УДК 519.85

АНАЛІЗ І СКЛАДНІСТЬ АЛГОРИТМУ ЗОБРАЖЕННЯ ОДНОЗВ'ЯЗНОГО НЕОПУКЛОГО МНОГОКУТНИКА У ВИГЛЯДІ ОБ'ЄДНАННЯ ОПУКЛИХ МНОГОКУТНИКІВ

Пандорін О.К., Панкратов О.В., Новожилова М.В.

Задача, що розглядається у статті, відноситься до класу задач обчислювальної геометрії і знаходить застосування у багатьох оптимізаційних задачах геометричного проектування, машинної графіки тощо.

Розглянемо неопуклий замкнений однозв'язний многокутник M^*, заданий множиною вершин $M = \{m_i \mid i=1,2,...,n\}$, впорядкованою за обходом його контура у довільному напряму. Нехай ребро r_i з'єднує вершину m_i з вершиною, що є наступною за напрямом обходу.

Необхідно зобразити об'єкт M^* у вигляді об'єднання опуклих замкнених многокутників:

$$M^* = \bigcup_{l=1}^{g} M^*_l ; \qquad (1)$$

Відомо кілька підходів до зображення неопуклого многокутника у вигляді сукупності його опуклих підмножин.

"Жадні" алгоритми побудови "товстого" покриття початкового неопуклого об'єкта M^* сім'єю опуклих многокутників розглядаються в [1]. Розвитком даної ідеї є "картопляний" алгоритм [1] зображення (1) неопуклого многокутника M^*. Цей алгоритм зменшує кількість опуклих підмножин, що складають покриття M^*. Обчислювальна складність найшвидшого "картопляного" алгоритму дорівнює $O(n^7)$, де n - кількість вершин початкового многокутника.

Більш вишуканими алгоритмами розв'язання даної задачі можна вважати два алгоритми, що є кращими як з точки зору обчислювальної складності, так і з точки зору зменшення об'єма інформації про геометричну структуру (1).

Остання властивість має особливе значення під час розв'язання таких задач оптимізаційного проектування, як розміщення неопуклих многокутників у багатокутній області розміщення з метою оптимізації певного крітерія якості, наприклад, мінімізації певної метричної характеристики області розміщення. Сучасні методи локальної оптимізації таких задач [3,4] дуже ефективно працюють, коли неопуклі многокутники зображені як сукупність опуклих підмножин.

Зауважимо, що при порівнянні алгоритмів необхідно брати до уваги не тільки обчислювальну складність, але і такі характеристики, як кількість опуклих підмножин у сукупності $\bigcup_{l=1}^{g} M^*_l$ і кількість нових вершин та ребер, що породжуються. У залежності від області застосування алгоритму з гіршою оцінкою може бути віддано перевагу перед більш швидким, але менш економним алгоритом.

Ідея першого алгоритму викладена в [2] та складається з послідовності таких кроків:

Алгоритм 1.

Крок 1. Покласти індекс q=1.

Крок 2. Визначити величину внутрішнього кута ψ_q вершини m_q. Якщо $\psi_q > \pi$, - перехід до кроку 3., інакше - до кроку 5.

Крок 3. Визначити сторону $[m_g, m_{g+1}]$ многокутника M^*, що має з променем $[m_{q-1}\ m_q,)$ непустий перетин: точку m_{q_g}. Якщо таких сторін більш ніж одна, взяти найближчу до вершини m_{i_q} точку m_{q_g}, для якої відстань $|m_{q_g}, m_{i_q}|$ є мінімальною.

Крок 4. Переформувати список вершин m_q, q=1,2,...n, додавши інформацію про нову вершину m_{q_g}. Тоді n=n+1.

Крок 5. q= q +1. Якщо $q \leq n$ - перехід до кроку 2., інакше - закінчити.

Відзначимо, що алгоритмічна реалізація даного підходу може бути подана як у вигляді рекурсивного алгоритму, так і у вигляді ітераційного алгоритму.

Обчислювальна складність алгоритму авторами роботи [2] не розглядається, але складає не більш $O(n^2)$ від кількості вершин початкового многокутника.

Рис.1 є ілюстрацією дії алгоритму 1. Початковий многокутник M^* зображений у вигляді сукупності п'яти опуклих многокутників M^*_l, l=1,2,...,5.

