Тема 6 МОДЕЛІ ТА МЕТОДИ ОПТИМІЗАЦІЇ НА МЕРЕЖАХ

6.1 Типові задачі оптимізації на мережах та методи їх розв'язання

Задачі оптимізації на мережах є важливим розділом дослідження операцій і широко застосовуються для моделювання та аналізу систем транспортування, зв'язку, логістики, виробничих процесів, комп'ютерних мереж і проєктного управління. Мережа зазвичай подається у вигляді графа, вершини якого відповідають об'єктам або станам системи, а дуги – можливим зв'язкам між ними з певними характеристиками (вартість, пропускна здатність, тривалість тощо).

Задача найкоротшого шляху полягає в знаходженні шляху з мінімальною сумарною вагою між двома вершинами мережі. Вони застосовується в навігації, маршрутизації даних, плануванні перевезень. Методи розв'язання цієї задачі: алгоритм Дейкстри, алгоритм Беллмана-Форда, алгоритм Флойда-Воршелла.

Задача максимального потоку полягає у визначенні максимального можливого потоку від джерела до стоку за умови обмеженої пропускної здатності дуг. Вона використовується для аналізу транспортних, інформаційних та енергетичних мереж. Методи розв'язання задачі: алгоритм Форда-Фалкерсона, алгоритм Едмондса-Карпа, алгоритм Дініца.

Задача мінімальної вартості потоку передбачає знаходження потоку заданої величини з мінімальними сумарними витратами, вона є узагальненням транспортної задачі. Методи розв'язання: метод потенціалів, алгоритм послідовного найкоротшого шляху, симплекс-метод для мережевих задач.

Задача мінімального кістякового дерева полягає у побудові підмножини ребер, що з'єднує всі вершини мережі з мінімальною сумарною вагою. Застосовується при проєктуванні мереж зв'язку, електромереж, трубопроводів. Методи розв'язання: алгоритм Крускала, алгоритм Прима.

Задачі мережевого планування пов'язані з оптимізацією виконання проєктів у часі. Такі задачі дають змогу визначати тривалість проєкту, критичні роботи та часові резерви. Методи розв'язання: метод критичного шляху (CPM), метод PERT.

Задачі потоків з обмеженнями та узгодження. До них належать задачі призначення, паросполучення та узгодження в двочасткових графах. Вони використовуються у задачах розподілу ресурсів і планування. Методи розв'язання: угорський алгоритм, алгоритми пошуку максимального паросполучення.

Деякі з зазначених задач та методів їх розв'язання вивчаються здобувачами вищої освіти спеціальності «Комп'ютерні науки» в межах курсу «Дискретна математика». Тут коротко розглянемо деякі з таких задач.

6.2 Поняття і термінологія теорії графів та мереж

Теорія графів – це розділ математики, який вивчає властивості та структури графів і методи їх аналізу. Вона широко застосовується в комп'ютерних науках, теорії алгоритмів, логістиці, телекомунікаціях, транспортних і виробничих системах.

Граф – це математична модель, що описується парою

$$G = (V, E),$$

де V – множина вершин (вузлів),

E – множина ребер (дуг), які з'єднують пари вершин.

У контексті *мережевих задач* вершини зазвичай відповідають об'єктам або станам системи, а ребра – зв'язкам між ними (перевезення, передача інформації, залежності робіт тощо).

Основні терміни теорії графів:

- *вершина (вузол)* – елемент графа, що представляє об'єкт або пункт системи;
- *ребро* – зв'язок між двома вершинами в неорієнтованому графі;
- *дуга* – орієнтований зв'язок між вершинами;
- *суміжні вершини* – вершини, з'єднані ребром (дугою);
- *ступінь вершини* – кількість ребер, інцидентних вершині;
- *шлях* – послідовність вершин, з'єднаних ребрами;
- *цикл* – шлях, у якому початкова і кінцева вершини збігаються;
- *зв'язний граф* – граф, у якому між будь-якими двома вершинами існує шлях;
- *кістякове дерево* – зв'язний неорієнтовний граф без циклів.

Неорієнтований граф – це граф, у якому ребра не мають напряму, тобто зв'язок між вершинами є двостороннім.

Приклад неорієнтовного графа наведено на рис. 6.1. У цьому графі

- $1, 2, 3, 4, 5$ – вершини,
- $(1,2), (2,5), (1,5), (3,4)$ – ребра;
- цей граф містить цикл $1 - 2 - 5$;
- граф незв'язний.

