Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра математики

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: **«Локальне керування в стохастичних розподілених системах»**

Виконала: студентка 4-го року навчання

освітньої програми «Прикладна математика»,

спеціальності 113 Прикладна математика

Гак Софія Володимирівна

Керівник: Чорней Р. К.,

кандидат фіз.-мат. наук, доцент

Рецензент_____

(прізвище та ініціали)

Кваліфікаційна робота захищена з оцінкою _____

Секретар ЕК _____

«____» _____ 20____ р.

Київ – 2021

Зміст

Вступ

В роботі розглядається задача локального керування системами, поведінка яких еволюціонує в часі та просторі відповідно до певних стохастичних правил. Цей перехідний механізм визначений для кожної компоненти системи і враховує стани лише поточної компоненти і сусідніх їй. Одним з класів моделей, якими описують подібні явища, є Марковський процес з локальною і синхронною взаємодією над графом, коротко – Марковське випадкове поле. Це поняття детально вивчається в роботі [1], результати якої ми візьмемо за основу нашого дослідження, і розглянемо інший клас моделей, а саме стохастичні клітинні автомати (Probabilistic Cellular Automata, скорочено – PCA).

PCA складається з однакових простих компонент – клітин. Зазвичай вони розташовуються на вершинах одно-, двох- чи трьохвимірної дискретної решітки клітин, мають певний стан зі скінченної множини дискретних станів системи та в кожний момент часу оновлюють його згідно зі стохастичним перехідним правилом, що враховує стани лише сусідніх клітин. Виродженим випадком PCA є клітинні автомати, правила переходу яких є детерміністичними. Попри свою простоту, вони можуть мати доволі складну поведінку. Математик Стівен Вольфрам описав 4 різні поведінкові класи, притаманні правилам клітинних автоматів, згідно з якими вони еволюціють [4]: від найпростішого випадку з однорідним станом, де всі клітини поступово набувають одного кольору або значення, до утворення складних структур.

Актуальним прикладом застосування PCA є моделювання поширення епідемії, де під клітинами розуміємо людей,їх стани – здоровий, хворий, одужала, а перехід в інший стан диктується ймовірностями, що позначають схильність захворіти та враховують стани індивідуумів, з якими контактує поточна людина. PCA можуть стати в нагоді в моделюванні поширення пожежі в лісах, дослідження транспортного руху на дорогах і утворення заторів та ін. [2], [5].

Математично PCA можна описати як систему ланцюгів Маркова, визначених на кожній клітині вже згаданої решітки. Всі вони синхронно переходять зі стану в стан таким чином, що розподіл майбутніх станів залежить від поточних станів сусідніх клітин – Марковська властивість в просторі. Для локального керування такими системами ми введемо додаткову компоненту

в перехідні ймовірності РСА – локальну стратегію – якою зможемо впливати на їх еволюцію, подібно до того, як це описано в роботі [1].

Ввівши додатково поняття однокрокових витрат, зможемо асоціювати модель керованих РСА з Марковським процесом прийняття рішення. Для пошуку оптимальної за критерієм очікуваних витрат стратегії як розв'язку задачі керування РСА розглянемо застосування ітеративного методу покращення стратегії, запропонованим С. Дерманом в його роботі [3], присвяченій дослідженню Марковських процесів прийняття рішення зі скінченним простором станів.

Об'єктом дослідження є стохастичні клітинні автомати як моделі стохастичних розподілених систем. Метароботи – дослідити та описати локальне керування РСА, базуючись на результатах робіт [1], [3] та розробити програмну реалізацію алгоритму для вирішення задачі керування.

Робота складається з 3 розділів. В першому розділі описуються стохастичні клітинні автомати, їх структура та динаміка. Другий розділ присвячений поняттям рішення та стратегії, властивості локальності, Марковості та стаціонарності для введення цих конструкцій в динаміку РСА; розглядається Марковський процес прийняття рішення. В третьому розділі наводиться приклад моделі простих стохастичних клітинних автоматів, описується задача локального керування ними та представлено її розв'язок за допомогою ітеративного методу покращення стратегії [3], реалізованого мовою програмування Python.

Розділ 1

Стохастичні клітинні автомати

1.1 Структура

Основною структурою взаємодії компонент PCA є дискретна *решітка* $L(V,E) := \mathbb{Z}^d, d > 1$, в якій множина вершин V позначає розташування клітин, множина ребер E – зв'язки взаємодії між парами клітин. Якщо клітини k_i і k_j взаємодіють, то $\{k_i, k_j\} \in E$. Зазвичай розглядають одно-, двох- чи трьохвимірні решітки (відповідно $\mathbb{Z}, \mathbb{Z}^2, \mathbb{Z}^3$). Надалі розглядатимемо лише скінченну кількість клітин та ототожнюватимемо кожну з них з її координатами на решітці. Наприклад, якщо решітка L є одновимірною, то вважатимемо клітини на ній лінійно впорядкованими: $k_1 := 1, k_2 := 2, ..., k_N := N$, де N – кількість клітин.

