ПІВНІЧНО-ЗАХІДНИЙ УНІВЕРСИТЕТ
Фатультет комп’ютерної інженерії
та комп’ютерних наук

Алгоритм оптимізації з обмеженим використанням пам’яті для задач з обмеженнями-границями

за

Річардом Х. Бердом (Richard H. Byrd), Пейхуангом Лу (Peihuang Lu),
Джорджем Ноцедалем (Jorge Nocedal) та Цийю Чжу (Ciyou Zhu)

Технічний звіт NAM-08

Перевірено в травні 1994-го року

Резюме

В даній роботі описується алгоритм для розв'язання задач нелінійної оптимізації високої розмірності з простими обмеженнями у вигляді границь. Він базується на методі проекції градієнта та застосовує БФГШ-матрицю з обмеженим використанням пам'яті для апроксимації гесіану цільової функції. Показано, як застосовувати переваги форми апроксимації з обмеженим використанням пам'яті для ефективної реалізації алгоритму. Наведено також результати чисельних тестів на множині задач високої розмірності.

Ключові слова: оптимізація з обмеженнями у вигляді границь, метод з обмеженим використанням пам'яті, нелінійна оптимізація, квазіньютонівський метод, оптимізації функцій високої розмірності.

Скорочений заголовок: метод з обмеженим використанням пам'яті.

1. Вступ

В даній роботі описується квавзіньютонівський алгоритм для розв'язання задач нелінійної оптимізації високої розмірності з простими обмеженнями на змінні у вигляді границь. Дана задача описується наступним чином: (1.1–2)

$$f(x) \to \min \tag{1.1}$$

$$\textit{за умови} \quad l \le x \le u \tag{1.2}$$

де $f : \Re^n \to \Re$ є нелінійною функцією, градієнт g якої доступний, l та u задають верхню і нижню границі змінних, а число змінних n за припущенням достатньо велике. Алгоритм не потребує другої похідної чи знання про структуру цільової функції, і тому може бути застосований, коли обчислення матриці Гессе є не практичним. Крок квавзіньютонівського методу з обмеженим використанням пам'яті застосовується для апроксимації матриці Гессе таким чином, щоб потрібна для збереження матриці пам'ять була пропорційна до n.

Описаний у даній роботі алгоритм подібний до алгоритмів, запропонованих Конном (Conn), Гоулдом (Gould) і Тоінтом (Toint) [9] та Море (Moré) і Торалдо (Toraldo) [20] тим, що в ньому метод проекції градієнту використовується для визначення множини активних обмежень на кожній з ітерацій. Наш алгоритм відрізняється від зазначених методів використанням лінійного пошуку (як альтернативи до методів довірчого регіону), проте основною відмінністю є використання БФГШ-матриці з обмеженим використанням пам'яті для апроксимації гессіана цільової функції. Властивості таких матриць з обмеженим використанням пам'яті мають значні наслідки в реалізації методу, які будуть обговорюватися пізніше. Було встановлено, що грунтуючись на компактному поданні матриць з обмеженим використанням пам'яті, запропанованому Бердом (Byrd), Ноцедалем (Nocedal) та Шнабелем (Schnabel) [6], обчислювальна ціна однієї ітерації алгоритму може бути зведена до порядку n.

Ми використовували метод проекції градієнта [16], [17], [3] для визначення активної множини обмежень, тому що поточні дослідження [7], [5] виявили, що цей підхід володіє добрими теоретичними властивостями, та тому що він виявляється ефективним при розв'язанні багатьох задач високої розмірності [8], [20]. Окрім цього деякі важливі компоненти нашого алгоритму можуть бути корисними і окремо в інших конструкціях, де матриці з обмеженим використанням пам'яті застосовуються для апроксимації гессіана цільової функції.

2. Схема алгоритму

На початку кожної ітерації задані поточне наближення x_k, значення функції f_k, градієнт g_k і додатньовизначене наближення з обмеженним використанням пам'яті B_k. Це дозволяє сфосмувати квадратичну модель f функції у точці x_k:

$$m_k(x) = f(x_k) + g_k^T(x - x_k) + \tfrac{1}{2}(x - x_k)^T B_k(x - x_k) \tag{2.1}$$

Так само, як і в методі дослідженому Конном (Conn), Гоулдом (Gould) та Тоінтом (Toint) [9] алгоритм приблизно мінімізує $m_k(x)$, враховуючи обмеження у вигляді границь (1.2). Це досягається використанням спершу методу проекції градієнта для встановлення множини активних границь, та подальшою мінімізацією $m_k(x)$, трактуючи ці границі як обмеження-рівності.

