Wrocław, 28 kwietnia 2021

Zaawansowane **T**echniki **O**ptymalizacji

Laboratorium 5

Wybrane problemy losowego i lokalnego poszukiwania rozwiązań

prowadzący: *dr inż. Jarosław Rudy*

1 Cel laboratorium

Celem laboratorium jest zapoznanie się z specyfiką i problemami rozwiązywania wybranych zadań optymalizacji dyskretnej z wykorzystaniem metaheurystyk poszukiwania losowego i lokalnego. Zagadnienie obejmuje zdefiniowanie reprezentacji rozwiązania w zależności od problemu, wybór implementacji algorytmu oraz kalibrację (dobór parametrów).

2 Przebieg zajęć

Laboratorium obejmuje zajęcia nr 9 i 10 (4 godziny zajęć). Praca odbywa się w ramach grup dwuosobowych. Każda grupa otrzymuje do zrealizowania jeden problem optymalizacji dyskretnej z poniższej listy:

1. Kwadratowe zagadnienie przydziału (problem 2.4).
2. Dyskretny problem plecakowy (problem 2.5).
3. Problem komiwojażera (problem 2.6).
4. Szeregowanie zadań na jednej maszynie z ważoną sumą opóźnień $w_i T_i$ (problem 2.7).
5. Problem przepływowy dla ustalonej liczby maszyn np. $m = 3$ (problem 2.8).

Dla otrzymanego problemu należy zrealizować **jedną** z poniższych opcji:

1. Implementacja metody poszukiwania losowego (RS). Na wyższą ocenę przekształcona w algorytm symulowanego wyżarzania (SA),
2. Implementacja metody poszukiwania zstępującego (DS). Na wyższą ocenę przekształcona w algorytm poszukiwania z zakazami (TS),

oraz odpowiednią kalibrację i analizę algorytmu (jak opisano dalej).

Poszczególne problemy opisane są w osobnym pliku PDF na stronie kursu. Metody rozwiązania opisane są w dalszej części instrukcji. Przydziału problemów i metod do grup dokonuje prowadzący podczas zajęć.

Naliczanie oceny zaczynamy od 0. Poszczególne części oceny można otrzymać za:

- +2.0 za algorytm w wersji podstawowej (RS lub DS),
- +2.0 za algorytm w wersji rozszerzonej (odpowiednio SA lub TS),
- +1.0 za wykonanie dostrojenia i badań (patrz dalej).

Dostrojenie i badanie algorytmu polega na sprawdzeniu efektywności (wpływu na jakość rozwiązań) różnych elementów algorytmu. W zależności od problemu zwykle obejmuje to:

- sprawdzenie różnych sąsiedztw,
- sprawdzenie różnych rozwiązań początkowych,
- kalibracja parametrów liczbowych,
- inne, w zależności od algorytmu (schemat chłodzenia, forma listy tabu).

Należy zestawić ze sobą co najmniej dwie wersje danego algorytmu (lepiej i gorzej skalibrowaną). Ze względu na czas potrzebny na wykonanie badań pojedyncze uruchomienie algorytmu powinno trwać do 10 sek. (z wyjątkiem sytuacji gdy np. testujemy jakość od czasu działania).

Uwaga: w przypadku pisania wersji rozszerzonej warto zapisać sobie wersję podstawową (może być pomocna przy ustalaniu oceny gdy wersja rozszerzona jest niedokończona lub zawiera błędy).

3 Metaheurystyki poszukiwania lokalnego i losowego

W kwestii przypomnienia: heurystyką nazywamy metodę rozwiązania danego problemu (algorytm), która nie gwarantuje otrzymania rozwiązania optymalnego (a często nawet rozwiązania dopuszczalnego). Metaheurystyka jest w uproszczeniu heurystyką „wyższego poziomu". Ściślej, metaheurystyka jest ogólnym schematem (opisem) pozwalającym na tworzenie algorytmów w oparciu o ten schemat. Metaheurystyka definiuje pewne ogólne pojęcia lub mechanizmy, które należy następnie przełożyć na dany problem. Metaheurystyka sama w sobie jest więc bardziej ogólną metodą niż konkretnym algorytmem. Dopiero konkretna implementacja metaheurystyki dla konkretnego problemu stanowi algorytm rozwiązania.

