Зміна фази руху циліндра 4 на протилежний по відношенню до циліндра 2 теж не покращує процес розкочування фарби (рис. 4б). З аналізу графіків (рис. 3, 4) бачимо, що транспортне запізнення впливає лише на перехідний процес і майже не впливає на рівномірність товщини шару фарби на виході. Зміна позиції розтирального циліндра з третьої на четверту, тобто наближення його до виходу, спричиняє збільшення коливань товщини шару фарби, що передається на задруковуваний матеріал. А рух циліндрів зі зміщенням за фазою на 180°, що забезпечується в друкарських машинах, не приводить до відчутного поліпшення рівномірності шару фарби на виході розкочувальної групи.

1. Алексеев Г.А. Красочные аппараты ротационных машин высокой и плоской печати. М, 1980. 2. Верхола М.І., Гук І.Б. Моделювання та аналіз впливу траєкторії руху розтирочного циліндра на процес розкочування фарби // Комп'ютерні технології друкарства: Зб. наук. пр. Львів. 2001. № 6. С. 337–352. 3. Верхола М.І., Гук І.Б., Луцків М.М. Модель осьового розкочування фарби у тривалковій фарбовій групі // Наукові записки. Львів: УАД. 1999. Вип. 1. С. 50–52. 4. Раскин А.Н., Ромейков И.В. и др. Технология печатных процессов. М, 1989.

УДК 681.513:519.713

ОСНОВНІ ЗАДАЧІ ВІЗУАЛІЗАЦІЇ ГРАФІВ, ЩО ОПИСУЮТЬ ТОПОЛОГІЇ ПОЛІГРАФІЧНИХ СИСТЕМ

Р. Б. Дунець, Т. М. Басюк

Описуються основні задачі, які потрібно розв'язати для візуалізації графів у програмах топологічного аналізу.

Описываются основные задачи, которые необходимо решить для визуализации графов в программах топологического анализа.

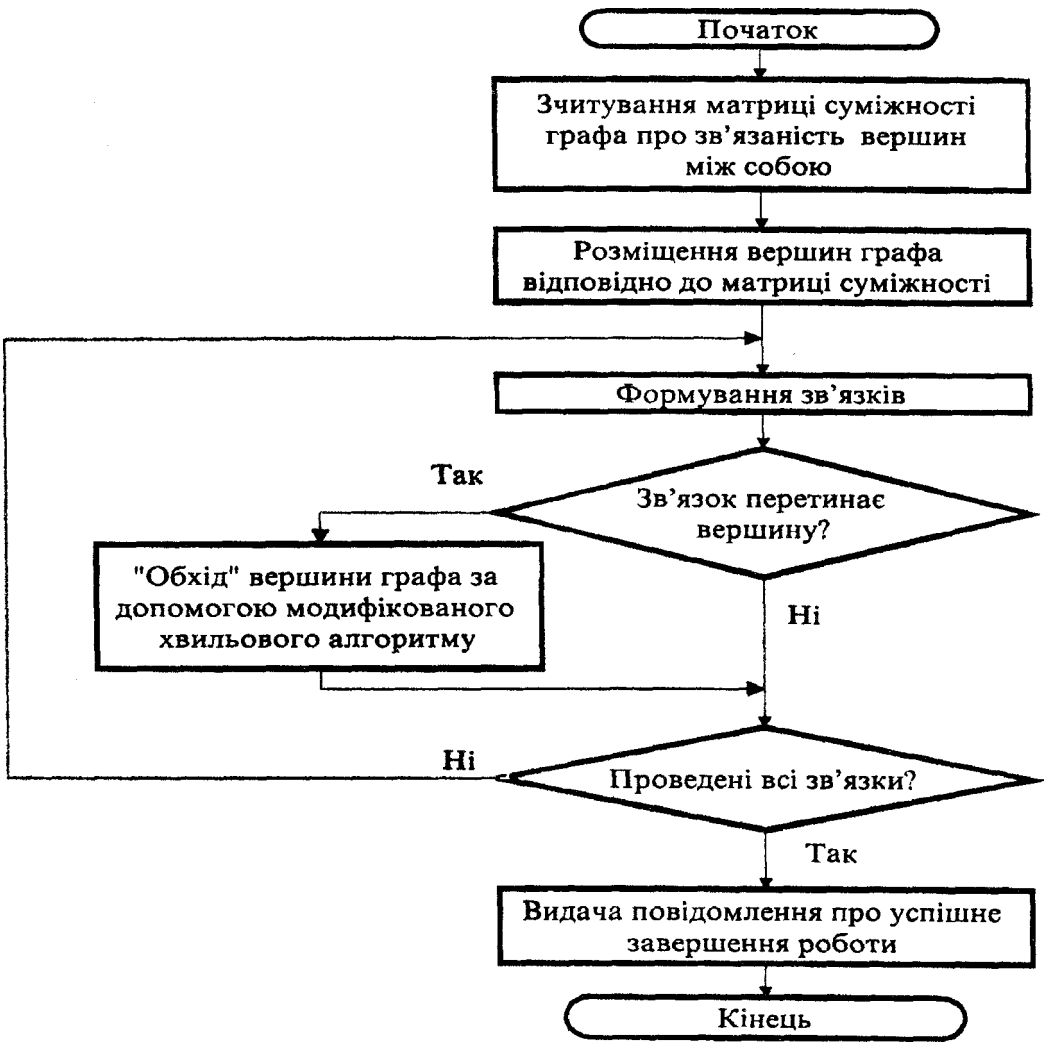
З інтенсивним впровадженням комп'ютерної техніки виникла потреба в створенні різних прикладних програм, у тому числі програм топологічного аналізу [2]. Відомо, що топологічні методи аналізу широко застосовуються при проектуванні різноманітних систем: електричних [8], електромеханічних [14], електронних [1], мікроелектронних [9], поліграфічних [11] тощо.

