УДК 519.173:004.92

Т. М. Басюк
Національний університет “Львівська політехніка”,
кафедра інформаційних систем та мереж

ЗАБЕЗПЕЧЕННЯ ПРОЦЕСУ ВІЗУАЛІЗАЦІЇ ДАНИХ У СЕРЕДОВИЩІ ВІДКРИТИХ СИСТЕМ

Описано особливості проведення процесу відображення даних, представлених у матричній формі на площині. Проаналізовано відомі методи візуалізації даних та визначено відсутність механізмів із коректного відображення даних, представлених матрицями із забезпеченням критеріїв візуалізації. З огляду на те було запропоновано методики: відображення ідентифікаційних міток; моделювання міжвершинних відстаней та розташування вершин з погляду зручності сприйняття та оцінювання. Здійснено функціональну реалізацію системи згідно з профілями побудови відкритих гетерогенних систем.

Ключові слова: візуалізація, матриця, відкрита система, ідентифікаційна мітка, вершина графу, профіль.

The article describes the features of the process of displaying data presented in matrix form on the plane. Analysis of known methods of data visualization that showed a lack of mechanisms from the correct data displays presented by matrices with the provision of visualization criteria was conducted. Considering this fact, the following methods were proposed: the display of ID tags; modelling between vertex distance and the location of tops in terms of ease of perception and evaluation. Functional implementation of the system under construction profiles of open heterogeneous systems is carried out.

Key words: visualization, matrix, open system, ID tags, top graph, profile.

Вступ. Загальна постановка проблеми

Сучасні інформаційні системи візуалізації даних проектують в динамічних умовах, у яких критичною мірою є “час реакції” користувача. Зокрема, в системах медичної діагностики чи керування технологічними процесами зазначений параметр впливає на швидкість прийняття аргументованих рішень, а відтак, на якість результату. З огляду на це від якості отриманих зображень значною мірою залежить не лише час їх аналізу, але й правильність зроблених висновків, що накладає певні обмеження на їх відображення [1].

У процесі проектування систем візуалізації даних множину технологічних процесів подають переважно у вигляді двовимірної структури даних – графів. Проблема полягає в тому, що один і той самий математичний опис графу можна подати еквівалентними зображеннями, які є однаковими за структурою зв’язків, проте різними за наочністю, що по-різному сприймаються оператором [2]. Внаслідок аналізу виявилось, що зображення графів, які найсприятливіші для аналізу та оцінювання, повинні відповідати таким критеріям [3]: рівномірність розташування; мінімальна кількість взаємних перетинів дуг; неприпустимість перетину площини вершини графу; мінімальна площа зображення.

Що стосується технологічного забезпечення задачі візуалізації даних, то особливістю є тенденції до інтеграції, а саме функціональної інтеграції, відмінною рисою якої є об’єднання неоднорідних інформаційних ресурсів, що раніше функціонували автономно, в єдину інтегровану розподілену систему, яка обслуговує підрозділи й служби об'єкта [4]. Використання такого підходу сприяло появі нових напрямів проектування архітектури й структури прикладних програмних комплексів, що реалізують концепції відкритих інформаційних систем. Своєю чергою, зростаюча

складність таких систем вимагає застосування додаткових вимог до методів і засобів проектування. Проектуючи таку відкриту систему, необхідно знайти рішення, що дозволяє забезпечити взаємодію служб, які підтримуються різними операційними системами, що є надзвичайно складною задачею. З огляду на те актуальною є задача розроблення математичного забезпечення процесу візуалізації даних з підтриманням механізму відкритих систем.

Зв'язок висвітленої проблеми із важливими науковими та практичними завданнями

Розроблення математичного забезпечення процесу візуалізації даних є складною науково-практичною задачею, яка представляється у вигляді двох підзадач: розміщення вершин графу та відображення дуг між ними. При розташуванні вершин необхідно окремо виділити представлення особливих типів графів (лінійних, тора, дерева), зображення яких сприймаються у певному вигляді, тому саме так їх потрібно візуалізувати. Відображення дуг повинно здійснюватись згідно з вимогами оформлення та відповідати прийнятим критеріям відображення [3].

З огляду на те, обидві задачі є вкрай важливими, оскільки безпосередньо впливають як на якість кінцевих зображень, так і на необхідну функціональність технологічних засобів. Окреслена задача являє собою складний процес, під час якого необхідно застосовувати розширений математичний та алгоритмічний апарат. Зокрема, можна застосувати як методи аналізу матриць, так і алгоритми розташування чи пошуку найкоротших шляхів. Розв'язання цієї науково-практичної задачі дасть змогу уніфікувати процес візуалізації даних у гетерогенному середовищі та надасть засоби із адаптації відомих методів до задач побудови інтелектуальних систем у предметній галузі.

Аналіз останніх досліджень та публікацій

Роботи із проектування засобів відображення та візуалізації даних розпочались ще в період першого покоління розвитку електронно-обчислювальних машин [5]. З часом створювані методи та алгоритми удосконалювались, а застосовувані підходи видозмінювались залежно від наявних технологічних засобів. Сьогодні для проектування інтелектуальних систем відображення даних використовують такі форми представлення: графові моделі (відображення зв'язків між технологічними вузлами) [6], блок-схеми (опис алгоритму роботи системи) [7], UML діаграми (опис уніфікованого процесу розроблення програмного забезпечення) [8]. При цьому візуалізація як блок-схем алгоритмів, так і UML діаграм не викликає особливих труднощів та забезпечується відповідними середовищами, зокрема: Borland Together Architect, Very Hardware Description Language, IBM Rational Rose, але візуалізація графів принципово відрізняється. Оскільки початкові моделі, які необхідно відобразити (матричні масиви даних), не містять інформації щодо просторового розташування, це сприятиме отриманню множини еквівалентних зображень різної наочності. Сьогодні розроблено значний математико-алгоритмічний апарат візуалізації даних [9], зокрема відомі методи [2, 6, 9, 10], не призначені для візуалізації даних, представлених у вигляді матричних структур, оскільки: не враховують якості сприйняття (основний акцент приділяється мінімальній довжині дуг) та не враховують кінцевих розмірів зображень (виникають труднощі в процесі їх візуалізації без застосування опцій масштабування). Крім того, наявні методи не враховують особливостей функціонування в гетерогенних системах, оскільки не містять засобів із уніфікації пропонованих підходів до різних типів апаратно-програмних платформ [11, 12]. Отже, актуальною є задача розроблення математичного забезпечення процесу візуалізації даних представлених у вигляді графів згідно із стандартами відкритих систем.

Основні завдання дослідження та їх значення

Метою дослідження є розроблення математичного забезпечення процесу візуалізації даних, поданих у вигляді графів у середовищі відкритих систем. Проведене дослідження надасть засоби із побудови графів на площині згідно з відомими критеріями відображення [3].

Для досягнення поставленої мети необхідно вирішити такі основні завдання: проаналізувати відомі методи візуалізації даних; розробити методи перетворення матричного подання у графові структури із забезпеченням критеріїв відображення; розробити структуру відкритої інформаційної системи візуалізації даних. Результати виконаної роботи у своїй сукупності сприяють вирішенню

актуального завдання створення математичного забезпечення процесу візуалізації даних у середовищі відкритих систем.

Основні результати досліджень

Збільшення продуктивності комп'ютерних систем надає нові можливості з моделювання множини об'єктів, аналіз яких потребує опрацювання великих обсягів інформації. Сьогодні наявні апаратні засоби, які обробляють інформацію зі швидкістю декількох петабайтів за секунду на базі комп'ютерів з продуктивністю десятки у ГФлопс. З огляду на це виведення такого обсягу інформації вимагає можливості її перетворення у спеціалізовані графічні зображення, оскільки у багатьох випадках зміст стає зрозумілим лише при візуальному представленні. Оскільки перетворення числових даних у схематичні зображення – процес вкрай важкий, відображення отриманих результатів само по собі є областю складних і важливих наукових досліджень. Саме тому алгоритми, комп'ютерна архітектура і програмні засоби візуалізації постійно вдосконалюються для забезпечення можливості виконання детального аналізу отримуваних обсягів даних [7]. Загальну структуру процесу відображення в комп'ютерних засобах зображено на рис. 1.

