ЛЕКЦІЯ 11. ПЕРЕТИН

По перше - широкий спектр практичних застосувань у задачах: видимості, розпізнавання образів, трасування та розташування, лінійного програмування, архітектурного проектування, модельного дизайну, проектування електронної та електронно –обчислювальної техніки, тощо.

По друге – при дослідженні складності алгоритмів геометричних задач. Побудова та аналіз розв'язків задач про перетин дозволяють визначити структурні особливості деяких геометричних задач та дати відповіді на ряд фундаментальних теоретичних проблем. Зокрема, складність розв'язання задачі про простоту многокутника.

11.1 Задачі вилучення невидимих ліній та поверхонь.

Задача вилучення невидимих ліній і поверхонь являється однією з найбільш складних у комп'ютерній графіці. При побудові вірного зображення тривимірних об'єктів необхідно вміти визначати, які частини об'єктів будуть видимі при заданому проектуванні (на картинну площину), а які будуть закриті іншими гранями. При розв'язанні розглядуваної задачі використовують два шляхи: технічний та алгоритмічний.

ЗАДАЧА П1.(ПОБУДОВА ПЕРЕТИНУ).*Задано два геометричних об'єкти. Необхідно побудувати їх перетин.*

11.2 Задачі розпізнавання образів

Однією із задач розпізнавання образів є задача класифікації геометричних об'єктів. Так, наприклад, має місце задача розпізнавання на площині многокутників (опуклий, зірковий, простий).При цьому важливим моментом цього розпізнавання є класифікація перетинів ребер (прямих).

ЗАДАЧА КЛАСИФІКАЦІЇ. *Задано N точок, кожна з яких ідентифікується по при належності до одного з m класів. Необхідно нову точку класифікувати (віднести до одного із класів).*

Приклад 11.1 Необхідно на заданих класах людей, нехай це будуть класи "ЖІНКИ" та "ЧОЛОВІКИ", за віком та ростом класифікувати людину, яка має певний зріст та вагу (рис 11.1).

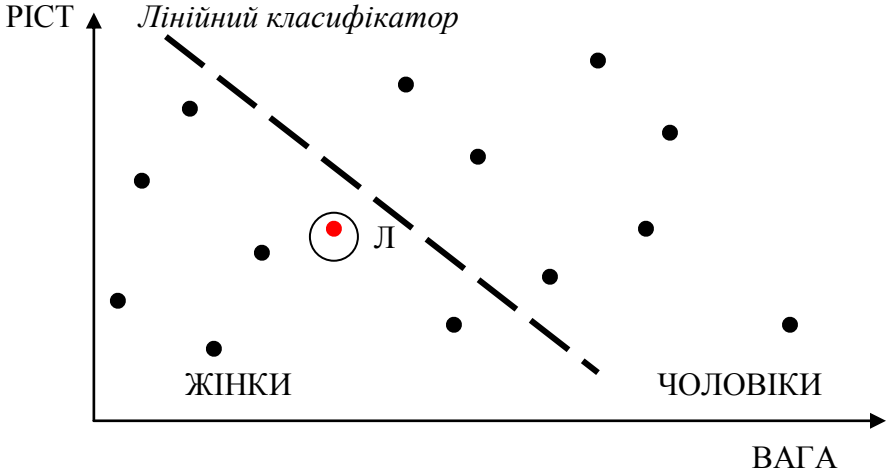


Рис.11.1 Задача класифікації з двома змінними

Для розв'язання поставленої задачі необхідно створити *лінійний класифікатор*- функція, яка визначатиме за одну операцію клас для нового елемента.

Означення 11.1 Дві множини S_1 і S_2 називаються *лінійно роздільними,* тоді і лише тоді, коли існує така гіперплощина H, яка розділяє ці множини так, що елементи цих множин лежать по різні сторони H.

Теорема 11.1 *Дві множини точок лінійно роздільні тоді і лише тоді, коли їх опуклі оболонки не перетинаються.*

ЗАДАЧА П2. (ПЕРЕВІРКА ПЕРЕТИНУ). *Задано два геометричних об'єкти. Необхідно перевірити чи перетинаються вони.*

11.3 Трасування та розташування

Мінітюаризація та масштабування сучасної електронної та електронно-обчислювальної техніки потребує автоматизацію процесу проектування та розташування окремих елементів та модулів технічних пристроїв. Існуючі евристичні методи дають результати, проте точність та коректність їх інколи викликає сумніви. Постає необхідність в одержаних результатах проводити детальну попарну перевірку елементів схем.

ЗАДАЧА П3. (ПОПАРНІ ПЕРЕТИНИ).*Задано N геометричних об'єктів. Необхідно визначити усі попарні перетини.*

11.4 Двовимірний випадок.

11.4.1 Перетин опуклих многокутників

ЗАДАЧА П1.1 (ПОУДОВА ПЕРЕТИНУ ОПУКЛИХ МНОГОКУТНИКІВ). *Задано два многокутники: P з L вершинами та Q із M вершинами. Необхідно визначити їх перетин.*

Теорема 11.2 *Перетином опуклих L-кутника і M-кутника є опуклий многокутник, який має не більше L+M вершин.*

Один із підходів щодо розв'язання цієї задачі полягає у розбитті площини на області, в, кожній з яких перетин многокутників обчислюється просто.

Теорема 11.3 *Перетин опуклих L-кутника і M-кутника можна побудувати за час $\theta(L+M)$.*

Другий підхід полягає в наступному. Припустимо, що многокутники перетинаються ($P \cap Q \neq \varnothing$), та розглянемо многокутник P^*, границею якого є послідовність почергових ділянок границь многокутників P і Q (рис. 11.2). Враховуючи опуклість обох многокутників можна виділити два ланцюги: *зовнішній* та *внутрішній,* які утворюють серповидну область, яка обмежена парою точок перетину (початкова та кінцева). Можна стверджувати наступне: вершина з одного з многокутників належить "серпу", якщо вона лежить між початковою та кінцевою точками або являється кінцем ребра внутрішнього ланцюга, який містить кінцеву точку серпа (рис. 11.3).

