УДК 004.925

Д.В. КОЦУР[1]

ПОРІВНЯЛЬНИЙ АНАЛІЗ АЛГОРИТМІВ СПРОЩЕННЯ МНОГОКУТНИКІВ ТА ЛАМАНИХ ЛІНІЙ

[1]Київський національний університет імені Тараса Шевченка,
03680, просп. Академіка Глушкова, 4д, м. Київ,Україна, E-mail: dkotsur@gmail.com

Анотація. У статті проаналізовано існуючі на сьогодні алгоритми спрощення многокутників та ламаних ліній на площині. Проведено порівняльний аналіз оцінок складності алгоритмів, експериментально встановлено час виконання алгоритмів та побудовано криві залежності ступені спрощеності многокутника від значень похибки алгоритму.

Ключові слова: многокутник, ламана лінія, візуалізація, алгоритм спрощення.

Аннотация. В статье проанализированы существующие методы упрощения многоугольников и ломаных линий на плоскости. Проведен сравнительный анализ оценок сложности алгоритмов, экспериментально установлено время выполнения алгоритмов и построены кривые зависимости степени упрощенности многоугольника от значений допустимой погрешности метода.

Ключевые слова: многоугольник, ломаная, визуализация, алгоритм упрощения.

Abstract. The article deals with the analysis of the existing algorithms for polygons and polylines simplification. A comparative analysis of the complexities of the algorithms has been carried out. The performed experiments allowed us to measure the empirical execution time of the algorithms and to establish the dependence of the simplicity degree on the tolerable error of the algorithm.

Keywords: polygon, polyline, visualization, simplification algorithm.
DOI: 10.31649/1681-7893-2018-36-2-5-13

ВСТУП

Ефективне представлення геометричних об’єктів у вигляді многокутників та ламаних ліній є критичним у багатьох прикладних задачах. Зокрема, у задачах візуалізації даних [4] на етапі растеризації многокутника для відображення на дисплеї можлива ситуація, коли деякі вершини многокутника (відстань між якими менша ніж роздільна здатність одного пікселя) будуть відображені як єдина вершина. В даному випадку заміна близьких вершин однією дозволяє зменшити час відображення многокутника. При цьому візуально геомет-ричні властивості многокутника залишаються незміненими. Таким чином, спрощення мно-гокутника шляхом зменшення кількості вершин та використання його під час візуалізації дозволяє зменшити час роботи етапу растеризації. Іншим прикладом є розв’язування на-ближених геометричних задач (наприклад, локалізація об’єкта [2]), де спрощення геомет-ричного об’єкта пришвидшує роботу алгоритму даючи при цьому результат з незначною похибкою. При цьому алгоритм спрощення має видаляти лише ті вершини многокутника, які найменшим чином впливають на форму об’єкта представленого многокутником. Отже, використання алгоритмів спрощення многокутників дозволяє оптимізувати виконання ба-гатьох операцій над многокутниками, а тому детальний порівняльний аналіз існуючих на сьогодні алгоритмів спрощення є актуальної темою для дослідження, і дозволить будувати більш ефективні обчислювальні системи.

Метою роботи є аналіз та порівняння існуючих на сьогодні алгоритмів спрощення многокутників та ламаних ліній.

ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

Нехай деякий геометричний об'єкт представлений за домопогою многокутника (ламаної) $P = \{p_1, p_2, ..., p_N\}$, що складається з послідовності N точок $p_1, p_2, ..., p_N$ на площині. Задача спрощення многокутника (ламаної) полягає у тому, щоб представити об'єкт на площині за допомогою іншого такого многокутника (ламаної) $P^* = \{p_1^*, p_2^*, ..., p_M^*\}$, що $M < N$, а послідовність точок многокутника P^* є підпослідовністю точок многокутника P.

1. ОГЛЯД АЛГОРИТМІВ СПРОЩЕННЯ МНОГОКУТНИКІВ ТА ЛАМАНИХ ЛІНІЙ

На сьогоднішній день найбільш відомими та використовуваними алгоритмами спрощення многокутників та ламаних ліній на площині є наступні алгоритми:

1. Алгоритм Рамера-Дугласа-Пекера (DP). Алгоритм [5] є рекурсивним та здійснює вибірку підмножини вершин многокутника (ламаної лінії) P наступним чином:

1.1. Якщо P містить одну або дві точки ($|P| < 3$) результатом алгоритму буде множина точок P;

1.2. Інакше, знаходимо таку точку $p_k \in P$, що відстань d між точкою p_k та відрізком, що сполучає точки p_0 та p_N, є найбільшою. Якщо $d \le \varepsilon$, де $\varepsilon > 0$ допустиме відхилення алгоритму, тоді результатом роботи алгоритму є множина, що складається лише з двох точок: $\{p_0, p_N\}$;