Для того, щоб це зробири, розглянемо спочатку кусково-лінійний шлях

$$x(t) = P(x_k - tg_k, l, u),$$

отриманий в результаті проекції напрямку найшвидшого спуску на припустимий регіон, де

$$P(x,l,u) = \begin{cases} l_i & \text{якщо } x_i < l_i \\ x_i & \text{якщо } x_i \in [l_i, u_i] \\ u_i & \text{якщо } x_i > u_i \end{cases}. \tag{2.2}$$

Далі обчислемо узагальнену точку Коші (Cauchy point) x^C, визначену як перший локальний мінімум одномірної, кусково-лінійної квадратичнї функції

$$q_k(t) = m_k(x(t)).$$

Змінні, значення яких у точці x^C відповідають нижній чи верхній границі, утворюючи активну множину $A(x^C)$, фіксуються. Далі розглянемо отриману задачу квадратичного програмування стосовно підмножини вільних змінних.

$$\min\{m_k(x) : x_i = x_i^C, \quad \forall i \in A(x^C)\} \tag{2.3}$$

$$\text{за умови} \quad l_i \le x_i \le u_i, \quad \forall i \notin A(x^C) \tag{2.4}$$

Спочатку точно чи наближено розв'яжемо (2.3), ігноруючи накладені на вільні змінні границі, що може бути зроблено як прямими, так і ітеративними методами на підмножині вільних змінних, або двостороннім наближенням, в якому активні границі враховуються через множники Лагранжа. Якщо застосовується ітеративний метод, то точка x^C використовується як початкова точка для даної ітерації. Далі скорочуємо шлях до розв'язку так, щоб задовільнити границям (2.4).

Після отримання наближеного розв'язку $\overline{x_{k+1}}$ цієї задачі, обчислюємо нове наближення x_{k+1} шляхом лінійного пошуку впроховж напрямку $d_k = \overline{x_{k+1}} - x_k$ так, щоб задовільнити на достатню умову спуску:

$$f(x_{k+1}) \le f(x_k) + \alpha_k g_k^T d_k \tag{2.5}$$

та спромогтися виконати вимогу кривизни:

$$\left|g_{k+1}^T d_k\right| \le \beta \left|g_k^T d_k\right| \tag{2.6}$$

де λ_k — довжина кроку, α, β — параметри, що мають в нашому коді значення 10^{-4} і 0,9 відповідно. Метод лінійного пошуку, який гарантує, що наближення залишаються у межах області допустимих розв'язків, описаний у §6. Далі визначаємо градієнт у точці x_{k+1}, обчислюємо нове наближення до гессіану з обмеженим використанням пам'яті B_{k+1} та починаємо нову ітерацію.

Оскільки в нашому алгоритмі кожна апроксимація гессіану B_k додатньовизначена, наближене розв'язання задачі квадратичного програмування (2.3)–(2.4) визначає напрямок спуску $d_k = \overline{x_{k+1}} - x_k$ для цільвої функції f. Для того щоб це побачити, спочатку зауважимо, що узагальнена точка Коші x^C, яка є точкою мінімуму функції $m_k(x)$ на спроектованому напрямку найшвидшого спуску, задовільняє на рівняння $m_k(x_k) > m_k(x^C)$, якщо проекція градієнту ненульова. Оскільки точка $\overline{x_{k+1}}$ знаходиться на шляху від точки x^C до точки мінімуму задачі (2.3), впродовж якого $m_k(x)$ убуває, значення функції m_k в точці x_{k+1} не перевищує її значення в x^C. Тому маємо

$$f(x_k) = m_k(x_k) > m_k(x^C) \geq m_k(\overline{x_{k+1}}) = f(x_k) + g_k^T d_k + \tfrac{1}{2} d_k^T B_k d_k .$$

Ця нерівність передбачає, що $g_k^T d_k < 0$, якщо B_k — додатньовизначена матриця і d_k — ненульовий вектор. У даному документі не наводиться аналіз сходження та дослідження можливості зигзагування. Однак, виходячи з використання методу проекції градієнта при обчисленні кроку, ми вважаємо, що можна застосувати аналіз сходження, подібний до наведеного у роботах [7] та [9], та що зигзагування є проблемоє лише у вироджених випадках.