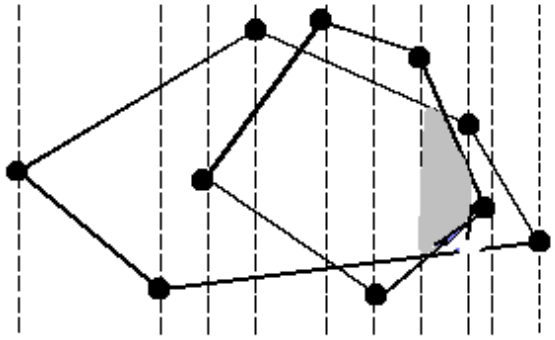


Рис.11.2

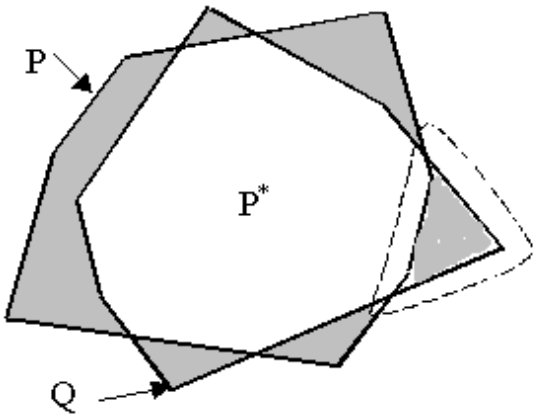
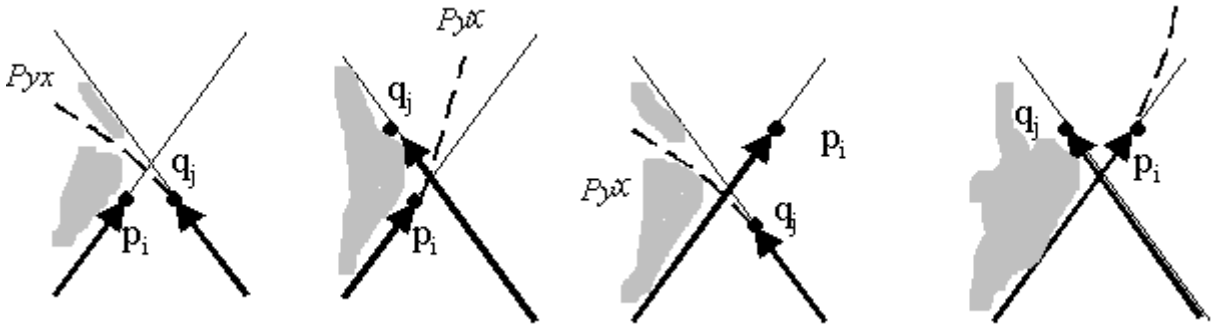


Рис. 11.3

Алгоритм

1. Нехай маємо два упорядкованих списки вершин у вигляді циклічно упорядкованої послідовності проти годинникової стрілки (p_1, p_2, ..., p_L) для P та (q_1, q_2, ..., q_M) для Q.
2. Припустимо, що рух ведеться по обом ланцюгам одночасно і p_i , q_j поточні вершини многокутників, а поточні ребра закінчуються в p_i , q_j.
3. Позначимо через $h(p_i)$ ($h(q_j)$) півплощину визначену (p_{i-1} p_i), яка містить многокутник P(Q). Точка з P^* розташована всередині перетину $h(p_i)$ і $h(q_j)$.
4. Крок просунення. В залежності від розташування поточних вершин і поточного ребра відбувається черговий крок просунення. Можливі чотири основних варіанти розташування (рис. 11.4). Стратегія така: необхідно рухатись так, щоб не потрапити на границю, поточне ребро якої може містити не виявлену точку перетину.
5. Випадки кроку просунення:
   1) вибір довільний і, обирається, наприклад, Q;
   2) просуваємось по P, оскільки поточне ребро (q_{j-1} q_j) із Q може містити невиявлений перетин;
   3) просуваємось по Q, оскільки поточне ребро (p_{i-1} p_i) із P може містити невиявлений перетин;
   4) усі перетини на поточному ребрі з P уже виявлені, в той час як поточне ребро із Q ще може містити невиявлений перетин.



випадок (1) *випадок (2)* *випадок (3)* *випадок (4)*

Рис.11.4

Для не виродженого випадку (коли границя P містить вершину Q і навпаки) можна написати формальну реалізацію:

Procedura (ПОПБУДОВА ПЕРЕТИНУ ОПУКЛИХ МНОГОКУТНИКІВ)
begin *i:= j: = k: = 1:*
repeat
begin if (($p_{i-1}\,p_i$) і ($q_{j-1}\,q_j$) перетинаються) **then**
вивід перетину ;
РУХ (* *і або j* збільшується *);
end;
until *k= 2(L+M);*
if (не знайдено перетинів**) then**
begin if $p_i \in Q$ **then** $P \subseteq Q$
else if $q_j \in P$ **then** $Q \subseteq P$
else $P \cap Q = \varnothing$
end;
end.

11.4.2 Перетин зіркових многокутників.

Результатом перетину зіркових многокутників P і Q взагалі може бути множина многокутників. І якщо P і Q мають по N вершин, то в найгіршому випадку кількість перетинів буде пропорційна N^2.

Теорема 11.4. *Пошук перетинів двох зіркових N – вершинних многокутників потребує у гіршому випадку* $\Omega(N^2)$*часу.*

Звідси випливає, що задача вилучення не видимих ліній теж матиме у гіршому випадку оцінку $\Omega(N^2)$.

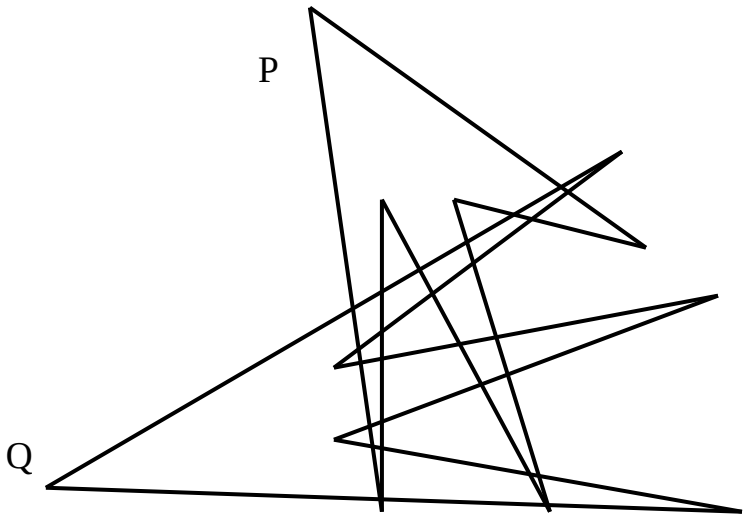


Рис. 11.5 Перетин зіркових многокутників.