Zaletą metaheurystyk jest to, że można je stosować dla szerokiego zakresu problemów często uzyskując dobre jakościowo wyniki przy akceptowalnych (i kontrolowanych) zasobach obliczeniowych. Istnieje też duża swoboda w szczegółach implementacji oraz rozbudowy algorytmu. Wadą jest konieczność dostosowania metaheurystyki do każdego nowego problemu (zwłaszcza jeśli chcemy wykorzystać jego specyficzne własności) oraz często konieczność kalibracji algorytmu.

W praktyce metaheurystyki są najczęściej metodami iteracyjnymi. Zaczynają od ustalenia rozwiązania początkowego (lub zbioru takich rozwiązań), po czym w każdej iteracji zmieniają je na inne rozwiązanie (potencjalnie lepsze. W niektórych przypadkach wymaga się wręcz by każda iteracja dostarczała poprawy, aczkolwiek jest to rzadsze w praktyce).

3.1 Reprezentacja rozwiązania

Pierwszą kwestią w implementacji algorytmu metaheurystycznego jest wybór reprezentacji rozwiązania. Często reprezentacja nie jest bezpośrednia. Przykładowo, dla wielu problemów szeregowania zadań rozwiązaniem jest zasadniczo harmonogram tj. czasy w których należy rozpocząć wykonywanie poszczególnych zadań na poszczególnych maszynach. Jednakże w praktyce często przyjmuje się reprezentację w postaci kolejności zadań. Odpowiedni wybór reprezentacji pozwala ograniczyć liczbę rozwiązań (w szczególności wyeliminować część lub całość rozwiązań niedopuszczalnych). Wybór reprezentacji rozwiązania ma też wpływ na sposób liczenia funkcji celu.

Rozpatrzmy prosty problem optymalizacji testu jednokrotnego wyboru z laboratorium nr 1. Załóżmy, że jest n pytań, każde ma 4 warianty odpowiedzi. W takim przypadku w zupełności wystarczy reprezentacja bezpośrednia w postaci wektora długości n, którego elementami są liczby ze zbioru $\{-1, 0, 1, 2, 3\}$ (wartość -1 oznacza że na pytanie nie udzielono odpowiedzi). Tak zdefiniowana przestrzeń zawiera dokładnie 5^n różnych rozwiązań i wszystkie są dopuszczalne. Inną możliwą reprezentacją jest wektor $4n$ zmiennych binarnych (jedynka oznacza zakreślenie wariantu, zero zaś

brak zakreślenia). Tak zdefiniowana przestrzeń ma 2^{4n} rozwiązań, z czego jednak tylko 5^n jest dopuszczalnych. Ta reprezentacja jest więc zdecydowanie nadmiarowa.

3.2 Ruch i sąsiedztwo

Zasadniczo wszystkie metaheurystyki rozważane w ramach tego laboratorium należą do kategorii metaheurystyk lokalnego poszukiwania. Kluczowym w ich przypadku jest zdefiniowanie pojęcia ruchu i sąsiedztwa. Przez ruch rozumiemy funkcję (zwykle wieloargumentową), która zamienia dane rozwiązanie x w rozwiązanie x', tak że oba rozwiązania są w pewnym sensie do siebie podobne – są swoimi sąsiadami. Zbiór wszystkich sąsiadów rozwiązania x nazywamy jego sąsiedztwem, a liczbę sąsiadów rozmiarem sąsiedztwa. Z matematycznego punktu widzenia powoduje to, że przestrzeń rozwiązań jest przestrzenią topologiczną.