В нашому алгоритмі використовується апроксимація гессіану B_k БФГШ-матрицями з обмеженим використанням пам'яті (Ноцедаль (Nocedal) [21] та Берд (Byrd), Ноцедаль (Nocedal) і Шнабель (Schnabel) [6]). Навіть якщо ці матриці і не застосовують переваг структурованості задаі, вони потребують лише незначної кількості пам'яті та, як буде показано далі, дозволяють виконати обчислення узагальненої точки Коші і мінімізацію підмножини (*вільних змінних*) за $O(n)$ операцій. Тому новий алгоритм має обчислювальні вимоги, подібні до алгоритму з обмеженим використанням пам'яті (*О-БФГШ* / L-BFGS) для задач без обмежень, описаного Ліу (Liu) і Ноцедалем (Nocedal) [18] та Гілбертом (Gilbert) і Лемарешалем (Lemaréchal) [14].

У наступних трьох розділах детально описуються матриці з обмеженим використанням пам'яті, обчислення узагальненої точки Коші та мінімізація квадратичної функції на підмножині (*вільних змінних*).

3. БФГШ-матриці з обмеженим використанням пам'яті

В нашому алгоритмі БФГШ-матриці з обмеженим використанням пам'яті зображуються в компактній формі, описаній Бердом (Byrd), Ноцедалем (Nocedal) і Шнабелем (Schnabel) [6]. На кожній ітерації алгоритм зберігає незначну кількість, скажімо для визначеності m, коректуючих пар $\{s_i, y_i\}, i = k-1,...,k-m$, де

$$s_k = x_{k+1} - x_k, \quad y_k = g_{k+1} - g_k .$$

Ці коректуючі пари містять інформацію про кривизну функції та разом з БФГШ-формулою визначають матрицю з обмеженим використанням пам'яті на ітерації B_k. Питання в тому, як щонайкраще представити (*подати*) ці матриці без їх явного формування.

В [6] запропоновано використовувати компактну форму (або зовнішній добуток) для того, щоб визначити матрицю з обмеженим використанням пам'яті B_k виходячи з $n \times m$ коректуючих

матриць.

$$Y_k = [y_{k-m},...,y_{k-1}], \quad S_k = [s_{k-m},...,s_{k-1}] \tag{3.1}$$

Точніше, в [6] показано, що якщо θ — додатній параметер маштабування, та якщо m коректуючих пар $\{s_i, y_i\}_{i=k-1}^{k-m}$ задовільняють на вимогу $s_i^T y_i > 0$, то матриця, отримана оновленням θI через використання m-разів БФГШ-формули та пар $\{s_i, y_i\}_{i=k-1}^{k-m}$, може бути записана як

$$B_k = \theta I - W_k M_k W_k^T \tag{3.2}$$

де

$$W_k = \begin{bmatrix} Y_k & \theta S_k \end{bmatrix}, \tag{3.3}$$

$$M_k = \begin{bmatrix} -D_k & L_k^T \\ L_k & \theta S_k^T S_k \end{bmatrix}^{-1}, \tag{3.4}$$

і де L_k та D_k — $m \times m$ матриці

$$(L_k)_{i,j} = \begin{cases} (s_{k-m-1+i})^T (y_{k-m-1+j}), & \textit{якщо } i > j \\ 0, & \textit{у протилежному випадку} \end{cases}, \tag{3.5}$$

$$D_k = diag[s_{k-m}^T y_{k-m},...,s_{k-1}^T y_{k-1}]. \tag{3.6}$$

(Слід зауважити, що формула (3.2) містить лише незначні зміни відносно формули (3.5) в [6]). Зазначимо також, що оскільки матриця M_k має розмірність $2m \times 2m$, а число m обирається невеликим цілим, то витратами на обчислення зворотньої матриці в (3.4) можна нехтувати. В [6] показано, що при використанні компактного подання (3.2) різноманітні обчислення B_k стають „недорогими”. Зокрема, добуток матриці B_k на вектор, який часто зустрічається в алгоритмі, наведеному в даному документі, може розраховуватися (*виконуватися*) ефективно.