Граф на рис. 6.2 є зв'язним. Він не містить циклів. Тому він являє собою кістякове дерево.

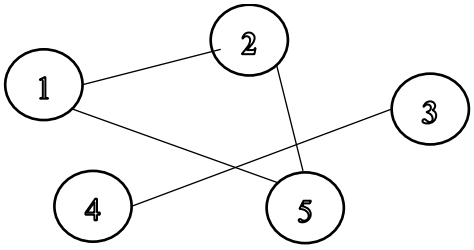

Рис. 6.1 – Неорієнтовний незв'язний граф

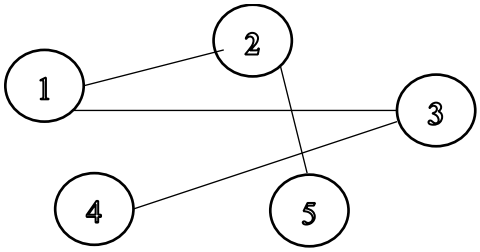

Рис. 6.2 – Неорієнтовний зв'язний граф

Приклади застосування неорієнтовних графів:

- проєктування електричних або комп'ютерних мереж;
- задача мінімального кістякового дерева (алгоритми Прима, Крускала);
- аналіз структур без напрямку руху.

Орієнтований граф – це граф, у якому кожне ребро має напрям і називається дугою.

Приклад орієнтованого графа зображено на рис. 6.3.

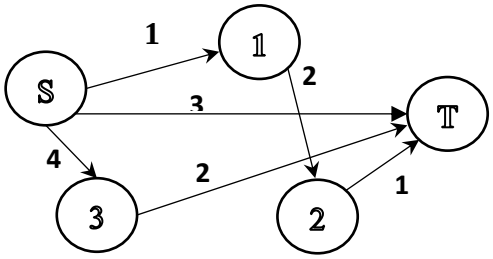

Рис. 6.3 – Орієнтовний граф

Тут:

– дуга 1→2 означає рух або зв'язок лише від вершини 1 до вершини 2;
– напрям має принципове значення;
– шлях можливий лише за напрямом стрілок.

Для орієнтованих графів вводять:

– *вхідний ступінь вершини* – кількість дуг, що входять у вершину;
– *вихідний ступінь вершини* – кількість дуг, що виходять із вершини;
– *ві́ток (джерело)* – вершина, в якій дуги беруть початок і жодна не закінчується в ній;
– *стік* – вершина, в якій дуги закінчуються і жодна не бере початок.

Приклади застосування:

– транспортні та логістичні мережі;
– комп'ютерні мережі й маршрутизація;
– мережеве планування (CPM, PERT);
– задачі максимального потоку.

Мережі як спеціальний випадок графів

Мережею називають граф (орієнтований або неорієнтований), у якому кожному ребру (дузі) поставлено у відповідність певні характеристики: вартість, пропускну здатність, довжину, тривалість тощо. Така характеристика називається *вагою* ребра (дуги).

Рис. 6.3 є прикладом мережі. Тут:

– *S* – джерело;
– *T* – стік;
– дуги мають напрям і мають обмеження на потік.

Таблиця 6.1 – Порівняння орієнтованих і неорієнтованих графів

Ознака	Неорієнтований граф	Орієнтований граф
Напрям зв'язків	Відсутній	Присутній
Тип зв'язку	Двосторонній	Односторонній
Основні задачі	Кістякові дерева	Потоки, шляхи
Приклад	Електромережі	Транспортні мережі

Теорія графів надає універсальний апарат для моделювання та аналізу складних систем. Неорієнтовані графи використовуються для опису симетричних зв'язків, тоді як орієнтовані графи є основою для моделювання потоків, процесів і мережевих задач у дослідженні операцій та комп'ютерних науках.

6.3 Задача мінімізації мережі та її розв'язання методом Крускала

6.3.1 Постановка задачі мінімізації мережі

Нехай задано множину об'єктів (пунктів, вузлів), між якими можливе прокладання з'єднань. Для кожної пари об'єктів, що може бути безпосередньо

з’єднана, відома вартість (або довжина) відповідного з’єднання. Необхідно побудувати таку мережу з’єднань, яка забезпечує зв’язність усіх об’єктів, тобто можливість передачі інформації, ресурсів або енергії між будь-якою парою вузлів (безпосередньо або через інші вузли), та при цьому має мінімальну сумарну вартість з’єднань.