Dla wspomnianego problemu optymalizacji testu ruch możemy zdefiniować następująco. Niech x_i będzie wybranym wariantem odpowiedzi w pytaniu i. Wtedy ruch jest funkcją $f(x, i, w)$, której działanie polega na przypisaniu $x_i = w$. Takie zdefiniowane sąsiedztwo ma rozmiar dokładnie $4n-1$ (jest tak, ponieważ jednym z sąsiadów x jest sam x). Zauważmy, że przy takiej definicji sąsiedztwa można przejść z dowolnego rozwiązania w dowolne inne za pomocą $n-1$ ruchów. Często pożądane jest też by sąsiedztwo posiadało własność połączenia tzn. by dla dowolnego rozwiązania początkowego istniał skończony ciąg ruchów prowadzących do jakiegoś rozwiązania optymalnego. Wiele sąsiedztw posiada taką własność, aczkolwiek znane są również skuteczne sąsiedztwa jej nieposiadające. Zwykle można zdefiniować wiele sąsiedztw dla tego samego problemu (dla tych samych lub różnych reprezentacji). Przykładowo, dla reprezentacji z wektorem $4n$ zmiennych binarnych ruchem może być funkcja $f(x, i)$, która zamienia wartość i-tego bitu wektora na przeciwny. Wcześniej jednak napisaliśmy, że ta reprezentacja zawiera rozwiązania też niedopuszczalne. Przy generacji (przeglądaniu) rozwiązań sąsiednich należy więc pomijać rozwiązania niedopuszczalne[1].

Uwaga: odpowiednie zdefiniowanie ruchu jest bardzo istotne i zależy w dużej mierze od problemu.

3.3 Poszukiwanie losowe

Poszukiwanie losowe (Random Search, RS) jest prostą metaheurystyką. Poniżej przedstawiono ogólny schemat dla problemów minimalizacji:

1. Ustal rozwiązanie początkowe x.
2. Dopóki warunek stopu nie jest spełniony:
   2.1. Wylosuj rozwiązanie x' spośród sąsiadów x.
   2.2. Jeśli $f(x') < f(x)$ to wykonaj $x \leftarrow x'$.
3. Wypisz rozwiązanie x.

Gdzie $f(x)$ jest wartością funkcji celu dla rozwiązania x.

Rozwiązanie początkowe najczęściej jest również losowe (lub najlepsze z np. 100 losowych) albo dostarczane przez prosty algorytm deterministyczny (często zachłanny). Losowanie sąsiada odbywa się z równym prawdopodobieństwem dla wszystkich sąsiadów. Warunki stopu mogą być różnorodne, jednak najczęstszym warunkiem jest przekroczenie założonej liczby iteracji (która może być stała lub zależna od rozmiaru problemu). Innymi możliwościami są:

- osiągnięto minimum lokalne,
- przekroczony czas obliczeń,
- przekroczona liczba iteracji od czasu ostatniej poprawy,
- kombinacja więcej niż jednego warunku stopu.

[1] W praktyce istnieją algorytmy, które rozważają rozwiązania niedopuszczalne.

3.4 Poszukiwanie zstępujące

Poszukiwanie zstępujące[2] (descending search, DS) jest bardzo zbliżone do poszukiwania losowego. Różnica polega na tym, że kandydatem na zastąpienie rozwiązania aktualnego jest najlepsze rozwiązanie z sąsiedztwa. Przykładowych schemat metody:

1. Ustal rozwiązanie początkowe x.
2. Dopóki warunek stopu nie jest spełniony:
   2.1. $x' = x$.
   2.2. Dla każdego x_s będącego sąsiadem x.
      2.2.1. Jeśli $f(x_s) < f(x')$ to wykonaj $x' \leftarrow x_s$.
   2.3. Jeśli $f(x') < f(x)$ to wykonaj $x \leftarrow x'$.
3. Wypisz rozwiązanie x.

Punkt 2.1. ma za zadanie ustalić początkową wartość dla x'. Równoważnie można ten punkt usunąć i nie sprawdzać warunku $f(x_s) < f(x')$ dla pierwszego sąsiada tylko przypisać go bezwarunkowo.

Zauważmy, że najlepszy sąsiad x' zastępuje aktualne rozwiązanie x tylko jeśli jest od niego lepszy (podobnie jak dla metody RS). Warunki stopu są analogiczne jak dla metody RS. Należy również nadmienić, że choć RS musi być losowe (przynajmniej częściowo), to DS nie posiada zakazu losowości (np. w kwestii ustalania rozwiązania początkowego). Algorytmy czysto deterministyczne są jednak czasami preferowane ze względu na powtarzalność wyników.