Існує подібне подання зворотньої БФГШ-матриці з обмеженим використанням пам’яті H_k, яка апроксимує матрицю, зворотню до гессіану:

$$H_k = \frac{1}{\theta} I - \overline{W_k}\,\overline{M_k}\,\overline{W_k}^T, \tag{3.7}$$

де

$$\overline{W_k} = \begin{bmatrix} \frac{1}{\theta} Y_k & S_k \end{bmatrix},$$

$$\overline{M_k} = \begin{bmatrix} 0 & R_k^{-1} \\ -R_k^{-T} & R_k^{-T}(D_k + \frac{1}{\theta} Y_k^T Y_k) R_k^{-1} \end{bmatrix}^{-1},$$

та

$$(R_k)_{i,j} = \begin{cases} (s_{k-m-1+i})^T (y_{k-m-1+j}), & \text{якщо } i \le j \\ 0, & \text{у протилежному випадку} \end{cases} . \tag{3.8}$$

(Слід зауважити, що рівняння (3.7) містить лише незначні зміни відносно рівняння (3.1) в [6]).

Також існують подання матриць з обмеженим використанням пам'яті, аналогічні до наведеного, для інших квазіньютонівських методів, таких як СР1 (SR1) та ДФП (DFP) (див. [6]), які в прнципі можна використовувати при визначенні B_k в нашому алгоритмі. Тут розглядається лише БФГШ-метод, тому що наш значний досвід у розв'язанні задач (*оптимізації*) без обмежень вказує на те, що БФГШ-метод є вдалим (*виконується добре*).

Оскільки обмеження-границі задачі можуть заважати виконанню вимоги (2.6) при лінійному пошуку (див. §6), то неможливо гарантувати, що умова $s_k^T y_k > 0$ завжди буде виконуватися (обговорюється Деннісом (Dennis) та Шнабелем (Schnabel) [12]). Тому для підтримки додатньовизначеності БФГШ-матриці з обмеженим використанням пам'яті, коректуюча пара $\{s_i, y_i\}, i = k-1,...,k-m$ не враховується, якщо умова кривизни

$$s_k^T y_k > eps\|y\|^2 \tag{3.9}$$

не задовільняється для малої додатньої константи eps. Якщо це трапляється, ми не видаляємо найстарішу коректуючу пару, як зазвичай робиться на кроці алгоритму з обмеженим використанням пам'яті. Це значить, що серед m-напрямків у S_k та Y_k фактично можуть бути деякі напрямки з індексами меншими за $k-m$.

4. Узагальнена точка Коші.

Мета процедури, описаної в даному розділі, полягає у знаходженні першого локального мінімуму квадратичної моделі впродовж кусково-лінійного шляу, отриманого проектуванням точок напрямку найшвидшого спуску, $x_k - tg_k$, на область допустимих розв'язків. Позначимо $x^0 = x_k$ та, впродовж даного розділу, опускатимумо індекс зовнішніх ітерацій k так, щоб g, x і B позначали g_k, x_k і B_k. Використовуватимо нжні індекси для позначення компонент вектора; наприклад g_i позначає i–тий елемент вектора g. Верхні індекси використовуватимемо для позначення ітерацій впродовж кусково-лінійного пошуку точки Коші.

Для визначення точок зупинки (*зламу кривої*) по кожному з координатних напрямків обчислюються:

$$t_i = \begin{cases} (x_i^0 - u_i)/g_i & \text{якщо } g_i < 0 \\ (x_i^0 - l_i)/g_i & \text{якщо } g_i > 0 \\ \infty & \text{у протилежному випадку} \end{cases} . \tag{4.1}$$

та сортуться $\{t_i, i = 1,...,n\}$ у зростаючому порядку для того, щоб отримати впорядковану множину $\{t^j : t^j \le t^{j+1}, j = 1,...,n\}$. Це досягаєтьс шляхом використання алгоритму пірамідального сортування [1]. Далі робиться пошук впродовж кусково-лінійного шляху $P(x^0 - tg, l, u)$, який можна зобразити формулою:

$$x(t_i) = \begin{cases} x_i^0 - tg_i & \text{якщо } t \le t_i \\ x_i^0 - t_i g_i & \text{у протилежному випадку} \end{cases}$$