11.4.3 Перетин прямолінійних відрізків

Важливим фактом являється те, що набір сформульованих задач можна розв'язати шляхом зведення їх до розв'язання задачі про перетин N прямолінійних відрізків: чи будуть чи ні попарно перетинатися відрізки.

Задача П2.1 (ПЕРЕВІРКА ПЕРЕТИНУ ПРЯМОЛІНІЙНИХ ВІДРІЗКІВ)
Задано N прямолінійних відрізків на площині. Необхідно визначити факт перетину хоча б двох із них.

Задача П2.2 (ПЕРЕВІРКА ПЕРЕТИНУ МНОГОКУТНИКІВ)
Задано два простих многокутники P і Q із M і N вершинами відповідно. Чи перетинаються вони?

Для опуклих многокутників має місце факт: якщо P і Q перетинаються, то або P містить Q або Q містить P, або деяке ребро P перетинає одне із ребер Q. У випадку простих многокутників при довільному перетині обидва ребра будуть належати різним многокутникам. Якщо позначити T(N) як час розв'язання задачі П2.1, то за час T(N+M) можна визначити факт перетину P і Q.Якщо перетинів не знайдено, то лишається перевірка $P \subset Q$ або $Q \subset P$.

Якщо P лежить всередині Q, то усі вершини P лежать всередині Q, один раз застосувати перевірку належності точки за час O(N), використовуючи будь-яку вершину P. Якщо ця вершина за межами Q, то необхідно таким само способом перевірити $Q \subset P$ за час O(N).

Теорема 11.5 *Перетин простих многокутників зводиться за лінійний час до перевірки перетину прямолінійних відрізків.*

ПЕРЕВІРКА ПЕРЕТИНУ МНОГОКУТНИКІВ $\propto_N$ ПЕРЕВІРКА ПЕРЕТИНУ ПРЯМОЛІНІЙНИХ ВІДРІЗКІВ.

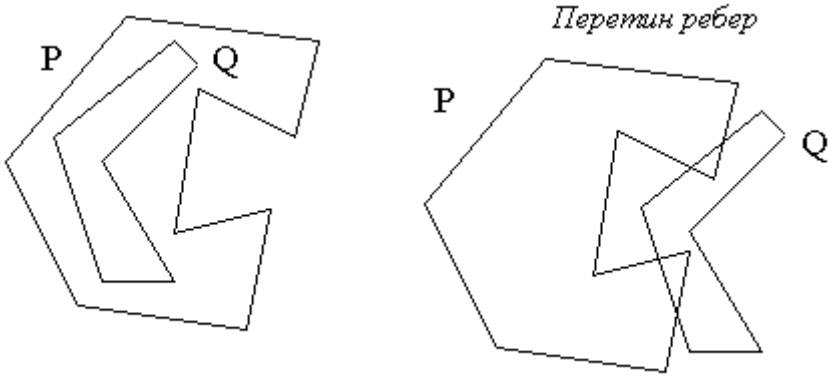
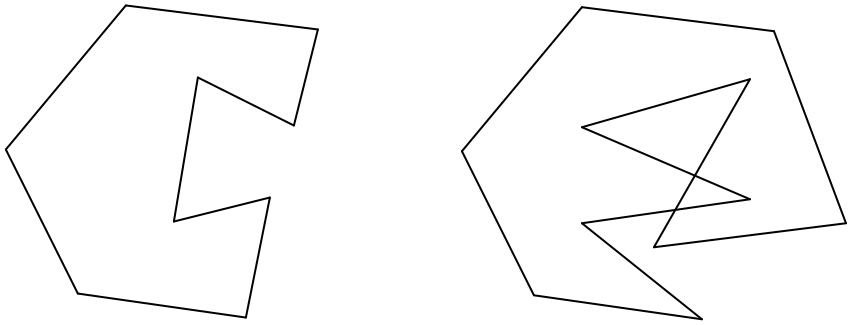


Рис. 11.6

Задача П2.3 (ПЕРЕВІРКА ПРОСТОТИ МНОГОКУТНИКА)
Задано N – кутник. Чи є він простим?

Згідно з означення простого многокутника: многокутник буде простим тоді і лише тоді, коли ніяка пара його не суміжних ребер не перетинається.
Звідси випливає:
ПЕРЕВІРКА ПРОСТОТИ МНОГОКУТНИКА $\propto_N$ ПЕРЕВІРКА ПЕРЕТИНУ ПРЯМОЛІНІЙНИХ ВІДРІЗКІВ.



Простий *Не простий*

Рис. 11.7

11.5 Задачі лінійного програмування та узагальнення перетину.

ЗАДАЧА П4. (ЗАДАЧА ПЕРЕТИНУ ПІВПРОСТОРІВ).

Областю допустимих рішень задач лінійного програмування є перетин півплощин, які визначені множинами їх обмежень. Цільова функція досягає максимуму в одній з вершин опуклого многокутника. Тут немає конкретної множини точок, серед яких необхідно знайти вершини оболонки, замість цього є набір півплощин, які обмежують оболонку, а потрібно знайти вершини. Очевидно, що після побудови загальної області для даних N об'єктів можна отримати розв'язок задачі лінійного програмування. Однак, потрібно знайти лише ту вершину, на якій цільова функція екстремальна (максимальна чи мінімальна).

Теорема: *Задачу лінійного програмування з двома змінними та N обмеженнями можна розв'язати за час O(N log N). Після того як задача розв'язана, екстремум нової цільової функції можна знайти за час O(log N).*

Порівняємо цей підхід із симплекс-методом. Якщо відомий один початковий розв'язок, то симплекс-метод реалізує рух від вершини до вершини по допустимій області, затративши O(N) часу на кожному кроці. Легко можна бачити, що в найгіршому випадку в симплекс-методі потрібно буде пройтись по всім вершинам із затратою загального часу O(N^2). Далі для максимізації нової цільової функції симплекс-метод повинен перевірити кожне обмеження, затративши O(N) часу. Іншими словами симплекс-метод не є оптимальним.

