НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«На правах рукопису»

УДК 51-77

«До захисту допущено»
В.о. завідувача кафедри
_____ М.В.Грайворонський
“___”_____2018 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності: 113 Прикладна математика

на тему: Захист персональних геоданих на основі лінійної оптимізації з невизначеністю

Виконав (-ла): студент (-ка) 2 курсу, групи ФІ-71мп
(шифр групи)

Конторчук Назарій Іванович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник к.ф.-м.н., доц. Смирнов Сергій Анатолійович
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
Спеціальність (спеціалізація) – 113 Прикладна математика («Математичні методи комп'ютерного моделювання»)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ М.В.Грайворонський
(підпис)

«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Конторчуку Назарію Івановичу
(прізвище, ім'я, по батькові)

1. Тема дисертації
Захист персональних геоданих на основі лінійної оптимізації з невизначеністю

науковий керівник дисертації к.ф.-м.н., доц. Смирнов Сергій Анатолійович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «15 » листопада 2018 р. № 4171-с

2. Термін подання студентом дисертації 10.12.2018 р.

3. Об'єкт дослідження _____
_____

4. Вихідні дані _____
_____
_____
_____

5. Перелік завдань, які потрібно розробити _____
_____
_____
_____
_____
_____

6. Орієнтовний перелік ілюстративного матеріалу _____
_____
_____

7. Орієнтовний перелік публікацій _____

_____

_____

8. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка

Студент _____ (підпис) _____ (ініціали, прізвище)

Науковий керівник дисертації _____ (підпис) _____ (ініціали, прізвище)

* Консультантом не може бути зазначено наукового керівника магістерської дисертації.

РЕФЕРАТ

Обсяг роботи 71 сторінок, 7 джерел літератури.

Об’єктом дослідження є захист приватності в аспекті просторового позиціонування.

Предметом дослідження є структурний аналіз систем захисту локації суб’єкта за допомогою функції розподілу.

Методами дослідження даної задачі були спеціальна задача лінійного програмування, основні методи лінійної оптимізації.

Результатом роботи є отримана спеціальна задача лінійного програмування в результаті аналізу контексту задачі і структури матриці. Отримані результати є повністю новими, адже дана задача раніше не була розв’язана для всіх елементів, а відомі методи мають дуже велику складність. Складність спеціальної задачі лінійного програмування логарифмічна, що в кілька порядків менше.

В подальшому, цю задачу можна розвивати ввівши зв'язок між відношенням ймовірностей і лінійною ймовірністю через функцію оптимізації і розглядати, що дане відношення означає з точки зору самої задачі. Дану задачу можна розвивати, зробивши програмну реалізацію даного алгоритму.

ПРОСТОРОВА НЕРОЗРІЗНЕНІСТЬ, ДИФЕРЕНЦІЙНА ПРИВАТНІСТЬ, ЛОКАЦІЙНА ПРИВАТНІСТЬ, ПРОСТОРОВО НЕРОЗРІЗНЕНИЙ МЕХАНІЗМ

РЕФЕРАТ

Объем работы 71 страниц, 7 источников литературы.

Объектом исследования является защита приватности в аспекте пространственного позиционирования.

Предметом исследования является структурный анализ систем защиты локации субъекта с помощью функции распределения.

Методами исследования данной задачи были специальная задача линейного программирования, основные методы линейной оптимизации.

Результатом работы является полученная специальная задача линейного программирования в результате анализа контекста задачи и структуры матрицы. Полученные результаты являются полностью новыми, ведь данная задача ранее не была решена для всех элементов, а известные методы имеют очень большую сложность. Сложность специальной задачи линейного программирования логарифмическая, что на несколько порядков меньше.

В дальнейшем, эту задачу можно развивать введя связь между отношением вероятностей и линейной вероятностью через функцию оптимизации и рассматривать, что данное отношение означает с точки зрения самой задачи. Данную задачу можно развивать, сделав программную реализацию данного алгоритма

ПРОСТРАНСТВЕННАЯ НЕРАЗЛИЧИМОСТЬ, ДИФФЕРЕНЦИАЛЬНАЯ ПРИВАТНОСТЬ, ЛОКАЦИОННАЯ ПРИВАТНОСТЬ, ПРОСТРАНСТВЕННО НЕРАЗЛЕЧИМЫЙ МЕХАНИЗМ

.

ABSTRACT

The volume of work is 71 pages, 7 sources of literature.

The object of research is the protection of privacy in the aspect of spatial positioning.

The subject of the study is a structural analysis of the protection systems of the subject's location using the distribution function.

The methods of studying this problem were a special problem of linear programming, the main methods of linear optimization.

The result of the work is the obtained special problem of linear programming as a result of the analysis context problem and the structure of the matrix. The obtained results are completely new, because the given problem was not solved for all elements earlier, and the known methods have a very big complexity. The complexity of the special problem of linear programming is logarithmic, which is several orders of magnitude smaller.

In the future, this problem can be developed by introducing a relationship between the ratio of probabilities and linear probability through the optimization function and to consider that this relationship means from the point of view of the task itself. This task can be developed by making a software implementation of this algorithm.

GEO-INDISTINGUISHABLE, DIFFERENTIAL PRIVACY, LOCATION PRIVACY, GEO-INDISTINGUISHABLE MECHANISM

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

LBS (Location Based Services) – системи, що базуються на місцезнаходженні;

GPS (Global Positioning System) – система глобального позиціонування;

X – множина точок альтернатив ($x \in X$);

$\pi(x)$ – ймовірність присвоєння місцезнаходження x;

K – механізм вибору звітованого значення;

AOI – область інтересів;

AOR – область пошуку;

POI – географічні об'єкти, відмічені на карті спеціальними позначками;

Вступ

В даній роботі розглядається задача оптимізації просторово нерозрізненого механізму. Дана задача викликає великий інтерес з точки зору прикладної задачі. В її основу лягла проблема знаходження балансу між просторовими даними користувача й інформацією необхідної для задоволення запиту користувача.

Актуальність даної задачі в тому, що в наш час порушується питання приватності користувача при використанні LBS. Це порушує його законні права і зловмисники можуть легко використовувати отриману інформацію в своїх цілях. Вирішення даної задачі дозволить з'ясувати, яку кількість інформації користувачеві потрібно надати для отримання найкращої відповіді на запит. Тому дана проблема є актуальною й розглядається в даній роботі.

Метою і завданням даної роботи є пошук оптимальних значень, при яких можна буде зашифрувати реальне місце розташування користувача і надати йому достатньої інформації на запит.

Об'єктом дослідження є захист приватності в аспекті просторового позиціонування.

Предметом дослідження є структурний аналіз систем захисту локації суб'єкта за допомогою функції розподілу.

Методами дослідження даної задачі були спеціальна задача лінійного програмування, основні методи лінійної оптимізації.

Наукова новизна отриманих результатів заключається в тому, що початкову систему вперше було зведено до вигляду спеціальної задачі лінійного програмування.

Практичне значення отриманих результатів в тому, що складність отриманої задачі набагато менша, ніж було запропоновано в літературі в даному напрямку. І ще однією особливістю результатів є те, що раніше реалізований симплекс метод не рахував при великих кількостях елементів на проранжованій караті. Дані результати

можна використовувати для швидкого і якісного обрахування оптимального значення задачі просторового позиціонування.

1 ПОНЯТТЯ ЛОКАЛЬНОЇ ПРИВАТНОСТІ

Інформація про місцезнаходження фізичних осіб в останні роки стає ціннішою тому це викликає неабиякий інтерес до систем які відслідковують просторові дані. Такі системи прийнято називати - позиційними системи. Вони включають в себе Location Based Services (системи, що базуються на місцезнаходженні), в яких користувач отримує в режимі реального часу, послуги, пов'язані з його поточним місцезнаходженням. І алгоритми інтелектуального аналізу місце розташування даних, що використовуються для визначення точок інтересу і шаблони трафіку.

Використання LBSs, було орієнтовано на мобільні пристрої,з урахуванням зростаючої популярності, оснащених GPS чіпами. Дослідження, проведене в США, показує, що в 2012 році 46% дорослого населення країни користується смартфоном, причому, 74% з них використовують LBSs.

В даному розділі буде введено поняття просторової нерозрізненості, а також розглянуто як використовувати дані поняття для запису задачі оптимізації просторово нерозрізненного механізму.

1.1 Системи захисту приватності з урахуванням просторових даних

Системи на основі визначення місця розташування продемонстрували багато переваг для окремих людей і суспільства в цілому. Проте зростаючий вплив інформації розташування користувачів піднімає важливі питання конфіденційності. Вона легко пов'язує її з іншою інформацією, яку людина зазвичай хоче захистити: шляхом збору і обробки точних даних про місцезнаходження на регулярній основі, можна зробити висновок про місце розташування будинку або роботи, сексуальні переваги, політичні погляди індивідуума , релігійні нахили і т.д. У своїй крайній формі, моніторингу та контролю місця розташування індивідуума була навіть описана як форма рабства.

З метою усунення цих недоліків, скористаємося наступним визначенням конфіденційності для LBSs, а також імовірнісний метод, який дозволяє користувачеві розкрити досить інформації про місцезнаходження, щоб отримати бажану послугу, задовольняючи вищезгадане поняття конфіденційності. Вона базується на узагальненні поняття диференційної приватності. Як диференційна приватність, дане поняття абстрагується від додаткової інформації хакера, які визначаються в залежності від знання про фактичне місцезнаходження користувача.

У поточному прикладі розглянемо користувача, розташованого в Києві, який хоче запросити провайдера LBS запит про оточуючі ресторани без визначення його місцезнаходження, тобто шляхом розкриття деякої приблизної інформації, замість його точного місцезнаходження. Вирішальне питання: які гарантії конфіденційності користувач може очікувати в цьому випадку? Для того, щоб формалізувати це поняття, будемо вважати рівень конфіденційності в радіусі. Тепер ми можемо ввести поняття просторвої нерозрізненості:

Механізм ε-просторово нерозрізненний, якщо для будь-якого радіусу $r > 0$, користувач має певний рівень εr-приватності в межах r.

З визначення слідує, що користувач захищений в межах будь-якого радіуса з рівнем $l = \varepsilon r$, що зростає зі збільшенням відстані. При невеликому радіусу, наприклад, r = 1 км, l малий, це гарантує, що провайдер не може визначити мысце розташування користувача в межах. На великих выдстаннях від користувача, наприклад, при r = 1000 км, l стає великим, дозволяючи постачальнику LBS зробити висновок, що з високою ймовірністю користувач знаходиться в Києві, а не, скажімо, в Лондоні.