Таким чином, задача формалізується як задача *побудови мінімального кістякового дерева зваженого неорієнтованого графа* і може бути розв’язана, зокрема, алгоритмами Крускала або Прима. Викладемо алгоритм Крускала в загальному вигляді.

6.3.2Алгоритм Крускала побудови мінімального кістякового дерева

Вхідні дані:

- неорієнтований зв’язний зважений граф

$$G = (V, E),$$

де кожному ребру $e \in E$ поставлено у відповідність вагу $\omega(e)$.

Вихідні дані:

- мінімальне кістякове дерево $T \subseteq E$.

Кроки алгоритму:

1. *Ініціалізація.*

Вважати, що множина ребер T кістякового дерева порожня:

$$T = \emptyset.$$

Кожну вершину графа розглядати як окрему компоненту зв’язності.

2. *Сортування ребер.*

Впорядкувати всі ребра графа E за зростанням їх ваг:

$$\omega(e_1) \le \omega(e_2) \le \cdots \le \omega\left(e_{|E|}\right),$$

де $|E|$ −кількість ребер графа G.

3. *Послідовний перегляд ребер.*

Послідовно переглядати ребра у відсортованому списку.

Для кожного ребра $e = (u, v)$:

- якщо вершини u і v належать *різним компонентам зв’язності*, то:
  - додати ребро e до множини T;
  - об’єднати відповідні компоненти;
- інакше (якщо додавання ребра утворює цикл) – ребро пропустити.

4. *Умова завершення.*

Алгоритм завершується, коли

$$|T| = |V| - 1.$$

Результат

Множина ребер T утворює *мінімальне кістякове дерево,* тобто:

- граф є зв’язним;
- не містить циклів;
- сумарна вага ребер

$$\sum_{e \in T} \omega(e)$$

є мінімальною серед усіх кістякових дерев графа G.

Зауваження:

- алгоритм Крускала є *жадібним алгоритмом*, який на кожному кроці побудови розв'язку робить локально оптимальний вибір, тобто обирає найкращий варіант за заданим критерієм на поточному етапі, не переглядаючи попередні розв'язки і не аналізуючи всі можливі майбутні наслідки, з метою отримання оптимального або наближено оптимального глобального розв'язку;
- алгоритм застосовується *лише до неорієнтованих графів.*

Приклад 6.1 Для заданого графа (див. рис. 6.4):

- скласти змістовну постановку певної техніко-економічної задачі як задачі мінімізації мережі;
- знайти у побудованому графі мінімальне кістякове дерево за допомогою алгоритмів Крускала;
- зробити висновки в термінах постановки задачі.

Розв'язання.

Змістовна постановка задачі

Комунальне підприємство планує модернізацію системи електропостачання житлового району. Відоме розташування трансформаторних підстанцій та можливі варіанти прокладання кабельних ліній, які можуть з'єднувати підстанції між собою, а також з центральною розподільчою підстанцією. Необхідно спроєктувати таку схему прокладання кабельних ліній мінімальної загальної довжини, яка забезпечить електропостачання кожної трансформаторної підстанції від центральної (безпосередньо або через інші підстанції). Відомі довжини можливих кабельних ліній (у кілометрах), що з'єднують усі підстанції між собою та з центральною підстанцією.

Математична модель. Математичною моделлю цієї задачі є її зображення у вигляді мережі (див. рис. 6.4).