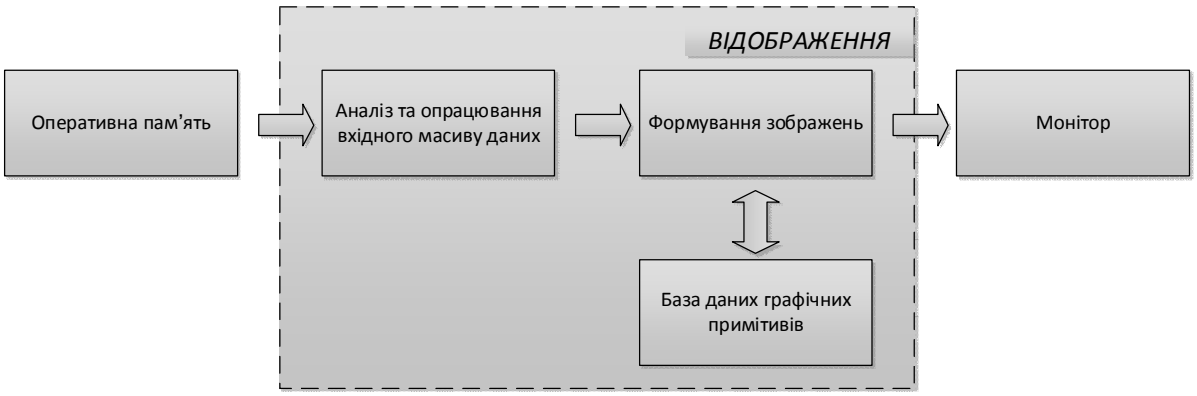

Рис. 1. Загальна структура процесу відображення

А саме інформацію, яка міститься в оперативній пам'яті комп'ютера і підлягає відображенню, спочатку аналізують для виявлення, наприклад, структур файлів даних, особливих точок, параметрів зображення тощо. Потім введені дані опрацьовуються й здійснюється власне процес візуалізації. Отже, відображення – це процес перетворення деякої множини початкових даних на масив точок зображення; при цьому від її роботи залежить якість сприйняття виведеної інформації.

Аналіз методів візуалізації

Візуалізація даних настільки широко увійшла в повсякденне життя суспільства, що сьогодні немає жодної галузі науки чи техніки, у якій не застосовуються механізми відображення. Що стосується досліджуваної задачі, то найближчою за змістом предметною областю є мікропроцесорна техніка, оскільки в ній візуалізація також передбачає два етапи: розташування вузлів та проведення зв'язків між ними. При цьому початковими умовами на першому етапі є множина вузлів A, розмір комутаційного простору X та схема з'єднань елементів L. Розв'язок отримують пошуком позиції окремого вузла з метою відображення всіх з'єднань. Задача розташування вузлів при цьому зводиться до визначення координат центрів елементів, за яких цільова функція набуває екстремальних значень та їх заокруглення до цілочислового значення координат.

Найпоширенішими є такі класи алгоритмів розташування вузлів: градієнтні, динамічних моделей, випадкового пошуку та евристичні. Градієнтні методи реалізують ідею розташування вузлів, при якій забезпечується мінімальна довжина з'єднань. Зазвичай їх застосовують за незначної кількості елементів (менше сорока), в іншому випадку елементи розташовують із значною нерівномірністю, що суперечить критеріям відображення [3]. В алгоритмах побудови динамічних моделей елементи представляються у вигляді точок, на які діють сили притягання та відштовхування, при цьому їх величина залежить від кількості зв'язків [2]. Результатом роботи

алгоритму є таке розташування, при якому дія зазначених сил зрівноважується. Недоліком алгоритмів є значна нерівномірність розташування на площині конструктиву. В алгоритмах випадкового пошуку елементи розміщуються на основі зв'язаності – кількості дуг відповідної пари вершин. Результатом роботи є множина зображень, еквівалентних за структурою зв'язків [9]. Застосування евристичних алгоритмів ґрунтується на використанні накопичених або апріорно закладених знань і є значно ефективнішим порівняно з методами повного перебору, проте знайдені розміщення не є зазвичай оптимальними варіантами, а належать до множини можливих [10].

Потім здійснюють процедуру відображення зв'язків між вузлами. Алгоритми, які застосовуються в мікропроцесорній техніці, поділяють на дві групи: хвильові та ортогональні. Перший тип побудований на використанні принципів Лі та враховує технологічну специфіку подальшого монтажу, а саме зв'язок здійснюється скануванням всього поля конструктиву та знаходженням найкоротшої віддалі [2]. Ортогональні алгоритми є більш швидкодіючими за хвильові та застосовуються в процесі проектування друкованих плат із металізованими посадковими місцями, проте особливістю є одержання значної кількості переходів між шарами та множини паралельних з'єднань [13], що не забезпечують прийняті критерії відображення. Окрім цих алгоритмів, з метою візуалізації у двовимірних реалізаціях застосовують алгоритми реалізації відповідних цільових функцій. До них належать алгоритми [2]: Дейкстри, Флойда, Йена, Краскала, Прима. Ці алгоритми забезпечують обхід вершин, проте не дають засобів із сплайнового відображення дуг [14], що є значним недоліком з погляду подальшого оцінювання.

Результати аналізу свідчать, що відомі методи та алгоритми візуалізації не повною мірою задовольняють критерії відображення даних. Тому виникає потреба у створенні нового математичного забезпечення, що надасть необхідний апарат для здійснення процесу візуалізації даних та проектування відповідного програмного забезпечення, що функціонуватиме в гетерогенному програмно-апаратному середовищі.

Відображення ідентифікаційних міток у процесі візуалізації

Перед візуалізацією даних, представлених у вигляді матриць, необхідно провести множину досліджень, пов'язаних із визначенням майбутніх розмірів вершин, достатніх для розташування ідентифікаційних міток, моделювання міжвершинних відстаней та максимальних кількостей вершин графу, які можна розмістити в одному ярусі. Початковою інформацією при цьому є розмір основного шрифту та діагональ екрана відображення. Відомо [14], що текст в графічних редакторах розміщується в умовному прямокутнику, а його координати (X_M,Y_M) визначаються положенням лівого верхнього кута (рис. 2).

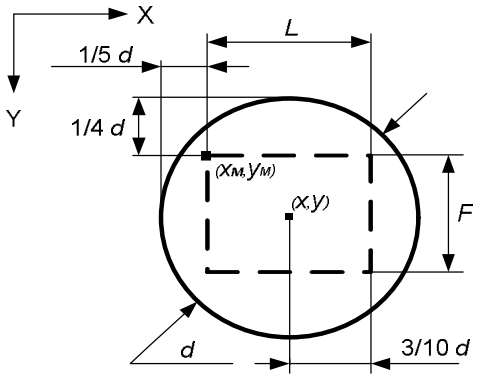

Рис. 2. Знаходження координат розташування ідентифікаційних міток

При цьому висота текстового прямокутника F – це розмір шрифту, який задається в пунктах (*пт*). З огляду на те ширина L цього прямокутника залежить від кількості символів, що складають ідентифікаційну мітку. Ширина одного і того самого символу однакового розміру може бути різною залежно від типу шрифту та з врахуванням міжсимвольної відстані може знаходитися в

межах *0,3÷0,5F*. Для подальших розрахунків приймемо, що ширина символу дорівнює *0,4F*. Найчастіше як ідентифікаційну мітку застосовують порядковий номер вершини, тоді кількість символів визначатиметься виразом:

$$m = Int(\lg N + 1), \tag{1}$$

де *N* – кількість вершин графу; *Int* – ціла частина числа.

Визначимо значення діаметра вершини як деяку функцію від розміру шрифту $d = j(F)$. З огляду на те при кількості символів мітки $m < 3$ (для графу з кількістю вершин до 99) ширина текстового прямокутника буде меншою *F*, а при кількості $m = 3$ (до 999 вершин графу) ширина дорівнюватиме максимально допустимій *1,2F*, тобто буде не більшою за $\frac{6}{10}d$. У цьому випадку діаметр вершини визначають за виразом:

$$\frac{1}{2}d = \frac{1}{72}E_д\ F\ K_в, \tag{2}$$

де *Ед* – значення дюйма як одиниці діагоналі пристрою відображення; *F* – розмір тексту мітки; *Кв* – коефіцієнт, який враховує особливості пристрою відображення; $\frac{1}{72}$ – коефіцієнт переходу між значенням величини пункту та міліметрами.

