ЗМІСТ

ВСТУП

Орієнтовану мережу можна інтерпретувати як деяку транспортну мережу і використовувати її для вирішення задач про потоки речовин в системі трубопроводів. Уявімо, що деякий продукт передається по системі від джерела, де даний продукт виробляється, до стоку, де він споживається. Джерело виробляє продукт з деякою максимальною швидкістю, а стік з тією ж швидкістю споживає продукт. Потоком продукту в будь-якій точці системи є швидкість руху продукту. За допомогою транспортних мереж можна моделювати течію рідин по трубопроводах, рух деталей на складальних лініях, передачу струму по електричним мережам, інформації - в інформаційних мереж і т. д. Кожне орієнтоване ребро мережі можна розглядати як канал, по якому рухається продукт. Кожен канал має задану пропускну здатність, яка характеризує максимальну швидкість переміщення продукту по каналу. Вершини є точками перетину каналів. Через вершини, відмінні від джерела і стоку, продукт проходить не накопичуючись.

У задачі про максимальний потік ми хочемо знайти максимальну швидкість пересилання продукту від джерела до стоку, при якій не будуть порушуватися обмеження пропускної здатності. Ця проблема була поставлена Т.Є. Харрісом навесні 1955 року, який разом з відставним генералом Ф.С. Россом запропонував спрощену модель залізничного транспортного потоку і висунув саме цю спеціальну задачу як центральну, підказаною цією моделлю. Незабаром після цього був висловлений в якості гіпотези, а потім і встановлений головний результат - теорема «Про максимальний потік і мінімальний розріз».

У даній роботі розглядається узагальнений метод Форда-Фалкерсона для транспортної мережі з обмеженою пропускною здатністю дуг а також з ціною за транспортування одиниці продукту через мережу. В якості алгоритму побудови рішення був обраний алгоритм Баскера-Гоуєна.

1 Задача про максимальний потік

1.1 Постановка задачі

Орієнтованої мережею називається граф $G = [V, E]$, який складається із сукупності V елементів $x, y, ...$ разом з множиною E деяких впорядкованих пар (x, y) елементів, взятих з V.

Вузли - елементи множини V.

Дуги - елементи множини E. Можливість дуги (x,x) вилучається.

Поставимо кожній дузі (x, y) у відповідність деяке число $c(x, y)$,яке називається пропускною здатністю дуги. Пропускна здатність показує яка кількість речовини може пройти по цій дузі в одиницю часу.

Потоком (flow) в мережі є дійсна функція $f : V x V -> R$ задовольняє трьом умовам:

а) обмеження пропускної здатності

$$\forall u,v \in V f(u,v) \leqslant c(u,v)$$

Потік з однієї вершини в іншу не повинен перевищувати задану пропускну здатність.

б) антисиметричність

$$f(u,v) = -f(v,u) \forall u,v \in V$$

Потік з вершини u в вершину v протилежний потоку у зворотному напрямку.

в) збереження потоку

$$\forall u \in V/s,t \sum_{v \in V} f(u,v) = 0$$

Сумарний потік, що виходить з вершини, що не є джерелом або стоком дорівнює нулю. Величина потоку визначається як сумарний потік, що виходить з джерела.

$$|f| = \sum_{v \in V} f(s,v)$$

Будемо називати s джерелом t стоком, а інші вузли - проміжними.

Задача про максимальний потік (maximum flow poblem) полягає в знаходженні потоку максимальної величини. Математично постановка виглядає так:

$$\begin{aligned} &max \quad v = f(s,V) \\ &f(x,V) - f(V,X) = 0, \qquad x \neq s,t, \\ &0 \leqslant f(x,y) \leqslant c(x,y), \quad (x,y) \in E, \end{aligned}$$

1.2 Метод Форда-Фалкерсона

Метод Форда-Фалкерсона базується на трьох важливих концепціях. Це залишкові мережі, що збільшують шляхи і розрізи. Метод є ітеративним. Спочатку значення потоку встановлюється нуль. $f(u,v) = 0 \forall u,v \in V$ На кожній ітерації величина потоку збільшується за допомогою пошуку збільшуючого шляху (деякого шляху від джерела до стоку вздовж якого можна відправити більший потік) і подальшого збільшення потоку. Цей процес повторюється до тих пір, поки вже неможливо відшукати збільшуваного шлях.