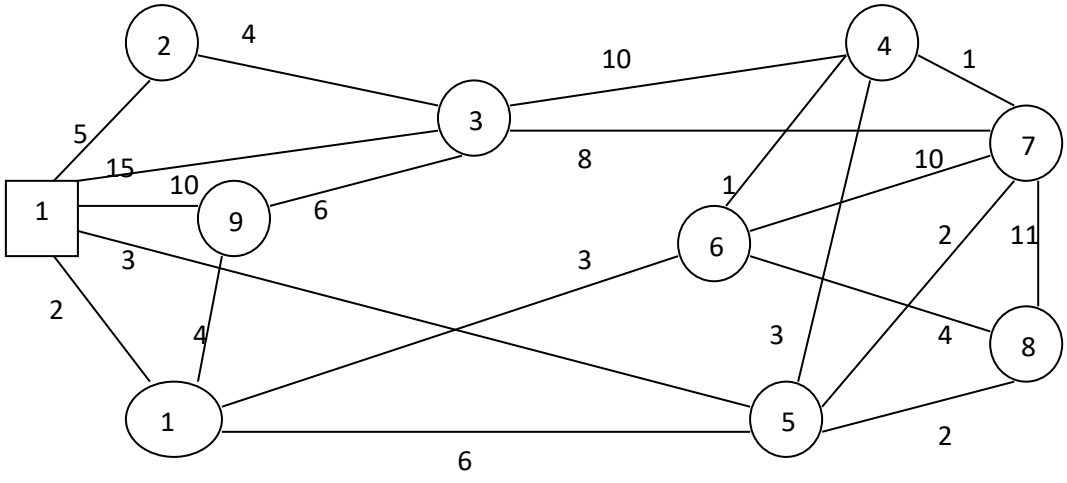

Рис. 6.4 – Мережева модель задачі

Нехай вершина 1 мережі відповідає центральній розподільчій підстанції, а вершини 2–10 – трансформаторній підстанції. Ребра, що з'єднують пари вершин

мережі, позначають кабельні лінії, а їхні ваги – довжину відповідної лінії. Необхідно знайти в заданій мережі кістякове дерево мінімальної ваги, яке з'єднує вершину 1 з усіма іншими вершинами мережі.

Розв'яжемо початкову задачу за допомогою побудованої математичної моделі, використовуючи *алгоритм Крускала*. Для цього впорядкуємо ребра за зростанням їхніх ваг(див. табл. 6.2).

Таблиця 6.2 – Реалізація алгоритму Крускала

Ребро, l	$(4,7)^*$	$(4,6)^*$	$(1,10)^*$	$(5,7)^*$	$(5,8)^*$	$(1,5)^{*(-)}$	$(6,10)^{-(*)}$	$(4,5)^-$	$(9,10)^*$	$(2,3)^*$
Вага ребра, $w(l)$	1	1	2	2	2	3	3	3	4	4
Ребро, l	$(6,8)^-$	$(1,2)^*$	$(3,9)^-$	$(5,10)^-$	$(3,7)^-$	$(1,9)^-$	$(3,4)^-$	$(6,7)^-$	$(7,8)^-$	$(1,3)^-$
Вага ребра, $w(l)$	4	5	6	6	8	10	10	10	11	15

В табл. 6.2 знаком «*» позначені ребра, що включаються до кістякового дерева, знаком «– » – не включаються.

При реалізації алгоритму включається одне з ребер (5,8) або (1,5), чим пояснюються позначення «*(-)» або «-(*)» відповідно.

Таким чином, дана задача має два оптимальні розв'язки (кістякові дерева мінімальної ваги), побудована на рис. 6.5.