Для опису топології (зв'язків між компонентами систем) застосовуються аналітичні [4], графічні [6, 12, 15] та матричні [3,12] способи зображення. Аналітичні способи забезпечують компактність запису топологій на папері, але вони трудомісткі для введення й опрацювання в комп'ютерах і не є наочними. Графічні способи зображення є наочними, але також трудомісткими, як аналітичні. Матричні способи зображення найзручніші для зберігання та опрацювання в комп'ютерах, але не мають такої наочності, як графічні.

У програмах топологічного аналізу, як правило, введення топологій проводиться в графічному зображенні з подальшим його автоматичним перетворенням у машинне зображення, наприклад, у матриці суміжності. Після необхідних перетворень над топологіями результат виводиться, зазвичай, у машинних зображеннях як масив даних. Тому актуальною є задача відображення (візуалізації) у вигляді графів топологій, що представлені матрицями.

Загальний алгоритм візуалізації графів (див. рисунок) передбачає такі основні етапи: читання в пам'ять комп'ютера матриці суміжності, що описує топологію, яку потрібно вивести на екрані у вигляді графа; розміщення вершин графа; формування зв'язків між вершинами графа відповідно до заданої матриці.

Перший етап не викликає жодних труднощів, оскільки він є підготовчим у процесі візуалізації і зв'язаний лише з операцією читання матриці суміжності, яка в пам'яті комп'ютера зберігається як двомірний масив однорозрядних двійкових чисел.



Алгоритм візуалізації графів

На другому етапі виникає задача встановлення послідовності розміщення вершин, вибору місця для розташування кожної вершини графа в межах заданої площі екрана. Тут труднісь полягає в тому, що жодна з відомих мов опису та перетворення графів не містить ознаки чи параметра, що вказує на взаємне розташування вершин графа на площині. Основними задачами, які потрібно розв'язати на цьому етапі, є формування критеріїв, за якими, аналізуючи матрицю суміжності, можна визначити послідовність розміщення вершин на екрані, а також критеріїв, за котрими встановлюється місце виведення вершини на площині екрана.

На третьому етапі рисуються лінії зв'язків між вершинами графа. Для розв'язання цієї задачі можна було б застосувати алгоритм Лі знаходження мінімального шляху між двома вершинами [7], так званий хвильовий алгоритм чи алгоритм пошуку в ширину. Відповідно до нього спочатку треба подібно до принципу Гюйгенса, де кожна точка хвильового фронту є джерелом вторинної хвилі, із заданої вершини А відвідати всі суміжні з нею вершини. Кожна вершина, яку відвідали, стає джерелом нової хвилі. Якщо ж суміжної вершини не виявлено, то здійснюється повернення до тієї вершини, з якої потрапили в поточну, та робиться наступна спроба. При досягненні кінцевої вершини В прокладається шлях між двома точками в зворотному напрямку – від точки В до точки А. На цьому шляху кожен раз вибирається точка, що розміщується найближче до точки А. Послідовність вибраних точок становить шуканий шлях.

