УДК 004.415.3
Сопіжук Б.В., Демидюк М.А.
Луцький національний технічний університет

ДОСЛІДЖЕННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ ПОШУКУ ОПТИМАЛЬНОГО МАРШРУТУ З ДИНАМІЧНИМ ВИБОРОМ АЛГОРИТМУ

Дана стаття описує алгоритми пошуку оптимального маршруту. Під вибором методу пошуку оптимального маршруту мається на увазі вибір найбільш кращого алгоритму пошуку шляху на графі. Для пошуку оптимального маршруту на графах відомо багато алгоритмів, які б дозволяли враховувати умови задачі. Ці алгоритми на зважених графах з вагами ребер. До таких алгоритмам відносяться: алгоритм Дейкстри, мурашиний алгоритм. Вони дають можливість прокладати шлях з початкової точки або в кінцеву точку, або в усі можливі точки. Далі розглянемо можливі алгоритми.

Ключові слова: інформаційно-комп'ютерна система, gps, оптимальний шлях, критерії, мурашиний алгоритм, додаткові параметри, алгоритм Дейкстри.
Форм. 4. Літ. 5.

Постановка проблеми. Інфраструктури міст стрімко розвиваються, з'являються нові будівлі, шляхи руху транспорту, відповідно карти маршрутів швидко застарівають, і ускладнюється можливість пошуку оптимального маршруту. Так само для спеціалізірованого транспорту та екстрених служб завжди існує необхідність у наявності найбільш швидкого шляху, оскільки від втраченого часу на пересування можуть залежати людські життя. Тому на сьогоднішній день актуальним є створення системи, яка дозволить максимально швидко і якісно прокладати найкоротший маршрут з урахуванням додаткових параметрів.

Існуючі системи пошуку оптимального маршруту не використовують клієнт-серверну архітектуру, і в основному націлені для пошуку маршруту одним конкретним користувачем. За цим, дослідження зводяться до аналізу використовуваних на даний момент систем, а так само тестуванню можливих алгоритмів пошуку шляху і вибором найбільш оптимальних з них.

Аналіз останніх досліджень та публікацій. На сьогоднішній день системи пошуку оптимального шляху досить широко поширені і ефективно використовуються всіма видами транспорту. Принцип роботи таких систем полягає у визначенні місця розташування об'єкта та прокладання оптимального маршруту пересування.

Система NAVITEL: підтримуються пристрої, оснащені як з вбудованими так і з зовнішніми GPS-приймачами; система дозволяє завантаження докладних карт міст і областей з номерами будинків, назвами вулиць, станцій метро; можливість створювати власні карти на основі викладених у вільному доступі фрагментів за допомогою картографічного редактора GPSMapEdit, експортуючи їх у формат "Навітел Навігатор"; карта в "Навітел Навігатор" може відображатися як у двомірному, так і в тривимірному вигляді; система підтримує автороутінг з урахуванням стану дорожньої мережі (наявність платних трас, розворотів) і заторів; запис треків ведеться у форматах MPS і GPX (імпортувати їх можна з інших додатків, наприклад з Ozi Explorer); реалізований гнучкий алгоритм побудови маршрутів, що дозволяє користувачеві будувати маршрути з урахуванням власних уподобань; маршрути об'їзду з урахуванням завантаженості магістралей і даних про поточну дорожню ситуацію (пробки, затори т.п.); присутній режим ДЕМО (режим імітації) - функція попереднього перегляду маршруту, імітація руху по маршруту; наявність функції Навітел Пробки (функція відображення пробок на дорогах, дані для відображення пробок надходять від самих користувачів); наявність функції SpeedCam - оновлювана база даних про системи контролю швидкості (камери, радари) на дорогах (можна скачувати з сайту або користуватися альтернативними збірками); вимагає від 14 МБ оперативної пам'яті (залежно від завантаженої картки); GPS-приймачі: підтримка протоколів NMEA-0183, SiRF binary і Garmin; зовнішній приймач може підключатися через послідовний порт, USB, BlueTooth, роз'єми CompactFlash, SDIO.

