	Edges
1	2,5,4,9	6	2,5+,7
2	1,3,6,8	7	3,6,8+
3	2,4,7,9	8	2,7+, 9
4	1,3,5,9	9	1,3,4,8
5	1,4,6+		

Operator EX (2)

- Tworzenie potomka odbywa się przy wykorzystaniu tablicy krawędzi, przy pomocy następującego algorytmu (algorytm dodaje wierzchołki do początkowo pustej listy, w każdym kroku utrzymując ciąg wierzchołków):

    1. v=losowy wierzchołek
    2. usuń z tablicy krawędzi wszystkie odniesienia do v.
    3. Zbadaj listę krawędzi dla v. Jeżeli to możliwe przejdź do
        a) wierzchołka połączonego krawędzią wspólną (znak +).
        b) wierzchołka, połączonego krawędzią nie wspólna, przy czym spośród kilku możliwości wybierz taki wierzchołek, którego lista krawędzi jest najkrótsza (w przypadku kilku list o tej samej długości wybierz jeden z wierzchołków losowo).
    4. Jeżeli lista krawędzi v jest pusta spróbuj wykonać pkt 3. dla drugiego końca ciągu).
    5. Jeżeli pkt. 4 się nie udał, dodaj do listy nowy wierzchołek losowo.
    6. Jeżeli nie skonstruowałeś jeszcze całej permutacji v=dodany wierzchołek i idź do punktu 2.

- Tylko w przypadku realizacji pkt. 5 dodawana jest krawędź nieobecna u rodziców.

Operator EX – przykład (Eiben & Smith, 2003)

Element	Edges	Element	Edges
1	2,5,4,9	6	2,5+,7
2	1,3,6,8	7	3,6,8+
3	2,4,7,9	8	2,7+, 9
4	1,3,5,9	9	1,3,4,8
5	1,4,6+		

Choices	Element selected	Reason	Partial result
All	1	Random	[*1*]
2,5,4,9	5	Shortest list	[*1 5*]
4,6	6	Common edge	[*1 5 6*]
2,7	2	Random choice (both have two items in list)	[*1 5 6 2*]
3,8	8	Shortest list	[*1 5 6 2 8*]
7,9	7	Common edge	[*1 5 6 2 8 7*]
3	3	Only item in list	[*1 5 6 2 8 7 3*]
4,9	9	Random choice	[*1 5 6 2 8 7 3 9*]
4	4	Last element	[*1 5 6 2 8 7 3 9 4*]

Algorytm hybrydowy

- Algorytm łączący algorytm ewolucyjny z algorytmem przeszukiwań lokalnych.

- Idea: uruchom (być może nie do końca) algorytm 2-optymalny lub 3-optymalny dla każdego rozwiązania stworzonego przez krzyżowanie i mutację.

- Przy czym heurystyka poszukiwań lokalnych może być iterowana:
  - raz (jedna iteracja).
  - do momentu zatrzymania się (nie można dalej ulepszyć rozwiązania).
  - do momentu zatrzymania się, ale nie więcej niż k iteracji.

- Rozwiązania w populacji początkowej stwórz uruchamiając algorytm 2-opt albo 3-opt (do momentu zatrzymania się) na permutacjach wygenerowanych losowo.

Materiały dodatkowe

- M. M. Sysło, N. Deo, J. S. Kowalik, Algorytmy optymalizacji dyskretnej z programami w języku Pascal, WNT, Warszawa 1999. (rozdział 3.8 poświęcony jest problemowi komiwojażera)
- http://www.or.deis.unibo.it/research_pages/tspsoft.html - Strona z linkami do heurystycznych oraz dokładnych implementacji algorytmów TSP oraz z linkami do appletów demonstrujących pracę tych algorytmów.
- http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ TSPLIB jest biblioteką instancji problemów komiwojażera powszechnie stosowaną w porównywaniu algorytmów. Zawiera dane do problemu 13509 miast w USA. Wszystkie problemy w TSPLIB zostały dokładnie rozwiązane. Uwaga na kodowanie współrzędnych ! (proszę przeczytać dokumentację).
- http://www.tsp.gatech.edu/index.html - strona twórców pakietu CONCORDE, za pomocą którego rozwiązano optymalnie problem TSP dla 13509 miast. Zawiera kod oraz dane przykładowe w formacie TSPLIB.
- D. Whitley, D. Hains, A. Howe, Tunneling between optima: partition crossover for the traveling salesman problem, Proceedings of the 11th Annual conference on Genetic and evolutionary computation, s. 915-922, 2009, opis operatora PX.
- Y. Nagata and S. Kobayashi. Edge Assembly Crossover: A High-Power Genetic Algorithm for the Traveling Salesman Problem. 7th International Conf. on GAs, Morgan Kaufmann, 1997. opis operatora EAX.

Wymagania odnośnie projektu

- Na ocenę bardzo dobry : implementacja trzech operatorów krzyżowania, w tym jednego przechowującego jak najwięcej krawędzi (EAX albo PX albo EX).
- Na ocenę dobry dwa operatory krzyżowania.
- Operatory Inversion mutation i Scramble mutation (W przypadku scramble mutation warto dodać sterowanie liczbą przestawianych elementów, tak aby malała ona wraz z postępem algorytmu).
- Implementacja heurystyki przeszukania lokalnego (np. 3-opt) nie musi być własna, ale trzeba a) podać w raporcie z projektu źródło b) wiedzieć jak ta heurystyka działa.
  - Uwaga, ze względu na mniejszą złożoność obliczeniową proponuję aby w algorytmie hybrydowym wykorzystać heurystykę 2-opt.
- Badania eksperymentalne z wykorzystaniem danych z TSPLIB albo innego zbioru danych ze znanym rozwiązaniem optymalnym.