Київський національний університет

імені Тараса Шевченка

Факультет комп'ютерних наук та кібернетики

Кафедра обчислювальної математики

Кваліфікаційна робота

на здобуття ступеня бакалавра

за спеціальністю 113 Прикладна математика

на тему:

Децентралізовані алгоритми для варіаційних нерівностей

Виконала студентка 4-го курсу

Пірковець Катерина Геннадіївна _____

Науковий керівник:

доктор фіз.-мат. наук, професор

Семенов Володимир Вікторович _____

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Роботу розглянуто й допущено до захисту на засіданні кафедри обчилювальної математики

«___» _____ 2021 р.,

протокол № ___

Завідувач кафедри

С. І. Ляшко _____

Київ – 2021

РЕФЕРАТ

Обсяг роботи 26 сторінок, 5 ілюстрацій, 6 таблиць, 5 джерел посилань, 1 додаток.

Ключові слова: ВАРІАЦІЙНА НЕРІВНІСТЬ, ДЕЦЕНТРАЛІЗОВАНИЙ АЛГОРИТМ, МОНОТОННИЙ ОПЕРАТОР, ЕКСТРАГРАДІЄНТНИЙ МЕТОД.

Об’єктом роботи є децентралізовані алгоритми розв’язання варіаційних нерівностей з монотонними операторами та опукло-угнутих сідлових задач. Предметом роботи порівняльний аналіз ефективності децентралізованих алгоритмів.

Мета даної роботи полягає в розробці децентралізованих аналогів відомих екстраградієнтних алгоритмів для монотонних варіаційних нерівностей та опукло-угнутих сідлових задач, а також в порівнянні їх ефективності в розподілених системах різних структур. В якості початкових методів обрані наступні: алгоритм Корпелевич, алгоритми Попова та Tseng’a.

Методи розробки: комп’ютерне моделювання, розробка програмного продукту. Інструменти розробки: мова програмування python, бібліотеки mpi4py, multiprocessing і numpy, середа розробки Pycharm.

Результати роботи: розроблено програмний засіб розв’язання варіаційних нерівностей за допомогою децентралізованих варіантів алгоритмів Корпелевич, Попова та Tseng’a, виконано порівняльний аналіз роботи програмного засобу на тестових задачах, проведено оцінку та аналіз результатів.

ЗМІСТ

ВСТУП

Суттєва частка задач дослідження операцій, математичної економіки, теорії управління та інші можуть бути подані в формі задач про рівновагу (задач рівноважного програмування), для розв’язання яких сьогодні існує безліч методів, зокрема методів градієнтного типу. Переважно надають увагу частковому, але найбільш важливому випадку варіаційних нерівностей. Відомо, що при неоптимізаційних постановках для збіжності найбільш простих градієнтних методів необхідно виконання посилених умов монотонності. Збіжність без модифікацій задачі забезпечується в задачах екстраградієнтного типу [1].

Загальним недоліком класичних ітераційних методів є послідовне виконання ітерацій, що призводить до неефективного використання ресурсів сучасних обчислювальних машин. Випереджальний розвиток високопродуктивної обчислювальної техніки в порівнянні з розвитком програмного забезпечення робить затребуваними методи і алгоритми, що використовують кілька процесорів. У зв'язку з цим розробка децентралізованих методів та алгоритмів розв'язання, а також децентралізація існуючих алгоритмів є актуальними напрямами в розвитку сучасних технологій математичних методів моделювання складних оптимізаційних систем і ітераційних методів для їх розв’язання.

З розвитком таких технологій, як Blockchain та хмарні обчислення, децентралізовані мережі стали зростаючою тенденцією в сучасному бізнес-середовищі. Наразі багато організацій намагаються знайти застосування таких систем. Візьмемо для прикладу Bitcoin, оскільки це найпопулярніший варіант їх використання. Жодна організація не володіє мережею Bitcoin, а мережа – це сума всіх вузлів, які спілкуються між собою для підтримання суми Bitcoin, яку має кожен власник рахунку.

З точки зору обчислень, децентралізована мережева архітектура розподіляє навантаження між кількома машинами, замість того, щоб покладатися на єдиний центральний сервер. Ця тенденція еволюціонувала завдяки стрімкому прогресу настільних та портативних комп'ютерів, які зараз володіють продуктивністю, що значно перевищує потреби більшості бізнес-додатків; це означає, що залишкову обчислювальну потужність можна використовувати для розподіленої обробки.

