Оптимізація мережі ланцюгів постачання за умов невизначеності

Підготував Дубов Ілья Володимирович

н. к. Чорней Руслан Костянтинович

Мета роботи

- Дослідити класу оптимізаційних задач за умов невизначеності
- Розглянути теоретичну базу від задач лінійного програмування, до більш абстрактних моделей за умов невизначеності
- Розглянути приклади задач і навести відповідні розрахунки.

Постановка задачі

1. Лінійне програмування

- Детерміністичне
- Враховує один сценарій розвитку подій
- Одноетапна модель

$$z = c^T x \to \min(\max)$$

$$Ax \leq b$$

$$x_i \geq 0$$

2. Стохастичне програмування

- Враховує фактор невизначеності (ризик)
- Враховує багато сценаріїв розвитку подій
- Є багатоетапною моделлю

$$z(x) = z(x, g(x, \Omega))$$

$$\forall \omega \in \Omega : f_\omega(x) = 0$$

$$\min(z) - ?$$

$$P = (p_1, p_2, ..., p_s)$$

$$\sum_{i=1}^{s} p_i = 1$$

Двоетапне стохастичне лінійне програмування (SLP)

Дано:

1. Ймовірнісний простір сценаріїв **Ω**
2. Очікуване значення втрат (ризик) — як критерій
3. Предмет оптимізації для кожного сценарію (лінійна функція), а також на момент до настання сценарію

Знайти:

1. Розв’язок на момент до настання сценарію
2. Розв’язок для кожного сценарію
3. Оптимізоване значення

$$z(x) = c^T x + \mathbb{E}[Q(x, x_\omega)]$$

$$Ax \leq b$$

$$x \geq 0$$

$$\min(z) - ?$$

$$Q(x, x_\omega) = q_w^T x_\omega$$

$$B_\omega x + C_\omega x_\omega \leq h_\omega$$

$$x_\omega \geq 0$$

Детерміністичний аналог

$$
\begin{aligned}
\min\{z(x) = \; & c^T x_0 + p_1 q_1^T x_1 + p_2 q_2^T x_2 + \ldots + p_n q_n^T x_n\} \\
& Ax_0 \leq b \\
& B_1 x_0 + C_1 x_1 \leq h_1 \\
& B_2 x_0 + C_2 x_2 \leq h_2 \\
& \vdots \\
& B_n x_0 + C_n x_n \leq h_n \\
& x_0, x_1, \ldots, x_n \geq 0
\end{aligned}
$$

Приклад. Проблема логістики

Гуманітарна організація готується до природної катастрофи у місцевому регіоні, в якому знаходиться **7 міст**. Оцінюється, що ураган точно зруйнує міста **A і B** (ймовірність 75%); можливо **ще** зруйнує міста **C, D та E** (ймовірність 20%); або **всі** міста (ймовірність 5%).

Люди, які проживають в цих містах, потребуватимуть допомоги протягом **10 днів**, допоки будуть виконуватися роботи по відновленню міст. Розраховується надати наступні засоби першої допомоги: **аптечки, сухі пайки, бутельовану воду, набори гігієни, набори дитячого харчування, та набори гігієни для людей похилого віку.**

Для їх забезпечення організація планує орендувати до **5-ти складських приміщень** неподалік регіону, де будуть запаковані всі засоби для подальшої відправи в міста. У випадку, якщо засобів не буде вистачати, організація може доставити необхідну кількість з **центрального складу**, але за більшими тарифами.

Засоби відправляються **партіями по 100 штук** одного засобу зі складу до міста.

Враховуючи вищезазначені умови, та наведені табличні дані, мінімізуйте очікувані витрати організації. Визначте, які складські приміщення має орендувати організація, яким чином вони мають бути заповнені, і яким чином налаштувати доставу партій засобів першої допомоги при кожному сценарію. Якими будуть витрати організації у кожному випадку?