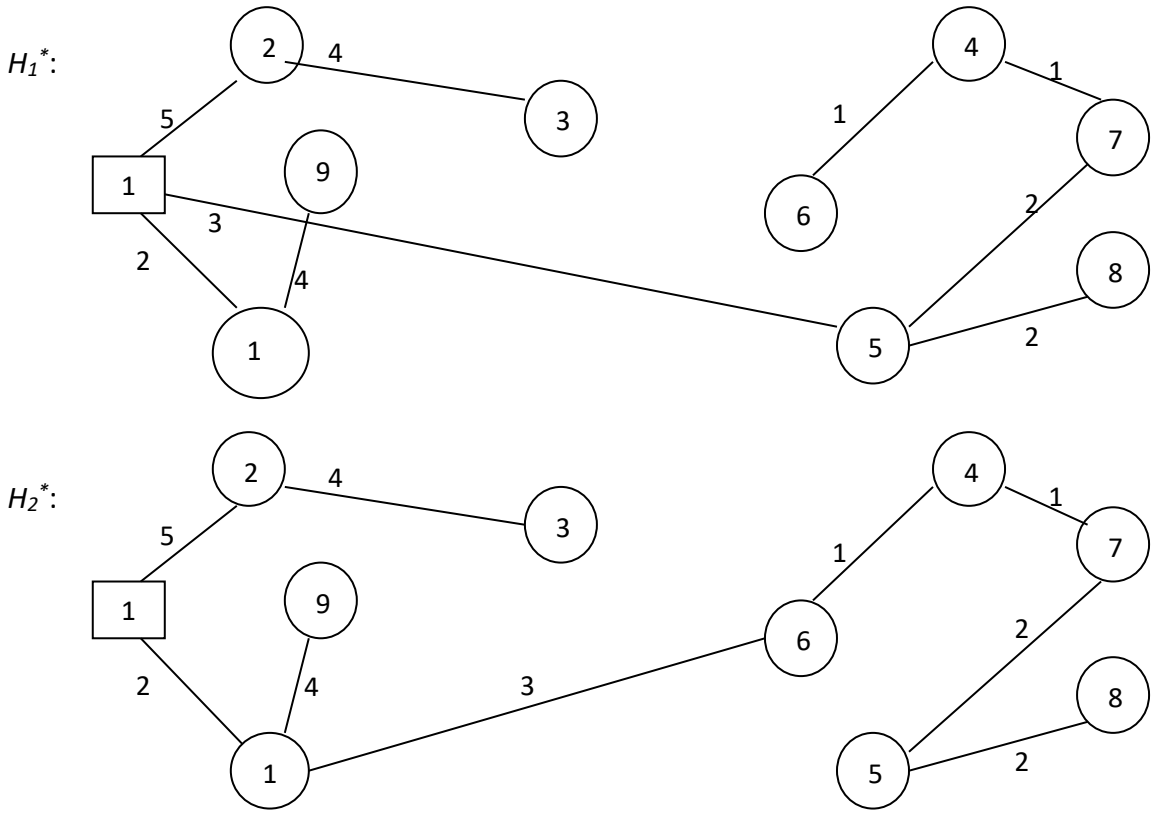

Рис. 6.5 – Оптимальні розв'язки задачі

Розрахуємо вагу мінімального кістякового дерева:

$$\omega(l) = 1 + 1 + 2 + 2 + 2 + 3 + 4 + 4 + 5 = 24.$$

Відповідь. Для прокладки системи електропостачання житлового району знадобиться мінімум 24 км кабельних ліній, укладеного двома можливими способами так, як показано на рис. 6.5.

6.4 Задача про максимальний потік та її розв'язування за алгоритмом Форда-Фалкерсона

6.4.1 Постановка задачі про максимальний потік

Розглянемо $G = (V, E)$ – орієнтовний граф. Нехай вершина $S'\epsilon V$ – виток, а $S''\epsilon V$ – стік. Припустимо, в G існує рівно один виток S' і один стік S''.

Функція φ: A$\rightarrow R_+$ – *пропускна спроможність*, тобто $\varphi(l)$ – пропускна спроможність дуги l, яка визначає максимальну кількість потоку, що проходить по дузі l.

Потік $\mu(u, v)$ – величина, що задовольняє умови:

- обмеження пропускної здатності:

$$0 \le \mu(u, v) \le \phi(u, v);$$

- умова збереження потоку для всіх $v \in V \setminus \{s, t\}$:

$$\sum_{u:(u,v)\in E} \mu(u, v) = \sum_{w:(v,w)\in E} \mu(v, w).$$

Величиною потоку називають число

$$\rho(\mu) = \sum_{(S',v)\in E} \mu(S', v).$$

Теорема 6.1 В графі $G = (V, E)$ сумарний потік вздовж всіх дуг, що виходять з витоку, дорівнює сумарному потоку вздовж всіх дуг, що входять у стік, тобто:

$$\sum_{(S\prime,v)\in E} \mu(S', v) = \sum_{(v,S\prime\prime)\in E} \mu(v, S'').$$

Постановка задачі про максимальний потік.

Нехай $G = (V, E) -$ орієнтований граф, що має одне джерело та один стік. Нехай φ: A$\rightarrow R_+$ – пропускна спроможність дуг цього графа. $M = \{\mu\} -$ множина всіх потоків у графі G, де $\mu -$ величина потоку.

Серед усіх потоків у графі G необхідно знайти потік максимальної величини, тобто:

$$\rho(\mu) \rightarrow max\ ,\ \text{де}\ \mu \in M.$$

6.4.2 Алгоритм Форда-Фалкерсона розв'язання задачі про максимальний потік

Викладемо *алгоритм Форда-Фалкерсона для знаходження максимального потоку* у загальному вигляді.

Вхідні дані:

орієнтована мережа

$$G = (V, E),$$

у якій:

- $S' \in V$ – джерело,
- $S'' \in V$ – стік ($S' \neq S''$),
- кожній дузі $(u, v) \in E$ поставлено у відповідність пропускну здатність $\phi(u, v) \geq 0$.

Вихідні дані:

максимальний потік ρ з вершини S' до вершини S''.

Залишкова мережа G_μ – мережа, що описує можливість збільшення потоку.

