Zaawansowane programowanie

wykład 2: metoda przeszukiwania tabu

prof. dr hab. inż. **Marta Kasprzak**
Instytut Informatyki, Politechnika Poznańska

Metoda przeszukiwania tabu

[Fred Glover: Tabu Search — Part I, *ORSA Journal on Computing* 1 (1989) 190–206; Tabu Search — Part II, *ORSA Journal on Computing* 2 (1990) 4–32]

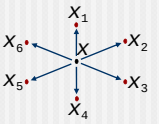
- *Metoda przeszukiwania tabu* (ang. *tabu search*) jest metaheurystyką realizującą heurystyczną procedurę przeszukiwania lokalnego, wychodzącą jednak poza obszar lokalnego optimum
- W tym podejściu *reguły deterministyczne* są preferowane nad probabilistycznymi
- Metoda przeszukiwania tabu operuje na *pojedynczym rozwiązaniu*, które iteracyjnie jest poprawiane pod kątem optymalizacji funkcji celu

Rozwiązanie początkowe

- *Bieżące rozwiązanie*, na którym dokonywane są operacje w metodzie, jest — niekoniecznie liniowym — zapisem rozwiązania problemu. Może to być np. ciąg indeksów elementów, dwuwymiarowa lista następników elementów, kwadratowa macierz sąsiedztwa grafu, itp. Zazwyczaj im większa struktura, tym większe zużycie pamięci i czasu w algorytmie
- *Rozwiązanie początkowe*, od którego rozpoczynane są obliczenia, może być losowe, choć zwykle lepszy efekt uzyskuje się generując je wstępną heurystyką (np. zachłanną)
- W razie stosowania restartów procedury przeszukiwania, kolejne rozwiązanie „początkowe” można wygenerować w oparciu o dotychczasowe obserwacje. Przykładowo, można je złożyć z najrzadziej używanych dotąd elementów (aby wystartować z innego obszaru przestrzeni rozwiązań) albo korzystać z fragmentów najlepszych (zróżnicowanych) rozwiązań

Sąsiedztwo

- Każde rozwiązanie $x \in X$ *przestrzeni rozwiązań* (ang. *solution space*) ma przypisane *sąsiedztwo* (ang. *neighborhood*) $N(x) \subset X$, którego każdy element $x' \in N(x)$ jest osiągany z x przez operację zwaną *ruchem* (ang. *move*)
- Ruch ma na celu zwykle niedużą zmianę w obrębie rozwiązania. Może być np. wstawieniem, usunięciem, przesunięciem elementu rozwiązania, zamianą miejscami pary elementów, itp.
- X stanowi zazwyczaj zbiór rozwiązań dopuszczalnych (gdy ruchy są ograniczone do dopuszczalnych), ale można stosować wyjątki



x, x_i — *elementy przestrzeni rozwiązań*

$N(x) = \{x_1, x_2, x_3, x_4, x_5, x_6\}$

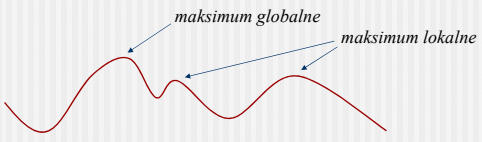
— *ruch*

Sąsiedztwo

- Zastosowanie wszystkich zdefiniowanych ruchów w stosunku do bieżącego rozwiązania może zaowocować zbyt dużym sąsiedztwem. Przegląd takiego sąsiedztwa wraz z wyliczeniem wartości funkcji celu może zająć zbyt wiele czasu
- Sąsiedztwo można ograniczyć do krótszej listy kandydatów (ang. *candidate list*), biorąc pod uwagę jedynie najbardziej obiecujące ruchy
- Przykładowo, można wybierać ruchy zapamiętane jako najskuteczniejsze w przeszłości lub ruchy odnoszące się do najsłabszych fragmentów rozwiązania
- Skrócenie obliczeń uzyskuje się także poprzez ewaluację funkcji oceny prostszej niż funkcja celu dla całego rozwiązania, np. pewnego jej przybliżenia albo przez oszacowanie względnej wartości lokalnej zmiany wywołanej przez ruch

