Журнал обчислювальної та прикладної математики **2015, №1(118)** Journal of Computational & Applied Mathematics

УДК 519.85

РОЗВ'ЯЗАННЯ ЗАЛЕЖНИХ ВІД ЧАСУ ЗАДАЧ ПОШУКУ НАЙКОРОТОГО ШЛЯХУ

А. І. Павленко

Резюме. Розглядаються алгоритми для планування маршрутів в транспортних мережах, які розроблені останнім часом. Подано огляд підходів до моделювання розкладу у динамічних графах. Описані специфічні підходи розв'язання залежної від часу задачі пошуку найкоротшого шляху з врахуванням розкладу громадського транспорту.

Вступ

Пошук найкоротших шляхів у динамічних графах є практичною задачею із багатьох сфер людської діяльності: маршрутизація дорожніх мереж, планування подорожей, розкладів залізничних сполучень тощо. Ключовим компонентом сучасних систем маршрутизації (наприклад Google Maps, SkyScanner, GPS навігатори тощо) є алгоритми, які розраховують маршрути подорожей в залежності від пунктів відправки і прибуття, часу подорожі, користувацьких умов тощо. Такі алгоритми повинні бути швидкими і забезпечувати оптимальні чи близькі до них розв'язки для будь-якого запиту користувача.

Транспортні мережі зазвичай є зазвичай *динамічними* — з непередбачуваними затримками, заторами тощо. Цей факт значно ускладнює проблеми пошуку найкоротших шляхів, тому багато сучасних методів мають два етапи — процедуру *попередньої обробки* вхідних статичних даних для пришвидшення роботи алгоритму (наприклад, аналіз топології графу, пошук ключових транзитних вузлів, пропускна спроможність автошляхів тощо) і власне процедуру запиту — пошук оптимального шляху відповідно до параметрів моделі в конкретний час з врахуванням умов користувача (наприклад, з врахуванням інформації про поточну швидкість руху на автошляху, затримки рейсу тощо). Алгоритми, які застосовуються у реальних системах масового обслуговування, повинні працювати в режимі реального часу з прийнятною ресурсоємністю в масштабах великих регіонів.

В реальних транспортних мережах найкращі шляхи зазвичай залежать від часу відправлення. Наприклад, деякі автошляхи постійно перевантажені у години пік, а потяги і автобуси відправляються з різною частотою протягом дня. Якщо необхідно прибути у деякий пункт якомога раніше з врахуванням часу відправлення (або, симетрична задача — якомога пізніше відправитись щоб прибути до деякого зазначеного часу), то говорять про залежну від часу *задачу пошуку найкоротшого шляху* (Time-Dependent

Shortest Path Problem), для якої характерне призначення функцій часу в якості функцій вартості для дуг.

В загальному випадку транспортна мережа моделюється як зважений орієнтований граф (мультиграф), ваги дуг якого відображають параметри (час подорожі, відстань, ціна тощо), які необхідно оптимізувати. Дане дослідження присвячене підходам розв'язання залежної від часу задачі пошуку найкоротшого шляху між двома пунктами з врахуванням розкладу. Така постановка притаманна задачі пошуку оптимального маршруту між заданими пунктами мережами громадського транспорту. Типові запити у таких задачах — пошук найкращого часу відправлення при мінімізації загального часу подорожі.

1. Формальна модель залежної від часу задачі пошуку найкоротшого шляху

Дано орієнтований зважений граф $G = (V, A)$, де $V = \{v_1, ..., v_n\}$ — множина n вершин; A – множина дуг, початкова $s \in V$ і кінцева вершини $d \in V$, t_0 — час відправлення з вершини s. Кожній дузі $a = (i, j)$ відповідає вартість $c_{ij}(t)$, де t — час відправлення з пункту i. Вартість може вимірюватись у матеріальних витратах, тривалості, відстані або будь-якій іншій мірі ресурсних витрат. Довжиною шляху є сума вартостей всіх його дуг. Найменший шлях між двома пунктами позначається $g(i, j)$. Метою задачі пошуку найкоротшого шляху є знаходження найкоротшого шляху (або найдешевшого, найшвидшого в залежності від вимірюваних ресурсів) між однією початковою вершиною і однією цільовою (задача *one-to-one*), між однією початковою і усіма іншими вершинами (задача *one-to-all*), між усіма вершинами і однією цільовою (*all-to-one*). Дане дослідження концентрується на задачі *one-to-one*.

Загалом залежна від часу задача пошуку найкоротших шляхів не є новою. У 1966 році Кук і Хелсі [1] сформулювали її і запропонували методи розв'язання на основі принципів оптимальності Белмана [2]. Для спрощення питання існування і для зручності обчислення було запропоновано ввести дискретну часову множину $S = \{t_0, t_0 + 1, t_0 + 2, \ldots, t_0 + T\}$, причому $c_{ij}(t)$ може приймати будь-яке додатне ціле значення для $t \in S = \{t_0, t_0 + 1, t_0 + 2, \ldots, t_0 + T\}$, $i \neq j$. Якщо дуги (i, j) не існує, то $c_{ij}(t) = \infty$. Нехай $E_i(t)$ — множина усіх шляхів, у яких виконується вихід з пункту i в час $t \in S$ і досягається d за скінченну вартість (час) після скінченної кількості кроків. Оскільки вартість для кожного шляху $E_i(t)$ додатна, то серед них можна знайти шлях з мінімальною вартістю. Для $t \in S$ автори визначили $f_i(t)$ — мінімальна вартість для шляху з $E_i(t)$, $i = 1, 2, \ldots, d-1$, $f_d(t) = 0$, де мінімальна вартість $f_i(t)$ досягається принаймні одним з шляхів з i в d за скінченну кількість кроків. Використовуючи принцип оптимальності [3] вони запропонували подати залежну від часу задачу пошуку шляху $f_i(t) = \min\limits_{j \neq i}(c_{ij}(t) + f_j(t + c_{ij}(t))), \quad i = 1, 2, \ldots, d-1$, $f_d(t) = 0$.

