Metoda UCT w stochastycznych problemach transportowych

mgr inż. Maciej Świechowski
promotor: prof. Jacek Mańdziuk

Seminarium Metody Inteligencji Obliczeniowej 25.06.2015

Plan prezentacji

- Krótkie przypomnienie omawianego wariantu problemu
- Metodologia weryfikacji rozwiązań
- Podejście UCT
  - Nowe zmiany
  - Dostępne akcje
- Metody porównawcze
- Wyniki
- Krótkie podsumowanie

Stochastyczny problem transportowy

Konkretny problem: **CVRPwTJ**

Capacitated Vehicle Routing problem with Traffic Jams

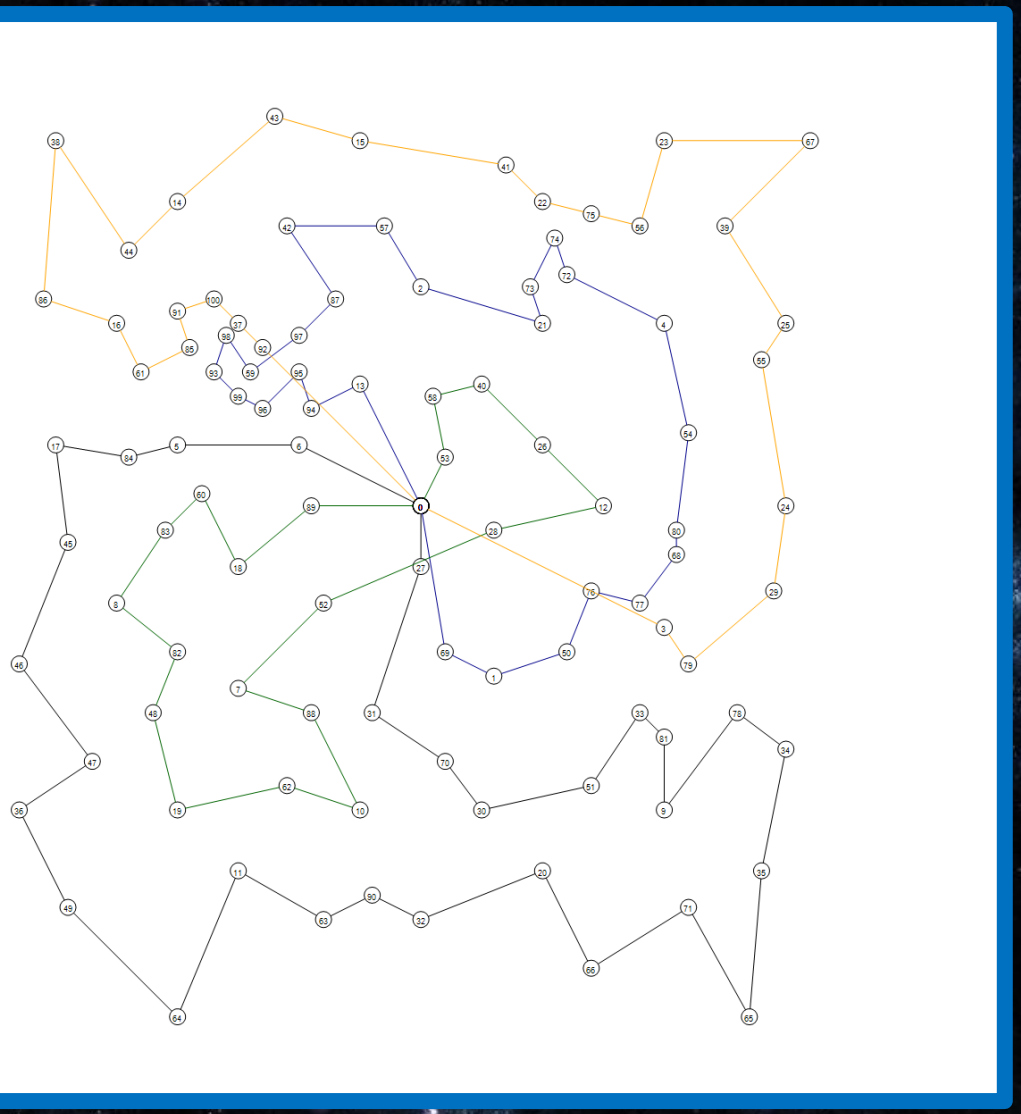
VRP – minimalizacja sumy tras ciężarówek na grafie klientów

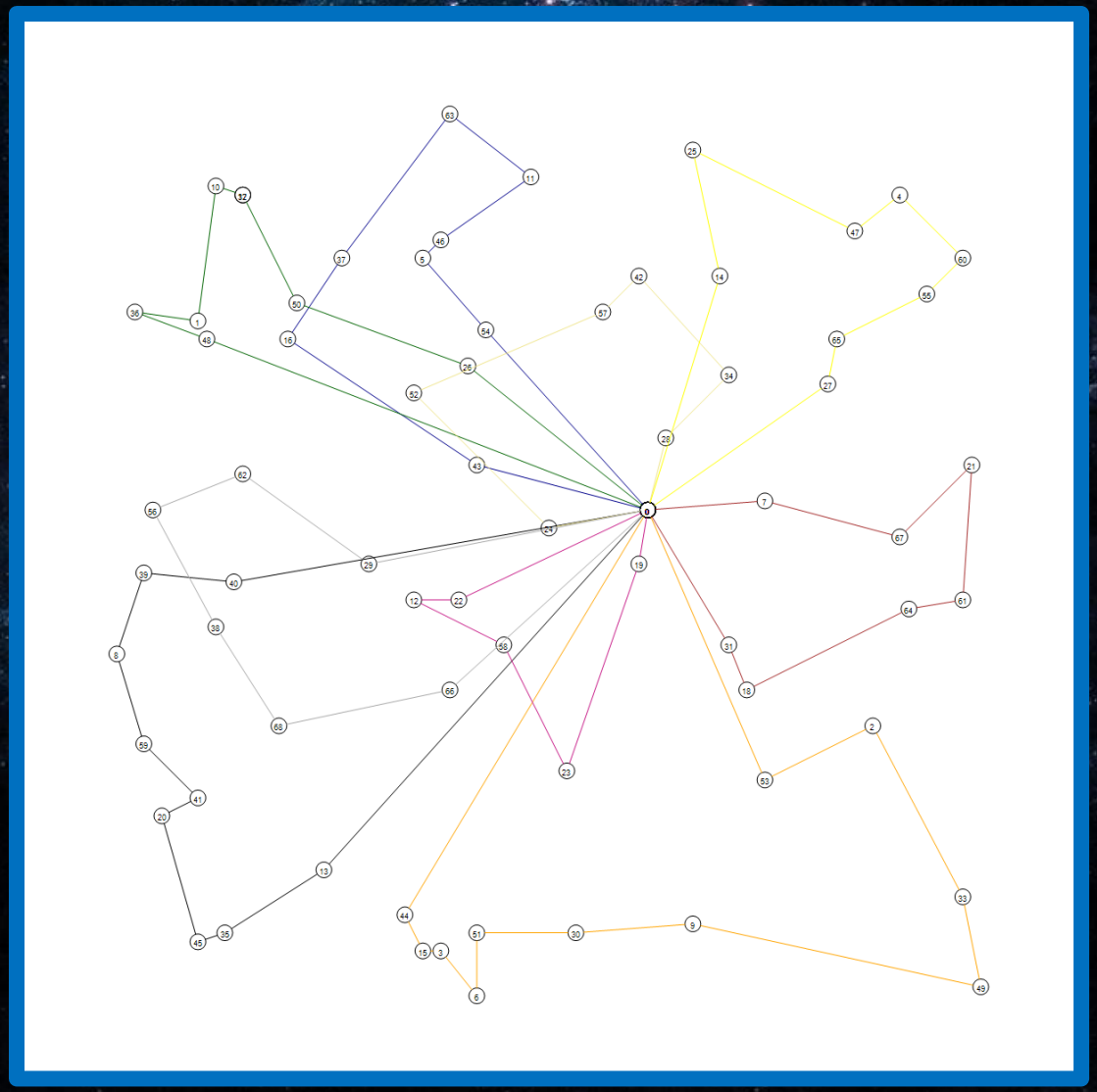
- trasy tworzą podział zbioru klientów
- każda trasa zaczyna się i kończy w zajezdni (wyróżniony wierzchołek)

Capacitated – klienci mają zdefiniowaną wielkość zamówienia a ciężarówki ograniczenie pojemności

Traffic Jams – występuje dynamiczne zakorkowanie: rozwiązanie problemu jest weryfikowane symulacyjnie

Rozwiązanie początkowe = statyczne





Rozwiązanie początkowe

Algorytm **Improved Clarke Wright Savings**

- punkt wyjściowy dla każdej testowanej metody (może z niego skorzystać)
- rozwiązuje problem dla wersji statycznej (bez korków)

