Maksymalizacja przychodów ze sprzedaży pieczywa za pomocą algorytmu programowania liniowego

Programowanie liniowe to metoda, której celem jest optymalizacja wyników. Możemy maksymalizować zysk lub przychody, dążyć do minimalizacji ryzyka utraty ciągłości produkcji lub minimalizacji kosztów transportu. Optymalizować można praktycznie wszystko.

Zazwyczaj procesy produkcyjne w piekarniach, procesy dostarczania pieczywa i kompletowania produktów, wszystkie codzienne procesy związane z działalnością operacyjną są skomplikowane i trudne do reorganizacji. Procesy te wchodzą w szereg wzajemnych interakcji rywalizując o ograniczone zasoby piekarni. Często zadajemy sobie pytanie: czy nie lepiej produkować mniej tego produktu, a więcej tego drugiego? Czy warto jeździć tak daleko z pieczywem zamiast skoncentrować się na bliższej sprzedaży?

W praktyce wszystkie procesy same się jakoś układają i piekarnia działa dalej. Jednak naiwne jest oczekiwanie, że wszystkie procesy, które się kiedyś samoistnie ułożyły i jakoś wzajemnie dopasowały, są ustawione optymalnie. Być może każdego dnia tracimy setki, a może tysiące złotych na niepotrzebną sprzedaż czy niewłaściwą organizację załadunku.

Aby wszystko lepiej zorganizować nie trzeba być matematykiem. Wystarczy nauczyć się opisywać procesy w swojej piekarni za pomocą prostych wzorów.

Na przykład chleb A sprzedawany jest za 2,7 zł, a chleb B za 3,5 zł.

Ilość sprzedawanego chleba A oznaczamy jako x_A, a B jako x_B.

Wzór na połączone przychody ze sprzedaż obu chlebów będzie miał postać:

$2{,}7x_A + 3{,}5x_B \rightarrow max$

Do wzoru dopisałem strzałkę max, ponieważ zazwyczaj chcemy osiągać maksymalne przychody.

Przychody są ograniczone liczbą sprzedawanych bochenków x_A i x_B. Mamy określone zdolności produkcyjne, a lokalna społeczność ma ograniczone potrzeby żywieniowe. Powinniśmy więc sformułować ograniczenia dla równania sprzedaży, np.: $0 < x_A < 200$. Tak formułuje się najprostsze zadanie programowania liniowego. W jego wyniku możemy otrzymać optymalne proporcje w produkcji chlebów A i B, które pozwolą nam zarobić najwięcej.

Jednym z najprostszych zadań optymalizacji jest problem wyboru. Przedsiębiorca musi podjąć decyzję na tak lub na nie, mając do wyboru wiele alternatywnych propozycji.

Przykład problemu wyboru

Przedsiębiorstwo zrzeszające dwie nowoczesne piekarnie ogłosiło przetarg na codzienną dostawę pieczywa do sieci supermarketów. Analitycy wyliczyli, że zrzeszenie piekarni jest w stanie zaopatrywać tylko trzy sieci supermarketów. Dostawami zainteresowały się cztery sieci sprzedaży. Każda z sieci przedstawiła propozycje finansowe za miesięczne dostawy pieczywa.

oferty supermarketów:	Kontrakt A	Kontrakt B	Kontrakt C
Fenix	129 tys. zł	115 tys. zł	125 tys. zł
Jaszczurka	121 tys. zł	119 tys. zł	122 tys. zł
PDT	118 tys. zł	110 tys. zł	114 tys. zł
REKSIO	120 tys. zł	126 tys. zł	126 tys. zł

Celem naszego zadania jest maksymalizacja łącznych przychodów ze sprzedaż pieczywa. Zadanie wydaje się łatwe. Wystarczy wybrać tę sieć detaliczną, która zaoferuje najwyższą cenę za dostawy. Celem jest uzyskanie najwyższych łącznych przychodów.