Sąsiedztwo

- Zazwyczaj wybierany jest taki ruch, który prowadzi do największej poprawy wartości funkcji celu (ew. jej przybliżenia) lub do najmniejszego pogorszenia jej wartości, gdy poprawa w obrębie sąsiedztwa nie jest możliwa
- Ta zasada dążenia do optimum lokalnego jest elementem strategii *intensyfikacji* w metodzie

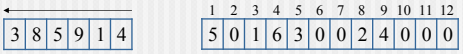


Lista tabu

- Po osiągnięciu w wyniku serii ruchów lokalnego optimum kolejny ruch może spowodować jedynie pogorszenie bieżącego rozwiązania (chwilowe). Zgodnie z przyjętą zasadą wybierany jest ruch skutkujący najmniejszym pogorszeniem
- Bardzo łatwo w ten sposób cofnąć się do obszaru już przeszukanego. W konsekwencji można zapętlić poszukiwania w obrębie stale tego samego obszaru
- Strukturą uniemożliwiającą powrót do ostatnio odwiedzonych punktów przestrzeni rozwiązań jest *lista tabu* (ang. *tabu list*). Zamieszczone na niej ostatnio osiągnięte fragmenty rozwiązania czy ostatnio wykonane ruchy blokują ich ponowne osiągnięcie/wykonanie przez zadaną liczbę iteracji
- Dopuszczalne sąsiedztwo jest teraz zdefiniowane jako $N^*(x) = N(x) \setminus T$, gdzie T to rozwiązania ze statusem tabu

Lista tabu

- Listę tabu realizuje się najczęściej jako listę/tablicę typu FIFO (ang. *first-in-first-out)*. Zawiera ona wszystkie bieżące atrybuty o statusie tabu
- Atrybut najczęściej opisuje ruch (np. wstawienie czy usunięcie elementu), także fragment rozwiązania
- Inną postacią listy tabu jest tablica o liczbie pól równej liczbie atrybutów, każde jej pole zawiera wartość równą liczbie iteracji, przez którą dany atrybut będzie posiadał status tabu. Tablica taka zajmuje więcej pamięci niż tradycyjna lista tabu, ale za to umożliwia sprawdzenie statusu atrybutu w stałym czasie



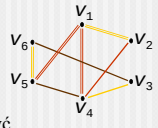
lista FIFO *tablica wszystkich atrybutów*

Lista tabu

- Zbyt krótka lista tabu może spowodować cykliczne powracanie do już wygenerowanych rozwiązań. Zbyt długa może zablokować zbyt wiele ruchów
- W ogólności krótkie listy tabu skutkują przeszukiwaniem okolicy lokalnego optimum, długie umożliwiają wyjście poza tę okolicę. Długość listy można zmieniać w trakcie działania algorytmu w zależności od przyjętej w danej chwili strategii intensyfikacji czy dywersyfikacji
- Długość listy tabu jest często stała w trakcie działania algorytmu. Długość najwłaściwszą dla danego problemu i danego rozmiaru instancji można w pewnym stopniu oszacować teoretycznie, lecz zazwyczaj na jej dobór wpływa seria wstępnych testów
- Może być więcej niż jedna lista tabu w algorytmie, wtedy pamiętają one różnego rodzaju atrybuty

Lista tabu — przykład

- *Minimum k-tree problem*: poszukiwanie w grafie nieskierowanym drzewa złożonego z k krawędzi takiego, że suma wag tych krawędzi jest minimalna
- Ruch zdefiniowany jest jako równoczesne wstawienie do rozwiązania jednej krawędzi i usunięcie innej
- Można przykładowo nadać status tabu na tę samą liczbę iteracji obu operacjom: dla krawędzi wstawianej będzie to zakaz usuwania, dla usuwanej — wstawiania
- Ponieważ jednak krawędzi w grafie mamy zazwyczaj dużo więcej niż k, warto rozważyć implementację dłuższej listy tabu dla krawędzi usuwanych niż wstawianych



k=4

wagi: 1, 2, 3

Kryterium aspiracji

- Wybierany ruch (lub jego konsekwencje) nie powinien zawierać atrybutów znajdujących się na liście tabu. Nie zawsze jednak można lub należy zachować tę prawidłowość. Zasadę pozwalającą pominąć status tabu nazywa się *kryterium aspiracji* (ang. *aspiration criterion*)
- Przykłady kryterium aspiracji:
  - ruch ze statusem tabu może być wybrany, gdy prowadzi do poprawy najlepszego rozwiązania (najlepszej wartości funkcji celu) otrzymanego do tej pory
  - gdy w pewnym momencie obliczeń sąsiedztwo danego rozwiązania okaże się zbyt małe, lub lista tabu zbyt długa, wszystkie możliwe ruchy będą miały przydzielony status tabu; wykonuje się wtedy ruch z początku listy, tzn. o najmniejszym okresie oczekiwania na opuszczenie listy