Для розв'язання задачі Кук і Хелсі запропонували модифікацію схеми ітерацій Белмана для знаходження найкоротшого шляху між двома вершинами за скінчену кількість ітерацій. Була визначена максимальна межа T, розрахувавши прямий перехід з i в d — $c_{id}(t_0)$. Якщо прямого переходу немає, то шукається шлях з проміжними вершинами для визначення верхньої оцінки. За допомогою такої оцінки усі переходи, більші за верхню межу, не приймаються до розгляду.

Дрейфус [5] припустив, що загальним алгоритмом Дейкстри можна розв'язувати динамічну задачу, поставлену Куком і Хелсі, так само ефективно, як і загальні статичні задачі. У [1] було доведено, що це можливо лише якщо мережа володіє властивостями FIFO.

Властивості FIFO і узгодженості вартості. Динамічний граф є графом FIFO, якщо кожна дуга графу задовольняє властивості FIFO. Дузі (i, j) притаманна властивість FIFO (*first in first out*), якщо $t + c_{ij}(t) \leq (t + k) + c_{ij}(t + k)$, $\forall t, k \geq 1$, t, k — моменти часу. Практично це означає, що якщо транспортний засіб відправляється з деякого пункту пізніше, він не може прибути до пункту призначення раніше, ніж транспортний засіб, який вирушив з початкового пункту вчасно. Властивість узгодження вартості аналогічна FIFO для іншого виду ресурсів, тобто дуга (i, j) вважається узгодженою за вартістю, якщо при виході з пункту i вартість шляху менша, ніж при виході з пункту j.

2. Планування оптимального маршруту громадським транспортом.

Постановка задачі. Для задачі планування оптимального маршруту в якості вхідних даних зазвичай є *розклад руху* транспорту. Він подається множиною *зупинок* (автобусні зупинки, залізничні платформи, аеропорти тощо), множиною *маршрутів слідування* (автобусні, трамвайні, залізничні маршрути тощо) [6]. Маршрут слідування — послідовність зупинок, які повинен відвідати транспортний засіб в конкретний час, — являє собою множину елементарних сполучень. Елементарне *сполучення* задається двома зупинками, між якими курсує транспортний засіб, а також час відправлення і прибуття.

Ключовою відмінністю мереж громадського транспорту від автомобільних є невід'ємна наявність залежності від часу, адже деякі сегменти мережі стають досяжними тільки в зазначені моменти часу. Тому початковою задачею при розв'язанні залежних від часу задач пошуку найкоротших шляхів є подання розкладу у графі. Існують два основні підходи для подання часу у мережах — time-expanded (розширений за часом) і time-dependent (залежний від часу).

Проста розширена за часом модель подання розкладу у графі (time-expanded model). Підхід ґрунтується на тому факті, що розклад складається з *залежних від часу подій* (час відправлення транспорту), які відбуваються в дискретні моменти часу. Ідея полягає у побудові *просторово-часового графу* (space-time graph) або *графу подій* (event graph), який «розгортає» граф у часі. Грубо кажучи, для кожної події розкладу

створюється вершина, а для подання елементарних сполучень або очікування на станціях використовуються дуги. Таким чином, кожному сполученню $conn \in CONN$ ($CONN$ — множина сполучень) відповідає дуга і дві вершини — пункти відправлення $u_{dep}(conn)$ і прибуття $u_{arr}(conn)$. В залежності від критерію оптимізації, дугам присвоюються деякі фіксовані ваги. Вершинам завжди неявно відповідають деякі значення часу $\tau(u)$. В нашому випадку час вершини відправлення задається часом відправлення відповідного сполучення $\tau_{dep}(conn)$, а час вершини прибуття задається часом $\tau_{arr}(conn)$. Кожній вершині відповідає зупинка $p(u)$, причому кожній зупинці можуть відповідати кілька вершин.

Для кожного сполучення $conn \in CONN$ в модель додається дуга $(u_{dep}(conn), u_{arr}(conn))$ між вершинами відправлення і прибуття цього сполучення. Для здійснення пересадок на зупинках до моделі додаються (незалежно для кожної зупинки $p \in STOPS$, де $STOPS$ — множина зупинок) дуги, призначені для неявного забезпечення часу для пересадки. В тому випадку, коли розклад є періодичним, для кожної зупинки додається дуга (u, v) з найпізнішої до найранішої вершини зупинки, дозволяючи переходи з одного періоду розкладу до іншого. Звичайно, в такому випадку граф стає циклічним.