Залишкові мережі

Нехай задана транспортна мережа $G(V,E)$ з джерелом s і стоком t. Нехай f деякий потік в G. Розглянемо пару вершин $u,v \in V$. Величина додаткового потоку, який ми можемо направити з u в v, щоб не перевищити пропускну здатність $c(u,v)$ є залишковою пропускною здатністю ребра (u,v) і задається формулою:

$$c_f(u,v) = c(u,v) - f(u,v)$$

Для транспортної $G(V,E)$ мережі і потоку f залишковою мережею в G, породженою потоком f є мережа $G_f = (V,E_f)$ где

$$E_f = (u,v) \in VxV : c_f(u,v) > 0$$

Таким чином по кожному ребру залишкової мережі або залишковому ребру можна направити потік більше нуля.

Збільшуючі шляхи

Для заданої транспортної мережі $G = (V,E)$ і потоку f збільшуючим шляхом p є простий шлях з s в t в залишковій мережі G_f. Максимальна величина, на яку можна збільшити потік уздовж кожного ребра p збільшуючого шляху називається пропускною здатністю шляху і задається формулою:

$$c_f(p) = min\{c(u,v) : (u,v) \in p\}$$

.

Розрізи транспортних мереж

Розрізом транспортної мережі $G(V,E)$ називається розбиття множини вершин на множини S та T такі що $s \in S, t \in T$. Якщо f потік - то чистий потік через розріз (S,T) визначимо як $f(S,T)$. Пропускну здатність розрізу (S,T) визначимо відповідно $c(S,T)$. Мінімальним розрізом є розріз, пропускна здатність якого серед всіх розрізів мінімальна. Як видно, потік через розріз, на відміну від пропускної здатності розрізу, може включати і від'ємні доданки.

Теорема 1 (Про максимальний потік і мінімальний розріз). Для будь-якої мережі максимальна величина потоку з s в t дорівнює мінімальній пропускній здатності розрізу, що відокремлює s и t. [1]

2 Задача про максимальний потік мінімальної вартості

2.1 Постановка задачі

Нехай кожній дузі відповідає не лише пропускна здатність а також і велечина $c_{ij} > 0$ яка дорівнює вартості транспортування одиниці товару через ребро $(i,j) \in E$ мережі. Задача пошука потока із s в t заданої потужності v і мінімальної вартості має вигляд:

$$Z = \sum_{(i,j) \in E} c_{ij} x_{ij} \to min_x;$$

$$\sum_{i:(i,j) \in E} x_{ij} - \sum_{k:(j,k) \in E} x_{jk} = \begin{cases} -v, & j = s; \\ 0, & j \neq s,t; \\ v, & j = t; \end{cases}$$

$$0 \leq x_{ij} \leq b_{ij}, (i,j) \in E$$

2.2 Алгоритм Баскера-Гоуєна

Для розв'язування задачі про максимальний потік мінімальної вартостіі будемо використовувати алгоритм Баскера-Гоуєна:

а) Знайдемо поток мінімальної вартості із s в t.

б) З'ясуемо максимальну величину потоку яку можна пропустити через цей шлях.

в) Якщо ця велечина більша за потрібну потужність мережі візмемо її рівною потрібній потужності.

г) Збільшити поток по цьому ланцюгу на максимальну величину(але так щоб загальний поток через мережу не перевищував потрібний)

д) Розрахувати ціну за транспортування потужності. Додати до загальної ціни потоку.

е) Якщо поток по мережі дорівнюе заданому то припинити роботу алгоритму. Інакше перейти на крок a.

3 Приклади

Задача 1

У деякої компанії "Аврора"є фабрика в Берліні, що виробляє стільці, а в Бремені склад, де вони зберігаються. Компанія орендує місце на вантажівках інших фірм для доставки стільців з фабрики на склад. Оскільки вантажівки їздять по певних маршрутах між містами і мають обмежену вантажопідйомність, компанія може перевозити не більше певної кількості ящиків на день між містами. Також, через трафік в містах за день по місту може проїхати тільки певна кількість машин. Компанія не може вплинути на маршрути і пропускну здатність. Її завдання визначити, яку найбільшу кількість ящиків за один день можна відвантажувати, а потім виробляти саме таку кількість, оскільки не має сенсу виробляти більше стільців, чим можна відправити на склад.