Кожна клітина k перебуває в одному з дискретних станів зі скінченного *локального простору станів* $S_k = \{x_k^1, x_k^2, ..., x_k^{n_k}\}$. Глобальний простір станів, або *простір конфігурацій* PCA, $S^V = \times_{k \in V} S_k$ описує сукупність станів всієї мережі клітин. Конфігурацію позначатимемо $x = \{x_k : k \in V\}$, де $x_k \in S_k$ – стан клітини k. Аналогічно для деякої підмножини $K \in V$ $x_K = \{x_k : k \in K\}$.

Околом клітини k називатимемо множину $V_k = \{k\} \cup \{l : \{k,l\} \in E\} \in V$. Для одновимірних PCA $V_k = \{k-r, ..., k-1, k, k+1, ..., k+r\}, r \in \{0, 1, 2...\}$, r – радіус; окіл з *найближчими сусідами*, де $r = 1$, $V_k = \{k-1, k, k+1\}$. У двохвимірних PCA популярними є окіл фон Неймана та окіл Мура (рис.1.1).

Зауваження 1.1.1. *Важливою ознакою клітин PCA є їх структурна однорідність, враховуючи й те, як для кожної з них визначені околи. У випадку їх скінченної кількості околи крайніх клітин решітки не такі, як у внутрішніх, тому застосовують так звані періодичні межові умови. Наприклад, для одновимірних решіток* $V_1 = \{N, 1, 2\}$, $V_N = \{N-1, N, 1\}$.

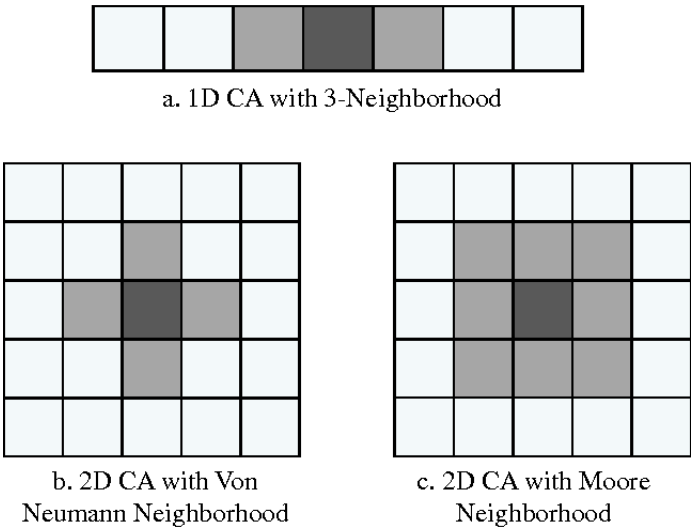

Рис. 1.1: Околи з $r = 1$

1.2 Динаміка

Клітинні автомати еволюціонують в часі та просторі. В кожний (дискретний) момент часу t PCA має певну конфігурацію $x^t = (x_k^t, k \in V)$. Ця конфігурація є реалізацією векторного стохастичного процесу $\xi^t = (\xi_k^t, k \in V)$ з простором станів S^V над решіткою L. Динаміку цього процесу можна описати двома основними характеристиками: *властивості Маркова в часі і просторі (локальність) та синхронність.*

Означення 1.2.1. *Перехідні ймовірності $\xi = (\xi_k^t : k \in V)$ є локальними, якщо розподіл ξ клітини k в момент $t+1$ залежить тільки від розподілу клітин з його околу в момент t:*

$$P(\xi_k^{t+1} = x_k^{t+1} | \xi^0 = x^0, \xi^1 = x^1, ..., \xi^t = x^t) = P(\xi_k^{t+1} = x_k^{t+1} | \xi_{V_k}^t = x_{V_k}^t)$$

Перехідні ймовірності $\xi = (\xi_k^t : k \in V)$ є синхронними, якщо

$$P(\xi^{t+1} = x^{t+1} | \xi^t = x^t) = \prod_{k \in V} P(\xi_k^{t+1} = x_k^{t+1} | \xi^t = x^t)$$

Позначимо

$$T_k(x_k | y_{V_k}) = P(\xi_k = x_k | \xi_{V_k} = y_{V_k})$$

$$T(x|y) = \prod_{k \in V} T_k(x_k | y_{V_k})$$

Означення 1.2.2. *$T_k(x_k|y_{V_k})$ для кожної клітини $k \in V$ називаються локальними ядрами переходу, а $T(x|y)$ – ядром переходу процесу $\xi = (\xi_k^t : k \in V)$*

Перехідний механізм РСА описується ядром переходу $T(x|y)$. Це означає, що всі клітини на кожному кроці еволюції синхронно переходять в нові локальні стани за стохастичними правилами, які залежать від локальних станів клітин з їх околу. На глобальному рівні РСА переходить з однієї конфігурації в іншу.