Якщо $m > 3$, тобто кількість вершин дорівнює тисячі і більше, або коли мітки є відмінні від чисел, діаметр вершини визначатиметься залежно від ширини текстового блоку *L*:

$$\frac{6}{10}d = \frac{1}{72}E_д\ K_в L. \tag{3}$$

При цьому ширину текстового блоку визначають так:

$$L = \frac{4}{10}mF. \tag{4}$$

Визначимо залежності координат ідентифікаційної мітки як функції від координат центру вершини, тобто $X_M = y_1(X)$, а $Y_M = y_2(Y)$. Висота ідентифікаційної мітки не залежить від кількості символів та дорівнює розміру шрифту:

$$Y_M = Y - \frac{1}{144}FE_д K_в. \tag{5}$$

Горизонтальна координата X_M залежить від ширини мітки і визначиться як:

$$X_M = X - \frac{L}{2}. \tag{6}$$

Підставивши вираз (4) у (6) з врахуванням коефіцієнта $\frac{1}{72}$, отримаємо:

$$X_M = X - \frac{1}{360}E_д mFK_в. \tag{7}$$

Вираз $\frac{1}{360}E_д FK_в$ є певною константою, яку можна визначити ще до початку візуалізації. У процесі візуалізації на етапі виведення ідентифікаційних міток її множать на кількість символів кожної вершини.

Моделювання міжвершинних відстаней на площині екрана

Після визначення параметрів вершин графу необхідно визначити значення міжвершинних відстаней. Передбачено, що місця розташування вершин знаходитимуться на однаковій відстані, утворюючи тим самим певну канву. Розташовувати вершини пропонується зі сталим кроком, що дорівнює 3d між центрами вершин. При цьому максимальна кількість вершин, що може бути розташована на екрані монітора, залежить від розміру його діагоналі, а саме від видимої діагоналі [14].

$$D_в = D_м * k, \tag{8}$$

де *Dв, Dм* – видима та регламентована діагоналі монітора; *k* – коефіцієнт переходу між *Dм* та *Dв*.

Для моніторів на електронно-променевій трубці, які найчастіше застосовуються в науково-дослідних структурах, зазначений коефіцієнт визначають за виразом:

$$k = \frac{D_м - 0{,}8}{D_м}. \tag{9}$$

У зазначених пристроях виведення відношення ширини до висоти становить три до чотирьох, отже, універсальний вираз для визначення ширини та висоти екрана матиме вигляд:

$$\begin{aligned} W &= 0{,}8\, D_в\, E_д, \\ H &= 0{,}6\, D_в\, E_д, \end{aligned} \tag{10}$$

де W – видима ширина екрана; H – видима висота екрана.

Робоча площина екрана є меншою (рис. 3) за регламентовану, оскільки частину вікна займають елементи програми (поле команд, скролінги, піктографічне меню тощо).

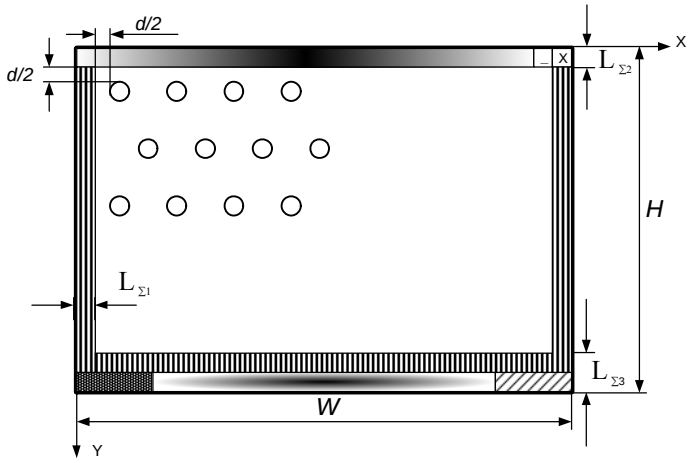

Рис. 3. Розташування вершин на екрані монітора

Координати центрів вершин визначаємо як $X_i = 2{,}5d + 3dg$ для непарних рядків (*j*) та $X_i = d + 3dg$ для парних. Значення ординати знаходять за виразом: $Y_i = d + 3dj$. Отже, максимальну кількість вершин в рядку та кількість рядків визначатимуть за виразами (11):

$$\begin{aligned} N_r^{'} &= Int\left(\left(W - 2L_{\sum 1} - 2d\right)/3d + 1\right), \\ N_r^{''} &= Int\left(\left(W - 2L_{\sum 1} - 3{,}5d\right)/3d + 1\right), \\ N_c &= Int\left(\left(H - L_{\sum 2} - L_{\sum 3} - 2d\right)/3d + 1\right). \end{aligned} \tag{11}$$

Якщо візуалізоване зображення графу за більшої кількості його вершин не поміщається в робочому полі вікна програми, то для його перегляду застосовуватимуть смуги прокручування. Таким чином, можна зробити висновок про те, що на екрані монітора з погляду дотримання критеріїв візуалізації доцільно застосовувати коло з діаметром від 7 до 10 мм, що забезпечить наочність візуалізованих графів у масштабі один до одного при розмірі тексту від 10 до 14 пт.

Розташування вершин графу на площині

Відповідно до наведеної методики знаходження міжвершинних відстаней, розташування вершин графа на площині передбачає такі основні етапи: перетворення початкової матриці у ярусно-паралельну форму; визначення початкової вершини; знаходження списку вершин та відповідних координат та їх розташування за рівнями графу [15].

На першому кроці знаходять матрицю рядка перетворенням її на ярусно-паралельну форму [16]. При цьому значимі елементи сформованої матриці належать вершинам графу, які формують певний ярус, а кількість отриманих матриць-рядків визначає кількість ярусів. На другому кроці

визначають початкову вершину, яка зокрема в графі „дерево” є лише однією. Розташовують її в першому рядку. Номер позиції першої вершини визначають як: $g_0 = N_r/2$. На третьому кроці визначають послідовність розміщення вершин кожного ярусу за рядками, а номер місця в рядку кожної наступної вершини визначають за залежністю: $g_s = (-1)^s \cdot S + g_{s-1}$. Потім визначають координати вершин ярусу на площині екрана.

Проілюструємо роботу описаного розташування вершин графу “дерево” на площині при розмірі шрифта 14 пт, що заданий такою матрицею:

$$A = \begin{vmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1\\
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{vmatrix}.$$

Перетворення матриці *A* на ярусно-паралельну форму сприятиме утворенню таких матриць:

$$D_1 = |\ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ |;$$
$$D_2 = |\ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ |;$$
$$D_3 = |\ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ |;$$
$$D_4 = |\ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1\ |;$$
$$D_5 = |\ 1\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ |;$$

Початковою є третя вершина “дерева”, що міститься в першому ярусі. Визначимо кількість зв’язків кожної вершини графу множенням початкової матриці на одиничний вектор-стовпець, отримуючи матрицю рядок $|\ 0\ \ 0\ \ 2\ \ 0\ \ 0\ \ 1\ \ 3\ \ 0\ \ 2\ \ 0\ \ 0\ \ 0\ \ 3\ \ 2\ \ 2\ \ 1\ \ 0\ |$.

З виразу (11) знайдемо значення: $N_r^{'} = 10; N_r^{''} = 9; N_c = 7; d = 10 мм$ та визначимо номери місць та координати вершин кожного ярусу. Перший ярус графу ($N_1 = 1$) розташовується в першому рядку ($j=0$). Оскільки $N_1 < N_r^{'}$, то номер місця третьої вершини дорівнюватиме: $g = 4$, а її координати будуть такі:

$$X_3 = d + 3dg = 10 + 3 \cdot 10 \cdot 4 = 130 мм,$$
$$Y_3 = d + 3dg = 10 + 3 \cdot 10 \cdot 0 = 10 мм.$$

Другий ярус містить дві вершини “6” та ”7”. ($N_2 = 2,\ \ N_2 < N_r^{''}$). Всі вершини другого ярусу розташуються в другому рядку (j=1). Спочатку буде розташована вершина за номером “7”, а номер її місця і координати матимуть такі значення (j=1, i=4):

$$X_7 = 2{,}5d + 3di = 25 + 3 \cdot 10 \cdot 4 = 145мм,$$
$$Y_7 = d + 3dj = 10 + 3 \cdot 10 \cdot 1 = 40мм.$$