Interpretacja matematyczna problemu

Celem ćwiczenia jest podpisane trzech kontraktów na dostawę pieczywa tak, aby osiągać co miesiąc najwyższy wspólny przychód ze sprzedaży. W poniższej tabeli kontrakty są zapisane jako kolumny, zaś supermarkety jako wiersze.

	K1	K2	K3
A	129	115	125
B	121	119	122
C	118	110	114
D	120	126	126

Można sobie wyobrazić, że powstaje siatka, gdzie K1, K2 i K3 to dostawy, a litery A, B, C, D to sklepy.

Każda wartość w siatce to oferowana kwota za dostawy miesięczne. Algorytm musi więc rozpatrzyć wszystkie ceny wszystkich kontraktów oferowane przez wszystkich chętnych do zakupu.

Ponieważ sieci detaliczne mogą podpisać tylko jeden kontrakt, algorytm musi wybrać odpowiednią konfigurację ofert.

System musi więc dla każdej kratki odpowiedzieć na pytanie:

0 – sieć detaliczna nie podpisze kontraktu,

1 – sieć detaliczna podpisze kontrakt.

Komputer musi więc odpowiedzieć na każdą z niewiadomych w naszej tabelce. Niewiadome oznaczamy jako x_{wk}.

	K1	K2	K3
A	x11	x12	x13
B	x21	x22	x23
C	x31	x32	x33
D	x41	x42	x43

Kontrakty są trzy, a chętnych do ich zakupu jest czterech. Jedna z sieci sklepów nie podpisze więc kontraktu.

Konieczne jest więc stworzenie dodatkowej kolumny oznaczającej brak podpisania kontraktu. Tego wymaga matematyka, ponieważ dla modelu ilość kolumn musi równać się ilości wierszy w tabeli.

Tworzenie równań matematycznych

Programowanie liniowe jest metodą wskazywania optymalnych rozwiązań za pomocą równań i nierówności matematycznych. Zoptymalizować można każdy, nawet najbardziej zawiły problem ekonomiczny. Cały problem polega na tym, aby nauczyć się zapisywać zadania menadżerskie za pomocą równań matematycznych.

Jak wspomniałem, każda z niewiadomych może być rozwiązana jako 0 lub 1.

Tak więc pokazujemy to w sposób matematyczny. Musimy też wspomnieć, że x_{ij} musi być liczbą całkowitą (gdzie w – oznacza ilość sieci detalicznych, a k oznacza ilość kontraktów).

$$0 \geqslant x_{(w,k)} \geqslant 1 \ldots \forall w \in (1,2,3,4), k \in (1,2,3,4)$$

$$x_{(w,k)} \in G \ldots \forall w \in (1,2,3,4), k \in (1,2,3,4), G = 0 \vee G = 1$$

Żadna sieć detaliczna nie może podpisać więcej niż jednego kontraktu.

Jak pamiętamy, sklepy w naszej siatce reprezentowane są za pomocą wierszy.

Tak więc:

$$\begin{cases} x_{11} + x_{12} + x_{13} + x_{14} = 1A(Fenix) \\ x_{21} + x_{22} + x_{23} + x_{24} = 1B(Jaszczurka) \\ x_{31} + x_{32} + x_{33} + x_{34} = 1C(PDT) \\ x_{41} + x_{42} + x_{43} + x_{44} = 1D(Reksio) \end{cases}$$

Gdzie np. niewiadoma x_{12} oznacza podpisanie przez sieć detaliczną A (Fenix) wieloletniego kontraktu K2 na dostawy pieczywa za 115 tys. zł. Jeżeli algorytm wskaże, że $x_{12} = 1$ kontrakt taki zostanie podpisany, jeżeli $x_{12} = 0$ kontrakt taki nie zostanie podpisany.