Децентралізовані мережі пропонують широкий спектр переваг порівняно зі звичайними централізованими мережами, як-от підвищену надійність системи, масштабність та конфіденційність.

Однією з найважливіших переваг децентралізованого управління мережею є той факт, що немає жодної реальної точки відмови – це тому, що машини окремих користувачів не залежать від єдиного центрального сервера для обробки всіх процесів. Децентралізовані мережі також набагато легше масштабувати, оскільки користувач може просто додати до мережі більше машин, щоб підвищити обчислювальну потужність.

Крім цього, децентралізована мережева архітектура забезпечує більшу конфіденційність, оскільки інформація збирається не в одній точці, а натомість проходить через безліч різних точок. Це значно ускладнює відстеження в мережі.

Однак негативним фактором є те, що для децентралізованих мереж потрібно більше машин, а це означає більше ресурсів для обслуговування та потенційних проблем, що, в свою чергу, завдає додаткове навантаження на IT-ресурси користувача.

Застосування децентралізованих систем:

- Приватні мережі – однорангові вузли, об'єднані між собою для створення мережі.

- Криптовалюта – вузли приєднуються, щоб стати частиною системи, в якій обмінюються цифрові валюти без будь-яких слідів та місцезнаходження тих, хто проводить операцію. Ми можемо побачити публічну адресу та кількість переданих біткойнів, проте ці публічні адреси можна змінювати і, отже, важко відстежити.

Мета даної роботи полягає в розробці децентралізованих варіантів обраних екстраградієнтних алгоритмів для монотонних варіаційних нерівностей та сідлових задач, а також порівнянні їх ефективності в розподілених системах різних структур. В якості початкових методів обрані наступні: Корпелевич, Попова та Tseng.

Актуальність використання варіаційних нерівностей як сучасного інструменту моделювання і чисельного розв’язку оптимізаційних задач, обумовлена їх універсальністю та застосовністю при розв’язанні задач дослідження операцій в економіці, логістиці, техніці та інших сферах.

ЗАДАЧА

З появою величезного різноманіття напрямів та широкого вибору літератури, що пов’язана з варіаційними нерівностями, тема стала класичною складовою при вивченні часткових диференційних рівнянь. Ось декілька захоплюючих напрямків, де використовуються варіаційні нерівності [2]:

- Фінансова сфера.

  Американський пут-опціон після зміни змінних перетворюється на однофазову проблему Стефана. Більш загально, можна сказати, що будь-яка проблема похідної безпеки там, де є функція раннього вправлення, містить в собі задачу без обмежень.
- Фазові перетворення.

  Рівняння Аллена-Кана, одна з моделей кінетики росту зерен в полікристалах, трактується як параболічна варіаційна нерівність. Також варто згадати проблему надпровідних вихорів.
- Системи та контактні проблеми.

  Проблеми статичного контакту, проблеми фрикційного контакту та питання, пов’язані з тепловим розширенням, можуть розглядатися як варіаційні нерівності, часто для опису систем, або для їх узагальнення.

Технічно кажучи, варіаційна нерівність є варіаційною задачею або еволюційною задачею з опуклими обмеженнями. Оскільки множина, де обмеження є активними, заздалегідь невідома, задача з вільно межею виникає досить часто, і, дійсно, варіаційні нерівності нерозривно пов’язані із задачами з вільними обмеженнями.

Постановка задачі

Розглянемо деякий оператор A, що діє на підмножині C гільбертового простору H.

Означення 1.

Кажуть, що для точки $x \in C$ виконується *варіаційна нерівність*, якщо

$$\langle A(x), y - x\rangle \geq 0, \ \forall y \in C. \tag{1.1}$$

Твердження 1.

У випадку $C = H$ виконання варіаційної нерівності для точки x рівносильне виконанню рівності $A(x) = 0$.

Твердження 2.

Для задачі

$$f \to \min_{C} \tag{1.2}$$

у випадку опуклості як f так і C критерієм того, що точка x є розв'язком є виконання нерівності

$$\langle f'(x), y - x\rangle \geq 0, \ \ \forall y \in C. \tag{1.3}$$

Доведення.

Запишемо лінійну апроксимацію f:

$$f(y) = f(x) + \langle f'(x), y - x\rangle + o(\|y - x\|).$$

Припустимо тепер, що другий доданок менше нуля для якогось $y = x + z$, тоді

$$f(x + z) - f(x) = \langle f'(x), z\rangle + o(\|z\|).$$

Розглянемо (з опуклості C випливає, що $x + \varepsilon z \in C$, а отже можемо підставляти таке y [3]) тепер $y = x + \varepsilon z$, отримаємо

$$f(x+\varepsilon z)-f(x)=\langle f'(x),\varepsilon z\rangle+o(\|\varepsilon z\|).$$

З визначення $o(\cdot)$ зрозуміло, що при $\varepsilon\to+0$ знак правої частини визначає перший доданок.