Наступною в рядку буде вершина за номером “6”, що матиме такі значення: ($j=1,\ i=3$):

$$X_6 = 2{,}5d + 3di = 25 + 3 \cdot 10 \cdot 3 = 115мм,$$
$$Y_6 = d + 3dj = 10 + 3 \cdot 10 \cdot 1 = 40мм.$$

Аналогічно визначають координати решти ярусів графу, в яких кількість вершин менша за кількість можливих місць вершин у рядках, а тому кожен ярус займатиме по одному рядку:

Третій ярус (j=2):

$$X_{13} = 130мм,\ X_{15} = 100мм,\ X_{16} = 160мм,\ X_4 = 70мм,$$
$$Y_{13} = 70мм,\quad Y_{15} = 70мм,\quad Y_{16} = 70мм,\quad Y_4 = 70мм.$$

Четвертий ярус (j=3):

$$X_9 = 145мм,\ X_{14} = 115мм,\ X_8 = 175мм,$$
$$Y_9 = 100мм,\ Y_{14} = 100мм,\ Y_8 = 100мм,$$
$$X_{10} = 85мм,\ X_{12} = 205мм,\ X_{17} = 55мм,$$
$$Y_{10} = 100мм,\ Y_{12} = 100мм,\ Y_{17} = 100мм.$$

П’ятий ярус (j=4):

$$X_1 = 130мм,\ X_2 = 10мм,\ X_5 = 160мм,\ X_{11} = 70мм,$$
$$Y_1 = 130мм,\ Y_2 = 130мм,\ Y_5 = 130мм,\ Y_{11} = 130мм.$$

Оскільки кількість ярусів графу не перевищує кількості рядків, то його зображення розміщується на екрані монітора без застосування смуг прокручування. На рис. 4 наведено результат розташування вершин.

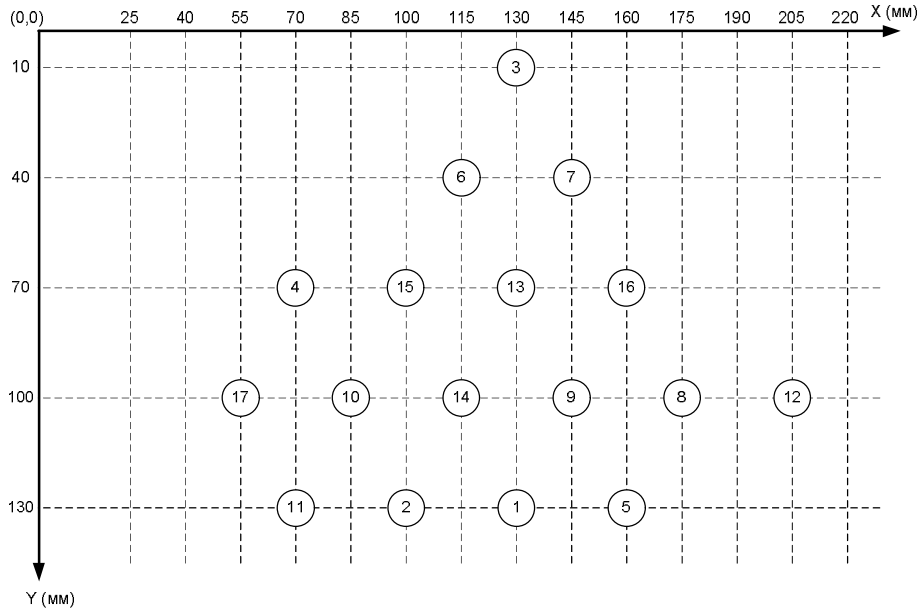

Рис. 4. Розташування вершин графу на екрані монітора

Відображення зв’язків між вершинами графу

Після розташування вершин здійснюють процедуру проведення зв’язків із забезпеченням критеріїв відображення. А саме спершу моделюють можливість проведення дуг у вигляді прямих та виявляють ті, що взаємно перетинаються. Потім із застосуванням модифікованого методу парних перестановок здійснюють процедуру зменшення кількості перетинів. Блок-схему алгоритму для графів типу “дерево” наведено на рис. 5.

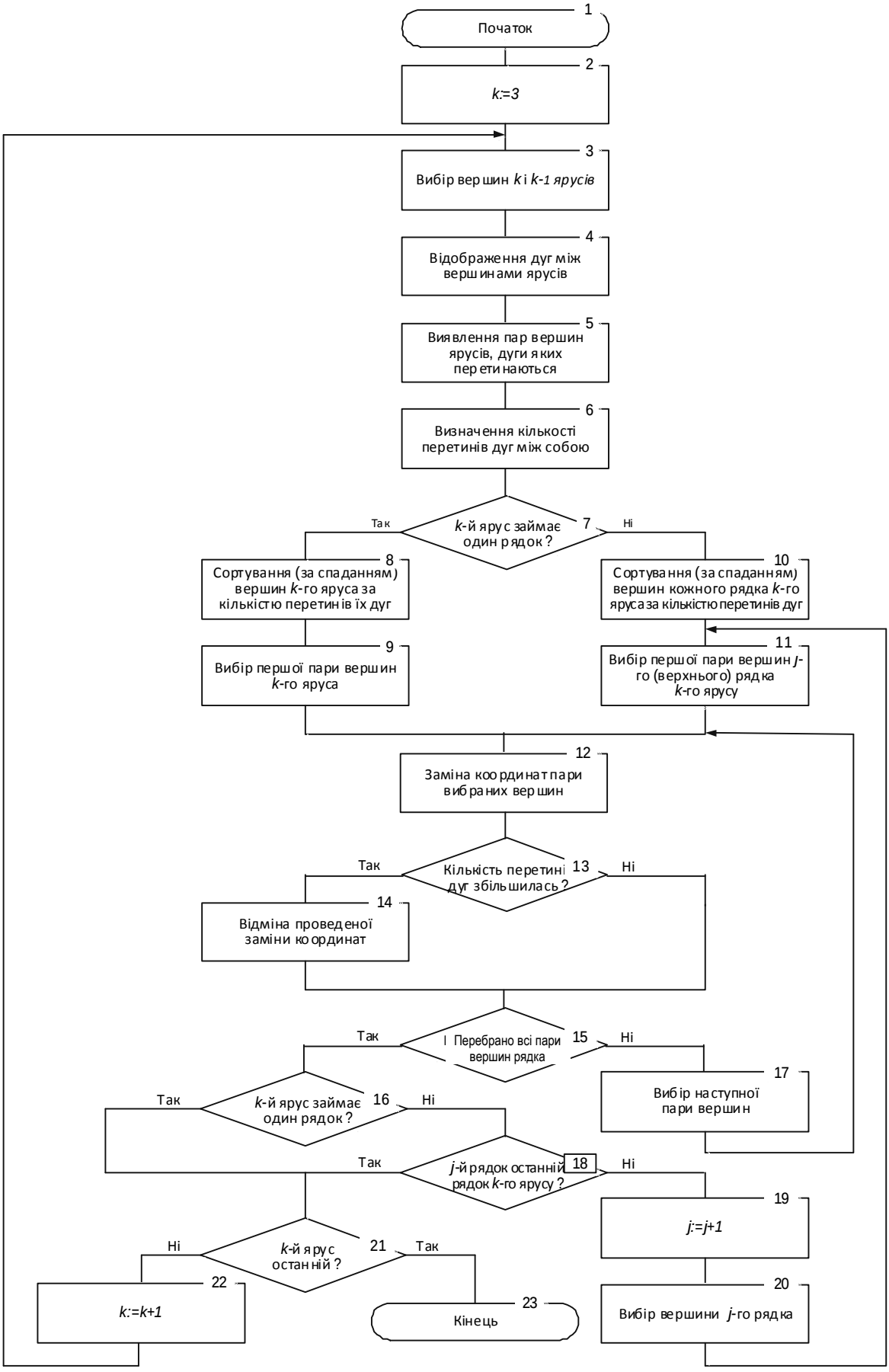

Рис. 5. Блок-схема алгоритму парних перестановок для графів "дерево"

Проілюструємо роботу цього алгоритму парних перестановок на конкретному прикладі, а саме розглянемо зменшення кількості перетинів дуг для графу „дерево”, вершини всіх ярусів якого займають по одному рядку (рис. 4), а координати вершин визначено згідно із вищеописаними залежностями. Зокрема у графі (рис. 6) є три вершини „4”, „15” та „13”, дуги яких перетинаються.