1.3. Розбиваємо множину P на дві підмножини $P_L = \{p_1, p_2, ..., p_k\}$ та $P_R = \{p_k, p_{k+1}, ..., p_N\}$;

1.4. Рекурсивно запускаємо алгоритм для кожної з підмножин P_L та P_R;

1.5. Нехай $P_L^* = \{p_1, p_{q_2}, p_{q_3}, ..., p_k\}$ – результат рекурсивного виклику алгоритму для підмножини P_L, $P_R^* = \{p_k, p_{s_1}, p_{s_2}, ..., p_N\}$ – результат рекурсивного виклику для підмножини P_R. Тоді результатом роботи алгоритму є об'єднання цих двох множин $P^* = P_L^* \cup P_R^* = \{p_1, p_{q_2}, p_{q_3}, ..., p_k, p_{s_1}, p_{s_2}, ..., p_N\}$.

Таким чином, час виконання даного алгоритму залежить від значення вхідного параметру допустимого відхилення $\varepsilon > 0$. При чому більші значення параметру призводять до меншої глибини рекурсій, а отже і швидшого часу виконання алгоритму.

2. Алгоритм Вісвалінга-Уайатта (VW). Алгоритм [6] базується на видаленні точок ламаної, які утворюють разом зі своїми сусідами трикутники найменшої площі. Для кожної точки p_i обчислюється площа трикутника утвореного p_i, p_{i-1}, p_{i+1}. Отримані площі додаються в чергу з пріоритетом разом з індексами відповідних точок. Поки черга не порожня, послідовно обробляємо її елементи і для кожного елемента (що складається зі значення площі та індексу точки k) перевіряємо чи площа трикутника менша за задане порогове значення A (мінімальна ефективна площа трикутника). Якщо умова виконується, то видаляємо точку p_k та оновлюємо значення площ трикутників для точок p_{i-1} та p_{i+1}. Якщо ж, площа поточного трикутника більша за A, робота алгоритму завершується.

3. Алгоритм Реймана-Уіткема (RW). Алгоритм [7] є ітеративним, ініціалізація відбувається наступним чином: будується пряма лінія l, що проходить через перші дві точки p_1 та p_2 множини P. Для кожної наступної вершини $p_i \in P, i > 2$ послідовно рахується відстань d_i від p_i до лінії l. Якщо для деякої вершини $p_k \in P$ відстань d_k до прямої l є більшою ніж допустиме відхилення алгоритму $\varepsilon > 0$, тоді частина ламаної $p_1, p_2, ..., p_{k-1}$ представляється одним відрізком між точками p_1 та p_{k-1}. Алгоритм ініціалізується знову, але тепер будується пряма лінія між точками p_{k-1} та p_k. Аналогічно відбувається перевірка наступних точок $p_j, j > k$ поки не досягнута остання точка ламаної.

4. Алгоритм Опхайма (OP). Алгоритм [8] працює аналогічним чином як алгоритм Реймана-Уіткема, та може розглядатися як алгоритм Реймана-Уіткема з обмеженнями. Алгоритм Опхайма має два допустимих значення відхилення $\varepsilon_{\min} > 0$ та $\varepsilon_{\max} > 0$. Ініціалізація відбувається так само як в алгоритмі Реймана-Уіткема – будується пряма лінія l між двома точками. Проте в даному алгоритмі друга точка $p_s \in P$ вибирається як остання точка ламаної, що належить $\varepsilon_{\min}$ околу першої точки p_f (на початку це $\varepsilon_{\min}$-окіл точки p_1). Якщо ж $\varepsilon_{\min}$ окіл першої точки p_f порожній, то в якості другої точки p_s ми беремо наступну за першою точку: $s = f + 1$ (на початку роботи алгоритму це p_2). Далі послідовно рахуються відстані d_i від точки $p_i, i > s$ до лінії l поки не виконується $d_i > \varepsilon_{\min}$ або відстань між p_i та початковою точкою p_f не перевищує друге значення відхилення $\varepsilon_{\max}$. Якщо одна з умов виконалася для деякого $k > s$, то частина ламаної між p_f та p_{k-1} спрощується до відрізка, який сполучає ці дві точки. Таким

чином, за допомогою $\varepsilon_{\max} > 0$ здійснюється контроль максимальної довжини результуючого стягуючого відрізка.