Тобто, права частина буде від'ємною для якогось достатньо малого ε. Але тоді від'ємною буде і ліва частина, $f(x+\varepsilon z)-f(x)<0$. Але це означає, що $f(x+\varepsilon z)<f(x)$. Отже, x не є точкою мінімуму f на C. Отримали протиріччя. □

Зауваження.

У випадку відсутності опуклості або f, або C, або і того й іншого, цей критерій перетворюється на необхідну умову.

Розглянемо оптимізацію з обмеженнями, тобто задачу

$$f(x)\xrightarrow[\substack{g_i(x)\le 0\\ i=1..n}]{} min. \tag{1.4}$$

Для цієї задачі можна побудувати функцію Лагранжа:

$$L(x,y)=f(x)+\textstyle\sum_{i=1}^{n} y_i g_i(x), \tag{1.5}$$

де y_i – множина Лагранжа.

Постає задача пошуку сідлової точки (справді, якщо у f мінімум в $\bar{x}$, то у L в $(\bar{x},\bar{y})$ буде мінімум по x і максимум по y, і навпаки) функції L.

Означення 2.

Точка $(\bar{x},\bar{y})$ називається *сідловою точкою* функції L, якщо

$$L(\bar{x},y)\le L(\bar{x},\bar{y})\le L(x,\bar{y})\quad \forall x\,\forall y \tag{1.6}$$

Тобто по x маємо мінімум в $\bar{x}$, а по y – максимум в $\bar{y}$.

Можна переписати ці умови наступним чином:

$$\begin{cases}\langle\nabla_1 L(\bar{x},\bar{y}),x-\bar{x}\rangle\ge 0 & \forall\, x\in C_1\subseteq H_1,\\ \langle-\nabla_2 L(\bar{x},\bar{y}),y-\bar{y}\rangle\ge 0 & \forall\, y\in C_2\subseteq H_2.\end{cases} \tag{1.7}$$

За необхідності ці нерівності можемо об’єднати в одну:

$$\langle \nabla_1 L(\bar{x}, \bar{y}), x - \bar{x} \rangle + \langle -\nabla_2 L(\bar{x}, \bar{y}), y - \bar{y} \rangle \geq 0. \quad (1.8)$$

Ми намагаємося знайти точку $x \in C$, яка задовільняє варіаційній нерівності

$$\langle A(x), y - x \rangle \geq 0, \qquad \forall y \in C,$$

де оператор A є монотонним.

Якщо поглянути на цю задачу як на задачу знаходження нерухомої точки оператора

$$T: x \longmapsto P_C(x - \rho A x), \quad (1.9)$$

де $\rho > 0$, то ці міркування приведуть нас до наступного алгоритму:

$$x_{k+1} = P_C(x_k - \rho_k A x_k). \quad (1.10)$$

Перехід до децентралізованих алгоритмів

У централізованій топології мережі кожен працівник потребує або безпосередньо спілкуватися з центральним вузлом, або опосередковано спілкуватися з усіма іншими працівниками на кожній ітерації [4]. Однак, коли пропускна здатність мережі низька або затримка мережі висока, продуктивність буде значно погіршена. Отже, має сенс обмежити «коло спілкування» кожного вузла тільки тими сусідами, комунікація з якими корисна.

При переході до децентралізованих алгоритмів, нам доведеться ввести додаткові пояснення.