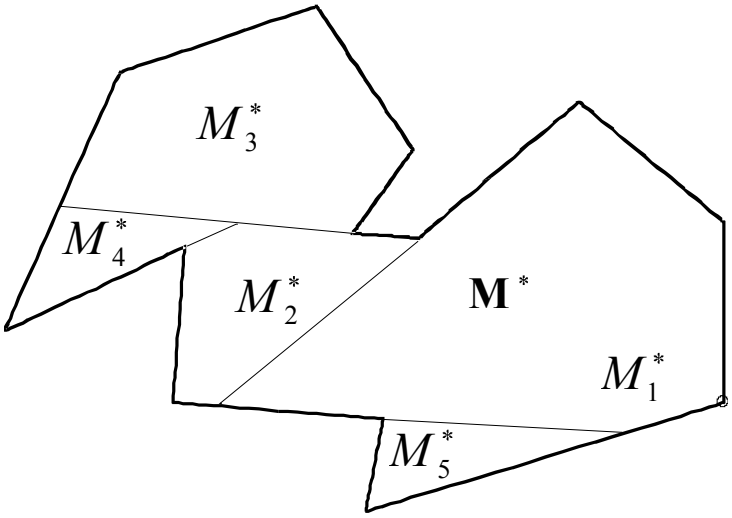

Рис. 1

Розглянемо другий алгоритм.

Оберемо довільне ребро многокутника M^* і сформуємо впорядковану множину T, у яку включимо вершини, інцидентні цьому ребру - хай це ребро r_i і $T=\{m_i, m_{i+1}\}$.

Будемо, починаючи з вершини m_{i+2}, послідовно включати у множину T вершини за вибраним напрямом обходу, доки многокутник T^*, заданий множиною T як множиною вершин, являє собою опуклу підмножину M^*.

На наступній стадії повторюємо описану процедуру з тією відмінністю, що вершини беруться з множини M починаючи з m_{i-1} у зворотному напрямі і додаються у початок множини T.

Наприкінці отримаємо множину T у вигляді:

$$T =\{ m_{i-a}, \ldots, m_i, m_{i+1}, \ldots, m_{i+b} \}. \qquad (2)$$

Зауваження 1. Діючи таким чином, ми формально можемо отримати індекси, менші 1 чи більші n. Залишаючи для спрощення формулу (2) незмінною, будемо надалі мати на увазі, що індексу i, меншому, 1 відповідає індекс i+n, а індексу j, більшому n - індекс j-n.

Зауваження 2. Якщо у T потрапили усі вершини з M, то M^* - опуклий многокутник.

Зрозуміло, може виявитися, що немає можливості додати до T бодай однієї вершини за наведеної умови. У цьому випадку треба повторювати процедуру формування множини T послідовно для усіх ребер з M^*, і обов'язково знайдеться хоч одне, для котрого відповідна множина T має не менш за три елементи (тобто хоча б одне з a і b, відмінне від 0).

Зауваження 3. Можна скоротити перегляд ребер, якщо аналізувати тільки ребра, інцидентні вершинам з внутрішнім кутом, меншим за 180^0.

Далі ми просто викреслюємо з M вершини $m_{i-a+1}, \ldots, m_i, m_{i+1}, \ldots, m_{i+b-1}$, отримавши таким чином зображення многокутника M^* у вигляді об'єднання опуклого многокутника T^*, що заданий множиною

$$T=\{m_{i-a}, \ldots, m_i, m_{i+1}, \ldots, m_{i+b}\}$$

та многокутника M_1^*, що заданий множиною вершин

$$M_1=\{m_1, \ldots, m_{i-a}, m_{i+b}, \ldots, m_n\}$$

(як і вище для спрощення ми вважаємо, що i-a>1 та i+b<n).

Надалі треба сформувати многокутник T_1^* за описаною процедурою, потім T_2^* і т.п., доки на k-му етапі не отримаємо $T_k^*=M_k^*$. Таким чином ми отримаємо M^* у вигляді об'єднання опуклих многокутників T_i^*, i=1,2, . . . , k.

Розглянемо тепер складність алгоритму побудови множини T.

Хай множина T^k з k вершин описує опуклу підмножину M^*, і ми додаємо до T (k+1)-й елемент. Треба проаналізувати, чи описує множина T^{k+1} опуклу підмножину M^*.

Цей аналіз, у свою чергу, можна розкласти на два етапи.

I етап. Аналіз того, чи є опуклим многокутник, описаний впорядкованою множиною вершин T^{k+1}.