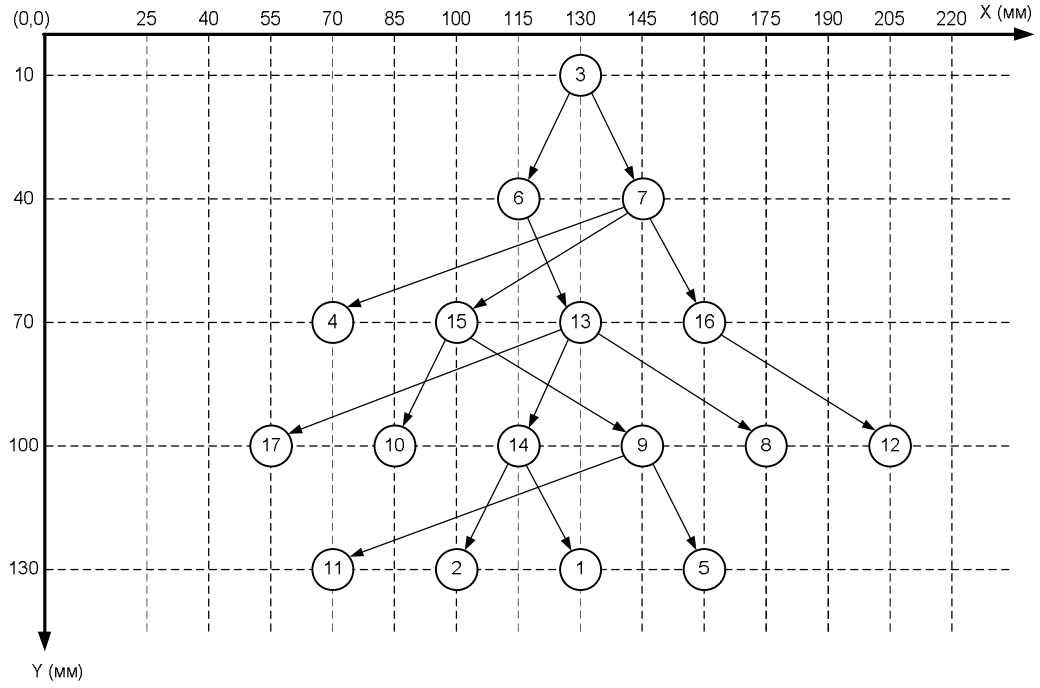

Рис. 6. Приклад відображення графу „дерево” після проведення дуг

Дуга вершини за номером „13” має два перетини, а решта – по одному. Тоді згідно з алгоритмом вибираємо вершини за номерами „13” та „4” і замінюємо їхні координати. У результаті отримаємо граф, зображений на рис. 7.

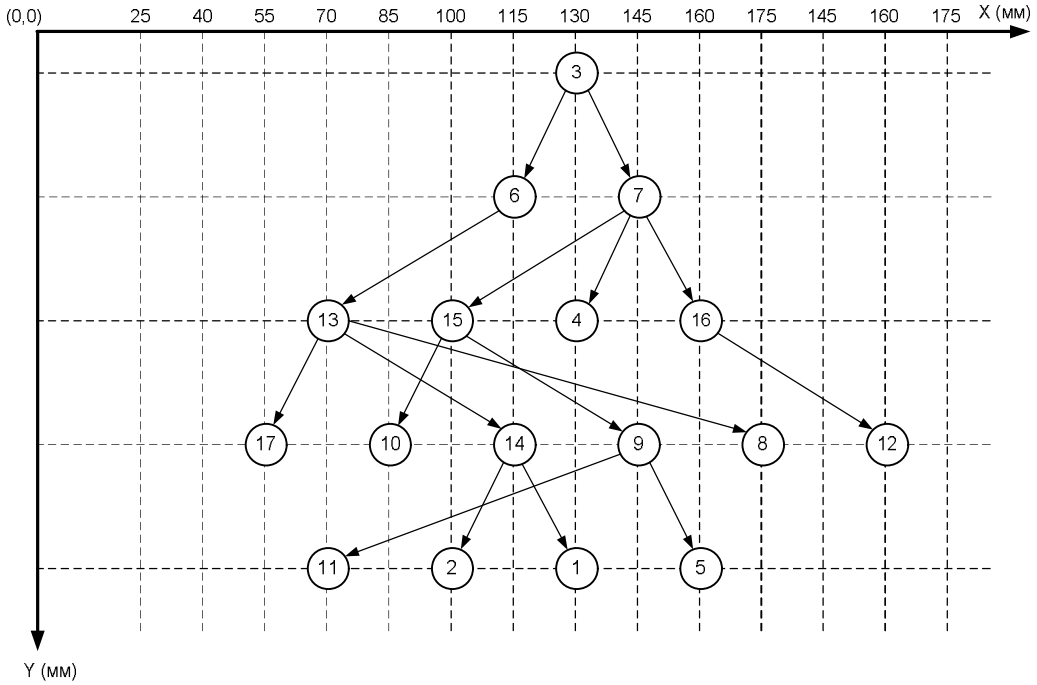

Рис. 7. Приклад відображення графу „дерево”
після заміни вершин „4” та „13” місцями

Оскільки перетинів у цьому ярусі більше немає, то переходимо до вершин четвертого ярусу. Серед них є вершини „8”, „9”, „10” і „14”, дуги яких перетинаються, зокрема дуги вершин за номерами „8” та „10” мають по два перетини, а решта вершин – по одному. Тоді згідно з алгоритмом вибираємо вершину за номером „8” та шукаємо відповідні перетини. З рисунка видно, що вона перетинає дуги вершин за номерами „9” та „10”. Першою парою вершин, координати місць яких будемо змінювати, будуть вершини „8” та „9”. В результаті отримаємо граф, що зображений на рис. 8.

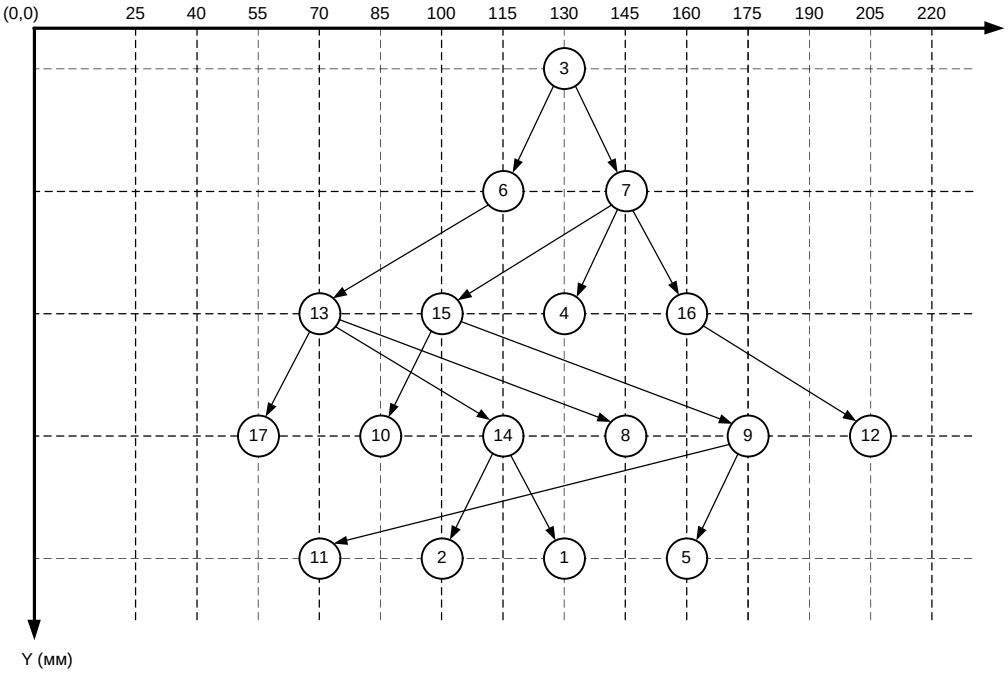

Рис. 8. Приклад відображення графу „дерево” після заміни вершин „8” та „9” місцями

Як видно з рис.8, кількість перетинів дуг після перестановки не збільшилась, а, навпаки, зменшилась. Аналогічно переставляють вершини за номерами „8” та „10”. Результат перестановки зображено на рис. 9.

