Wrocław, 6 kwietnia 2022

Algorytmy **M**etaheurystyczne

Laboratorium 2

Algorytmy lokalnego poszukiwania

prowadzący: *dr inż. A. Gnatowski, dr inż. R. Idzikowski, dr inż. J. Rudy*

Algorytmy przeszukiwania lokalnego opierają się na eksplorowaniu zbioru rozwiązań poprzez iteracyjne modyfikowanie pewnego rozwiązania (rozwiązania bieżącego). W każdym kroku generowane jest *otoczenie* bieżącego rozwiązania, t.j. zbiór podobnych rozwiązań. Z otoczenia wybierane jest najbardziej obiecujące rozwiązanie, które staje się nowym bieżącym rozwiązaniem. Algorytm przeszukiwania z zabronieniami[1] (z ang. *Tabu Search*, TS) rozszerza ten schemat przez dodanie mechanizmu zabronień (listy tabu), którego celem jest umożliwienie opuszczania ekstremów lokalnych oraz wpadania algorytmu w cykle. Za każdym razem kiedy rozwiązanie jest wybierane z otoczenia, jego reprezentacja jest umieszczana na liście tabu. Ponieważ zabronione jest wybieranie rozwiązania którego reprezentacja znajduje się na liście tabu, algorytm może opuścić ekstremum lokalne. Uproszczony schemat TS pokazano na rysunku 1.

Krok 1 Pierwszym krokiem algorytmu TS (i algorytmów przeszukiwania lokalnego w ogóle), jest wygenerowanie rozwiązania początkowego. Do typowych podejść należą: generowanie rozwiązania losowego oraz wykorzystanie rezultatu heurystyki. Metoda generowania rozwiązania początkowego może mieć duży wpływ na efektywność algorytmu metaheurystycznego, szczególne jeżeli budżet obliczeniowy jest bardzo ograniczony. Przykłady metod generowania rozwiązania początkowego dla TSP:

1. rozwiązanie losowe;
2. heurystyka, taka jak 2-Opt lub algorytm LK[2] (jedna z najlepszych heurystyk dla symetrycznego TSP).

Krok 2 Kolejnym krokiem algorytmu jest wygenerowanie otoczenia rozwiązania π, zbioru $N(\pi)$. Metoda konstruowania otoczenia ma kluczowy wpływ na efektywność TS. Jednym z popularniejszych, prostych otoczeń dla TSP, jest otoczenie typu *invert*, omówione przy okazji algorytmu 2-Opt. Do mniej efektywnych ruchów (w kontekście TSP), często stosowanych do generowania otoczeń dla innych problemów, należą ruchy *swap* lub *insert*. Ruch *swap* polega na zamianie kolejności dwóch elementów permutacji. Na przykład dla

$$\pi = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10), \tag{1}$$

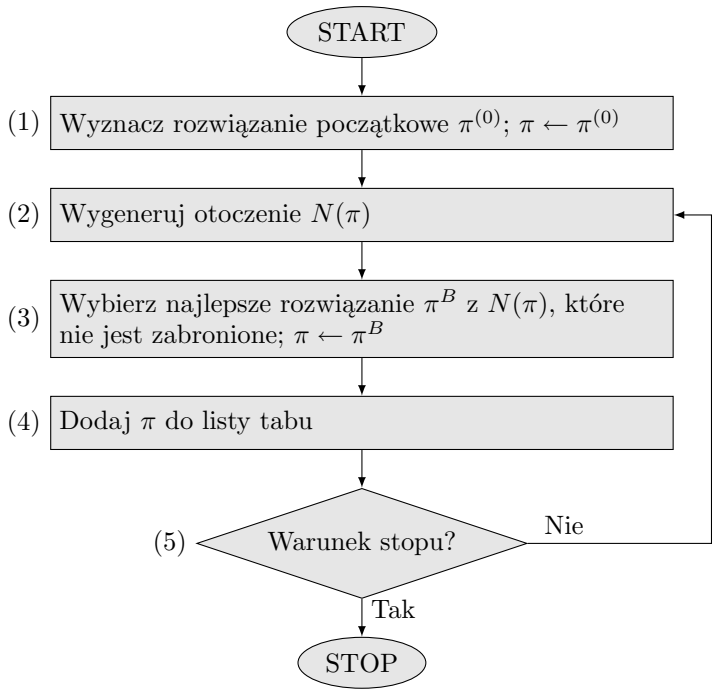
ruch $\text{swap}(\pi, 4, 7)$ generuje

$$\pi' = (1, 2, 3, 7, 5, 6, 4, 8, 9, 10). \tag{2}$$

W praktyce często stosuje się zwielokrotnione ruchy (tzw. k-opt) oraz różnego rodzaju akceleracje, które realizują kroki 2 i 3 w sposób łączny, w czasie krótszym niż wynikałoby to z podejścia naiwnego. Na przykład, wyznaczenie wartości funkcji celu dla każdego elementu otoczenia oddzielnie, wygenerowanego za pomocą ruchu swap, może zająć $O(n^2 \cdot n)$ czasu. Natomiast, przy zastosowaniu akceleracji, ewaluacja może zająć $O(n^2)$ czasu.