Реалістична розширена за часом модель подання розкладу у графі (time-expanded model). В реалістичній моделі для моделювання пересадок вводяться транзитні вузли $u_{tr}(conn)$ причому $p(u_{tr}(conn)) = p_{dep}(conn)$. Для кожної події відправлення створюється одна *пересадочна вершина*, з'єднана з відповідною вершиною відправлення $u_{dep}(conn)$. Вага дуги, що відображує сполучення між пересадочною вершиною і вершиною відправлення, рівна нулю. Для забезпечення мінімального часу пересадки на деякій станції, між кожною вершиною прибуття u і вершиною v (яка також відповідає зупинці p) з відповідним найменшим часом також додається дуга, причому $\tau(u) + \tau_{transfer}(p) \leq \tau(v)$, де $\tau_{transfer}(p)$ — час пересадки.

Для забезпечення можливості залишатися в тому ж потязі на проміжних зупинках створюється додаткова дуга, яка з'єднує вершину прибуття і відповідну вершину відправлення, які відповідають маршруту одного потягу.

Основною перевагою розширеного за часом графу є його незалежність за часом, тобто дуги мають фіксовану вагу. Ця перевага робить можливим застосування класичних методів пошуку найкоротших шляхів.

Проста залежна від часу модель подання розкладу у графі (time-dependent model).

На відміну від розширеного за часом графу, залежні від часу графи мають значно менші розміри (порядку кількості зупинок і маршрутів розкладу). Даний підхід використовує спеціальні залежні від часу функції тривалості подорожі (time-dependent travel time functions) в якості ваг дуг.

Ключова ідея даного підходу — комбінація кількох елементарних сполучень в єдину дугу за допомогою *залежних від часу функцій тривалості подорожі.* Введемо простір функцій F, який складається з функцій часу подорожей у вигляді: $f : \Pi \to Z_{\geq 0}$. Кожна функція $f \in F$ співставляє

(переводить) *час відправлення* у *час подорожі* (або вартість). Час відправлення обирається з інтервалу Π — період операції у розкладі, час подорожі може приймати довільні невід'ємні цілі числа (з огляду на прибуття після півночі).

Розглянемо просту залежну від часу модель мережі. Дано розклад і відповідна множина елементарних сполучень $CONN$. Модель будує направлений граф $G = (V, A)$ створюючи вершину u_p, яка відповідає зупинці $p \in STOPS$. До множини A додаються дуги (p_1, p_2) тільки в тому випадку, якщо існує елементарне сполучення з p_1 в p_2, тобто $p_{dep}(conn) = p_1$ і $p_{arr}(conn) = p_2$. Вартості дуг визначаються в термінах функцій часу подорожі, тобто $c : A \to F$, де F — простір функцій. Отже, кожна дуга $(p_1, p_2) \in A$ містить саме ті точки сполучення, які відповідають елементарним сполученням з p_1 в p_2.

Реалістична залежна від часу модель подання розкладу у графі. Модель ґрунтується на тому судженні, що пересадки на одному і тому ж рейсі не бувають оптимальними. Таким чином, модель групує елементарні сполучення по рейсу. Нехай $\Re_p$ — множина маршрутів, що проходять через зупинку $p \in STOPS$. В моделі створюються вершини зупинок $p \in V$ (як і раніше), але додатково створюються вершини маршрутів r_p для кожної вершини маршруту з $\Re_p$.

Для кожного маршруту (рейсу) $r \in \Re$ розкладу і двох послідовних задіяних по маршруту зупинок $p_i, p_j \in STOPS$ створюється залежна від часу дуга маршруту $(r_{p_i}, r_{p_j}) \in A$. Залежна від часу функція часу подорожі створеної дуги містить точку сполучення, причому для кожного її елементарного сполучення $conn \in CONN$ дійсні твердження $p_{dep}(conn)=p_i$, $p_{arr}(conn) = p_j$, $r(conn) = r$. Для здійснення пересадок додаються дуги пересадок, які поєднують вершини зупинок і усі відповідні вершини маршруту. Отже, додаються дуги (p, r_p) і (r_p, p) для кожної зупинки $r_p \in \Re_p$. Модель враховує мінімальний час пересадки встановлюючи $c(p, r_p) = \tau_{transfer}(p)$ для усіх дуг, що ведуть з вершини зупинки до вершинам маршруту, де $\tau_{transfer}(p)$ — мінімальний час пересадки на зупинці p. Відповідно, $c(r_p, p) = 0$для усіх дуг, що ведуть з вершин маршруту до вершини зупинки.

Багатокритеріальні задачі. Інколи необхідно розширити функцію вартості кількома критеріями. Наприклад, деякі типи транспорту не можуть використовувати деякі сегменти транспортної мережі. Таким чином, можна адаптувати фазу попередньої обробки таким чином, щоб відповідні дуги заборонялись на етапі запиту, або виконувати оновлення метрик для кожного типу транспорту.

Пошуковий запит може бути гнучкішим. Наприклад, турист може захотіти зробити подорож трохи довшою, але з більшою кількістю туристичних пунктів. В такому випадку мають справу з багатокритеріальною задачею. В такому випадку шляхи порівнюються з врахуванням усіх критеріїв, а метою є пошук множини Парето, тобто максимальну множину непорівнюваних шляхів. Розв'язок такої задачі було запропоновано у [7] алгоритмом багатокритеріальної установки міток (Multicriteria Label-Setting — MLS),

який розширює алгоритм Дейкстри. Для кожної вершини зберігається набір недомінуючих міток. Кожна мітка є кортежем, де кожний елемент (запис) позначає один критерій оптимізації. Пріоритетна черга формується за значенням міток а не вершин. На кожній ітерації обирається найменша мітка L і скануються інциденті дуги $a = (u, v)$ до вершини u, пов'язаної з L. Виконується додавання вартості a до L і потім об'єднання L в набір міток вершини v з заміною недомінуючих міток.