Необхідно зазначити, що метод явної побудови допустимого політопа нереалістичний для задачі лінійного програмування при великих вимірах, оскільки кількість вершин може експоненційно залежати від кількості вимірів.

Одна з вразливих характеристик симплекс-методу полягає в тому, що, хоч і відома його експоненційна залежність від кількості вимірів у найгіршому випадку, на практиці цей алгоритм поводить себе дуже

чудово. Схожу поведінку демонструє і метод, заснований на перетині півплощин для широкого класу вихідних даних. Якщо математичне очікування розмірів розв'язків підзадач мале, то крок злиття в алгоритмі "розподіляй та володарюй" можна реалізувати за менше ніж лінійний час. Це буде мати місце, якщо велика кількість півплощин є *надлишковою*, т.т. не утворює ребер допустимої області. Покажемо, що для випадкових вихідних даних досить велика кількість півплощин є надлишковою.

Інтуїтивно зрозуміло та практично легко уявити довільний природний розподіл імовірності положення випадкових точок на площині. Однак, якщо застосувати фундаментальне геометричне перетворення, то можна зробити наступне спостереження. Кожній прямій поставимо у відповідність півплощину (що містить початок координат). Припустимо, що нам дісталась множина S із N випадкових точок, котрі лежать у якій-небудь області, і ми відображаємо їх на півплощині. Точки, котрі лежать на випуклій оболонці S, перейдуть у таку множину прямих, кожна з яких містить ребро із загальної області перетину цих півплощин. Тому отримаємо, що математичне очікування кількості не надлишкових півплощин у множині з N півплощин для даного випадку дорівнює $O(N^p)$, $p < 1$; схожі результати вірні і для інших, досить природних випадкових моделей. Звідси випливає, що для задачі перетину N півплощин отримуємо лінійну оцінку середньої поведінки алгоритму. А це веде до оцінки O(N) для середньої поведінки алгоритму розв'язку задачі лінійного програмування з двома змінними. Ми бачимо, що математичне очікування кількості надлишкових півплощин - таких, які не визначають ребер допустимої області, - дуже велике, що може пояснити чудову поведінку симплекс-методу.

Однак не можна заспокоювати себе тим, що математичне очікування тимчасової оцінки алгоритму побудови допустимої області пропорційне O(N), оскільки така побудова не є необхідною для розв'язку задачі лінійного програмування. Багато років це спостереження було безрезультатним, до тих пір поки недавно надзвичайно вдалий метод, заснований на ньому, не був незалежно відкритий Меджіддо (1983) та Дайером(1984). Цей метод відкидає не тільки надлишкові обмеження, але також і обмеження, котрі гарантують відсутність вершини, котра дає екстремум цільової функції. Цей метод базується на застосуванні до множини точок на площині такого лінійного перетворення, при якому цільова функція дорівнює одній із двох координат, скажімо, ординаті цієї площини. Після цього задача зводиться до пошуку екстремального значення деякої кусково-лінійної опуклої функції від абсциси. Ключовий момент полягає в тому, що оскільки потрібно визначити лише екстремальне значення Xo, то не має потреби явно будувати цю опуклу функцію, котра неявно задана множиною лінійних обмежень.

Більш детально, вихідну задачу

Мінімізувати $ax + by$

при умовах $a_i x + b_i y + c_i \le 0$, $i = 1,2, ..., N$.

можна перетворити, покладаючи Y = ax + by та X = x, наступним чином (оскільки тут a та b одночасно не дорівнюють нулю, припустимо без утрати загальності, що $b \neq 0$):

Мінімізувати Y

при умовах $\alpha_i X + \beta_i Y + c_i \leq 0$, i= 1, 2,..., N, де $\alpha_I = (a_i - (a/b)* b_i)$, а $\beta_i = b_i/b$.

В новій задачі потрібно обчислити найменше значення Y на вершинах опуклого многокутника P (допустимої області), визначеного цими обмеженнями.

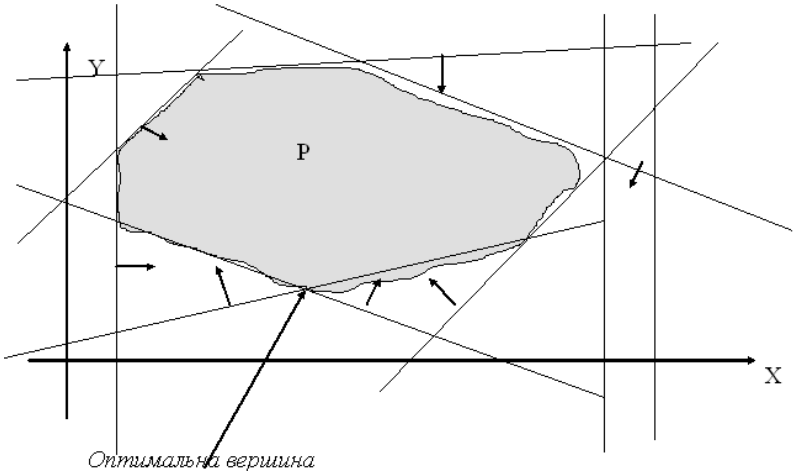


Рис.10.

Щоб уникнути побудови всієї границі P, будемо діяти наступним чином. В залежності від того, чи буде β_I нулем, від'ємним або додатнім числом, розіб'ємо множину індексів {1,…, N} на підмножини I_0, I_-, I_+ відповідно.

Усі обмеження з індексами з I_0 є вертикальними прямими (т.т. паралельні осі Y) і визначають допустимий інтервал для X наступним чином:

$u1 \leq X \leq u2$,

$u1 = \max\{-c_i / \alpha_i : i \in I_0\}$

$u2 = \min\{-c_i / \alpha_i : i \in I_0\}$.

