5.Міські транспортно-пересадочні вузли і логістика / Є.О. Рейцен, К.О. Томкевич // Містобудування та територіальне планування. – 2004. – Вип.17. – С.276-291.

6.Рекомендации по проектированию общественно-транспортных центров (узлов) в крупных городах / ЦНИИП градостроительства. – М.: Госстрой России, 1997. – 195 с.

Отримано 21.02.2012

УДК 656.02.2

О.Ф.КУЗЬКІН, канд. техн. наук
Запорізький національний технічний університет

ПОШУК ШЛЯХІВ У МАРШРУТНИХ МЕРЕЖАХ МІСТ МЕТОДОМ ВІДГАЛУЖЕНЬ І МЕЖ

Розроблено алгоритм розв'язку задачі пошуку найкоротших за часом шляхів у міських маршрутних мережах транспорту загального користування з урахуванням тривалості пересадок методом відгалужень і меж.

Предложен алгоритм решения задачи поиска кратчайших по времени путей в городских маршрутных сетях транспорта общего пользования с учетом времени пересадок методом ветвей и границ.

The shortest path search algorithm in the public transport network is worked out taking into account time of transfers by a branch-and-bound method.

Ключові слова: міська маршрутна мережа, найкоротший шлях, пересадка, метод відгалужень і меж.

Задача пошуку найкоротших шляхів виникає при проектуванні і удосконаленні маршрутних мереж транспорту загального користування у містах, розподілі пасажирських кореспонденцій за шляхами прямування, визначенні завантаження ділянок маршрутних мереж і пасажиропотоків. Слід зауважити, що ця задача є значно складнішою, ніж задача пошуку найкоротшого шляху на транспортній мережі, оскільки при її розв'язку необхідно враховувати маршрутні обмеження і специфічні особливості, які полягають у тому, що:

1) в процесі здійснення поїздки між двома заданими зупинками у деяких випадках виникає необхідність зміни маршруту та (або) виду транспорту (тобто, здійснення пересадки);

2) навіть якщо між парою зупинок існує безпересадочний шлях, він може бути не найкоротшим за критерієм тривалості чи відстані;

3) тривалість пересадки у загальному випадку складається з тривалості пішого переходу пасажира між зупинками змінюваних маршрутів та тривалості очікування транспорту. Перша складова залежить від просторового розміщення зупинок (зупиночних майданчиків, станцій) на вулично-дорожній мережі, а друга – від інтервалу руху на зупинці маршруту, на який виконується пересадка.

4) тривалість руху між зупинками на маршрутній мережі залежить від технічної швидкості руху маршрутних транспортних засобів, яка, у свою чергу, залежить від виду міського пасажирського транспорту, типів використовуваного рухомого складу та режимів руху на маршрутах.

Задача оптимальної маршрутизації у міських транспортних мережах відноситься до класу NP-повних і може бути розв'язана тільки шляхом повного перебирання всіх можливих варіантів [1]. Як показано у [2], кількість можливих варіантів маршрутних схем дорівнює $2^{n(n-1)}-1$, де n – кількість транспортних районів у місті або існуючих зупинок громадського транспорту. Це значення дуже швидко зростає зі збільшенням n, і вже при n=10 складає приблизно $1{,}24\times10^{27}$ варіантів, повне перебирання яких є неможливим у розумний час навіть з використанням найсучаснішої обчислювальної техніки. При будь-якому додаванні, видаленні, зміні траси, режимів руху бодай одного маршруту, всі найкоротші шляхи потребують повного перерахунку. Таким чином, ефективність отримання непоганого наближеного розв'язку шляхом спрямованого перебирання обмеженої кількості варіантів маршрутних схем напряму залежить від обчислювальної ефективності алгоритму пошуку найкоротших шляхів на маршрутній мережі.