Ще одним підходом розв'язання багатокритеріальних задач пошуку найкоротших шляхів є оптимізація лінійної комбінації критеріїв.

3. Специфічні підходи розв'язання залежної від часу задачі пошуку найкоротшого шляху

Алгоритм Дейкстри. Цей алгоритм оперує двома структурами даних — черга пріоритетів вершин Q і мітки часу прибуття $\tau(u)$ для кожної вершини $u \in V$, які ініціалізуються нескінченністю. На першому етапі задається $\tau(s) = \tau$ і s додається до черги Q з міткою τ. Далі з черги пріоритетів обирається вершина u з мінімальною міткою часу прибуття. На наступному кроці алгоритм аналізує усі вихідні дуги $a = (u, v)$. Для кожної дуги створюється мітка орієнтовного часу прибуття $\tau_{tent}(v) = \tau(u) + c(a)$ в вершину v. Якщо $\tau_{tent}(v)$ покращує $\tau(v)$, тобто $\tau_{tent}(v) \leq \tau(v)$, тоді дуга a релаксується: $\tau(v)$ оновлюється значенням $\tau_{tent}(v)$, до черги Q додається вершина v з міткою $\tau_{tent}(v)$. Алгоритм завершується, коли черга пріоритетів порожня.

Час роботи алгоритму Дейкстри визначається структурою даних черги пріоритетів Q.

Для розв'язання задачі пошуку оптимального шляху з урахуванням розкладу громадського транспорту необхідно враховувати той факт, що цільові вершини s і d в розширеному за часом графі невідомі. Вхідними даними є початкова і цільова зупинки p_s і p_d. Вершина s визначається серед усіх вершин відправлення (для простої моделі) або вершин пересадки (для реалістичної моделі) початкової зупинки p_s за критерієм мінімальної мітки $\tau(s) \geq \tau$, де τ — найраніший можливий час відправлення. Час відправлення з вершини s ініціалізується значенням $\tau(s)$. Нажаль, вершина d залишається невідомою до етапу завершення алгоритму. Проте для розширеного за часом алгоритму Дейкстри (Time-Expanded Dijkstra) пошук можна вважати завершеним, якщо буде аналізуватись будь-яка вершина u зупинки p_d. Перша проаналізована вершина серед вершин зупинки p_d відповідає найранішому часу прибуття в p_d.

У випадку залежної від часу моделі графу (time-dependent model), зупинки p_s і p_dявно відповідають вершинам s і d. Орієнтовна мітка $\tau_{tent}(v)$ розраховується за допомогою залежної від часу функції f_a для моменту часу $\tau(u)$ — найранішого часу прибуття в u.

Двонаправлений пошук. Найпростіший підхід поліпшення часу виконання алгоритму Дейкстри в статичному випадку — застосування двонаправленого пошуку, тобто додатковий запуск (обернений) алгоритму Дейкстри з кінцевого пункту до початкового. Алгоритм завершує свою роботу тоді, коли деяка v вершина видаляється з черг обох запусків. На відміну від

статичних задач, двонаправлений пошук алгоритмом Дейкстри потребує значних модифікацій, оскільки час прибуття в кінцевий пункт є невідомим.

Алгоритм A*. Одним із суттєвих критеріїв оцінки ефективності алгоритмів на графах, які працюють в режимі реального часу, є критерій кількості розглянутих вершин до моменту знаходження розв'язку. Простір пошуку алгоритму Дейкстри можна візуалізувати як коло навколо вихідної вершини (Рис. 1). Метою цілеспрямленого алгоритму (goal-target) або пошуку A* є направлення пошуку в сторону цілі.

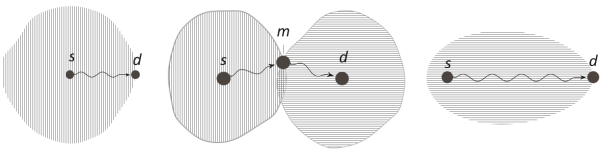

Рис 1. Схематичне зображення простору пошуку алгоритму Дейкстри (зліва), двонаправлений пошук (посередині), A* алгоритму (справа)

Алгоритм A* [8] наслідує ідею алгоритму Дейкстри, але в якості оцінки вершин використовує евристичну функцію $h(v)$ і обирає активну вершину з найменшим $g(v)+h(v)$, де $g(v)$ — найменший шлях від вершини v до цільової вершини. $h(v)$ можна вважати нижньою межею (*lower bound*) вартості (відстані, часу) від вершини v до цільової вершини d. Через це вершини, які розташовані ближче до цільової будуть розглянуті алгоритмом раніше. Вибір евристики $h(v)$ залежить від постановки задачі і не є частиною алгоритму A*, проте він визначає коректність і ефективність розробленого алгоритму A*. Очевидно, що алгоритм Дейкстри є частковим випадком A* з $h \equiv 0$.