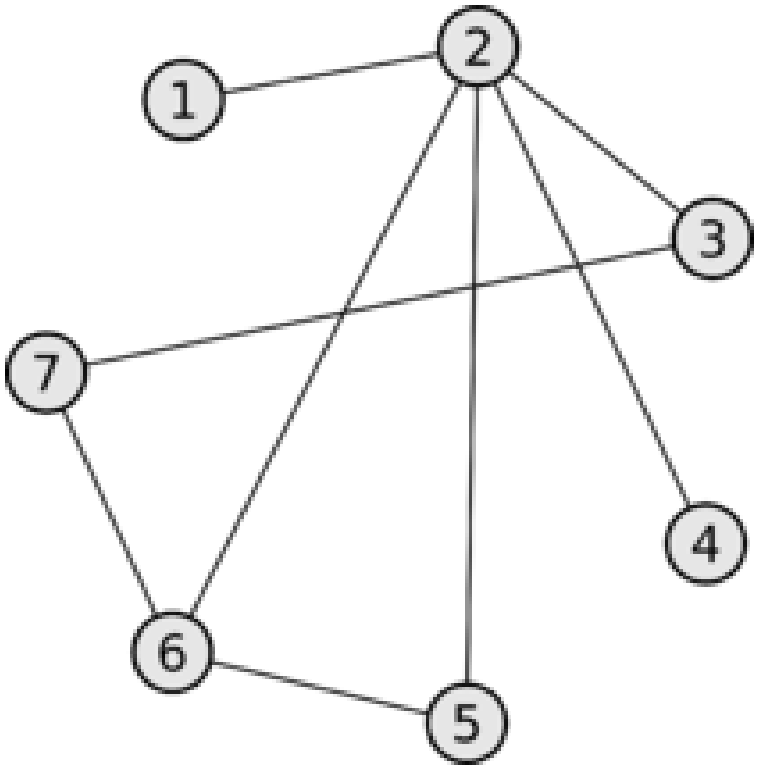

Рис.1. Приклад децентралізованої архітектури.

На Рис.1 наведено приклад децентралізованої системи з семи процесорів, кожен з яких містить своє початкове значення та обчислює свою частину алгоритму на кожній ітерації. Після цього з’єднані вузли обмінюються між собою інформацією щодо поточних обрахунків та переходять до наступного кроку.

Подамо оператор A у вигляді суми

$$A = \sum_{i=1}^{p} A_i. \tag{2.1}$$

Введемо матрицю усереднення W. Зауважимо, що ця матриця буде симетричною та бістохастичною.

Таким чином, якщо подіяти матрицею усереднення на вектор

$$\bar{x} = (x_1, \dots, x_M),$$

де M – кількість вузлів у системі, отримаємо:

$$W^t \bar{x} \xrightarrow[t \to +\infty]{} (\bar{x}, \bar{x} \dots \bar{x}), \tag{2.2}$$

де $\bar{x} = \frac{1}{M} \sum_{i=1}^{M} x_i$.

АЛГОРИТМИ

Алгоритм Попова

Класичний алгоритм [5]**:**

Ініціалізація: обираємо елементи $x_1, y_0, \lambda \in \left(0, \frac{1}{3L}\right)$.

Покладаємо $k = 1$.

Крок 1: обчислюємо

$$y_k = P_C(x_k - \lambda A y_{k-1})$$

Крок 2: обчислюємо

$$x_{k+1} = P_C(x_k - \lambda A y_k)$$

Якщо $x_{k+1} = x_k = y_k$, то зупиняємося і шуканий розв'язок – x_k. Якщо $x_{k+1} \neq x_k$, то йдемо далі.

Покладаємо $k = k + 1$ і завершуємо ітерацію.

Децентралізований алгоритм:

Розглянемо децентралізований алгоритм Попова для формулювання, коли маємо справу з частинним випадком задачі (1.1), а саме:

$$Ax = 0 \qquad A: H \to H,$$

(тобто коли $C = H$).

Отже, отримуємо задачу без обмежень і наступний процес:

Покладаємо $k = 1$.

Крок 1: обчислюємо

$$Y_k = (WX_k) - \lambda \vec{A}(WY_{k-1})$$

Крок 2: обчислюємо

$$X_{k+1} = (WX_k) - \lambda \vec{A} Y_k$$

Якщо $X_{k+1} = X_k = Y_k$, то зупиняємося і шуканий розв'язок – X_k. Якщо $X_{k+1} \neq X_k$, то йдемо далі.

Обмін даними між вузлами за заданою структурою.

Покладаємо $k = k + 1$ і завершуємо ітерацію.

Алгоритм Корпелевич

Класичний алгоритм:

Ініціалізація: обираємо елементи $x_1, \lambda \in \left(0, \frac{1}{L}\right)$.

Покладаємо $k = 1$.

Крок 1: обчислюємо

$$y_k = P_C(x_k - \lambda A x_k)$$

Якщо $x_k = y_k$, то зупиняємося і шуканий розв'язок – x_k. Якщо $y_k \neq x_k$, то йдемо далі.

Крок 2: обчислюємо

$$x_{k+1} = P_C(x_k - \lambda A y_k)$$

Покладаємо $k = k + 1$ і завершуємо ітерацію.