Зауваження 1.2.1. *Відсутність в $T_k(x_k|y_{V_k})$ індексування параметром t вказує на однорідність в часі процесу ξ. Тобто ймовірність перейти в стан x зі стану y однакова для довільного моменту t:*

$$P(\xi_k^{t'+1} = x_k|\xi_{V_k}^{t'} = y_{V_k}) = P(\xi_k^{t''+1} = x_k|\xi_{V_k}^{t''} = y_{V_k})$$

У зв'язку з цими властивостями переходу та структурою можемо ототожнювати РСА з Марковським процесом з локальною і синхронною взаємодією над графом [1].

Якщо ξ є незвідним, то для нього існує стаціонарний розподіл $\pi = (\pi_k : k \in V)$, який можемо знайти із системи рівнянь [1]:

$$\sum_{x \in S^V} T(y|x)\pi(x) = \pi(y), \quad y \in S^V \tag{1.1}$$

На рис. 1.2 зображено просторово-часову діаграму РСА з виродженим розподілом станів – детерміністичних клітинних автоматів. Еволюція системи зображена зверху вниз з початковою конфігурацією, в якій всі клітини є світлими (стан 0), окрім центральної темної (стан 1). Синхронні переходи клітин зі стану в стан визначаються правилами під діаграмою, що враховують лише стани клітин з околу в попередній момент часу.

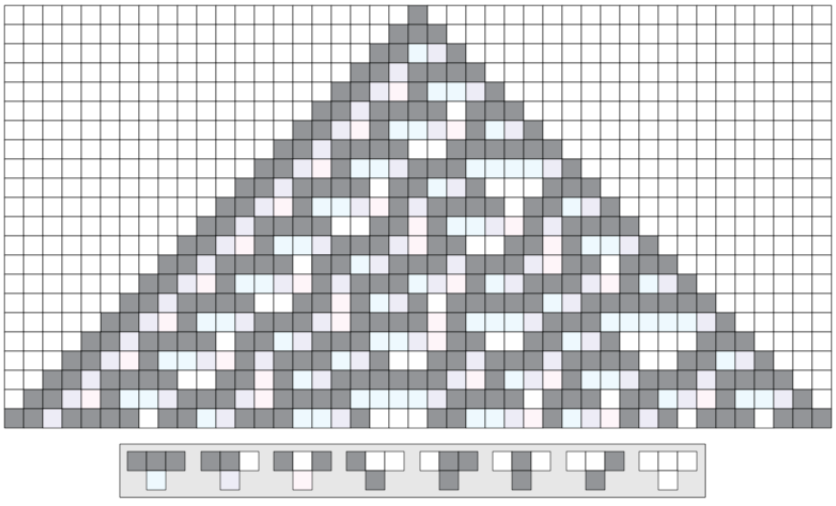

Рис. 1.2: Правило 30

Розділ 2

Керовані стохастичні клітинні автомати

В цьому розділі ми введемо конструкцію керування [1] в ядро переходів стохастичних клітинних автоматів, щоб мати можливість впливати на поведінку РСА задля максимізації прибутку або ж мінімізації втрат.

Означення 2.0.1. *Нехай D_k – множина можливих рішень в клітині k. Тоді множиною всіх можливих рішень над L є $D^V = \times_{k\in V} D_k$.*

В кожній клітині в довільний момент часу t приймається рішення з множини можливих рішень D_k, яке залежить від історії системи до t – послідовності станів $\{x^0, ..., x^t\}$.

Означення 2.0.2. *Залежним від історії $\{x^0, ..., x^t\}$ рішенням в клітині k в момент часу t є*

$$\Delta_k^t = \Delta_k^t(x^0, ..., x^t) \in D_k$$

Відповідно загальне рішення в момент t:

$$\Delta^t = \{\Delta_k^t : k \in V\} \in D^V$$

Стратегією в клітині k називатимемо послідовність рішень, прийнятих в ній:

$$\delta_k = \{\Delta_k^t, t \in \mathbb{N}\}$$

Стратегією для всієї системи клітин є сукупністю поклітинних стратегій:

$$\delta = (\delta_k : k \in V)$$

Надалі ми розглядатимемо стаціонарні локальні Марковські стратегії.