Benchmark	Optimal	Initial	Init/Optimal	Optimal/Init
A54-k7	1167	1201	103%	97%
A69-k9	1168	1211	104%	96%
A80-k10	1764	1861	105%	95%
P101-k4	681	765	112%	89%
P45-k5	510	573	112%	89%
P19-k2	212	238	112%	89%
E30-K4	?	534	?	?
E51-K5	521	585	112%	89%
E76-K15	?	1073	?	?
E76-K7	682	738	108%	92%

Porządkujące zasady

Aktualizacja:

- Krokowa, dyskretna
- Ciężarówki są przesuwane zgodnie ze swoim kursem
- Ciężarówki poza zajezdnią muszą mieć kurs
- Przejazd między lokalizacjami to operacja atomowa
- Co najmniej jedna ciężarówka musi się ruszyć (*ang. net change*) (przeciwna sytuacja oznacza koniec symulacji)

Metodologia testowania metod

Testowanie podczas symulacji danego benchmarku

- **N** powtórzeń z innymi realizacjami korków

Korek określony trzeba rozkładami:

- prawdopodobieństwo **p ∈ {0.02, 0.05, 0.15}**
- czas trwania **TTL := uniform<int>(2,5)**
- intensywność **I := uniform<int>(10,20)**

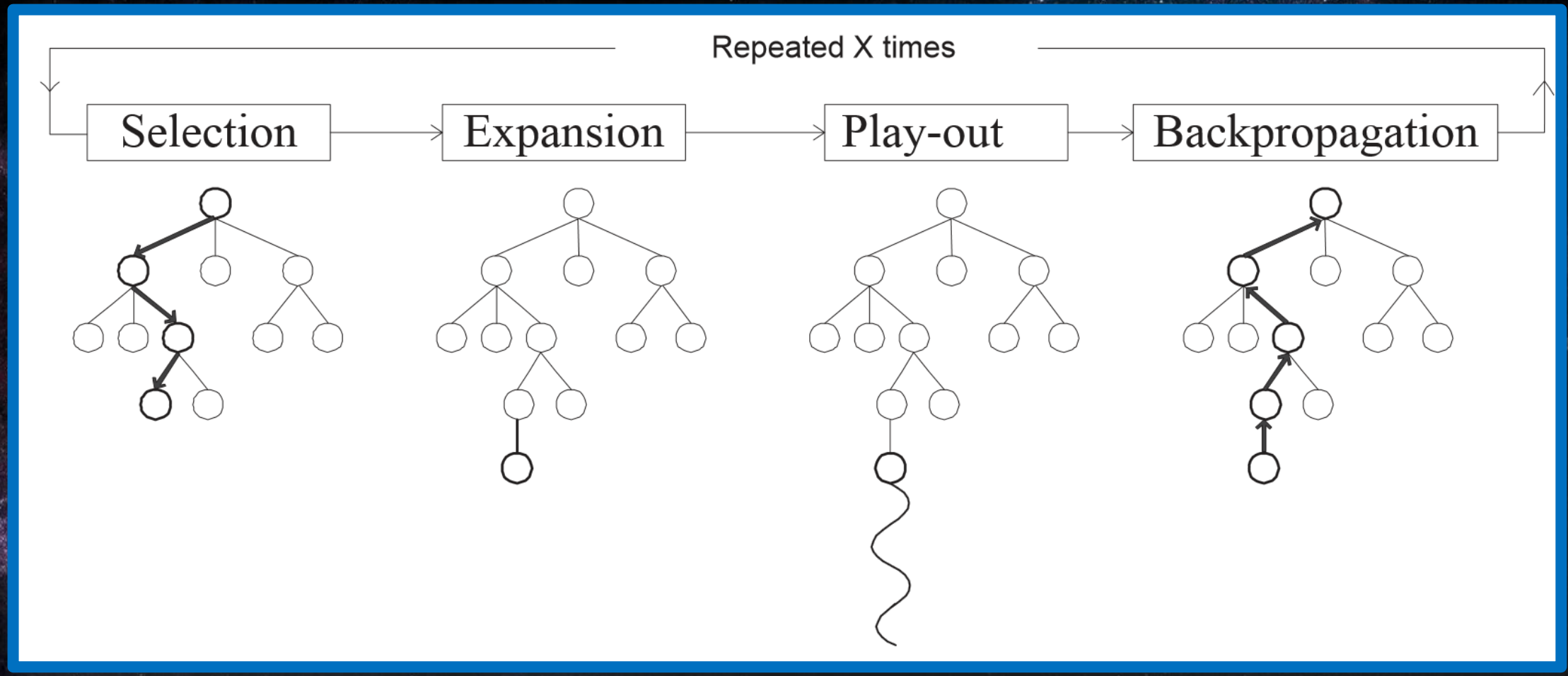
Metody nie znają konkretnych realizacji korków, lecz mają dostęp do rozkładów prawdopodobieństwa, na podstawie których są generowane korki.

Krótkie przypomnienie algorytmu **UCT** w **CVRPwTJ**...

Główne zmiany:

- Redukcja złożoności drzewa UCT - **tabele transpozycji**
- Skrócenie symulacji – **early cutoff**
- Drobna modyfikacja zestawu akcji

Metoda UCT

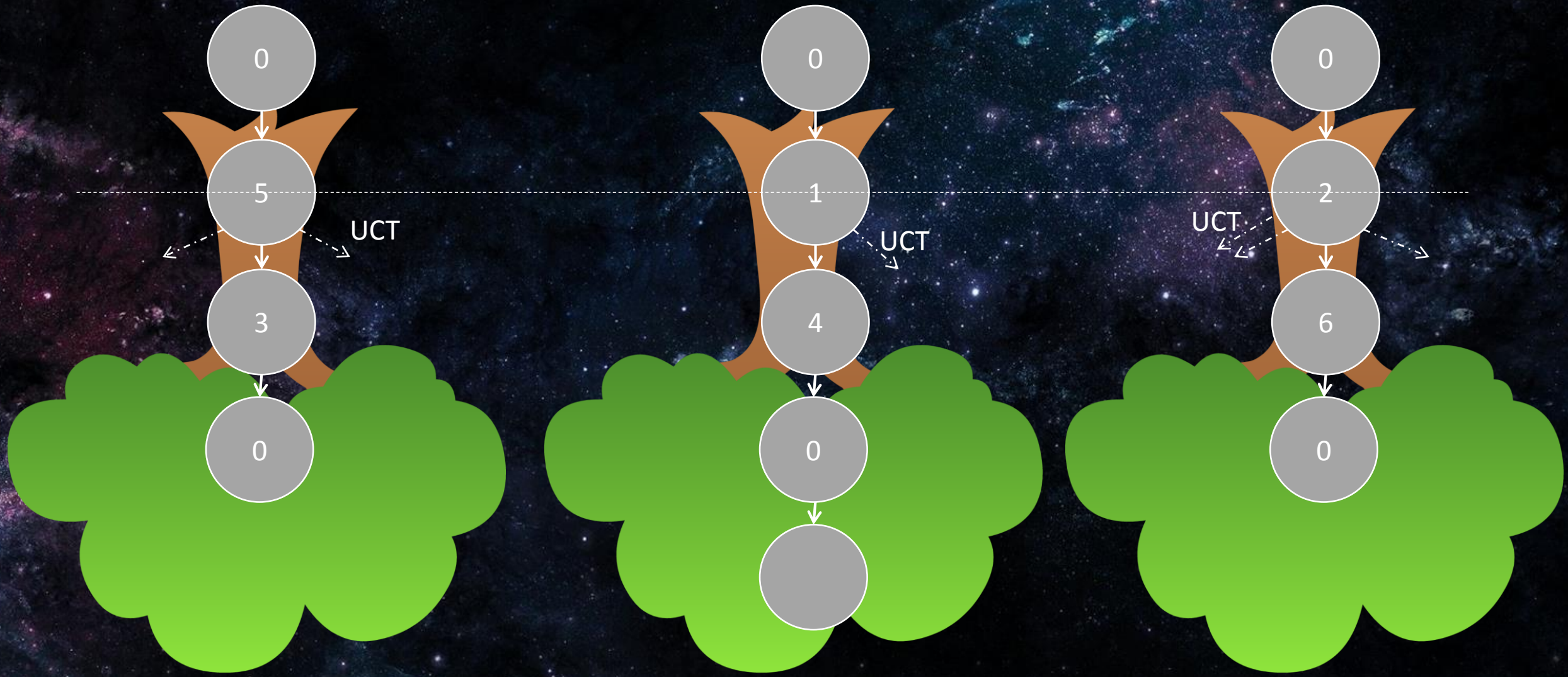

Repeated X times
Selection
Expansion
Play-out
Backpropagation

Metoda UCT

- Buduje ***k*** drzew UCT, gdzie ***k*** to liczba tras w rozwiązaniu początkowym
- Wierzchołek w drzewie to **ciężarówko-stan**
- Symulacja wykonuje ***k*** algorytmów UCT równolegle
  - Są synchronizowane na danym poziomie – głębokości w drzewie