У [9] алгоритм A* був застосований для розв'язання залежної від часу задачі пошуку найкоротшого шляху з врахуванням розкладу руху транспорту. Граф подається залежною від часу моделлю. Для кожної вершини u нижня межа до цільової зупинки p_d розраховується запуском оберненого пошуку (з p_d) використовуючи статичні нижні межі залежних від часу функцій в якості значень вартості дуг.

Алгоритм ALT. Для пришвидшення пошуку найкоротших шляхів у великих графах було запропоновано алгоритм ALT (*A*, landmarks, and triangle inequality*) [10]. Ідея алгоритму полягає у виконанні препроцесингу і обробці запитів. Схема алгоритму наведена на Рис. 2.

```
Препроцесинг
  Обрати декілька вершин як орієнтири
  Для кожного орієнтиру
    Розрахувати найкоротші шляхи до всіх вершин
    Зберегти шляхи
Обробка запиту
  Використати A*
  Якщо деяка дуга лежить між поточною вершиною і орієнтиром
    Надати дузі пріоритет при виборі активної
```

Рис 2. Схема алгоритму ALT

На етапі препроцесингу обирається невелика множина вершин в якості орієнтирів (landmarks) і зберігаються відстані між орієнтирами і всіма вершинами у графі. При запиті найкоротшого шляху з s в d застосовується алгоритм A* з використанням нерівності трикутника для розрахунку нижньої межі для відстаней (v, d) для будь-якої вершини v. Тобто, для будь-якого орієнтиру l_i статичної задачі вірно: $g(v, d) \geq g(v, l_i) - g(d, l_i)$. Якість нижніх меж (а отже і ефективність запиту) залежить від вибору орієнтирів на етапі препроцесингу.

Для пошуку найранішого часу прибуття однонаправлений алгоритм ALT був адаптований для розширеної за часом моделі [11] і залежної від часу моделі [12]. В обох випадках вибір орієнтирів і розрахунок відстаней до них виконується на допоміжному графі зупинок: вершини відповідають зупинкам у розкладі, дуга додаються між двома зупинками p_i, p_j, якщо існує пряме сполучення від p_i до p_j без проміжних зупинок. Вартості дуг відображують нижні межі часу подорожі між інцидентними зупинками.

Геометричні контейнери. Іншим цілеспрямованим (goal directed) методом є геометричні контейнери. На етапі попередньої обробки алгоритм розраховує для кожної дуги $a = (u, v) \in A$ мітку $L(a)$, яка являє собою множину вершин V_a, до яких найкоротший шлях з вершини u проходить через дугу a. Замість явного збереження V_a, $L(a)$ узагальнює цю множину використовуючи геометричну інформацію (координати) вершин з V_a. Під час запиту якщо цільова вершина d не належить $L(a)$, з пошуку можна безпечно вилучити розгляд a. У [13] множину V_a апроксимують сектором (з центром в u), який покриває усі вершини V_a. У [14] використовують еліптичні контейнери і випуклі оболонки (convex hull). Недоліком геометричних контейнерів є необхідність істотних обчислень для усіх пар вершин.

Геометричні контейнери були широко протестовані для розширених за часом моделей [13], [14] при розв'язанні задачі пошуку найранішого часу прибуття до цільового пункту.

Arc Flags і SHARC. Підхід Arc Flags (прапори дуг) [15] не використовує геометричні дані. Протягом попередньої обробки граф ділиться на K комірок, які грубо збалансовані (мають однакову кількість вершин) і мають невелику кількість граничних вершин. Кожна дуга зберігає булевий вектор розмірністю K(arc flags), де i-й елемент додатній, якщо дуга належить найкоротшому шляху до деякої вершини з комірки i. Алгоритм

пошуку відсікає дуги, які не містять додатній біт для комірки, яка містить d. Arc Flags для комірки i розраховуються побудовою дерев найкоротших шляхів з кожної граничної вершини (комірки i), виставляючи i-й елемент (флаг) для усіх дуг дерева. Рис. 3 показує схему роботи алгоритму [16]. Вершини і *arc flags* розфарбовані згідно їх комірок.

