Optymalizacja wykorzystania materiału w procesie rozkroju rur

Jakub Pelczar

27 stycznia 2017
v0.1.1

Spis treści

1 Wstęp

2 Knapsack Problem - Problem plecakowy

Problem plecakowy jest zagadnieniem optymailzacyjnym. Problem ten swoją nazwę wziął z analogii do rzeczywistego problemu pakowania plecaka. Rozwiązując ten problem zarówno w praktyce jak i teorii trzeba zachować reguły określające ładowność plecaka dotyczące objętości i nośności plecaka. Knapsack Problem zaczął być intensywnie badany po pionierskiej pracy Dantziga[3] w późnych latach 50 XX wieku. Znalazł on natychmiast zastosowanie w przemyśle oraz w zarządzaniu finansami. Z teoretycznego punktu widzenia, problem plecakowy często występuję jako relaksacja róznorodnych problemów programowania całkowitego[8].

2.1 Zastosowanie

Problem plecakowy stosowany jest nie tylko w sytuacji wynikającej bezpośrednio z nazwy. Znajduje on zastosowanie w wielu dziedzinach życia oraz nauki. Diffi i Helman[4] w 1976 roku oraz Merkle i Helman[7] w 1978 roku zaproponowali problem plecakowy jako podstawę do enkrypcji kluczy prywatnych. Jednakże podejście to w latach późniejszych zostało złamane przez środowisko kryptograficzne i jego miejsce zajęły bardziej odporne algorytmy.

"Knapsack problem" jest stosowany również podczas załadunku kontenerów służacych do przewozu materiałów drogą morską. Ładowność oraz gabaryty ładowanych elementów są ograniczane przez budowę i wytrzymałość kontenera.

Problem ten stosowany jest również w dziedzinie finansów. Jest on podstawowym narzędziem do optymalizacji portfela inwestycyjnego. Poprzez uogólnienie i modyfikacje problemu plecakowego zjawiska ekonomiczne mogą być modelowane z większą dokładnością. Przykładowo możliwe jest zakupienie 0, 1, 2 lub więcej akcji inwestycyjnych, a zakup kolejnych akcji może przynieść obniżenie przychodu.

Wiele problemów związanych z planowaniem może być przyrównana do problemu plecakowego gdzie czas wykonywania operacji na maszynie jest zasobem deficytowym. Jest on szczególnie uwydatniony gdy od aktywności maszyny zależy kapitał przedsiębiorstwa. Poprzez rozwiązanie problemu plecakowego możliwe jest przewidzenie zapotrzebowania na materiały podaczas procesu tak aby warunki zamówinia zostały spełnione[1].

Kolejnym zagadnieniem wynikającym z problemu plecakowego jest problem optymalnego rozkroju, zostanie on przedstawiony w rozdziale section 3.

2.2 Różnorodność problemu plecakowego

Wszystkie elementy z rodziny tego problemu wymagają pewnego zestawu elementów które mogą zostać wybrane w taki sposób że zysk zostanie zmaksymalizowany, a pojemość placaka lub plecaków nie zostanie przekroczona.

Wszystkie typy problemu należą do rodziny problemów $NP - trudnych$ co oznacza, że raczej nispotykane jest rozwiązanie problemu z użyciem algorytmów wielomianowych. Możliwe są różne warinaty problemu zależna od rozmieszczenia elementów oraz ilości plecaków[8]:

- *Problem plecakowy 0-1* - każdy element może być wybrany tylko raz. Problem polega na wyborze n elementów dla których suma profitów p_j jest największa, bez konieczności osiągnięcia całkowitej pojemności c. Może być sformułowany jako problem maksymalizacji:

$$\begin{aligned} &\text{maksymalizacja} && \sum_{j=1}^{n} p_j x_j, \\ &\text{w odniesieniu do} && \sum_{j=1}^{n} w_j x_j \leq c, \\ & && x_j \in \{0,1\}, \qquad j = 1, \ldots, n \end{aligned} \tag{2.1}$$

  gdzie x_j jest wartością binarną. Jeżeli $x_j = 1$ wtedy j-ty element powinien znaleźć się w plecaku, w innym przypadku $x_j = 0$.

- *Ograniczony problem plecakowy* - każdy element może być wybrany ograniczoną ilość razy. Zmianą w obecnym problemie względem problemu 0-1 jest ograniczona m_j ilość elementów j:

$$\begin{aligned} &\text{maksymalizacja} && \sum_{j=1}^{n} p_j x_j, \\ &\text{w odniesieniu do} && \sum_{j=1}^{n} w_j x_j \leq c, \\ & && x_j \in \{0,1 \ldots, m_j\}, \quad j = 1, \ldots, n \end{aligned} \tag{2.2}$$

- *Nieograniczony problem plecakowy* - jest rozszerzeniem problemu ograniczonego o nielimitowaną liczbę dostępnych elementów:

$$\begin{aligned} &\text{maksymalizacja} && \sum_{j=1}^{n} p_j x_j, \\ &\text{w odniesieniu do} && \sum_{j=1}^{n} w_j x_j \leq c, \\ & && x_j \in N_0, \qquad j = 1, \ldots, n \end{aligned} \tag{2.3}$$

  Każda zmienna x_j w metodzie niograniczonej zostanie ograniczona poprzez pojemność c, gdy waga każdego z elementów jest równa przynajmniej jeden. W ogólnym przypadku transformacja problemu nieograniczonego w ograniczony nie przynosi korzyści