3.5 Symulowane wyżarzanie

Powyższe dwie metody mają jedną zasadniczą wadę – po znalezieniu optimum lokalnego (tj. rozwiązania, które jest niegorsze niż wszyscy jego sąsiedzi) nie nastąpi już zmiana sąsiedztwa ani żadna dalsza poprawa rozwiązania. Możemy powiedzieć, że metody te oparte były na intensyfikacji poszukiwań tj. skupianiu się na rozwiązaniach bliskim rozwiązaniom poprzednim („dobre rozwiązania są blisko dobrych"). Taka intensyfikacja jest pożądana, ale powinna być uzupełniona przez dywersyfikację – mechanizm (czasami) kierujący proces poszukiwań w inne (niekoniecznie losowe), niesprawdzone jeszcze obszary przestrzeni rozwiązań. Umiejętne sterowanie balansem intensyfikacja-dywersyfikacja (znanym też jako eksploatacja-eksploracja) jest często mocno nietrywialne.

Z powyższego powodu opracowano ulepszone wersje metod DS i RS. Ulepszoną wersją DS jest Symulowane Wyżarzanie (Simulated Annealing, SA). Metoda jest inspirowana procesem wyżarzania w metalurgii. Rozwiązanie jest postrzegane jako analogia rozkładu (struktury) cząstek substancji, a wartość funkcji celu jest analogią energii układu. Kluczowa jest obserwacja, że zmiana struktury układu zachodzi łatwiej w wyższej temperaturze. Z tego względu w metodzie SA istnieje temperatura, której wartość zmienia się w trakcie algorytmu (sposób tej zmiany nazywany jest schematem chłodzenia). Metoda dopuszcza zaakceptowanie gorszego rozwiązania niż aktualne (dając szansę na opuszczenie minimum lokalnego), a prawdopodobieństwo tego jest zależne od aktualnej temperatury i różnicy wartości funkcji celu rozwiązań (w analogii do różnicy energii stanów substancji).

Poniżej przedstawiono przykładowy schemat metody SA (dla problemu minimalizacji):

1. Ustal rozwiązanie początkowe x oraz temperaturę początkową t.
2. Wykonaj $x_{\text{best}} \leftarrow x$.
3. Dopóki warunek stopu nie jest spełniony:

[2]Nazwa dla problemów minimalizacji. Dla problemów maksymalizacji mówi się o metodzie wspinaczkowej (hill climbing).

3.1. Wylosuj rozwiązanie x' spośród sąsiadów x.

3.2. Jeśli $f(x') < f(x)$ to wykonaj $x \leftarrow x'$. W przeciwnym razie wykonaj $x \leftarrow x'$ z prawdopodobieństwem równym $p(x, x', t)$.

3.3. Jeśli $f(x) < f(x_{\text{best}})$ to wykonaj $x_{\text{best}} \leftarrow x$.

3.4. Zaktualizuj t w zależności od przyjętego schematu chłodzenia.

4. Wypisz rozwiązanie x_{best}.

Funkcja $p(x, x', t)$ zwana jest funkcją akceptacji. Najczęściej przyjmuje ona postać:

$$p(x, x', t) = e^{\frac{-(f(x')-f(x))}{t}}. \tag{1}$$

Istnieje kilka często spotykanych schematów chłodzenia. Najpopularniejszy jest schemat geometryczny, gdzie aktualizuje się temperaturę poprzez $t \leftarrow \alpha t$, gdzie α jest współczynnikiem chłodzenia, zwykle w przedziale od 0.95 do 0.9999. Choć schemat nazywa się geometrycznym, to podstawa bliska 1 sprawia, że wykres temperatury jest zbliżony do liniowego. Drugim często spotykanym schematem jest schemat liniowy: $t \leftarrow t - \beta$. W przypadku tego schematu należy uważać, by temperatura nie spadła poniżej wartości 0.