Місто	Населення
A	10000
B	12000
C	5000
D	20000
E	8500
F	9200
G	15400

Табл. 3: Населення міст

Аптечка	15$
Сухий пайок	10$
Вода	0.5$
Набір гігієни	5$
Набір дитячого харчування	7$
Набір гігієни для людей похилого віку	12.5$

Табл. 6: Вартості засобів першої допомоги

Вікові категорії		Потреби на 1 душу населення на 1 день					
Вікова група	Відсоток населення	Аптечки	Сухі пайки	Бутильована вода	Набори гігієни	Набори дитячого харчування	Набори гігієни для людей похилого віку
0-4	5%	0.02	0	2	0.05	0.5	0
5-14	17.5%	0.04	0.3	3	0.08	0	0
15-64	65%	0.05	0.33	4	0.1	0	0
65+	12.5%	0.07	0.3	3.5	0	0	0.2

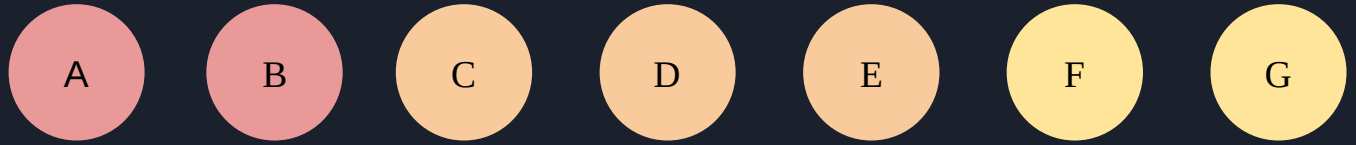
Табл. 4: Розподіл потреб по віковим категоріям

Склади	Ціна перевезення однієї партії до міста ($)							Ціна оренди на 1 день ($)	Ємність (одиниці засобів)
	A	B	C	D	E	F	G		
1	50	40	35	35	25	30	45	1000	800 000
2	45	45	30	40	50	55	50	700	600 000
3	47	49	58	60	35	40	35	900	720 000
4	25	40	60	70	65	40	55	350	320 000
5	55	35	40	45	48	35	65	490	440 000
Центральний	250	200	190	210	280	275	290	-	-

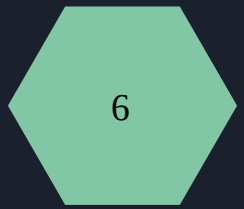
Табл. 5: Інформація про складські приміщення

Схематичне зображення

Міста



Склади



Перший етап

Витрати:

1. Засоби першої допомоги
2. Оренда складів

Обмеження:

1. Кількість всіх засобів на складі j не має перевищувати ємність складу

$$z = \sum_j w_j \sum_i p_i x_{ij} + 10 \sum_j w_j f_j + \mathbb{E}[Q_k]$$

$$\sum_i x_{ij} \leq C_j$$

$$\min(z) - ?$$

Другий етап

Витрати:

1. Доставa необхідних засобів з орендованих складів у необхідні міста
2. Доставa засобів з центрального складу в міста, у випадку нестачі.

Обмеження:

1. Потреби кожного міста у кожному засобі мають бути задовольнені
2. Кількість відправленого засобу з конкретного складу не має перевищувати кількість засобів на складі

$$Q_k = \sum_{s \in \omega_k} \sum_i \left(\sum_j w_j c_{js} y_{ijs}^k + c_{6s} y_{i6s}^k + 100 p_i y_{i6s}^k \right)$$

$$100 \sum_{j^*} y_{ij^*s}^k \geq 10 \cdot S_s E_i$$

$$x_{ij} - 100 \sum_{s \in \omega_k} y_{ijs}^k \geq 0$$

Приклад реалізації

```
solutions = []
for warehouse_setup in warehouse_permutations:
  # --- PREPARE VARIABLES ---
  # Indicies of the warehouses that are rented
  w = np.where(warehouse_setup == 1)[0]
  w_with_central = np.append(w, 5)

  # All supply variables
  x_labels = [(i,j)
    for i in range(total_supply_types)
    for j in w
  ]

  # All delivery variables
  y_labels = [(k,i,j,s)
    for k in range(K)
    for i in range(total_supply_types)
    for j in w_with_central
    for s in Omega[k]
  ]

  # Variables-indicies mapper
  label_to_index = {
    **{('x', i, j): idx for idx, (i, j) in enumerate(x_labels)},
    **{('y', k, i, j, s): idx + len(x_labels) for idx, (k, i, j, s) in enumerate(y_labels)}
  }
  index_to_label = {
    idx: ('x', i, j) for idx, (i, j) in enumerate(x_labels)
  }
  index_to_label.update({
    idx + len(x_labels): ('y', k, i, j, s) for idx, (k, i, j, s) in enumerate(y_labels)
  })
```

```
# Objective function
total_vars = len(label_to_index)
c = np.zeros(total_vars)

# --- OBJECTIVE FUNCTION ---
# Total rental price to be included after the solution
total_rental_price = T * np.sum(warehouse_rent_prices[w])

# First stage. Initial supplies cost
for (i, j) in x_labels:
  idx = label_to_index[('x', i, j)]
  c[idx] += supply_prices[i]

# Second stage. Delivery cost
for (k, i, j, s) in y_labels:
  idx = label_to_index[('y', k, i, j, s)]
  if j == 5:
    c[idx] += p[k] * (delivery_prices[j][i] + N * supply_prices[i])
  else:
    c[idx] += p[k] * delivery_prices[j][i]
```