5. Алгоритм Ланга (LA). Алгоритм спрощення [9] використовує регіон пошуку фіксованого розміру (у *K* точок). Перша і остання точки цього регіону пошуку утворюють відрізок, який використовується для обчислення перпендикулярної відстані до кожної проміжної точки у межах цього регіону з *K* точок. Якщо відстань до деякої проміжної точки регіону перевищує задане значення відхилення алгоритму $\varepsilon > 0$, то регіон пошуку зменшується шляхом виключення його останньої точки. Таким чином, розглядається уже регіон з $K-1$ точкою. Цей процес повторюється поки всі розраховані відстані не перевищують визначеного $\varepsilon > 0$, або коли немає більше проміжних точок. Всі проміжні точки видаляються, і новий регіон пошуку (розміром *K* точок) визначається, починаючи з останньої точки старого області пошуку. Отже, в результаті роботи алгоритму спрощуються частини многокутника (ломаної лінії), що мають розмір менше ніж K точок.

6. Алгоритм Чжао-Заальфельда (ZS). В основі роботи алгоритму [10] є поняття ε -сектора для точок p та q, що визначається як наступна множина точок: $Q(p,q,\varepsilon) = \{z \in R^2 \mid d(p,q,z) \le \varepsilon\}$, де $d(p,q,z)$ - це перпендикулярна відстань між точкою q та прямою, що сполучає точки p та z. Робота алгоритму розпочинається з ініціалізації початкового ε -сектора $Q^* := Q(p_f, p_{f+1}, \varepsilon)$, $f = 1$ для перших двох точок ламаної p_1 та p_2. Для кожної наступної вершини $p_k, k > 2$ здійснюється перевірка $p_k \in Q^*$. У випадку, якщо умова $p_k \in Q^*$ виконується, здійснюється оновлення поточного ε -сектора $Q^* := Q^* \cap Q(p_f, p_k, \varepsilon)$. Інакше, частина ламаної $p_f, p_{f+1}, ..., p_{k-1}$ спрощується до відрізка p_f, p_{k-1}, також здійснюється реініціалізація поточного ε -сектора $Q^* := Q(p_{k-1}, p_k, \varepsilon)$. Процедура повторюється поки не оброблено всі точки. Зокрема, в [10] доведено, що видалені точки лежать в межах смуги шириною не більше ніж 2ε.

7. Алгоритм "Перпендикулярна відстань" (PD). В алгоритмі [11] послідовно розглядаються трійки точок p_{i-1}, p_i, p_{i+1}, $i = 2,3,..., N-1$. Рахуються перпендикулярні відстані d_i від точок p_i до відрізка, що сполучає сусідні точки p_{i-1} та p_{i+1}. Всі точки p_i для яких $d_i < \varepsilon$, де $\varepsilon > 0$ - задане порогове значення, видаляються. Вся процедура повністю повторюється *K* разів, де *K* – параметр алгоритму.

2. АНАЛІЗ АЛГОРИТМІВ

Порівняльний аналіз часових оцінок складності в залежності від кількості точок *N* вхідного многокутника або ламаної лінії наведено в Табл. 1.

Таблиця 1. **Порівняльний аналіз оцінок складності алгоритмів спрощення**

Назва алгоритма	**Оцінка складності (в середньому)**	**Оцінка складності (в найгіршому випадку)**
Алгоритм Рамера-Дугласа-Пекера	$O(N \log N)$	$O(N^2)$
Алгоритм Вісвалінгам-Уайатта	$O(N \log N)$	$O(N \log N)$
Алгоритм Реймана-Уіткема	$O(N)$	$O(N)$
Алгоритм Опхайма	$O(N)$	$O(N)$
Алгоритм Ланга	$O(NK)$	$O(NK^2)$
Алгоритм Чжао-Заальфельда	$O(N)$	$O(N)$
Алгоритм "Перпендикулярна відстань"	$O(NK)$	$O(NK)$

Для аналізу часу роботи описаних вище алгоритмів було використано контури фігур з набору даних MPEG-7 CE Shape-1, який відображає найтиповіші форми фігур в задачах комп'ютерного зору та аналізу зображень. Контури об'єктів були отримані шляхом обробки бінарних зображень за допомогою алгоритму «Рухомі квадрати» [12]. Після цього контури були інтерпольовані за допомогою кубічних параметричних періодичних сплайнів. На основі представлення фігур у вигляді сплайнів було отримано (за домопомогою методів дискретизації кривих) многокутники з різною кількістю точок, які в подальшому було використано для оцінювання залежності часу роботи алгоритму від кількості вершин вхідних многокутника. Для аналізу алгоритмів спрощення було використано 1282 зовнішні контури фігур набору даних MPEG-7 CE Shape-1.

