Optymalizacja. Przeszukiwanie „tabu”

dr inż. Maciej Komosiński

Instytut Informatyki
Politechnika Poznańska
www.cs.put.poznan.pl/mkomosinski

Naturalny sposób powstania algorytmu

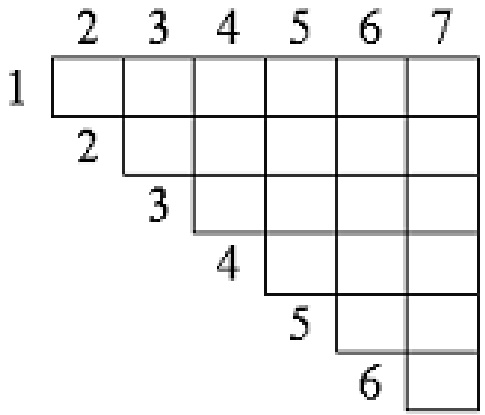
- Algorytm największego spadku
  niezdolność wyjścia z lokalnych optimów!
- Ulepszony algorytm największego spadku (akceptacja ruchów niepolepszających, podzbiór sąsiedztwa $V \subset N$)
  cykle!
- Lista „tabu”
  za dużo zakazanych ruchów!
- Poziom aspiracji – akceptacja atrakcyjnych ruchów zakazanych
  Algorytm przeszukiwania „tabu”.

Przeszukiwanie „tabu”

- Główna idea – wykorzystanie pamięci
- Zapamiętywanie rozwiązań lub ruchów (zmian)

Przykład zapamiętywania ruchów „tabu” (0)

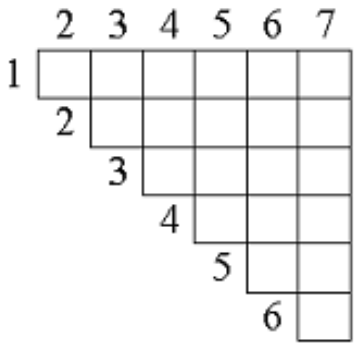
Struktura listy tabu:



Wewnątrz numer kadencji (ilość iteracji tabu do końca).

Iteracja 0 (punkt startowy, kryterium maksymalizacji)
Rozwiązanie bieżące – wartość=10

2	5	7	3	4	6	1



ruch		
5	**4**	**6**
7	4	4
3	6	2
2	3	0
4	1	-1

Przykład zapamiętywania ruchów „tabu” (1, 2)

Iteracja 1

Rozwiązanie bieżące – wartość=16

2	4	7	3	5	6	1

	2	3	4	5	6	7
1						
2						
3						
4	3					
5						
6						

ruch		
3	**1**	**2**
2	3	1
3	6	-1
7	1	-2
6	1	-4

Iteracja 2

Rozwiązanie bieżące – wartość=18

2	4	7	1	5	6	3

	2	3	4	5	6	7	
1		3					**T**
2							
3							
4	2						**T**
5							
6							

ruch		
1	3	-2
2	**4**	**-4**
7	6	-6
4	5	-7
5	3	-9

Przykład zapamiętywania ruchów „tabu” (3, 4)

Iteracja 3

Rozwiązanie bieżące – wartość=14

4	2	7	1	5	6	3

	2	3	4	5	6	7
1		2				
2			3			
3						
4			1			
5						
6						

T

ruch		
4	**5**	**6**
5	3	2
7	1	0
1	3	-3
2	6	-6

Iteracja 4

Rozwiązanie bieżące – wartość=20

5	2	7	1	4	6	3

	2	3	4	5	6	7
1		1				
2			2			
3						
4			3			
5						
6						

ruch		
7	1	0
4	3	-3
6	3	-5
5	4	-6
2	6	-8

Recency-based memory vs. frequency-based memory

Częstość wystąpień poszczególnych ruchów może być wykorzystana do rozrzucenia obliczen po całej przestrzeni rozwiązań dopuszczalnych (dywersyfikacja). Ruchy często powtarzane uzyskują karę w przypadku, gdy nie polepszaja wartości rozwiązania. Zwykle częstość ta jest w jakiś sposób normalizowana (dzielenie przez wykonaną lub maksymalną liczbę iteracji).

Dywersyfikacja jest przydatna tylko w określonych warunkach (np. gdy nie ma żadnych polepszeń).

Iteracja 26. $p = v - \text{kara\_za\_częstość}$

5	2	7	1	4	6	3

	1	2	3	4	5	6	7
1				3			
2							
3	3						
4	1	5					
5		4		4			
6			1				
7	2			3			

T

ruch	v	p
1	4	3	3
2	4	-1	-6
3	**7**	**-3**	**-3**
1	6	-5	-5
6	5	-4	-6

Przeszukiwanie tabu

```
procedure PRZESZUKIWANIE_TABU
begin
    INICJALIZUJ(xstart, xbest, T)
    x := xstart
    repeat
        GENERUJ(V ⊂ N(x))
        WYBIERZ(x')      //najlepsze f w V + aspiracja
        UAKTUALNIJ_LISTĘ_TABU(T)
        if f(x') ≤ f(xbest) then xbest := x'
        x := x'
    until WARUNEK_STOPU
end
```