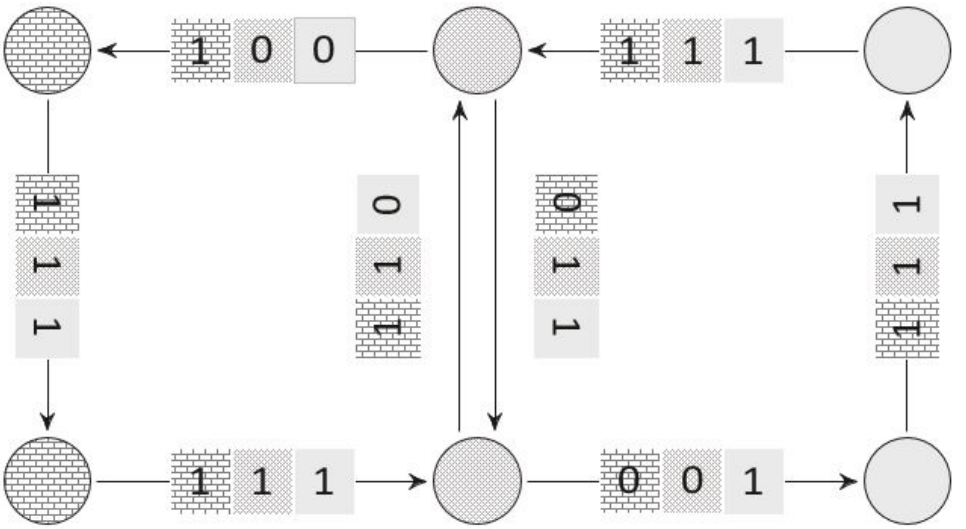

Рис 3. Arc Flags на малих графах

У [11] підхід Arc Flags був адаптований для розширеної за часом моделі. У графі зупинок (побудованому аналогічно підходу ALT) визначаються комірки. На другому кроці для кожної граничної зупинки p комірки C і кожної з її вершин прибуття виконується обернений пошук в розширеній за часом моделі. Було показано, що у мережах громадського транспорту між парами вершин може існувати багато шляхів однакової довжини, що робить вибір вирішальних правил особливо важливим. Arc Flags можна комбінувати з ALT і методом блокування вершин — технікою уникнення аналізу зайвих шляхів графу.

Алгоритм SHARC, який комбінує Arc Flags і мітки найкоротших шляхів, був протестований на залежній від часу моделі для запитів маршрутів найранішого часу прибуття [12]. Окрім того, комбінація Arc Flags і алгоритму багатокритеріальної установки міток (MLS) була використана для розрахунку повних множин Парето відповідно до критеріїв часу прибуття і кількості пересадок для реалістичної залежної від часу моделі [17].

У залежних від часу графах додатній біт виставляється у випадку, якщо дуга зустрічається принаймні один раз в день на найкоротшому шляху до відповідної комірки. З метою покращення продуктивності, використовують різні набори прапорів для різних часових інтервалі дня.

Підходи на основі сепараторів. Планарні графи містять ефективно обчислювальні сепаратори. Хоча дорожні мережі не є планарними, було виявлено, що вони також містять сепаратори. Цей факт використовується методами цього розділу [18].

Суть процедури вершинного сепаратора (Vertex Separators) – бажано невелика підмножина вершин$S \subset V$, при видаленні яких граф G розбивається на декілька (бажано збалансованих) комірок (компонентів). Цей сепаратор може бути використаний для побудови *overlay (багатошарового)* графу G'над S.

Багатошаровий граф даного графу $G = (V, E)$ на підмножині вершин $S \subseteq V$ — граф з множиною вершин S і дугами, які відповідають найкоротшим шляхам в G. Для кожної пари вершин $u, v \in S$, найкоротші $u - v$-шляхи мають однакову довжину у Gі G' [19]. Крім того, у G' бажано мати мінімальну кількість дуг.

У [20] будується граф з «суттєвих» вершин: для кожної пари $u, v \in S$ дуга (u, v) додається до багатошарового графу якщо найкоротший шлях $u-v$ не містить інших вершин. Такий підхід був розвинутий до багаторівневої ієрархії сепараторів $V \supset S_1 \supset S_2 \supset ... \supset S_k$ з kрівнями. Такий багатошаровий граф містить дуги між вершинами-сепараторами на кожному рівні, а також для кожної комірки i-го рівня — дуги між вершинами-сепараторами i-го рівня і вершинами-сепараторами $i - 1$-го рівня (Рис. 4).

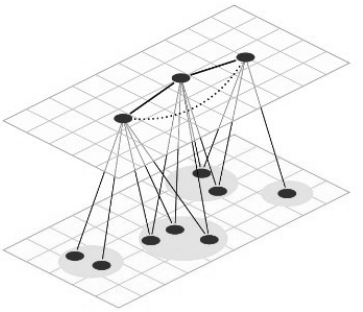

Рис 4. Багаторівневий overlay граф з двома рівнями. Крапками позначено вершинній сепаратори нижнього (помаранчевого) і верхнього (зеленого) рівнів.

Підхід активно досліджується і демонструє відносно добрі результати для пошуку наближених найкоротших шляхів на ненаправлених планарних версіях мереж доріг.

Використання багатошарових графів для пришвидшення запитів був запропонований в контексті планування маршрутів громадським транспортом. У [20] будуються багатошарові графи одного рівня між важливими вузловими станціями у розширеній за часом моделі. Багаторівневі багатошарові графи, які ґрунтуються на вершинних сепараторах, були розроблені в контексті розширених у часі [20].

Скорочення ієрархій (Contraction Hierarchies). Підхід полягає у використанні концепції міток найкоротших шляхів, запропонований у [21]. Мітка найкоротшого шляху (*або скорочуюча дуга*) — це дуга (u, v), яка не обов'язково існує у початковому графі, але визначає найкоротший шлях з u до v в графі G. Метою є розширення графу G мітками таким чином,

щоб запити на великі відстані використовували ці мітки в для пропуску «неважливих» вершин. Метод скорочення ієрархій реалізовує цю ідею повторним виконанням процедури скорочення вершин. Протягом етапу попередньої обробки метод евристично впорядковує вершини згідно їх важливості і потім скорочує в оберненому порядку. Для скорочення вершини v вона тимчасово вилучається з графу G, а між кожною парою її сусідніх вершин u і w додаються мітки найкоротшого шляху в тому випадку, якщо найкоротший шлях з u в w єдиний і містить v. Алгоритм запиту запускає двонаправлений пошук з s в d на скороченому графі G. Методи [22], [23] також використовують подібні ідеї.

Метод скорочення ієрархій був адаптований для реалістичної залежної від часу моделі для обчислення маршрутів найранішого прибуття. Для практичного використання просте застосування алгоритму для графу маршрутів спричиняє створення надто великої кількості міток найкоротшого шляху. Тому скорочення виконується тільки на щільному графі, в якому одній зупинці відповідає одна вершина.