Kryterium aspiracji

- Przykłady kryterium aspiracji cd.:
  - gdy jesteśmy na etapie wychodzenia z obszaru lokalnego optimum, statusu tabu można pozbawić ruch o znaczącym wpływie na postać bieżącego rozwiązania, powodujący skok w odległy punkt przestrzeni rozwiązań
  - można także usunąć status tabu z ruchu o małym wpływie na postać rozwiązania, jeśli od czasu umieszczenia go (lub jego atrybutów) na liście tabu wykonany został skok do innego obszaru przestrzeni rozwiązań
  - gdy w trakcie stałego poprawiania jakości rozwiązania pewien ruch otrzyma status tabu i późniejsze jego wykonanie nie zakłóci tendencji wzrostu, to można go wykonać, gdyż nie spowoduje powrotu do ostatnio otrzymanych rozwiązań

Dywersyfikacja rozwiązań

- Strategia *dywersyfikacji* realizowana jest poprzez czynności mające na celu skok do innego obszaru przestrzeni rozwiązań
- Elementem tej strategii może być długa lista tabu, wyprowadzająca poszukiwanie poza dotychczasowy obszar
- Można dopuścić wykonanie co jakiś czas ruchów bardziej wpływających na postać rozwiązania niż standardowe. Dopuszcza się w tej metodzie elementy losowości
- Innym sposobem mogą być restarty procedury poszukiwania, czyli rozpoczynanie całego procesu od nowego rozwiązania początkowego, zlokalizowanego w innej części przestrzeni rozwiązań. Często pamięć i zmienne są wtedy zerowane
- Do dywersyfikacji można wykorzystać także niektóre z rodzajów pamięci zdefiniowanych w metaheurystyce

Podział pamięci ze względu na czas

- *Pamięć krótkotrwała* (ang. *short-term memory*) służy zapamiętywaniu ostatnio wykonanych ruchów czy atrybutów rozwiązań w celu uniknięcia odwrócenia ruchu oraz wpadnięcia w cykle ruchów
- *Pamięć długotrwała* (ang. *long-term memory*) służy zapamiętywaniu ruchów/rozwiązań w dłuższym okresie czasu, w celu wynagradzania ruchów najlepszych bądź wykonywanych rzadko, albo karania ruchów najgorszych bądź występujących często (lub fragmentów rozwiązań zamiast ruchów). Pamięć może gromadzić informacje przez cały czas obliczeń algorytmu albo pomiędzy restartami, może być o stałej długości (FIFO) lub obejmować wszystkie zdarzenia w zadanym przedziale czasu

Podział pamięci ze względu na zawartość

- Lista tabu jest rodzajem *pamięci określającej* (ang. *attributive memory*), tzn. zapamiętuje informacje o atrybutach/cechach ruchów prowadzących do rozwiązania
- Drugim typem pamięci w metaheurystyce jest *pamięć bezpośrednia* (ang. *explicit memory*), pamiętająca kompletne rozwiązania, zazwyczaj najlepsze dotychczas odwiedzone
- Pamięć bezpośrednia może także zapamiętywać atrakcyjnych, lecz nieodwiedzonych dotąd sąsiadów najlepszych rozwiązań (do intensyfikacji poszukiwań). Może służyć unikaniu odwiedzania rozwiązań więcej niż raz, osiąga jednak wtedy duże rozmiary

Podział pamięci ze względu na funkcję

- *Recency-based memory* — pamięć gromadząca informacje o ostatnio wykorzystanych atrybutach, np. lista tabu
- *Frequency-based memory* — pamięć oparta na częstości wykorzystania atrybutów, może korzystać z miar:
  - *transition measure* — liczba iteracji, w których dany atrybut został użyty do wykonania ruchu, czyli wpłynął na postać rozwiązania
  - *residence measure* — liczba iteracji, podczas których atrybut należał do rozwiązania; wysoka wartość *residence measure* może świadczyć o atrakcyjności atrybutu, gdy otrzymujemy rozwiązania wysokiej jakości, lub wręcz przeciwnie, gdy rozwiązania są słabe