Для здійснення порівняльного аналізу роботи алгоритмів на наборі даних MPEG-7 CE Shape-1 було обчислено наступні величини:

а) *відстань Гаусдорфа* [13] між двома многокутниками**:**

$$d_H(P,P_S) := \max\left(\sup_{p\in P}\inf_{p_S\in P_S} dist(p,p_s), \sup_{p_S\in P_S}\inf_{p\in P} dist(p,p_s)\right), \quad (1)$$

де P - многокутник (ламана лінія) до застосування алгоритму спрощення, P_S - многокутник (ламана лінія), що є результатом алгоритму спрощення, $dist(p,p_s)$ - Евклідова відстань між точкою p на многокутнику P та точкою p_s на многокутнику P_S;

б) *ступінь (відсоток) спрощеності* многокутника (ламаної лінії) P_S по відношенню до початкового многокутника (ламаної лінії) P:

$$SR(P,P_S) = \frac{|P|-|P_S|}{|P|}\cdot 100\%, \quad (2)$$

де $|P|$ - це кількість вершин многокутника.

Таким чином, відстань Гаусдорфа $d_H(P,P_S)$ дає змогу оцінити схожість спрощеного многокутника P до початкового многокутника P_S (похибку алгоритму спрощення), а $SR(P,P_S)$ - зменшення кількості точок (у відсотках) в P по відношенню до початкового многокутника P_S.

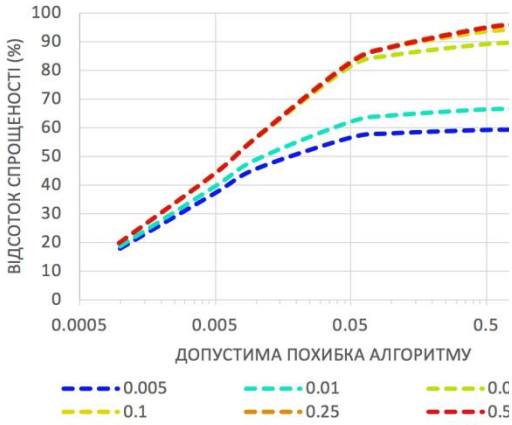
Таблиця 2. – Параметри алгоритмів спрощення та застосовувані евристичні правила

Назва алгоритму	**Парамети алгоритму**	**Евристичні правила**
Алгоритм Рамера-Дугласа-Пекера	$\varepsilon > 0$ - значення допустимого видхілення алгоритму	відсутні
Алгоритм Вісвалінга-Уайатта	$A > 0$ - мінімальна ефективна площа трикутника	відсутні
Алгоритм Реймана-Уіткема	$\varepsilon > 0$ - перпендикулярне допустиме видхілення алгоритму	відсутні
Алгоритм Опхайма	$\varepsilon_{\min}, \varepsilon_{\max} > 0$ - допустимі значення видхілення алгоритму	$\varepsilon_{\max} = +\infty$
Алгоритм Ланга	$\varepsilon > 0$ - перпендикулярне допустиме видхілення алгоритму R - розмір (число точок) регіону пошуку алгоритму	$R = \theta N$, $\theta \in \{0.05, 0.1, 0.2, 0.25, 0.5, 1.0\}$, де N - кількість точок многокутника
Алгоритм Чжао-Заальфельда	$\varepsilon > 0$ - значення допустимої напівширини сектора;	відсутні
Алгоритм "Перпендикулярна відстань"	$\varepsilon > 0$ - перпендикулярне допустиме видхілення алгоритму K - кількість ітерацій алгоритму	$R = \theta N$, $\theta \in \{0.05, 0.1, 0.2, 0.25, 0.5, 1.0\}$, де N - кількість точок многокутника

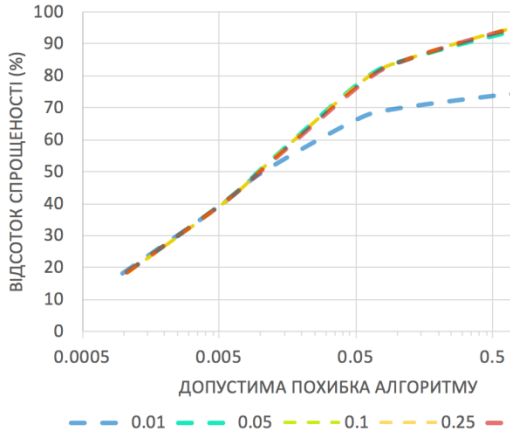
Щоб порівняти якість роботи алгоритмів спрощення необхідно встановити залежність між значенням допустимої похибки алгоритму d_H та середнім відсотком спрощеності результуючих многокутників. Для цього потрібно з'ясувати залежність допустимої похибки алгоритму d_H (відстані Гаусдорфа) від вхідних параметрів алгоритму. У випадку алгоритмів з єдиним параметром, таку залежність можна отримати шляхом перебору параметра алгоритму. Проте процес перебору ускладнюється у випадку алгоритмів спрощення, які мають два параметри. Тому при виборі другого параметру було застосовано евристичні правила, що наведені у табл. 2.

