Sprawozdanie

Anna Hellmann Monika Krakowska Szymon Halski Michał Kowalik	Badania Operacyjne i Logistyka
Informatyka Techniczna III. rok, V. semestr, gr. 2 nr. zespołu	

Link do repozytorium:
https://github.com/kl0cek/intermediary-method-project

Cel projektu

Celem projektu *intermediary-method-project* było stworzenie aplikacji webowej do wspomagania rozwiązywania problemów transportowych z wykorzystaniem metody pośrednika (ang. Intermediary Method). Aplikacja miała umożliwiać wprowadzanie danych, ich wizualizację oraz wykonanie obliczeń prowadzących do zoptymalizowanego planu transportowego.

Zagadnienie Pośrednika

Zagadnienie pośrednika to jedno z podejść do rozwiązywania problemów transportowych, wykorzystywana do znajdowania planu minimalizującego koszty przewozu towarów z wielu źródeł do wielu odbiorców. Polega na iteracyjnym usprawnianiu planu transportowego przy pomocy tzw. "pośredników", czyli wyznaczaniu tras alternatywnych, które mogą przynieść oszczędności kosztowe.

W aplikacji metoda ta została zaimplementowana w backendzie – w postaci algorytmu analizującego początkowy plan bazowy i wyznaczającego nowe lepsze trasy dostaw.

Funkcja obliczająca metodę pośrednika

Ścieżka do pliku: /app/api/project/[projectId]/solve/route.ts

Główna funkcja backendowa aplikacji odpowiada za:

- wpisywanie danych wejściowych (macierz kosztów, zapasy, popyt),
- zbudowanie bazowego planu transportowego,
- wyznaczenie potencjałów (metoda uzupełniania),
- sprawdzenie możliwości poprawy planu,
- wykonanie iteracji optymalizacyjnych zgodnie z metodą pośrednika.

Fragment kodu obrazujący kluczowe etapy obliczeń:

```
// Zbalansujmy problem (dodajemy fikcyjny wiersz/kolumnę, jeśli trzeba)
const totalSupply = supply.reduce((sum, x) => sum + x, 0)
```

```
const totalDemand = demand.reduce((sum, x) => sum + x, 0)

if (totalSupply > totalDemand) {
  demand.push(totalSupply - totalDemand)
  sellPrice.push(0)
  cost.forEach(row => row.push(0))
} else if (totalDemand > totalSupply) {
  supply.push(totalDemand - totalSupply)
  buyPrice.push(0)
  cost.push(Array(origDemand.length).fill(0))
}

const m = supply.length
const n = demand.length

// Budujemy model LP
const model: any = {
  optimize: 'profit',
  opType: 'max',
  constraints: {},
  variables: {}
}

// ograniczenia podaży: Σ_j x_ij ≤ supply[i]
supply.forEach((s, i) => {
  model.constraints[`s${i}`] = { max: s }
})

// ograniczenia popytu: Σ_i x_ij ≥ demand[j]
demand.forEach((d, j) => {
  model.constraints[`d${j}`] = { min: d }
})

// zmienne x_i_j z funkcją celu profit = sellPrice[j] - buyPrice[i] - cost[i][j]
for (let i = 0; i < m; i++) {
  for (let j = 0; j < n; j++) {
    const varName = `x_${i}_${j}`
    const unitProfit = sellPrice[j] - buyPrice[i] - cost[i][j]
    model.variables[varName] = {
      profit: unitProfit,
      [`s${i}`]: 1,
      [`d${j}`]: 1
    }
  }
}

// Rozwiązujemy
const sol = Solver.Solve(model)

// Parsujemy plan transportu
const plan: PlanItem[] = []
```

```
for (const [key, value] of Object.entries(sol)) {
  if (key.startsWith('x_') && typeof value === 'number' && value > 0) {
    const [, si, sj] = key.split('_')
    plan.push({ i: +si, j: +sj, qty: value })
  }
}
```

Ten fragment ilustruje:

- przekształcenie danych wejściowych do postaci zbalansowanej,
- utworzenie modelu optymalizacji z uwzględnieniem zysków,
- rozwiązanie problemu za pomocą biblioteki javascript-lp-solver,
- oraz ekstrakcję optymalnego planu działania.