Означення 2.0.3. *(1) Стратегія $\delta = (\delta_k : k \in V)$ називається локальною, якщо рішення в клітині приймається на основі історії станів її околу. Тобто для довільних станів $x^0, ..., x^t \in S$ виконується рівність:*

$$\Delta_k^t(x^0, ..., x^t) = \Delta_k^t(x_{V_k}^0, ..., x_{V_k}^t)$$

(2) Локальна стратегія $\delta = (\delta_k : k \in V)$ *називається локальною Марковською стратегією, якщо рішення, яке приймається в клітині в момент часу* t*, залежить тільки від поточного стану її околу. Тобто для довільних станів* $x^0, ..., x^t \in S$*:*

$$\Delta_k^t(x_{V_k}^0, ..., x_{V_k}^t) = \Delta_k^t(x_{V_k}^t)$$

(3) Локальна Марковська стратегія $\delta = (\delta_k : k \in V)$ *є стаціонарною, якщо рішення в клітині обирається незалежно від часу:*

$$\Delta_k^{t'}(x_{V_k}) = \Delta_k^{t''}(x_{V_k}) = \Delta_k(x_{V_k}) \in D_k$$

Тоді стаціонарна локальна Марковська стратегія є вектором, k-та компонента якого є рішенням в клітині k залежно від стану її околу x_{V_k}:

$$\delta = (\delta^k, k \in V) = (\Delta_k(x_{V_k}), k \in V)$$

Ввівши структуру керування у вигляді рішень, ми тепер маємо можливість не тільки спостерігати за еволюцією системи, але й впливати на її поведінку, тому локальні ядра переходу отримують загальніше позначення, яке вже враховує не тільки поточний стан автоматів, але й рішення, прийняті в них:

$$P(\xi_k^{t+1} = x_k | \xi_{V_k}^t = y_{V_k}) \to P(\xi_k^{t+1} = x_k | \xi_{V_k}^t = y_{V_k}, \Delta_k^t(y_{V_k}) = d_k)$$

$$T_k(x_k | y_{V_k}) \to T_k(x_k | y_{V_k}, d_k), \quad d_k \in D_k$$

Відповідно ядро переходу набуває вигляду

$$T(x|y) \to T(x|y,d), \quad d \in D^V$$

$$T(x|y,d) = \prod_{k \in V} T_k(x_k | y_{V_k}, d_k)$$

Тепер РСА мають простори станів, рішень, перехідні ймовірності з Марковською властивістю. Якщо додатково ввести поняття однокрових нагород або ж витрат, які залежать від конфігурації автоматів та прийнятого рішення, то РСА цілком нагадуватимуть Марковський процес прийняття рішення. Далі розглянемо основні поняття про цей процес.

2.1 Марковський процес прийняття рішення

Особа має змогу впливати на поведінку стохастичної системи, яка еволюціонує в часі, приймаючи рішення або виконуючи дії. Метою є вибір такої послідовності дій (стратегії), завдяки яким динаміка системи буде оптимальною відповідно до деякого наперед заданого критерію оптимальності.

Означення 2.1.1. *Марковським процесом прийняття рішення (Markov decision process, скорочено – MDP) називається набір* $(S, A, P(.|.,.), r(.,.))$, *де*

- S – *простір станів системи*
- A – *множина можливих дій або рішень, простір рішень*
- $P(s'|s,a) = P(s^{t+1} = s'|s^t = s, a^t = a)$ – *ймовірність того, що в наступний момент часу* $t+1$ *система перейде в стан* s', *якщо нині, в момент часу* t, *вона в стані* s *і особа приймає рішення* a
- $r(s,a)$ – *однокрокова нагорода або витрата, яку отримує особа, приймаючи рішення* a, *коли система перебуває в стані* s

Надалі вважатимемо $r(.,.)$ витратами.

Зауваження 2.1.1. *Перехідні ймовірності* $P(s'|s,a)$ *задовольняють властивості Маркова; поточний стан містить усю необхідну інформацію про минуле і поточне системи для її майбутнього стану.*

Важливим поняттям в MDP є *стратегія* δ, яке збігається із введеним раніше означенням (2.0.2).

Основною задачею є знаходження стратегії δ, яка задовольнятиме певному критерію оптимальності. Одним з таких критеріїв є мінімізація очікуваних дисконтовних нагород з нескінченним горизонтом $(T \to \infty)$ [3]

$$E^\delta \sum_{t=0}^{T} \gamma^t r(s^t, a^t) \to \min_\delta,$$

де E^δ – математичне очікування у випадку прийняття рішень згідно зі стратегією δ, γ – коефіцієнт дисконтування, $\gamma \in [0,1]$.

Ще одним критерієм є мінімізація середніх очікуваних витрат в одиницю часу [3].

Позначимо

$$C_T^\delta := E^\delta \sum_{t=0}^{T} r(s^t, a^t),$$

тобто C_T^δ – очікувані загальні витрати роботи системи до часу $t = T$, якщо δ є стратегією, згідно якої особа керує системою.

Тоді середніми очікуваними витратами в одиницю часу для $T \to \infty$:

$$R^\delta = \lim_{T\to\infty} \frac{C_T^\delta}{T+1},$$

а критерій матиме вигляд

$$R^\delta \to \min_\delta \tag{2.1}$$

Розділ 3

Задача локального керування стохастичними клітинними автоматами

3.1 Модель та постановка задачі

Наведемо приклад простого РСА з можливістю контролювати його поведінку. Нехай маємо 5 скінченних клітинних автоматів з дискретним часом, $t \in \mathbb{N}$. Вони визначені на вершинах решітки $L = L(V, E) = \mathbb{Z}$, $|V| = 5$. Окіл для кожної клітини – найближчі сусіди, тобто для довільної клітини $k \in \{1, ..., 5\}$ $V_k = \{k-1, k, k+1\}$. Для крайніх клітин застосовано періодичні межові умови: $V_1 = \{5, 1, 2\}$, $V_6 = \{4, 5, 1\}$.