Шаблони пересування. Шаблони пересування — спеціальний підхід пришвидшення роботи алгоритмів мереж громадського транспорту [24]. Він заснований на тому спостереженні, що у багатьох оптимальних подорожей є спільний шаблон пересування, тобто послідовність зупинок пересадок. Ці шаблони можна обчислювати на етапі попередньої обробки для усіх пар зупинок і часу відправлення. Під час запиту будується спеціальний *граф запиту*, який є об'єднанням шаблонів пересування для початкової і цільової зупинок. Дуги графу запиту відображують прямі сполучення між зупинками (без пересадок) і можуть бути швидко оцінені. На наступному кроці може бути застосований алгоритм Дейкстри або алгоритм багатокритеріальної установки міток (MLS) для графу запиту значно меншого розміру.

Якщо визначення шаблонів пересування для усіх пар зупинок на етапі попередньої обробки є надто ресурсоємним, тоді можна застосувати двоетапний підхід. На першому етапі обирається множина важливих транзитних вузлів. Між такими вузлами виконується пошук глобальних шаблонів пересування. Для нетранзитних вершин виконується пошук локальних шаблонів до транзитних вузлів. Нажаль дослідження показали неприйнятний час роботи алгоритму для графів континентального масштабу. Тому було запропоновано на етапі попередньої обробки розглядати шаблони пересування з обмеженою кількістю зупинок (двома). Таким чином етап попередньої обробки транзитної мережі Північної Америки зайняв більше 3000 годин, а запити виконуються в межах 10мс.

Підхід шаблонів пересування використовується в Google Transit [24], [25].

TRANSIT. Підхід маршрутизації транзитних вузлів був адаптований для планування подорожей громадським транспортом [26], [27] — було запропоновано алгоритм TRANSIT.

Підхід маршрутизації транзитних вузлів (TNR) використовує таблицю відстаней підмножини вершин. Протягом етапу попередньої обробки обирається невелика множина $T \subseteq V$ транзитних вузлів і розраховується відстань між вершинами між усіма вершинами попарно підмножини. Для всіх інших вершин $u \in V \backslash T$розраховується відповідна множина вершин $A(u) \subseteq T$, які називаються вершинами доступу (до вершини u). Транзитна вершина $v \in T$ вважається вершиною доступу вершини u, якщо існує найкоротший шлях P з u в графі G так, що вершина v — найперша вершина в P. Алгоритм запиту використовує таблицю відстаней для вибору такого шляху, який мінімізує відстань $s - A(s) - A(d) - d$. Зазначимо, що отриманий результат є некоректним, якщо найкоротший шлях не містить вершину з T . Отже, фільтр локальності спочатку визначає чи може запит бути локальним (не містить вершини з T) . Якщо це так, запускається альтернативний алгоритм пошуку найкоротшого шляху для обчислення правильної відстані. На Рис. 5 зображена схема TNR запиту. Червоні і сині крапки — вузли доступу вершин s і d. Стрілки вказують на відповідні рядки (стовпці) таблиці відстаней. Виділені записи таблиці відповідають вузлам доступу, які мінімізують відстань $s - d$. Для кращої ефективності зазвичай використовують кілька рівнів транзитних вузлів і вузлів доступу. Для ефективності алгоритму критичним є вибір множини транзитних вузлів. Природньо в якості транзитних вузлів обирати вершинні сепаратори і граничні вершини сепараторів-дуг.

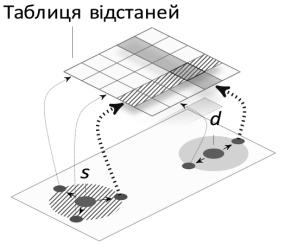

Рис 5. TNR запит

Первинна обробка алгоритму TRANSIT використовує (невеликий) граф зупинок для визначення транзитних вузлів, між якими будується таблиця подорожей з мінімальним часом подорожей. Кожна зупинка p зберігає множину вузлів доступу $A(p)$, визначену на розширеному за часом графі запуском локальних пошуків з кожної події відправлення p до транзитних зупинок. На етапі запиту використовуються вершини доступу зупинок p_s і p_d і таблиця відстаней для вирішення глобальних запитів. Для локальних запитів, виконується запуск алгоритму A*.

1. Висновки

Розглянута залежна від часу задача пошуку найкоротшого шляху, яка використовується, зокрема, для планування подорожі громадським транспортом. Проаналізовано підходи до подання розкладу громадського транспорту у вигляді графа — розширеного за часом моделі графу і залежна від

часу модель. Показані модифікації графу для відображення реалістичних моделей. Досліджені останні підходи розв'язання залежних від часу задач пошуку найкоротших шляхів з врахуванням розкладу.