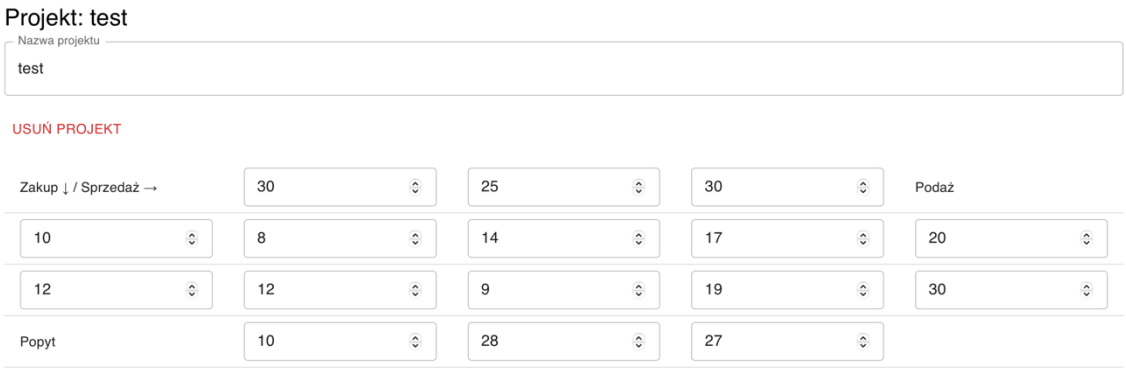
Zastosowano klasyczne podejście do rozwiązywania problemu transportowego – w tym sortowanie, aktualizację macierzy rozwiązań oraz optymalizację kosztów całkowitych.

Użyte technologie

- **React + Next.js** – frontend aplikacji oparty o React z wykorzystaniem App Routera (Next.js), pozwalający na SSR oraz dynamiczne zarządzanie trasami.
- **TypeScript** – silne typowanie poprawiające stabilność i przewidywalność kodu.
- **Material UI (MUI)** – gotowe komponenty UI do budowy interfejsu (tabele, inputy, przyciski).
- **Next.js API Routes** – logika metody pośrednika i operacje na danych realizowane po stronie serwera.

Funkcjonalności aplikacji

- Tworzenie projektów i zapisywanie danych wejściowych (koszty, popyt, podaż).



- Formularze do edycji i wizualizacji macierzy kosztów.

Zyski Jednostkowe

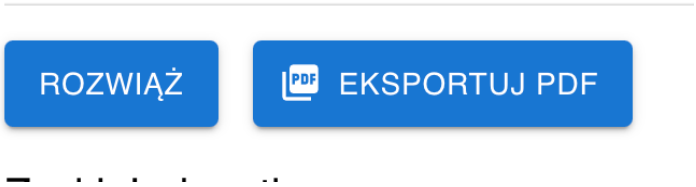
O1	O2	O3
12.00	1.00	3.00
6.00	4.00	-1.00
30.00	25.00	30.00

- Wykonywanie metody pośrednika i prezentacja zoptymalizowanego rozwiązania i tabela z wynikami (przydziały transportowe + koszt całkowity).

Plan Transportu

Dostawca	Odbiorca	Ilość
2	3	2
2	2	28
1	1	10
1	3	10
3	3	15
Łączny zysk		**710.00**

- Automatyczne wyznaczanie początkowego planu bazowego i eksport wyników do pliku.



Wnioski

Aplikacja pozwala na praktyczne zastosowanie metody pośrednika w kontekście klasycznego zagadnienia transportowego. Dzięki odpowiedniej implementacji możliwe jest przeprowadzenie pełnego cyklu optymalizacji – od wprowadzenia danych, przez analizę, aż po generowanie optymalnego planu z uwzględnieniem minimalnych kosztów.

Rozwiązanie to może być użyteczne zarówno w celach edukacyjnych (ilustracja metody) jak i jako narzędzie wspomagające planowanie logistyki w prostszych scenariuszach rzeczywistych. Zaletą jest czytelna wizualizacja oraz pełna automatyzacja obliczeń.