3. РЕЗУЛЬТАТИ АНАЛІЗУ

Для алгоритмів Ланга та «Перпендикулярна відстань» було оцінено залежність середнього відсотку спрощеності результуючих многокутників від значення порогу дозволеної похибки алгоритму. При проведенні експерименту для другого параметру алгоритмів використовувалися евристики зазначені в табл. 2. Встановленні в результаті тестування графіки залежностей відображені на рис. 1(а) для алгоритму Ланга та на рис. 1(б) для алгоритму «Перпендикулярна відстань».



а) Алгоритм Ланга
$\theta \in \{0.005, 0.01, 0.05, 0.1, 0.25, 0.5\}$



б) Алгоритм “Перпендикулярна відстань”:
$\theta \in \{0.01, 0.05, 0.1, 0.25, 0.5\}$

Рис. 1. Оцінка залежності середнього відсотка спрощеності результуючого многокутника від порогового значення d_H похибки для алгоритмів Ланга (а) та «Перпендикулярна відстань» (б).

Графіки на рис. 1(а) показують, що для алгоритму Ланга значення евристичного параметру $\theta > 0.25$ не впливають на збільшення відсотку спрощеності. Таким чином, відбувається насичення алгоритму, а тому значення $\theta > 0.25$ лише будуть сповільнювати обчислення при цьому не впливаючи істотно на результат роботи алгоритму. У випадку алгоритму «Перпендикулярна відстань» алгоритм насичується при $\theta > 0.1$, про що свідчать графіки на рис. 1(б).

Графіки на рис. 2 показують, що для $d_H < 0.005$ найбільше спрощення многокутника досягається за допомогою алгоритму Вісвалінга-Уайатта (VW). Проте, при $d_H \geq 0.005$ найвищий відсоток стиснення досягається алгоритмом Чжао-Заальфельда (ZS). Наступні алгоритми мають ідентичні показники ефективності спрощення: алгоритм Рамера-Дугласа-Пекера (DP), алгоритм Вісвалінга-Уайатта (VW), алгоритм Опхайма (OP), алгоритм Ланга (LA), алгоритм Чжао-Заальфельда (ZS), алгоритм "Перпендикулярна відстань" (PD). Алгоритм Реймана-Уіткема (RW) показує найгіршу (приблизно на 10% нижчу) ефективність роботи по відношенню до решти алгоритмів. Також слід зазначити, що різниця ефективності роботи алгоритмів зменшується при збільшенні значення похибки алгоритму.