Хвильовий алгоритм характеризується відносною простотою знаходження мінімального шляху між двома вершинами [10]. Але задача візуалізації графа потребує не стільки знаходження мінімального шляху між вершинами, скільки перетину ребер між собою та забезпечення мінімального шляху обходження вершини лінією зв'язку тоді, коли на шляху від вершини А до вершини В є інші вершини, з якими дані вершини не мають зв'язків. У цьому випадку для зв'язку між вершинами можна використати модифікований хвильовий алгоритм, прийнявши за однакові розміри та накреслення вершин. Модифікація алгоритму стосуватиметься спочатку пошуку шляху у разі взаємного перетину ліній зв'язків між вершинами графа. Далі модифікація алгоритму повинна торкатись виявлення ситуацій перетину лінії зв'язку з іншими вершинами графа і забезпечення „обходу" вершини графа зв'язком, що її перетинає. Для організації „обходу" необхідно сформулювати критерії, на підставі яких можна було б забезпечувати потрібну відстань від такого зв'язку до вершини й вибирати тип лінії зв'язку (відрізок прямої, відрізок дуги тощо).

Окремою задачею є вибір мови програмування для створення прикладної програми топологічного аналізу, що передбачає візуалізацію графа топології. При створенні першого покоління програм для роботи з графами використовувалися мови програмування Fortran, Algol, PL\1, Сі. Для розв'язання теоретико-графових задач застосовувалися й непроцедурні мови, такі, як мова функціонального програмування LISP і логічного програмування PROLOG. Однак через недостатню ефективність і технологічні труднощі розробки великих програмних систем на цих мовах вони не підходять для створення універсальних програм [10].

З розвитком об'єктно-орієнтованого (візуального) програмування (ООП) розпочалася розробка об'єктно-орієнтованих програм для роботи з графами. Використання засобів ООП при розв'язанні теоретико-графових задач дає істотні переваги порівняно з традиційним структурним підходом, оскільки сам граф, його вершини і ребра є „готовими" об'єктами, даними самою природою задачі. При роботі з графами найвиразніше проявляються такі:

програмний код стає компактнішим і поліпшується його розуміння;

з'являється можливість візуально спостерігати й вносити зміни в структуру графів.

При розробці візуальної оболонки для роботи з графами виникають також численні технічні труднощі. Реалізація всіх необхідних структур даних у рамках однієї програми навряд чи можлива і виправдана, тому універсальна програма для роботи з графами вимагає серйозної програмної „інфраструктури" у вигляді різних бібліотек [7,10].

Перераховані задачі можуть викликати сумнів щодо доцільності створення універсальної програми для роботи з графами, однак існують вагомі аргументи на користь вирішення цього питання. По-перше, реалізовані в такій програмі алгоритми можуть бути доброю основою для створення більш спеціалізованих алгоритмів і програм, спрямованих на розв'язання конкретних прикладних задач. По-друге, розуміння ефективності не завжди є визначальним: постійний ріст продуктивності ЕОМ усе частіше виводить на перший план технологічність і швидкість розробки програмного забезпечення (зрозуміло, це не означає, що програміст не повинен прагнути до ефективного використання обчислювальних ресурсів). Поряд з промисловим використанням універсальна програма для роботи з графами може застосовуватись з навчальною метою [5], а також для підтримки теоретичних досліджень, пов'язаних з алгоритмами і програмами розв'язання задач теорії графів.

Таким чином, успішне розв'язання вищезазначених задач візуалізації графів дозволить створити прикладну програму топологічного аналізу поліграфічних систем з гнучким інтерфейсом користувача щодо виведення опрацьованої інформації як на екрані монітора, так і на папері у зручному для сприйняття вигляді.