Децентралізований алгоритм:

Розглянемо тепер децентралізований алгоритм Корпелевич для формулювання, коли маємо справу з частинним випадком задачі (1.1), а саме:

$$Ax = 0 \quad A: H \to H,$$

(тобто коли $C = H$).

Отже, отримуємо задачу без обмежень і наступний процес:

Покладаємо $k = 1$.

Крок 1: обчислюємо

$$Y_k = (WX_k) - \lambda\vec{A}(WX_k)$$

Якщо $X_k = Y_k$, то зупиняємося і шуканий розв'язок – X_k. Якщо $Y_k \neq X_k$, то йдемо далі.

Крок 2: обчислюємо

$$X_{k+1} = (WX_k) - \lambda\vec{A}Y_k$$

Обмін даними між вузлами за заданою структурою.

Покладаємо $k = k + 1$ і завершуємо ітерацію.

Алгоритм Tseng

Класичний алгоритм:

Ініціалізація: обираємо елементи $x_1, \lambda \in \left(0, \frac{1}{L}\right)$.

Покладаємо $k = 1$.

Крок 1: обчислюємо

$$y_k = P_C(x_k - \lambda A x_k)$$

Якщо $x_k = y_k$, то зупиняємося і шуканий розв'язок – x_k. Якщо $y_k \neq x_k$, то йдемо далі.

Крок 2: обчислюємо

$$x_{k+1} = y_k - \lambda(Ay_k - Ax_k)$$

Покладаємо $k = k + 1$ і завершуємо ітерацію.

Децентралізований алгоритм:

Розглянемо децентралізований алгоритм Tseng для формулювання, коли маємо справу з частинним випадком задачі (1.1), а саме:

$$Ax = 0 \qquad A: H \to H,$$

(тобто коли $C = H$).

Отже, отримуємо задачу без обмежень і наступний процес:

Покладаємо $k = 1$.

Крок 1: обчислюємо

$$Y_k = (WX_k) - \lambda\vec{A}(WX_k)$$

Якщо $X_k = Y_k$, то зупиняємося і шуканий розв'язок – X_k. Якщо $Y_k \neq X_k$, то йдемо далі.

Крок 2: обчислюємо

$$X_{k+1} = Y_k - \lambda\left(\vec{A}Y_k - \vec{A}(WX_k)\right)$$

Обмін даними між вузлами за заданою структурою.

Покладаємо $k = k + 1$ і завершуємо ітерацію.

ПРАКТИЧНА ЧАСТИНА

Приклад

Для порівняння ефективності роботи децентралізованих варіантів алгоритмів було обрано класичний приклад.

Нехай допустимою множиною є весь простір $C = \mathbb{R}^m$, а

$$F(x) = Ax,$$

де A – квадратна $m \times m$ матриця, елементи якої визначаються наступним чином:

$$a_{i,j} = \begin{cases} -1, & j = m-1-i > i, \\ 1, & j = m-1-i < i, \\ 0, & else. \end{cases}$$

Зауваження.

Нумерація стовпчиків і рядків починається з нуля. Варіант для нумерації з одиничним стартовим елементом буде містити таку заміну: $m + 1$ замість $m - 1$.

Це визначає матрицю, чия бічна діагональ складається з половини одиниць та половини мінус одиниць, решта елементів будуть нульовими.

Ось приклади таких матриць для розмірностей $m = 2$ та $m = 4$:

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Для парних значень m нульовий вектор є розв’язком відповідної варіаційної нерівності (1.1). Для всіх обчислювальних експериментів в якості початкових даних було обрано $x_0 = (1, \dots, 1),\ \varepsilon = 10^{-3},\ \lambda = 0.4.$

Графи зв’язків

В даній роботі розглядаються випадки для децентралізованих систем з 10 та 20 агентів та наведених нижче графів зв’язків.

1) Повний граф (K_4 та K_8)