Фабрику будемо вважати джерелом (s), склад - стоком (t).
Маршрути між містами будемо вважати ребрами мережі.
Вантажопідйомність вантажівок будемо вважати обмеженням пропускної спроможності на ребрах $c(u,v)$. Трафік в містах будемо вважати обмеженням пропускної здатності в вершинах $k(v)$. Занесемо дані в таблицю, де кожен елемент - обмеження пропускної здатності між відповідними містами (нуль означає що маршрут відсутній):

0	10	15	0	0
0	0	20	25	0
0	0	0	0	30
0	0	0	0	20
0	0	0	0	0

обмеження пропускних здатності у містах задано наступним вектором:

100	20	30	40	100

В результаті роботи програми було отримано наступне рішення:

0	10	15	0	0
0	0	10	0	0
0	0	0	0	25
0	0	0	0	0
0	0	0	0	0

Максимальний потік в мережі - 25 одиниць продукції.

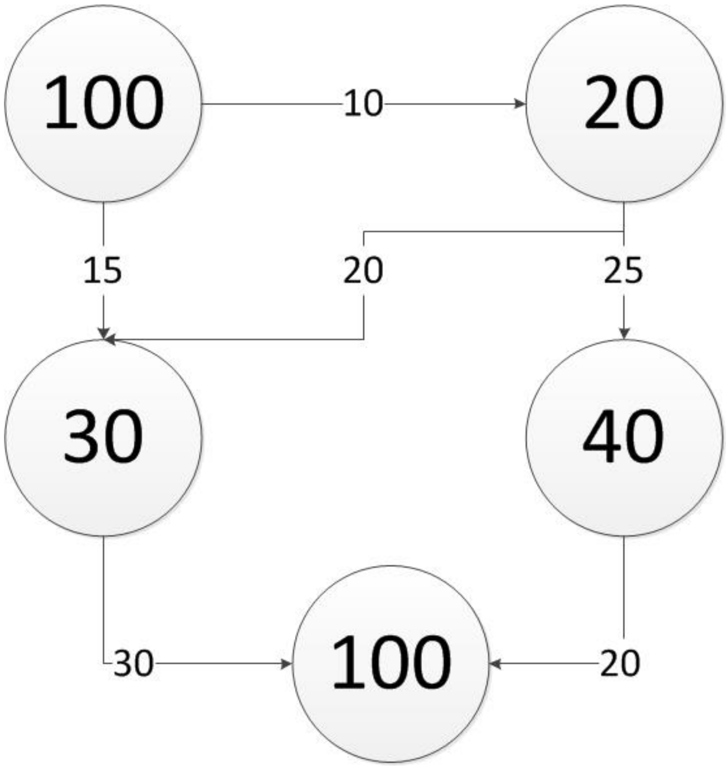

Рис. 3.1 Транспортна мережа

Задача 2

У пік туристичного сезону часто складно доїхати з Києва до Одеси, через те що залізниця не може перевезти всіх бажаючих, тому що вона може запустити тільки певне число поїздів по маршруту. Керівництво залізниці хоче з'ясувати, який максимальний потік пасажирів можливо перевезти з Одеси в Київ, і порівняти з очікуваним потоком пасажирів, який відомий на основі статистичних даних за попередні роки. Кожен маршрут між станціями може пропускати тільки певна кількість вагонів у день. Також кожна станція може обробляти тільки певну кількість вагонів у день (приєднувати і перевіряти). Керівництво знає, що для задоволення потреб необхідно щоб щодня до Одеси можна було перевезти 250 чоловік.

Зведемо дану задачу до задачі про максимальний потік.

Будемо вважати Київ джерелом s а Одесу стоком t. Колії між станціями будемо вважати ребрами мережі, з обмеженнями на пропускну здатність $c(x,y)$ заданої в вигляді таблиці.

Станції будемо вважати вузлами транспортної мережі з обмеженнями на вузлах $k(x)$ заданими в векторній формі.