Аналіз публікацій [2-4] показав, що авторами не приділено достатньої уваги даній задачі. Так, у [2] наведений приклад розробки раціональної маршрутної схеми на доволі простому прикладі та зазначено, що пошук найкоротших шляхів з врахуванням тривалості пересадок розраховується методом потенціалів по аналогії з пошуком найкоротших шляхів без врахування тривалості пересадок. Однак, у такому випадку цей метод зводиться до перебирання всіх можливих варіантів прямування пасажира, здійснити який можна лише для транспортних мереж з малою кількістю транспортних районів. Аналогічний підхід пропонується і в роботі [3], без конкретної реалізації використовуваного алгоритму. В роботі [4] для побудови маршрутної схеми відшукуються шляхи з однією, двома та трьома пересадками, метод пошуку таких шляхів не наведено. Загальний алгоритм пошуку найкоротших шляхів з врахуванням тривалості пересадок і всіх особливостей задачі, наведених вище, подано у роботі [5]. Алгоритм побудований на принципі розширення множини вершин графа маршрутної мережі і додавання у нього додаткових ребер, які відповідають пересадкам. Пошук шляхів на розширеному графі здійснюється найефективнішим на сьогодні алгоритмом Дейкстри, який має поліноміальну обчислювальну складність у найкращому випадку $O(n\log n+m)$, де n – кількість вершин графа, а m – кількість його ребер. Такий підхід до пошуку шляхів є універсальним, однак у

реальних задачах розширений граф маршрутної мережі матиме значно більшу кількість вершин і ребер, ніж первинний граф транспортної мережі. Якщо позначити як R наявну кількість маршрутів, а $\overline{s}$ – середню довжину маршруту, виражену у кількості зупинок, то розширений граф маршрутної мережі матиме $n = R \times \overline{s}$ вершин. Це значення набагато перевищує кількість вершин транспортної мережі і швидко зростає зі збільшенням кількості маршрутів та їх середньої довжини. Таким чином, реалізація алгоритму Дейкстри для розширеного графа маршрутної мережі потребує значних обчислювальних потужностей і пам'яті ЕОМ.

Метою даної статті є розробка ефективного швидкодіючого алгоритму пошуку найкоротших за тривалістю шляхів на маршрутній мережі з врахуванням тривалості пересадок, побудованому на використанні методу відгалужень і меж.

Викладення і реалізацію алгоритму покажемо на прикладі. На рис.1 наведена схема маршрутної мережі, яка налічує n=8 вершин (центри транспортних районів) та $R = 6$ маршрутів (позначені напівжирним шрифтом). Кожний маршрут будемо представляти у вигляді упорядкованого списку вершин, через які він проходить, таким чином, маємо: маршрут 1 – (1, 2, 3); маршрут 2 – (2, 3, 4, 5); маршрут 3 – (1, 5); маршрут 4 – (1, 6, 7); маршрут 5 – (7, 8); маршрут 6 – (5, 8).

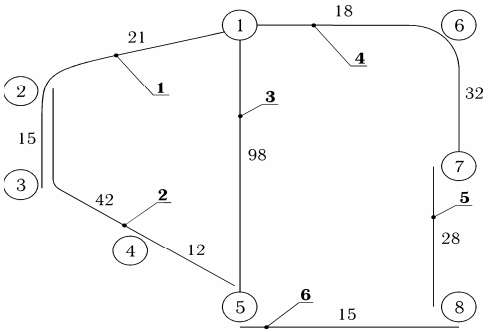

Рис.1 – Схема маршрутної мережі

У даному прикладі приймемо, що тривалість руху між вершинами є постійною і не залежить від маршруту, а тривалість пересадки у деякій i-й вершині залежить тільки від її номеру. Якщо це не так, алгоритм легко модифікувати для урахування всіх особливостей, зазначених вище. У даній статті ставиться за мету лише проілюструвати математико-алгоритмічний підхід до розв'язання поставленої задачі, а конкретні специфічні умови можуть бути легко введені на етапі його програмної

реалізації. Тривалість руху між вершинами маршрутної мережі (у хвилинах) наведена поруч з відповідними ланками маршрутів. Тривалості пересадок у кожній з вершин маршрутної мережі (у хвилинах) наведено в табл. 1.

Таблиця 1 – Тривалості пересадок у вершинах мережі

Номер вершини, k	1	2	3	4	5	6	7	8
Тривалість пересадки τ_k, хв.	4	7	9	1	3	5	1	2

Необхідно знайти найкоротший за тривалістю шлях з врахуванням тривалості пересадок між вершинами $p=1$ та $q=5$.