Еволюція системи описується керованим Марковським ланцюгом з локальною та синхронною взаємодією – парою (ξ, δ) – над решіткою L [1]. Процес ξ приймає значення з простору станів $S^V = \{(x_1, ..., x_5) \in \{0, 1\}^5\}$, а стратегія δ покладемо стаціонарною локальною Марковською стратегією, тобто з означення (2.0.3) в кожній клітині незалежно від часу приймається рішення $d_k \in D_k$ відповідно до поточного стану сусідніх клітин, $d_k = \Delta(x_{V_k})$. Вважатимемо множину D_k фіксованою для всіх клітин: $D_k = D = \{0, 1\}$. Тоді глобально до системи застосовується керування $(d_0, ..., d_5) \in D^V = \times_{k \in V} D$.

Отже, маємо $2^5 = 32$ стани системи – всі булеві вектори розмірності 5:

$$x^0 = (0,0,0,0,0) \quad x^1 = (0,0,0,0,1) \quad x^2 = (0,0,0,1,0)$$
$$x^3 = (0,0,0,1,1) \quad x^4 = (0,0,1,0,0) \quad x^5 = (0,0,1,0,1)...$$

Так само задаються рішення:

$$d^0 = (0,0,0,0,0) \quad d^1 = (0,0,0,0,1) \quad d^2 = (0,0,0,1,0)$$
$$d^3 = (0,0,0,1,1) \quad d^4 = (0,0,1,0,0) \quad d^5 = (0,0,1,0,1)...$$

Стан околу клітини y_{V_k} є вектором з 3 координатами, кожна з яких може мати 2 значення. Відповідно маємо 8 різних можливих околів V_k:

$$N^0 := (0,0,0) \quad N^1 := (0,0,1) \quad N^2 := (0,1,0) \quad N^3 := (0,1,1)$$
$$N^4 := (1,0,0) \quad N^5 := (1,0,1) \quad N^6 := (1,1,0) \quad N^7 := (1,1,1)$$

Перехід до нового стану клітини здійснюється згідно зі стохастичними локальними ядрами переходу $T_k(x_k|y_{V_k}, d_k)$. Оскільки стан клітини x_k приймає лише два значення, то достатньо задати ймовірності $p(N_k, d_k) := T_k(1|.,.)$, а $q(N_k, d_k) = T_k(0|.,.) = 1 - p(N_k, d_k)$. Маємо наступні ймовірності $p(N_k, d_k)$:

$V_k \backslash d_k$	0	1
N^0	1/2	1/3
N^1	3/4	1/3
N^2	1/4	2/3
N^3	2/3	3/4
N^4	3/4	1/2
N^5	1/2	1/4
N^6	1/4	1/3
N^7	5/6	2/3

Ядро переходу $T(x|y,d)$ має вигляд:

$y \backslash x$	x^0	...	x^{31}
x^0	$q(N^0,d_1)q(N^0,d_2)q(N^0,d_3)q(N^0,d_4)q(N^0,d_5)$	...	$p(N^0,d_1)p(N^0,d_2)p(N^0,d_3)p(N^0,d_4)p(N^0,d_5)$
x^1	$q(N^4,d_1)q(N^0,d_2)q(N^0,d_3)q(N^1,d_4)q(N^2,d_5)$	...	$p(N^4,d_1)p(N^0,d_2)p(N^0,d_3)p(N^1,d_4)p(N^2,d_5)$
x^2	$q(N^0,d_1)q(N^0,d_2)q(N^1,d_3)q(N^2,d_4)q(N^4,d_5)$	...	$p(N^0,d_1)p(N^0,d_2)p(N^1,d_3)p(N^2,d_4)p(N^4,d_5)$
x^3	$q(N^4,d_1)q(N^0,d_2)q(N^1,d_3)q(N^3,d_4)q(N^6,d_5)$	...	$p(N^4,d_1)p(N^0,d_2)p(N^1,d_3)p(N^3,d_4)p(N^6,d_5)$
x^4	$q(N^0,d_1)q(N^1,d_2)q(N^2,d_3)q(N^4,d_4)q(N^0,d_5)$	...	$p(N^0,d_1)p(N^1,d_2)p(N^2,d_3)p(N^4,d_4)p(N^0,d_5)$
...	...	...	...
x^{30}	$q(N^3,d_1)q(N^7,d_2)q(N^7,d_3)q(N^6,d_4)q(N^5,d_5)$	...	$p(N^3,d_1)p(N^7,d_2)p(N^7,d_3)p(N^6,d_4)p(N^5,d_5)$
x^{31}	$q(N^7,d_1)q(N^7,d_2)q(N^7,d_3)q(N^7,d_4)q(N^7,d_5)$	...	$p(N^7,d_1)p(N^7,d_2)p(N^7,d_3)p(N^7,d_4)p(N^7,d_5)$