Algorytm powinien również wziąć pod uwagę zasadę, że każdy kontrakt może być podpisany tylko raz. Zapisujemy to za pomocą poniższego układu równań.

$$\begin{cases} x_{11} + x_{21} + x_{31} + x_{41} = 1KontraktK1 \\ x_{12} + x_{22} + x_{32} + x_{42} = 1KontraktK2 \\ x_{13} + x_{23} + x_{33} + x_{43} = 1KontraktK3 \\ x_{14} + x_{24} + x_{34} + x_{44} = 1Brakkontraktu \end{cases}$$

Definiowanie funkcji celu

Celem jest maksymalizacja przychodów z tytułu miesięcznej sprzedaży kontraktów dla trzech sieci supermarketów. Powyższe wzory wskazują, że każda sieć supermarketów może podpisać tylko jeden kontrakt i że każdy kontrakt może być podpisany tylko raz. Ponieważ wynikiem analizy optymalizacji ma być zbiór liczb całkowitych 0 lub 1, funkcja celu powinna przyjąć postać:

$$\sum_{w \in W, k \in K} x_{(w,k)} c_{(w,k)} \rightarrow \max$$

gdzie C oznacza zaproponowaną cenę za dostawy pieczywa. Powyższe wyrażenie można zapisać w nieco bardziej przystępnej formie jako:

$$\begin{cases} x_{11} + x_{12} + x_{13} + x_{14} = 1A(Fenix) \\ x_{21} + x_{22} + x_{23} + x_{24} = 1B(Jaszczurka) \\ x_{31} + x_{32} + x_{33} + x_{34} = 1C(PDT) \\ x_{41} + x_{42} + x_{43} + x_{44} = 1D(Reksio) \end{cases}$$

$$\begin{cases} F(x_{w,k}) \\ = 129x_{11} + 115x_{12} + 125x_{13} + 0x_{14} \\ +121x_{21} + 119x_{22} + 122x_{23} + 0x_{24} \\ +118x_{31} + 110x_{32} + 114x_{33} + 0x_{34} \\ +120x_{41} + 126x_{42} + 126x_{43} + 0x_{44} \rightarrow max \end{cases}$$

Obliczenie algorytmów za pomocą biblioteki Pulp

Najtrudniejszym elementem programowania liniowego jest opisanie zjawiska za pomocą równań i nierówności matematycznych.

Poniżej zaprezentowałem rozwiązanie wykonane za pomocą bezpłatnej biblioteki Pulp w środowisku języka programowania Python.

```
from pulp import *
from fractions import Fraction

prob = LpProblem("Zadanie_przetargowe",LpMaximize)
```

```
x11=LpVariable("x11",lowBound=0,upBound=None, cat="Integer")
x12=LpVariable("x12",lowBound=0,upBound=None, cat="Integer")
x13=LpVariable("x13",lowBound=0,upBound=None, cat="Integer")
x14=LpVariable("x14",lowBound=0,upBound=None, cat="Integer")
x21=LpVariable("x21",lowBound=0,upBound=None, cat="Integer")
x22=LpVariable("x22",lowBound=0,upBound=None, cat="Integer")
x23=LpVariable("x23",lowBound=0,upBound=None, cat="Integer")
x24=LpVariable("x24",lowBound=0,upBound=None, cat="Integer")
x31=LpVariable("x31",lowBound=0,upBound=None, cat="Integer")
x32=LpVariable("x32",lowBound=0,upBound=None, cat="Integer")
x33=LpVariable("x33",lowBound=0,upBound=None, cat="Integer")
x34=LpVariable("x34",lowBound=0,upBound=None, cat="Integer")
x41=LpVariable("x41",lowBound=0,upBound=None, cat="Integer")
x42=LpVariable("x42",lowBound=0,upBound=None, cat="Integer")
x43=LpVariable("x43",lowBound=0,upBound=None, cat="Integer")
x44=LpVariable("x44",lowBound=0,upBound=None, cat="Integer")
```