1. Базилевич Р.П. Декомпозиционные и топологические методы автоматизированного проектирования электронных устройств. Львов,1981. 2. Дунець Р. Концепція створення програмного пакета топологічного аналізу // Комп'ютерні технології друкарства: Зб. наук. пр. Львів, 1999. 3. Дунець Р.Б. Алгоритм пошуку контурів у топології схем систем керування // Поліграфія і видавнича справа. 1997. № 32. С. 103–108. 4. Зарічний М.М. Топологія функторів і монад у категорії компактів. К., 1993. 5. Зыков А.А. Основы теории графов. М., 1987. 6. Кристофидес Н. Теория графов. Алгоритмический подход. М., 1978. 7. Майника Э. Алгоритмы оптимизации на сетях и графах. М., 1981. 8. Максимович Н.Г. Теория графов и электрические цепи. Львов, 1987. 9. Мельник Р.А. Алгоритми ієрархічного моделювання площинної та просторової топології НВІС.

Львів, 1999. 10. Нечепуренко М.И., Попков В.К., Майнагашев С.М. и др. Алгоритмы и программы решения задач на графах и сетях. Наука (сибирское отделение), Новосибирск, 1990. 11. Рак Ю.П. Малі друкарські системи: прогнозування, аналіз, синтез. К., 1999. 12. Савчак І. Алгоритм та програма виявлення деревовидних структур схем систем керування за допомогою матриць інциденцій // Комп'ютерні технології друкарства: Зб. наук. пр. Львів, 1999. 13. Харари Ф. Теория графов. М., 1973. 14. Dunets' R. Topology analysis algorithms of electromechanical schemes / Наукові праці конференції „Комп'ютерні технології друкарства: алгоритми, сигнали, системи „ДРУКОТЕХН-96": Львів, 16–18 жовтня 1996 р. Львів: УАД, 1996. 15. Wu Y., Tsukiyama S., Marek-Sadowska M. Graph based analysis of 2-D FPGA routing // IEEE Trans. Computer-Aided Design. 1996. Vol.15. № 1. P. 33–44.

УДК 655.027

ЧАСТОТНО-ГРАДАЦІЙНІ ХАРАКТЕРИСТИКИ ЦИФРОВОГО РАСТРУВАННЯ ЗА ПРИНЦИПОМ ПОШИРЕННЯ ПОХИБКИ*

Н.С. Стефанишина, М.В. Шовгенюк, В.О. Дудяк

Пропонується новий метод побудови частотно-градаційної характеристики для кількісної оцінки цифрового растрування зображень. На основі розробленої комп'ютерної програми проаналізовано декілька алгоритмів цифрового растрування за принципом поширення похибки й побудовано для них частотно-градаційні характеристики.

Предлагается новый метод построения частотно-градационной характеристики для количественной оценки цифрового растрирования изображений. На основе разработанной компьютерной программы проанализировано несколько алгоритмов цифрового растрирования по принципу распространения погрешности и построено для них частотно-градационные характеристики.

Основна ідея частотно-модульованого растрування – формування градаційної шкали шляхом задання певного значення просторової частоти для кожного конкретного рівня інтенсивності. Для реалізації її в сучасних технологіях цифрового растрування використовуються кілька базових алгоритмів. Найпростіший з них – поелементне порівняння цифрового зображення з матрицею критичних значень, так званий алгоритм „точка-в-точку". Раструючи таким чином, отримують елементи однакового розміру, але конфігурація растрових елементів містить періодичні компоненти, помітні як текстура. Складніший алгоритм заповнення фрактальної кривої передбачає створення елементів певного розміру і певної конфігурації для кожного окремого рівня інтенсивності без використання матриці порогових значень. При цьому створюється неперіодичний візерунок, який задається характером фрактальної кривої. Але алгоритми такого роду надто складні, мають низьку швидкодію та ще ряд недоліків, пов'язаних з особливостями фрактальних кривих [2].

Найбільш поширеними та універсальними є алгоритми, що базуються на принципі розповсюдження похибки. Результат опрацювання за таким алгоритмом – нерегулярний візерунок із растрових елементів однакового розміру. Хоч за швидкодією вони поступаються алгоритмам „точка-в-точку", але є не такими складними й громіздкими, як процес заповнення фрактальних кривих.

Загальний алгоритм поширення похибки описується у такій послідовності:

вибір неопрацьованого елемента півтонового зображення так званого піксела (від англ. picture);

порівняння значення обраного елемента з критичним значенням (найчастіше 127);

обчислення похибки;

надання елементу бінарного значення 0 чи 1 (біле чи чорне);

* Автори щиро вдячні Т.Є. Крохмальському та С.П. Глушаку за консультації й допомогу в розробленні комп'ютерних програм цифрового растрування зображення.