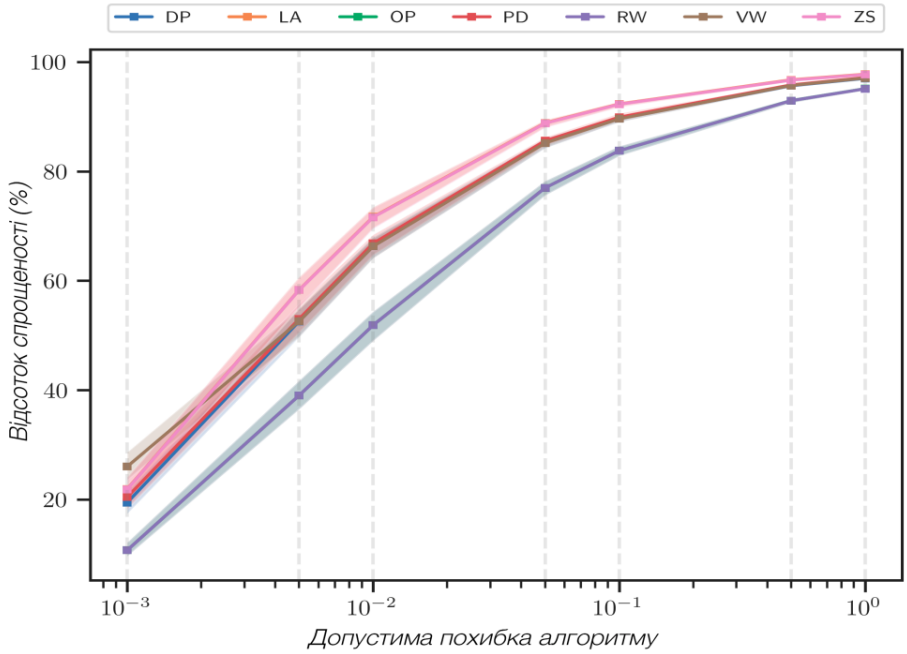
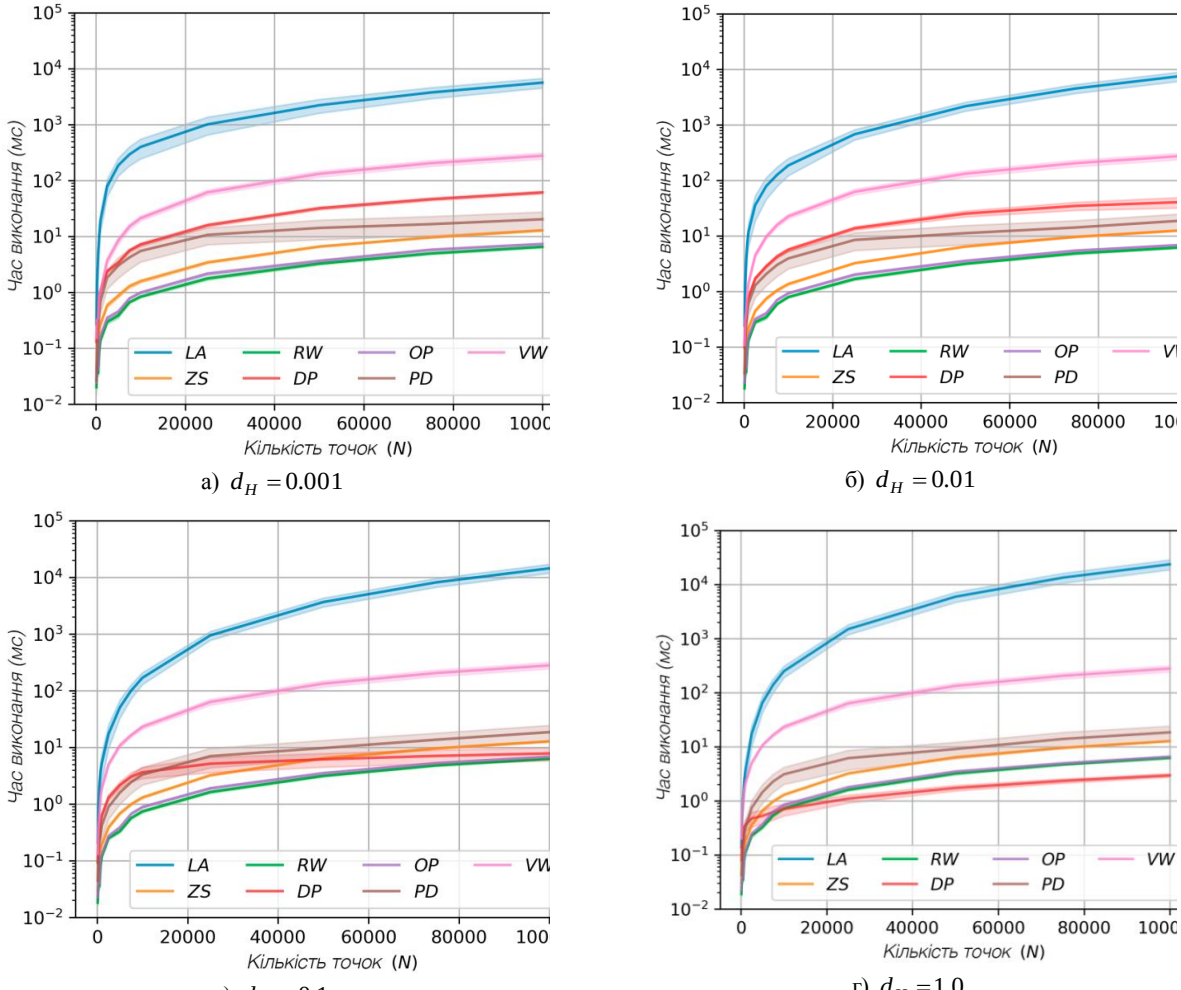


Рис. 2. Оцінка залежності спрощеності результуючого многокутника від порогового значення похибки



а) $d_H = 0.001$ б) $d_H = 0.01$

в) $d_H = 0.1$ г) $d_H = 1.0$

Рис. 3. Порівняння середнього часу роботи алгоритмів у залежності від кількості точок N многокутника (ламаної лінії) для значень допустимої похибки $d_H \in \{0.001, 0.01, 0.1, 1.0\}$.