Розглянутий в подальшому механізм дозволяє досягати просторової нерозрізненості за додавання лапласівского шуму. Розглядаючи певну лінійну версію розподілу Лапласа, що дозволяє малювати точки в просторово нерозрізненним способом. Крім того, це можена зробити ефективно, з використанням полярних координатах. Однак, як стандартні (цифрові) додатки вимагають кінцевого представлення місць, тому необхідно задати крок дискретизації. Така операція ставить під загрозу гарантії конфіденційності, з

причин, аналогічних ефектів округлення кінцевої точності операцій. Розглянемо, як зберегти просторову нерозрізненость, з урахуванням зменшення рівня конфіденційності, і як налаштувати параметри конфіденційності для того, щоб отримати бажаний рівень конфіденційності.

Розглянемо соціологічне дослідження механізму конфіденційності в контексті LBSs. Припустимо просту клієнт-серверну архітектуру, де користувачі обмінюються даними через довірені мобільні додатки (клієнт - зазвичай встановлений в смартфон) з невідомим / ненадійного постачальником LBS (сервер - як правило, працює на хмарі). Надалі будемо розрізняти помірно-позиційно чутливі і далеко розташовані додатки, чутливі до LBS. Перша категорія відповідає додаткам LBS, що пропонують послуги, які не сильно покладаються на точність інформації про місцезнаходження, надані користувачем. Приклади таких додатків є додатком прогнозу погоди і додатки LBS для вилучення певного виду POI (наприклад, заправних станції). Посилення такого роду LBSs з просторовою нерозрізненістю вимагаються тільки, щоб заплутати розташування користувача за допомогою механізму розподілу Лапласа.

Даний приклад працює коли об’єкт знаходиться в межах другої категорії: Для користувача, який сидить в кафе Два Маго, інформація про довколишні ресторани несе значно меншу цінність, ніж інформація про ресторанаи навколо його місця розташування. Підвищення рівня розташування чутливої LBSs з гарантією конфіденційності є більш складним завданням. Даний підхід полягає в реалізації наступних трьох етапів:

1. Реалізація механізму розподілу Лапласа на клієнтському додатку, щоб повідомити серверу LBS дані користувача з заплутаною місцевістю, а не його реальне місце розташування x.

2. У зв'язку з тим, що інформація, отримана від сервера про POI навкруги певного радіусу, площа інформаційного пошуку POI повинна бути збільшена. Таким чином, якщо користувач бажає отримати інформацію про POI в межах, скажімо, 300 м від x, клієнтського додатку він повинен буде запросити інформацію про POI в межах 1 км . Називемо область інтересів (AOI) і як область пошуку (AOR).

3. Клієнтську програму повинні фільтрувати з урахуванням витягнутої інформації POI для того, щоб забезпечити користувача з необхідною інформацією. AOI повинен завжди повністю міститься в AOR. На жаль, через імовірнісний характер такого механізму збурень, ця умова не може бути. Таким чином, для того, щоб забезпечити просторову нерозрізненість, то AOR повинен бути визначений незалежно від випадкового розташування. Так як ми не можемо гарантувати, що AOI повністю міститься в AOR, введемо поняття точності, яке вимірює вірогідність такої події. Надалі ми будемо посилатися на додаток LBS в абстрактних термінах, так як характеризується механізмом розташування K і фіксованим радіусом AOR.

1.2 Диференційна приватність

Диференційна приватність є поняттям приватності з області статистичних баз даних. Його мета полягає в тому, щоб захистити дані індивідуума при публікації інформації про базу даних. Диференційна приватність вимагає, щоб зміни даних одного користувача мали незначний вплив на результат запиту. Тобто, щоб ймовірність того, що запит повертає значення V, коли застосовується до бази даних D, в порівнянні з імовірністю, щоб повідомити те ж значення при застосуванні до сусідніх D^0 бази даних, була менше зазначеного значення. Тобто різниця між D, D^0 була в межах похибки даних однієї людини. Досягти цього можна за допомогою додавання контрольованого випадкового шуму на виході запиту, наприклад, шуму Лапласа. Перевага цього поняття в тому, що механізм може бути показаний диференційовано приватними незалежно від будь-якої додаткової інформації якою може володіти зловмисник.

Диференційна приватність також використовується в контексті конфіденційного розташування. Синтетичний метод генерації даних може бути використаний для публікації статистичної інформації в моделях диференційної приватності. Метод просторового розкладу використовується для забезпечення диференційної конфіденційності в базі даних з шаблоном локалізації. Диференційна

конфіденційність може успішно застосовуватися, коли публікується агрегована інформація про кількох користувачів. З іншого боку, характер цього поняття погано підходить при застосуванні, в яких тільки одна людина бере участь. Секрет в даному випадку розташування одного користувача. Таким чином, диференціальна приватність вимагає, щоб будь-яка зміна в цьому місці мала незначний вплив на результат, що унеможливлює повідомити будь-яку корисну інформацію від бази даних.

Ця проблема вирішується шляхом фіксації анонімності місцезнаходження і вимагаючи, щоб ймовірність повідомити розташування r будь-якого з цих місць було однаковим.

Це властивість досягається шляхом додавання шуму Лапласа до кожної декартової координати незалежно один від одного. Існують, однак, дві проблеми. По-перше, вибір анонімності, по-друге, сама властивість досить слаба;

1.3 *K*-анонімність

Поняття к-анонімності є найбільш використовуваним визначенням приватності для позиційних систем в літературі. Багато систем в цій категорії спрямовані на захист особистості користувача, вимагаючи, що зловмисник не може зробити висновок, що користувач виконує запит, серед безлічі *K* різних користувачів. Такі системи виходять за рамки нашої проблеми, так як ми зацікавлені в захисті місця розташування користувача. З іншого боку, *к*-анонімність також використовується для захисту місця розташування користувача (іноді називається *к*-різноманітими в даному контексті), вимагаючи, щоб він не відрізнявся серед безлічі K точок. Один із способів для досягнення цієї мети є використання фіктивних місць розташування. Ця методика включає в себе генерацію K - 1 правильно вибраних фіктивних точок, а також виконання запитів K до постачальника послуг, використовуючи реальні і фіктивні розташування. Інший спосіб досягнення *к*-анонімності через. Це

передбачає створення регіону невидимості, який включає до пунктів обміну деякого інтересу, а потім запитом постачальник послуг маскує його для цього регіону.

Навіть якщо сторона знання не проявляється в явній формі в визначенні *к*-анонімності, система не може бути доведена, щоб задовільнити цю ідею, якщо припущення не зроблені про додаткову інформацію зловмисника. Наприклад, фіктивні розташування корисні, якщо вони виглядають однаково, ймовірно, буде реальне місце розташування з точки зору зловмисника. Будь-яка додаткова інформація, дозволяє виключити будь-яку з цих точок, які мають низькі ймовірності, що це реальне місце.

Часто використовуються контрзаходи, щоб уникнути цієї проблеми: наприклад, враховують такі поняття, як повсюдність, затори і однорідність для створення фіктивних точок, в спробі змусити їх виглядати реалістично. Таким же чином, враховується інформацію зі сторони користувача, щоб побудувати область невидимості. Такі контрзаходи мають свої недоліки: по-перше, вони ускладнюють обрахунки, також потрібні додаткові дані, які повинні бути прийняті до уваги (наприклад, точну інформацію про навколишнє середовище або місце розташування найближчих користувачів), що робить їх застосування в режимі реального часу за допомогою портативного пристрою складним. Крім того, фактична сторона інформації атакуючого може бути просто несумісна з припущенням, що робиться.

Захист наборів розташування. Раніше ми розглядали варіант в якому користувач має єдине місце, в якому він хоче зв'язуватися з провайдером послуг в залежності від його місцерозташування . На практиці, набір розашування є загальним для користувача, який матє кілька точок інтересу, наприклад набір останніх місць або набір місць він часто відвідує. У цьому випадку користувач може побажати повідомити провайдеру деяку інформацію, яка залежить від усіх точок; це може бути або вся сукупність точок, або будь-яка загальна інформація, наприклад, їх центроїда. Як і з однією локацією, конфіденційність залишається обов'язковою вимогою; постачальник має право отримати тільки приблизну інформацію про місця, коли їх точне значення має бути конфіденційне. У цьому розділі

розглядається, як просторова нерозрізненність поширюється на випадок, коли локації є кортежем точок $x = (x_1, \dots, x_n)$,.

Як і в випадку однієї точки, поняття відстані має вирішальне значення для нашого визначення. Визначимо відстань між двома кортежами точок $x = (x_1, \dots, x_n)$, $x' = (x'_1, \dots, x'_n)$, як $d_\infty(x, x') = \max_i d(x_i, x'_i)$

Зрозуміло, що вибір метрики буде в межах радіусу r: при $d_\infty(x, x') \leq$ r, це означає, що всі x, x' перебувають в межах відстані на r один від одного. Прослуховування l-приватності в межах радіусу r означає, що два кортежа не більше ніж відстані на відстані r один від одного. Ми можемо зробити запит до кортежу точок, шляхом незалежного застосування існуючого механізму K^0 до кожної окремої точки, і повідомити необхідний кортеж. Виходячи з кортежу $x' = (x'_1, \dots, x'_n)$, незалежно один від одного застосуємо K^0 до кожного X і отримаємо точку z_i, а потім отримаємо кортеж $z = (z_1, \dots, z_n)$. Таким чином, ймовірність того, що комбінований механізм K^0 повідомляє z, починаючи від x, є функцією від ймовірності, щоб отримати кожну точку z_i, починаючи з відповідною точкою x_i, тобто $K(x)(z) = \prod_i K^0(x_i)(z_i)$. Наступне питання в тому, який рівень приватності K задовольняє умова. Для простоти розглянемо кортеж тільки двома точками (x_1, x_2),, і припустимо, що K^0 задовольняє просторовій нерозрізненості. На перший погляд, можна було б очікувати, що комбінований механізм K також задовольняє – просторовій нерозрізненості, проте це не так. Проблема полягає в тому, що ці дві точки можуть бути пов'язані між собою, таким чином, що спостереження про x_1 буде розкривати інформацію про x_2, і навпаки. Розглянемо, наприклад, крайній випадок, в якому $x_1 = x_2$. Маючи два зауваження щодо тієї ж точки це знижує рівень конфіденційності, тому ми не можемо очікувати, що комбінований механізм забезпечить той же рівень секретності.

Проте, якщо K^0 задовольняє просторовій нерозрізненості, то можна показати, що K^0 буде задовільняти просторовій нерозрізненості, тобто рівень конфіденційності, який встановлює масштаб лінійний відносно n. Завдяки цій масштабованості, незалежно один від одного методика застосування механізму до

кожної точки корисна тільки коли число точок невелике. Проте цього достатньо для деяких додатків, таких як тематичне дослідження. Однак слід зазначити, що цей метод ні в якому разі не краще. Аналогічно стандартам диференційної приватності, кращі результати могли б бути досягнуті шляхом додавання шуму на весь кортеж x, замість додавання до кожної окремої точки. Ми вважаємо, що за допомогою таких методів, ми можемо досягти просторової нерозрізненості для великої кількості місць з розумним шумом, що призводить до практичних механізмів для дуже мобільних додатків.