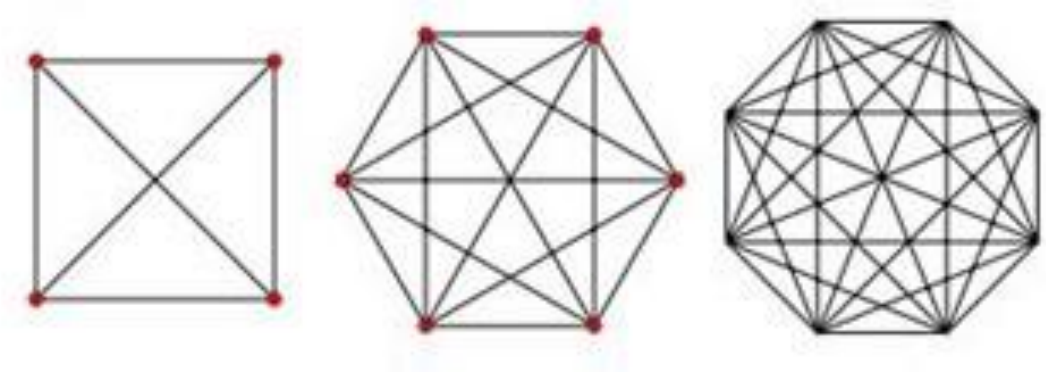

Рис.2. K_4, K_6 і K_8

2) Цикл (C_4 та C_8)

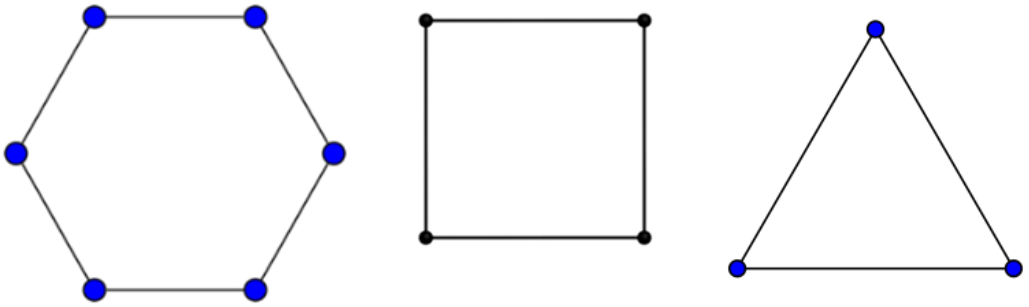

Рис.3. C_6, C_4 і C_3

3) Зірка (S_4 та S_8)

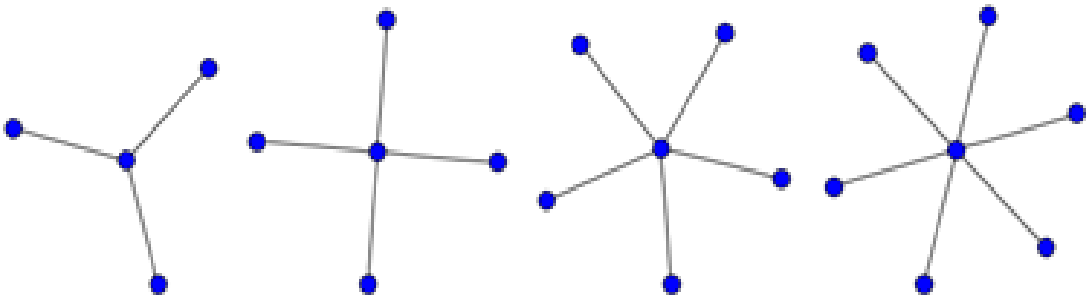

Рис.4. S_3, S_4, S_5 і S_6

4) Колесо (W_4 та W_8)

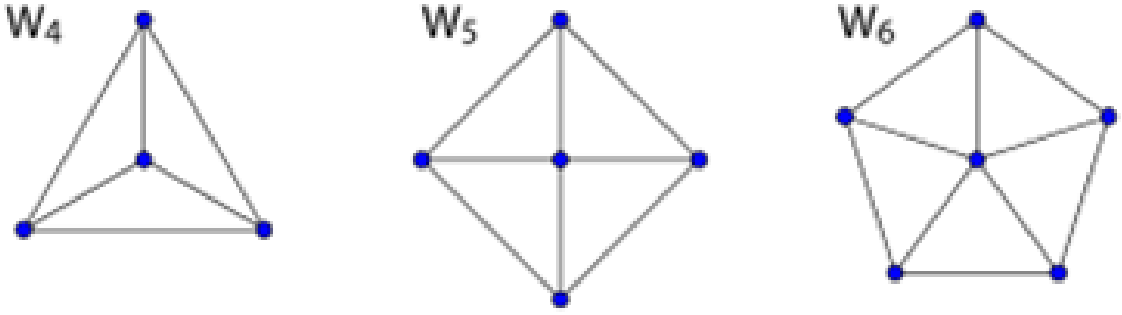

Рис.5. Приклади графів W_n

Результати

Для всіх обчислювальних експериментів в якості початкових даних було обрано $x_0 = (1, \dots, 1)$, $\varepsilon = 10^{-3}$, $\lambda = 0.4$. Розглядались дві розмірності матриць A:

$$m = 100 \text{ та } m = 500.$$

Нижче наведені отримані результати для кожної пари «метод-розмірність».