Система Nav N Go iGO 8: поставляється на карті пам'яті - microSD об'ємом 2 Гб з адаптерами на MiniSD і SD; працює в різних пристроях з операційною системою - Windows Mobile і не вимагає установки; загальний об'єм карт складає близько 1,6 Гб; рельєф як на географічних картах; мінімальна підтримувана частота процесора пристрою - 300 МГц (рекомендується 400МГц) і 64Мб оперативної пам'яті (рекомендується 128 Мб); функція прокладання маршруту між заданими точками існує в чотирьох режимах: «Простий», «Короткий», «Швидкий» і « Економний »; є вибір типу транспортного засобу:« Автомобіль »,« Таксі »,« Автобус »,« Транспорт екстрених служб »,« Велосипед »і навіть« Пішохід »; можливість обліку типів доріг, по

яких рухається користувач (« Грунтові », «Що вимагають дозволу», «Шосе», «Платні дороги», «Дороги з періодичною оплатою», «Пароми», «Планування перетину вулиць»); система попереджає про наявність на дорозі камер стеження заздалегідь (відомості про камери можна оновлювати з сайту виробника); підтримується операційна система Windows Mobile 5, 6, 6.1; 6.5; 6.5.3; можливість використання сенсорного екрану QVGA або VGA або PNA; підтримувані дозволи екрану: 240x240, 240x320, 240x400, 320x234, 320x240, 320x320, 400x234, 400x240, 416x234 , 480x234, 480x272, 480x800 & 800x480.

Система Garmin Mobile XT: для роботи з системою потрібно смартфон або кишеньковий комп'ютер оснащений модулем GPS і підтримуваний технологію Bluetooth; програма може працювати як з вбудованим модулем GPS так і з зовнішнім; система підтримує роботу з мобільними пристроями під управлінням операційних систем Symbian, Windows Mobile або Palm OS; для запуску програми потрібно мінімум 8 Мб оперативної пам'яті (при подальшій роботі споживання може зрости і до 20Мб); необхідно наявність карти пам'яті об'ємом мінімум 128Мб (рекомендується 256Мб і більше); система поставляється на карті пам'яті формату microSD; система дозволяє передати по SMS географічні координати (широту і довготу) місця розташування об'єкта, якщо він користується смартфоном; програма може шукати тільки прості адреси виду «назва вулиці та номер будинку» (пошук корпусів неможливий); відображає фотографії пам'яток, поблизу з якими знаходиться користувач; величезна кількість сервісів, випускають точки POI для системи, як необхідні користувачеві об'єкти на мапі, так і оповіщення про постах ДПС; функція «Де я?» дозволяє дізнатися місце розташування користувача і адресу, за якою він перебуває; функція «Дорожній комп'ютер» дозволить дізнатися час, що минув з моменту старту користувача, і виділити з нього час руху і час стоянок, обчислити максимальну швидкість, середню швидкість з урахуванням і без урахування стоянок; за допомогою сервісу Garmin Online можна отримати інформацію про дорожні затори, погоду в місці перебування користувача, цінах на паливо і камерах стеження на автомагістралях; система підтримує відкриття 4-х файлів карток відразу.

Невирішені частини проблеми. Можливе подальше удосконалення архітектури систем пошуку маршруту, оскільки в існуючих системах відсутня можливість робити моніторинг пересуваються об'єктів, а так само можливість зміни роботи системи у випадку позаштатних ситуацій. Так само можливо удосконалення клієнтської частини системи шляхом установки датчиків на транспортний засіб, таких як давач вібрації, давач вологості.

Метою даного дослідження є створення можливості використовувати будь-які алгоритми пошуку оптимального шляху на графах з можливістю їх динамічного підключення.

Основні результати дослідження. Для дослідження обрано два алгоритми пошуку оптимального шляху, це алгоритм Дейкстри і мурашиний алгоритм.

Тестування і порівняння результатів роботи алгоритмів проводилося за такими характеристиками:

- Час виконання алгоритму;
- Кількість точок на карті;
- Кількість початкових точок.

Після проведення тестування обох алгоритмів було виявлено, що за наявності однієї початкової точки алгоритм Дейстра в рази швидше справляється із завданням пошуку маршруту, ніж розроблена модифікація мурашиного алгоритму. Але при збільшенні кількості початкових точок до максимально можливого, модифікація мурашиного алгоритму виконує пошук оптимального шляху практично в два рази швидше алгоритму Дейкстри.