Припустимо, що розглядається інтервал $\left[t^{j-1}, t^{j}\right]$. Визначимо i–у точоку зупинки (*зламу кривої*) як

$$x^{j-1} = x(t^{j-1})$$

така, що на $\left[t^{j-1}, t^{j}\right]$

$$x(t) = x^{j-1} + \Delta t d^{j-1},$$

де

$$\Delta t = t - t^{j-1}$$

та

$$x(t_i) = \begin{cases} x_i^0 - tg_i & \text{якщо } t \le t_i \\ x_i^0 - t_i g_i & \text{у протилежному випадку} \end{cases}. \tag{4.2}$$

Використовуючи цю нотацію запишемо квадратичну функцію (2.1) на лінійному сегменті $\left[x(t^{j-1}), x(t^{j})\right]$, як

$$\begin{aligned} m(x) &= f + g^T(x - x^0) + \tfrac{1}{2}(x - x^0)^T B(x - x^0) \\ &= f + g^T(z^{j-1} + \Delta t d^{j-1}) + \tfrac{1}{2}(z^{j-1} + \Delta t d^{j-1})^T B(z^{j-1} + \Delta t d^{j-1}), \end{aligned}$$

де

$$z^{j-1} = x^{j-1} - x^0. \tag{4.3}$$

Тому на лінійному сегменті $\left[x(t^{j-1}), x(t^{j})\right]$ функція $m(x)$ може бути записана як квадратична відносно Δt,

$$\begin{aligned} m(t) &= (f + g^T z^{j-1} + \tfrac{1}{2} z^{j-1^T} B z^{j-1}) + (g^T d^{j-1} + d^{j-1^T} B z^{j-1})\Delta t + \tfrac{1}{2}(d^{j-1^T} B d^{j-1})\Delta t^2 \\ &\equiv f_{j-1} + f_{j-1}^{'}\Delta t + \tfrac{1}{2} f_{j-1}^{''}\Delta t^2 \end{aligned}$$

де параметри цієї квадратичної функції такі

$$f_{j-1} = f + g^T z^{j-1} + \tfrac{1}{2} z^{j-1^T} B z^{j-1}$$

$$f_{j-1}^{'} = g^T d^{j-1} + d^{j-1^T} B z^{j-1} \tag{4.4}$$

$$f_{j-1}^{''} = d^{j-1^T} B d^{j-1} \tag{4.5}$$

Диференціюючи $m(t)$ та прирівнюючи до нуля, отримуємо $\Delta t^* = -f_{j-1}^{'} / f_{j-1}^{''}$. Коли матриця B додатньовизначена, це значить, що передбачений мінімум $t^{j-1} + \Delta t^*$ лежить на інтервалі $\left[t^{j-1}, t^{j}\right)$. У протилежному випадку узагальнена точка Коші співпадає з $x(t^{j-1})$, якщо $f_{j-1}^{'} \ge 0$, та поза або на $x(t^{j})$, якщо $f_{j-1}^{'} < 0$.

Якщо узагальнена точка Коші не була знайдена після дослідження інтервалу $\left[t^{j-1}, t^{j}\right]$, ми

встановлюємо

$$x^j = x^{j-1} + \Delta t^{j-1} d^{j-1}, \quad \Delta t^{j-1} = t^j - t^{j-1}, \tag{4.6}$$

та оновлюємо похідні за напрямком $f_j^{'}$ і $f_j^{''}$ так, щоб пошук перейшов на наступний інтервал. Припустимо ненадовго, що лише одна змінна стає активною у точці t^j та позначимо індекс цієї змінної через b. Тоді , $t_b = t^j$ і можна обнулити відповідний елемент напрямку пошуку,

$$d^j = d^{j-1} + g_b \varepsilon_b, \tag{4.7}$$

де ε_b — b-ий одиничнй вектор. З формул (4.3) і (4.6) випливає

$$z^j = z^{j-1} + \Delta t^{j-1} d^{j-1}. \tag{4.8}$$

Тому, використовуючи (4.4), (4.5), (4.7) та (4.8), отримуємо

$$\begin{aligned} f_j^{'} &= g^T d^j + d^{jT} B z^j \\ &= g^T d^{j-1} + g_b^2 + d^{j-1T} B z^{j-1} + \Delta t^{j-1} d^{j-1T} B d^{j-1} + g_b \varepsilon_b^T B z^j \\ &= f_{j-1}^{'} + \Delta t^{j-1} f_{j-1}^{''} + g_b^2 + g_b \varepsilon_b^T B z^j \end{aligned} \tag{4.9}$$