В загальному випадку його можна представити у вигляді

$$T(x|y,d) = \prod_{k \in V} (1 - x_k) - (-1)^{x_k} p(y_{V_k}, d_k)$$

Перебуваючи в момент часу t в стані $\xi^t = x^t$ і будучи керованою рішенням d^t, система зазнає однокрових втрат $r(x^t, d^t) >= 0$, яку в нашій задачі задамо функцією:

$$r(x^t, d^t) = \sum_{k \in V} [(c_k - u_k)x_k + u_k b_k],$$

де $b = (2, 4, 3, 3, 1)$ – додаткові витрати за прийняті в клітині рішення, $c = (1, 2, 3, 4, 5)$ – додаткові витрати за перебування клітини в стані 1.

Задача керування РСА полягає у відшуканні такої стратегії δ, при якій середні очікувані витрати в одиницю часу (2.1) будуть мінімальними:

$$R^{\delta} \to \min_{\delta}$$

3.2 Ітеративний метод покращення стратегії

Розглянемо ітеративний метод покращення стратегії для розв'язку задачі Марковського процесу прийняття рішення, докладно описаний в [3]. Його умовно можна поділити на дві частини: пошук значень $v = (v(x) : x \in S^V)$ та крок покращення стратегії.

Пошук значень v:

Обрати деяку стратегію δ та, попередньо визначивши для неї однокрокові витрати $r(x, \delta(x))$ та ядро переходів $T(x|y, \delta(y))$, знайти витрати R^{δ} та значення v із системи рівнянь:

$$\begin{cases} R^{\delta} + v(y) = r(y, \delta(y)) + \sum_{x \in S^V} T(x|y, \delta(y))v(x), y \in S^V \\ \sum_{x \in S^V} \pi^{\delta}(x)v(x) = 0 \end{cases} \tag{3.1}$$

Покращення стратегії:

(1) Для кожного стану y знайти кращі рішення – визначити множини D^y, елементами яких будуть рішення d, що задовольняють одній з умов:

$$\sum_{x \in S^V} T(x|y, d)R^{\delta} < R^{\delta} \tag{3.2}$$

або

$$\begin{cases} \sum_{x \in S^V} T(x|y, d)R^{\delta} = R^{\delta} \\ r(y, d) + \sum_{x \in S^V} T(x|y, d)v^{\delta}(x) < R^{\delta} + v^{\delta}(y) \end{cases} \tag{3.3}$$

(2) Сформувати нову стратегію δ', яка для кожного стану y диктуватиме рішення $d \in D^y$. Якщо є стани, для яких множина $D^y = \varnothing$, то для них залишаємо рішення з попередньої стратегії δ.

Ці два основні кроки потрібно повторювати, допоки хоча б одна з множин $D^y \neq \varnothing$. В цьому випадку нова стратегія δ' буде відмінною від попередньої δ хоча б в одному стані та $R^{\delta'} \leq R^{\delta}$. Інакше $\delta' = \delta$ і тоді δ' вважаємо оптимальною стратегією.

3.3 Програмна реалізація методу та розв'язок задачі

Метод покращення стратегії було реалізовано як набір функцій мовою Python. Розглянемо їх застосування на прикладі розв'язку задачі з 3.1.
Обрахуємо всі можливі ядра переходу та однокрокові витрати для простору станів X та простору рішень D^V за допомогою функцій
`transKernels(X, D, P)` та `rewards = stepRewards(X, D, b, c)` відповідно де масив `P` := $p(N_k, d_k)$.

```
Ts = transKernels(X, D, P)
rewards = stepRewards(X, D, b, c)
```

`Ts[d][y][x]` - ймовірність переходу $y \to x$ при рішенні d
`rewards[x][d]` - нагорода за рішення d в стані x

Оберемо за початкову стратегію $\delta = (0, ..., 0, ..., 0)$. Вона є масивом з індексами рішень $d \in D^V$.

```
strat = np.array([0 for x in X])
```

Обчислимо для неї ядро переходу $T^\delta = T^\delta(y, x)$, однокрокові витрати $r^\delta = r(x, \delta(x))$.
В T записуємо ймовірності $T[\delta(y)][y][x] = T^\delta[y][x]$ для кожного стану y та x

```
T = np.array([[Ts[strat[y_idx]][y_idx][x_idx] for x_idx in range(len(X))]
 for y_idx in range(len(X))])

r = np.array([rewards[x_idx][strat[x_idx]] for x_idx in range(len(X))])
```