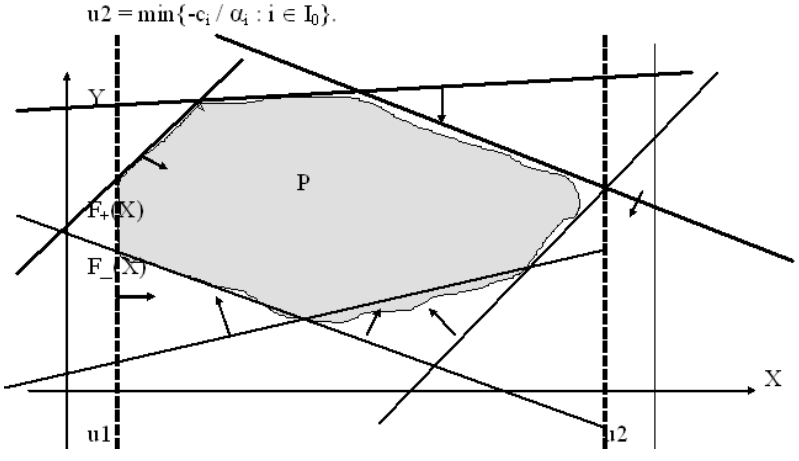


Рис. 11

З іншого боку, покладаючи $\delta i = -(\alpha_i / \beta_i)$ та $\gamma i = -(c_i / \beta_i)$, отримаємо, що всі обмеження з I_+ мають вигляд:

$Y \le \delta iX + \gamma i$, $i \in I_+$,

так що вони, разом узяті, визначають кусочно-лінійну, випуклу вверх ф-ю $F_+(X)$ вигляду:

$$F_+(X) = \min_{i \in I+}(\delta iX + \gamma i).$$

Аналогічно обмеження з I_- у сукупності визначають кусочно-лінійну, випуклу вниз ф-ю $F_-(X)$ вигляду:

$$F_-(X) = \max_{I \in I_-}(\delta iX + \gamma i).$$

Потім отримуємо перетворене обмеження $F_-(X) \le Y \le F_+(X)$, а оскільки розв'язується задача лінійного програмування на min, то $F_-(X)$ і є нашою цільовою ф-ю. Задача має наступний вигляд:

мінімізувати $F_-(X)$

при умовах $F_-(X) \le F_+(X)$,

$u1 \le X \le u2$.

Ця нова ситуація зображена на Рис.2, де показані зв'язки між u1, u2, $F_-(X)$, $F_+(X)$ та границею P.

Елементарною операцією, що використовується в цьому методі, є обчислення ф-й $F_-(x)$, $F_+(x)$ та їх нахилів по обидві сторони від заданого значення $x \in X$. (Позначимо через $fL_-(x)$ та $fR_-(x)$ наклони $F_-(x)$ ліворуч та праворуч від x відповідно; аналогічно визначаються $fL_+(x)$ та $fR_+(x)$.) Покажемо тепер, що цю елементарну операцію, котра називається *обчисленням функцій*, можна виконати за час O(N). Розглянемо, наприклад, функцію $F_-(x)$. Маємо вираз

$$F_-(X) = \max_{I \in I_-}(\delta iX + \gamma i).$$

котрий можна обчислити за час, пропорційний $|I_-| = O(N)$. Якщо існує тільки одне значення i, що дорівнює io, на котрому досягається $F_-(x)$, то $fL_-(x) = fR_-(x) = \delta io$, інакше (якщо існують два таких значення i1 та i2), то $fL_-(x) = \min(\delta i1, \delta i2)$ та $fR_-(x) = \max(\delta i1, \delta i2)$, оскільки $F_-(x)$ випукла вниз.

Справедливе наступне твердження, що для довільного $x \in [u1,u2]$ можна отримати один із наступних результатів за час O(N):

1. x недопустиме, а задача не має розв'язку;
2. x недопустиме, але відомо, по який бік від x (ліворуч або праворуч) можуть лежать всі допустимі зн-ня x;
3. x допустиме, і відомо, по який бік від x лежить min $F_-(x)$;
4. x дає мінімальне значення ф-ї $F_-(x)$.

Дійсно, якщо функція $H(X) = F_-(x) - F_+(x)$ додатна в точці x, то x недопустимо. Розглядаючи нахили $F_-(x)$ та $F_+(x)$ при $X = x$, маємо

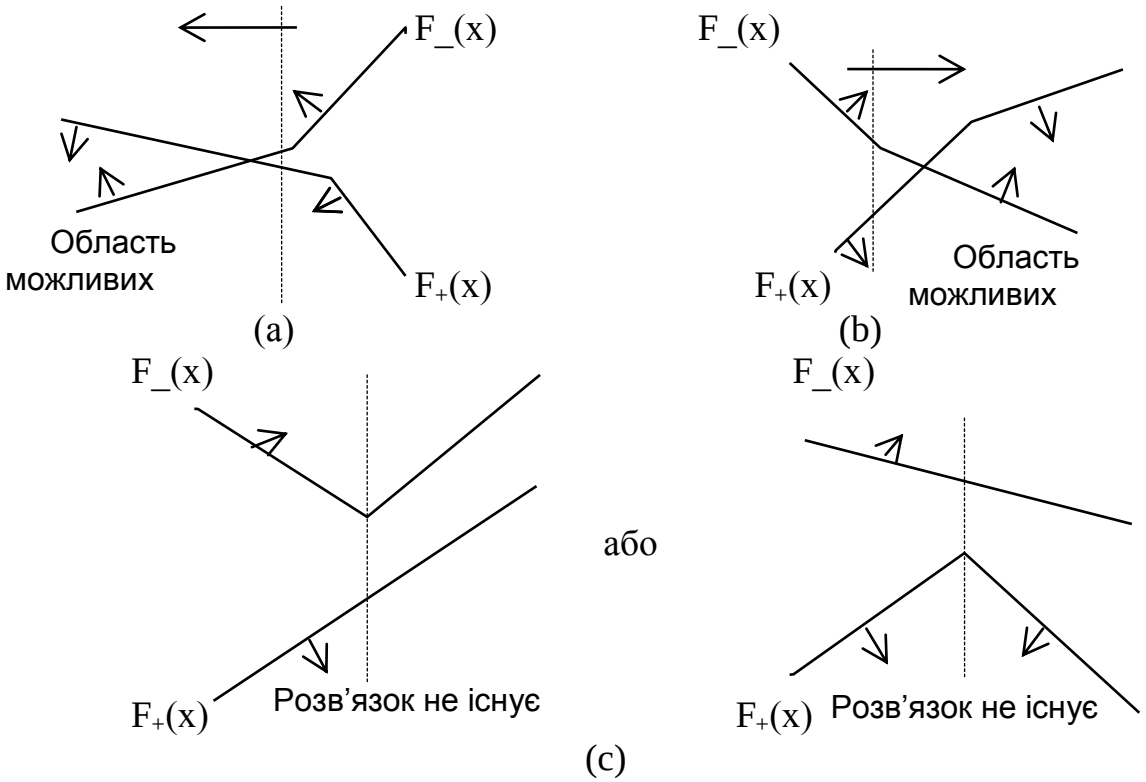


Рис. 12

* якщо $fL\_(X) > fL_+(X)$, то H(X) зростає в точці X, і допустимі значення x можуть знаходитись тільки ліворуч від X (Рис. 3 (a));

* якщо $fR\_(X) < fR_+(X)$, то аналогічно допустимі значення x можуть знаходитись тільки праворуч від X (Рис. 3 (b));

* нарешті, якщо $fL\_(X) \le fL_+(X)$ та $fR\_(X) \ge fR_+(X)$, то функція H(X) досягає min у точці X - задача не має розв'язку (Рис. 3 (c), випадок 1).