Wprowadzamy funkcję celu

```
# objective function
prob += 129*x11+115*x12+125*x13+0*x14+121*x21+119*x22+122*x23+0*x24+118*x31
        +110*x32+114*x33+0*x34+120*x41+126*x42+126*x43+0*x44, "Maksymalizacja przychodów"
```

Wprowadzamy warunki ograniczające

```
# OGRANICZENIA DLA kontraktów (CZWARTA KOLUMNA TO kontrakt FANTOMOWy)
prob += x11 + x21 + x31+ x41  == 1
prob += x12 + x22 + x32+ x42  == 1
prob += x13 + x23 + x33+ x43  == 1
prob += x14 + x24 + x34+ x44  == 1
```

```
# OGRANICZENIA DLA KUPUJĄCYCH (CZWARTy wiersz TO kontrakt FANTOMOWy)
prob += x11 + x12 + x13 + x14 == 1  # Kupujący A
prob += x21 + x22 + x23 + x24 == 1  # Kupujący B
prob += x31 + x32 + x33 + x34 == 1  # Kupujący C
prob += x41 + x42 + x43 + x44 == 1  # Kupujący D
```

Rozwiązanie problemu

```
print("prob.solve",prob.solve())
# status of the solution
print(f"Status: {LpStatus[prob.status]}")
```

```
prob.solve 1
Status: Optimal
```

```
for v in prob.variables():
    print(f"{v.name} = {str(Fraction(v.varValue).limit_denominator())}")
```

```
x11 = 1
x12 = 0
x13 = 0
x14 = 0
x21 = 0
x22 = 0
x23 = 1
x24 = 0
x31 = 0
x32 = 0
x33 = 0
x34 = 1
x41 = 0
x42 = 1
x43 = 0
x44 = 0
```

Obliczenie maksymalnego przychodu ze sprzedaży:

```
# maximum value of the objective function
print(f"max (xij) = {str(Fraction(value(prob.objective)).limit_denominator())}")
```

```
max (xij) = 377
```

Algorytm wskazał, że:

– sieć detaliczna *Fenix* powinna podpisać kontrakt K1 na dostawy pieczywa za 129 tys. zł miesięcznie,

– sieć delikatesów *Jaszczurka* powinna podpisać kontrakt K3 za 122 tys. zł,

– z siecią supermarketów PDT nie podpiszemy kontraktu,

– sieć sklepów REKSIO powinna podpisać kontrakt K2 za 126 tys. zł.

Warto zwrócić uwagę, że 122 tys. za kontrakt K3 to nie była najwyższa oferta zakupu. Jednak połączenie wszystkich ofert daje maksymalny przychód. Algorytm wskazał na konfigurację ofert, która pozwoli na osiągnięcie najwyższego możliwego do osiągnięcia przychodu ze sprzedaży pieczywa do supermarketów.

Podsumowanie

Wyobraźmy sobie świat bez ludzi, w którym gospodarką zarządzają roboty, autonomiczne systemy zarządzania, które zawsze podejmują optymalne, czyli najlepsze decyzje. Wszyscy wiemy, że prędzej czy później do tego dojdzie. Teraz wchodzimy w okres przejściowy, czyli czas, w którym maszyny zaczynają zastępować ludzi w podejmowaniu decyzji.

Kiedyś ludzkość wchodziła w epokę maszyn parowych. Ci, którzy pierwsi mieli takie maszyny zdobywali fortuny, bo produkowali szybciej, taniej i lepiej. Produkowali optymalnie.

Obecnie wchodzimy w świat zarządzania automatycznego. Aby maszyna rozwiązała problem matematyczny, musi go zrozumieć przez wzory matematyczne. Żyjemy w okresie przejściowym, ludzie zapisują równania matematyczne, a maszyny je rozwiązują. Optymalizacja może przynieść ogromne korzyści, tylko trzeba zacząć ją stosować.

Wojciech Moszczyński