Отримаємо

$y\backslash x$	x^0	x^1	x^2	x^3	x^4	...	x^{27}	x^{28}	x^{29}	x^{30}	x^{31}
x^0	0.031	0.031	0.031	0.031	0.031	...	0.031	0.031	0.031	0.031	0.031
x^1	0.012	0.004	0.035	0.012	0.012	...	0.035	0.035	0.012	0.105	0.035
x^2	0.012	0.035	0.004	0.012	0.035	...	0.012	0.035	0.105	0.012	0.035
x^3	0.008	0.003	0.016	0.005	0.023	...	0.016	0.07	0.023	0.141	0.047
x^4	0.012	0.012	0.035	0.035	0.004	...	0.105	0.012	0.012	0.035	0.035
...	...	...	...	...	...	...	...	...	...	...	...
x^{27}	0.003	0.017	0.007	0.035	0.003	...	0.058	0.006	0.029	0.012	0.058
x^{28}	0.003	0.008	0.008	0.023	0.001	...	0.234	0.009	0.026	0.026	0.078
x^{29}	0.003	0.007	0.003	0.007	0.001	...	0.174	0.029	0.058	0.029	0.058
x^{30}	0.003	0.003	0.001	0.001	0.017	...	0.012	0.174	0.174	0.058	0.058
x^{31}	0	0.001	0.001	0.003	0.001	...	0.08	0.016	0.08	0.08	0.402

i

x	x^0	x^1	x^2	x^3	x^4	...	x^{27}	x^{28}	x^{29}	x^{30}	x^{31}
$r(x, \delta(x))$	0	5	4	9	3	...	12	6	11	10	15

Знаючи T^{δ}, знайдемо стаціонарний розподіл $\pi^{\delta} = \pi^{\delta}(x)$ за (1.1):

```
pi = stationaryDist(T)
```

x	x^0	x^1	x^2	x^3	x^4	...	x^{27}	x^{28}	x^{29}	x^{30}	x^{31}
$\pi(x)$	0.015	0.019	0.019	0.025	0.019	...	0.046	0.032	0.046	0.046	0.066

Знайдемо розв'язок системи рівнянь (3.1)

```
v = getValues(T, r, pi)
```

x	x^0	x^1	x^2	x^3	x^4	...	x^{27}	x^{28}	x^{29}	x^{30}	x^{31}
$v(x)$	-10.165	-5.311	-5.436	-0.615	-6.525	...	5.122	-1.052	2.913	1.313	13.171

і $R^{\delta} =$ `v[0]` $= 8.642$ Зробимо крок покращення стратегії, сформувавши нову стратегію згідно з умовами (3.2) і (3.3):

```
newStrat = np.array([setDy(y_idx, D, rewards, Ts, v, strat) for y_idx in
    range(len(X))])
```

Нова стратегія $\delta' = \delta'(x)$:

x	x^0	x^1	x^2	x^3	x^4	x^5	x^6	...	x^{25}	x^{26}	x^{27}	x^{28}	x^{29}	x^{30}	x^{31}
$\delta'(x)$	0	0	1	0	1	0	1	...	0	1	1	1	0	1	17

$\delta' \neq \delta$ і, до того ж, $R^{\delta'} = 8.229 < R^{\delta}$. Продовжуючи ці кроки, в результаті отримаємо оптимальну стратегію $\delta^* = \delta^*(x)$:

x	x^0	x^1	x^2	x^3	x^4	x^5	x^6	...	x^{25}	x^{26}	x^{27}	x^{28}	x^{29}	x^{30}	x^{31}
$\delta^*(x)$	0	0	1	0	1	0	1	...	0	1	1	1	0	1	1

зі значенням критерію $R^{\delta^*} = 8.166$ за 5 ітерацій.

Всі кроки вище зібрані в одній функції `stratIter(X, D, P, b, c, strat)`, яка повертає оптимальну стратегію, витрати R, початкову стратегію `strat` і кількість ітерацій, необхідну для знаходження оптимальної стратегії. Якщо початкова стратегія `strat` не задана, то вона обирається випадковим чином.

```
optStrat, R, initStrat, iterCnt = stratIter(X, D, P, b, c)
```

Вивід програми вказує на знаходження тієї самої оптимальної стратегії:

```
Optimal strategy
[0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1]
with the expected average cost per unit time of 8.166
from the initial strategy
[10 3 10 17 19 6 1 11 24 7 11 27 12 30 11 19 29 13 30 21 23 29 15 7
29 19 11 8 28 27 30 0]
within 2 iterations
```

Висновки

У цій роботі було досліджено стохастичні клітинні автомати як моделі стохастичних розподілених систем. Було описано їх структуру та поведінку в просторі та часі. Введено поняття локальних рішень і стратегій для можливості керування PCA подібно до керування марковським випадковим полем. Розглянуто приклад простої моделі та задачу локального керування для неї. Ототожнили постановку задачі з Марковським процесом прийняття рішення, що дало змогу застосувати ітеративний метод покращення стратегії, попередньо запрограмувавши його.