Istotnym elementem jest również wybór temperatury początkowej. Ze względu na postać wzoru (1) temperatura początkowa powinna być w jakiś sposób zależna od spodziewanej (typowej, możliwej) różnicy pomiędzy x' oraz x. Jednym ze sposobów jest wylosowanie pewnej liczby (np. 1000) rozwiązań losowych i obliczeniu rozstępu (różnicy pomiędzy maksimum i minimum) wartości funkcji celu. Następnie można użyć wartości rozstępu przy obliczaniu temperatury początkowej.

Warunki stopu dla metody SA mogą być identyczne jak dla metody RS. Dodatkowo często spotykany jest warunek osiągnięcia odpowiednio niskiej temperatury końcowej. W przedstawionym powyżej schemacie jest to jednak równoważne kryterium liczby iteracji tzn. temperatura końcowa zostaje osiągnięta po wiadomej z góry liczbie iteracji, zależnie od przyjętej temperatury początkowej, końcowej i parametrów α (lub β).

3.6 Poszukiwanie z zakazami

Poszukiwanie z zakazami (Tabu Search, Taboo Search, TS) jest zaawansowaną wersją metody poszukiwania lokalnego. Podobnie jak w metodzie DS przeglądamy całe sąsiedztwo, w celu znalezienia najlepszego sąsiada. Pierwsza różnica polega na tym, że w przypadku DS najlepszy sąsiad x' zastępował aktualne rozwiązanie x tylko jeśli był lepszy (wspinaczka w górę wzgórza lub schodzenie w dół doliny). W przypadku metody TS, x' zastępuje x nawet wtedy gdy jest x' jest gorsze od x. Taki zabieg daje szansę na opuszczenie minimum lokalnego.

Jednakże dużą wadą powstałego rozwiązania jest możliwość powstania cykli. W szczególności najlepszy sąsiadem x_a może być x_b, a najlepszym sąsiadem x_b może być x_a. W tej sytuacji algorytm bez końca przemieszczał by się pomiędzy tymi dwoma rozwiązaniami. Modyfikacja pozwoliła więc uniknąć cykli rozmiaru 1, ale nie dłuższych. Aby zminimalizować ryzyko powstania cykli wprowadza się do algorytmu pamięć, która ma za zadanie unikać odwiedzania rozwiązań już sprawdzonych.

Najpowszechniejszym rodzajem pamięci jest pamięć krótkoterminowa. W najprostszym podejściu pamięć ta przyjmuje postać listy o zadanej długości na której przechowuje się niedawno odwiedzone rozwiązania. Najnowsze elementy dodawane są na początek, a te które przekroczą rozmiar listy są usuwane. Rozwiązania będące na liście nazywane są rozwiązaniami zakazanymi (tabu) i są ignorowane podczas przeglądu sąsiedztwa (z wyjątkiem mechanizmu kryterium aspiracji, opisanego dalej). Łatwo zauważyć, że ruch będzie zakazany przez liczbę iteracji równą długości listy tabu.

Choć jest to „najdokładniejsze" podejście, to jest mało wydajne. Sprawdzenie czy dane rozwiązanie x jest na liście tabu wymaga pesymistycznie czasu $O(LC)$, gdzie L jest długością listy zaś C jest rozmiarem rozwiązania (np. w problemie testu jednokrotnego wyboru trzeba sprawdzić czy

wszystkie n pytań ma wybrany ten sam wariant). Z tego względu w praktyce stosuje się podejście uproszczone. Zamiast przechowywać rozwiązania, lista przechowuje ruchy, a ściślej przechowuje ruch odwrotny. Innymi słowy, jeśli z rozwiązania x przechodzimy do x' za pomocą pewnego ruchu, to na liście powinien pojawić się ruch odwrotny, czyli taki który pozwala uzyskać x z x'. Jest tak, gdyż chcemy zakazać powrotu, który mógłby spowodować cykl. W praktyce ruch jest przechowywany jako jego atrybuty. Dla opisanego wcześniej ruchu danego funkcją $f(x, i, w)$, atrybutami są i oraz w.