* випадок, коли $H(X) \le 0$, т.т точка X є допустимою, розглядається нижче (випадок 3 та 4).

Тепер стратегія розв'язку починає прояснюватися. Будемо намагатися так вибрати абсцису X', в якій виконується обчислення, щоб, якщо алгоритм і не завершується одразу ж, то по меншій мірі фіксовану долю α від кількості активних на даний момент обмежень можна було відкинути (причому, кожне відкинуте обмеження, напевно, не повинно містити крайніх вершин). Якщо мети досягнуто, то після $\log_{1/(1-\alpha)} N$ кроків потужність множини обмежень стає досить малою й прийнятною для прямого розв`язку задачі. При такому припущенні на і-му кроці кількість активних обмежень не перевищує $(1-\alpha)^{i-1}N$ і необхідна обробка завершиться за час, не більший ніж $K(1-\alpha)^{i-1}N$, де К-деяка константа. Тоді сумарний час роботи T(N) оцінюється зверху наступним чином:

$$T(N) \le \sum_{i=1}^{\log_{1/(1-\alpha)}N} K(1-\alpha)^{i-1}N < KN/\alpha,$$ тобто лінійно залежить від N і це оптимальна оцінка.

Розглянувши випадки легко показати, що можна отримати величину $\alpha=1/4$.

Аналіз розв'язку

Згідно теореми про те, що ЗЛП із 2-ма змінними та N обмеженнями можна розв'язати за оптимальний час 0(N), робимо висновок про те, що існує алгоритм, котрий дозволяє це зробити. Розглянемо цей алгоритм.

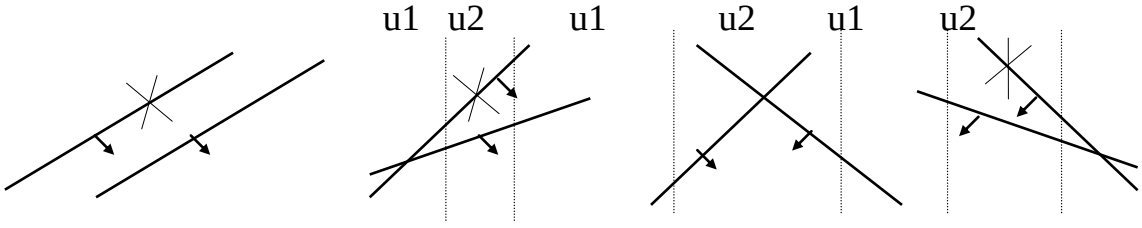
Алгоритм, котрий буде розглянутий нижче, відкидає надлишкові обмеження, а також обмеження, котрі гарантують відсутність вершини, котра дає екстремум цільової ф-ї. Цей метод заснований на застосуванні до точок площини такого лінійного перетворення, при якому цільова ф-я стає рівною одній з двох координат, наприклад, ординаті цієї площини. Після цього задача зводиться до пошуку екстремального значення деякої кусочно-лінійної опуклої ф-ї від абсциси. Ключовий момент полягає у тому, що оскільки вимагається визначити лише екстремальне значення Xo, то не потрібно явно будувати цю опуклу ф-ю, котра неявно задана множиною лінійних обмежень.

Алгоритм розв'язку

Попередня обробка полягає у ітеративному повторюванні наступних кроків:

Крок 1. Розбиваємо множину обмежень зверху на пари обмежень. Визначаємо точки їх перетину. Якщо точка перетину не існує (прямі паралельні), або точка перетину не попадає в допустиму область u1<X<u2, то одне з двох обмежень можна вилучити.

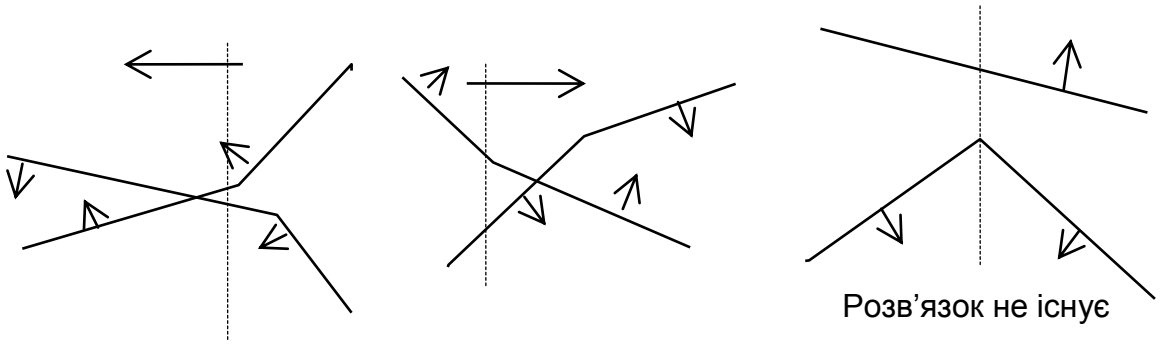
╳ ■ позначення вилучення обмеження.