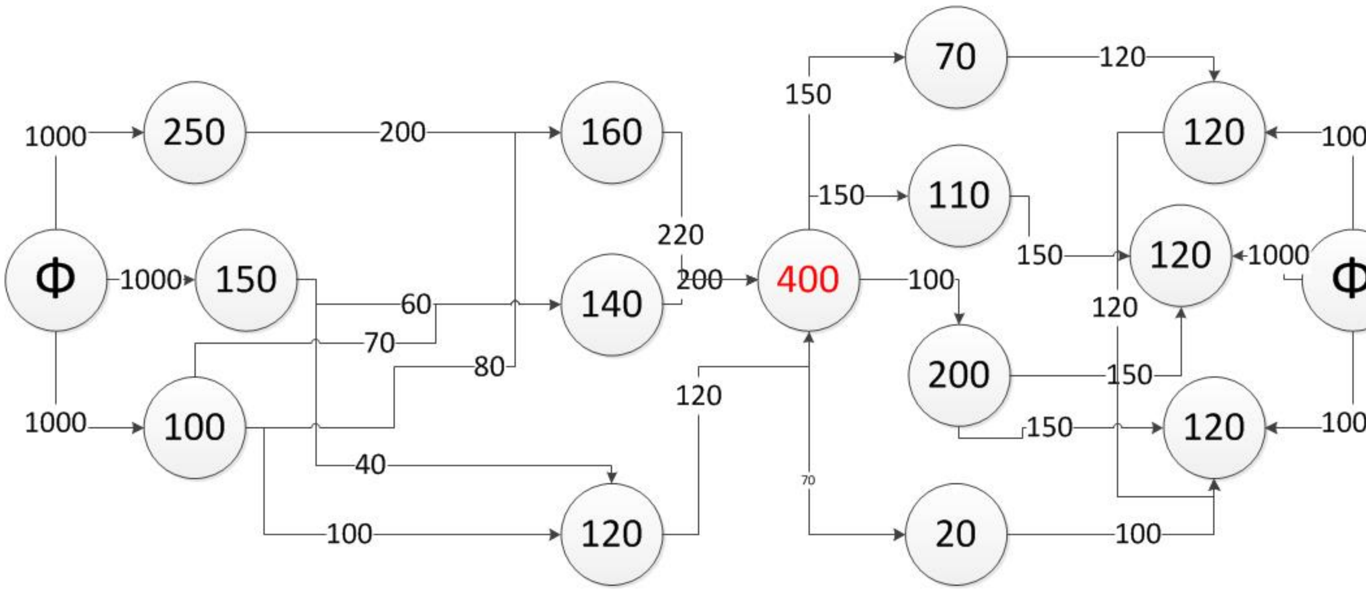

Рис. 3.2 Транспортна мережа

0	100	70	180	0	0	0	0	0	0	0	0
0	0	0	0	120	0	0	0	0	0	0	0
0	0	0	0	0	70	0	0	0	0	0	0
0	0	0	0	0	130	60	0	0	0	0	0
0	0	0	0	0	0	0	90	100	0	0	0
0	0	0	0	0	0	0	0	70	70	0	0
0	0	0	0	0	0	0	0	70	30	0	0
0	0	0	0	0	0	0	0	100	0	120	90
0	0	0	0	0	0	0	0	0	0	0	110
0	0	0	0	0	0	0	0	60	0	150	0
0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0

обмеження пропускних здатності на станціях задано наступним вектором:

900	100	100	100	100	100	150	100	300	120	250	900

В результаті роботи програми було отримано наступне рішення:

0	0	0	0	0	0	0	0	0	0	0	0
100	0	0	0	0	0	0	0	0	0	0	0
70	0	0	0	0	0	0	0	0	0	0	0
90	0	0	0	0	0	0	0	0	0	0	0
0	100	0	0	0	0	0	0	0	0	0	0
0	0	70	30	0	0	0	0	0	0	0	0
0	0	0	60	0	0	0	0	0	0	0	0
0	0	0	0	90	0	0	0	0	0	0	0
0	0	0	0	10	70	30	0	0	0	0	0
0	0	0	0	0	30	30	0	0	0	0	0
0	0	0	0	0	0	0	0	0	60	0	0
0	0	0	0	0	0	0	90	110	0	60	0

Максимальний потік 260 пасажирів на день. Цього вистачає для задоволення попиту.

Задача 3

У старому центрі одного з міст рух машин влаштовано так, що великій кількості машин, що їде ввечері в спальний район в будь-якому випадку доводиться проїжджати через кільцеве перехрестя, із за чого створюються затори. ДАІ спільно з комунальними підприємствами вирішило, що місто потребує побудови нової дороги, яка б розвантажила це перехрестя. Але їм необхідно знати, скільки смуг необхідно новій дорозі. Для цього необхідно дізнатися, скільки машин обиратимуть нову дорогу замість старої. Кожна дорога має обмежену пропускну спроможність. Кожне перехрестя також має обмежену пропускну спроможність.