- *Problem plecakowy wielokrotnego wyboru* - elementy powinny być wybierane z klas rozłącznych. Problem ten jest generalizacją problemu 0-1. Możliwy jest wybór dokładnie jednego elementu j z każdej grupy N_i, $i = 1, \ldots, k$:

$$\begin{aligned} &\text{maksymalizacja} && \sum_{i=1}^{k} \sum_{j \in N_i} p_{ij} x_{ij}, && \\ &\text{w odniesieniu do} && \sum_{i=1}^{k} \sum_{j \in N_i} w_{ij} x_{ij} \leq c, && \\ & && \sum_{j \in N_i} x_{ij} = 1, && i = 1, \ldots, k, \\ & && x_j \in \{0, 1\}, && i = 1, \ldots, k, \quad j \in N_i. \end{aligned} \tag{2.4}$$

  Zmienna binarna $x_{ij} = 1$ określa że j-ty element został wybrany z i-tej grupy. Ograniczenie $\sum_{j \in N_i} x_{ij} = 1, \quad i = 1, \ldots, k$ wymusza wybór dokładnie jednego elementu z każdej grupy.

- *Wielokrotny problem plecakowy* - mozliwość wypełnienia wielu pleckaków. Jeśli jest możliwość załadowania n elmentów do m pleckaów o róznych pojemnościahc c_i w taki sposób że zysk będzie jak największy:

$$\begin{aligned} &\text{maksymalizacja} && \sum_{i=1}^{k} \sum_{j \in N_i} p_{ij} x_{ij}, && \\ &\text{w odniesieniu do} && \sum_{j=1}^{n} w_j x_{ij} \leq c_i, && i = 1, \ldots, m \\ & && \sum_{j \in N_i} x_{ij} \leq 1, && i = 1, \ldots, k, \\ & && x_j \in \{0, 1\}, && i = 1, \ldots, m, \quad j = 1, \ldots, n. \end{aligned} \tag{2.5}$$

  Zmienna $x_{ij} = 1$ określa że j-ty element powinien zostać umieszczony w i-tym plecaku, podczas gdy ogranicznie $\sum_{j=1}^{n} w_{ij} x_{ij} \leq c_i$ zapewnia że restrykcja dotycząca pojemności plecaka zostanie zachowana. Ogranicznie $\sum_{j \in N_i} x_{ij} \leq 1$ zapewnia że każdy element zostanie wybrany tylko raz.

- *Bin-packing problem* - bardzo często spotykana wersja problemu plecakowego/ Problem ten polega na umieszczeniu n elementów w jak

najmniejszej liczbie opakowań:

$$\begin{aligned}
&\text{maksymalizacja} && \sum_{i=1}^{n} y_i && \\
&\text{w odniesieniu do} && \sum_{j=1}^{n} w_j x_{ij} \leq c y_i, && i = 1, \ldots, n, \\
& && \sum_{i=1}^{n} x_{ij} = 1, && j = 1, \ldots, n, \\
& && y_i \in \{0, 1\}, && i = 1, \ldots, n, \\
& && x_{ij} \in \{0, 1\} \qquad i = 1, \ldots, m, && j = 1, \ldots, n,
\end{aligned} \tag{2.6}$$

gdzie y_i określa czy i-te opakowanie zostało użyte, a x_{ij} stanowi czy j-ty element powinen zostać umieszczony w i-tym opakowaniu

- *Welokrotnie ograniczony problem plecakowy* - najbardziej ogólny typ który jest problemem programowania całkowitego z dodatnimi współczynnikami:

$$\begin{aligned}
&\text{maksymalizacja} && \sum_{j=1}^{n} p_j x_j, && \\
&\text{w odniesieniu do} && \sum_{j=1}^{n} w_j x_j \leq c_i, && i = 1, \ldots, m, \\
& && x_j \in N_0, && j = 1, \ldots, n.
\end{aligned} \tag{2.7}$$

2.3 Możliwe rozwiązania

Problem plecakowy należy do grupy problemów $\mathcal{NP}$-Trudnych. Rozwiązanie problemów z tej grupy jest co najmniej tak trudne, jak rozwiązanie każdego problemu z całej klasy $\mathcal{NP}$. Problem $\mathcal{NP}$-Trudny to problem obliczeniowy dla którego znalezienie rozwiązania problemu nie jest możliwe z wielomianową złożonością obliczeniową. Problemy $\mathcal{NP}$-Trudne obejmują zarówno problemy decyzyjne jak również problemy przeszukiwania czy też problemy optymalizacyjne.

Rozwiązanie problemu plecakowego jest możliwe przy użyciu różnych metod:

- *Metoda podziału i ograniczeń* - Metoda ta często jest stosowana do problemu plecakowego od momentu gdy Kolesar [6] zaprezentował pierwszy algorytm w 1967 roku.

- *Programownaie dynamiczne* - Gdy zostaną dodane warunki brzegowe wtedy algorytm ten staje się "zaawansowaną" formą metody podziału i ograniczeń.

- *Relaksacja przestrzeni stanów* - relaksacja programowani dynamicznego gdzie współczynniki są skalowane przez pewną stałą wartość.