В стовпці «Час» наведено час обрахування алгоритмом, «К-ть ітерацій» відповідає за к-ть ітерацій, проведених для досягнення бажаної точності, а «Точність» містить в собі значення, видані програмою по модулю, оскільки шукана відповідь – це нульовий вектор.

Граф	Точність (m=100)	Час (m=100)	К-ть ітерацій (m=100)
K_4	3.28294596e-07	0.03996	212
K_8	1.28634546e-07	0.05462	224
C_4	3.64787366e-07	0.03845	203
C_8	3.68754622e-07	0.03973	220
S_4	3.28255396e-07	0.03850	209
S_8	2.24457478e-07	0.05562	225
W_4	3.34784576e-07	0.03838	204
W_8	3.2476365e-07	0.05172	231

Табл.1. Результати для методу Корпелевич (m=100).

Граф	Точність (m=500)	Час (m=500)	К-ть ітерацій (m=500)
K_4	7.28407541e-08	0.093723	223
K_8	6.96254681e-08	0.108383	236
C_4	7.45406343e-08	0.092213	213
C_8	7.47407462e-08	0.093493	232
S_4	7.13563441e-08	0.092263	219
S_8	6.98557485e-08	0.109383	237
W_4	7.34644641e-08	0.092143	214
W_8	6.97407644e-08	0.58935	244

Табл.2. Результати для методу Корпелевич (m=500).

Граф	Точність (m=100)	Час (m=100)	К-ть ітерацій (m=100)
K_4	2.0474863e-07	0.031246	140
K_8	8.8734355e-08	0,33477	150
C_4	2.3735353e-07	0,2990612	134
C_8	2.1265633e-07	0,3280746	147
S_4	2.0766352e-07	0,3079884	138
S_8	2.0362465e-07	0,323611	145
W_4	3.0444663e-07	0,2945976	132
W_8	2.9364533e-07	0,3325382	149

Табл.3. Результати для методу Попова (m=100).

Граф	Точність (m=500)	Час (m=500)	К-ть ітерацій (m=500)
K_4	1.11967639e-07	0.062506	148
K_8	1.03066352e-07	0,067574	160
C_4	1.0766352e-07	0,059549	141
C_8	1.09625468e-08	0,064195	152
S_4	1.28634546e-07	0,061238	145
S_8	1.68754622e-07	0,064617	153
W_4	2.08754622e-07	0,058704	139
W_8	1.26475274e-07	0,067151	159

Табл.4. Результати для методу Попова (m=500).

Граф	Точність (m=100)	Час (m=100)	К-ть ітерацій (m=100)
K_4	3.28294596e-07	0.04073	212
K_8	1.28634546e-07	0.05572	224
C_4	3.64787366e-07	0.03672	203
C_8	3.68754622e-07	0.04024	220
S_4	3.28255396e-07	0.03915	209
S_8	2.24457478e-07	0.05743	225
W_4	3.34784576e-07	0.03940	204
W_8	3.02476365e-07	0.05271	231

Табл.5. Результати для методу Tseng (m=100).

Граф	Точність (m=500)	Час (m=500)	К-ть ітерацій (m=500)
K_4	7.28407541e-08	0.128002	223
K_8	6.96254681e-08	0.917385	236
C_4	7.45406343e-08	1.731295	213
C_8	7.47407462e-08	1.902658	232
S_4	7.13563441e-08	0.901428	219
S_8	6.98557485e-08	0.918548	237
W_4	7.34644641e-08	0.901308	214
W_8	6.97407644e-08	1.398515	244

Табл.6. Результати для методу Tseng (m=500).

Отже, з отриманих результатів видно, що найбільш швидким методом для класичної задачі виявився децентралізований варіант методу Попова.

Методи Корпелевич та Tseng показали однакову точність при рівній кількості ітерацій, хоча метод Корпелевич мав більшу швидкість обчислень. До того ж, при збільшенні розмірності матриці вони обидва спрацювали краще, ніж метод Попова. Однак, для більшості структур мережі при меншій розмірності ($m = 100$) метод Попова виявився більш вдалим.

Також можна зробити висновок, що в цілому при збільшенні кількості вузлів мережі час роботи алгоритмів збільшується.