Приклад реалізації

```python
# --- CONSTRAINTS ---
A_ub = []
b_ub = []

# First stage. Capacity constraints
for j in w:
  row = np.zeros(total_vars)
  for i in range(total_supply_types):
    idx = label_to_index[('x', i, j)]
    row[idx] = 1
  A_ub.append(row)
  b_ub.append(warehouse_capacities[j])

# Second stage. Fullfillment of needs
for k in range(K):
  for s in Omega[k]:
    for i in range(total_supply_types):
      row = np.zeros(total_vars)
      for j in w_with_central:
        idx = label_to_index[('y', k, i, j, s)]
        row[idx] = N
      A_ub.append(-row) # Invert due to linprog
      b_ub.append(-T * supply_types_estimated_needs[i] * cities_population[s]) # Invert due to linprog

# Second stage. Stocked supplies restriction
for k in range(K):
  for i in range(total_supply_types):
    for j in w:
      row = np.zeros(total_vars)
      idx = label_to_index[('x', i, j)]
      row[idx] = -1
      for s in Omega[k]:
        idx = label_to_index[('y', k, i, j, s)]
        row[idx] = N
      A_ub.append(row)
      b_ub.append(0)
```

```python
if (np.array(A_ub).shape[0] == 0):
  A_ub = np.zeros((1, total_vars))

if (np.array(b_ub).shape[0] == 0):
  b_ub = np.zeros(1)

res = opt.linprog(c, A_ub=A_ub, b_ub=b_ub, method='highs', integrality=1)
solutions.append({
  'status': res.success,
  'fun': res.fun + total_rental_price,
  'w': w,
  'x': { index_to_label[i]: val for i, val in enumerate(res.x) }
})
```

```python
best_func = np.inf
best_sol = None

for i, sol in enumerate(solutions):
  if (sol['fun'] < best_func):
    best_func = sol['fun']
    best_sol = sol

best_sol
```

Приклад реалізації (перший етап)

```
first_stage_raw = {k: v for k, v in best_sol['x'].items() if k[0] == 'x'}
first_stage_decision = pd.DataFrame()
for ((_, i, j), v) in first_stage_raw.items():
  if (v == 0):
    continue
  first_stage_decision = pd.concat([first_stage_decision, pd.DataFrame({
    'supply': [i],
    'supply_type': [supply_types[i]],
    'warehouse': [j],
    'amount': [v]
  })], ignore_index=True)

first_stage_decision
```

Розв’язок (перший етап)

- Склади 1, 2 та 4
- 11,000 аптечок. Усі в 4 складі.
- 67,100 сухих пайків. Усі в 4 складі.
- 1,371,000 бутлів води. 771,000 в 1 складі, решта у 2.
- 18,000 наборів гігієни. Усі в 1 складі.
- 5,500 наборів дитячого харчування. Усі в 1 складі.
- 5,500 наборів гігієни для людей похилого віку. Усі в 1 складі.

supply	supply_type	warehouse	amount
0	first-aid-kits	3	11000
1	dry-rations	3	67100
2	bottled-water	0	771000
2	bottled-water	1	600000
3	hygiene-kits	0	18000
4	child-food-kits	0	5500
5	elderly-hygiene-kits	0	5500

Приклад реалізації (другий етап)