2.3.1 Metoda podziału i ograniczeń

Algorytm ten polega na wypisaniu wszystkich możliwych rozwiązań używając struktury drzewiastej. Algorytm przechodzi kolejno po gałęziach drzewa które reprezentują podzbiory rozwiązania. Każda gałąź jest sprawdzana zadanymi warunkami brzegowymi i jest odrzucana jeśli nie poprawi rozwiąznaia. Przedstawione zostanie rozwiązanie nieograniczonego problemu plecakowego (eq. (2.3)) [2]. Współczynniki $w_1, \ldots, w_m, p_1, \ldots, p_m$ oraz c są nieujemne. Stosunek p_j/w_j jest wartością jednej jednostki długości j-tego elementu. Stosunek ten jest *wydajnością* zmiennej x_j. Pierwszym krokiem algorytmu jest posortowanie zmiennych w porządku malejącym względem wydajności:

$$p_1/w_1 \geq p_2/w_2 \geq \cdots \geq p_m/w_m \tag{2.8}$$

Dla posortowanych elementów każde rozwiązanie optymalne (eq. (2.3)) spełnia warunek:

$$c - \sum_{j=1}^{m} w_j x_j < w_m \tag{2.9}$$

Głównym elementem algorytmu jest stworzenie drzewa wyliczeń. Przykładowo dla problemu który zawiera 13 rozwiązań:

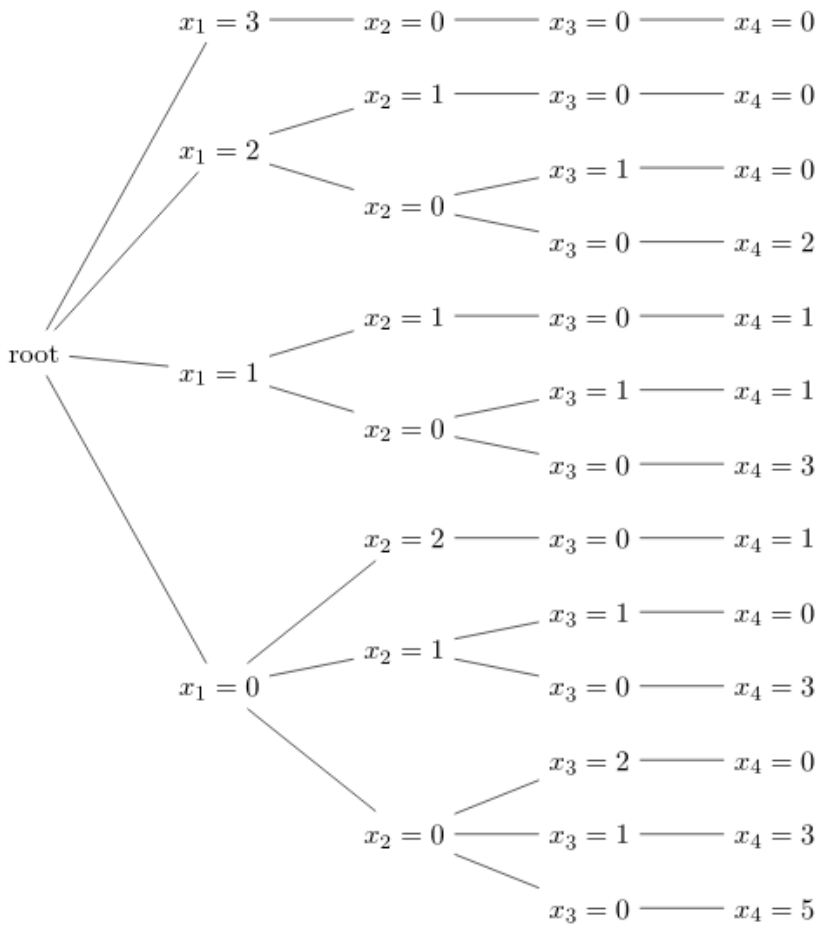
$$\begin{aligned} &\text{maksymalizacja} && 4x_1 + 5x_2 + 5x_3 + 2x_4 \\ &\text{w odniesieniu do} && 33x_1 + 49x_2 + 51x_3 + 22x_4 \leq 120 \\ & && x_j \in N_0 \end{aligned}$$

drzewo będzie miało 13 liści (fig. 2.1). Jeśli z jednego węzła wychodzi więcej niż jedna gałąź wówczas potomek o większej przechowywanej wartości jest umieszczany wyżej. Każdy następny węzeł jest obliczany według wzoru:

$$\begin{aligned} x_j &= \lfloor (c - \sum_{i=1}^{j-1} w_i x_i)/w_j \rfloor \qquad i = 1, 2, \ldots, m \\ x_1 &= \lfloor c/w_1 \rfloor \end{aligned}$$

Aby odrzucić węzły które nie mogą polepszyć rozwiązania i pozostawić tylko te gałęźie które dają szansę na rozwiązanie optymalne, należy przejść następujące kroki: Poszukując rozwiązania $x_1, x_2, \ldots, x_m$ ustawione zostaje $k = m - 1$. Jeśli zachodzi taka potrzeba zmienna k jest dekrementowana dopóki nie zostanie znalezione $x_k > 0$. Wówczas $x_k = x_{k-1}$, a wartości $x_{k+1}, x_{k+2}, \ldots, x_m$ są otrzymywane ze wzoru (section 2.3.1).