Література

1. Cooke K. L., Halsey E. The shortest route through a network with time-dependent intermodal transit. — J. Math. Anal. Appl, 1966. — P. 493–498.
2. Bellman R. On a routing problem. — Quarterly of Applied Mathematics, 1958. — P. 87–90.
3. Bellman R. Dynamic Programming — Princeton University Press, Princeton, New Jersey, 1957.
4. Dreyfus S. E. An appraisal of some shortest-path algorithms. — Operations Research, 1969. — P. 395-412.
5. Kaufman D. E., Smith R. L. Fastest paths in time-dependent networks for intelligent vehicle-highway systems application. // J. Intelligent Transportation Systems. — 1993. — №1. — P. 1–11.
6. Pajor T. Algorithm Engineering for Realistic Journey Planning in Transportation Networks [Text] : Dissertation. Doktors der Naturwissenschaften : Tag der m?ndlichen Pr?fung: 15 November 2013 / Thomas Pajor. — 2013. — 266 p.
7. Muller-Hannemann M. Timetable information: Models and algorithms. / Matthias Muller–Hannemann, Frank Schulz, Dorothea Wagner, and Christos Zaroliagis // In Algorithmic Methods for Railway Optimization / Lecture Notes in Computer Science, Springer. — 2007. — V. 4359 — P. 67–90.
8. Hart P. E., Nilsson N., Raphael B. A formal basis for the heuristic determination of minimum cost paths. // IEEE Transactions on Systems Science and Cybernetics. — 1968. — №4. — P. 100–107.
9. Disser Y., Muller–Hannemann M., Schnee M/ Multi-criteria shortest paths in time-dependent train networks. // In Proceedings of the 7th Workshop on Experimental Algorithms (WEA'08) / Lecture Notes in Computer Science, Springer. — 2008. — V. 5038. — P. 347–361.
10. Goldberg A., Harrelson C. Computing the shortest path: A* search meets graph theory. // Technical Report MSR-TR-2004-24, Microsoft Research. — 2004. — 25 p.
11. Delling D., Pajor T., Wagner D. Engineering timeexpanded graphs for faster timetable information. // Robust and Online Large-Scale Optimization / Lecture Notes in Computer Science, Springer. — 2009. — V. 5868 — P. 182–206.
12. Delling D. Time-dependent SHARC-routing. // Algorithmica 60. — 2011. — №1. — P. 60–94.
13. F. Schulz., Wagner D., Weihe K. Dijkstra's algorithm on-line: An empirical case study from public railroad transport. // ACM Journal of Experimental Algorithmics. — 2000. — №5. — P. 1–23.
14. Wagner D., Willhalm T., Zaroliagis C. Geometric containers for efficient shortest-path computation. // ACM Journal of Experimental Algorithmics. — 2005. — №10. — P. 1–30.
15. Hilger M., Kцhler E., Mцhring R. H., Schilling H. Fast point-to-point shortest path computations with arc-flags. // In The Shortest Path Problem: Ninth DIMACS Implementation Challenge. — 2009. — V. 74. — P. 41–72.
16. Delling D. Engineering and augmenting route planning algorithms : Ph.D. thesis, Universitдt Karlsruhe (TH), Fakultдt fur Informatik, 2009. — 2009.

17. Berger A., Delling D., Gebhardt A., Muller–Hannemann M. Accelerating time-dependent multi-criteria timetable information is harder than expected / Annabell Berger // Proceedings of the 9th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'09), OpenAccess Series in Informatics (OASIcs). — 2009.
18. Delling D., Goldberg A. V., Razenshteyn I., Werneck R. F. Graph partitioning with natural cuts. // In 25th International Parallel and Distributed Processing Symposium (IPDPS'11). — 2011. — P. 1135–1146.
19. Holzer M., Schulz F., Wagner D. Engineering multi-level overlay graphs for shortest-path queries. // Invited for ACM Journal of Experimental Algorithmics. — 2007.
20. F. Schulz. Using multi-level graphs for timetable information in railway systems / Frank Schulz, Dorothea Wagner, and Christos Zaroliagis // Proceedings of the 4th Workshop on Algorithm Engineering and Experiments (ALENEX'02), Lecture Notes in Computer Science, Springer. — 2002. — V. 2409. — P. 43–59.
21. Geisberger R., Sanders P., Schultes D., Vetter Ch. Exact routing in large road networks using contraction hierarchies // Transportation Science. — 2012. — V. 3. — P. 388–404.
22. Schultes D., Sanders P. Dynamic highway-node routing. // In Proceedings of the 6th Workshop on Experimental Algorithms (WEA'07). — 2007. — V. 4525. — P. 66–79.
23. Sanders P., Schultes D. Engineering highway hierarchies. // ACM Journal of Experimental Algorithmics. — 2012. — №17. — P. 1–40.
24. Bast H., Carlsson E., Eigenwillig A., Geisberger R., Harrelson Ch., Raychev V., Viger F. Fast routing in very large public transportation networks using transfer patterns. // In Proceedings of the 18th Annual European Symposium on Algorithms (ESA'10) / Lecture Notes in Computer Science, Springer. — 2010. — V. 6346. — P. 290–301.
25. Google Transit System [Electronic resource] // Google. — Mode of access: http://maps.google.com/landing/transit/index.html .
26. Antsfeld L., Walsh T. Finding multi-criteria optimal paths in multi-modal public transportation networks using the transit algorithm. // In Proceedings of the 19th ITS World Congress, 2012.
27. Sanders P., Schultes D. Robust, almost constant time shortest-path queries in road networks. // In The Shortest Path Problem: Ninth DIMACS Implementation Challenge. — 2009. — V. 74 of DIMACS Book. — P. 193–218.

Інститут кібернетики імені В.М.Глушкова Національної академії наук України, Проспект Академіка Глушкова, 40, Київ, 03680, Україна.

Надійшла 01.09.2014