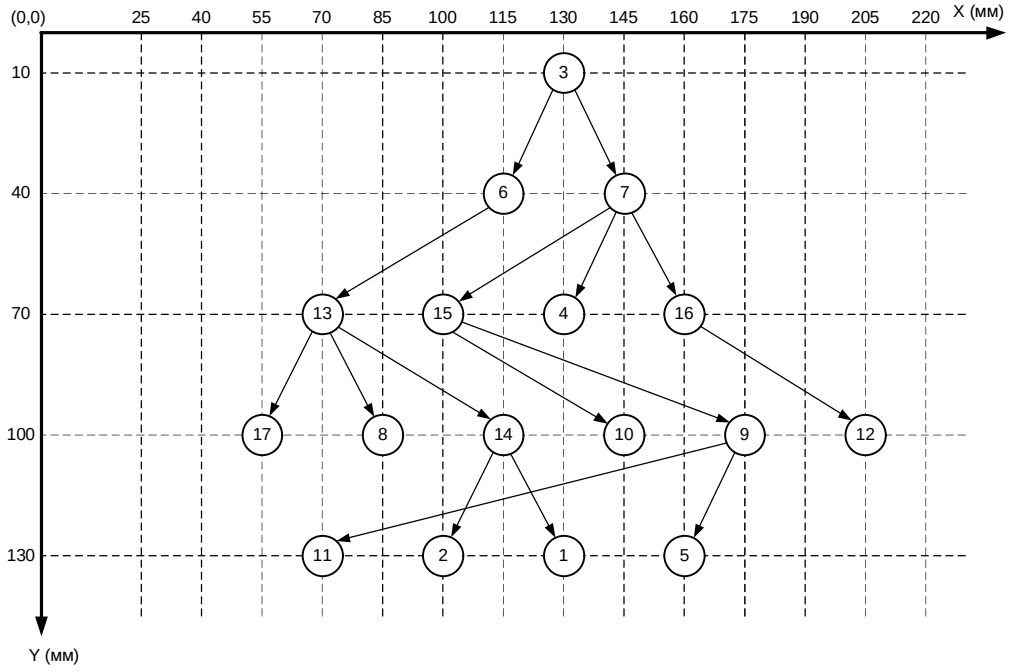

Рис. 9. Приклад зображення графу „дерево” після переміни місць вершин „8” та „10” місцями

Між вершинами третього та четвертого ярусів після парних перестановок перетинів дуг немає. Серед вершин останнього ярусу (рис. 9) є три вершини – „1”, „2” і „11”, дуги яких перетинаються, причому дуга вершини за номером „11” має два перетини, а дуги інших вершин – по одному. Вибираємо для перестановок вершини за номерами „11” та „1”. Після чергової заміни отримаємо граф зображений на рис. 10.

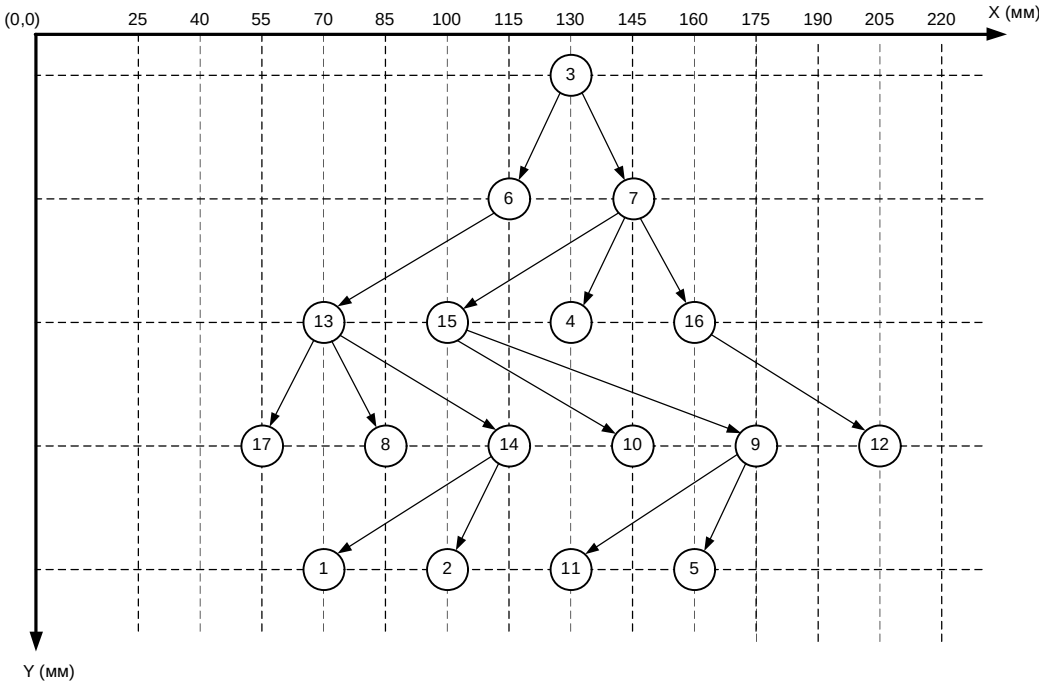

Рис. 10. Приклад відображення графу „дерево”
після проведення всіх парних перестановок

Як видно з утвореного зображення (рис. 10), проведена візуалізація дала змогу отримати зображення з відсутністю міждугових перетинань відповідно до сформованих критеріїв відображення.

Проектування профілів відкритої системи візуалізації даних

Наступним етапом дослідження стала побудова програмного забезпечення візуалізації даних. При цьому основний акцент стосувався розроблення кросплатформеного додатка, що передбачав необхідність формування профілів відкритої системи, під якими розуміють сукупність базових стандартів, виокремити які необхідно для реалізації заданих функцій системи. Особливістю досліджуваної задачі є те, що при побудові профілів необхідно враховувати певну суперечливість умов, у яких відбуваються процеси стандартизації інформаційних технологій. Визначення номенклатури профілів, які необхідно застосовувати для систем візуалізації даних, пов’язані з вибором таких рішень побудови, які забезпечують можливість заміни окремих її компонентів, не торкаючись інших функціональних частин [17].

З огляду на це необхідно розробити структуру профілів інформаційної системи візуалізації даних, яка б забезпечила можливість побудови зазначеної системи відповідно до стандартів відкритих систем та забезпечити можливу її модифікацію чи подальше вдосконалення із збереженням функціональної структури. Номенклатура побудови профілів відкритої системи візуалізації даних тісно пов'язана з декомпозицією структури цієї системи, що дає змогу її подилити на взаємодіючі підсистеми й компоненти [12]. При цьому для кожного компонента, що виділяється в структурі системи, визначають склад функцій, які він виконує, та взаємозв’язок з іншими компонентами. Поділ системи на взаємодіючі частини представляє ієрархічну залежність відповідно до декомпозиції за принципом “згори донизу” та має багаторівневу структуру, що відповідає ієрархії профілів [18].

Система візуалізації даних в загальному випадку як відкрита система поділяється на прикладне програмне забезпечення, що реалізує функції основного призначення системи, та середовище інформаційної системи, яке надає прикладним програмам необхідні сервіси й послуги. На всіх стадіях життєвого циклу проектування системи візуалізації даних вибираються, а потім застосовуються такі функціональні профілі: профіль прикладного програмного забезпечення, профіль середовища системи візуалізації, профіль захисту інформації, профіль життєвого циклу та профіль інфраструктури проекту.

Прикладне програмне забезпечення цієї системи є проблемно-орієнтованим і визначає основні її функції. Функціональні профілі повинні містити базові стандарти, а при їх застосуванні необхідно їх узгоджувати. Необхідність такого узгодження виникає зокрема при застосуванні стандартизованих API інтерфейсів, зокрема інтерфейсів додатків із середовищем їх функціонування, інтерфейсів додатків із засобами захисту інформації [18]. При узгодженні функціональних профілів уточнюються профілі середовища системи візуалізації та профілі інструментальних засобів створення, супроводу й розвитку прикладного програмного забезпечення. Використання функціональних профілів є неможливим без впровадження технологічних профілів: профілю життєвого циклу прикладних програмних засобів та профілю інфраструктури забезпечення проекту. Схема взаємозв'язку функціональних та допоміжних профілів, що підтримують створення, супровід та розвиток інформаційної системи візуалізації даних, наведено на рис. 11.