Звітність агрегатного розташування це ще один цікавий випадок, коли потрібно повідомити деяку загальну інформацію, отриману від x, наприклад центроїда кортежу. Загалом, потрібно буде повідомити результат запиту $F: X^n \rightarrow X$. Як і в разі стандартного диференціала конфіденційності можна обчислити реальну відповідь f(x) і додавання шуму шляхом застосування механізму K до нього.

При роботі з агрегатними даними, стандартний диференціал конфіденційності стає життєздатним варіантом. Проте, необхідно також вивчити втрату корисності, викликаної додаванням шуму. Це в значній мірі залежить від застосування: диференцйна конфіденційність підходить для публікації агрегованих запитів з низькою чутливістю, а це означає, що зміни в даних однієї особи мають відносно невеликий вплив на результаті.

З іншого боку, інформація про місцезнаходження часто має високу чутливість. Найпростіший приклад є випадком, коли ми хочемо опублікувати повний кортеж точок. Але чутливість може бути високою навіть для агрегованої інформації: розглянемо випадок публікації центроїди 5 користувачів, розташованих в будь-якій точці світу. Зміна одного користувача може дуже вплинути на їх центр ваги, забезпечивши тим самим диференційну приватність з вимогами додавання шуму Для просторової нерозрізненості необхідно враховувати відстань між точками при обчисленні чутливості. У разі центроїда, невеликі (з точки зору відстані) зміни в кортежі мають невеликий вплив на результат, таким чином, просторова нерозрізненість може бути досягнута з набагато меншим рівнем шуму.

1.4 Просторова нерозрізненість

Основна ідея цього поняття в тому, що для будь-якого радіусу r> 0, користувач має певну приватність в межах r, цей рівень конфіденційності пропорційний радіусу. Зауважимо, що параметр відповідає рівню секретності на одну одиницю відстані. Для користувача вимоги до конфіденційності описуються кортежем (l, r), де r це радіус в якому він знаходиться, l це рівень конфіденційності який він бажає для заданого радіусу. У цьому випадку досить вимагати e= l/r. Це дозволить забезпечити рівень конфіденційності в межах r і відповідний рівень для всіх інших радіусів.

Спочатку вводиться проста модель. Почнемо з множини точок альтернатив X, як правило можливі місця розташування користувача. Крім того, нехай Z буде безліччю можливих повідомлених значень, які в загальному випадку можуть бути довільними, що дозволяє повідомляти місце розташування даної області, набори місць і т.д. Проте, для спрощення обрахунків іноді розглядають Z також як просторові точки.

В подальшому для спрощення задачі в нас множина Z буде співпадати з X. Так як таке припущення не впливає на розв'язок подальшої задачі.

В подальшому досить раціонально користуватися ймовірностями. Наприклад, у випадку коли зловмисник може мати додаткову інформацію про місцезнаходження користувача, знаючи, що він, ймовірно, буде відвідувати майдан Незалежності, в той час навряд чи буде купатися в Дніпрі. Побічна інформація атакуючого може бути змодельована за допомогою попереднього розподілу π на X, де π (x) ймовірність присвоєння місцезнаходження x.

Механізм K є ймовірнісною функцією для вибору звітованого значення. Тобто K є функцією присвоєння кожному розташуванню $x \in X$, розподіл ймовірностей на Z, де K (x) (Z`) є ймовірність того, що повідомлення про точку належить множині $Z` \subseteq Z$, при відомому місце розташування користувача x.

Порівняємо продуктивність даного механізму іншими. Звичайно, не цікаво, порівнянювати з точки зору просторової нерозрізненості, так як інші механізми, як

правило, не задовольняють цим властивостям. Порівняємо з наступними механізмами:

1. Механізм заплутування представлений. Цей механізм працює в окремих місцях, так званих регіонах, він повідомляє місцезнаходження (регіон), що вибирається випадковим чином відповідно до розподілу ймовірностей, яка залежить від місця розташування користувача. Розподіл генерується автоматично за допомогою інструменту, який призначений, щоб забезпечити оптимальну конфіденційність для заданої якості обслуговування і даного зловмисника. Важливо відзначити, що в присутності іншого супротивника оптимальність не гарантовано. Ця залежність від попереднього є ключовою відмінністю по відношенню до даного підходу, який абстрагується від побічної інформації противника.

2. Механізм простої невидимості. При такому підході, область інтересу розділена на зони, розмір яких залежить від рівня секретності який ми хочемо досягти. Потім механізм повідомляє зону з точним місце розташування об'єкта. Цей метод задовольняє к-анонімність де К числа місць в межах кожної зони.

В обох випадках ми повинні розділити область інтересу на кінцеве число областей, що представляють можливі місця.

1.5 Оптимальний просторово нерозрізненний механізм

Побудований механізм, має в якості введення і виведення попередньо визначену кінцеву множину X розташуваннь. Так, наприклад, X може бути побудований шляхом ділення карти в кінцевому числі областей (довільного розміру і форми), і вибору в X репрезентативного місця для кожного регіону. Також припускається, що π передує над X, що представляє ймовірність користувача, який є в кожній локації, в будь-який момент часу. Цей приклад є аналогом добре відомого прикладу Террі Гросс з огляду на приватність метрики d_X (як правило, евклідова метрика) і параметр конфіденційності, мета полягає в тому, щоб побудувати d_X-механізм K таким чином, що втрата якості обслуговування по відношенню до

показника якості d_Q є мінімальним. Ця властивість формально визначається наступним чином:

З урахуванням попереднього π , метрики d_X, параметру конфіденційності і метрику якості d_Q, механізм К OptQL (π, d_Q) тоді і тільки тоді:

1. К: d_X-приватність
2. Для всіх механізмів K^0, якщо K^0 d_X –приватність: QL (К, π, d_Q) ≤ QL (K^0, π, d_Q)

Отже, можна побудувати оптимальний механізм вирішення задачі лінійного прогламування, мінімізуючи QL (К, π, d_Q), задовольняючи d_X -приватність:

$$\sum_{x,z\in X} \pi_x k_{xz} d_Q(x,z) \to min \qquad (1.1)$$

$$k_{xz} \le e^{\varepsilon d_x(x,x')} k_{x'z} \qquad x,x',z \in X \qquad (1.2)$$

$$\sum_{x,z\in X} k_{xz} = 1 \qquad x \in X \qquad (1.3)$$

$$k_{xz} \ge 0 \qquad x,z \in X \qquad (1.4)$$

Легко бачити, що механізм К, породжений попереднім завданням оптимізації і є шуканий OptQL (π, d_Q).

Висновки до розділу 1

В цьому розділі було детально розглянуто такі поняття як просторова нерозрізненість, диференційна приватність, локальна приватність. Було введено поняття просторово нерозрізненого механізму, а також описано, як звести даний механізм до механізму оптимізації. Також описано роль і важливість даної задачі в наш час, адже дана проблема досліджується багатьма серйозними компаніями. З метою знаходження балансу між наданням інформації і захисту приватності.

2 ВИБІР МЕТРИКИ

Служби, розташовані на основі місцезнаходження, піднімають важливі питання конфіденційності стосовно приватної інформації, що виявляє факт виявлення точного місцезнаходження для постачальників послуг. Щоб захистити конфіденційність користувачів, академічне співтовариство запропонувало різноманітні механізми збереження конфіденційності в місцях, які в основному працюють на зміні фактичного місцезнаходження користувачів перед їх виявленням постачальник послуг.

Оцінка конфіденційності цих пропозицій зазвичай не вважає стратегічним противником, сприяючи гонку озброєнь, в якій оборона та напади досягають один одного без будь-яких чітких гарантій конфіденційності місця проживання. Щоб протидіяти цьому ефекту, останні зусилля спрямовані на те, щоб скоротити гонку озброєнь, або вбудуючи вражаючі знання про процес проектування, або надаючи гарантії, незалежні від попереднього супротивника.

Зосередимо увагу на спорадичних механізмах захисту користувачів, орієнтованих на рандомізацію, які зберігають конфіденційність, повідомляючи про шумну версію реального місцезнаходження постачальнику послуг відповідно до розподілу ймовірності. Ці механізми є адекватними для додатків, що вимагають нечастого виявлення місця, і можуть виконуватись локально користувачем. У цьому сценарії підходи, в яких враховуються змагальні знання в процесі проектування, базуються на байєсовському моделюванні противника і знаходимо оптимальні механізми, що генерують шум, за допомогою лінійної оптимізації, в якій шукають цільову цільову цільову конфіденційність при наявності обмежень у використанні. З іншого боку, підходи, які забезпечують конфіденційність гарантій, незалежно від попереднього супротивника, базуються на геоінвідуальності, адаптації диференційованої конфіденційності до двовимірних просторів, що використовується низкою робіт.

Геонерозрізненність може бути досягнута оптимально з точки зору корисності за допомогою дорогого лінійного програмування, або недостатньо оптимально,

використовуючи ефективні методи переробки, що підвищують корисність запиту. Нарешті, підходи Байєса та підходу до геоіндивідуальності були поєднані Шокрі з метою отримання механізмів, що гарантують відмінності геоінвідуальності при досягненні гарної ефективності проти байєсівського противника. За рекомендацією Shokri et al, яка була прийнята спільнотою стандартом, всі ці підходи використовують правильність супротивника, тобто, наскільки сума оцінки супротивника відповідає правильній відповіді, для оцінки конфіденційності місця розташування.

Як правило, правильність супротивника вимірюється як її очікувана помилка оцінки, де ця помилка моделюється за допомогою деякої метрики відстані між реальним розташуванням та оцінкою супротивника. У даній роботі ми прагнемо розуміти властивості механізмів, що виводяться за допомогою цих стратегій розробки. Цільове поняття конфіденційності є очікуваною помилкою оцінки, супротивником є багато оптимальних механізмів, які відповідають бажаним обмеженням втрат якості. Хоча це може здатися вигідним, ми показуємо, що після такої оптимізації ціль може призвести до вибору наївних механізмів, які, очевидно, забезпечують мало конфіденційності, наприклад, чергуючи експозицію фактичного розташування користувача та далекого розташування. Дійсно, цей механізм відповідає в середньому обмеженням проблеми.

Тим не менш, це призводить до невеликої невизначеності для супротивника, фактично забезпечуючи фальшивим сприйняттям конфіденційності. Для протидії такому ефекту ми стверджуємо, що в залежності від уподобань користувача пошук для оптимального механізму збереження конфіденційності місцеположення повинен враховувати більше критеріїв, ніж помилка, що суперечить вірі, встановленої Шокрі та ін. Як приклади додаткових метрик для керування розробкою механізмів захисту ми пропонуємо використовувати інформаційно-теоретичні показники, наприклад, умовну ентропію, або найгіршу міру втрати якості. Розглянемо ефективні методи побудови механізмів щодо цих критеріїв та демонструємо, що метод відновлювання, введений для покращення корисності методів, що базуються на геоінформаційності,

насправді є простою загальною схемою побудови оптимального механізму з точки зору очікувана помилка оцінки з будь-якого механізму обфускації.