Алгоритм Дейкстри - алгоритм на графах, винайдений нідерландським ученим Е. Дейкстрой в 1959 році. Знаходить найкоротшу відстань від однієї з вершин графа до всіх інших. Алгоритм працює тільки для графів без ребер негативної ваги. Алгоритм широко застосовується в програмуванні і технологіях, наприклад, його використовує протокол OSPF для усунення кільцевих маршрутів.

На вхід даному алгоритму надходить вихідний зважений граф, з вказівку початкової та кінцевої вершин між якими необхідно знайти найкоротший шлях. В результаті отримуємо довжину шляху і послідовність пунктів складових даний шлях.

Особливості:

Вершини можуть мати постійні або тимчасові мітки λ.

Вводиться додатковий масив тимчасових міток $\theta j=\theta(x_j)$, якій присвоюється значення рівне номеру вершини, яка дала постійну мітку вершині x_j.

Алгоритм закінчує свою роботу, коли кінцева вершина отримує постійну мітку. Значення цієї мітки дорівнюватиме найкоротшому шляху, а зворотна послідовно міток θ дасть цей найкоротший шлях.

Опишемо роботу алгоритму.

а) початковій вершині присвоюється мітка $\lambda=0$, а решті вершин мітки $\lambda=\infty$. Так як мітки можуть бути двох типів (постійні і тимчасові), то початкова вершина отримує постійну мітку, а інші тимчасові. Постійне мітка буде дорівнює довжині найкоротшого шляху від початкової вершини до даної. Індексом Р будемо позначати номер вершини, що отримала постійну мітку останньої.

б) для всіх вершин, суміжних з вершиною, яка отримала постійну позначку, та які ще мають тимчасову мітку, виконується перевірка:

якщо $\lambda_j > \lambda_p\ l(x_p,x_j)$ то $\lambda_j=\lambda_p+ l(x_p,x_j)$, тобто зменшуються індекси вершин, які мають тимчасові мітки. А в масив послідовності вершин найкоротшого шляху заноситься індекс постійної мітки, яка дала значення зменшеним тимчасовим мітках: $\theta(x_j)=p$

в) визначаємо безліч вершин з тимчасовими мітками, тобто, віднімаємо з вихідної безлічі вершин вершини з постійними мітками. Серед безлічі вершин з тимчасовими мітками знаходимо вершину з мінімальною міткою і робимо її постійною.

Кроки 2-3 виконуються доти поки кінцева вершина не отримає постійну мітку. Значення цієї мітки дорівнюватиме найкоротшому шляху, а зворотна послідовно міток θ дасть цей найкоротший шлях.

Мурашині алгоритми грунтуються на імітації природних механізмів самоорганізації мурах, використання яких ілюструється в наступних розділах на прикладі оптимізації маршруту комівояжера.

Мурашиний підхід схожий до задачі комівояжера

Завдання комівояжера полягає у пошуку найкоротшого замкнутого маршруту, що проходить через усі міста рівно один раз. Вибір завдання комівояжера для ілюстрації ідей мурашиних алгоритмів обумовлений наступним:

1. Завдання наочно інтерпретується в термінах поведінки мурашок - переміщення комівояжера і мурах інтуїтивно порівнянні.

2. Це NP-складна задача, яка на недетермінованої машині Тьюринга вирішується за поліноміальний час.

3. Це традиційний тестовий полігон (benchmark problem) для методів комбінаторної оптимізації. Існує велика база тестових завдань про комівояжера і методів їх вирішення, що дозволяє порівняти ефективність мурашиних алгоритмів оптимізації з іншими підходами.

4. Це дидактична задача, для якої можна легко, без зловживання технічними подробицями алгоритму пояснити процес знаходження оптимального рішення.

5. Це перша комбінаторна задача, вирішена мурашиними алгоритмами.

Розглянемо, як реалізувати чотири складові самоорганізації мурах під час оптимізації маршруту комівояжера. Багаторазовість взаємодії реалізується ітераційним пошуком маршруту комівояжера одночасно кількома мурашками. При цьому кожен мураха розглядається як окремий, незалежний комівояжер, вирішальний своє завдання. За одну ітерацію алгоритму кожен мураха робить повний маршрут комівояжера. Позитивний зворотний зв'язок реалізується як імітація поведінки мурах типу «залишення слідів-переміщення по слідах». Чим більше слідів залишено на стежці - ребрі графа в задачі комівояжера, тим більше мурах буде пересуватися по ній. При цьому на стежці з'являються нові сліди, які залучають додаткових мурах. Для задачі комівояжера позитивний зворотний зв'язок реалізується наступним стохастичним правилом: ймовірність включення ребра графа в маршрут мурахи пропорційна кількості феромону на ньому.