Література

[1] Chornei R. K., Daduna H., Knopov P. S. Control of Spatially Structured Random Processes and Random Fields with Applications. — New York: Springer Science + Business Media, Inc., 2006. — 261 p.

[2] P.-Y. Louis and F.R. Nardi (eds.), Probabilistic Cellular Automata, Emergence, Complexity and Computation 27. — Springer International Publishing AG, 2018

[3] C. Derman. Finite State Markovian Decision Processes. Academic Press, New York, London, 1970.

[4] Wolfram S. A New Kind of Science. — Champaign, Illinois : Wolfram Media, Inc, 2002. — 1197 p.

[5] Ilachinski A. Cellular Automata — A Discrete Universe. –– Singapore : World Scientific, 2002. –– Reprint of the first edition from 2001.

Додатки

A

```
def genStatesActions(num, el, array, idx):
  if idx == num:
    for j in el:
      array.append(j)
    return

  el[idx] = 0
  genStatesActions(num, el, array, idx + 1)

  el[idx] = 1
  genStatesActions(num, el, array, idx + 1)
```

Лістинг 1: Генерування булевих векторів

```
def neighbours(x, k):
  N = [x[(k-1) % len(x)], x[k], x[(k+1) % len(x)]]
  return int("".join(str(i) for i in N),2)

def stepRewards(X, D, b, c):
  n = len(b)
  allRewards = []
  for x in X:
    reward = []
    for d in D:
      reward = np.append(reward, sum([(c[k] - d[k])*x[k] + d[k]*b[k] for k
    in range(n)]))
    allRewards.append(reward)
  return np.array(allRewards)

def transKernels(X, D, P):
  Ts = []
  for d in D:
    T = [0 for x in X]
      for y_idx, y in enumerate(X):
        T[y_idx] = [np.prod([(1 - x[k]) - (-1)**(x[k])*P[neighbours(y, k)][d
    [k]] for k in range(len(y))]) for x in X]
      Ts.append(T)
  return np.array(Ts)

def stationaryDist(T):
  A=np.append(np.transpose(T)-np.identity(len(T)),[np.ones(len(T))],axis=0)
  b = np.transpose(np.append(np.zeros(len(T)), 1))
  pi = np.linalg.solve(np.transpose(A).dot(A), np.transpose(A).dot(b))
  return np.array(pi)

def getValues(T, r, pi):
```

```
  n = len(pi)+1
  r = np.append(r, 0)
  A = np.subtract(np.identity(n-1), T)
  A = np.insert(A, 0, 1, axis=1)
  A = np.append(A, np.insert(pi, 0, 0))
  A = A.reshape(n, n)
  v=np.linalg.solve(A, r)
  return np.array(v)

def setDy(y_idx, D, rewards, Ts, v, strat):
  c2 = []
  R = v[0]
  for d_idx in range(len(D)):
    cond1_left = (Ts[d_idx][y_idx]*R).sum()
    cond2_left = rewards[y_idx][d_idx] + Ts[d_idx][y_idx].dot(v[1:])
    cond2_right = R + v[y_idx+1]
    if cond1_left == R and cond2_left < cond2_right:
      c2.append((d_idx, cond2_left))
    if len(c2)>0:
      Dy = (min(c2, key = lambda t: (t[1])))[0]
    else:
      Dy = strat[y_idx]
  return Dy
```

Лістинг 2: Базові функції

```
def betterStrat(X, D, rewards, Ts, v, strat):
  newStrat = [setDy(y_idx, D, rewards, Ts, v, strat) for y_idx in range(len(
    X))]
  return np.array(newStrat)

def stratIter(X, D, P, b, c, strat = None):
  Ts = transKernels(X, D, P)
  rewards = stepRewards(X, D, b, c)

  if strat is None:
    strat = np.random.randint(0, high=len(D), size=len(X))

  initStrat = strat
  iterCnt=0

  T = [[Ts[strat[y_idx]][y_idx][x_idx] for x_idx in range(len(X))] for y_idx
    in range(len(X))]
  r = [rewards[x_idx][strat[x_idx]] for x_idx in range(len(X))]
  pi = stationaryDist(T)
  v = getValues(T, r, pi)
  newStrat = betterStrat(X, D, rewards, Ts, v, strat)

  while not np.array_equal(newStrat, strat):
    iterCnt += 1
    strat = np.copy(newStrat)
    T = [[Ts[strat[y_idx]][y_idx][x_idx] for x_idx in range(len(X))] for
   y_idx in range(len(X))]
    r = [rewards[x_idx][strat[x_idx]] for x_idx in range(len(X))]
    pi = stationaryDist(T)
    v = getValues(T, r, pi)

    newStrat = betterStrat(X, D, rewards, Ts, v, strat)

```

```
31    return np.array(strat), v[0], initStrat, iterCnt
```

Лістинг 3: Основні функції

.