Obserwacje

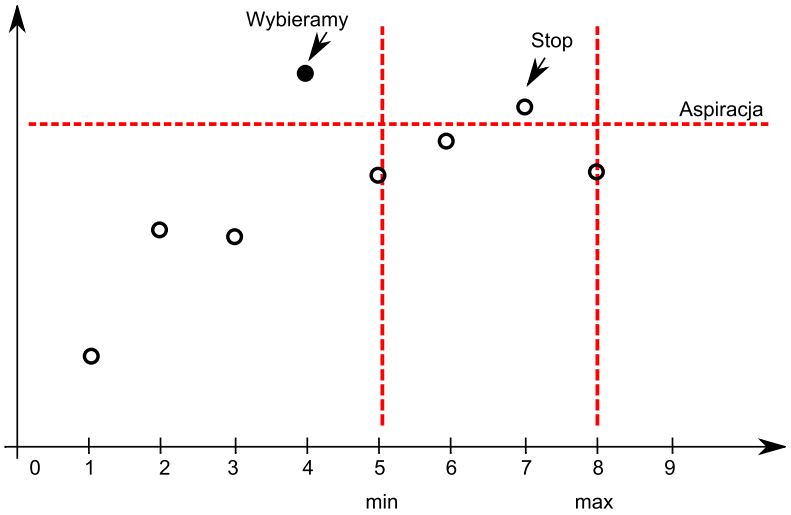
- Algorytm jest deterministyczny
  - Autor TS: „zły wybór strategiczny jest lepszy niż dobry wybór losowy” (bo daje jakieś informacje, wiedza wzrasta)
- Nowość: zbiór V (lista kandydatów)

Lista kandydatów

- Po co: żeby w każdej iteracji nie przeglądać i nie oceniać całego sąsiedztwa
- Jaki podzbiór sąsiadów N ma być kandydatami $V \subset N$?
  - kandydaci = dobrzy sąsiedzi
  - wyodrębnić regiony sąsiedztwa zawierające ruchy o pewnych pożądanych cechach...

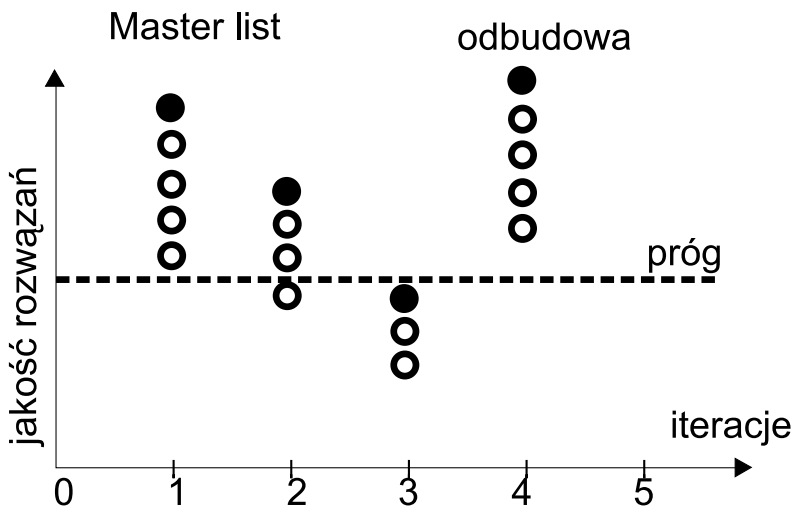
Konstrukcja listy kandydatów. Strategia 1: „Aspiracja plus”

- próg jakości ruchu – bazuje na historii przeszukiwania
- przeszukiwanie sąsiedztwa, aż do znalezienia sąsiada lepszego o pewną wartość progową (plus)
- liczba kandydatów rośnie aż do osiągnięcia wartości progowej
- wybieramy najlepszy znaleziony dotąd ruch
- Min $\leq$ ilość sąsiadów sprawdzonych $\leq$ Max
- poziom aspiracji może być zmienny w trakcie przeszukiwania
- strategia zwraca 1 lub więcej dobrych sąsiadów



Konstrukcja listy kandydatów. Strategia 2: „Elitarna lista kandydatów” (*master list*)

- Motywacja – dobry ruch jeśli nie jest wykonany w bieżącej iteracji, będzie wciąż dobry w kilku następnych
- tworzenie master list – sprawdzenie wszystkich lub większości ruchów, wybór k najlepszych (k jest parametrem)
- w następnych iteracjach aktualnie najlepszy ruch ze zbioru k ruchów jest wybierany do wykonania, aż jakość ruchu spadnie poniżej danego progu, lub zostanie osiągnięta pewna liczba iteracji



Kryteria aspiracji

- Cel: określić kiedy restrykcje „tabu” mogą zostać pominięte
- Pierwsze kryterium aspiracji: usunąć ograniczenie „tabu” wtedy, gdy ruch da rozwiązanie lepsze od znalezionego do tej pory
- **Aspiracja domniemana**
  - Jeżeli wszystkie ruchy są „tabu” i nie są dozwolone przez inne kryteria, to wybierany jest ruch najmniej „tabu”
- **Aspiracja wg kryterium**
  - Forma globalna – ruch „tabu” jest akceptowany jeżeli $c(x') < best\_cost$
  - Forma regionalna (przestrzeń dzieli się na podregiony R) – ruch „tabu” jest akceptowany jeżeli $c(x') < best\_cost(R)$

Intensyfikacja i dywersyfikacja (*exploitation and exploration*)

- Intensyfikacja
  - W dobrych rejonach
  - Powrót do najlepszego rozwiązania znalezionego do tej pory
  - Skrócenie listy Tabu *(short term memory)*
  - Pamięć długotrwała *(long term memory)*
    - każde rozwiązanie lub ruch to zbiór komponentów
    - zapamiętywanie komponentów dobrych ruchów lub rozwiązań w czasie obliczeń
    - w czasie intensyfikacji ruchy i rozwiązania włączają dobre komponenty
    - Pamięć długotrwała daje możliwość „uczenia się”
- Dywersyfikacja
  - Dla rzadko odwiedzanych rejonów
  - Karanie często wykonywanych ruchów – ucieczka z rejonu
- Mechanizmy te modyfikują jakoś funkcję celu:
  $f' = f + Int + Div$