Також було оцінено залежність часу роботи алгоритмів від кількості точок вхідних многокутників для фіксованих значень $d_H \in \{0.001, 0.01, 0.1, 1.0\}$, що відображено на рис. 3. При цьому для кожного значення *N* генерувалися 200 довільних многокутників з N вершинами, інші 200 многокутників було отримано з набору даних MPEG-7 CE Shape-1. Час роботи алгоритмів було протестовано на комп'ютері з процесором Intel Core i7, 2.2GHz 16GB RAM та усереднено для кожного значення *N*.

Графіки на рис. 3 показують, що найповільніше працюють алгоритми Ланга (LA) та Вісвалінга-Уайатта (VW). Для значень допустимої похибки $d_H \le 0.1$ найшвидшими алгоритмами є алгоритм Реймана-Уіткема (RW), а також алгоритм Опхайма. При $d_H > 0.1$ алгоритм Рамера-Дугласа-Пекера (DP) показує значне покращення часу роботи, що зв'язано зі зменшенням глибини рекурсії алгоритму.

Таким чином, якщо час роботи алгоритму не є критичним, алгоритм Вісвалінга-Уайатта (VW) дозволяє досягнути найбільшого спрощення при невеликій допустимій похибці $d_H < 0.005$, при $d_H \ge 0.005$ найбільше спрощення досягається алгоритмом Чжао-Заальфельда (ZS). Якщо ж час роботи є критичним, а значення допустимої похибки $d_H \le 0.1$, то алгоритм Опхайма дозволяє отримати результат найшвидше, при цьому многокутник буде лише на 2-5% менш спрощеним ніж у випадку алгритму Чжао-Заальфельда (ZS). При великому значенні допустимої похибки $d_H > 0.1$ та великій кількості точок вхідного многокутника *N*>10000 найшвидше працює алгоритм Рамера-Дугласа-Пекера (DP).

ВИСНОВКИ

У результаті порівняльного аналізу існуючих алгоритмів спрощення многокутників (ламаних) було встановлено їх часові оцінки складності, а також оцінки складності у найгіршому випадку. Більшість алгоритмів мають лінійну складність, за винятком алгоритмів Рамера-Дугласа-Пекера та Вісвалінга-Уайатта, які мають складність $O(N \log N)$, де N – це кількість вершин многокутника (ламаної). Складність деяких алгоритмів (наприклад, алгоритми Ланга та «Перпендикулярна відстань») залежить від максимального числа ітерацій K або розміру регіону спрощення R. У випадку $K \sim N$ або $R \sim N$, ці алгоритми мають квадратичну складність. Наведені оцінки складності дають змогу оцінити та порівняти асимптотичну поведінку роботи алгоритмів при $N \to \infty$. Проте, на практиці час роботи алгоритму залежить також від констант та множників, що не враховуються в асимптотичній оцінці.

Тому було проведено чисельні експерименти та оцінено залежність часу роботи алгоритмів від розміру вхідних даних – кількості точок многокутника (ламаної лінії). У результаті проведених експериментів було виявлено, що найшвидшими алгоритмами спрощення є алгоритми Реймана-Уіткема та Опхайма, що мають лінійну складність.

Якість роботи алгоритмів спрощення було оцінено за допомогою емпірично обчислених кривих, які відображають залежність відсотку спрощеності результуючого многокутника від допустимого значення похибки алгоритму. Отримані оцінки якості стиснення многокутників дають можливість вибрати алгоритм, який для заданого значення порогу точності може представити многокутник (ламану) за допомогою найменшої кількісті точок зберігаючи його основні геометричні властивості (по відношенню до метрики Гаусдорфа).

Було проведено чисельні експерименти над многокутниками з набору даних MPEG-7 CE Shape-1 (1282 многокутника), який представляє найбільш типові форми об'єктів у прикладних системах. У результаті експериментів було також встановлено, що для значень допустимої похибки алгоритму, що не перевищують 0.005, найбільше спрощення многокутників досягається алгоритмом Вісвалінга-Уайатта. Якщо ж значення допустимої похибки алгоритму перевищує 0.005, тоді найоптимальніше представлення многокутника можна отримати за допомогою алгоритму Чжао-Заальфельда. У випадку алгоритму Реймана-Уіткема, многокутники(ламані) представляються найбільшою кількістю точок.

СПИСОК ЛІТЕРАТУРИ

1. B. P. Buttenfield, R. B. McMaster, (1991). Map Generalization: making rules for knowledge representation. New York: John Wiley & Sons.