Powyższe podejście pozwala skrócić czas sprawdzenia czy ruch r jest zakazany do czasu $O(L)$, gdyż dwa ruchy możemy porównać w czasie stałym. W wielu przypadkach można wprowadzić jeszcze jedną modyfikację. Otóż zamiast przechowywać listę tabu w postaci listy, tworzymy tablicę o tylu wymiarach ile jest atrybutów ruchu. W problemie testu tablica będzie miała dwa wymiary, jeden wielkości 4 (są cztery warianty w każdym pytaniu) oraz drugi wielkości n (tyle ile jest pytań). W elemencie $[i][w]$ zapisujemy numer iteracji, od której ruch $f(x, i, w)$ jest już dozwolony[3]. Początkowo wszystkim elementom tablicy przypisujemy wartość 0 (wszystkie ruchy są dozwolone). Gdy ruch $f(x, i, w)$ staje się zakazany, to w miejsce $[i][w]$ wpisujemy wartość $L + \text{iter}$, gdzie L jest liczbą iteracji przez które ruch jest zakazany (parametr ten nazywamy kadencją), zaś iter jest numerem bieżącej iteracji. Przykładową tablicę dla $n = 6$ pokazano w Tabeli 1. Jeśli iter $= 21$ to zakazane są ruchy postaci $f(x, 2, 2)$ oraz $f(x, 5, 3)$. W takiej formie sprawdzenie czy ruch jest zakazany odbywa się w czasie $O(1)$, poprzez test czy iter $\geqslant [i][w]$.

	$w=1$	$w=2$	$w=3$	$w=4$
$i=1$	0	0	14	0
$i=2$	11	22	15	0
$i=3$	19	13	21	0
$i=4$	0	20	0	16
$i=5$	0	12	23	0
$i=6$	18	0	17	0

Tabela 1: Przykładowa zawartość tablicy tabu dla problemu testu jednokrotnego wyboru

Ostatnim często spotykanym elementem jest tzw. kryterium aspiracji, które służy do wybiórczego ignorowania reguły tabu. Ściślej sąsiad x' jest brany pod uwagę jeśli ruch prowadzący do niego nie jest zakazany lub x' spełnia kryterium aspiracji. W najprostszej formie dla problemów minimalizacji x' spełnia kryterium aspiracji, jeśli $f(x') < f(x_{\text{best}})$, gdzie x_{best} jest najlepszym obecnie znanym rozwiązaniem. Stosuje się też zbliżone formy typu $f(x') < 0.9f(x_{\text{best}})$ czy $f(x') < 1.1f(x_{\text{best}})$.

Biorąc pod uwagę powyższe informacje, podstawowy schemat metody TS dla problemów minimalizacji wygląda następująco:

1. Ustal rozwiązanie początkowe x.
2. Wykonaj $x_{\text{best}} \leftarrow x$.
3. Dopóki warunek stopu nie jest spełniony:
   - 3.1. $x' \leftarrow \varnothing$
   - 3.2. Dla każdego x_s będącego sąsiadem x.
     - 3.2.1. Jeśli (ruch prowadzący z x do x_s nie jest zakazany lub x_s spełnia kryterium aspiracji) oraz $f(x_s) < f(x')$ to wykonaj $x' \leftarrow x_s$.
   - 3.3. Wykonaj $x \leftarrow x'$ oraz dodaj ruch prowadzący z x' do x do listy tabu.
   - 3.4. Jeśli $f(x) < f(x_{\text{best}})$ to wykonaj $x_{\text{best}} \leftarrow x$.

[3]Paradoksalnie to uproszczenie powoduje, że to czy rozwiązanie x jest zakazane nie zależy wprost od x.

4. Wypisz rozwiązanie x_{best}.

Przyjmujemy że $f(\varnothing) = \infty$. Ponadto kadencja powinna być zawsze mniejsza niż rozmiar sąsiedztwa, gdyż w przeciwnym wypadku istnieje szansa, że wszystkie ruchy będą zakazane, a żaden sąsiad nie będzie spełniał kryterium aspiracji.