

Przemysł 4.0 - Laboratorium  
VRP – algorytmy metaheurystyczne

prowadzący: mgr inż. Radosław Idzikowski

---

## 1 Wprowadzenie

Celem laboratorium jest zapoznanie się z algorytmami metaheurystycznymi i implementacją wybranego dla problemu marszrutyzacji pojazdów. Cechą algorytmów metaheurystycznych jest możliwość stosowania ich do rozwiązywania różnych problemów. Przedstawione poniżej algorytmy: symulowane wyżarzanie (*simulated annealing*) i przeszukiwanie z zakazami (*Tabu Search*) w zaproponowanej formie można zaimplementować dla dowolnego problemu permutacyjnego.

## 2 Symulowane wyżarzanie

Algorytm metaheurystyczny o charakterze probalistycznym inspirowany procesem metalurgicznym. Na początku należy przyjąć temperaturę początkową  $T_0 > 0$  co będzie symbolizowane przez podgrzanie metalu czyli naszego rozwiązania. Jako rozwiązanie początkowe możemy przyjąć: (1) permutację naturalną, (2) losową, (3) wynik alg. zachłannego. Warunkiem zatrzymania algorytmu jest spadek temperatury  $T$  naszego rozwiązania poniżej temperatury końcowej  $T_{end}$ . Dla każdej temperatury będziemy wykonywać  $L$  iteracji wewnętrznych (epok). W każdej epoce należy wykonać jeden pojedynczy losowy ruch:

- zamień (*swap*),
- wstaw (*insert*),
- odwróć (*twist/reverse*),
- zamień sąsiedni (*adjacent swap*).

Przez całe działanie algorytmu wykonujemy jeden typ ruchu (dopuszczalne są ruchy hybrydowe). Jeśli Rozwiązanie po wykonaniu ruchu jest lepsze niż aktualne, należy je zaakceptować (podmienić), w przeciwnym wypadku z pewnym prawdopodobieństwem  $p$  również możemy je zaakceptować:

$$p = e^{\frac{\Delta C_{\max}}{T}} \quad (1)$$

gdzie

$$\Delta C_{\max} = C_{\max}(\pi) - C_{\max}(\pi_{\text{new}}) \quad (2)$$

Prawdopodobieństwo akceptacji gorszego rozwiązania będzie malało wraz ze spadkiem temperatury oraz przy bardzo dużej różnicy wartości funkcji celu. W procesie metalurgicznym gorący materiał jest bardziej podatny na zmiany niż zimny. Idea algorytmu polega na przeglądaniu początkowo jak największego sąsiedztwa, również przechodząc przez gorsze rozwiązania, aby na końcu zbiec do któregoś minimum lokalnego akceptując już tylko lepsze rozwiązania. Po wykonaniu wszystkich iteracji wewnętrznych (epok) należy obniżyć temperaturę według ustalanego schematu chłodzenia:

- liniowego  $T' = T - x$ ,
- geometrycznego  $T' = \alpha T$ ,
- logarytmicznego  $T' = \frac{T}{\ln(it+1)}$ , gdzie  $it$  – nr iteracji algorytmu.

Alternatywnie algorytm można przerwać w momencie niezaakceptowania ani jednego rozwiązania podczas po wykonaniu epok.

Parametry zaczerpnięte z literatury oraz doświadczenia:

parametr	wartości		
$T_0$	$10^2$	$10^3$	$10^4$
L	$\sqrt{n}$	$n$	$n^2$
x	$\frac{T_0}{10^3}$	$\frac{T_0}{10^4}$	$\frac{T_0}{10^5}$
$\alpha$	0.97	0.95	0.90