Зрозуміло, що між заданими вершинами у загальному випадку існує декілька шляхів різної тривалості, причому кількість цих шляхів конечна. Таким чином, можна перебрати всі можливі шляхи і вибрати з них найкоротший. Однак, повний перебір всіх можливих варіантів у реальних задачах може бути неефективним, або взагалі не здійсненним з огляду на їх велику кількість. Значно скоротити кількість варіантів перебору можна за допомогою методів пошуку з поверненням, до яких належить метод відгалужень і меж. Ідея методу відгалужень і меж при розв'язанні задач комбінаторної оптимізації полягає у послідовному розбитті множини всіх можливих розв'язків задачі на підмножини, які не перетинаються, з подальшим аналізом отриманих елементів розбиття та відкиданням тих з них, які заздалегідь не містять оптимального розв'язку.

Розв'язок поставленої задачі методом відгалужень і меж полягає у виконанні таких кроків (у викладенні алгоритму будемо використовувати поняття *довжина* та *відстань* у їх часовому виразі).

Крок 1. Розрахунок матриці найкоротших відстаней на маршрутній мережі без врахування необхідності пересадок.

Для розрахунку матриці найкоротших відстаней $D=\left(d_{ij}\right)_{i,j=\overline{1,n}}$ найбільш доцільним є використання динамічного алгоритму Флойда-Воршала [6] складності $O(n^3)$, який відшукує найкоротші відстані між всіма парами вершин мережі. При цьому умову необхідності виконання пересадок через наявність маршрутних обмежень *відкидаємо*.

Крок 2. Пошук *нижньої оцінки* довжини найкоротшого шляху $h(T)$.

У якості нижньої оцінки довжини найкоротшого шляху приймається довжина найкоротшого шляху між початковою та кінцевою верши-

нами без врахування пересадок. У нашому прикладі найкоротшим між вершинами 1 та 5 буде шлях $\tilde{L} = (1–2–3–4–5)$ довжиною $d_{15} = 90$ хв. Таким чином, $h(T)$ =90 хв.

Крок 3. Пошук *верхньої оцінки* довжини найкоротшого шляху $H(T)$.

Встановлення верхньої оцінки довжини найкоротшого шляху можна зробити декількома способами.

Спосіб 1. У найгіршому випадку можна припустити, що пересадки виконуються у кожній проміжній вершині шляху $\tilde{L} =$. Тоді верхня оцінка довжини найкоротшого шляху складе

$$H(T) = h(t) + \sum_{i \in \tilde{L}} \tau_i = h(t) + \tau_2 + \tau_3 + \tau_4 = 107 \text{ хв.} \qquad (1)$$

Цей найпростіший і найшвидший спосіб, однак, не є ефективним, оскільки у більшості випадків дає завищену верхню оцінку, в той час як ми зацікавлені у максимальному її зниженні.

Спосіб 2. Передбачає використання «жадібного» алгоритму, який полягає у наступному. Будемо рухатись від вершини 1 до вершини 5 таким чином, щоб по можливості здійснювати на шляху прямування якнайменшу кількість пересадок. Рушаючи з вершини 1, виберемо маршрут найбільшої довжини на шляху $\tilde{L}$ – це маршрут 1, яким без пересадок можна дістатися вершини 3. У цій вершині виконуємо пересадку на маршрут 2, яким досягаємо кінцевої вершини 5. Таким чином, на цьому шляху пересадку необхідно зробити тільки у вершині 3, звідки верхня оцінка довжини найкоротшого шляху дорівнюватиме

$$H(T) = h(t) + \tau_3 = 99 \text{ хв.}, \qquad (2)$$

що на 8 хв. менше, ніж оцінка, отримана за способом 1.

Спосіб 3. За цим способом відшукується найкоротший шлях на послідовності вершин, що складають шлях $\tilde{L}$. Для цього розглянемо цю послідовність і перетворимо її на спрямований ациклічний граф (рис.2).