та

$$\begin{aligned} f_j^{''} &= d^{jT} B d^j \\ &= d^{j-1T} B d^{j-1} + 2 g_b \varepsilon_b^T B d^{j-1} + g_b^2 \varepsilon_b^T B \varepsilon_b \\ &= f_{j-1}^{''} + 2 g_b \varepsilon_b^T B d^{j-1} + g_b^2 \varepsilon_b^T B \varepsilon_b \end{aligned} \tag{4.10}$$

Єдиними трудомісткими операціями в формулах (4.9) і (4.10) є

$$\varepsilon_b^T B z^j, \quad \varepsilon_b^T B d^{j-1}, \quad \varepsilon_b^T B \varepsilon_b,$$

які можуть потребувати $O(n)$ операцій, оскільки B — плотна матриця з обмеженим використанням пам’яті. Таким чином виявляється, що обчслення узагальненої точки Коші може потребувати $O(n^2)$ операцій, оскільки в найгіршому випаду можуть перевірятися усі n сегментів кусково-лінійного шляху. Така трудомісткість може виявитися надмірно високою для задач високої розмірності. Однак підстановка БФГШ-формули з урахуванням обмеженого використання пам’яті (4.2) у формули (4.9)-(4.10) дає

$$f_j^{'} = f_{j-1}^{'} + \Delta t^{j-1} f_{j-1}^{''} + g_b^2 + \theta g_b z_b^j - g_b w_b^T M W^T z^j \tag{4.11}$$

$$f_j^{''} = f_{j-1}^{''} - 2\theta g_b^2 - 2 g_b w_b^T M W^T d^{j-1} + \theta g_b^2 - g_b w_b^T M w_b \tag{4.12}$$

де w_b^T позначає b-ий рядок матриці W. Єдиними випадками, де в формулах (4.11) і (4.12) необхідно виконувати $O(n)$ операцій, є $W^T z^j$ та $W^T d^{j-1}$. Виходячи з (4.7) та (4.8), слід однак зауважити, що на кожній ітерації z^j і d^j оновлюються шляхом простих обчислень. Тому, якщо зберігати два $2m$-мірних вектори

$$p^j \equiv W^T d^j = W^T (d^{j-1} - g_b \varepsilon_b) = p^{j-1} + g_b w_b,$$

$$c^j \equiv W^T z^j = W^T (z^{j-1} + \Delta t^{j-1} d^{j-1}) = c^{j-1} + \Delta t^{j-1} p^{j-1},$$

то оновлення $f_j^{'}$ і $f_j^{''}$ з використанням виразів

$$f_j^{'} = f_{j-1}^{'} + \Delta t^{j-1} f_{j-1}^{''} + g_b^2 + \theta g_b z_b^j - g_b w_b^T M c^j$$

$$f_j^{''} = f_{j-1}^{''} - 2\theta g_b^2 - 2 g_b w_b^T M p^{j-1} + \theta g_b^2 - g_b w_b^T M w_b$$

потребуватие лише $O(m^2)$ операцій. Якщо більш ніж одна змінна становиться активною у точці , що є нетиповим, ми повторюємо щойно описаний процес оновлення, перевіривши до цього новий інтервал $\left[t^{j-1}, t^j\right]$. Таким чином ми спромогаємося досягти додаткового зменшення трудомісткості обчислення узагальненої точки Коші.