---

**Algorithm 1** simulated annealing algorithm

---

```

1:  $T \leftarrow T_0$ 
2:  $\pi \leftarrow \text{INITSOLUTION}()$ 
3: while  $T > T_{\text{end}}$  do
4:   for  $k = 1$  to  $L$  do
5:      $i \leftarrow \text{RANDOMINT}(1, n)$ 
6:      $j \leftarrow \text{RANDOMINT}(1, n)$ 
7:      $\pi_{\text{new}} \leftarrow \pi.\text{MOVE}(i, j)$ 
8:     if  $\text{CALCULATE}(\pi_{\text{new}}) > \text{CALCULATE}(\pi)$  then
9:        $r \leftarrow \text{RANDOMDOUBLE}(0, 1)$ 
10:      if  $r \geq e^{-\frac{\Delta C_{\text{max}}}{T}}$  then
11:         $\pi_{\text{new}} \leftarrow \pi$ 
12:      end if
13:    end if
14:     $\pi \leftarrow \pi_{\text{new}}$ 
15:    if  $\text{CALCULATE}(\pi) < \text{CALCULATE}(\pi^*)$  then
16:       $\pi^* \leftarrow \pi$ 
17:    end if
18:  end for
19:   $T \leftarrow \text{REDUCETEMPERATURE}(T)$ 
20: end while

```

---

### 3 Przeszukiwanie z zakazami

Algorytm metaheurystyczny możemy pozbawić losowości jeśli jako rozwiązanie początkowe przyjmujemy permutację naturalną lub wynik algorytmu zachłannego zamiast rozwiązania losowego. Warunkiem stopu algorytmu jest liczba iteracji lub limit czasowy. Algorytm sprawdzi całe sąsiedztwo wybranego typu i wybiera najlepszego sąsiada o ile nie znajduje się na liście tabu (można wprowadzić mechanizm przełamania listy tabu jeśli sąsiad jest znacząco lepszy od najlepszego rozwiązania znalezionej do tej pory). Algorytm zawsze musi przejść do któregoś sąsiada, nawet jeśli jest gorszy niż aktualne rozwiązanie. Następnie ruch jest zabraniany na zadaną liczbę iteracji (kadencja). Algorytm możemy modyfikować zależnie od celu jaki chcemy uzyskać. Algorytm w trakcie działania może zmieniać kadencję: (1) zmniejszać, aby dokładniej przeszukiwać sąsiedztwo w okolicach optimum lokalnego (intensyfikacja) lub (2) zwiększać, aby wymusić szybsze wyjście z optimum lokalnego i przejść w inny obszar przeszukiwań (dywersyfikacja). Innym narzędziem dywersyfikacji jest metoda zdarzeń krytycznych, która wymusza przelosowanie aktualnego rozwiązania po zadanej liczbie iteracji bez poprawy.

---

**Algorithm 2** Tabu Search

---

```
1:  $\pi \leftarrow \text{INITSOLUTION}()$ 
2:  $\text{tabuList.CLEAR}()$ ;
3: for  $it = 1$  to  $itLimit$  do
4:    $C_{best} \leftarrow \infty$ 
5:   for  $j = 1$  to  $n$  do
6:     for  $k = j + 1$  to  $n$  do
7:       if  $\text{tabuList}[j, k] < it$  then
8:          $\pi_{\text{new}} \leftarrow \pi.\text{MOVE}(i, j)$ 
9:         if  $\text{CALCULATE}(\pi_{\text{new}}) < C_{best}$  then
10:           $C_{best} \leftarrow \text{CALCULATE}(\pi_{\text{new}})$ 
11:           $j^* \leftarrow j$ 
12:           $k^* \leftarrow k$ 
13:        end if
14:      end if
15:    end for
16:  end for
17:   $\pi \leftarrow \pi.\text{MOVE}(j^*, k^*)$ 
18:   $\text{tabuList}[j^*, k^*] \leftarrow it + cadance$ 
19:  if  $\text{CALCULATE}(\pi) < \text{CALCULATE}(\pi^*)$  then
20:     $\pi^* \leftarrow \pi$ 
21:  end if
22: end for
```

---