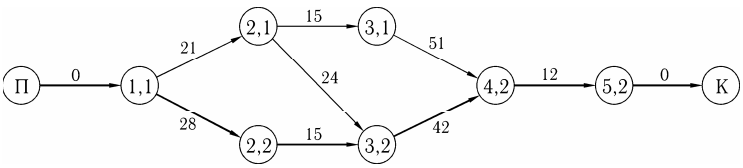

Рис. 2 – Спрямований ациклічний граф

Граф будується наступним чином. Початкова (П) і кінцева (К) вершини необхідні на той випадок, коли розпочати і завершити поїздку можна декількома маршрутами. Ці вершини з'єднуються спрямованими дугами нульової довжини з вершинами, які відповідають початку та закінченню шляху (у нашому випадку, вершинами 1 та 5). Кожну вершину ациклічного графа будемо позначати двома цифрами (k, r), де k – номер вершини графа маршрутної мережі, а r – номер маршруту, який проходить через неї. Кожній з вершин графа маршрутної мережі буде відповідати підмножина вершин ациклічного графа. Кількість елементів такої підмножини дорівнює кількості маршрутів, які проходять через відповідну вершину графа маршрутної мережі.

Довжина дуг ациклічного графа визначається за правилом:

$$t[(i,r_1);\ (j,r_2)] = \begin{cases} t_{ij}, & \text{якщо } r_1 = r_2 \wedge (i,j) \in \tilde{L}; \\ t_{ij} + \tau_j, & \text{якщо } r_1 \neq r_2 \wedge (i,j) \in \tilde{L}; \\ +\infty, & \text{якщо } (i,j) \notin \tilde{L}, \end{cases} \qquad (3)$$

де t_{ij} – відстань між суміжними вершинами i та j графа маршрутної мережі, хв.

Наприклад, дуга між вершинами (1,1) та (2,2) відповідає прямуванню з вершини i =1 до вершини j =2, при цьому у вершині 2 виконується пересадка з маршруту $r_1 = 1$ на маршрут $r_2 = 2$. ЇЇ довжина дорівнює відстані між вершинами 1 та 2 $t_{12} = 21$ хв. з додаванням тривалості пересадки у вершині 2 $\tau_2 = 7$ хв., тобто t = [(1,1); (2,2)] = 21 + 7 = 28 хв.

На побудованому ациклічному спрямованому графі відшукується найкоротший шлях з вершини (П) до вершини (К). Це можна зробити за лінійний час за допомогою загальновідомого алгоритму [6]. Застосування цього алгоритму для графа, наведеного на рис.2, дає найкоротший шлях (П) – (1,1) – (2,2) – (3,2) – (4,2) – (5,2) – (К) з однією пересадкою з маршруту 1 на маршрут 2 у вершині 2 (позначений жирними дугами) і довжиною 97 хв. Таким чином, верхня оцінка довжини найкоротшого шляху зменшилась ще на дві хвилини і дорівнює $H(T)$ = 97 хв.

Після виконання кроків 2 та 3 і знаходження оцінок можна стверджувати, що шуканий найкоротший шлях буде не коротшим ніж $h(t)$ = 90 хв. і не довшим, ніж $H(T)$ =97 хв. Зауважимо, що у випадку $h(t) = H(T)$ шлях $\tilde{L}$ *є оптимальним* і розв'язок на цьому припиняється.

Процес розв'язання задачі зручно представити у вигляді «дерева»,

наведеного на рис. 3, на гілках якого будемо розміщувати підмножини допустимих розв'язків, а поруч з вершинами дерева – вказувати *нижню* оцінку відповідної підмножини. Таким чином, на початку розв'язання вершина «дерева» – підмножина «всі розв'язки» має оцінку $h(t)$ =90 хв.

Крок 4. Розгалуження кореня дерева пошуку розв'язків.

Для розгалуження кореня дерева пошуку розв'язків переглядаємо всі маршрути, з використанням яких може розпочатися шлях пересування з початкової вершини. У нашому прикладі при русі з вершини 1 можна скористатися маршрутами 1, 3 та 4. Кожному з них відповідатиме гілка, яка виходить з кореня дерева пошуку розвязків. Нижня оцінка кожної з цих вершин приймається рівною нижній оцінці довжини найкоротшого шляху $h(t)$, тобто $b(1,1) = b(1,3) = b(1,4) = h(T) = 90$ хв. (рис.3). Як і раніше, перша цифра позначення вершини дерева є її номером на маршрутній мережі, а друга цифра – позначає використовуваний маршрут.