- Brak klasycznej fazy Monte Carlo, bo jest połączona z fazą UCT
- Drzewa są rozbudowywane przy pomocy bogatej rodziny dostępnych akcji

Metoda UCT



Ciężarówko-stan

Węzeł opisuje stan względem danej ciężarówki (⇔ trasy)

Ciężarówko-stan unikalnie opisany jest przez:

- Bieżącą lokalizację ciężarówki: ***CurrentCity***
- Pozostałą wolną pojemność ciężarówki: ***CapacityLeft***
- Pozostały zbiór miast do odwiedzenia przez ciężarówkę: ***CitiesToVisit***

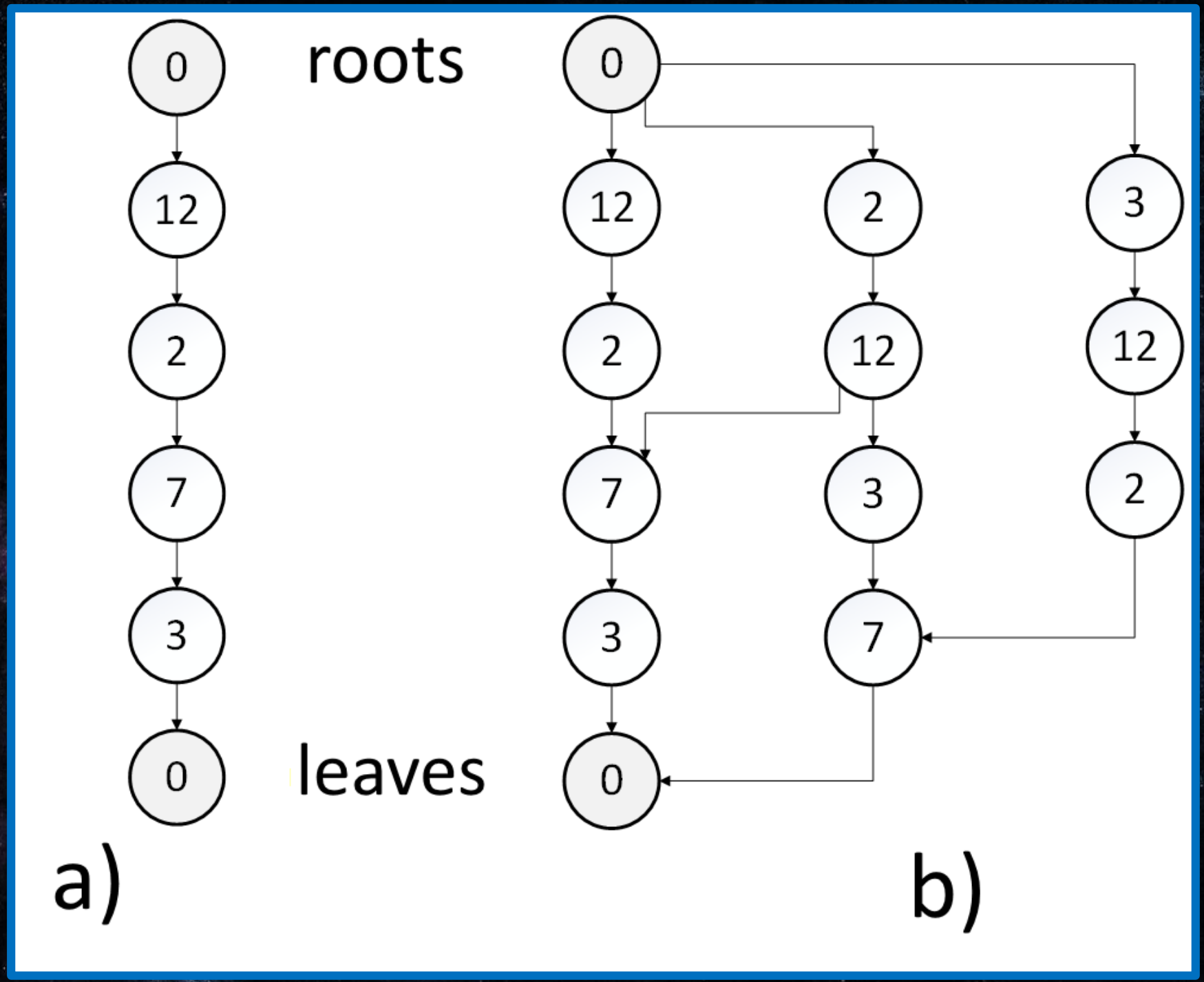
Na potrzeby wydajnej optymalizacji jest:

- hashowanie stanów
- kilka dodatkowych pól (nieistotnych dla zrozumienia idei algorytmu)
- heurystyka określająca kolejność miast w *CitiesToVisit*

Przechowywanie bazowej trasy do obróbki

W momencie gdy chcemy przypisać, jako rezultat akcji, węzeł drzewa (ciężarówko-stan), sprawdzamy czy istnieje transpozycja

- jeżeli nie istnieje, to tworzymy nową i zwracamy nowy węzeł tak jak go stworzyliśmy
- jeżeli istnieje, to sprawdzamy czy zaplanowana trasa w nowym węźle jest statycznie krótsza niż w przechowywanej transpozycji. Jeżeli jest, to podmieniamy w niej plan. Zwracamy transpozycje.

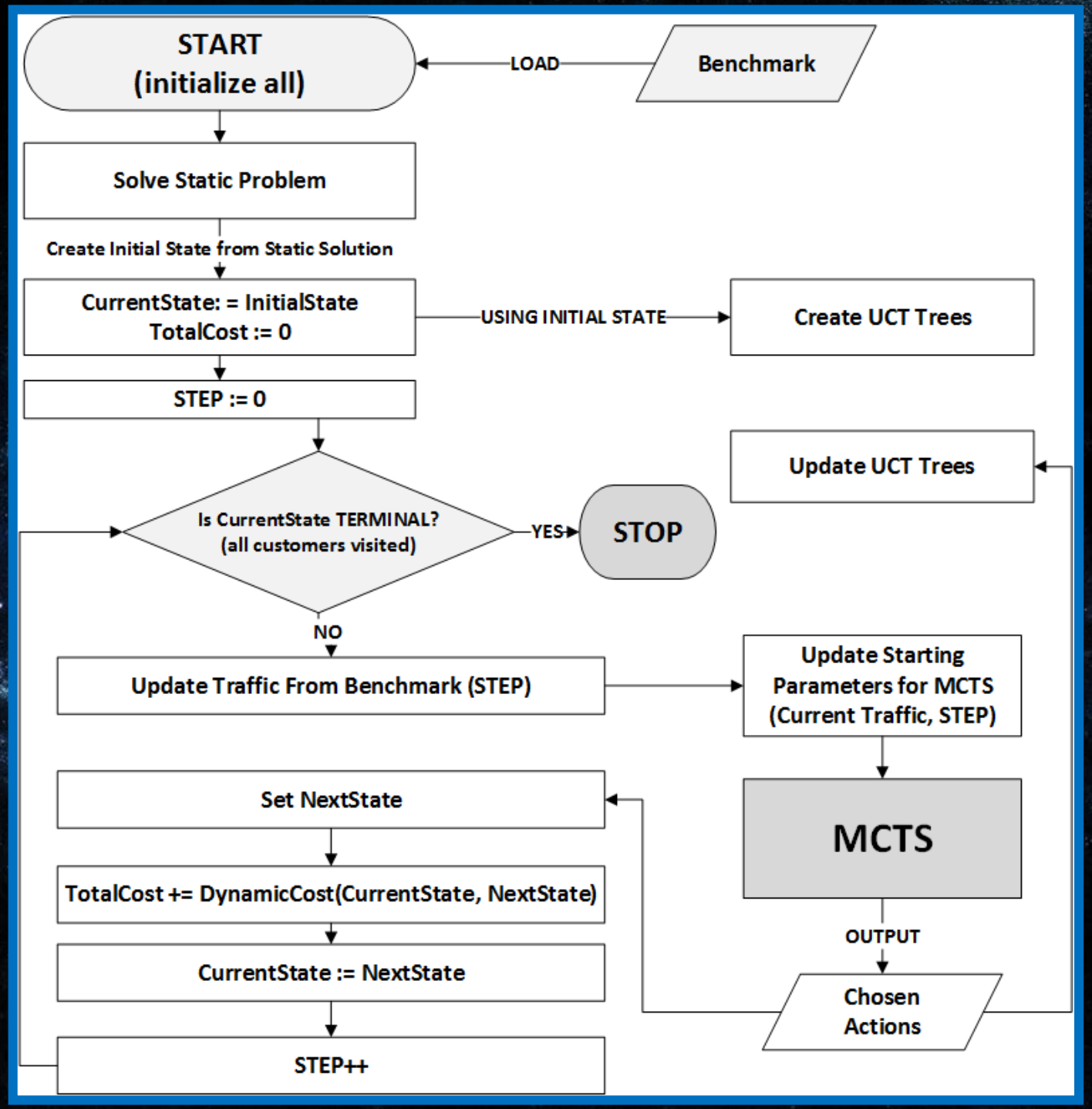

roots
0
12
2
7
3
0
leaves
a)
0
12
2
7
3
0
2
12
3
7
12
2
b)