Кроки алгоритму

1. *Ініціалізація.*

Покласти початковий потік рівним нулю:

$$\mu(u, v) = 0 \ \ \forall (u, v) \in E.$$

2. *Побудова залишкової мережі.*

Для кожної дуги $(u, v) \in E$ визначити залишкову пропускну здатність:

$$\phi_\mu(u, v) \ = \ \phi(u, v) \ - \ \mu(u, v),$$

а також зворотну дугу (v, u) з пропускною здатністю:

$$\phi_\mu(v, u) \ = \ \mu(u, v).$$

3. *Пошук збільшувального шляху.*

У залишковій мережі G_μ знайти будь-який шлях P від джерела S' до стоку S'', такий що

$$\phi_\mu(u, v) > 0 \text{ для всіх дуг } (u, v) \in P.$$

4. *Збільшення потоку.*

Визначити мінімальну залишкову пропускну здатність уздовж шляху:

$$\alpha = \min_{(u,v) \in P} \phi_\mu(u, v).$$

Для кожної дуги $(u, v) \in P$:

- якщо $(u, v) \in E$, покласти
$$\mu(u, v) = \mu(u, v) + \alpha;$$
- якщо $(v, u) \in E$, покласти
$$\mu(v, u) = \mu(v, u) - \alpha.$$

5. *Повторення.*

Повернутися до кроку 2, доки існує збільшувальний шлях із S' до S''.

6. *Завершення.*

Коли збільшувальних шляхів більше не існує, потік ρ є *максимальним.*

Результат

Максимальний потік дорівнює

$$\rho(\mu) = \sum_{(S', v) \in E} \mu(S', v).$$

Зауваження:

- алгоритм є *жадібним:* на кожному кроці локально збільшує потік;
- час роботи залежить від способу вибору збільшувального шляху;
- модифікація з пошуком найкоротшого (за кількістю дуг) шляху відома як *алгоритм Едмондса-Карпа*.

Приклад 6.2 Задано зважений орієнтований граф (мережева модель). Для заданої мережі:

- скласти змістовну постановку певної техніко-економічної задачі як задачі про максимальний потік;
- знайти потік максимальної величини за допомогою алгоритму Форда-Фалкерсона;
- зробити висновки в термінах постановки техніко-економічної задачі.

Розв'язання.

Змістовна постановка задачі

Великий логістичний центр здійснює приймання вантажів із двох морських портів. Порти з'єднані зі складським комплексом мережею автомобільних шляхів, кожен з яких має обмежену пропускну здатність, що визначається станом дорожнього покриття та організацією руху. Необхідно визначити такий розподіл вантажопотоку в межах заданої транспортної мережі, за якого сумарний обсяг вантажів, доставлених до логістичного центру з обох портів за одиницю часу, буде максимальним.

Математична модель в термінах теорії графів.

Нехай S_1', S_2', – два морські порти, S'' – логістичний центр (складський центр); вершини *1–4* – проміжні вузли, через які проходить дорожня мережа; *дуги мережі* – це шосейні дороги, що з'єднують морські порти з містом; *вага дуги* – максимальний пасажиропотік, який може пройти цією дорогою за одиницю часу.

Математична модель у формі орієнтованого графа подана на рис. 6.6.

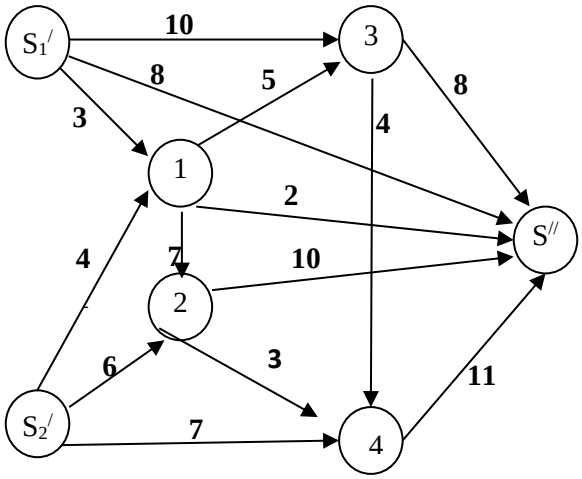

Рис. 6.6 – Мережева модель задачі

Розв'яжемо дану задачу як задачу про максимальний потік у мережі з використанням алгоритму Форда-Фалкерсона.

Спочатку потрібно звести орієнтовний граф до виду, в якому наявний лише один сток і один виток. У даному випадку вводимо фіктивний виток S', з'єднуючи його з фактичними витоками S_1' і S_2',. Ваги з'єднуючих дуг визначаємо як суму ваг тих дуг, що виходять з S_1', S_2',. Вага дуги (S', S_1') дорівнюватиме 21, а дуги $(S', S_2') -$ 17 (див рис. 6.7).