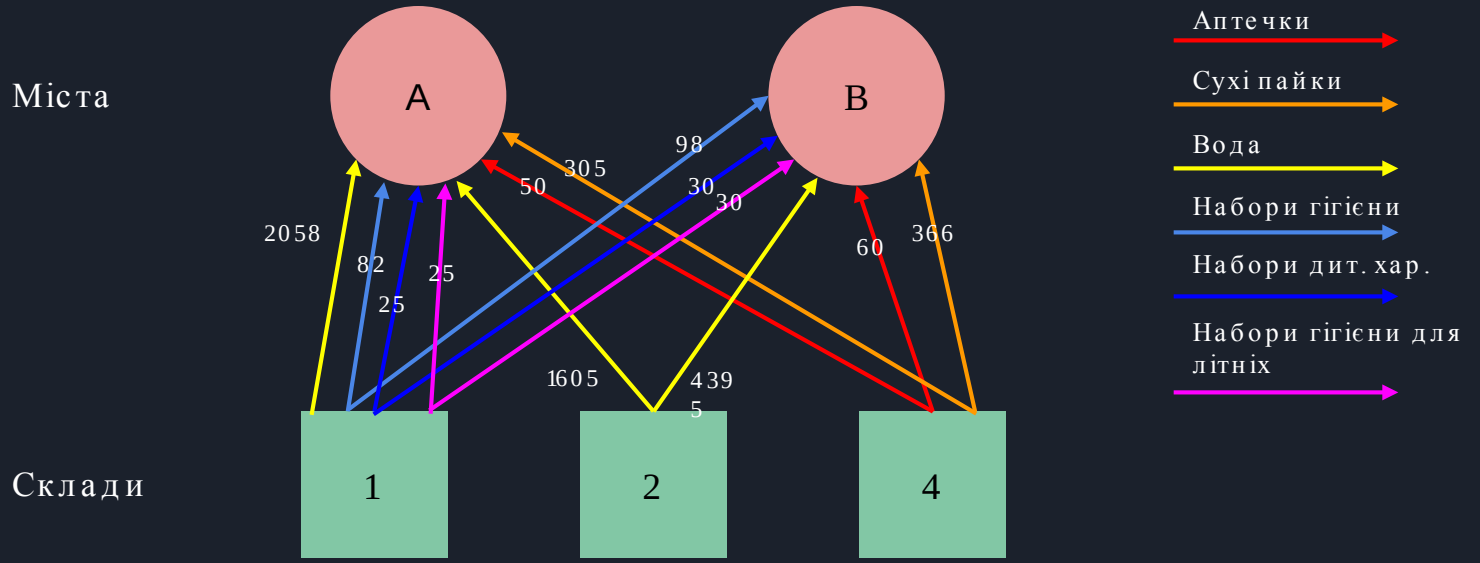
```python
def get_scenario_df(sol, k):
  x_vars = { key: v for key, v in sol['x'].items() if key[0] == 'x'}
  y_vars = { key: v for key, v in sol['x'].items() if key[0] == 'y' and key[1] == k}

  x_values = np.sum(v * supply_prices[i] for (var, i, j), v in x_vars.items())
  rental_price = T * np.sum(warehouse_rent_prices[sol['w']])
  y_values = np.sum(v * delivery_prices[j][i]
    for (var, scenario, i, j, s), v in y_vars.items()
    if j != 5
  )
  y_values_central = np.sum(v * (delivery_prices[j][i] + N * supply_prices[i])
    for (var, scenario, i, j, s), v in y_vars.items()
    if j == 5
  )
  total_value = x_values + rental_price + y_values + y_values_central

  scenario_raw = { key: v for key, v in sol['x'].items() if key[0] == 'y' and key[1] == k}
  scenario_decision = pd.DataFrame()
  for ((_, k, i, j, s), v) in scenario_raw.items():
    if (v == 0):
      continue
    scenario_decision = pd.concat([scenario_decision, pd.DataFrame({
      'supply': [i],
      'supply_type': [supply_types[i]],
      'warehouse': [j],
      'city': [s],
      'amount': [v]
    })], ignore_index=True)

  return (scenario_decision, total_value, x_values + rental_price)
```

Розв'язок (другий етап, перший сценарій)

```
(scenario1_decision, value1, first_stage_value) = get_scenario_df(best_sol, 0)
print(value1, first_stage_value)
scenario1_decision
```

supply	supply_type	warehouse	city	amount
0	first-aid-kits	3	0	50
0	first-aid-kits	3	1	60
1	dry-rations	3	0	305
1	dry-rations	3	1	366
2	bottled-water	0	0	2058
2	bottled-water	1	0	1605
2	bottled-water	1	1	4395
3	hygiene-kits	0	0	82
3	hygiene-kits	0	1	98
4	child-food-kits	0	0	25
4	child-food-kits	0	1	30
5	elderly-hygiene-kits	0	0	25
5	elderly-hygiene-kits	0	1	30

Розв’язок (другий етап, перший сценарій)