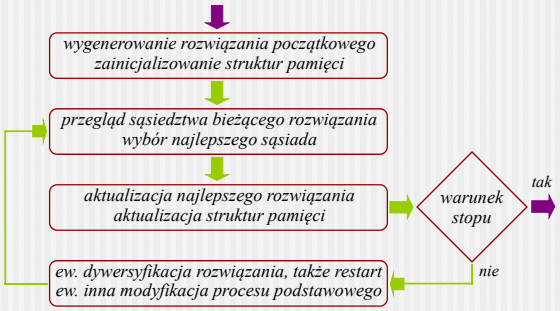
Podział pamięci ze względu na funkcję

- *Quality-based memory* — pamięć oparta na jakości, używana do zidentyfikowania elementów wspólnych dla dobrych rozwiązań lub ścieżek prowadzących do dobrych rozwiązań. Na jej podstawie preferuje się czynności skutkujące dobrymi rozwiązaniami albo obarcza karami czynności prowadzące do słabych rozwiązań
- *Influence-based memory* — uogólnienie *quality-based memory*; zapamiętuje wpływ dokonanych wyborów na rozwiązania, nie tylko pod względem wartości funkcji celu, lecz także struktury rozwiązania. Pamięć ta może wspomagać np. badanie dopuszczalności generowanych rozwiązań czy ich regionalności (gdy chcemy skoczyć w inny obszar przestrzeni rozwiązań)

Pamięć a strategie — przykłady użycia

- *Frequency-based memory* można użyć w strategii dywersyfikacji, np. przy restarcie procesu obliczeniowego, do złożenia nowego rozwiązania z najrzadziej używanych dotąd elementów. Można też, w trakcie procesu obliczeniowego, w mniejszym stopniu wpłynąć na postać rozwiązania poprzez wykorzystanie w części rzadko używanych atrybutów
- *Quality-based memory* można użyć w strategii intensyfikacji, np. składając nowe rozwiązania z fragmentów dobrych rozwiązań. Aby wpłynąć na postać rozwiązania w mniejszym zakresie, można wybierać pojedyncze ruchy prowadzące w przeszłości do dobrych rozwiązań

Ogólny schemat metody *tabu search*



Metoda przeszukiwania tabu — parametry

- Parametrami tej metaheurystyki umożliwiającymi dostosowanie jej do indywidualnych potrzeb są:
  - *długość listy tabu*
  - *rozmiar innych rodzajów pamięci* używanych w algorytmie
  - *liczba iteracji*; można przyjąć z góry zadaną liczbę iteracji albo np. liczbę iteracji bez poprawy funkcji celu; często definiowane są dwie lub więcej wartości składające się na liczbę iteracji, np. liczba kroków intensyfikacji, następująca po nich liczba kroków dywersyfikacji, liczba takich cykli i liczba restartów
  - *inne*

Metoda przeszukiwania tabu a losowość

- Metaheurystyka *tabu search* dopuszcza losowość, przy czym autor metody sugeruje minimalne jej użycie. Prawie zawsze zastosowanie przemyślanej strategii deterministycznej prowadzi do lepszych wyników niż zastosowanie losowości
- Nawet słaba deterministyczna strategia daje więcej informacji niż losowa, gdyż jest skonstruowana z uwzględnieniem cech danego problemu w celu poprawienia jakości rozwiązania (czego nie można powiedzieć o wyborach losowych). Słabą zasadę wyboru deterministycznego można poprawić na podstawie uzyskiwanych wyników
- Wybory deterministyczne są preferowane w *tabu search* także ze względu na pojedyncze rozwiązanie, na którym operujemy. Losowe wybory lepiej sprawdzają się, gdy mamy do dyspozycji dużo więcej „materiału eksperymentalnego”

Literatura — cd.

- Fred Glover, Manuel Laguna, *Tabu Search*, Kluwer Academic Publishers, Boston, 1997.
- http://spot.colorado.edu/~glover/