Орієнтовані шляхи від умовного джерела S' до стоку S'' та реалізація алгоритму Форда-Фалекрсона виписано нижче для поставленої задачі.

1) $S' \to S_2' \to 4 \to S'', \quad \alpha_1 = min\{17; 7; 11\} = 7, \quad (S_2'; 4) \downarrow;$
2) $S' \to S_2' \to 1 \to 2 \to S'', \quad \alpha_2 = min\{10; 4; 7; 10\} = 4, \quad (S_2'; 1) \downarrow;$
3) $S' \to S_2' \to 2 \to S'', \quad \alpha_3 = min\{6; 6; 6\} = 6, \quad (S'; S_2'), (S_2'; 2), (2; S'') \downarrow;$
4) $S' \to S_1' \to 3 \to S'', \quad \alpha_4 = min\{21; 10; 8\} = 8, \quad (3; S'') \downarrow;$
5) $S' \to S_1' \to 3 \to 4 \to S'', \quad \alpha_5 = min\{13; 2; 4; 4\} = 2, \quad (S_1'; 3) \downarrow;$
6) $S' \to S_1' \to S'', \quad \alpha_6 = min\{11; 8\} = 8, \quad (S'; S'') \downarrow;$
7) $S' \to S_1' \to 1 \to 3 \to 4 \to S'', \quad \alpha_7 = min\{3; 3; 5; 2; 2\} = 2, (3; 4), (4; S'') \downarrow;$
8) $S' \to S_1' \to 1 \to S'', \quad \alpha_8 = min\{1; 1; 2\} = 1, \quad (S'; S_1'), (S_1'; 1) \downarrow.$

Розв'язання оформимо на мережевій моделі (рис. 6.7).

Величина максимального потоку, знайденого за допомогою даного алгоритму, дорівнює:

$$\rho_{max} = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_6 + \alpha_7 + \alpha_8 =$$
$$= 7 + 4 + 6 + 8 + 2 + 8 + 2 + 1 = 38.$$

Оптимальне розв'язання цієї задачі, тобто потік максимальної величини, зображено на рис. 6.8.

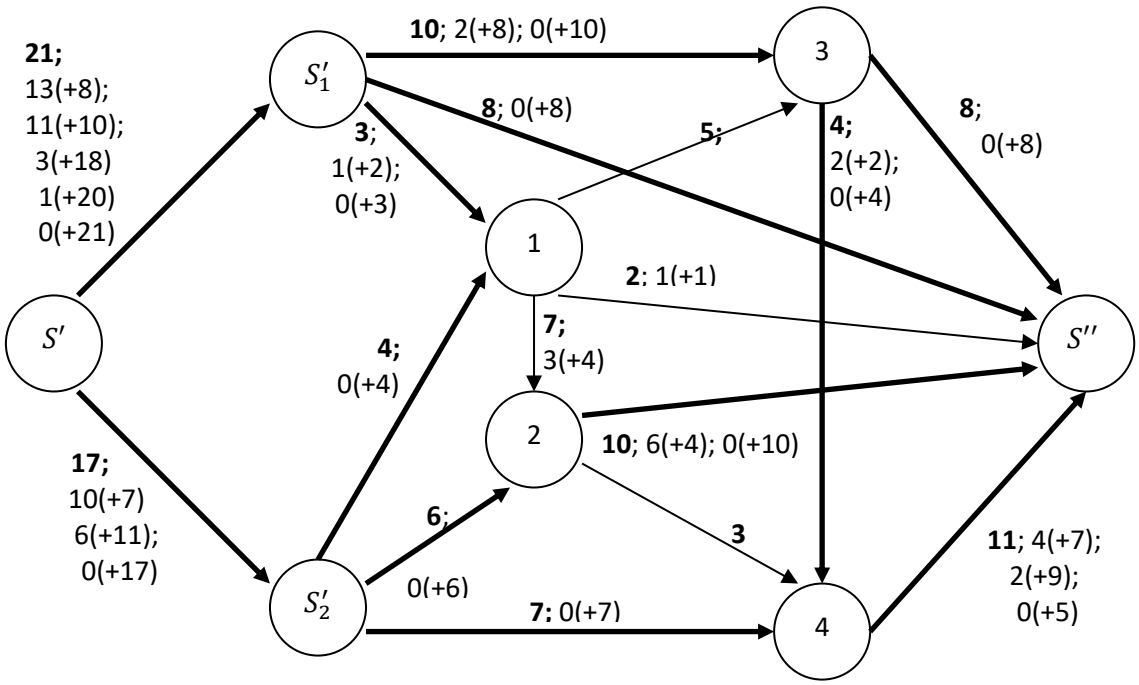

Рис. 6.7 – Розв'язання задачі про максимальний потік за алгоритмом Форда-Фалкерсона