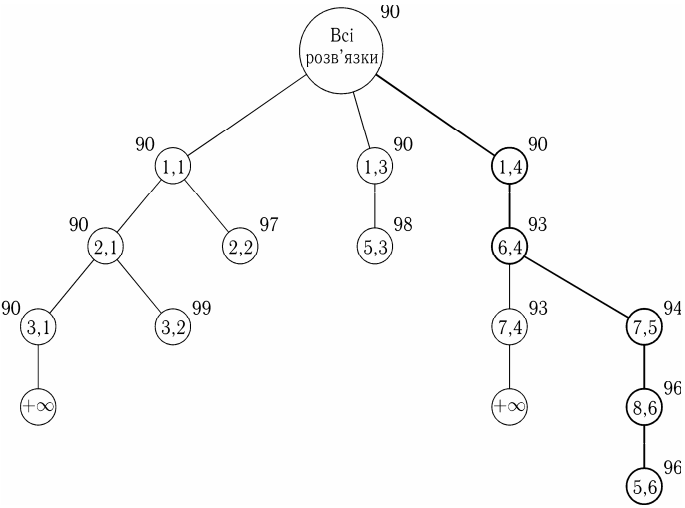

Рис.3 – Дерево пошуку розв'язків задачі

Крок 5. Вибір вершини-кандидата для розгалуження.

В якості вершини-кандидата для розгалуження обирається нерозгалужена вершина (k^*, r^*) з *мінімальним* значенням оцінки. При цьому з розгляду наперед виключаються всі вершини, оцінка яких перевищує верхню оцінку довжини найкоротшого шляху $H(T)$. У нашому прикладі після виконання кроку 4 вершини (1,1), (1,3) та (1,4) мають однакову

мінімальну оцінку, яка дорівнює $b^* = b_{\min}$ = 90 хв. У такому випадку можна обрати будь-яку з них. Нехай це буде вершина (1,1).

Крок 6. Перевірка вершини-кандидата та її оцінки.

Якщо $k^* = q$, то *оптимальний розв'язок знайдено* і довжина найкоротшого шляху дорівнює b^*. Інакше слід перейти до виконання кроку 7. Наразі, у нашому прикладі (k^*=1)$\neq (q = 5)$.

Крок 7. Розгалуження вершини-кандидата.

Для розгалуження вибраної на кроці 5 вершини-кандидата (k^*, r^*) розглядаємо наступну вершину маршрутної мережі k', суміжну вершині k^* на маршруті r^* (при цьому варіант повернення до вже відвіданої попередньої вершини на цьому маршруті виключається). Для вершини-кандидата (1,1) це буде вершина k'=2. Далі переглядаємо всі маршрути, яким належить вершина 2. Це маршрути $r_2^{(1)}$=1 та $r_2^{(2)}$=2. Таким чином, з вершини-кандидата (1,1) маємо два відгалуження – у вершини (2,1) та (2,2), як показано на рис.3.

У випадку, якщо вершина k^* є *кінцевою* на маршруті r^*, то такій вершині буде відповідати відгалуження у вигляді гілки у вершину дерева пошуку розвязків з оцінкою $+\infty$, що буде позначати неможливість подальшого розгалуження з цієї вершини-кандидата.

Крок 8. Розрахунок оцінок відгалужених вершин.

Оцінка кожної з вершин, відгалужених на кроці 7, визначається за формулою

$$b' = b^* + t_{k^*k'} + d_{k'q} - d_{k^*q} + f(k', r^*, r'), \tag{4}$$

де b^* – оцінка попередньої вершини дерева пошуку розв'язків, хв.; $t_{k^*k'}$ – відстань між вершинами k^* та k' за маршрутом r^*, хв; $d_{k'q}$ – найкоротша відстань між відгалуженою вершиною k' та кінцевою вершиною q на маршрутній мережі без врахування пересадок (див. крок 1), хв.; d_{k^*q} – найкоротша відстань між попередньою вершиною дерева пошуку розв'язків та кінцевою вершиною q на маршрутній мережі без врахування пересадок, хв.; $f(k', r^*, r')$ – функція, що враховує тривалість пересадки

$$f(k', r^*, r') = \begin{cases} 0, & \text{якщо } r^* = r'; \\ \tau_{k'}, & \text{якщо } r^* \neq r'. \end{cases} \tag{5}$$

Покажемо, наприклад, обчислення оцінки для відгалуженої вершини (2,2). Оскільки попередньою вершиною дерева розв'язків є вершина (1,1), це відгалуження відповідає пересуванню з вершини k^*=1 до вершини k'=2 з використанням маршруту r^*=1 з пересадкою у вершині 2 на маршрут r'=2. Оцінка вершини (2,2) у відповідності до формули (4)

$$b(2,2) = b(1,1) + t_{12} + d_{25} - d_{15} + f(2,1,2) = 90 + 21 + 69 - 90 + 7 = 97 \text{ хв.}$$

Тут $f(2,1,2) = \tau_2 = 7$ хв.