Оцінимо ефективність різних механізмів відповідно до різних критеріїв конфіденційності, використовуючи два реальні набори даних про місцезнаходження, в результаті чого, загалом, механізми, оптимальні для одного критерію, не обов'язково добре працюють на інших.

2.1 Метрика, що базується на функції втрати якості

Ми розглянемо два можливі визначення втрати якості: середні втрати та найгірша втрата. Для цього ми введемо dQ (x, z) - функцію, яка кількісно визначає, якою мірою якість послуги втрачається користувачем, який повідомляє адресу виведення z, коли вона зацікавлена в місці вводу x. Більші величини dQ (x, z) вказують на більшу втрату, а отже - і гіршу корисність для користувача. Канонічним вибором для цієї функції є евклідова відстань: dQ (x, z) = || x? z || 2. Зверніть увагу, що dQ (·) не повинно бути метрикою в математичному сенсі: це може бути будь-яка функція, яка вказує місце введення та виведене місце в значення втрати (наприклад, показник корисності на основі почуття.

Середня втрата вимірює якість, яку користувач втрачає в середньому, і може бути записаний як:

$$\bar{Q}(f,\pi) = \sum_{x \in X} \int_0^{R^2} \pi(x) \cdot f(z|x) \cdot d_Q(x,z)\, dz$$

Цей показник є типовим вибором корисності в пов'язаній літературі, оскільки це дуже інтуїтивно. Ця метрика також має перевагу в тому, що вона є лінійною за допомогою механізму f, що дуже корисно для зменшення обчислювальних витрат алгоритмів проектування механізму. Більше того, він робить аналіз оптимальних алгоритмів з точки зору середньої втрати, що піддається розрахунку.

Враховуючи функцію, яка кількісно визначає точку, як визначено вище, $d_Q(x,z)$, найгірший втрата визначається як: з

$$Q^+(f,\pi) = \max_{\substack{x,z \\ \pi(x)>0 \\ f(z|x)>0}} d_Q(x,z)$$

Найгірший випадок втрати означає, скільки утиліти втрачає користувач у найгіршому випадку. Наприклад, якщо $d_Q(x,z)$ - евклідова відстань, і користувач хоче запитати про x, то механізм з $Q^+(f,\pi) \leq 2$ км забезпечує, щоб вихід z не перевищував 2 км від x. Ця властивість дуже корисна для багатьох додатків, які орієнтовані на послуги, що знаходяться поруч із типом послуг, оскільки якщо зареєстроване місце розташування знаходиться дуже далеко від потрібного розташування, то результат запиту для користувача загалом буде марним.

2.2 Приватні метрики

Ми представляємо зараз три поняття про конфіденційність: середня помилка противника, умовна ентропія заднього розподілу та геонерозрізненність. Середня помилка. Середня помилка - де-факто стандарт для вимірювання конфіденційності місця, оскільки Shokri та ін. стверджував, що некоректність визначає конфіденційність користувачів. Подумайте, що противник знає попереднє? і механізм f, обраний користувачем. За допомогою цієї інформації вона видає оцінку $\acute{x} \epsilon \acute{X}$ розташування входу користувача $\acute{x}$. Вибір $\acute{X}$ залежить від обчислювальної потужності супротивника. Оскільки ми припускаємо, що користувач має право повідомляти про будь-яке місце розташування в R^2, ми також припускаємо необмеженого супротивника, який може оцінити місцезнаходження на всьому світі $\acute{X} = R^2$. При спостереженні z противник може побудувати функцію масової маси задніх значень над входами, позначені як p (x | z):

$$p(x|z) = \frac{\pi(x) \cdot f(z|\, x)}{\sum_{x\acute{\epsilon}X} \pi(\acute{x}) \cdot f(z|\acute{x})}$$

Нехай $d_p\,(x,\, \acute{x})$ - це функція, яка кількісно визначає величину помилки супротивника при визначенні місця розташування входу $\acute{x}$ коли вхідне місце насправді x. Як і в випадку середньої втрати $\bar{Q}$, ця функція $d_p\,(\cdot)$ необов'язково

повинна бути метрикою (наприклад, вона може включати в себе чутливість користувача до семантичної інформації, що вступає в супротивника, таку як у). З огляду на розташування виходу z, оптимальне рішення для противника з точки зору мінімізації середньої похибки є

$$\acute{x}(z) = \underset{\acute{x}\epsilon R^2}{\operatorname{argmin}} \left\{ \sum_{x\epsilon X} p(x|\, z) \cdot d_p(x,\, \acute{x}) \right\}$$

Похибка середнього супротивника або лише середня помилка визначається як середня помилка, понесена супротивником, який обирає оцінку x оптимально для кожного спостережуваного z. Нехай $f_Z(z) = \sum_{x \epsilon X} \pi(x) \cdot f(z \mid x)$ - функція щільності імовірності z. Тоді середня помилка:

$$P_{AE}(f, \pi) = \int_0^{R^2} f_Z(z) \sum_{x \epsilon X} p(x|z) \cdot d_p\big(x,\, \acute{x}(z)\big) dz$$

$$= \int_0^{R^2} \min_{\acute{x} \epsilon R^2} \left\{ \sum_{x \epsilon X} \pi(x) \cdot f(z|\, x) \cdot d_p(x, \acute{x}) \right\} dz$$

Зверніть увагу, що механізми, розроблені з P_{AE}, по суті захищають від стратегічного противника, оскільки в метриці входить оцінка противника. Цей показник був використаний як частина об'єкта дизайну в попередніх роботах, а також як спосіб порівняння ефективності в плані конфіденційності механізмів, розроблених з урахуванням інших цілей конфіденційності.

Умовна ентропія - це інформаційно-теоретичний показник, який може використовуватися для вимірювання невизначеності супротивника щодо реального розташування користувача при звільненні z. Після спостереження z противник створює задній $p(x \mid z)$ за допомогою. Невизначеність супротивника щодо значення x заданої z може бути виміряна як ентропія цього заднього:

$$H(x|\, z) = -\sum_{x \epsilon X} p(x\,|z) \cdot log(p(x\,|z))$$

Умовна ентропія вимірює середню ентропію заднього після звільнення z. Формально

$$P_{CE}(f,\pi) = \int_0^{R^2} f_Z(z) \cdot H(x|z)\, dz$$

де $f_Z(z)$ - функція щільності імовірності z, а $H(x|z)$ - функція z. Альтернативно, використовуючи лише попередні π та механізм f, умовна ентропія може бути записана як

$$P_{CE}(f,\pi) = -\sum_{x \in X} \int_0^{R^2} \pi(x) \cdot f\,(z|x) \cdot log\left(\frac{\pi(x) \cdot f\,(z|x)}{\sum_{\acute{x} \in X} \pi(x') \cdot f\,(z|x'))}\right) dz$$

Зауважте, що цей показник не залежить від географії проблеми, тобто від конкретних значень x або z. Якщо ми використовуємо логарифм basetwo у формулі, то P_{CE} може інтерпретуватися як скільки бітів інформації, яку супротивник потребує в середньому, щоб повністю визначити x. Цей показник було знято як можливий показник приватності в через незв'язаність із середньою помилкою. У цій роботі ми викриваємо такий висновок, який вказує на те, що з огляду на виключно правильність супротивника може призвести до розробки механізмів, які забезпечують низьку конфіденційність. У розділі 4 ми показуємо, як використання умовної ентропії як додаткової метрики приватності допомагає уникнути вибору цих небажаних механізмів.

Невідчужуваність геоінформації - це розширення поняття диференційованої конфіденційності, спочатку поняття конфіденційності в базах даних, до сценарію конфіденційності місця розташування. Спочатку він був запропонований в, а інші роботи продовжують дослідження на цій лінії. Формально, гео-невідмінність вимагає наступної умови, щоб бути виконаним механізмом збереження конфіденційності місцезнаходження f,

$$\int f\,(z|x)dz \leq e^{\epsilon \cdot d_P\,(x,x\prime)} \cdot \int f\,(z|x')dz, \forall x, x' \in X, \forall A \subseteq R^2$$

Ця вимога гарантує, що враховуючи площа A $\subseteq R^2$, ймовірність звітування точки z у цій області, якщо оригінальне розташування було x у будь-якому іншому місці $\acute{x}$ в межах деякої відстані навколо x, є подібним, а отже x і $\acute{x}$ мати деяку ступінь статистичної нерозрізненості. У цьому визначенні $d_P\ (x, \acute{x})$ - це функція, яка кількісно визначає як нерозрізнені x та $\acute{x}$: є: менші значення $d_P\ (x, \acute{x})$ вказують на більшу нерозрізненість, оскільки обмеження стає більш жорстким. Параметр конфіденційності в цьому визначенні є: більші значення? вкажіть більш обмежене обмеження, яке дасть можливість $f\ (z \mid x)$ та $f\ (z \mid \acute{x})$ бути більш різними, а отже, x і $\acute{x}$ стати більш помітним. Менші значення? примусити функції щільності імовірності $f\ (z \mid x)$ та $f\ (z \mid \acute{x})$ бути ближчими, забезпечуючи більшу конфіденційність. Зауважте, що якщо для одного місця введення x існує позитивна ймовірність звітування про вихід у регіоні A $\subseteq R^2$, $\int f\ (z \mid x)\ dz > 0$, то це також має бути правдою для будь-якого іншого входу $\acute{x}$. Також зауважте, що гео-нерозрізненість не залежить від попереднього π.

Типовий вибір $d_P\ (x, \acute{x})$ в гео-нерозрізненній метриці є евклідова відстань. Багато механізмів геометричної невідчужуваності залежать від того, що $d_P\ (x, \acute{x})$ - це метрика (зокрема, в тому, що вона задовольняє трикутну нерівність $d_P\ (x, \acute{x}) < d_P\ (x, z)\ + d_P\ (\acute{x}, z)$, щоб довести, що вони відповідають умовам.

Незважаючи на те, що гео-нерозрізненнісь, як правило, вважається гарантією конфіденційності, а не сама по собі метрика, ми можемо адаптувати її, щоб представляти еквівалентну концепцію нашої загальної метрики $P\ (f, \pi)$. З огляду на механізм, який забезпечує? -Гео-невідчужуваність, просто зрозуміти, що це теж $\acute{\varepsilon} > \varepsilon$ гео-нерозрізненність, якщо $\acute{\varepsilon} > \varepsilon$. Так як менше ε означає більшу конфіденційність, має сенс визначити рівень геоіндивідуальності, який забезпечується за механізмом f відповідно до найменшого ε, це гарантує.