Krok 3 Kolejnym krokiem algorytmu jest wybór obiecującego rozwiązania z $N(\pi)$. Rozwiązaniem takim jest rozwiązanie o najniższej wartości funkcji celu, a które jednocześnie nie jest reprezentowane na liście zabronień (tabu). Lista tabu to lista o ograniczonej długości, na którą trafiają reprezentacje rozwiązań które były wybierane jako bieżące rozwiązania w poprzednich iteracjach. Zwykle reprezentacją rozwiązania jest para indeksów definiujących ruch który został wykorzystany do jego wygenerowania. Na przykład, rozwiązanie π' z równania (2) może być reprezentowane na liście tabu przez nieuporządkowaną parę $\{4, 7\}$, ponieważ zostało utworzone przez ruch $\text{swap}(\pi, 4, 7)$. Często stosującą modyfikacją zasady zabronienia jest tak zwane kryterium aspiracji. Umożliwia

[1] Glover, F., Laguna, M., jul 1997. Tabu Search. Kluwer Academic Publishers.
[2] `https://doi.org/10.1287/opre.21.2.498`



Rysunek 1: Uproszczony schemat algorytmu przeszukiwania z zabronieniami

ono wybranie rozwiązania z otoczenia nawet jeżeli ma ono reprezentację na liście tabu (jest zabronione), jeżeli wartość funkcji celu tego rozwiązania jest lepsza od najlepszego dotychczas rozważanego rozwiązania. Jeżeli wszystkie rozwiązania w otoczeniu są zabronione, istnieje kilka możliwości dalszego postępowania, na przykład:

1. Zakończenie działania algorytmu.
2. Stopniowe odrzucanie najstarszych reprezentacji z listy tabu tak długo aż co najmniej jedno rozwiązanie nie jest zabronione.
3. Dokonanie restartu TS z innego rozwiązania początkowego.
4. Przejście do rozwiązania z listy nawrotów, czyli listy obiecujących rozwiązań tworzonych w trakcie działania algorytmu.
5. Tymczasowe skorzystanie z innego, szerszego otoczenia.

Krok 4 Kolejnym krokiem algorytmu jest umieszczenie na liście tabu reprezentacji tego rozwiązania. Jeżeli lista tabu przekroczy dozwoloną długość, usuwana jest najstarsza reprezentacja. Maksymalna długość listy tabu to istotny hiperparametr TS i jego wartość ma znaczący wpływ na efektywność działania metaheurystyki.

Krok 5 Po zakończeniu każdej iteracji, sprawdzany jest warunek stopu, czyli warunek zakończenia działania algorytmu. Warunki mogą być definiowane w różny sposób:

1. Osiągnięcie maksymalnej zakładanej liczby wykonanych iteracji.
2. Zakończenie czasu przewidzianego na obliczenia.
3. Osiągnięcie limitu wywołań funkcji celu (jeżeli nie jest stosowana akceleracja).
4. Liczba iteracji bez znalezienia nowego najlepszego rozwiązania.

Ostatecznie, rezultatem działania algorytmu jest najlepsze znalezione w trakcie działania algorytmu rozwiązanie.

Decyzje projektowe Poniżej znajduje się lista najważniejszych elementów metody TS co do których należy podjąć decyzję dotyczącą sposobu ich implementacji (elementy obowiązkowe) lub czy w ogóle będą zaimplementowane (elementy opcjonalne):

1. Implementacja deterministyczna vs implementacja probabilistyczna.
2. Wyboru rozwiązania początkowego.
3. Definicji sąsiedztwa (ruchu).
4. Sposób przeglądu sąsiedztwa.
5. Struktura, długość i sposób obsługi pamięci krótkoterminowej (lista tabu).
6. Pamięć długoterminowa.
7. Wykrywanie cykli/stagnacji + mechanizm resetów/powrotów.
8. Warunek stopu algorytmu.
9. Optymalizacje kodu.
10. Wykorzystanie obliczeń równoległych.