Відповідь. Перевезення вантажу з морських портів до логістичного центру слід здійснювати відповідно до схеми, поданої на рис. 6.8. При цьому сумарний обсяг вантажів буде максимальною і становитиме 38 т на годину (наприклад).

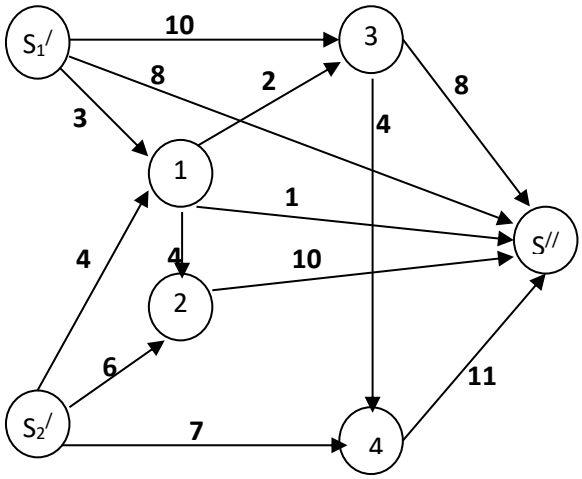

Рис. 6.8 – Оптимальний розв’язок задачі (максимальний потік)

6.5 Задача про максимальний потік як задача лінійного програмування

Задача максимального потоку може бути подана у вигляді задачі лінійного програмування. Водночас застосування симплекс-методу для безпосереднього розв’язання мережевих задач є неефективним. Разом із тим аналіз мережевих задач у формі задач лінійного програмування дає змогу виявляти моделі ЛП, які на перший погляд не мають мережевої структури, але можуть бути зведені до мережевих задач безпосередньо або після відповідних перетворень. Перевагою такого підходу є можливість використання спеціалізованих мережевих алгоритмів, що забезпечують істотне підвищення обчислювальної ефективності.

Побудуємо математичну модель задачі лінійного програмування про максимальний потік.

Змінні моделі. Нехай X_{ij} – потік, що проходить через дугу (i, j), тобто змінним лінійної моделі відповідає кожна дуга мережевої моделі.

Цільова функція моделі має формалізувати мету задачі – потік, що виходить із джерела, повинен бути максимальним. Сумарний потік, що виходить із джерела S', згідно з введеними змінними, дорівнює сумі $\sum_j X_{S'j}$ за всіма дугами, що виходять із вершини S'. Тоді цільову функцію можна записати у вигляді:

$$\rho = \sum_j X_{S'j} \to max.$$

Система обмежень. Відповідно до техніко-економічного змісту введених змінних, на них мають бути накладені такі обмеження:

1) *сумарний потік, що виходить із джерела S', дорівнює сумарному потоку, який надходить у стік S''.* Математично це можна записати так:

$$\sum_{(S';\ j)} X_{S'j} = \sum_{(i;\ S'')} X_{iS''};$$

2) *для будь-якої проміжної вершини мережі k сумарний потік, що входить у неї, дорівнює сумарному потоку, що виходить із неї.* Математично це записується так:

$$\sum_{(i;\ k)} X_{ik} = \sum_{(k;\ j)} X_{kj}\ ;$$

3) з визначення поняття потоку в мережі випливає умова обмеження на величину потоку:

$$0 \le X_{ij} \le \phi_{ij},$$

де ϕ_{ij}– пропускна здатність дуги (i, j).

Таким чином, математична лінійна модель задачі про максимальний потік має вигляд:

$$\rho = \sum_{j} X_{S'j} \to max; \tag{6.1}$$

$$\begin{cases} \sum\limits_{(S';\ j)} X_{S'j} = \sum\limits_{(i;\ S'')} X_{iS''}\ ; \\ \sum\limits_{(i;\ k)} X_{ik} = \sum\limits_{(k;\ j)} X_{kj}; \\ 0 \le X_{ij} \le \phi_{ij}. \end{cases} \tag{6.2}$$