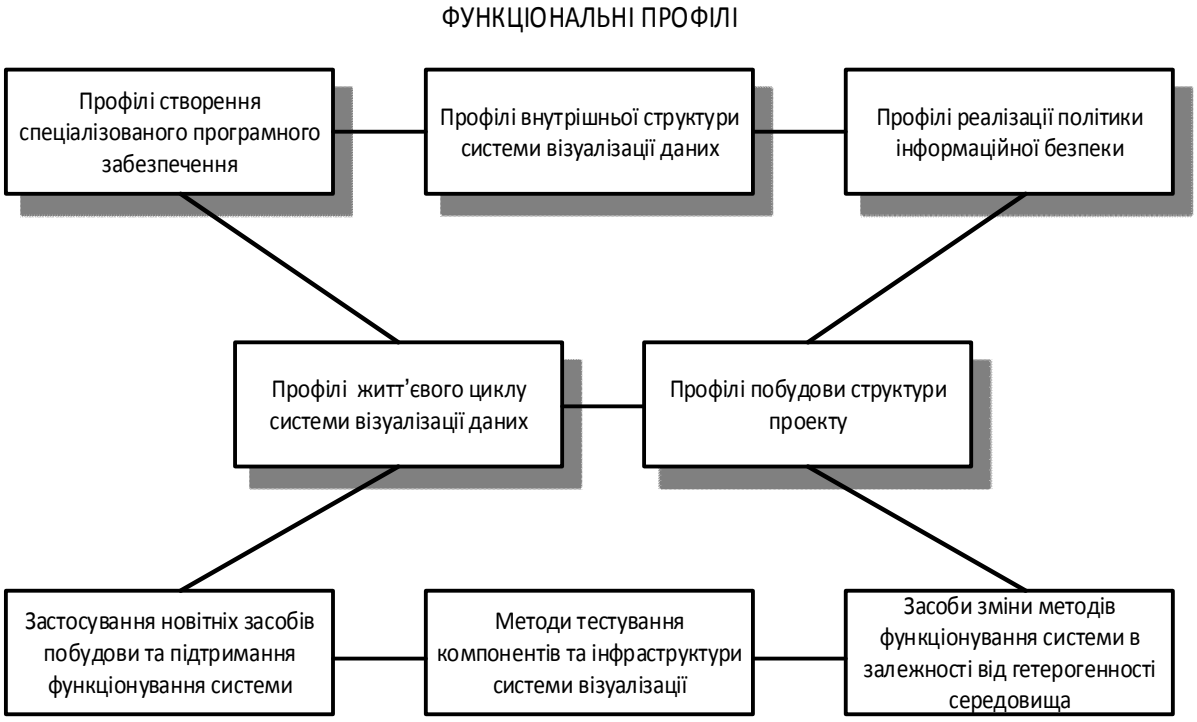

Рис. 11. Структура профілів системи візуалізації даних

Функціональні профілі системи візуалізації складаються із профілів компонентів, що реалізують прикладні функції або функції середовища. Деталізація функціональних профілів здійснюється у міру декомпозиції структури системи на складові під час її проектування. Застосування функціональних профілів полягає у виконанні таких робіт [19]:

– відповідність профілям програмних та апаратних засобів – здійснюється узгодження можливості взаємодії між програмними та апаратними платформами та здійснюється вибір технології проектування системи візуалізації даних;

– конструювання й проектування прикладного програмного забезпечення системи візуалізації даних відповідно до обраних профілів із використанням стандартизованих інтерфейсів передавання інформації;

– розроблення функціональних та структурних методів тестування як всієї системи, так і її компонентів на відповідність функціональним профілям;

– складання створених компонентів системи з використанням функціональних профілів побудови відкритих інформаційних систем.

Нормативні документи, що регламентують життєвий цикл системи візуалізації даних та її профілів, або задаються директивно в технічному завданні на створення системи, або вибираються розробником залежно від характеристик проекту. Зазначені документи, адаптовані й конкретизовані з врахуванням характеристик проекту й умов розроблення, визначають профіль життєвого циклу проектованої системи. В цьому профілі враховують набір етапів, робіт й операцій, пов'язаних з розробленням й застосуванням профілів, що специфікують прийняті проектні рішення. При цьому важливим моментом є ітераційність характеру формування та ведення профілів, пов'язаний з ітераціями процесів проектування й супроводу системи. Профіль життєвого циклу визначає всі стадії від створення до супроводу й розвитку системи, а також всі основні й підтримувальні процеси, які виконуються протягом життєвого циклу. Міжнародні стандарти, що регламентують життєвий цикл інформаційної системи візуалізації даних, сьогодні не є прийнятими. Тому пропонується для реалізації зазначеного класу профілів використовувати стандарт ISO 12207:1995 “Інформаційні технології. Процеси життєвого циклу програмного забезпечення” [17]. Такий підхід є виправданий, оскільки програмне забезпечення становить більшу частину вартості створення системи, а тривалість життєвого циклу програмного забезпечення фактично визначає тривалість життєвого циклу всієї системи.

Профіль середовища системи візуалізації визначає архітектуру її побудови відповідно до моделі DCE (Distributed Computing Environment), що підтримується консорціумом OSF (Open Software Foundation). Декомпозиція структури середовища функціонування, що виконується на стадії проектування, дозволяє деталізувати профіль середовища за функціональними областями еталонної моделі OSE/RM: графічного інтерфейсу користувача, об’єктно-орієнтованих систем управління базами даних, операційних систем.

Профіль захисту інформації забезпечує реалізацію політики інформаційної безпеки, що впроваджується відповідно до технічного завдання. Його побудова пов'язана з визначенням компонентів системи, які відповідають за сервіси й функції захисту інформації. Функціональна частина захисту інформації містить функції захисту, що реалізуються різними компонентами системи: операційною системою; прикладним програмним забезпеченням та полягає у створенні захищених інтерфейсів передавання даних та реалізації засобів авторизації.

Профіль інструментальних засобів визначає вибір методології й технології створення, супроводу й розвитку системи візуалізації даних. У цьому профілі описують обрану методологію проектування та наводять прийняті проектні рішення. Функціональна частина профілю інструментальних засобів охоплює функції централізованого керування й адміністрування, пов'язані з: контролем продуктивності й коректності функціонування системи, керуванням конфігурацією прикладного програмного забезпечення та доступом користувачів, налаштуванням додатків у зв'язку зі змінами прикладних функцій.

Структура відкритої системи візуалізації даних

Структура відкритої системи візуалізації даних (рис. 12) складається з двох взаємодіючих частин: функціональної та об’єктної. Функціональна частина містить прикладні програми, які реалізують алгоритми відображення даних, а об’єктна частина забезпечує їх виконання. Зазначена структура відкритої інформаційної системи дає змогу визначити інтерфейси та протоколи взаємодії додатками як між в межах однієї системи, так і між програмами двох чи більше взаємодіючих систем. Вона враховує той факт, що будь-яка інформаційна система може взаємодіяти з такими сутностями: з користувачем та зовнішнім середовищем. Взаємодія із зовнішнім середовищем

реалізується через сервіси віртуального терміналу, віддаленого виконання команд, передавання файлів та віддалену аунтифікацію. Взаємодія з користувачем реалізується за допомогою сервісів командного та графічного інтерфейсів.