Після визначення оцінок всіх відгалужених на кроці 7 вершин виконується повернення до кроку 5.

Результатом виконання алгоритму для розглянутого прикладу є отримання оптимального розв'язку у вершині (5,6) з оцінкою 96 хв. Відповідна гілка дерева пошуку розв'язків позначена на рис. 3 жирними лініями. Рухаючись від вершини (5,6) у напрямі кореня дерева легко знайти оптимальний шлях. Починаючи з кореня дерева, він відповідає послідовності вершин (1,4) – (6,4) – (7,5) – (8,6) – (5,6). Таким чином, найкоротший шлях між вершинами 1 та 5 заданої у прикладі маршрутної мережі проходить через вершини 1 – 6 – 7 – 8 – 5 та дорівнює 96 хв. При цьому необхідно виконати пересадки у вершинах 7 (з маршруту 4 на маршрут 5) та 8 (з маршруту 5 на маршрут 6).

Викладемо деякі міркування щодо обчислювальної ефективності та програмної реалізації представленого алгоритму.

1. Спочатку призначений для пошуку найкоротшого шляху між вибраною парою вершин, алгоритм насправді дає більше інформації в процесі пошуку цього шляху. Так, у розглянутому прикладі, відшукавши найкоротший шлях між вершинами 1 та 5, ми фактично знайшли ще й найкоротші шляхи від вершини 1 до всіх вершин маршрутної мережі, окрім вершини 4. Це значно скорочує обсяги розрахунків за алгоритмом у разі пошуку найкоротших шляхів від вибраної вершини до всіх інших чи між всіма парами вершин маршрутної мережі.

2. Реалізація кроку 5 потребує зберігання і сортування у порядку зростання оцінок всіх нерозгалужених вершин в процесі розв'язку. Найкращим при реалізації на ЕОМ з обчислювальної точки зору буде зберігання нерозгалужених вершин та їх оцінок у *бінарній купі*, яка виконує операції додавання та витягнення вершини за логарифмічний час $O(\log n)$. При цьому кожна розгалужена вершина до купи не повертається. Такий підхід дозволяє значно скоротити потужність множини нерозгалужених вершин, якщо на кроці 7 перед розгалуженням попередньо обчислити попередні оцінки кожної з вершин, суміжних вершині-кандидату за формулою

$$b' = b^* + t_{k^*k'} + d_{k'q} - d_{k^*q}, \qquad (6)$$

і у випадку $b' > H(T)$ відгалуження у цю вершину не виконувати.

Аналогічно, на кроці 8, якщо для деякої відгалуженої вершини $b' > H(T)$, цю вершину не додаємо до купи нерозгалужених.

3. Алгоритм легко модифікується для врахування практично всіх додаткових умов, зокрема, зазначених на початку статті. Наприклад, у реальних маршрутних системах на даний час серед основних критеріїв вибору пасажиром шляху прямування є сума грошових коштів, витрачених на поїздку. У маршрутних системах більшості міст України кожна пересадка призводить до збільшення цієї суми. Зауважимо, що у розглянутому прикладі найкоротшим за часом є шлях з двома пересадками, незважаючи на те, що два інших варіанти (1 – 2 – 3 – 4 – 5) та (1 – 5) потребують, відповідно, однієї пересадки та жодної. Таким чином, можна поставити задачу пошуку найкоротшого шляху між заданими вершинами, який включає не більше ніж $m = 0, 1, \ldots$ пересадок. Пошук безпересадочних маршрутів ($m = 0$) є тривіальним. Для реалізації випадку $m > 0$ необхідно для кожної вершини дерева пошуку розв'язків додатково зберігати величину Ω – кількість пересадок, виконаних на шляху до неї і не включати вершину до списку нерозгалужених у випадку $\Omega > m$.