Складність першого етапу дорівнює O(k), оскільки вже відомо, що T^k описує опуклий многокутник, і треба перевірити тільки (k+1)-шу вершину та два інцидентних їй ребра, одне з котрих належить многокутнику M^*, а інше з'єднує 1-шу та (k+1)-шу вершини з T^{k+1}.

II етап. Аналіз того, чи є опуклий многокутник, описаний впорядкованою множиною вершин T^{k+1}, підмножиною многокутника M^*.

Складність другого етапу дорівнює O(n-k), оскільки многокутник, описаний множиною T^{k+1}, тоді є опуклою підмножиною M^*, коли ребро, що поєднує 1-шу та (k+1) -шу вершини з T^{k+1}, не перетинає жодного з ребер M^*.

Таким чином, складність побудови кожної з множин T^k можна вважати рівною O(n-k)+ O(k)= O(n), а складність побудови усієї множини T - $O(n^2)$.

Оскільки можливий перегляд ребер у випадку, коли немає можливості додати до T^2 нових елементів, складність формування множин T та M_1 - $O(n^3)$.

Нарешті, оскільки у найгіршому випадку многокутник може бути розбитий щонайменше на (n-3) опуклі частини, слід визнати складність наведеного методу за $O(n^4)$.

Насправді ця оцінка досить груба, оскільки випадок з високою складністю одного з етапів відповідає низькій складності інших.

Наприклад, складність $O(n^2)$ будування множини T досягається у випадку, коли многогранник M^* опуклий, й у цьому випадку взагалі не треба перебирати ребер, а опуклих частин M^* буде не O(n), а тільки одна.

Загалом можна вважати, що складність методу, що пропонується, становить $O(n^3)$, але це не доведено.

Рис. 2 ілюструє дію другого алгоритму. Початковий многокутник M^* зображений як сукупність восьми опуклих многокутників M^*_l, l=1,2,...,8.

Зауваження 4. Внаслідок дії другого алгоритму можливо виникнення опуклих многокутників, у яких усі сторони не належать до зовнішнього контура многокутника M* (на рис.2 це многокутники M^*_3 і M^*_7. Вони заштриховані). Для моделювання та розв'язання певних оптимізаційних задач геометричного проектування (таких, наприклад, як розміщення скінченної множини орієнтованих чи неорієнтованих многокутників в обмеженій області розміщення) такі многокутники не становлять інтересу, тому інформацію про них можна не зберігати.

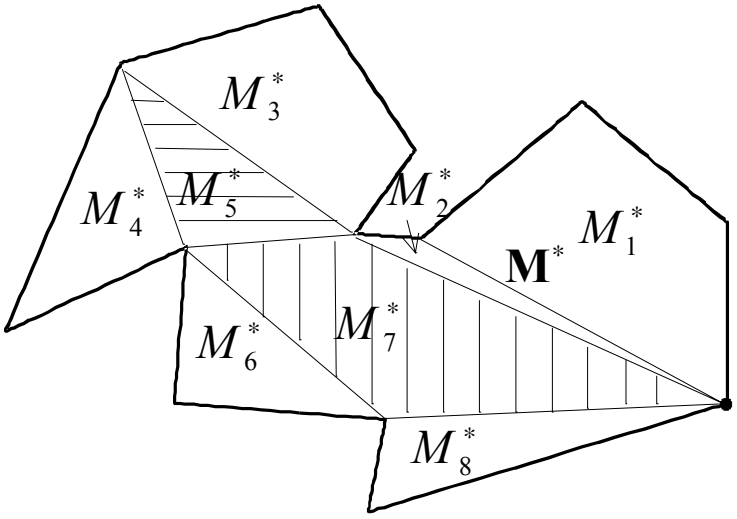

Рис. 2

При розв'язанні оптимізаційних задач геометричного проектування були реалізовані обидва методи побудови зображення об'єкта M* у вигляді (1).

У результаті, незважаючи на відмінність в оцінках обчислювальної складності, вибір було зроблено на користь другого алгоритму. При цій побудові кількість вершин зовнішнього контура геометричної структури $\bigcup_{l=1}^{g} M^*_l$ не зростає, тоді як при першому підході деякі сторони початкового многокутника точками перетину діляться на дві, що призводить до виродженості у задачах оптимізаційного геометричного проектування, яку треба буде обробляти тільки на етапі розв'язання задачі оптимізації.

Приклад. Нехай треба розмістити два многокутники у смузі заданої ширини так, щоб довжина смуги була мінімальною.