2. V. Tereshchenko, Y. Tereshchenko, (2017). Triangulating a region between arbitrary Polygons. International Journal of Computing, 16, 3, 160-165.

3. M. Berg, O. Cheong, M. Kreveld, M. Overmars, (2008). Computational Geometry: Algorithms and Applications, (3rd ed.). Berlin: Springer.

4. N. Mustafa, S. Krishnan, G. Varadhan, S. Venkatasubramanian, (2006). Dynamic simplification and visualization of large maps. International Journal of Geographical Information Science, 20, 3, 273-302.

5. D. Douglas, T. Peucker, (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. The Canadian Cartographer, 10, 2, 112–122. doi:10.3138/FM57-6770-U75U-7727

6. M. Visvalingam, J. D. Whyatt, (1993). Line generalisation by repeated elimination of points. Cartographic Journal, 30, 46-51.

7. K. Reumann, A. P. M. Witkam, (1973). Optimizing curve segmentation in computer graphics. In Proc. International Computing Symposium, 467–472.

8. H. Opheim, (1982). Fast data reduction of a digitized curve. Geo-Processing, 2, 33–40.

9. T. Lang, (1969). Rules for robot draughtsman. Geographical Magazine, 42, 50-51.

10. Z. Zhao, A. Saalfeld, (1997). Linear-time sleeve-fitting polyline simplification algorithms. In Proc. Annual Convention and Exposition Technical Papers, 214-223.

11. J. Song, R. Miao, (2016). A Novel Evaluation Approach for Line Simplification Algorithms towards Vector Map Visualization. International Journal of Geo-Information, 5, 12, 223. doi: 10.3390/ijgi5120223

12. C. Maple, (2003). Geometric design and space planning using the marching squares and marching cube algorithms. In Proc. International Conference on Geometric Modeling and Graphics, 90-95. doi: 10.1109/GMAG.2003.1219671

13. P. Cignoni, C. Rocchini, R. Scopigno, (1998). Metro: Measuring Error on Simplified Surfaces. Computer Graphics Forum, 17, 2, 167–174.

REFERENCES

1. B. P. Buttenfield, R. B. McMaster, (1991). Map Generalization: making rules for knowledge representation. New York: John Wiley & Sons.

2. V. Tereshchenko, Y. Tereshchenko, (2017). Triangulating a region between arbitrary Polygons. International Journal of Computing, 16, 3, 160-165.

3. M. Berg, O. Cheong, M. Kreveld, M. Overmars, (2008). Computational Geometry: Algorithms and Applications, (3rd ed.). Berlin: Springer.

4. N. Mustafa, S. Krishnan, G. Varadhan, S. Venkatasubramanian, (2006). Dynamic simplification and visualization of large maps. International Journal of Geographical Information Science, 20, 3, 273-302.

5. D. Douglas, T. Peucker, (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. The Canadian Cartographer, 10, 2, 112–122. doi:10.3138/FM57-6770-U75U-7727

6. M. Visvalingam, J. D. Whyatt, (1993). Line generalisation by repeated elimination of points. Cartographic Journal, 30, 46-51.

7. K. Reumann, A. P. M. Witkam, (1973). Optimizing curve segmentation in computer graphics. In Proc. International Computing Symposium, 467–472.

8. H. Opheim, (1982). Fast data reduction of a digitized curve. Geo-Processing, 2, 33–40.

9. T. Lang, (1969). Rules for robot draughtsman. Geographical Magazine, 42, 50-51.

10. Z. Zhao, A. Saalfeld, (1997). Linear-time sleeve-fitting polyline simplification algorithms. In Proc. Annual Convention and Exposition Technical Papers, 214-223.

11. J. Song, R. Miao, (2016). A Novel Evaluation Approach for Line Simplification Algorithms towards Vector Map Visualization. International Journal of Geo-Information, 5, 12, 223. doi: 10.3390/ijgi5120223

12. C. Maple, (2003). Geometric design and space planning using the marching squares and marching cube algorithms. In Proc. International Conference on Geometric Modeling and Graphics, 90-95. doi: 10.1109/GMAG.2003.1219671

13. P. Cignoni, C. Rocchini, R. Scopigno, (1998). Metro: Measuring Error on Simplified Surfaces. Computer Graphics Forum, 17, 2, 167–174.

Надійшла до редакції 04.08.2018 р.

КОЦУР Дмитро Вікторович – аспірант кафедри математичної інформатики, факультету комп'ютерних наук та кібернетики, Київського національного університету імені Тараса Шевченка, м. Київ, Україна.