Wrocław, 13 kwietnia 2021

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

Laboratorium 4

Wybrane problemy drzewiastego 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 algorytmów bazujących na przeszukiwaniu drzewiastym. Zagadnienie obejmuje metody obliczania dolnego i górnego ograniczenia, implementację dokładnych i przybliżonych metod rozwiązania oraz porównanie algorytmów.

2 Przebieg zajęć

Laboratorium obejmuje zajęcia nr 7 i 8 (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_iT_i (problem 2.7).
5. Problem przepływowy dla ustalonej liczby maszyn np. $m = 3$ (problem 2.8).

Dla wybranego problemu należy zrealizować poniższe zadania:

1. Implementacja metody podziału i ograniczeń (B&B).
2. Implementacja przeszukiwania snopowego (BS).

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

Naliczanie oceny zaczynamy od 0. Punkty można uzyskać za:

- implementację metody dokładnej (B&B) dla zadanego problemu (maksymalnie +2.0 do oceny),
- implementację metody przybliżonej (BS) dla zadanego problemu (maksymalnie +1.0 do oceny),
- jakość dolnych i górnych ograniczeń (maksymalnie +1.0 do oceny),
- proste badania porównawcze (jakość/wydajność/zakres zastosowania) obu algorytmów (maksymalnie +1.0 do oceny).

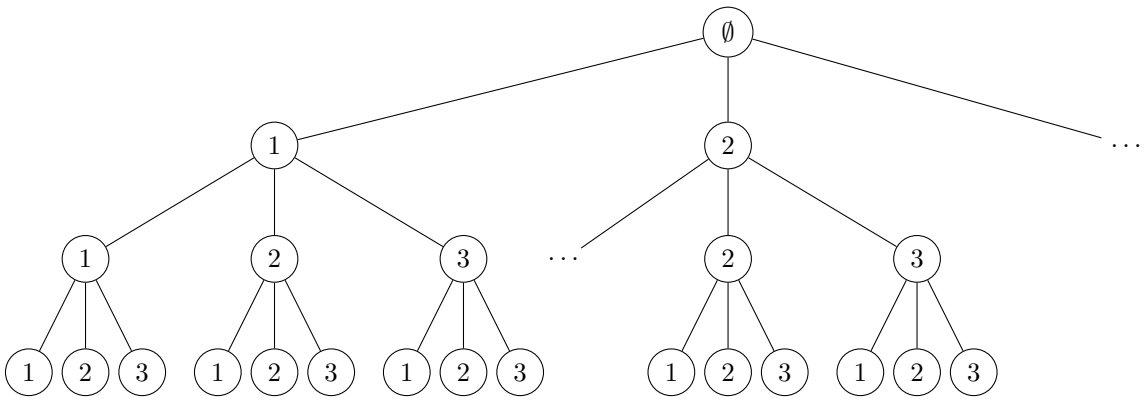
Uwaga 1: dopuszczalne jest zrealizowanie B&B i/lub BS bez stosowania dolnego/górnego ograniczenia (co zredukuje B&B do brute force'a, a BS do jego przybliżonej „uciętej" wersji), ale (1) algorytmy dalej muszą być drzewiaste (działać na podstawie przeglądu drzewa) oraz (2) redukuje to ocenę.

Uwaga 2: w przypadku pisania metody BS warto zapisać sobie działającą metodę B&B (może być pomocna przy ustalaniu oceny gdy metoda BS jest niedokończona lub zawiera błędy). Analogicznie, warto zdefiniować proste wersje ograniczeń, na wypadek gdyby wersje zaawansowane zawierały błędy.

3 Algorytmy drzewiastego poszukiwania rozwiązań

Większość problemów można sformułować jako proces decyzyjny. Proces zwykle zaczyna się od rozwiązania pustego, zaś w każdym kroku podejmowana jest pewna decyzja, która prowadzi do uzyskania kolejnej części rozwiązania. W ten sposób budowane jest rozwiązanie częściowe. Proces ten można przedstawić w postaci drzewa decyzyjnego. W węzłach drzewa będą przechowywane aktualne stany (rozwiązania częściowe). Decyzje będą reprezentowane przez krawędzie wychodzące z węzłów. Pełne rozwiązanie problemu otrzymamy po dotarciu do liścia w drzewie. Łatwo zauważyć, że liczba liści jest równa liczbie rozwiązań (rozmiarowi przestrzeni rozwiązań). Wniosek jest taki, że drzewo decyzyjne zawiera (dużo) więcej węzłów niż istnieje rozwiązań problemu.