Зведемо дану задачу до задачі про максимальний потік.

Будемо вважати центр міста - джерелом s а спальний район стоком t. Шляхи між перехрестями будемо вважати ребрами мережі, з обмеженнями на пропускну здатність $c(x,y)$ заданої в вигляді таблиці.

Перехрестя будемо вважати вузлами транспортної мережі з обмеженнями на вузлах $k(x)$ заданими в векторній формі.

Для вирішення завдання знайдемо спочатку максимальний потік вихідної мережі. Потім додамо нове ребро, яке буде відображати дорогу, яку збираються будувати. Задамо цьому ребру нескінченну пропускну здатність. Знайдемо новий максимальний потік. Різниця між цими двома величинами і буде шуканим числом машин, які користуватимуться новою дорогою.

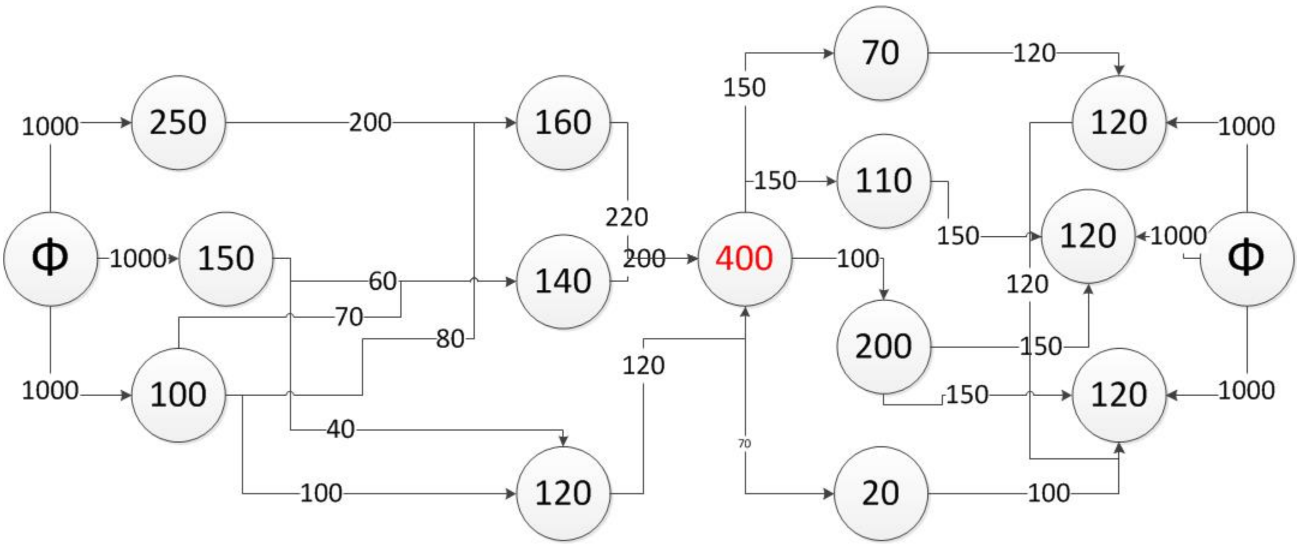

Рис. 3.3 Транспортна мережа

0	10000	10000	10000	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	100	70	80	0	0	0	0	0	0	0	0
0	0	0	0	40	60	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	200	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	120	0	0	0	0	0	0	0
0	0	0	0	0	0	0	200	0	0	0	0	0	0	0
0	0	0	0	0	0	0	220	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	150	150	100	70	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	120	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	150	0
0	0	0	0	0	0	0	0	0	0	0	0	0	150	150
0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	120
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

обмеження пропускних здібностей на перехрестях задано наступним вектором:

10000	100	150	250	120	140	160	400	70	110	200	20	120	120	120	10

В результаті роботи програми було отримано наступне рішення:

0	100	80	120	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	20	60	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	70	110	100	20	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	70	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	110	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	10	90	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	70
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	110
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Максимальний потік 300 машин.