Застосування такого імовірнісного правила забезпечує реалізацію та іншої складаю щей самоорганізації - випадковості. Кількість відкладали мурахою феромону на ребрі графа обернено пропорційно довжині маршруту. Чим коротше маршрут, тим більше феромону буде відкладене на відповідних ребрах графа і тим більше мурах буде використовувати їх при синтезі своїх маршрутів. Відкладений на ребрах феромон виступає як підсилювач, він дозволяє хорошим маршрутами зберігатися в глобальній пам'яті мурашника. Ці маршрути можуть бути поліпшені на наступних ітераціях алгоритму. Використання тільки позитивного зворотного зв'язку приводить

до передчасної збіжності рішень - до випадку, коли всі мурашки рухаються одним і тим же субоптимальних маршрутом. Для уникнення цього використовується негативний зворотний зв'язок - випаровування феромону. Час випаровування не повинно бути занадто великим, тому що при цьому виникає небезпека збіжності популяції маршрутів одного субоптимального рішенням. З іншого боку, час випаровування не повинно бути і занадто малим, так як це призводить до швидкого «забування», втрати пам'яті колонії і, отже, до некооперативних поведінки мурах. У поведінці мурах корпоративність є дуже важливою: безліч ідентичних мурах одночасно досліджують різні точки простору рішень і передають свій досвід через зміни осередків глобальної пам'яті мурашника. Для кожного мурашки перехід з міста і в місто j залежить від трьох складових: пам'яті мурахи (tabu list), видимості та віртуального сліду феромону.

Tabu list (пам'ять мурашки) - це список відвіданих мурахою міст, заходити в які ще раз можна. Використовуючи цей список, мураха гарантовано не потрапить в один і той же місто двічі. Ясно, що tabu list зростає при вчиненні маршруту і обнуляється на початку кожної ітерації алгоритму. Позначимо через J_{ik}, перелік міст, які ще необхідно відвідати мурашки k, що знаходиться в місті i. Зрозуміло, що Jik,, є доповненням до tabu list.

Видимість - величина, зворотна відстані: $ή_{ij} = 1 / D_{ij}$, де D_{ij} - відстань між містами i та j. Видимість - це локальна статична інформація, що виражає евристичне бажання відвідати місто j з міста i - чим ближче місто, тим більше бажання відвідати його. Використання тільки видимості, звичайно, є недостатнім для знаходження оптимального маршруту.

Віртуальний слід феромону на ребрі (i, j) являє підтверджене мурашиним досвідом бажання відвідати місто j з міста i. На відміну від видимості слід феромону є більш глобальною і динамічною інформацією - вона змінюється після кожної ітерації алгоритму, відображаючи придбаний мурахами досвід. Кількість віртуального феромону на ребрі (i, j) на ітерації t позначимо через τ_{ij} (t).

Важливу роль у мурашиних алгоритмах грає ймовірнісно-пропорційне правило, що визначає ймовірність переходу з міста i в місто j на t й ітерації:

$$\begin{cases} P_{ij,k}(t) = \dfrac{\left[\tau_{ij}(t)\right]^{\alpha} \times \left[\eta_{ij}\right]^{\beta}}{\sum_{l \in J_{i,k}} \left[\tau_{il}(t)\right]^{\alpha} \times \left[\eta_{ij}\right]^{\beta}} \\ P_{ij,k}(t) = 0, \textit{ якщо} \quad j \notin J_{i,k}, \end{cases} \tag{1}$$

де a і b - два регульованих параметра, що задають ваги сліду феромону і видимості при виборі маршруту. При a = 0 буде обраний найближче місто, що відповідає жадібному алгоритмом в класичній теорії оптимізації. Якщо b = 0, тоді працює лише феромон посилення, що тягне за собою швидке виродження маршрутів одного субоптимального рішенням.

Звернемо увагу, що правило (1) визначає лише ймовірності вибору того чи іншого міста. Власне вибір міста здійснюється за принципом «колеса рулетки»: кожне місто на ній має свій сектор з площею, пропорційною ймовірності (1). Для вибору міста потрібно кинути кульку на рулетку - згенерувати випадкове число, і визначити сектор, на якому ця кулька зупиниться.