Wzór UCT

- **Q(s,a) –** średnia sumaryczna długość tras przy wybraniu akcji **a** w stanie **s**
- **A(s) –** akcje legalne w stanie **s**
- **Q** aktualizowane po zakończonej symulacji (pamiętane są ścieżki)
- **C** ustawione po krótkim testowaniu na długość początkowego rozwiązania

$$a^* = arg \max_{a \in A(s)} \left\{ C \sqrt{\frac{\ln N(s)}{N(s,a)}} - Q(s,a) \right\}$$

Metoda UCT


START
(initialize all)
LOAD
Benchmark
Solve Static Problem
Create Initial State from Static Solution
CurrentState: = InitialState
TotalCost := 0
USING INITIAL STATE
Create UCT Trees
STEP := 0
Update UCT Trees
Is CurrentState TERMINAL?
(all customers visited)
YES
STOP
NO
Update Traffic From Benchmark (STEP)
Update Starting
Parameters for MCTS
(Current Traffic, STEP)
Set NextState
MCTS
TotalCost += DynamicCost(CurrentState, NextState)
OUTPUT
CurrentState := NextState
Chosen
Actions
STEP++

```
TreeSimulationStep()
{
    traffic = ResolveTraffic();
    foreach(UctNode node in inNodes)
    {
        legalActions = actionPreparer.ComputeLegal(node, inNodes, traffic);
        node.SelectedAction = UCT(legalActions);
    }
    inNodes.Sort(UCTComparer);
    outNodes.Clear();

    foreach(UctNode node in inNodes.Where(node => node.NonTerminal)
    {
        if(node.SelectedAction.LinkedAction != null) //akcja ma wpływ na inny węzeł
            SynchronizeActions(node, node.SelectedAction.LinkedAction.ToNode);
        ApplyAction(node.SelectedAction);
        if(node.SelectedAction.ResultNode.CityID != 0)
            outNodes.Add(node.SelectedAction.ResultNode);
        TotalCost += GetDistance(node.CityID, node.SelectedAction.ResultNode.CityID);
    }
    if(outNodes.Count == 0)
     return;
    swap(inNodes, outNodes);
}
```

„Early Cutoff”

- Symulacja wykonuje maksymalnie **K** kroków od startu i jest kończona. Koszt symulacji powiększany jest o sumę statycznych długości pozostałych (niedokończonych) tras.
- **K** wynosi tyle, co maksymalna możliwa długość trwania korka. Przyjęto **K = 5**.

Motywacja:

- Masywne przyspieszenie działania metody
- Metoda stosowana w domenie gier często z gorszą heurystyką niż mamy tutaj
- Za horyzontem 5 kroków nie ma już żadnego związku zakorkowania z zakorkowaniem w startowym stanie

Akcje

Akcje to lokalne modyfikacje tras

- definiują wynik modyfikacji
- są nierozłączne z precyzyjnymi **warunkami na legalność**

  (ten sam wynik modyfikacji przy różnych warunkach legalności to różne akcje!)

Dla akcji przechowywane są parametry UCT (wyniki, licznik wizyt)

Akcje modyfikujące 0 tras

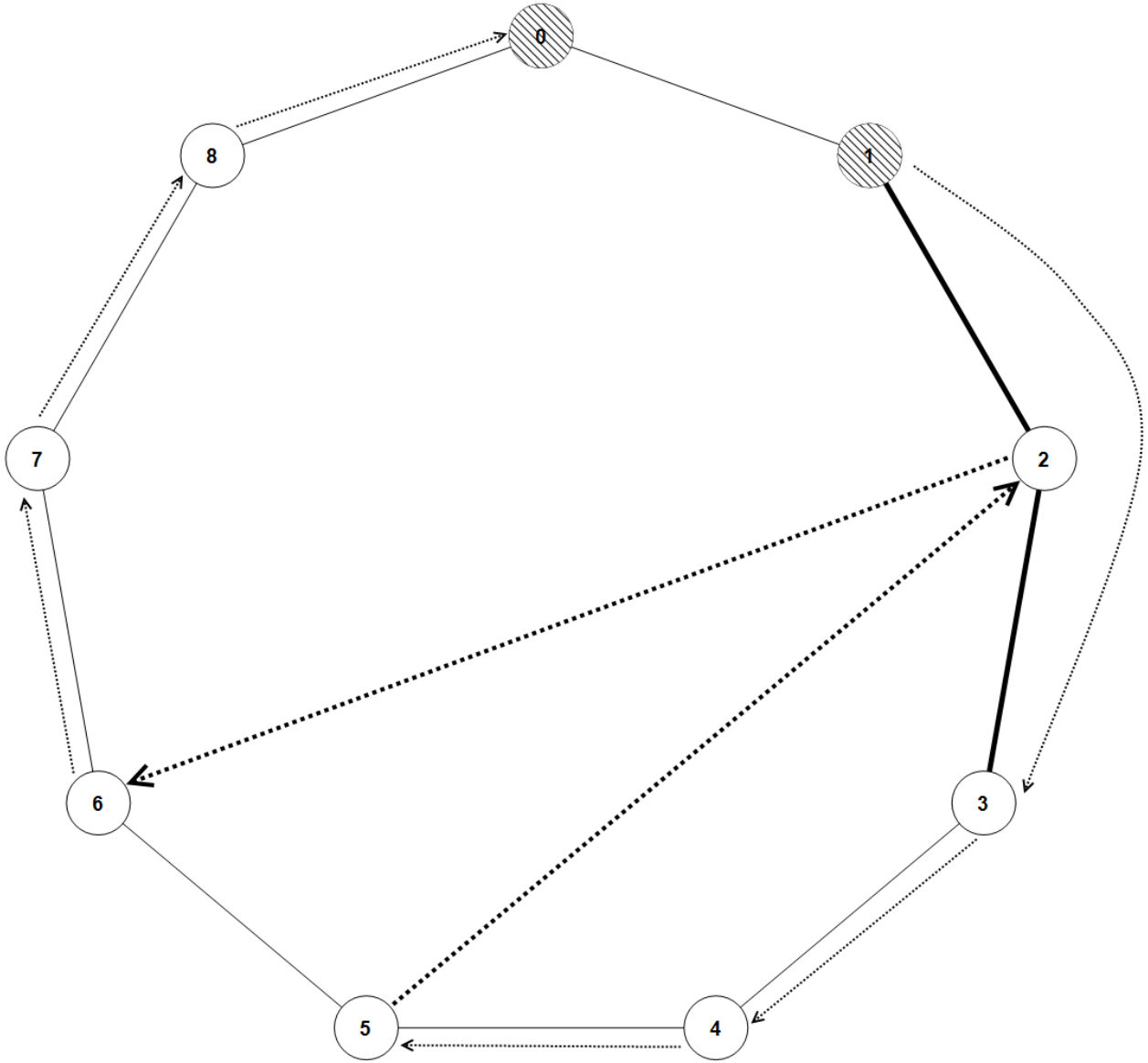
A0	Jedź dalej po zaplanowanej (niezakorkowanej) trasie
Legalność	• Trasa nie rozpoczyna się zakorkowaną krawędzią
Komentarz	

A1	Jedź dalej po zaplanowanej (zakorkowanej) trasie
Legalność	• Trasa rozpoczyna się zakorkowaną krawędzią
Komentarz	Fizyczna realizacja jest taka sama jak A0, ale akcje są **rozróżniane**. Warunki na legalność są wykluczające i dopełniające się – dzięki temu zawsze istnieje co najmniej 1 legalna akcja jak w dobrze zdefiniowanej grze.