Тепер введемо нове ребро , яке буде відповідати новій дорозі, що йде в об'їзд перехрестя

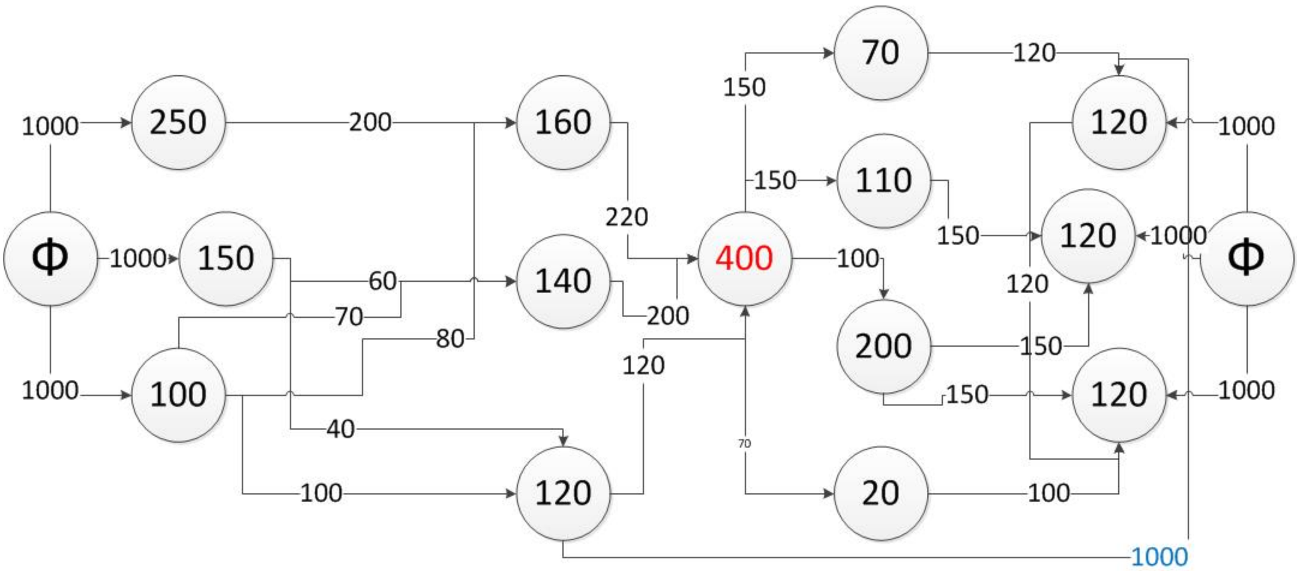

Рис. 3.4 Транспортна мережа

0	10000	0	10000	0	0	0	0	0	0	0	0	0	0	0
0	0	10000	0	100	70	80	0	0	0	0	0	0	0	0
0	0	0	0	40	60	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	200	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	120	0	0	0	0	1000	0	0
0	0	0	0	0	0	0	200	0	0	0	0	0	0	0
0	0	0	0	0	0	0	220	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	150	150	100	70	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	120	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	150	0
0	0	0	0	0	0	0	0	0	0	0	0	0	150	[illegible]
0	0	0	0	0	0	0	0	0	0	0	0	0	0	[illegible]
0	0	0	0	0	0	0	0	0	0	0	0	0	0	[illegible]
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Потім подивимося на скільки збільшиться потік при додаванні нового ребра, що б встановити скільки смуг повинно бути у нової дороги. Результуюче рішення:

0	100	90	160	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	90	10	0	0	0	0	0	0	0	0	0	0
0	0	0	0	30	60	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	160	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0
0	0	0	0	0	0	0	70	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	160	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	110	100	20	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	110	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	10	90	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	110
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Ми бачимо, що в результаті максимальний потік машин збільшиться на 50 одиниць. Значить достатньо буде побудувати дорогу з однією смугою в кожну напрямку.

Задача 4

Зерно з трьох зерносховищ доставляється на вантажівках чотирьом птахівничим фермам, при цьому деякі зерносховища не можуть безпосередньо поставляти зерно певним фермам. Пропускна здатність маршрутів від зерносховищ до птахівничих ферм обмежена кількістю використовуваних вантажівок і числом виконуваних щодня рейсів. Також дозволені перевезення зерна з першого в друге сховище, а з другого в третє, об'ємом до 50-ти тонн. Необхідно встановити, чи буде задоволено попит ферм.