Dla bieżącego rozwiązania $x_1^*, \ldots, x_m^*$ zachodzi $\sum_{i=1}^{m} p_i x_i^* = M$. Maksymalne k takie że $k \leq m - 1$ oraz $x_k > 0$ zostaje określone przechodząc



Rysunek 2.1: Drzewo wyliczeń możliwych rozwiązań

od węzłów $x_1, x_2, \ldots, x_m$ w stronę korzenia. Podobnie jak wcześniej niech $\bar{x}_i = x_i$ dla $i = 1, 2, \ldots, k-1$ oraz $\bar{x}_k = x_k - 1$ będą zmiennymi kandydującymi do rozwiązania. Aby okreslić czy $\bar{x}_i$ polepszy rozwiązanie x_i^*. Zgodnie z (eq. (2.8)) dla każdej zmiennej $x_{k+1}, x_{k+2}, \ldots, x_m$ wydajność wynosi maksymalnie p_{k+1}/w_{k+1}, tak więc

$$\sum_{i=k+1}^{m} p_i \bar{x}_i \leq \frac{p_{k+1}}{w_{k+1}} \sum_{i=k+1}^{m} w_i \bar{x}_i$$

połączone razem z (eq. (2.3)) zwraca:

$$\sum_{i=1}^{m} p_i \bar{x}_i \leq \sum_{i=1}^{m} w_i \bar{x}_i + \frac{p_i}{w_i}(c - \sum_{i=1}^{k} w_i \bar{x}_i). \tag{2.10}$$

Zgodnie z zasadami drzewa wyliczeń, nierówność

$$\sum_{i=1}^{k} p_x \bar{x}_i + \frac{p_{k+1}}{w_{k+1}}(c - \sum_{i=1}^{k} w_i \bar{x}_i) \leq M \tag{2.11}$$

określa że ścieżka $\bar{x}_1, \ldots, \bar{x}_k$ jest niegorsza niż pozostałe. Jeśli wszystkie współczynniki $p_1, \ldots, p_m$ są dodatnimi liczbami całkowitymi, wówczas również M jest liczbą całkowitą, a silna nierówność (2.11) może zostać zastąpiona słabą

$$\sum_{i=1}^{k} p_x \bar{x}_i + \frac{p_{k+1}}{w_{k+1}}(c - \sum_{i=1}^{k} w_i \bar{x}_i) < M + 1 \tag{2.12}$$

Dla wcześniejszego przykładu powyższy krok mający na celu redukcję drzewa, następuje inicjalizacja zmiennych:

$$\begin{aligned} x_1 &= \lfloor 120/33 \rfloor = 3 \\ x_2 &= \lfloor (120 - 99)/49 \rfloor = 0 \\ x_3 &= \lfloor (120 - 99)/51 \rfloor = 0 \\ x_4 &= \lfloor (120 - 99)/99 \rfloor = 0 \end{aligned}$$

Z powyższego wynika że początkowe rozwiązanie to $x_1^* = 3, x_2^* = x_3^* = x_4^* = 0$ oraz $M = 12$. Początkowo $k = 3$, następnie wystepuje redukcja k dopóki $k = 1$ z $x_k > 0$ nie zostanie znalezione. Wówczas $x_1 = 3$ zostaje zaminione na $x_1 = 2$. Przed sprawdzeniem gałęzi $x_1 = 2$ przeprowadzony zostaje test (2.12) z $k = 1$ oraz $\bar{x}_1 = 2$. Wówczas lewa strona nierówności wynosi

$$8 + \frac{5}{49}(120 - 66) = 13.5$$

i jest nie mniejsza niż $M + 1 = 13$, gałąź może być warta sprawdzenia. Następnie obliczenie kolejnej ścieżki

$$\begin{aligned} x_2 &= \lfloor (120 - 66)/49 \rfloor = 1 \\ x_3 &= \lfloor (120 - 115)/51 \rfloor = 0 \\ x_4 &= \lfloor (120 - 115)/99 \rfloor = 0 \end{aligned}$$

i zastąpienie poprzedniego rozwiązania przez $x_1^* = 2, x_2^* = 1, x_3^* = x_4^* = 0$ oraz $M = 13$. Powtórzony zostaje krok z redukcją $k = 3$ dopóki $k = 2$ oraz dopóki nie zostanie znalezione $x_k > 0$. Wówczas $x_2 = 1$ zostaje zamienione na $x_2 = 0$. Aby określić czy ścieżka $x_1 = 2, x_2 = 0$ jest warta sprawdzenia, zostaje przeprowadzony test (2.12) z $k = 2$ oraz $\bar{x}_1 = 2, \bar{x}_2 = 0$. Lewa strona nierówności wynosi

$$8 + \frac{5}{51}(120 - 66) = 13.3$$

co jest mniejsze niż $M + 1 = 14$, więc gałąź ta jest odcinana. Następnie k dalej jest dekrementowane, a kroki sa powtarzane. Dla $x_1 = 1$ wynik testu to $12.9 < 14$, a dla $x_1 = 0$ wynik to $12.2 < 14$ więc gałęzie te są odcinane. Tak więc optymalnym rozwiązaniem jest $x_1^* = 2, x_2^* = 1, x_3^* = x_4^* = 0$. Drzewo wyliczeń zostało zredukowane do postaci **??**. Jeśli odcięta jest gałąź $\bar{x}_1, \ldots, \bar{x}_k$ wówczas odcięt również zostaje pozostała część gałęzi bez przeprowadzania dodatkowych testów.