Akcje wpływające na 1 trasę

A2	Przesuń bieżącego klienta na koniec trasy
Legalność	• Trasa rozpoczyna się zakorkowaną krawędzią • Po wykonaniu akcji, trasa nie rozpoczyna się zakorkowaną krawędzią
Komentarz	• Pierwszy klient, który byłby odwiedzony jest zaplanowany jako ostatni przed powrotem do zajezdni • Nowym pierwszym klientem staje się poprzedni drugi

A3	Przesuń kolejnego klienta w miejsce statycznie optymalne
Legalność	• Trasa rozpoczyna się zakorkowaną krawędzią • Po wykonaniu akcji, trasa nie rozpoczyna się zakorkowaną krawędzią
Komentarz	Niech początek trasy to klient A. Szukamy takich sąsiadujących na trasie klientów B i C (B≠A oraz C≠A), aby minimalizować: |BA| + |AC| - |BC| *Wstawiamy A między B i C.*



we find such X which minimizes: d(X,2) + d(2, X+1) - d(X, X+1)
in this example X = 5

A4	Jako następnego klienta ustaw pierwszego, do którego nie ma korka z bieżącej pozycji
Legalność	• Trasa rozpoczyna się zakorkowaną krawędzią • Po wykonaniu akcji, trasa nie rozpoczyna się zakorkowaną krawędzią
Komentarz	• Poprzednio planowany następny klient staje się drugim do odwiedzenia

A5	Odwróć trasę
Legalność	• Trasa rozpoczyna się zakorkowaną krawędzią • Po wykonaniu akcji, trasa nie rozpoczyna się zakorkowaną krawędzią
Komentarz	Zajezdnia pozostaje na miejscu Np. [9, 3, 5, 7, 1, 0] staje się [1, 7, 5, 3, 9, 0]

A6	Jako następnego klienta ustaw dynamicznie najbliższego
Legalność	• Trasa nie rozpoczyna się zakorkowaną krawędzią
Komentarz	• Dynamicznie najbliższy = najmniej kosztowny przejazd z uwzględnieniem korków • Ocena akcji Q jest mnożona przez **1.15** (kara za zachłanne zachowanie)

A7	Jako następnego klienta ustaw drugiego dynamicznie najbliższego
Legalność	• Trasa nie rozpoczyna się zakorkowaną krawędzią
Komentarz	• Ocena akcji Q jest mnożona przez **1.15** (kara za zachłanne zachowanie)

Uwaga:

Żadne akcje nie są dublowane. Jeżeli dwie lub więcej identycznych akcji miałoby być legalnych, to dostępna jest tylko ta o mniejszym ID.

Np. A7 nie jest dostępna, jeżeli jest jeszcze tylko 2 klientów do odwiedzenia.

Akcje wpływające na 2 trasy

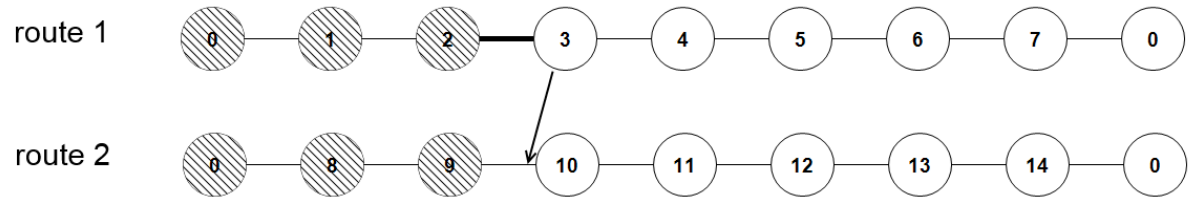
A8	Zakończ bieżącą trasę i utwórz nową z pozostałych klientów
Legalność	• Trasa jest całkowicie zakorkowana z bieżącej pozycji • Po wykonaniu akcji, trasa nie rozpoczyna się zakorkowaną krawędzią
Komentarz	• Z bieżącej pozycji ciężarówki istnieje korek do każdego klienta na trasie • Droga z bieżącej pozycji do zajezdni jest niezakorkowana • Droga z zajezdni do pierwszego klienta jest niezakorkowana • Jeżeli limit ciężarówek[1] nie został przekroczony, to nowa ciężarówka startuje od razu – jeżeli został, to bieżąca trasa jest sztucznie przedłużana przez zajezdnię po drodze.

A9	Przekaż klientów z bieżącej trasy R_i do drugiej trasy R_j i zakończ R_i
Legalność	• R_i jest całkowicie zakorkowana z bieżącej pozycji • R_j ma odpowiednio wolnego miejsca
Komentarz	• Klienci są dokładani do R_j w oryginalnej kolejności • Synergia z A8 • Synergia z ostatnią trasą rozwiązania początkowego (zwykle ma więcej miejsca)

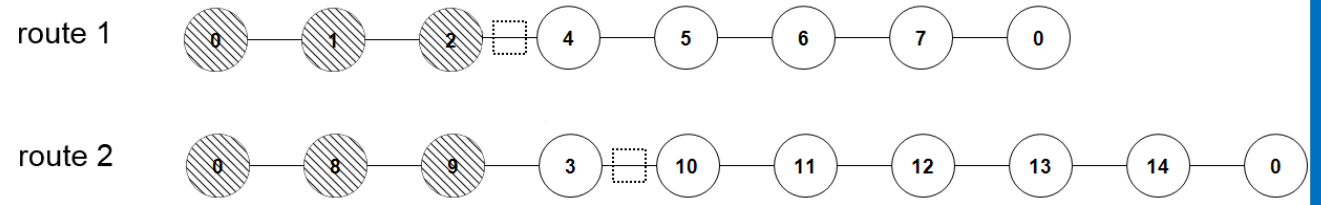
[1] $= \left\lceil \frac{1.4 * TotalDemand}{TruckCapacity} \right\rceil$, $\text{TotalDemand} = \sum_{i=1}^{N} (Demand_i)$

A10/A11	Wymiana klientów między trasami R_i i R_j w wersji MIN/MAX
Legalność	• R_i oraz R_j rozpoczynają się zakorkowanymi krawędziami • Po wykonaniu akcji, żadna trasa nie rozpoczyna się korkiem • Ograniczenia ładowności są zachowane dla obu tras
Komentarz	**Każda akcja w 4 wariantach (00, 01, 10, 11) gdzie bity oznaczają odwrócenie odpowiedniej trasy przed wykonaniem algorytmu wymiany.** **MIN** Wystartuj od oddania 1 wierzchołka z zakorkowanej trasy Jeżeli nie są przekroczone pojemności ciężarówek, to: **ZAKOŃCZ** Kontynuuj balansując pojemności – oddawaj wierzchołki z bardziej przepełnionej **MAX** Wystartuj od oddania 1 wierzchołka z zakorkowanej trasy Jeżeli nie są przekroczone pojemności ciężarówek, to: **ZAPAMIĘTAJ ROZWIĄZANIE** Kontynuuj balansując pojemności – oddawaj wierzchołki z bardziej przepełnionej

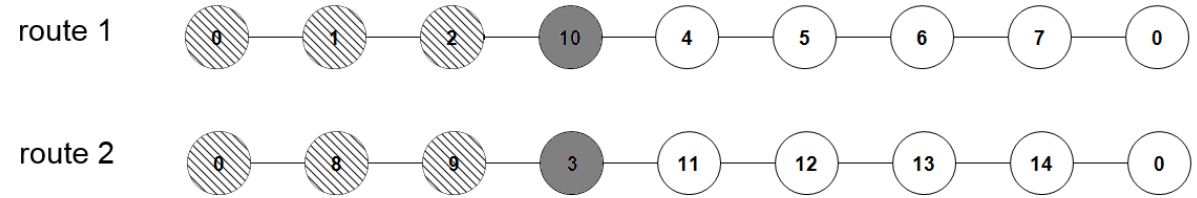
a) initial give away of the first customer from the jammed route



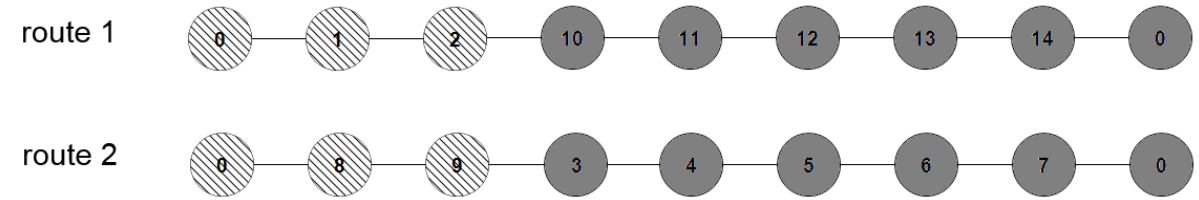
b) after the give away operation



c) possible result from the MIN version of the exchange
- stop as soon as the maximum truck capacity rule is not violated



d) possible result from the MAX version of the exchange
- find the last solution which does not violate the maximum truck capacity rule



Akcje wpływające na 2+1 trasy

A12	Zakończ trasy R_i i R_j i utwórz nową R_k
Legalność	• R_i jest całkowicie zakorkowana z bieżącej pozycji • Po wykonaniu akcji R_k nie jest zakorkowana • Suma wielkości zamówień z tras R_i i R_j nie przekracza pojemności ciężarówki • Limit ciężarówek nie jest wyczerpany (jak w przypadku akcji zakończenia 1 trasy)
Komentarz	**Akcja w 4 wariantach:** • $R_k = R_i + R_j$ • $R_k = R_j + R_i$ • $R_k = \text{Reverse}(R_i) + R_j$ • $R_k = \text{Reverse}(R_j) + R_i$ **+** konkatencja klientów bez doklejania zajezdni z pierwszej trasy

Trzy algorytmy genetyczne

Kodowanie osobników

Osobnik w populacji koduje pełne rozwiązanie K tras: **$[R_1, R_2, ..., R_K]$**

Trasa to wektor identyfikatorów oraz kilka dodatkowych pól np.