Крім того, оскільки ми визначаємо $P\ (f, \pi)$ Як величину, яка зростає з захистом користувачів, ми вирішили визначити нашу міру невідчужуваності

географів, $P_{GI}(f)$, як зворотну від найменшого ε гарантованого механізмом. З огляду на механізм f, виходить

$$P_{GI}(f) = \inf_{\substack{x,x' \in X \\ z \in R^2}} d_P(x, x') \cdot \left| log \frac{f(z|x)}{f(z|x')} \right|^{-1}$$

де ми допускаємо за згодою, що $\log\left(\frac{0}{0}\right) = 0$ і $\|x - \acute{x}\|_2$ це є евклідова відстань. Великі значення P_{GI} вказують на більший рівень приватності, а також механізм гарантує $1/P_{\mathrm{GI}}$ - гео-нерозрізненість.

2.3 Додаткові метрики

Нарешті, ми нарешті окреслюємо інші показники, які можна використовувати разом із середньою помилкою та середньою втратою якості, щоб оцінити конфіденційність механізмів, і залишити розробку механізмів, враховуючи їх як предмет для майбутньої роботи.

Гео-нерозрізненність за своєю суттю гарантує, що місце введення x буде відображено в сусідньому розташуванні з більшою ймовірністю, ніж на далеке розташування, яке вирішує проблему конфіденційності, яку ми проілюстрували за механізмом монети. Однак це поняття конфіденційності несумісне з обмеженням втрат якості в найгірших випадках за визначенням, через те, що $f(z \mid x) > 0$ означає $f(z \mid \acute{x}) > 0,\ \forall \acute{x} \in X$.

Можливим підходом до вирішення цієї корисної проблеми гео-нерозрізненності може бути розслаблення його визначення, що дає можливість незначного значення $\Delta \ll 1$, тобто

$$\int f(z|x)\, dz \leq e^{\varepsilon \cdot d_p(x,\ \acute{x})} \cdot \int f(z|x)\, dz + \Delta, \qquad \begin{matrix} \forall \acute{x} \in X \\ \forall A \subseteq R^2 \end{matrix}$$

Інші цікаві показники для оцінки конфіденційності механізмів - це ті, що базуються на найгіршому випуску. Наприклад, найгірший випадок середня помилка, визначена як

$$P_{WC-AE}(f,\pi) = \min_{\substack{z \in R^2 \\ f_z(z)>0}} \min_{\hat{x} \in R^2} \left\{ \sum_{x \in X} \pi(x) \cdot f(z|x) \cdot d_p(x,\hat{x}) \right\}$$

вимірює середню похибку оцінки супротивника у найбільш уразливому випуску. При застосуванні до монетарного механізму цей показник виявить конфіденційність, оскільки $P_{WC-AE}(f_{coin},\pi) = 0$.

З іншого боку, найгірший випадок умовної ентропії, визначається як

$$P_{WC-AE}(f,\pi) = \min_{\substack{z \in R^2 \\ f_z(z)>0}} \sum_{x \in X} p(x|z) \cdot f(x|z)$$

виявляє невпевненість, яку противник має після спостереження z у гіршому випадку (для користувача). Якщо є будь-яке вихідне значення z, що розповідає багато інформації про реальне розташування x (як це відбувається з кожним $z \neq z^*$ в механізмі монети), цей показник висвітлює його.

Показники, представлені в цьому розділі, додають додаткові параметри до процедури оцінки втрат конфіденційності та якості, виявлення функцій, не відображених у стандартному 2-мірному підході на основі середньої похибки та середньої втрати. Приклад цієї нової характеристики приватності показаний на рис. 2.1, де показана ефективність двох механізмів як 3-D ділянку P_{AE}, P_{CE} та $\bar{Q}$ разом з проекціями в площинах $P_{AE} - \bar{Q}$ та $P_{CE} - \bar{Q}$.

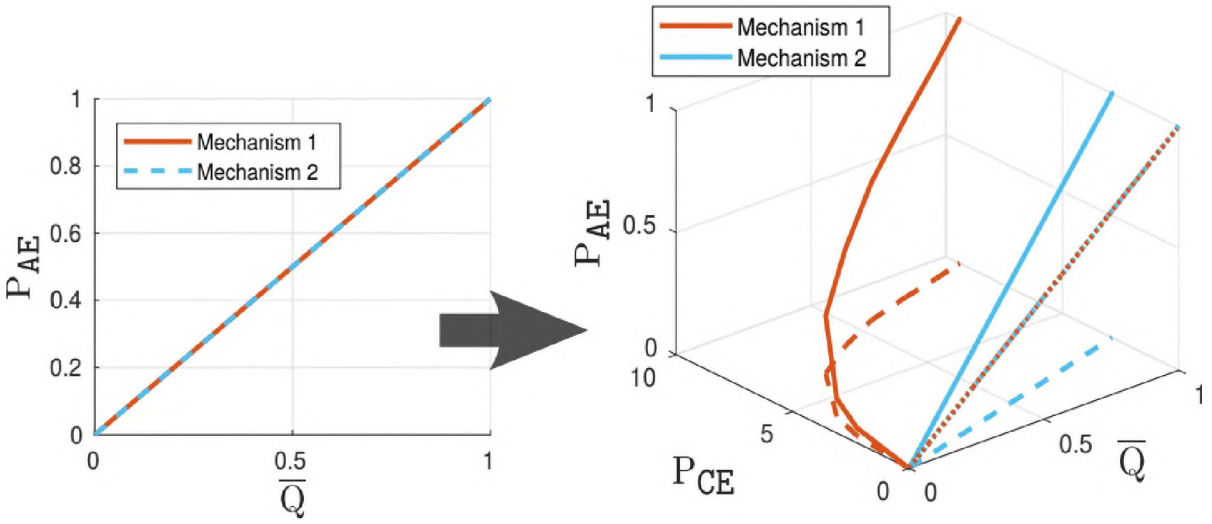


Рисунок 2.1 - Два механізми, які однаково виконуються в площині P_{AE} або $\bar{Q}$, можуть на практиці поводитися дуже по різному. Це проявляється при розгляді багатомірної характеристики конфіденційності

Висновки до розділу 2

В даному розділі було розглянуто різні метрики і описано переваги і недоліки кожної. Було описано, які метрики використовуються в яких випадках. В результаті, було прийнято рішення вибрати модифіковану евклідову метрику. Серйозним варіантом також була метрика, що базується на втраті якості. Вона серед розглянутих метрик дає найкращі результати, але через ускладнення системи, і через те що вона має схожу структуру з евклідовою метрикою. Вибір зупинився саме на ньому. Але як модифікацію можна вибрати і інші розглянуті метрики, або їхню суперпозицію.

3 ФОРМАЛЬНИЙ ОПИС ЗАДАЧІ СИНТЕЗУ СИСТЕМИ НЕРОЗРІЗНЕНОСТІ

В першому розділі було описано оптимальний механізм для отримання локацій, які будуть просторово нерозрізненими і давати найбільше інформації на наш запит. В даному розділі буде розглянуто більш детально саму задачу, а також описано можливі підходи до розгляду задачі, різні особливості і методи вирішення.

Даний розділ включатиме опис і порівняння відомих методів, а також буде описана спеціальна задача лінійного програмування. Особливістю наведеного методу заключається в тому, що він має логарифмічну складність, проте досить специфічну форму задавання початкових даних. Новизна в даному розділі в тому, що раніше не порівнювалися описані методи для даної задачі ніхто не намагався розглядати задачу з точки зору спеціальної задачі лінійного програмування. Описане нижче є важливою необхідністю для детального розгляду задачі і реальних оцінок ефективності певних методів.

3.1 Запис задачі синтезу системи нерозрізненості через ймовірності

Формально задачу синтезу системи нерозрізненості можна описати наступним чином:

Розв'язати задачу лінійного програмування задану формулами 1.1-1.4. Причому сама задача в залежності від n, де n – кількість елементів ранжованої карти, змінює значення невідомих. Що в цілому трішки змінює складність задачі, адже для кожного значення n потрібно перераховувати всі значення для оптимізації.

Для подальшого розгляду скористаємося рядом припущень. В формулі 1.1 ми в зв'язку з тим, що $\pi_x d_Q(x,z)$ це величина стала в рамках оптимізації, тобто параметр сталий тому замінимо його на c_i. З прикладного значення слідує, що величина k_{xz} є аналогом ймовірності розташування об'єкта в конкретній локації i

при відзвітованій *j* локації, тобто препозначимо, як p_{ij}. В результаті можна записати формулу 1.1 наступним чином:

$$\sum_{i,j \in X} c_i p_{ij} \rightarrow min \tag{3.1}$$

Запишемо тепер 1.3 і 1.4 відповідно до попередніх зауважень і враховуючи, що $\varepsilon = 1$, для спрощення адже з математичної точки зору це не сильно вплине остаточний результат. Тому запишемо остаточний результат як:

$$p_{ij} \leq e^{d_x(i,i')} p_{i'j} \tag{3.2}$$

$$\sum_{i,j \in X} p_{ij} = 1 \tag{3.3}$$

3.2 Опис альтернатив розв'язання. Оцінка складності

Дану задачу можна розглядати з багатьох сторін. Одним із варіантів розв'язання задачі може бути симплекс метод. Даний метод розглядався для вирішення даної задачі, але в початковому стані має велику складність. Адже розмірність початкової матриці $n^2 \times (n^3 + n)$, що значно ускладнює задачу в програмній реалізації. Що приблизно рівне n^5, а з врахуванням складності реалізації симплекс методу буде велике значення. Для прикладу час обрахування для n=20 займає близько однієї хвилини, а для значення n=80 необхідно затратити більше 30 хвилин. Це занадто багато, адже для реальних систем це великі значення і користувач не буде чекати пів години для того щоб визначити найоптимальніше значення. З цього слідує, що необхідно трішки спростити дану задачу, або підібрати оптимальніший алгоритм.

Для вирішення даної задачі можна розглянути задачу з точки зору геометрії. Для початку розглянемо нерівність 3.2 і перепишемо в зручнішому вигляді:

$$\frac{p_{ij}}{p_{i'j}} \leq e^{d_x(i,i')}$$

Очевидно, що відношення ймовірностей буде обмежене інтервалом $\left[0, e^{d_x(i,i')}\right]$. Що свідчить про многогранник в n^2-мірному просторі.

З рівняння 3.3 можна побачити, що це площина, що сполучає всі 1 кожної осі в n^2-мірному просторі. Пройшовши через всі точки перетину цих двох площин, отримаємо множину значень, що потенційно можуть бути оптимальними, тобто ми зменшили множину значень для подальшої оптимізації. В результаті отримуємо менше значень для перевірки на оптимальність. Але подібний підхід не особливо покращує попередні результати. Тому він також не підходить для реального використання.

3.3 Спеціальна задача лінійного програмування

Спеціальної задача лінійного програмування детально досліджувалась в роботі [7]. Дана задача досить цікава в даному випадку, тому що своїм виглядом трішки нагадує нашу. Проте відмінністю між ними є те, що в задачі синтезу системи нерозрізненості в нас вона обмежена додатковими параметрами на кінцях інтервалах, що унеможливлює подальшу оптимізацію в поточному вигляді. Хоча складність алгоритму є логарифмічною, що значно відрізняється від всіх попередньо розглянутих методів.