Rozważmy drzewo decyzyjne dla problemu optymalizacji testu jednokrotnego wyboru z laboratorium nr 1. Załóżmy, że jest n pytań, każde ma 3 warianty odpowiedzi. Proces decyzyjny polega w tym przypadku na wyborze wariantu odpowiedzi dla każdego pytania według pewnej przyjętej kolejności pytań. Najprościej przyjąć, że najpierw wybieramy odpowiedź na pytanie 1, potem pytanie 2 itp. Na każdym poziomie drzewa będziemy budować rozwiązanie częściowe, dodając nową decyzję do powstającego rozwiązania częściowego (wektora odpowiedzi). Stan początkowy jest wektorem pustym. Przy tym założeniu wysokość drzewa (liczba poziomów) wyniesie $n + 1$. Z pominięciem liści z każdego węzła będą wychodzić dokładnie 3 krawędzie do węzłów-dzieci na poziomie kolejnym. Warto zauważyć, że nie jest to prawda dla wszystkich problemów – dla niektórych problemów liczba dzieci jest zależna od położenia węzła w drzewie (w szczególności od tego na którym jest poziomie). Na rysunku 1 został przedstawiony schemat drzewa decyzyjnego dla wspominanego problemu testu jednokrotnego wyboru. Dla czytelności przedstawiono tylko część drzewa.



Rysunek 1: Drzewo decyzyjne dla testu jednokrotnego wyboru dla $n = 3$

Oczywistym rozwiązaniem problemu optymalizacji wydaje się sprawdzenie wszystkich ścieżek w drzewie decyzyjnym, tzn. zbudowanie wszystkich możliwych rozwiązań poprzez dotarcie do każdego liścia (dla omawianego problemu jest 3^n rozwiązań). Takie podejście nazywamy metodą siłową (*Brute Force*, BF). Zaletą jest szybkie (przy odpowiedniej implementacji) przechodzenie pomiędzy węzłami. Wadą – duża liczba węzłów do sprawdzenia. Istnieją jednak metody ograniczenia liczby sprawdzanych węzłów w oparciu o analizę rozwiązania częściowego, opisane poniżej.

3.1 Metoda podziału i ograniczeń

Metoda podziału i ograniczeń (*Branch and Bound*, B&B) polega na efektywnym przeglądaniu drzewa i „odcinaniu" części gałęzi, w których mamy gwarancje, że nie znajduje się optymalne rozwiązanie. W każdym etapie algorytmu, w momencie podjęcia decyzji należy ocenić potencjał danej drogi. W problemach **maksymalizacji** wykorzystuje się do tego górne ograniczenie (*Upper Bound*, UB). Wartość UB ogranicza od góry wartość funkcji celu jaką można uzyskać rozwijając dalej daną gałąź. Innymi słowy, jeśli dla danego rozwiązania częściowego $\hat{x}$ mamy (poprawnie wyliczone) UB, to dla każdego rozwiązania x, osiągalnego z $\hat{x}$ zachodzi UB $\geqslant f(\hat{x})$. Wartość rozwiązania końcowego w danej gałęzi nie będzie lepsza niż UB (może być równa lub mniejsza).

Zwykle istnieje wiele sposobów na obliczenie UB. Im lepsza (tzn. bliższa faktycznemu maksimum funkcji celu dla wszystkich rozwiązań osiągalnych z $\hat{x}$) jest wartość UB, tym większą część gałęzi będzie można odciąć. Jednakże lepsza wartość UB zwykle wymaga większego czasu na jej obliczenie (a jest ona obliczana w każdym węźle!). W związku z tym potrzebny jest kompromis pomiędzy czasem poświęconym na obliczanie UB, a czasem zaoszczędzonym dzięki obcinaniu gałęzi. Należy uważać, ponieważ w przypadku zawyżania wartości przez funkcję liczącą UB może się okazać, że algorytm dokona przeglądu zupełnego wszystkich gałęzi. W przypadku zaniżania wartości można doprowadzić do odcięcia gałęzi zawierającej rozwiązanie optymalne, co w konsekwencji będzie skutkowało błędnym działaniem algorytmu (jeśli utniemy wszystkie takie „optymalne" gałęzie).

Poprawna funkcja licząca UB musi uwzględniać dwie składowe: (1) znaną wartość funkcji celu rozwiązania częściowego oraz (2) oszacowanie najlepszej wartości funkcji celu możliwej do wystąpienia w wyniku niepodjętych jeszcze decyzji. W przypadku testu jednokrotnego wyboru (1) jest sumą punktów za wybrane warianty pytań na które już odpowiedzieliśmy. Z kolei (2) mogłoby być np. sumą punktów za pozostałe pytania, gdzie dla każdego pytania wybieramy wariant najlepszy (dający najwięcej punktów). Zauważmy, że jeśli zostało k pytań (każde po 3 warianty odpowiedzi) to wymaga to $3k$ operacji. Inną możliwością na obliczenie (2) opiera się na wyliczeniu na początku algorytmu (przed rozpoczęciem przeglądu drzewa) maksymalnej wartości ze wszystkich $3n$ wariantów występujących w całym teście. Oznaczmy tę wartość przez M. Wtedy jako wartość (2) można podać po prostu kM, co wymaga tylko jednej operacji mnożenia. Tak obliczone UB jest mniej dokładne, ale można je wyliczyć w czasie stałym.