ВИСНОВКИ

В даній роботі були розглянуті три децентралізовані алгоритми. Пріоритетним завданням дослідження була перевірка ефективності роботи алгоритмів при різних графах зв’язків між агентами. При виконанні програми (особливо в умовах використання реальної мережі, а не моделювання її на одному пристрої) велика кількість ресурсів витрачається на обмін даними між агентами, тому важливо виявити переваги і недоліки кожного з алгоритмів та сформувати певні поради по використанню того чи іншого алгоритму для кожного графу зв’язків при різній кількості агентів в мережі.

В результаті дослідження, що проводилося в даній роботі, виявилося, що методи Корпелевич та Tseng є еквівалентними в плані точності та кількості проведених ітерацій, однак метод Корпелевич працює швидше.

Найбільш швидким у випадках обох обраних розмірностей є метод Попова, проте він суттєво програє своїм конкурентам в точності у випадку більшої розмірності. З іншого боку при меншій розмірності його точність перевершує точність інших алгоритмів.

При збільшенні розмірності матриці збільшується також час обрахування, кількість ітерацій та точність отриманих результатів. Збільшуючи кількість агентів, ми позитивно впливаємо на точність алгоритму, але поступаємося часом.

В цілому можна зробити висновок, що для більших розмірностей доцільніше використовувати метод Корпелевич та структури типу «повний граф», а при менших краще працюватиме алгоритм Попова.

ДЖЕРЕЛА

1. Lyashko S. I., Semenov V. V., Voitova T. A. Low-cost modification of Korpelevich’s methods for monotone equilibrium problems. Cybernetics and Systems Analysis. 2011. Vol. 47. P. 631-639.
2. Kinderlehrer David, Stampacchia Guido An Introduction to Variational Inequalities and Their Applications. Philadelphia: Society for Industrial and Applied Mathematics. 2000.
3. Васильев Ф. П. Методы оптимизации. Москва: Факториал пресс. 2002.
4. Mingrui L., Wei Z., Youssef M., Xiaodong C., Jerret R., Tianbao Y., Payel D. A Decentralized Parallel Algorithm for Training Generative Adversarial Nets. arXiv preprint arXiv: 1910.12999. 2020.
5. Семенов В. В. Варіаційні нерівності: теорія та алгоритми. Київ: ВПЦ «КНУ». 2021.

ДОДАТОК А

Нижче наведено фрагменти коду програми на прикладі методу Попова з коментарями.

#починаємо відлік часу виконання програми

```
start = time.time()
```

#задаємо початкові значення (розмірність та вектор x)

```
m = 40
T = np.ones(m)
```

#задаємо граф зв'язків

```
#full graph
W = np.zeros((m,m))
for i in range (m):
    for j in range(m):
         W[i][j]=1/m
```

#задаємо матрицю A

```
A = np.zeros((m,m))
for i in range (m):
    for j in range(m):
        if (j==m-1-i) & (j>i):
            A[i][j]=-1
        elif (j==m-1-i) & (j<i):
            A[i][j]=1
        else:
            A[i][j]=0
```

#ініціалізація алгоритму

```
iteration_number = 1
x_current, x_next = x_initial, None
y_previous, y_current = y_initial, None
```

#крок 1 алгоритму

```
y_current = x_current - lambda_ * A.dot(y_previous)
```

#крок 2 алгоритму

```
x_next = x_current - lambda_ * A.dot(y_current)
```

#перевірка виконання умови зупинки

```
if (np.linalg.norm(x_current - y_current) < tolerance) and
(np.linalg.norm(x_next - y_current) < tolerance) or (iteration_number
== max_iterations):
    end = time.time()
```

```
    duration = end - start
    return x_current, iteration_number, duration
```

обмін інформацією між вузлами:

#вузол №1 надсилає своє значення x_current вузлу №2

```
comm.send(x_current, dest=2)
```

#вузол №2 отримує значення x_current від вузла №1

```
data1 = comm.recv(source=1)
```

#вузол №2 оброблює значення x_current, отримане від інших вузлів

```
0x_current=W[2][2]*x_current+ W[2][1]*data1+ W[2][3]*data2+
W[2][4]*data4
```

#перехід на наступну ітерацію

```
iteration_number += 1
x_current, x_next = x_next, None
y_previous, y_current = y_current, None
```

#виводимо отримані результати

```
return x_current, iteration_number
print('solution: ', solution)
print('iterationa: ', iteration_number)
```

#виводимо час роботи програми

```
end = time.time()
duration = end - start
print('duration: ', duration)
```