Опишемо саму спеціальну задачу лінійного програмування. Розглядається критеріальний простір розмірності n, в якому ваги кожного з критеріїв $f_1(\omega), f_2(\omega), \ldots, f_n(\omega)$, де $\omega \in \Omega$ - альтернативи, представлені інтервальними оцінками $x_1, x_2, \ldots, x_n$. Потрібно побудувати узагальнений критерій, що мінімізує ризик помилки, пов'язаний з неточністю оцінювання експертом або ЛПР вагових коефіцієнтів критеріїв.

Побудуємо розбиття критеріального простору на $n!$ конусів, кожен з яких задається упорядкуванням виду $f_{j1}(\omega) < f_{j2}(\omega) < \ldots < f_{jn}(\omega)$. Для кожної з внутрішніх точок довільного конусу буде, очевидно, виконуватися $e_1 < e_2 < \ldots < e_n$, де $e_j = f_{ji}(\omega)$.

Розглянемо для заданого впорядкування рішення задачі $\sum_i e_i x_i \to min$, де $0 \le e_i \le 1$; $0 \le \underline{x}_i \le x_i \le \underline{x}_i \le 1$.

Потрібно знайти таке рішення x* завдання, для якого $\sum_{i=1}^{n} x_i^* = 1$.

Остання умова означає, що рішеннями можуть бути точки перетину прямокутного гіперпаралелепіпеда, що задається обмеженнями, гіперплощиною . В силу опуклості області, що задається обмеженнями, її перетин гіперплощина - опуклий багатогранник, число можливих вершин якого від 1 до 2n-2. Кожна вершина багатогранного перетину задається як точка перетину гіперплощини з ребром паралелепіпеда (вершиною перетину не може бути внутрішня точка межі). Запропонована характеризація вершин інваріантна, тобто не залежить від розмірності критеріального простору.

Потрібно знайти таку вершину перетину, в якій лінійна функція приймає мінімальне значення (взагалі кажучи, рішення не є єдиним). Вершини перетину, як було зазначено вище, або лежать на ребрах паралелепіпеда, або збігаються з його вершинами. У першому випадку координати вершини - набір з *n-1* наступних в деякому порядку верхніх і нижніх значень, і однією компоненти x_k виду $\underline{x}_k \le \tilde{x}_k \le \overline{x}_k$: $\overline{x}_1, \underline{x}_2, \dots, \underline{x}_{k-2}, \overline{x}_{k-1}, \tilde{x}_k, \underline{x}_{k+1}, \dots, \overline{x}_n$, а в другому випадку подібна компонента відсутня: $\overline{x}_1, \underline{x}_2, \dots, \overline{x}_k, \underline{x}_{k+1}, \dots, \overline{x}_n$.

Таким чином, завдання зводиться до

а) пошуку єдиної вільної компоненти (якщо рішенням є точка на ребрі)

б) встановлення порядку проходження верхніх і нижніх обмежень для інших компонент.

У тому випадку, коли шуканим рішенням виявляється вершина паралелепіпеда, досить виконати підзадачу (б).

Далі розглядається деякого упорядкування коефіцієнтів $e_1 < e_2 < \dots < e_n$ виконується наступне:

Лінійна комбінація $\sum_{i=1}^{n} e_i x_i$, де $0 < e_1 < e_2 < \ldots < e_n$, $0 \le \underline{x}_i \le x_i \le \underline{x_i} \le 1$ приймає максимальне значення, x^*: $\sum_{i=1}^{n} x_i^* = 1$, в точці перетину з координатами $(\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_{k-1}, \tilde{x}_k, \overline{x}_{k+1}, \ldots, \overline{x}_n)$, і мінімальне значення - в точці виду $(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_{k-1}, \tilde{x}_k, \underline{x}_{k+1}, \ldots, \underline{x}_n)$.

Побудований на першому етапі критерій являє собою безперервну, визначену в будь-якій точці критериального простору функцію виду

$$L(e) = \overline{x}_1 e_1 + \overline{x}_2 e_2 + \cdots + \overline{x}_{k-1} e_{k-1} + \tilde{x}_k e_k + \underline{x}_{k+1} e_{k+1} + \cdots + \underline{x}_n e_n$$

де номер і значення компоненти $\tilde{x}$ визначаються для внутрішніх точок кожного з конусів, що задаються упорядкуванням $f_{j1}(\omega) < f_{j2}(\omega) < \ldots < f_{jn}(\omega)$, згідно з описаною процедурою. Таким чином, завдання прийняття рішень на множині альтернатив з векторними оцінками і заданими інтервально, ваговими коефіцієнтами критеріїв, зводиться до пошуку максимуму лінійної згортки в критеріальному просторі.

Основним способом вирішення задач багатокритеріальної оптимізації є метод агрегування. Такий висновок справедливий і для паретівского аналізу: кожен критерій природним чином визначає бінарне відношення переваги, а потім, як результат агрегування побудованих бінарних відносин, визначається поняття домінування по Парето. Серед інших важливих додатків методу агрегування переваг слід відзначити завдання колективного вибору.

У найпростішому (і найбільш поширеному) варіанті агрегування критеріїв при векторної оптимізації - метод лінійної згортки, використовуються вагові коефіцієнти, походження яких не цілком формалізуються. З цим пов'язані як переваги, так і недоліки методу. Звичайно, подібні процедури містять елемент хаосу. Однак вони результативні і нерідко допомагають знайти цілком задовільні рішення багатокритеріальних задач, які з математичної точки зору є не визначеними. Такі задачі зустрічаються в різних фізичних та соціальних реальних задачах, що викликає великий інтерес з точки зору прикладної математики.

Розглянемо дві обставини, що обмежують можливості коректного використання методу лінійної згортки. А саме, ми не маємо права стверджувати апріорі, що глобальна функція корисності (яка надається агрегованих критерієм) задає саме лінійну залежність від вихідних критеріїв. Однак, ряд результатів теорії корисності дозволяє побудувати з вихідного безлічі критеріїв т.зв. «Локальні функції корисності», і обгрунтувати лінійну залежність від них глобальної функції корисності. Таким чином, незважаючи на певні обмеження, метод лінійної згортки є досить універсальним, результативним і, як наслідок, широко поширеним методом.

Друге обмеження правомірності використання цього методу пов'язане з зазначеним вище евристичним характером процедур оцінювання вагових коефіцієнтів. Призначення цих процедур - як можна більш точно чисельно висловити неформальні, якісні думки експертів. З цього зрозуміло, що такі процедури принципово не можуть бути повністю алгоритмізовані, і завжди будуть мати людино-машинний характер. Методика організації та проведення такого роду процедур спеціально розроблялася і вважається цілком надійною. Було проведено безліч психологічних експериментів, в яких порівнювалися різні методи (не тільки чисто евристичні, але і з аксіоматичним обґрунтуванням) призначення ваг критеріїв. На жаль, вони призводять до різних результатів, що безпосередньо позначається на якості рішення і довірі до нього. Розгляд інтервального підходу дозволяє зменшити жорсткість (підвищити адаптивність) процедури оцінювання вагових коефіцієнтів.

Однак при цьому замість одного агрегованого критерію виникає цілий клас. І рішення задачі вибору найкращої альтернативи включає два етапи. Вибір одного з безлічі критеріїв передбачає врахування додаткових міркувань, що дозволяють зафіксувати значення ваг. Відмінності в способах вирішення цього завдання пов'язані, як правило, зі специфікою вимог, що пред'являються до результату.

Описана процедура побудови вагових коефіцієнтів, очевидно, не є оптимальною з точки зору обчислювальної складності. Для знаходження рішення потрібно зробити послідовний перегляд, в середньому n / 2 компонент вектора оцінок. Поліпшити цей показник до логарифмічного можна за рахунок організації проходу до методу двійкового пошуку. При цьому кількість кроків поліпшеної

процедури становить в середньому $O\ (log\ (n))$. Проте отриманої складності достатньо для користі використання в даній задачі.

Висновки до розділу 3

В даному розділі була сформована задача з математичної точки зору, а також було проведено аналіз відомих методів для пошуку оптимального значення. Була оцінена складність кожного методу. З цього розділу стає видно, що розглянуті методи не можуть бути використані для розв'язання. Дану задачу необхідно спростити або зробити певні маніпуляції для раціональнішої подачі задачі. Також в даному розділі розглядався метод розв'язання спеціальної задачі лінійного програмування, що має допустиму складність. Але не можна використовувати дану реалізацію через значні відмінності між базисними функціями.

4 ДОСЛІДЖЕННЯ ЗАДАЧІ ЛІНІЙНОГО ПРОГРАМУВАННЯ

В даному розділі розглянемо матрицю із ведемо до певного виду. Даний аналіз необхідний для спрощення задачі. Адже як зазначалося в попередньому розділі в даному вигляді розв'язок задачі має велику складність, тому її треба звести до простішої, для можливої подальшої реалізації.

Також в даному розділі розглянемо складність структури задачі після спрощень і оцінимо наскільки це спростить задачу на практиці.

4.1 Особливості структури початкової задачі

Розглянемо для початку нерівність 1.2. Запишемо спочатку дану нерівність в матричній формі задачі лінійного програмування. В результаті ми отримуємо матрицю розмірності $n^2 \times n^3$. При детальному розгляді її структури буде зрозуміло, що значна частина матриці заповнена нулями. І чим більше значення n, тим менше ненульових значень буде в даній матриці. Причому будуть і рядки заповненні нулями, що при програмній реалізації може значно вплинути на складність розв'язання поставленої задачі. Це перше на що потрібно звернути увагу при вивчені матриці. Дана особливість ще проявляється в тому, що в кожній стрічці є лише 2 ненульових значень, окрім вище описаного випадку. Це пояснюється самою структурою формули 1.2.

Іншою особливістю матриці є те, що будуть обернено пропорційні рівняння, що в цілому спрощує першу нерівність і дозволяє нам записати рівняння 1.2 в іншому вигляді:

$$p_{ij} \leq e^{d_x(i,i')} p_{i'j} \qquad i, i', j \in X$$

$$p_{i'j} \leq e^{d_x(i,i')} p_{ij} \qquad i, i', j \in X$$

Запишемо останню нерівність в іншому вигляді:

$$p_{ij} \geq e^{-d_x(i,i')} p_{i'j} \qquad i, i', j \in X$$

Об’єднаємо отримані нерівності і матимемо подвійну нерівність:

$$e^{d_x(i,i')}p_{i'j} \geq p_{ij} \geq e^{-d_x(i,i')}p_{i'j} \qquad i,i',j \in X$$

Для зручності використання отриманої формули зведемо її до більш зручного виду:

$$e^{d_x(i,i')} \geq \frac{p_{ij}}{p_{i'j}} \geq e^{-d_x(i,i')} \qquad i,i',j \in X \tag{4.1}$$

Нерівність 4.1 описує те, що відношення ймовірностей обмежено заданою метрикою.