- **IDs** = [7, 5, 4, 9, 2, 0]
- **CurrentCity** = 0
- **CapacityLeft** = 3
- Pola pomocnicze *(AlreadyMutated, InstantJam)*

Populacja początkowa to **N** identycznych osobników utworzonych ze statycznego rozwiązania

- Homogeniczność możliwa ze względu na silną mutację

optymalizacja genetyczna zawsze w bieżącym (faktycznym) zakorkowaniu

wszystko dzieje się w fazie mutacji

mutacja tworzy klony – selekcja uwzględnia **N** najlepszych osobników nieważne czy rodziców czy dzieci

funkcja oceny to sumaryczny koszt tras przy aktualnym zakorkowaniu

```
while(MainSimulation.Ended == FALSE)
{
   MainSimulation.CalculateTraffic()
   for(generation from 0 to GEN)
   {
      OutputPopulation.Clear()

      for each (Gene in CurrentPopulation: gene)
      {
         gene.SynchronizeTrafficFrom (MainSimulation)

         if(random < mutationProbability)
         {
            Gene clone = Mutate(gene)
            OutputPopulaton.Add(clone)
         }
         else
         {
            OutputPopulation.Add(gene)
         }
      }
      for each (Gene in OutputPopulation: gene)
         UpdateFitness(gene)

      OutputPopulation.SortByFitness()
      CurrentPopulation = TOP N From OutputPopulation
   }
   Result = TOP 1 from OutputPopulation
   MainSimulation.AdvanceMovement(Result)
   for each (Gene in CurrentPopulation: gene)
      gene.AdvanceMovement()
}
```

Mutacja algorytm genetyczny 1

- Następuje iteracja po całej populacji i każdy osobnik ma prawdopodobieństwo **P = 0.75**, że kwalifikuje się do mutacji.

- Jeżeli mutacja ma nastąpić, to trasy danego osobnika są losowo permutowane. Następuje iteracja po trasach i każda trasa mutuje się osobno z prawdopodobieństwami odpowiednio **0.95** (korek) i **0.05** (droga wolna) w zależności czy jest na niej bezpośredni korek czy nie.

- Mutacja polega na losowym wybraniu legalnej akcji modyfikującej trasę – akcje są dokładnie takie same jak w algorytmie UCT. Każda legalna akcja jest równie prawdopodobna.

Dodawanie nowych tras

Na początku mutacji wyliczane jest ile potencjalnie nowych tras może być dodanych. Mutacja powodująca dodanie nowej trasy jest legalna w zależności od bieżącej liczby tras w osobniku oraz globalnego parametru liczby maksymalnych tras (MAX_TRUCKS)

Testowane wartości:

- MAX_TRUCKS = 0
- MAX_TRUCKS = 1
- MAX_TRUCKS = 2
- MAX_TRUCKS = 3
- MAX_TRUCKS = 4

Mutacja algorytm genetyczny 2

- Operacja mutacji działa na całym osobniku (rozwiązaniu) a nie na trasach.
- Maksymalna **liczba mutacji** osobnika w jednej iteracji AG jest równa **liczbie tras** w osobniku.

Niech:

|R| - oznacza liczbę tras.

|RT| - oznacza liczbę tras zaczynających się korkiem w bieżącym stanie.

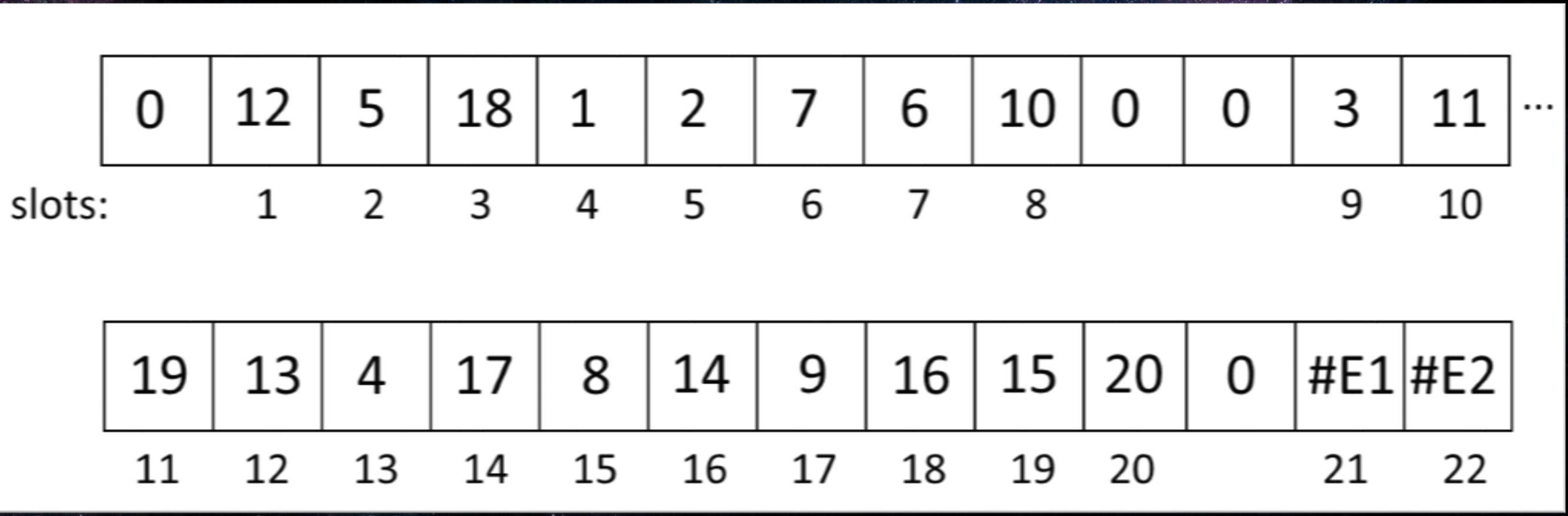
|RN| – oznacza liczbę tras niezaczynających się korkiem

- Oczywiście **|R| =| RT| + |RN|**

|RT| mutacji zachodzi z prawdopodobieństwem równym 0.95

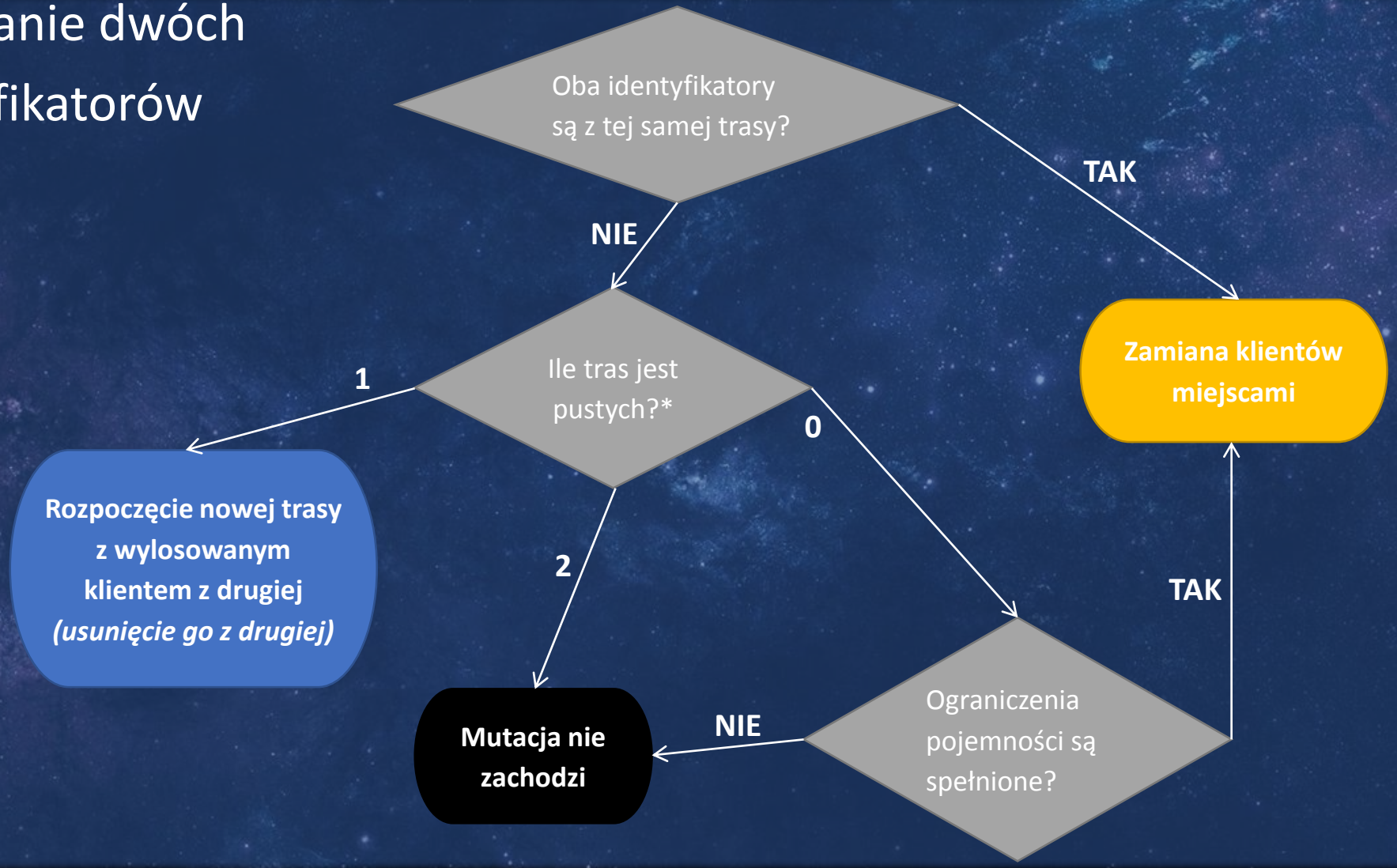
|RN| mutacji zachodzi z prawdopodobieństwem równym 0.05