Algorytm dla metody podziału i ograniczeń do rozwiązania problemu plecakowego, został przedstawiony poniżej

Algorytm 1 Metoda podziału i ograniczeń - problem plecakowy

```
1:  M := 0
2:  k := 0
3:  for j := k+1 TO m do
4:      x_j = ⌊(c − Σ_{i=1}^{j−1} w_i x_i)/w_j⌋
5:  k := m
6:  if Σ_{i=1}^{m} p_i x_i > M then
7:      M := Σ_{i=1}^{m} p_i x_i
8:      for j := 1 TO m do
9:          x_j^* = x_j
10: if k = 1 then
11:     stop
12: else
13:     k = k − 1
14: if x_k = 0 then
15:     idź do linii 10
16: else
17:     x_k = x_k − 1
18: if !Σ_{i=1}^{k} p_x x̄_i + (p_{k+1}/w_{k+1})(c − Σ_{i=1}^{k} w_i x̄_i) < M + 1 then
19:     idź do linii 3
20: else
21:     idź do linii 10
```

2.3.2 Programowanie dynamiczne

Metoda ta używana jest w przypadku gdy problem można podzielić na małe podproblemy które można rozwiązać rekursywnie. Rozwiązanie optymalne podproblemu jest również optymalnym rozwiązaniem problemu głównego. Przedstawione zostanie rozwiązanie problemu plecakowego rodzaju 0-1 [5].

Jeśli elementy są oznaczone jako $1, \ldots, n$ wtedy podproblem będzie odpowiedzialny za znalezienie optymalnego rozwiązania dla $S_k = \{1, 2, \ldots, k\}$. Niemożliwe jest opisanie rozwiązania końcowego S_n na podstawie podproblemów S_k. Rekursywne sformułowanie podproblemu:

$$B[k,w] = \begin{cases} B[k-1,w] & \text{jeśli} \quad w_k > w, \\ max\{B[k-1,w], B[k-1,w-w_k] + b_k\} & \text{jeśli} \quad w_k \leq w. \end{cases} \tag{2.13}$$

Z powyższego równania wynika że najlepszy podzbiór podproblemu S_k z całkowitą wagą w jest najlepszym podzbiorem dla S_{k-1} którego całkowita waga wymosi w lub jest najlepszym podzbiorem dla S_{k-1} którego całkowita waga wynosi $w - w_k$ plus k-ty element. Złożoność programowania dynamicznego to $O(n * W)$. Algorytm jako dane wejściowe przyjmuje maksymalną wartość ciężaru W, oraz dwie listy: listę wag $w_1, \ldots, w_n$ oraz odpowiadającą jej listę zysku $b_1, \ldots, b_n$.

Algorytm 2 Programowanie dynamiczne - problem plecakowy 0-1

```
1:  for w := 0 TO W do
2:      B[0,w] := 0
3:  for i := 1 TO n do
4:      B[i,0] := 0
5:  for i := 1 TO n do
6:      for w := 0 TO W do
7:          if w_i ≤ w then
8:              if b_i + B[i-1, w-w_i] > B[i-1, w] then
9:                  B[i, w] := b_i + B[i-1, w-w_i]
10:             else
11:                 B[i, w] := B[i-1, w]
12:         else
13:             B[i, w] := B[i-1, w]
```

3 Cutting Stock Problem - Problem optymalnego rozkroju

4 Metoda "Brutal Force"

4.1 Algorytm wyjściowy

Metoda ta opiera się zarówno na intuicji jak i na rozwiązaniu zaproponowanym przez Dantziga dla problemu plecakowego [3]. Jest to metoda która w prosty sposób - nie używając złożonych modeli matematycznych, pozwala osiągnąć optymalny rozkrój materiału.

Pierwszym krokiem jest posortowanie elementów wyściowych malejąco względem ich długości $l_1 \geq l_2 \geq ... \geq l_m$ i umieszczenie w ten sposób w kolejce.

Drugim krokiem jest pobranie pierwszego elementu z kolejki i sprawdzenie, jak wiele razy jego długość zawiera się w długości elementu bazowego. Obliczone zostaje ile materiału pozostało w elemencie bazowym po docięciu najdłuższych elementów. Następnie pobierany jest kolejny odcinek z kolejki. Następuje sprawdzenie ile razy zawiera się on w pozostałej długości.

$$\begin{aligned} a_1 &= [L/l_1], \\ a_2 &= [(L - l_1 a_1)/l_2], \\ a_3 &= [(L - (l_1 a_1 + l_2 a_2))/l_3], ... \end{aligned} \tag{4.1}$$

Kroki te powtarzane są dopóki kolejka się nie skończy.

Każdy element wyjściowy posiada określoną liczebność jaką powinien osiągnąć pod koniec procesu cięcia. Jeśli na danym etapie procesu cięcia wymagana liczba elementów danego typu spada do zera, wówczas jest on pomijany w dalszej pracy algorytmu. Koniecznie jest sprawdzenie czy liczba uzyskanych elementów danego typu jest mniejsza lub równa od wymaganej:

- Jeśli stwierdzenie jest prawdziwe - długość z której elementy są wycinane zostanie zmniejszona o liczbę wystąpień elementu pomnożoną przez jego długość, a licznik wymaganych odcinków danej długości zostanie zmniejszony o odpowiednią liczbę wystąpień
- Jeśli stwierdzenie jest fałszywe - długość z której elementy są wycinane zostanie zmniejszona o liczbę pozostałych wykrojów pomnożoną przez długość elementu, a licznik wymaganych odcinków danej długości zostanie ustawiony na zero.

Po zakończeniu przebiegu algorytmu dla jednego układu rozkroju, można określić ile razy będzie on użyty. Zostaje to wyznaczone poprzez obliczenie

$$g = floor\{min\{z_i/a_i\}\}, i \in [0..m], g \in Z \tag{4.2}$$

gdzie g to liczba ile razy dany schemat może zostać użyty, z to liczbność wyjściowego elementu i która pozostała do wycięcia, a to ilość wykrojów elementu i w bierzącym układzie, m to liczba długości umieszczonych w

rozkroju. Następnie licznik wymaganych odcinków elemntu i zostaje zmniejszony o ga_i.