Ця задача є задачею оптимізаційного геометричного проектування, і для її розв'язання застосовуються ітераційні методи локальної оптимізації [3, 4]. На кожній ітерації метода оптимізації необхідно обробляти систему рівнянь, яка задає точку у просторі параметрів розміщення об'єктів. Розглянемо ситуацію, що може виникнути на певній і-й ітерації метода (рис. 3). Многокутник M^*_i за допомогою першого алгоритму зображений як сукупність опуклих многокутників M_{i_1} і M_{i_2}.

У припущенні, що многокутники M_{i_1} і M_{i_2} мають один полюс (початок власної системи координат), система рівнянь, що визначає це розміщення (Рис.3), містить два лінійно-залежних рівняння.

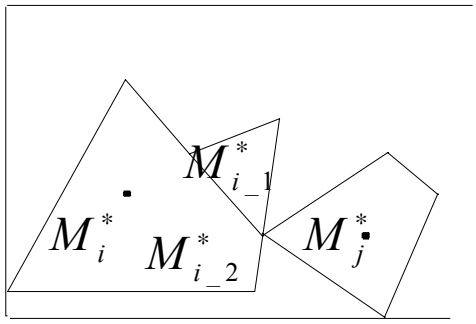

Рисунок 3.

Необхідно відмітити, що другий алгоритм має таку негативну особливість у порівнянні з першим алгоритмом, а саме, досить часто зустрічаються ситуації, коли з певною вершиною m_q об'єкта M*, при якій величина внутрішнього кута $\psi_q > \pi$, збігаються кілька вершин опуклих підмножин, що його складають (рис. 4).

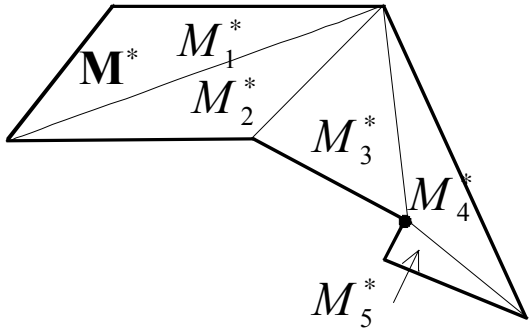

Рис. 4

У вершині об'єкта M*, що позначена кружком, збігаються вершини трьох його опуклих підмножин: M^*_3, M^*_4, M^*_5.

Але невелика модифікація цього алгоритму, що складається у продовженні сторін опуклих многокутників на певне δ, дозволяє усунути цей недолік (рис. 5), тому на роботі методів оптимізації в задачах геометричного проектування він не відбивається.

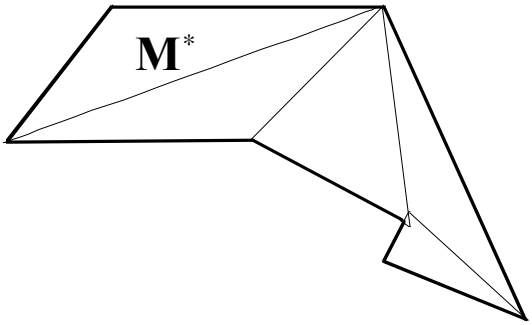

Рис. 5. Результат роботи модифікованого алгоритму.

Зауваження 5. Другий алгоритм наведений для однозв'язних многокутників. Але після певної нескладної модифікації він може бути застосований для зображення многозв'язних многокутників у вигляді сукупності опуклих многокутників без підвищення обчислювальної складності.

ЛІТЕРАТУРА

1. Chang J.S, Yap C.V. A polynopial solution for the potato-peeling problem // Discrete Comput. Geom. - 1986. -N 1. - P.155-182,.
2. Стоян Ю.Г., Гиль Н.И. Методы и алгоритмы размещения плоских геометрических объектов.- К.: Наук. думка, 1976. - 287 с.
3. Stoyan, Yu.G., Novozhilova, M.V., Modelling and Solving of the Allocation Problem of Non-Convex Polygons with Rotations // Proc. 17th International Conf. on System Modelling and Optimization - Prague (Czech Republic)- 1996. - P. 559-567.
4. Панкратов А.В., Пономаренко Л.Д. Оптимизационная задача упаковки многогранников (Линейная модель и метод решения) // АСУ и приборы автоматики. Харьков, 1987. - Вып. 83. - С. 14-19.