Зведемо дану задачу до задачі про максимальний потік.

Додамо фіктивне джерело s і фіктивний стік t. З'єднаємо джерело з усіма зерносховищами, пропускну здатність цих дуг покладемо нескінченно великою (в рамках конкретного завдання), наприклад 1000.Всі ферми з'єднаємо зі стоком дугами, також з нескінченною пропускною здатністю. Пропускні здатність інших маршрутів $c(x,y)$задамо в вигляді таблиці.

Сховища і ферми будемо вважати вузлами транспортної мережі з обмеженнями на вузлах $k(x)$ заданими в векторній формі.

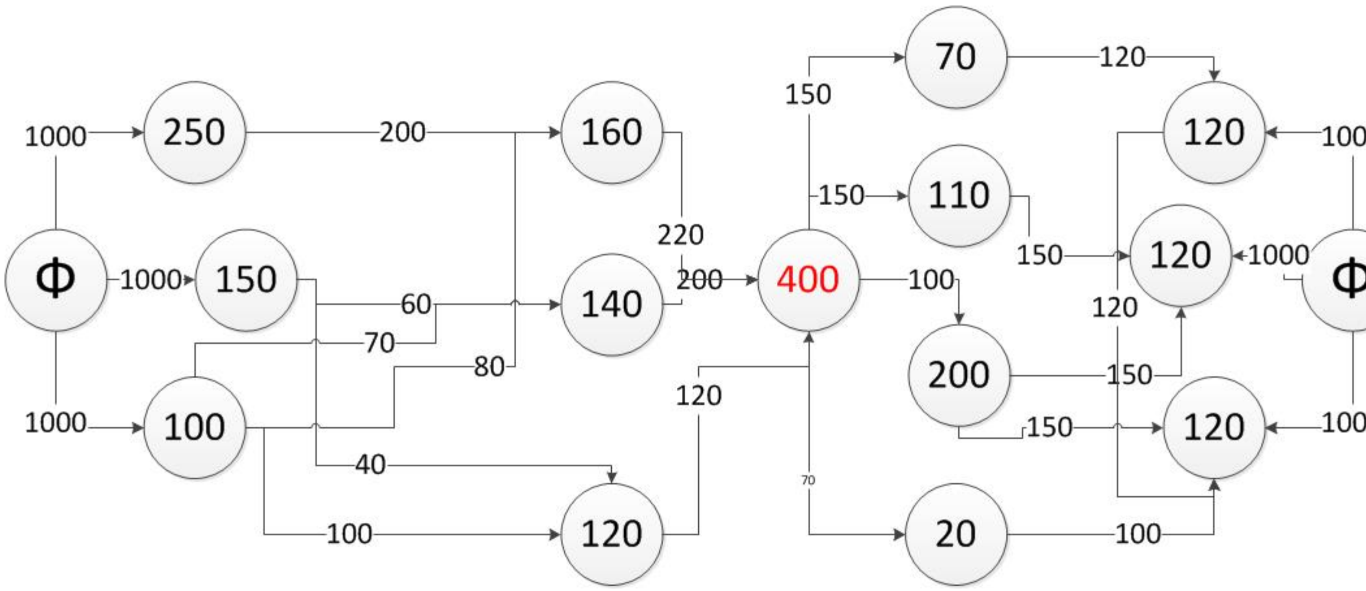

Рис. 3.5 Транспортна мережа

0	1000	1000	1000	0	0	0	0	0
0	0	50	0	40	10	0	80	0
0	0	0	50	0	0	15	90	0
0	0	0	0	100	50	30	40	0
0	0	0	0	0	0	0	0	1000
0	0	0	0	0	0	0	0	1000
0	0	0	0	0	0	0	0	1000
0	0	0	0	0	0	0	0	1000
0	0	0	0	0	0	0	0	0

обмеження пропускної здатності на станціях задано наступним вектором:

1000	40	80	180	180	10	60	30	1000

В результаті роботи програми було отримано наступне рішення:

0	40	45	140	0	0	0	0	0
0	0	0	0	40	0	0	0	0
0	0	0	0	0	0	15	30	0
0	0	0	0	100	10	30	0	0
0	0	0	0	0	0	0	0	140
0	0	0	0	0	0	0	0	10
0	0	0	0	0	0	0	0	45
0	0	0	0	0	0	0	0	30
0	0	0	0	0	0	0	0	0

Максимальний потік 225 тонн на день. Цього не вистачає для задоволення попиту. З рішення видно, що перша ферма отримує 140 тонн при попиті в 180, а третя 45, при попиті 60. У інших двох ферм попит буде задоволений.