Cały proces powtarzany jest do momentu aż wszytskie wymagane elementy zostaną wycięte.

4.2 Rozszerzenie o szerokość cięcia

W warunkach rzeczywistych elementy wycinane są za pomocą ostrza które ma niezerową grubość. Wówczas metodę obliczania należy rozszerzyć jeśli ma odpowiadać warunkom rzeczywistym. Szerokość cięcia wlicza się w odpad. Jest kilka przypadków wliczania szerokości ostrza.

Jeżeli element jest równy długości bazowej wówczas nie wlicza się szerokości cięcia. Natomiast jeżeli materiał bazowy ma zostać pocięty na kilka elmentów wówczas do każdego dolicza się szerokość cięcia. Szczególnym przypadkiem jest, gdy ostatni element wraz z szerokością ostrza jest dłuższy niż długość odcinka, który został po wycięciu wcześniejszych elementów.

Gdyby szerokość cięcia nie zostałą uwzględniona w obliczeniach wówczas dla elementu wejściowego o długości 6000mm i wymaganych odcinkach 4500mm oraz 1500mm, obie długości zostały wycięte z jednego segmentu materiału bazowego. Skutkiem takiego postępowania byłby element krótszy o szerokość ostrza. Zazwyczaj długość ta może być akceptowana jako toleracncja dokładności maszyny. Jednak dla poprawności obliczeń wielkość ta powinna zostać uwzględniona.

4.3 Rozszerzenie o wiele długości bazowych

Dla zmniejszenia odpadu można użyć kilku długości bazowych. Rozszerzenie to wprowadza następująca zmianę algorytmu: obliczenia układu muszą zostać powtórzone dla każdego elementu wejściowego. Następnie wybierany jest ten rozkrój, który daje mniejszy odpad. Modyfikacja ta znacząco wpływa na wydajność metody. Jeżeli n oznacza złożoność obliczeniową podstawowego algorytmu, a m oznacza liczbę odcinków wejściowych, wówczas nowa złożonośc obliczeniowa wynosi $m * n$.

4.4 Rozszerzenie o cenę materiału wsadowego

Rozszerzenie to wprowadza zmianę koncepcyjną. Każdy element bazowy posiada cenę za metr bieżący materiału, umożliwia to obliczenie kosztu odpadu i wybranie tańszej opcji wykroju.

4.5 Przykład

1. Dane wejściowe
    - 6000mm - 3$/mb
    - 7000mm - 2$/mb
    - szerokość cięcia: 10mm
2. Dane wyjściowe
    - 1x3500mm
    - 1x3000mm
    - 3x2000mm
    - 5x500mm
3. Przebieg algorytmu
    - Pierwszy rozkrój
        - 3500mm mieści się raz w 6000mm. Zostaje $2500-10 = 2490$mm.
        - 3000mm nie mieści się w 2490mm.
        - 2000mm mieści się raz w 2490mm. Zostaje $490-10 = 480$mm.
        - 500mm nie mieści się w 480mm.
        - Rozkrój 6000mm: 3500mm, 2000mm. Odpad $6000 - 5500 = 500 * 0.003 = 1.5$\$
        - ————
        - 3500mm mieści się dwa razy w 7000mm. Dostępny jest jeden odcinek 3500mm. Zostaje $3500 - 10 = 3490$mm.
        - 3000mm mieści sie raz w 3490mm. Zostaje $490-10 = 480$mm.
        - 2000mm nie mieści się w 480mm.
        - 500mm nie mieści się w 480mm.
        - Rozkrój 7000mm: 3500mm, 3000mm. Odpad $7000 - 6500 = 500 * 0.002 = 1.0$\$
        - ————
        - Wybrano rozkrój 3500mm, 2000mm na długości 7000mm ze względu na mniejszy koszt odpadu.
        - Do realizacji pososstało: 0x3500mm; 0x3000mm; 3x2000mm; 5x500mm
    - Drugi rozkrój
        - 2000mm mieści się trzy razy w 6000mm. Uwzględniając szerokość cięcia - zostaną użyte tylko dwa elementy od długości 2000mm. Zostaje $2000 - 2 * 10 = 1980$mm.

  - 500mm mieści się trzy razy w 1980mm. Zostaje $480 - 3 * 10 = 450$mm.
  - Rozkrój 6000mm: 2x2000mm, 3x500mm. Odpad $6000 - 5500 = 500 * 0.003 = 1.5$\$
  - ———
  - 2000mm mieści się trzy razy w 7000mm. Zostaje $1000 - 3 * 10 = 970$mm.
  - 500mm mieści się raz w 970mm. Zostaje $470 - 10 = 460$mm.
  - Rozkój 7000mm: 3x2000mm, 500mm. Odpad $7000 - 6500 = 500 * 0.002 = 1.0$\$
  - ———
  - Wybrano rozkrój 3x2000mm, 500mm na długości 7000mm ze względu na mniejszy koszt odpadu
  - Do realizacji pososstało: 0x3500mm, 0x3000mm, 0x2000mm, 4x500mm
- Trzeci rozkrój
  - 500mm mieści się dwanaście razy w 6000mm. Dostępne są cztery element 500mm. Zostaje $6000 - 4 * 500 - 4 * 10 = 3960$mm.
  - Rozkrój 6000mm: 4x500mm. Odpad $6000 - 4 * 500 = 4000 * 0.003 = 12$\$
  - ———
  - 500mm mieści się czternaście razy w 7000mm. Dostępne są cztery elementy 500mm. zostaje $7000 - 4 * 500 - 4 * 10 = 4960$mm
  - Rozkrój 7000mm: 4x500mm. Odpad $7000 - 4 * 500 = 5000 * 0.002 = 10$\$
  - ———
  - Wybrano rozkrój 4x500 na długości 7000mm ze względu na mniejszy koszt odpadu
  - Do realizacji pososstało: 0x3500mm, 0x3000mm, 0x2000mm, 0x500mm
- Podsumowanie
  - Rozkroje : 3500mm, 2000mm na długości 7000mm; 3x2000mm, 500mm na długości 7000mm; 4x500 na długości 7000mm.
  - Suma odpadów: $6000 * 0.002 = 12$\$