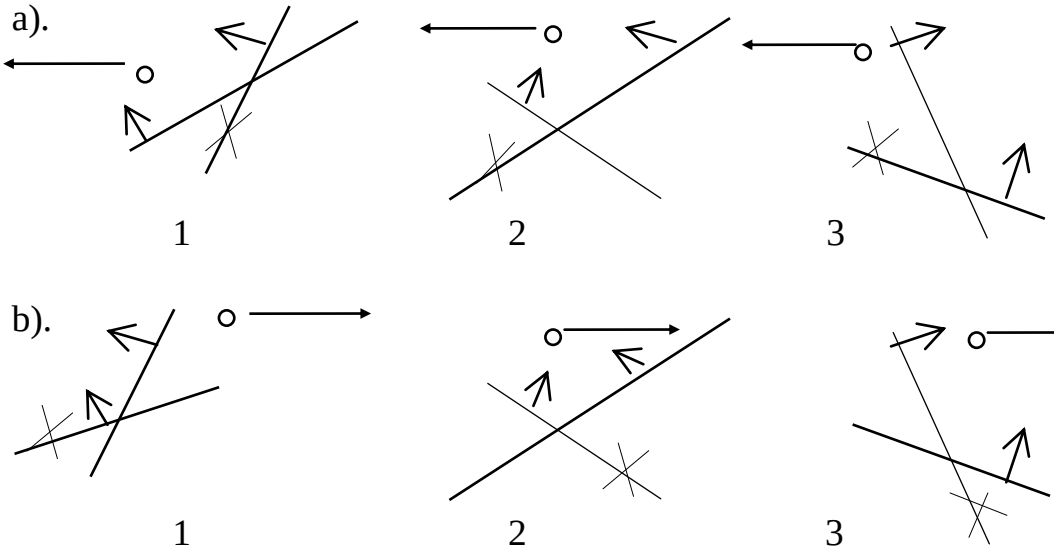
Крок 2. Аналогічно попередньому пункту розбиваємо на пари множину обмежень знизу. Відкидаємо несуттєві обмеження. Ті пари, точка перетину яких попадає в допустиму область, запам'ятовуються (множина K).

Крок 3. Якщо множина K порожня, переходимо до кроку 6. Інакше визначається медіана по X-координаті множини K пар обмежень знизу, створена на попередньому кроці.

Крок 4. Перевіряємо, чи є знайдена медіана множини К точкою оптимуму. Якщо ні, то визначається "ефективна" півплощина, тобто та півплощина, в якій може лежати оптимальна точка.



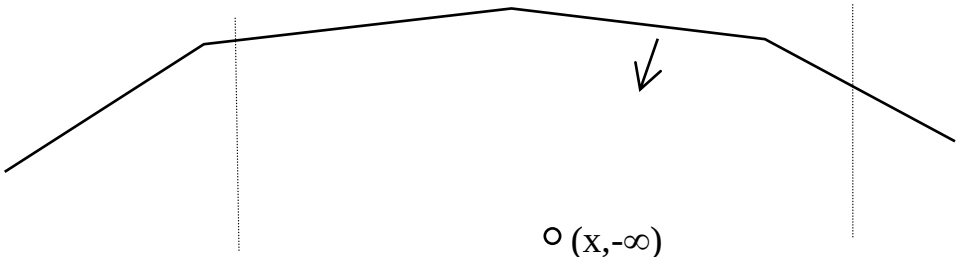
Крок 5. Вилучаємо несуттєві обмеження з пар множини К, точка перетину яких знаходиться у "неефективній" півплощині:



Крок 6. Якщо після виконання кроків 1-4 кількість обмежень не змінилась, або ми виконали $\log_{1/(1-1/4)}N = \log_{4/3}N$ кроків, переходимо до прямого розв'язку задачі. Інакше переходимо до пункту 1.

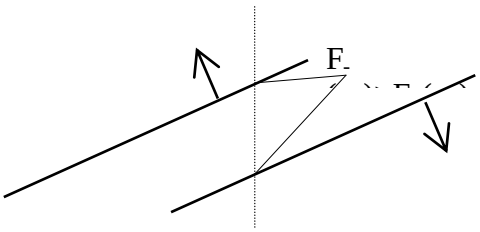
Прямий розв'язок задачі полягає у повному переборі точок перетину обмежень та визначенні оптимальної точки. Алгоритм дій :

У випадку відсутності обмежень знизу, незалежно від наявності чи відсутності обмежень зверху, розв'язком буде точка (x,-∞), де x-довільна точка осі OX.

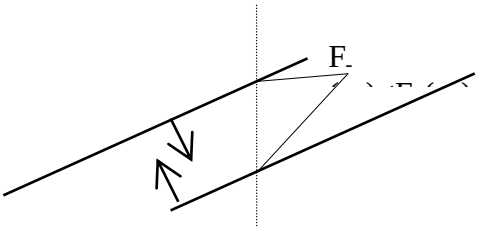


Інакше, для кожного обмеження знизу визначаємо допустимі точки його перетину з кожним з обмежень зверху та з іншими обмеженнями знизу. Для пар обмежень знизу, точка перетину яких є недопустимою, відкидаємо несуттєве, аналогічно кроку 5 попередньої обробки.

У процесі пошуку точок перетину можливі випадки, коли обмеження зверху паралельне обмеженню знизу. Перевіряємо, чи не є ситуація конфліктною. Так, якщо в довільній точці x_0 осі X значення на обмеженні знизу більше значення на обмеженні зверху, то задача не має розв'язку. Інакше продовжуємо прямий розв'язок задачі. Дана пара не впливає на розв'язок.



Конфлікт



Конфлікт відсутній, x_0

У процесі визначення точок перетину обираємо найкращу з цих точок за критерієм $Y \rightarrow \min$.

Розглянемо пряму P - обмеження знизу. Будемо вважати, що P задається рівнянням y=kx+b. Можливі випадки:

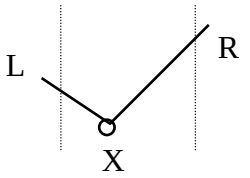
1) k = 0, тоді розв'язком буде довільна точка (x,b), $x \in (u1,u2)$

2) $k \neq 0$, обираємо x - довільна точка осі OX і розглядаємо аналогічно 1.2 чи 1.3, поклавши, що прямі R та L співпадають із P.