Зведемо двох індексний запис імовірності до одно індексного. Для цього введемо заміну таким чином, що $p_{ij} = x_k$ і $p_{i'j} = x_{k'}$ при $k = i * n + j$. Запишемо відповідно до введеної заміни формули 1.3, 1.4, 3.1 :

$$\sum_{k \in K} x_k = 1 \qquad x \in X \tag{4.2}$$

$$x_k \geq 0 \qquad x,z \in X \tag{4.3}$$

$$e^{d_x(i,i')} \geq \frac{x_k}{x_{k'}} \geq e^{-d_x(i,i')} \qquad i,i',j \in X \tag{4.4}$$

Ці формули включно з 1.1 описують дану задачу.

4.2 Зведення до спеціальної задачі лінійного програмування

Вигляд задачі має спільне з спеціальною задачею лінійного програмування. І подається вона в такому вигляді:

$$\sum_i a_i x_\text{i} \to extr$$

$$1 \geq \overline{x_i} \geq x_i \geq \underline{x_i} \geq 0$$

$$\sum_i x_i = 1$$

Необхідно звести задачу до такого ж виду. Для цього детально розглянемо формули 1.1, 4.2-4.4. Розпишемо суму:

$$\sum_{k,k'} \frac{x_k}{x_{k'}} = \sum_k \sum_{k'} \frac{x_k}{x_{k'}} = \sum_k x_k \sum_{k'} \frac{1}{x_{k'}} = \sum_{k'} \frac{1}{x_{k'}} = \sum_{k'} x_{k'}^{-1}$$

Даний перехід можливий за рахунок незалежності сум і формули 4.2. Отримана сума величина стала для кожного фіксованого n, тобто вона не залежить від k, k', тому ми можемо записати отримане в іншому вигляді:

$$\sum_{k'} x_{k'}^{-1} = m$$

$$\sum_{k'} (m x_{k'})^{-1} = 1$$

Розглянемо наступну суму:

$$\sum_{k,k'} c_k \frac{x_k}{x_{k'}} = \sum_{k} c_k x_k \sum_{k'} \frac{1}{x_{k'}} = \sum_{k} c_k x_k m = m \sum_{k} c_k x_k \to min$$

Це досягається з урахуванням попередньої формули, а також формули 1.1. Відповідно підсумувавши все і зробивши перетворення поділивши все отримане на величину m отримаємо наступне:

$$\sum_{k,k'} c_k \frac{x_k}{x_{k'} m} \to min$$

$$\sum_{k'} \frac{x_k}{x_{k'} m} = 1$$

$$\frac{e^{d_x(i,i')}}{m} \geq \frac{x_k}{x_{k'} m} \geq \frac{e^{-d_x(i,i')}}{m} \geq 0$$

Замінимо $\frac{x_k}{x_{k'} m}$ на x_t і матимемо:

$$\sum_{t} c_t x_t \to min$$

$$\sum_{t} x_t = 1$$

$$\frac{e^{d_x(i,i')}}{m} \geq x_t \geq \frac{e^{-d_x(i,i')}}{m} \geq 0 \qquad (4.5)$$

Отримана система і є спеціальною задачею лінійного програмування. Тобто для неї існує визначений алгоритм пошуку оптимального значення. І мінімальне значення буде досягатися в точці виду:

$$< \overline{x}_1, \overline{x}_2, \dots, \overline{x}_{l-1}, \tilde{x}_l, \underline{x}_{l+1}, \dots, \underline{x}_k >$$

3.3 Особливості отриманих результатів. Оцінка складності

Дана форма дозволяє нам знайти оптимальне значення виконуючи меншу кількість ітерацій. Складність даного алгоритму *O(log(k))*. Для порівняння з симплекс методом зведемо все до однієї змінної. Так як $k=n^2$, то підставивши, отримаємо складність даного методу, що буде складати $O(log(n^2))$, або $O(2*log(n))$. Якщо порівняти її з складністю симплекс методу то побачимо, що вона буде дуже відрізнятися. Адже розв'язок за допомогою симплекс методу має складність приблизно $O(exp(n^5))$. Тому отримане зведення значно спрощує вирішення даної задачі. Ще однією важливою особливістю є те, що програмна реалізація симплекс методу має свої обмеження, такий алгоритм просто не знаходить оптимального значення при n більших за 80. В таких випадках воно видає помилку в зв'язку з складністю алгоритму. Звичайно можна скористатися спрощеннями і намагатися звести дану задачу до поліноміальної складності, але навіть $O(n^5)$, це вже велика складність.

Перевага в складності очевидна, але окрім складності ми можемо знайти ще ряд цікавих речей. Розглянемо формулу 4.5 і розпишемо дану систему:

$$\sum_t c_t x_t \to min$$

Звернувшись до заміни $\frac{x_k}{x_{k'} m}$ на x_t і виконавши все в оберненому порядку отримаємо наступне:

$$\sum_{k,k'} c_t \frac{x_k}{x_{k'} m} \to min$$

Зафіксуємо величину min як змінну, що залежить від параметрів k, k' і позначимо як $a_{k,k'}$ і зробимо ряд логічних перетворень:

$$\sum_{k,k'} c_t \frac{x_k}{x_{k'} m} \to a_{k,k'}$$

$$\sum_{k'} \frac{1}{x_{k'} m} \sum_k c_t x_k \to a_{k,k'}$$

$$\sum_k c_t x_k \rightarrow a_{k,k'}$$

З врахуванням $\sum_{k'} \frac{1}{x_{k'} m} = 1$, так як раніше її виводили. Так як отриману функцію і потрібно було оптимізувати причому бачимо, що $a_{k,k'}$ не залежить від k', бачимо, що a_k випливає і з умови оптимізації x_k.

Ще однією важливою особливістю отриманої формули є те, що величина $\frac{e^{d_x(i,i')}}{m}$ може перевищувати 1. Але це не змінює сам розв'язок, просто при таких значення достатньо приймати максимальним значенням 1. Тобто не розглядати значення, які не особливо впливають на задачу адже вони виходять за наші обмеження. І дійсно, якщо розглядати це питання з точки зору геометричного представлення, то все що виходить за межі многогранника не матиме перетину з іншою площиною, тобто розв'язків реально бути там не може. А дане обмеження прирівнює всі значення, що виходять за межі.

Також потрібно звернути увагу на $\sum_{k'} x_{k'}^{-1} = m$. Дана припущення стає зрозумілим лише при детальному розгляді. Величини $x_{k'}^{-1}$ це ймовірності виду $p_{i'j}$, які є величинами, що залежать від i', j, проте сума забезпечує перебір по всім величинам ймовірностей. В результаті величина m не буде залежати від i', j, а лише від кількості елементів в проранжованій множині. Ця важлива особливість дозволяє здійснювати багато операцій, наприклад внесення під суму, як сталу величину.

Висновки до розділу 4

В даному розділі було зведено задачу до спеціальної задачі лінійного програмування, що дає змогу досить просто розв'язувати поставлену задачу з логарифмічною складністю. Також було описано особливості даних перетворень.

ВИСНОВКИ

В даній задачі розглядалась задача оптимізації просторово нерозрізненого механізму. В результаті роботи було зведено задачу до спеціальної задачі лінійного програмування, що значно спрощує пошук оптимального значення. Початкова задача виконана в повному обсязі. Даний результат дозволяє з задачі зі складністю n^{10} звести до задачі з логарифмічною складністю. Це дозволяє витрачати менше часу для розрахунків.

Вітчизняних аналогів немає, а якщо порівнювати з світовими аналогами то це значно ефективніше, ніж раніше описані підходи в літературі. Унікальність даної роботи в тому, що ніхто не розглядав матрицю і не намагався спростити задачі, що як виявилося є досить ефективним.

Рекомендую для подальшого дослідження розглянути відношення ймовірностей, що оптимізується і звести до лінійної функцій від ймовірності. Даний результат може дати нові залежності і при правильній інтерпретації це може пояснити деякі особливості задачі.

ПЕРЕЛІК ПОСИЛАНЬ

1 *N. E. Bordenabe, K. Chatzikokolakis, C. Palamidessi.* Optimal geo-indistinguishable mechanisms for location privacy. ACM SIGSAC Conference on Computer and Communications Security, 2014, pages 251-262.

2 *R. Shokri, G. Theodorakopoulos, C. Troncoso, J. P. Hubaux, J. Y. Boudec.* Protecting location privacy: optimal strategy against localization attacks. CCS, ACM, 2012, pages 617–627.

3 *M. E. Andrés, N.E. Bordenabe, K. Chatzikokolakis, C. Palamidessi.* Geo-indistinguishability: differential privacy for location-based systems. CCS, ACM, 2013, pages 901–914.

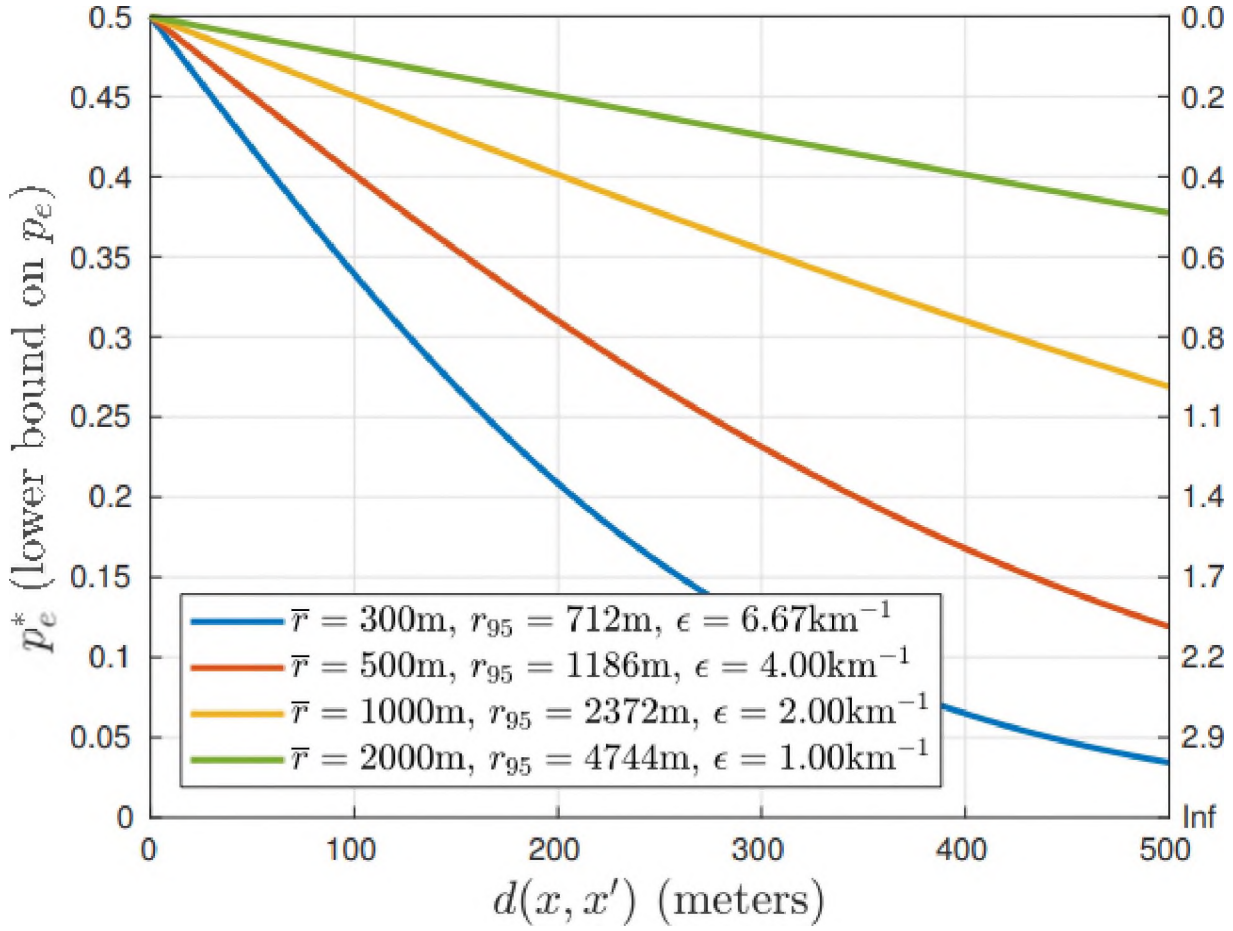
4 *S.S. Ho, S. Ruan.* Differential privacy for location pattern mining. SPRINGL, ACM, 2011, pages 17–24.