Każde miejsce (slot) w rozwiązaniu ma przypisany unikalny identyfikator.
Identyfikatory są kolejnymi liczbami naturalnymi i oznaczają miejsca na trasach.
Identyfikatorów jest tyle ile klientów na trasach (zajezdnia jest wykluczona)+ każda pusta trasa ma jeden identyfikator ***(identyfikator wskazuje na miejsce klienta lub początek pustej trasy)***



Mutacja algorytm genetyczny 2

Losowanie dwóch identyfikatorów



Mutacja algorytm genetyczny 3

- Operacja mutacji działa na całym osobniku (rozwiązaniu) tak jak wersja #2, a nie na trasach tak jak wersja #1 algorytmu genetycznego.
- Mutacja zachodzi zawsze, jeżeli jest co najmniej 1 zakorkowana trasa.
- Losowana jest zakorkowana trasa **R1**
- Losowana jest dowolna inna trasa **R2** (uwzględniając parametr K - puste trasy potencjalne)

Mutacja algorytm genetyczny 3

- Przypadek A. Wylosowano R2 puste

  Następuje **lambda-interchange(1,0)** z utworzeniem nowej trasy
- Przypadek B. Wylosowano R2 niepuste, z prawdopodobieństwem 0.67

  Następuje **lambda-interchange(1,1)** dla tras R1 i R2
- Przypadek C. Wylosowano R2 niepuste, z prawdopodobieństwem 0.33

  Następuje **lambda-interchange(1,0)** dla tras R1 i R2

Mrówki

Algorytm mrówkowy

Zrealizowany w myśl klasycznego algorytmu rozwiązującego Problem Komiwojażera

Każda mrówka znajduje pełne rozwiązanie problemu, ale w główne symulacji bierzemy jedynie kolejne przejazdy z najlepszego rozwiązania (potem jest szansa na przeliczenie)

Feromon odkładany na krawędziach

Minimalna wartość feromonu:

f_min = 0.1 * maksymalna odległość między dwoma miastami

Algorytm mrówkowy

Inicjalizacja na samym początku:

- Jeśli krawędź należy do rozwiązania początkowego (z którego zaczyna również UCT), to otrzymuje początkowy feromon = **3*f_min**

Inicjalizacja przed każdym krokiem głównej symulacji (weryfikującej):

- Zerowane jest najlepsze znalezione rozwiązanie
- Feromon resetowany jest do wartości początkowej.

Inicjalizacja przed każdą iteracją mrówki:

- Pełny bieżacy stan (pozycje ciężarówek, pozostałe pojemności, nieodwiedzone miasta, zakorkowanie) jest synchronizowane ze stanem głównej symulacji.

```
while UnvisitedCities.Count > 0
{
   for each active route: R
   {
      viableCustomers[] = Customers WHERE Demand <= R.CapacityLeft

      if(exists a non-jammed transit from R.LastCustomer to any
      customer ∈ viableCustomers[] OR (new truck is not available))
      {
         PseudoRouletteSelection(viableCustomers[])
      }
      else
      {
         FinishRoute(R)
         StartNewActiveRoute()
      }
      AdvanceMovement() //move trucks
      RemoveFinishedRoutes()
   }
}
```

```
BestSolution = null
for iteration from 0 to MAX_I
{
   for ant 0 to MAX_A
   {
      ResetTruckPositions //to real truck positions
      ResetTraffic //to real traffic
      AntIteration(ant)
      solution = ant.GetSolution()
      if(solution better than BestSolution){
         BestSolution = solution }
      Update Pheromone
   }
}
return BestSolution
```

Algorytm mrówkowy

Metoda kontynuacji trasy:

Z prawdopodobieństwem równym **0.75** stosujemy ruletkę, w przeciwnym przypadku wybieramy najkrótszą krawędź zachłannie.

Ruletka:

Wybiera spośród legalnych miast (brak wizyty, nieprzekroczona ładowność):

$$p_{ij} = \frac{\tau_{ij}^{\alpha} * \eta_{ij}^{\beta}}{\sum\left(\tau_{ij}^{\alpha} * \eta_{ij}^{\beta}\right)}, \eta_{ij} = \frac{f\_min}{d_{ij}}$$

d_{ij}- bieżący koszt krawędzi

$\alpha = \beta = 1$

Algorytm mrówkowy

Uaktualnienie feromonu następuje po zakończonej iteracji:

$$\tau_{ij}(t+1) = \max(f_{min}, \frac{1}{2}\tau(t)_{ij} + \sum_{m \in Mrówki} (\delta_{ij} * Q_m))$$

δ_{ij} = 0 – *krawędź nie należy do rozwiązania danej mrówki*

δ_{ij} = 1 – *krawędź należy do rozwiązania, ale nie było ono globalnie najlepsze*

δ_{ij} = 10 – *krawędź należy do globalnie najlepszego rozwiązania (mrówka elitarna)*

$Q_m = \frac{f\_min}{D_m}$, D_m - oznacza sumę długości tras rozwiązania znalezionego przez mrówkę

Dwie proste heurystyki

H1

- Realizujemy rozwiązanie statyczne krok po kroku, z zastrzeżeniem, ze jeżeli w kolejnym kroku trafiamy na korek, to weryfikujemy 2-OPTem możliwość zamiany par krawędzi (w ekstremalnej sytuacji - gdy korek jest mały albo inne miasta są daleko położone - możemy tez przejść po krawędzi zakorkowanej).

H2

- Jeżeli mamy korek [X,X+1] to próbujemy przejść do następnego miasta [X,X+2]- o ile nie ma tam korka - natomiast do [X+1] wracamy w kolejnym kroku (o ile znów nie będzie korka).

Dla obu heurystyk istnieje **jednorazowa** reguła utworzenia nowej trasy, jeżeli bieżąca trasa jest w pełni zakorkowana.

Parametry

Benchmark	N	Mrówki max(100, 2*N)	Iteracje mrówkowe	Generacje genetyczne	Rozmiar populacji	Symulacji UCT per krok
P19-k2	19	100	200	100	200	30 000
E30-K4	30	100	200	100	200	30 000
P45-k5	45	100	200	150	200	30 000
E51-K5	51	102	200	150	200	30 000
A54-K7	54	108	200	150	200	30 000
A69-k9	69	138	100	150	200	30 000
E76-K7	76	152	75	150	200	30 000
E76-K15	76	152	75	150	200	30 000
A80-k10	80	160	75	150	200	30 000
P101-k4	101	202	75	150	200	30 000