Інакше, якщо ми визначили точку, що є кандидатом на оптимум, далі будемо діяти таким чином: аналізуємо нахили ліворуч та праворуч від цієї точки. Можливі такі випадки:

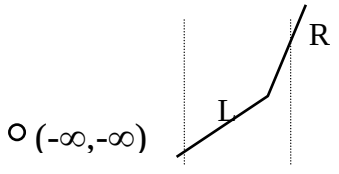
1. Точка, утворена перетином двох обмежень знизу. Тоді

1.1) якщо fL_(X)<0, fR_(X)>0, то це є точка оптимуму

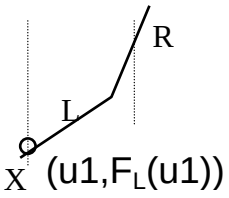


1.2) якщо fL_(X)>0, тоді

1.2.1) якщо u1 = - ∞,,то розв'язком є точка (- ∞,- ∞).

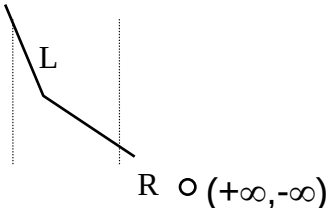


1.2.2) якщо u1 ≠ - ∞,то розв’язком є точка (u1,F_L(u1)).

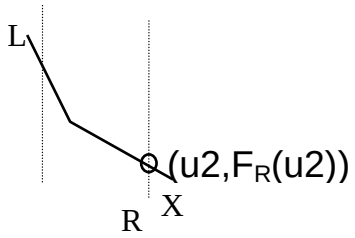


1.3) якщо fR_(X)<0, тоді

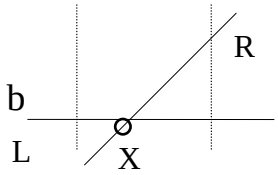
1.3.1) якщо u2 = + ∞, то розв’язком є точка (+ ∞,- ∞).



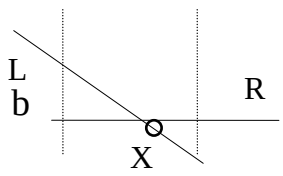
1.3.2) u2 ≠ + ∞, розв’язок - точка (u2,F_R(u2)).



1.4) якщо fL_(X)=0, fR_(X)>0, тоді розв’язком буде довільна точка (x,b), x∈(u1, Xp) u1 u2



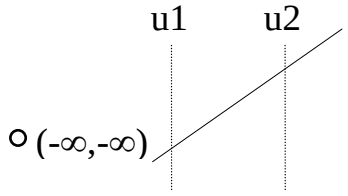
1.5) якщо fL_(X)<0, fR_(X)=0, тоді розв’язком буде довільна точка (x,b), x∈(Xp, u2) u1 u2



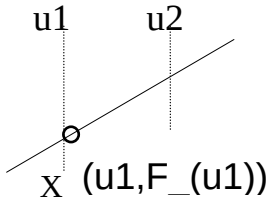
2. Точка, утворена перетином обмеженя знизу з u1 ∨ u2. Тоді

2.1) якщо f_(X)>0, тоді

2.1.1) якщо u1 = - ∞,то розв'язком є точка (- ∞,- ∞).

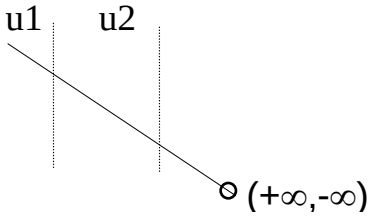


2.1.2) якщо u1 ≠ - ∞,то розв'язком є точка (u1,F_(u1)).

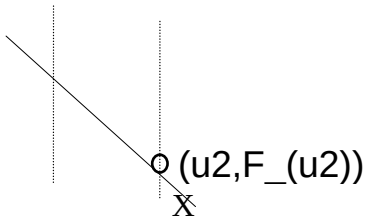


2.2) якщо f_(X)<0, тоді

2.2.1) якщо u2 = + ∞, то розв'язком є точка (+ ∞,- ∞).



2.2.2) u2 ≠ + ∞, розв'язок - точка (u2,F_(u2)).



ЛІТЕРАТУРА

1. Препарата Ф., Шеймос М. Вычислительная геометрия: Введение. - М.: Мир, 1989.
2. Ахо Ф., Хопкрофт Дж., Ултман Дж., Построение и анализ вычислительных алгоритмов. - М: Мир, 1979.
3. Вирт Н. Алгоритмы + структуры данных = программы. - М.: Мир, 1985.
4. Кнут Д. Искусство программирования для ЭВМ. Т. 1.
5. Основные алгоритмы. - М.: Мир, 1976.
6. Кнут Д. Искусство прогрсммирования для ЭВМ. Т. Сортировка и поиск. - М.: Мир, 1978.
7. Фокс А., Пратт М. Вычислительная геометрия. Применение в проектировании и на производстве. - М.: Мир, 1982.
8. Роджерс К. А. Укладки и покрытия. - М.: Мир, 1968.
9. Рейнгольд Э., Нивергельт Ю., Део Н. Комбинаторные алгоритмы: теория и практика.

- М.: Мир, 1980.
10. Препарата Ф., Маллер Д. Нахождение пересечения n подпространств за время O(n log n). Кибернетический сборник, вып. 21. - М.: Мир, 1984.
11. I.J. Balaban. An optimal algorithm for finding segments intersections. *In Proceedings of the Eleventh Annual Symposium on Computational Geometry*, ACM Press, New York, 1995. – pp. 211–219.
12. B. Chazelle. Intersecting is easier than sorting. *In Proceedings of the 16th Annual ACM Symposium on Theory of Computing,* 1984. - pp. 125– 134.
13. B. Chazelle, H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *Journal of the ACM* , 39 (1), 1992. - pp. 1–54.
14. Jan Vahrenhold. Line-segment intersection made in-place. *Computational Geometry* , 38 , 2007. – pp. 213–230.
15. Goodman J.E., O'Rourke J. Handbook of discrete and computational geometry (2ed., CRC, 2004). - pp. 551-554.
16. http://www.algorithmic-solutions.com/leda/
17. Kurt Mehlhorn and Stefan Näher. LEDA—a platform for combinatorial and geometric computing. *Cambridge University Press,* Cambridge, UK, 2009.- pp.733-735.
18. K. Mulmuley. A fast planar partition algorithm, I. *Journal of Symbolic Computation*, 10(3/4), 1990. -pp.:253-280.
19. В. Н. Терещенко, Т. Вознюк. К построению эффективного решения задачи пересечения отрезков // Proceedings “ The 20th International Conference on Computer Graphics and Vision (GraphiCon’2010)”. St. Petersburg, 2010, p. 218 -221.