Також, маючи дерево пошуку рішень, нескладно відшукати шляхи, що відрізняються від оптимального на задане значення при рішенні задачі розподілу пасажирських кореспонденцій по альтернативних шляхах прямування.

4. Реалізація алгоритму у інтерпретаторі Python 2.7 в середовищі розробки Eclipse на ЕОМ з процесором AMD Athlon™ 4200+ 2,2 ГГц та з 2 ГБ оперативної пам'яті для тестової маршрутної мережі з $n = 110$ вершинами та $R = 1008$ маршрутами показала, що на пошук найкоротшого шляху між вибраною парою вершин витрачається в середньому 1,32 мс машинного часу, що свідчить про придатність пропонованого алгоритму для рішення практичних задач.

1.Magnanti T.L. Network design and transpotation planning: models and algorithms / T. L. Magnanti, R. T. Wong // Transportation Science. – 1984. – №18(1). – P.1-55.

2.Геронимус Б.Л. Экономико-математические методы в планировании на автомобильном транспорте / Б.Л. Геронимус, Л.В. Царфин. – М.: Транспорт, 1988. – 192 с.

3.Хрущев М.В. Исследование методов маршрутизации автобусного транспорта в городах: Дисс. … д-ра экон. наук / М. В. Хрущев. – М., 2000. – 206 с.

4.Ігнатенко О.С. Організація автобусних перевезень у містах / О.С. Ігнатенко, В.С. Марунич. – К.: УТУ, 1998. – 196 с.

5.Кузькін О.Ф. Пошук найкоротших шляхів у міських маршрутних мережах / О. Ф. Кузькін // Східноєвропейський журнал передових технологій. – 2011. – №6/4(54). – С. 8-12.

6.Кристофидес Н. Теория графов. Алгоритмический подход / Н. Кристофидес. – М.: Мир, 1975. – 432 с.

Отримано 05.03.2012

УДК 656.11.021.2

Д.В.ЗАСЯДЬКО
Харківський національний автомобільно-дорожній університет

ЗМЕНШЕННЯ ТРАНСПОРТНОГО НАВАНТАЖЕННЯ НА ЦЕНТР МІСТА ШЛЯХОМ СТВОРЕННЯ СИСТЕМИ УМОВНО-КІЛЬЦЕВИХ ЗВ'ЯЗКІВ

Розглядається проблема транспортного навантаження центральних частин великих міст з радіальним плануванням, вказано на можливість зменшення транспортного навантаження на центральні частини шляхом відведення транзитних для центральної частини транспортних потоків, для чого запропоновано створення системи умовно-кільцевих транспортних зв'язків та запропоновано методику розрахунку потрібної пропускної спроможності окремих ділянок цієї системи.

Рассматривается проблема транспортной нагрузки центральных частей больших городов с радиальной планировочной структурой, указано на возможность уменьшения транспортной нагрузки на центральные части путём отведения транзитных для центральной части транспортных потоков, для чего предложено создание системы условно-кольцевых транспортных связей и предложено методику расчёта потребной пропускной способности отдельных участков этой системы.

The problem of traffic load of the central parts of big cities with a radial structure of the planning is considered, pointed to the possibility of reducing the traffic load on the central part, using the diversion of transit traffic through the central part of the city, using a proposed system of quasi-circular transport links and the proposed method of calculation of the required link capacity.

Ключові слова: транспортні потоки, транспортне навантаження, система умовно-кільцевих зв'язків, пропускна спроможність.

Однією з проблем великих міст, пов'язаною з функціонуванням їх транспортних систем є скупченість транспортних засобів у центральній частині міста. Причина криється у історичних особливостях розвитку міст. Рівень автомобілізації протягом років постійно зростає, а протяжність, розгалуженість та пропускна спроможність вулиць зростає не такими великими темпами, як того вимагає попит на пересування автомобілями. Внаслідок чого в центральних частинах міст виникає ускладненість та транспортні затори.

Проблема виникнення транспортних заторів зокрема і у центральних частинах міст розглядалася у багатьох роботах [1-4]. Проведений автором відеомоніторинг транспортних потоків на межах центральної