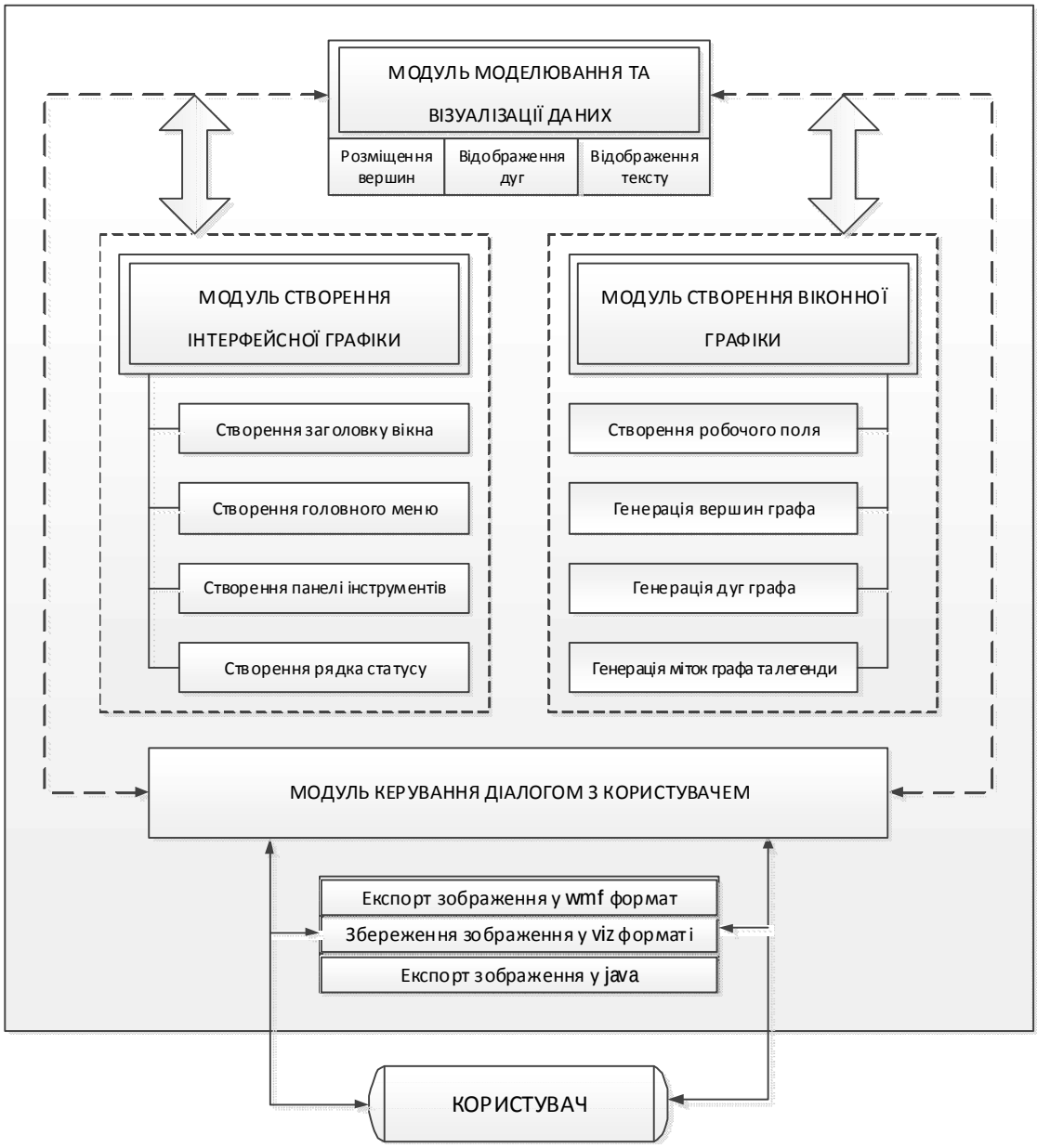

Рис. 12. Структура програми візуалізації даних

Взаємодія із зовнішнім середовищем реалізується з допомогою групи інтерфейсів (EEI – External Environment Interface), які визначають функціональну сумісність додатків з системами та реалізують такі сервіси: інтерфейс людино-машинної взаємодії та інформаційний обмін із зовнішнім середовищем. З огляду на це вся сукупність модулів, які складають прикладну програму, поділяється на дві частини: засоби інтерфейсу та процедури обслуговування процесу візуалізації графів. Стандартні засоби інтерфейсу згруповано у модулі створення інтерфейсної графіки, який складається з компонентів із створення: заголовка вікна, головного меню, панелі інструментів та рядка статусу. Окремим блоком є модуль створення віконної графіки, який містить програмні компоненти із створення робочого поля, формування вершин і дуг графу та компоненти для візуалізації ідентифікаційних міток та створення легенди.

Модуль моделювання та візуалізації зображення містить дані про розроблені способи розташування вершин, відображення дуг між ними та відображення ідентифікаційних міток.

Взаємодія з користувачем здійснюється за допомогою модуля керування діалогом з користувачем, застосовуючи зумовлені подіями методи їх опрацювання. На підставі конкретної події модуль керування діалогом з користувачем аналізує наявні процедури та функції, результатом чого є відображення на екрані сформованого зображення графу.

Висновки та перспективи подальших розвідок

У результаті проведеного дослідження проаналізовано відомі методи візуалізації, що показало відсутність засобів із коректного відображення даних, представлених матрицями. З огляду на це було запропоновано методики: відображення ідентифікаційних міток; моделювання міжвершинних відстаней на площині екрана та розташування вершин графу за критеріями відображення. Практичним втіленням роботи стала побудова структури системи візуалізації даних згідно із стандартами відкритих систем.

Зазначені особливості реалізації структури відкритої системи візуалізації даних передбачають значну надлишковість апаратної та програмної складових (надмірність системних ресурсів) порівняно з мінімальними ресурсами закритих, монолітних систем. Проте цей надлишок компенсується економією витрат на проектування й програмування інформаційної системи у випадку подальшого її використання в гетерогенному середовищі.

Подальші дослідження будуть скеровані на верифікацію розроблених модулів прикладної системи та їх адаптацію до функціонування в гетерогенному середовищі, що надасть засоби із формування на їх основі програмного забезпечення проміжкового рівня.

1. Хорн Р. Матричный анализ. / Р. Хорн, Ч. Джонсон. – М.: Мир, 2009. – 655 с. 2. Касьянов В. Н. Графы в программировании: обработка, визуализация и применение / В. Н. Касьянов, В. А. Евстегнеев. – СПб.: БХВ, 2010. – 1240 с. 3. Басюк Т. М. Критерії відображення графів в процесі візуалізації / Т. М. Басюк // Наукові записки УАД. – Львів: Українська академія друкарства. – 2004. – №. 7. –С. 60-63. 4. Лезер Н. Архитектура открытых распределенных систем / Н. Лезер. – К.: Бином, 2006. – 670с. 5. Зайцев С. С. Сервис открытых информационно-вычислительных сетей / С. С. Зайцев – М.: Радио и связь, 2010. – 235 с. 6. Годич О. В. Візуалізація даних, кластеризованих динамічно-інтервальною самоорганізовною картою / О. В. Годич, Т. П. Мазепа // Вісник Нац. ун-ту "Львівська політехніка". – № 743 : Інформаційні системи та мережі. – 2012. – С. 64–73. 7. Грицик В. В. Оцінка якості зображення / В. В. Грицик, В. В. Грицик // Вісник Нац. ун-ту "Львівська політехніка". – № 783 : Інформаційні системи та мережі. – 2014. – С. 82–92. 8. Козлов В. А. Открытые информационные системы / В. А. Козлов. – М.: Финансы и статистика, 2008. – 340 с. 9. Diestel R. Graph Theory. / R. Diestel – NY.: Springer-Verlag, 2005. – 422p. 10. Muller-Hannenmann M. Drawing trees, series-parallel digraphs, and lattices / M. Muller-Hannenmann – USA: Wiley Publishing, 2009. – 240p. 11. Lin X. Analysis of algorithms for drawing graphs / X. Lin – Dep. of Comput. Sci. Univ. of Queensland, – 2002. P.24. 12. Ананьин В. А. Корпоративные стандарты при разработке открытых систем / В. А. Ананьин – М.: Диалектика, 2005. – 670с. 13. Басюк Т. М. Аналіз та класифікація методів візуалізації / Т. М. Басюк // Поліграфія і видавнича справа. – Львів: Українська академія друкарства. – 2003. – №. 40. – С. 109–114. 14. Евченко А. OpenGL и DirectX: программирование графики./ А. Евченко – СПб.:Питер, 2012. – 414 с. 15. Басюк Т. М. Метод розміщення вершин графа в процесі візуалізації / Т. М. Басюк // Вісник Нац. ун-ту „Львівська політехніка". – 2004. – № 519. – С.3–10. 16. Дунець Р. Б. Структура програми перетворення графів у ярусно-паралельну форму / Р.Б. Дунець, Т.М. Басюк // Комп'ютерні технології друкарства. – Львів: Українська академія друкарства. – 2002. – №. 7. – С. 97–102. 17. Щербо В. К. Функциональные стандарты в открытых системах / В. К. Щербо – М.: МЦНТИ, 2007. – 640 с. 18. Бойченко А. К. Обобщенная модель открытых информационных систем / А. К. Бойченко. – М.: Data Communications, 2006. – 234 с. 19. Ахтырченко К. В. Распределенные объектные технологии в информационных системах / К. В. Ахтырченко. – М.: Диалектика, 2006. – 380 с.