4.6 Podsumowanie

Przedstawiony algorytm jest intuicyjny oraz zwraca poprawne wyniki. Główną wadą jest brak świadomości o następnym kroku oraz kolejnych wykrojach. Dla przykładu: Zosatło 1000mm materiału, do dyspozycji (z długości mniejszych niż 1000mm) jest odcinek 900mm oraz dwa elementy 480mm. Algorytm przydzieli odcinek 900mm, jednak lepszym wyborem byłoby użycie dwóch odcinków 480mm.

5 Metoda "Delayed Column Generation"

5.1 Algorytm

$$L \geq l_1 a_1 + \cdots + l_m a_m \tag{5.1}$$

$$b_1 a_1 + \cdots + b_m a_m > c \tag{5.2}$$

1. Określnie m poczatkowych rokrojów i ich kosztu w następujący sposób: dla każdego i wybranie długości bazowej L_j dla której $L_j > l_i$ i określenie i-tego rokroju jako wycięcie $a_{ii} = [L_j/l_i]$ elementów o długości l_i z długości L_j. Koszt i-tego rozkroju będzie równy kosztowi c_j długości L_j z której i-ta operacja wycina odcinki o długości l_i.

2. Uformowanie macierzy $\boldsymbol{B}$

$$\begin{matrix} 1 & -c_1 & -c_2 & \dots & -c_m \\ 0 & a_{11} & 0 & \dots & 0 \\ 0 & 0 & a_{22} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{mm} \end{matrix}$$

   gdzie a_{ii} jest ilością odcinków o długości l_i wyciętych w i-tym rozkroju z długości bazowej o koszcie c_j. Ostatnie m kolumn jets powiązane z rozkrojami. Dane te będą aktualizowane gdy zostanie znaleziony wynik który poprawi rozwiązanie.

   Utworzenie m $m+1$ wymiarowych wektorów kolumnowych $\boldsymbol{S_1}$,...,$\boldsymbol{S_m}$ odnoszących się do zmiennych dodatkowych, gdzie $\boldsymbol{S_i}$ posiada same zera z wyjątkiem wiersza $(i+1)$ w którym jest -1. Dodatkowo utworzenie $m+1$ wymiarowego wektora kolumnowego $\boldsymbol{N'}$ który jako pierwszy element przyjmuje 0, a w następnych i-tych wierszach posiada wartośic N_i.

   Obliczenie $\boldsymbol{B}^{-1}$ która wynosi:

$$\begin{matrix} 1 & c_1/a_{11} & c_2/a_{22} & \dots & c_m/a_{mm} \\ 0 & 1/a_{11} & 0 & \dots & 0 \\ 0 & 0 & 1/a_{22} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1/a_{mm} \end{matrix}$$

   Niech $\boldsymbol{N} = \boldsymbol{B}^{-1} \cdot \boldsymbol{N'}$. Sprawdzając czy pierwszy element z $\boldsymbol{B}^{-1} \cdot \boldsymbol{P}$ jest dodatni można określić czy istnieje możliwość polepszenia rozwiązania. Wektor kolumnowy $\boldsymbol{P}$ jets wektorem złożonym ze zmiennych nieuzytych w bieżącym rozwiązaniu, np. pierwszy element to negatywny koszt, a pozostałe m wierszy jest równe zmiennym a_{ij}.

3. Z powyższego puntku wynika że jeśli i-ta zmienna dodatkowa która nie wchodzi w skład rozwiązania może ulepszyć rozwiązanie wtedy i tylko wtedy gdy $(i+1)$ element pierwszego wiersza $\boldsymbol{B}^{-1}$ jest ujemny.

4. Jeśli nie jest możliwe polepszenie rozwiązania nalezy określić czy wprowadznie nowego rozkroju ulepszy bieżące rozwiązanie. Jets to możliwe poprzez sprawdznie czy dla L z kosztem c istnieje rozwiązanie nierówności 5.1 oraz 5.2, gdzie $b_1, \ldots, b_m$ to ostatnie m elementów w piwerwszym wierszu $\boldsymbol{B}^{-1}$. Jeśli te nierównoście nie posiadają rozwiąania dla dowolnej długości $L_1, \ldots, L_k$ z kosztem odpowiednio $c_1, \ldots, c_m$ wtedy bieżące rozwiązanie jest minimum. Rozwiązanie i jego koszt jest określone poprzez $\boldsymbol{N}$, gdzie pierwszy wiersz to koszt, a pozostałe m wierszy jest, w kolejności, odpowiednimi wartościami m-tej kolumny z $\boldsymbol{B}^{-1}$.

   Jeśli nowy rozkrój poprawi rozwiązanie wtedy formowany jest nowy wektor $\boldsymbol{P}$ ze współczynnikami, w kolejności $-c, a_1, a_2, \ldots, a_m$.