Зауаження. *Перевірка першого сегменту спроектованого шляху найшвидшого спуску під час обчислення узагальненої точки Коші потребує* $O(n)$ *операцій. Хоча усі наступні перевірки сегментів потребують лише* $O(m^2)$ *операцій, де* m *— число коректуючих векторів, які зберігаються у матриці з обмеженим використанням пам'яті.*

Оскільки m зазвичай невелике число, скажімо менше за 10, трудомісткістю перевірок усіх сегментів, що прямують за першим, можна нехтувати. Наведений нижче алгоритм описує більш детально, як цього досягти в обчисленнях. Зауважимо, що необов'язково відсліджувати n-мірний вектор z^j, оскільки лише елемент z_b^j, яки відповідає активній границі, потрібний для оновлення $f_j^{'}$ і $f_j^{''}$.

Алгоритм CP (*copy-protected*)**: Обчислення узагальненої точки Коші.**

Дані: x, l, u, g та $B = \theta I - WMW^T$

1 ***for*** $i = \overline{1,n}$ ***do***

обчислити

$$t_i := \begin{cases} (x_i^0 - u_i)/g_i & \text{якщо } g_i < 0 \\ (x_i^0 - l_i)/g_i & \text{якщо } g_i > 0 \\ \infty & \text{у протилежному випадку} \end{cases}$$

(n операцій)

$$d_i := \begin{cases} 0 & \text{якщо } t_i = 0 \\ -g_i & \text{у протилежному випадку} \end{cases}$$

end for

2 Ініціалізувати

$F := \{i : t_i > 0\}$

$p := W^T d$ ($2mn$ операцій)

$c := 0$

$f^{'} := g^T d = -d^T d$ (n операцій)

$f^{''} := \theta d^T d - d^T W M W^T d = -\theta f^{'} - p^T M p$ ($O(m^2)$ операцій)

Алгоритм CP (*copy-protected*)**: Обчислення узагальненої точки Коші .**

$\Delta t_{\min} := -\dfrac{f'}{f''}$

$t_{old} := 0$

$t := \min\{t_i : i \in F\}$ (використовується алгоритм пірамідального сортування)

$b := i$ *таке, що* $t_i = t$ (видалити b з F)

$\Delta t := t - 0$

3 Перевірити наступний сегмент

while $\Delta t_{\min} \geq t$ ***do***

$$x_b^{cp} := \begin{cases} u_b & \text{якщо } d_b > 0 \\ l_b & \text{якщо } d_b < 0 \end{cases}$$

$z_b := x_b^{cp} - x_b$

$c := c + \Delta t p$ ($O(m)$ операцій)

$f' = f' + \Delta t f'' + g_b^2 + \theta g_b z_b - g_b w_b^T M c$ ($O(m^2)$ операцій)

$f'' = f'' - \theta g_b^2 - 2 g_b w_b^T M p - g_b w_b^T M w_b$ ($O(m^2)$ операцій)

$p = p + g_b w_b$ ($O(m)$ операцій)

$d_b := 0$

$\Delta t_{\min} := -\dfrac{f'}{f''}$

$t_{old} := t$

$t := \min\{t_i : i \in F\}$ (використовується алгоритм пірамідального сортування)

$b := i$ *таке, що* $t_i = t$ (видалити b з F)

$\Delta t := t - t_{old}$

end while

4 $\Delta t_{\min} := \max\{\Delta t_{\min}, 0\}$

5 $t_{old} := t_{old} + \Delta t_{\min}$

6 $x_i^{cp} := x_i + t_{old} d_i, \quad \forall i$ *таких, що* $t_i \geq t$

7 ***for each*** $i \in F : t_i = t$ ***do***

видалити i з F

end for

8 $c := c + \Delta t_{\min} p$

На останньому кроці алгоритму оновлюється $2m$-мірний вектор c так, щоб перед виходом з алгоритму виходило:

$$c = W^T (x^C - x_k) \tag{4.13}$$

Цей вектор використовуватиметься для ініціалізації підпростору (*області*) мінімізації при застосуванні примітивного прямого методу чи методу сполучених градієнтів, про що йтиме мова у

наступному розділі.

При наведенні кількості виконуваних операцій беруться до уваги лише операції множення та ділення. Слід зауважити, що цикл містить не $O(n)$ обчислень. Якщо символом n_{int} позначити загальну кількість сегментів, що досліджуються, тоді сумарна трудомісткість алгоритму **CP** складає $(2m+2)n + O(m^2) \times n_{\text{int}}$ операцій плюс приблизно $n \log n$ операцій, які виконує алгоритм пірамідального сортування.