Problem	P	UCT	Static	H1	H2	Ants	GA 1 K=1	GA 1 K=3	GA 1 K=4	GA 2 K=1	GA 2 K=3	GA 2 K=4	GA 3 K=1	GA 3 K=3	GA 3 K=4
A54	15	1653,2	6275,0	3929,0	3023,6	2794,1	**1645,1**	1667,0	1663,0	2511,5	2160,4	2155,6	2564,6	2345,8	2270,7
A54	2	1259,3	1939,2	1281,0	1372,1	1751,6	1233,4	1240,1	1237,1	1237,1	**1207,4**	1216,1	1393,7	1380,7	1381,9
A54	5	1353,4	3072,4	1951,4	1732,0	1984,8	1318,6	1316,6	**1311,6**	1429,9	1346,9	1333,3	1651,3	1608,3	1641,1
A69	15	**1718,0**	6631,7	4096,3	3211,4	3072,7	1843,3	1809,4	1915,7	2953,9	2520,5	2672,0	3066,3	2783,7	2679,6
A69	2	1266,2	2005,7	1484,0	1381,1	1833,1	1255,4	1265,2	1272,6	1207,0	1189,7	**1185,9**	1480,2	1467,4	1471,6
A69	5	1393,4	3235,4	2039,8	1751,6	2064,9	1379,0	1397,5	**1376,5**	1525,9	1425,5	1457,5	1952,4	1928,2	1906,9
A80	15	**2624,5**	9066,5	5048,1	4254,5	4084,1	2695,5	2632,9	2626,9	4163,7	3511,1	3690,2	4303,4	3787,2	3818,3
A80	2	1947,4	2774,1	2224,2	2151,9	2347,8	1904,2	1908,7	1901,0	1799,7	1830,7	**1766,1**	2204,4	2253,5	2262,1
A80	5	2078,6	4100,6	2708,6	2453,3	2605,1	2022,7	**2011,7**	2022,1	2115,4	2072,7	2038,0	2730,6	2736,2	2794,4
P101	15	**1196,6**	5419,6	2141,7	1518,5	1638,5	1413,8	1418,5	1432,9	3746,2	3687,4	3711,8	2845,6	2750,9	2787,3
P101	2	**814,3**	1436,5	851,5	844,4	1086,6	873,4	874,2	872,4	1080,0	1011,6	1053,5	1089,9	1041,0	1054,5
P101	5	**901,3**	2552,0	1086,5	977,2	1144,9	1028,5	1009,6	1006,6	1761,8	1616,5	1728,2	1782,8	1689,4	1610,8
P19	15	351,9	1278,0	792,7	554,7	591,0	**312,5**	319,0	319,0	487,3	390,2	447,8	804,6	596,5	561,8
P19	2	247,0	388,9	258,2	253,4	291,7	237,1	237,2	237,2	248,1	229,5	**228,8**	264,9	259,4	260,8
P19	5	269,3	612,0	369,4	312,2	319,1	259,3	**256,8**	**256,8**	293,3	264,0	266,0	387,6	312,0	311,7
P45	15	**783,2**	3299,5	2262,9	1444,4	1296,5	872,3	878,1	876,3	1301,2	1171,1	1148,4	1229,3	1255,3	1238,9
P45	2	602,1	1007,7	699,0	647,7	772,3	636,7	632,5	632,5	602,1	**561,6**	575,9	679,8	679,8	674,8
P45	5	**647,0**	1759,6	969,7	780,7	858,8	693,5	689,0	690,9	736,5	674,0	659,0	886,1	883,1	882,8
E30-K4	15	823,2	2678,6	1933,4	1264,2	1216,9	846,5	**807,4**	834,8	1221,0	1105,2	978,7	1350,7	1350,9	1298,8
E30-K4	2	565,7	870,8	622,6	581,0	851,0	562,7	557,0	555,7	539,2	457,7	**442,7**	606,6	601,9	590,1
E30-K4	5	611,9	1330,2	868,7	676,8	941,6	591,5	581,5	601,1	621,8	**541,5**	549,2	814,4	821,6	798,5
E51-K5	15	**844,5**	3509,7	2107,4	1573,3	1359,7	972,8	981,2	938,5	1434,8	1292,4	1303,7	1256,6	1316,4	1275,0
E51-K5	2	616,2	989,2	720,7	684,9	892,9	651,4	639,1	641,9	637,9	598,6	**572,8**	690,1	694,6	683,7
E51-K5	5	**670,0**	1571,6	946,2	805,0	912,5	700,8	690,2	702,9	815,9	691,2	686,6	852,3	864,1	837,6
E76-K15	15	**1478,3**	5027,5	3392,7	2594,7	2795,9	1732,1	1676,3	1650,6	2267,6	2228,3	2148,7	1909,5	1939,0	1904,2
E76-K15	2	1114,2	1706,3	1311,7	1199,5	1370,2	1137,9	1113,6	1129,2	1081,6	1025,4	**1007,2**	1218,8	1242,4	1220,0
E76-K15	5	1199,1	2484,6	1791,7	1536,9	1546,0	1212,1	1191,2	**1171,7**	1237,7	1208,3	1178,2	1462,6	1438,4	1429,8
E76-K7	15	**1095,0**	4536,7	2569,5	1860,7	1608,2	1270,7	1204,2	1210,4	2386,0	2280,3	2187,4	2021,3	1861,0	1972,3
E76-K7	2	779,2	1318,7	911,0	852,1	1058,7	832,1	805,6	806,0	819,7	782,9	**750,3**	952,4	934,1	918,8
E76-K7	5	**835,9**	2130,0	1107,2	1005,8	1177,8	935,7	897,5	912,9	1086,5	1070,3	1001,0	1307,9	1266,8	1278,3
UCT / [x]		**1,00 (12)**	**0,44**	**0,69**	**0,79**	**0,71**	**0,97 (2)**	**0,98 (3)**	**0,97 (4)**	**0,82**	**0,88 (3)**	**0,89 (7)**	**0,73**	**0,75**	**0,76**

Problem	P	UCT (σ)	Static (σ)	H1 (σ)	H2 (σ)	Ants (σ)	GA 1 K=3 (σ)	GA 1 K=4 (σ)	GA 2 K=3 (σ)	GA 2 K=4 (σ)	GA 3 K=3 (σ)	GA 3 K=4 (σ)
A54	15	**143,2**	1441,9	924,4	909,7	854,4	264,6	195,0	301,3	289,0	369,7	250,6
A54	2	**47,3**	542,7	311,3	221,5	106,7	64,6	51,0	57,0	50,6	137,4	176,8
A54	5	83,9	887,7	495,0	449,4	423,8	69,9	**58,7**	110,6	102,1	142,0	211,1
A69	15	**136,4**	1170,9	879,3	669,5	927,2	260,1	327,7	352,8	430,4	424,1	423,6
A69	2	**41,7**	531,8	334,0	179,8	123,1	70,1	55,0	76,5	72,7	166,2	204,4
A69	5	113,6	644,1	484,1	342,6	367,2	**81,7**	88,1	144,5	169,4	296,6	286,8
A80	15	**153,9**	1437,4	748,7	811,4	1062,8	319,4	310,0	490,4	539,2	456,4	476,7
A80	2	72,1	625,2	311,7	301,9	257,5	**60,3**	63,2	90,9	103,8	277,7	319,1
A80	5	142,9	830,1	447,2	449,2	538,8	**76,2**	77,1	152,4	233,2	355,2	418,9
P101	15	**73,4**	801,5	478,3	256,2	469,8	155,7	119,5	640,6	498,1	460,6	393,2
P101	2	**24,9**	262,6	66,9	70,2	118,7	44,5	45,8	100,7	126,8	145,9	183,5
P101	5	**54,9**	547,3	195,7	117,7	139,0	64,2	60,2	225,6	212,2	261,5	241,1
P19	15	81,4	358,9	362,7	227,7	336,8	**57,4**	**57,4**	147,4	166,1	356,7	342,8
P19	2	**14,5**	214,9	40,8	33,7	26,0	15,2	15,2	29,6	19,2	35,2	39,1
P19	5	23,8	213,3	136,4	86,7	101,9	**22,3**	**22,3**	57,2	71,7	85,9	78,7
P45	15	**56,1**	733,3	632,1	357,9	408,6	129,3	130,0	239,3	219,9	291,0	219,6
P45	2	**21,5**	326,2	124,5	86,0	59,2	34,1	34,1	41,5	57,8	102,4	98,7
P45	5	**36,8**	411,9	260,0	172,2	180,7	49,8	51,1	80,4	70,2	199,9	167,8
E30-K4	15	**208,8**	705,7	615,2	484,6	423,6	275,0	293,1	445,8	375,4	569,2	383,7
E30-K4	2	81,9	369,8	162,6	108,8	100,8	92,1	81,1	49,7	**48,5**	108,1	100,3
E30-K4	5	107,3	459,9	316,4	189,4	219,6	104,3	109,2	**82,4**	157,7	248,9	203,4
E51-K5	15	**77,9**	688,9	565,1	452,9	400,9	168,8	150,6	224,4	274,9	311,8	248,5
E51-K5	2	**23,5**	240,6	106,2	98,1	115,5	40,5	29,8	52,2	36,0	98,7	89,0
E51-K5	5	**42,4**	386,3	288,9	143,9	122,0	54,2	64,7	68,6	74,7	144,1	125,6
E76-K15	15	**108,0**	824,7	592,6	503,6	853,2	245,3	248,2	413,3	327,6	230,9	202,0
E76-K15	2	**32,5**	382,7	177,5	112,0	139,8	34,9	40,4	62,8	56,0	119,9	109,8
E76-K15	5	54,7	533,9	289,3	260,8	345,8	68,1	**45,7**	116,1	123,7	172,5	166,0
E76-K7	15	**81,8**	838,0	562,4	461,0	470,5	167,9	159,8	452,1	448,8	188,8	470,7
E76-K7	2	**33,5**	295,5	142,6	114,0	67,6	45,8	40,5	67,7	49,1	163,3	122,8
E76-K7	5	**40,9**	460,4	250,9	187,8	296,6	54,6	63,8	162,9	140,8	243,2	216,9
Najmniejsze σ		**21**	**0**	**0**	**0**	**0**	**5**	**4**	**1**	**1**	**0**	**0**

Podsumowanie

Metoda UCT jest dobrym narzędziem do połączenia z algorytmem statycznym

- *Można testować różne modyfikacje tras w zależności od warunków zakorkowania*
- *Do samodzielnego wygenerowania rozwiązania się nie nadaje*

Podobne rezultaty algorytmu genetycznego #1 oraz UCT

- Można rozważyć dodanie operatora krzyżowania

Wszelkie pozostałe metody wyraźnie odstają

- UCT jest tak dobry lub pozostałe metody niedostatecznie dobre