4 Висновки

У даній роботі було розглянуто узагальнення задачі про максимальний потік для транспортної мережі з ціною на транспортування одиниці товару через ребро. Як алгоритм рішення був обрний алгоритм Баскера-Гоуєна.Була розроблена програма на мові c# що реалізує даний метод.Робота програми була протестовна на декількох прикладних задачах.

СПИСОК ВИКОРИСТАННОЇ ЛІТЕРАТУРИ

[1] Форд, Л.Р. Потоки в сетях. /Л.Р. Форд, Д.Р. Фалкерсон - Москва : Мир, 1966. - 273 с.

[2] Таха Хемди А. ВВедение в исследование операций /А. Таха - Москва : Вильямс, 2006. - 912 с.

[3] Вагнер Г. Основы исследования операций. Том 1./Г. Вагнер - Москва : Мир, 1973. - 336 с.

[4] Кормен Томас Х. Алгоритмы: построение и анализ./К.Х. Томас, Ч.Л. Лейзерсон, Р.Д. Риверс, К. Штайн - Москва : Вільямс, 2005. - 1296 с.

[5] Ху Т. Целочислнное програмированние и потоки в сетях/Т. Ху - Москва : Мир, 1974. - 513 с.

[6] Арсирий А.В. Сетевые модели./ А.В.Арсирій, Б.Ф.Трофимов, Є.М. Страхов - Одеса : Одеський національний університет імені І.І.Мечникова, 2011. - 42 с.

ДОДАТОК А

```
 private static FlowCost CalculateFlow(int[,] flows, int[,] costs, int s, int t, int
    neededFlow)
      {
          var totalFlow = 0;
          var totalCost = 0;
          if (s == t)
              throw new ArgumentException("input␣and␣output␣should␣be␣different␣points");
          MilestoneHist[] path = bfs(flows, costs, s, t);
          while (path[t] != null)
          {
              int maxFlow = getMaxFlowForPath(flows, path, s, t);
              if (neededFlow - totalFlow < maxFlow)
              {
                  maxFlow = neededFlow - totalFlow;
                  flows = updateFlows(maxFlow, flows, path, s, t);
                  totalCost += maxFlow * path[t].totalCost;
                  path = bfs(flows, costs, s, t);
                  totalFlow += maxFlow;
                  return new FlowCost { totalCost = totalCost, resultingFlows = flows };
              }
              flows = updateFlows(maxFlow, flows, path, s, t);
              totalCost += maxFlow * path[t].totalCost;
              path = bfs(flows, costs, s, t);
              totalFlow += maxFlow;
          }
          return new FlowCost { totalCost = totalCost, resultingFlows = flows };
      }

public static MilestoneHist[] bfs(int[,] rGraph, int[,] costGraph, int s, int t)
      {
          var debug = 0;
          int Size = rGraph.GetLength(0);
          MilestoneHist[] parent = new MilestoneHist[Size];
          parent[s] = new MilestoneHist { totalCost = 0, pointNum = -1 };
          bool[] visited = new bool[Size];

          Queue q = new Queue();
          q.Enqueue(s);

          while (q.Count != 0)
          {
              int u = (int)q.Dequeue();
              debug++;
              if (debug > 10000)
                  throw new StackOverflowException();

              for (int v = 0; v < Size; v++)
              {
                  if (v != s && rGraph[u, v] > 0)
                  {
                      if (parent[v] == null)
```

```
                {
                    parent[v] = new MilestoneHist { pointNum = u, totalCost = parent
                        [u].totalCost + costGraph[u, v] };
                    q.Enqueue(v);
                }
                else
                {
                    int newCost = parent[u].totalCost + costGraph[u, v];
                    int oldCost = parent[v].totalCost;
                    if (newCost < oldCost)
                    {
                        parent[v] = new MilestoneHist { pointNum = u, totalCost =
                            newCost };
                        q.Enqueue(v);
                    }
                }
            }
        }
    }
    return parent;
}
```