Розв'язок (другий етап, другий сценарій)

```
(scenario2_decision, value2, first_stage_value) = get_scenario_df(best_sol, 1)
print(value2, first_stage_value)
scenario2_decision
```

supply	supply_type	warehouse	city	amount
0	first-aid-kits	3	0	50
0	first-aid-kits	3	2	25
0	first-aid-kits	3	4	35
0	first-aid-kits	5	1	60
0	first-aid-kits	5	3	99
0	first-aid-kits	5	4	7
1	dry-rations	3	1	62
1	dry-rations	3	3	609
1	dry-rations	5	0	305
1	dry-rations	5	1	304
1	dry-rations	5	2	153
1	dry-rations	5	4	259
2	bottled-water	0	3	5720
2	bottled-water	0	4	1990
2	bottled-water	1	1	4395
2	bottled-water	1	3	1605
2	bottled-water	5	0	3663
2	bottled-water	5	2	1832
2	bottled-water	5	4	1124
3	hygiene-kits	0	0	82
3	hygiene-kits	0	2	41
3	hygiene-kits	0	3	57
3	hygiene-kits	5	1	98
3	hygiene-kits	5	3	106
3	hygiene-kits	5	4	70
4	child-food-kits	0	0	20
4	child-food-kits	0	2	13
4	child-food-kits	0	4	22
4	child-food-kits	5	0	5
4	child-food-kits	5	1	30
4	child-food-kits	5	3	50
5	elderly-hygiene-kits	0	0	25
5	elderly-hygiene-kits	0	1	30
5	elderly-hygiene-kits	5	2	13
5	elderly-hygiene-kits	5	3	50
5	elderly-hygiene-kits	5	4	22

Розв'язок (другий етап, третій сценарій)

```
(scenario3_decision, value3, first_stage_value) = get_scenario_df(best_sol, 2)
print(value3, first_stage_value)
scenario3_decision
```

supply	supply_type	warehouse	city	amount
0	first-aid-kits	3	1	60
0	first-aid-kits	3	3	50
0	first-aid-kits	5	0	50
0	first-aid-kits	5	2	25
0	first-aid-kits	5	3	49
0	first-aid-kits	5	4	42
0	first-aid-kits	5	5	46
0	first-aid-kits	5	6	76
1	dry-rations	3	3	609
1	dry-rations	3	5	62
1	dry-rations	5	0	305
1	dry-rations	5	1	366
1	dry-rations	5	2	153
1	dry-rations	5	4	259
1	dry-rations	5	5	219
1	dry-rations	5	6	469
2	bottled-water	0	3	7325
2	bottled-water	0	5	385
2	bottled-water	1	0	1183
2	bottled-water	1	2	1832
2	bottled-water	1	5	2985
2	bottled-water	5	0	2480
2	bottled-water	5	1	4395
2	bottled-water	5	4	3114
2	bottled-water	5	6	5641
3	hygiene-kits	0	1	54
3	hygiene-kits	0	6	126
3	hygiene-kits	5	0	82
3	hygiene-kits	5	1	44
3	hygiene-kits	5	2	41
3	hygiene-kits	5	3	163
3	hygiene-kits	5	4	70
3	hygiene-kits	5	5	75
4	child-food-kits	0	2	10
4	child-food-kits	0	4	22
4	child-food-kits	0	5	23
4	child-food-kits	5	0	25
4	child-food-kits	5	1	30
4	child-food-kits	5	2	3
4	child-food-kits	5	3	50
4	child-food-kits	5	6	39
5	elderly-hygiene-kits	0	0	25
5	elderly-hygiene-kits	0	2	7
5	elderly-hygiene-kits	0	5	23
5	elderly-hygiene-kits	5	1	30
5	elderly-hygiene-kits	5	2	6
5	elderly-hygiene-kits	5	3	50
5	elderly-hygiene-kits	5	4	22
5	elderly-hygiene-kits	5	6	39

Витрати

- Модель передбачає витрати: 3 316 289.25$
- Перший сценарій: 2 030 195$
- Другий сценарій: 5 739 740$
- Третій сценарій: 9 313 900$

Висновки

- SP добре моделює реальні оптимізаційні проблеми за умов невизначеності.
- Надає розв’язки в багато етапів.
- За конкретними критеріями може бути лінеаризована у вигляд SLP і зведена до детермінованої LP.
- Масштабується необмежено.

Дякую за увагу!