5. Wprowadznie zarówno dodatkowej zmiennej jak i nowego rozkroju może poprawić rozwiązanie. W obu przypadkach $\boldsymbol{P}$ będzie kolumnowym wektorem ze zmiennymi. Dla określenia nowych $\boldsymbol{B}^{-1}$ oraz $\boldsymbol{N}$ które opisują ulepszone rozwiązanie i jego koszt, co pozawala na przejście przez kkroki 3, 4 oraz kontynujacje kroku 5 w nastepujący sposób: Obliczenie $\boldsymbol{B}^{-1} \cdot \boldsymbol{P}$ - niech elementy wynikime będą elementy $y_1, \ldots, y_m, y_{m+1}$ oraz niech elementami bierzącego wektora $\boldsymbol{N}$ będą $x_1, \ldots, x_m, x_{m+1}$. Ustalenie i, $i \geq 2$ dla każdego $y_i > 0$, $x_i \geq 0$ oraz x_i/y_i jest najminiejsze i przypisanie tej wartości do zmiennej k. Minimalny stosunek powinien być zerem aby można było wykorzystać metodę degeneracji.

   Jeśli stosunek nie jest równy zero wtedy k-ty element wektora $\boldsymbol{P}$, y_k będzie elementem wokół którego zajdzie eliminacja Gaussa odbywająca się równocześnie na $\boldsymbol{B}^{-1}$, $\boldsymbol{B}^{-1} \cdot \boldsymbol{P}$ oraz $\boldsymbol{N}$. Eliminacja ta przebiega na macierzy $(m+1) \times (m+3)$ wymiarowej $\boldsymbol{G}$ uformowanej z $\boldsymbol{B}^{-1}$ poprzez dołączenie kolumn $\boldsymbol{B}^{-1} \cdot \boldsymbol{P}$ oraz $\boldsymbol{N}$. Pierwsze $m+1$ kolumn $\boldsymbol{G'}$ formuje nową macierz $\boldsymbol{B}^{-1}$, a kolumna $m+2$ to nowy wektor $\boldsymbol{N}$. Zależność między kolumnami $\boldsymbol{B}^{-1}$ a rozkrojami lub zmiennymi dodatkowymi jets aktualizowana poprzez usunięcie k-tej kolumny i podmienieniu jej na nowy rozkruj lub zmienną dodatkową.

   Degeneracja w razie wystąpienia może być obsłużona w tradycyjny sposób. Pewne środki ostrożności powinny zostać podjęte w celu uniknięcia cykliczności. Nowa kolumna $\boldsymbol{N}^1$ z dodatnimi elementami $x'_1, \ldots, x'_{m+1}$ która jest niezależna pd $\boldsymbol{N}$ jest dołączana do $\boldsymbol{G}$ i wybór takiego $y_i > 0$ dla którego $x_i = 0$ który jest elementem osiowym jets dokonywany na podstawie takiego i dla którego $x'_i > 0$ oraz x'_i/y_i jest najmniejsze. Gdy element osiowy zostanie wybrany, wówczas eliminacja Gaussa zachodzi tak jak w poprzednim przypadku na powiększonej macierzy

$\boldsymbol{G}$. Dodatkowa kolumna jest przechowywane w $\boldsymbol{G}$ dopóki istnieje takie i dla którego x_i/y_i jest dodatnie i skończone, jeśli warunek ten jest spełniony wówczas kolumna zostaje usunięta. Powinno to nastąpić w przypadku gdy nie istnieje takie i dla którego x_i/y_i oraz x'_i/y_i są dodatnie i skończone. Wówczas powinna zostać dodana kolumna $\boldsymbol{N}^2$ nizależna od $\boldsymbol{N}$ oraz $\boldsymbol{N}^1$. Podobnie dowolna liczba kolumn może zostac dodana i usunięta gdy przestanie byc potrzebna. Dopóki kolumny są niezależne w czasie dodawania i pozostają takie po eliminacji Gaussa, nie potrzeba więcej jak m nowych kolumn. Każda dodana kolumna definiuje nowy problem liniowy który eliminuje problem cykliczności tak długo aż degeneracja nie wystąpi.

5.2 Metody użyte w implementacji

5.2.1 Dwufazowa metoda simplex

5.2.2 Metoda podziału i ograniczeń

5.3 Przykład

6 Wyniki

6.1 Porównanie

6.2 Wnioski

6.3 Podsumowanie

7 Opis implementacji

7.1 Architektura

7.2 Java

7.3 Kotlin

7.4 JavaFX

8 Zakończenie

Spis rysunków

Literatura

[1] J. J. Bartholdi. The knapsack problem. In D. Chhajed and T. J. Lowe, editors, *Building Intuition*, chapter 2, pages 19 – 31. Springer US, 2008.

[2] V. Chvatal. *Linear Programming.* W.H. Freeman and Company, New York, 1984.

[3] G. B. Dantzig. Discrete variable extremum problems. *Operations Research*, 2:266 – 288, 1957.

[4] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644 – 654, 1976.

[5] S. Goddard. Lecture about dynamic programming 0-1 knapsack problem. `http://cse.unl.edu/~goddard/Courses/CSCE310J/`.

[6] P. J. Kolesar. A branch and bound algorithm for the knapsack problem. *Managment science*, 13:723 – 735, 1967.

[7] R. Merkle and M. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, 24:525 – 530, 1978.

[8] D. Pisinger. *Algorithms for Knapsack Problems.* PhD thesis, Dept. of Computer Science, University of Copenhagen, 1995.