Przejdźmy do samej operacji odcięcia (cały czas dla problemu **maksymalizacji**). Odcięcie gałęzi następuje w momencie jeśli wartość UB jest mniejsza niż wartość dolnego ograniczenia (*Lower Bound*, LB). LB jest ograniczeniem dolnym na wartość optymalną całego problemu. W praktyce za wartość LB przyjmowana jest wartość funkcji celu najlepszego znanego do tej pory rozwiązania. Dodatkowo, w celu przyśpieszenia algorytmu B&B, wartość początkowa LB wylicza się jeszcze przed rozpoczęciem przeglądu drzewa w oparciu o dodatkowe algorytmy. Zazwyczaj stosuje w tym celu (1) rozwiązanie losowe (lub najlepsze z pewnej liczby losowych), (2) rozwiązanie uzyskane pewnym algorytmem zachłannym. Ponadto, jeśli nie mamy żadnej wartości początkowej dla LB, a przeglądamy drzewo metodą inną niż przegląd wgłąb (*Depth-First Search*, DFS), to warto LB obliczyć na podstawie pierwszego liścia, który uzyskalibyśmy metodą DFS. Robimy tak dlatego, że inne metody przeglądu (wszerz, najpierw najlepszy itp.) mogą rozwinąć dużo węzłów nim dotrą do pierwszego liścia.

Pierwotnie algorytm był opisywany w wersji rekurencyjnej, ale aktualnie coraz częściej spotyka się jego wersję iteracyjną. Dla przeglądu najpierw-najlepszy węzły umieszcza się w kolejce priorytetowej, gdzie priorytetem jest wartość UB danego węzła. Dla przeglądu wgłąb węzeł umieszcza się na stosie (kolejce LIFO), a dla przeglądu wszerz – na kolejce FIFO.

Uwaga: dla problemów minimalizacji (większość przerabianych w trakcie zajęć) definicje LB i UB należy odwrócić tzn. wartość rozwiązania z danej gałęzi nie będzie lepsza niż LB, a rozwiązanie optymalne nie będzie gorsze niż UB (wartość już znaleziona). Poniżej przedstawiono ogólny schemat B&B dla problemu **minimalizacji**:

1. Wyznacz rozwiązanie początkowe x.

2. Wykonaj UB $\leftarrow f(x)$.

3. Umieść w kolejce Q wszystkie rozwiązania osiągalne z rozwiązania pustego, wraz z obliczonymi wartościami LB.

4. Dopóki Q nie jest pusta.

   4.1. Zdejmij węzeł v z Q według wybranej strategi.

   4.2. Jeśli v jest liściem i reprezentuje rozwiązanie x':

      4.2.1. Jeśli $f(x') <$ UB to wykonaj $x \leftarrow x'$ oraz UB $\leftarrow f(x')$.

   4.3. Jeśli v nie jest liściem, to dodaj jego dzieci do Q wraz z obliczonymi wartościami LB.

5. Wypisz rozwiązanie x.

3.2 Poszukiwanie snopowe

Metoda B&B jest dokładna, ale jej czas działania jest często nieakceptowalnie długi lub wręcz nieprzewidywalny. Istnieją sposoby na modyfikację B&B, które sprawiają że nie będziemy mieli gwarancji uzyskania rozwiązania optymalnego (metoda stanie się przybliżona a nie dokładna), ale jej czas działania stanie się dużo krótszy, a nawet będzie można go kontrolować.

Jedną z takich metod przybliżonych bazujących na idei B&B jest tzw. poszukiwanie snopowe (*Beam Search, BS*). Główną różnicą względem schematu B&B jest moment dodania nowych węzłów-dzieci do kolejki. Metoda B&B dodaje zawsze wszystkie takie węzły, zaś metoda BS dodaje maksymalnie k węzłów gdzie k może być parametrem algorytmu. Zwykle do dodania wybiera się k najbardziej obiecujące węzły (z najmniejszą wartością odpowiedniego ograniczenia). Pozostałe węzły są odrzucane. Ryzykujemy więc że odetniemy jakąś perspektywiczną gałąź. Ryzyko zależy m.in. od wartości k oraz jakości górnych i dolnych ograniczeń. Dodatkowo, k może zmieniać się w trakcie trwania algorytmu np. być zależne od poziomu danego węzła w drzewie, lub od czasu trwania algorytmu (np. liczby rozwiniętych węzłów do tej pory). W przeciwieństwie do B&B, w BS dużo częściej stosuje się strategię przeglądania wszerz (*Breadth First Search*).

Inna strategią jest „przeszacowywanie" górnych i dolnych ograniczeń. W metodzie B&B ograniczenia te muszą być poprawne tj. faktycznie ograniczać od góry/dołu odpowiednią wartość. Im bliżej wartości prawdziwej jest ograniczenie, tym więcej węzłów odcinamy. Jeśli jednak przeszacujemy wartość ograniczenia, to będziemy obcinać więcej węzłów niż powinniśmy. W przypadku metody B&B byłby to błąd. Jednakże dla metody BS może korzystnie wpłynąć to na czas działania algorytmu, kosztem jakości wyniku.