Зауважимо, що хоча правило (1) не змінюється протягом ітерації, значення ймовірностей Pij, k (t) для двох мурах в одному і тому ж місті можуть відрізнятися, т. к. $P_{ij\,k}$, () - функція від $J_{i\,k}$, - списку ще відвіданих міст мурахою k.

Після завершення маршруту кожен мураха k відкладає на ребрі (i, j) таку кількість феромону:

$$\Delta\tau_{ij,k}(t) = \begin{cases} \dfrac{Q}{L_k(t)}, & \textit{якщо} \quad (i, j) \in T_k(t), \\ 0, & \textit{якщо} \quad (i, j) \notin T_k(t). \end{cases} \tag{2}$$

де T_k (t) - маршрут, пройдений мурахою k на ітерації t; L_k (t) - довжина цього маршруту; Q - регульований параметр, значення якого вибирають одного порядку з довжиною оптимального маршруту.

Для дослідження всього простору рішень необхідно забезпечити випаровування феромону - зменшення в часі кількості відкладеного на попередніх ітераціях феромону.

Позначимо коефіцієнт випаровування феромону через р є [0, 1]. Тоді правило поновлення феромону (3) прийме вигляд:

$$\tau_{ij}(t+1) = (1-p)\cdot\tau_{ij}(t) + \Delta\tau_{ij}(t) \qquad (3)$$

На початку оптимізації кількість феромону приймається рівним невеликому позитивному числу t0. Загальна кількість мурах в колонії залишається постійним протягом виконання алгоритму. Численна колонія призводить до швидкого посилення субоптимальних маршрутів, а коли мурах мало, виникає небезпека втрати кооперативності поведінки через обмежена взаємодія і швидке випаровування феромону. Звичайно число мурах призначають рівною кількості міст - кожен мураха починає маршрут зі свого міста.

Для поліпшення часових характеристик мурашиного алгоритму вводять так званих елітних мурах. Елітний мураха посилює ребра найкращого маршруту, знайденого з початку роботи алгоритму. Кількість феромону (4), відкладали на ребрах найкращого поточного маршруту Т, приймається рівним Q / L^+, де L^+ - довжина маршруту Т. Цей феромон спонукає мурах до дослідження рішень, що містять кілька ребер найкращого на даний момент маршруту Т. Якщо в мурашнику є е елітних мурах, то ребра маршруту Т отримуватимуть загальне посилення:

$$\Delta\tau_e = \frac{e\cdot Q}{L^+} \qquad (4)$$

Висновки. У результаті проведеного моделювання роботи алгоритмів було встановлено, що для вирішення різних завдань кожен алгоритм має свої переваги. Тому в системі використовуються обидва алгоритми з можливістю динамічного підключення інших алгоритмів. Але все-таки специфіка даного завдання має на увазі пошук оптимального маршруту паралельно для декількох об'єктів, для чого розроблена модифікація мурашиного алгоритму впоралася приблизно в 2 рази краще, ніж алгоритм Дейкстри.

1. Електронні компоненти [Електронний ресурс].-Режим доступу: URL: http://www.chipinfo.ru/literature/chipnews/200110/9.html.-Системи супутникової навігації ГЛОНАСС і GPS.
2. Рейнгольд Э. Комбинаторные алгоритмы. Теория и практика. / Э.Рейнгольд, Ю. Нивергельт, Н. Део; пер. с англ. Е.П. Липатова. - М.: Мир, 1980. — 476 с.
3. Пушкарёва Г.В. Исследование и применение бионических методов и моделей для автоматизированного проектирования маршрутов обхода геометрических объектов [Электронный ресурс].- Компьютерная графика и представление GraphiCon '2005: науч.-техн. конф., 20-24 июня 2005г. : - 2005.
4. Сухарев А.Г. Курс методов оптимизации: Учебное пособие. - [2-е изд]. / А.Г. Сухарев, А.В. Тимохов, В.В. Федоров. – М.: ФИЗМАТЛИТ, 2005. – 368 с.
5. Харчистов Б.Ф. Методы оптимизации: учебное пособие / Б.Ф. Харчистов. – Таганрог.: ТРТУ, 2004. – 140 с.