5 *M. E. Andrés, K. Chatzikokolakis, N. E. Bordenabe, C. Palamidessi.* Geo-indistinguishability: differential privacy for location-based systems. ACM SIGSAC conference on Computer & communications security, 2013, pages 901-914.

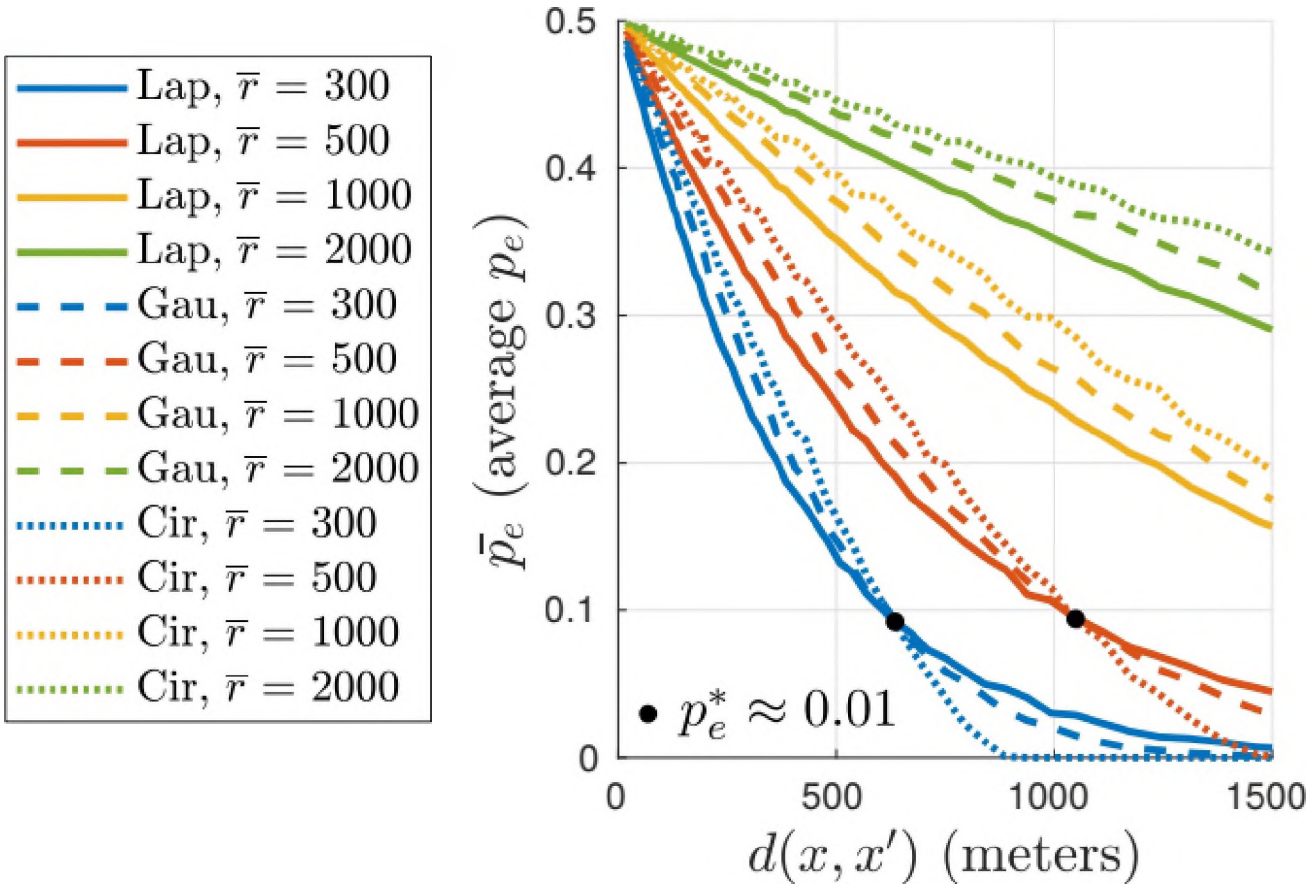
6 *K. Chatzikokolakis, M. E. Andr´es, N. E., Bordenabe, C. Palamidessi.* Broadening the scope of Differential Privacy using metrics. PETS, Springer, 2013, pages 82–102.

7 *S. A. Smirnov, I. S Gontarenko.* Guaranteed synthesis of scalar criterion for multicriteria optimization problem . Ukraine, Kiev, 2006, pages 99–106.

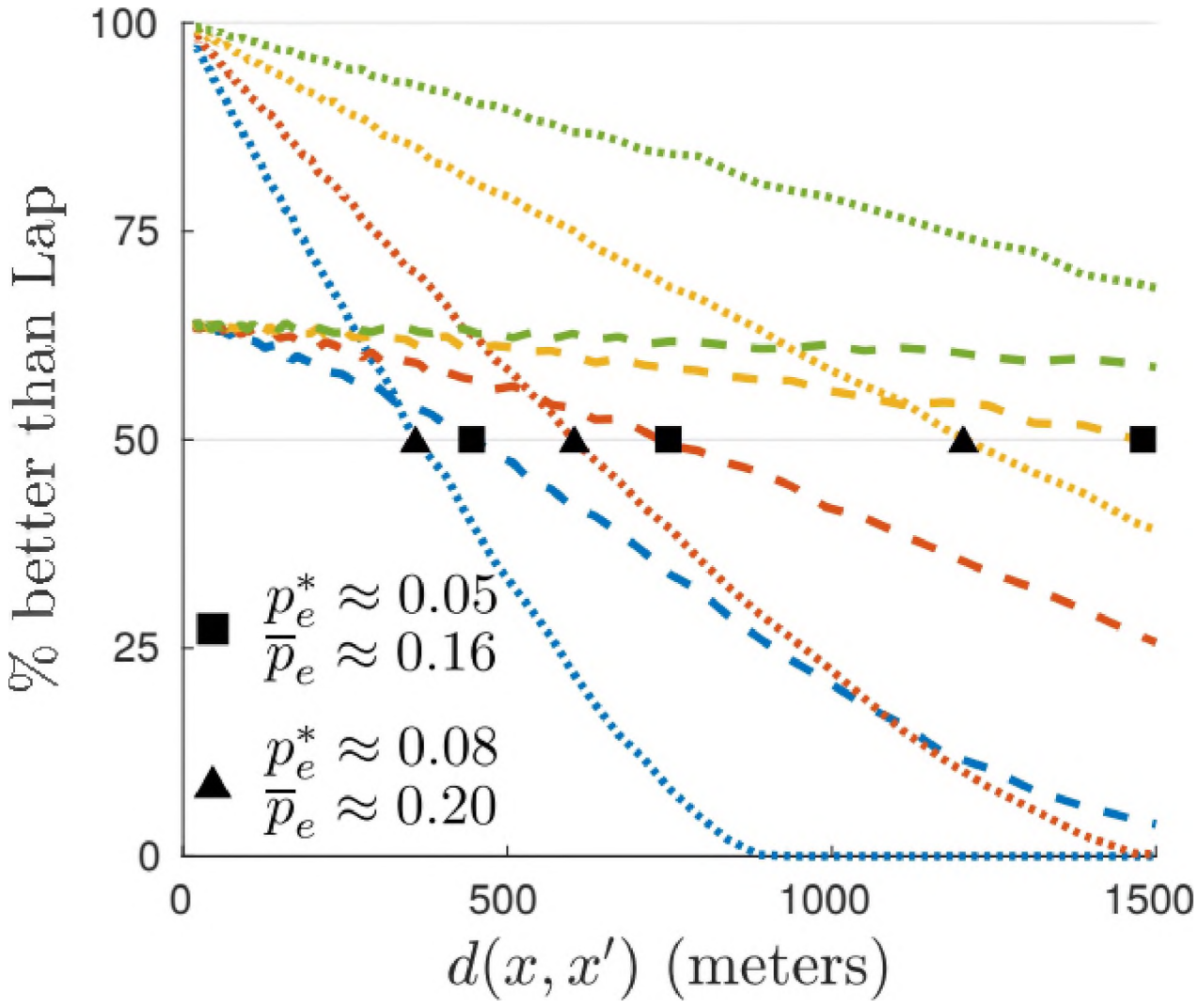
ДОДАТОК А.1


0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0.0
0.2
0.4
0.6
0.8
1.1
1.4
1.7
2.2
2.9
Inf
0
100
200
300
400
500
p_e^* (lower bound on p_e)
$d(x, x')$ (meters)
$\overline{r} = 300\text{m}$, $r_{95} = 712\text{m}$, $\epsilon = 6.67\text{km}^{-1}$
$\overline{r} = 500\text{m}$, $r_{95} = 1186\text{m}$, $\epsilon = 4.00\text{km}^{-1}$
$\overline{r} = 1000\text{m}$, $r_{95} = 2372\text{m}$, $\epsilon = 2.00\text{km}^{-1}$
$\overline{r} = 2000\text{m}$, $r_{95} = 4744\text{m}$, $\epsilon = 1.00\text{km}^{-1}$

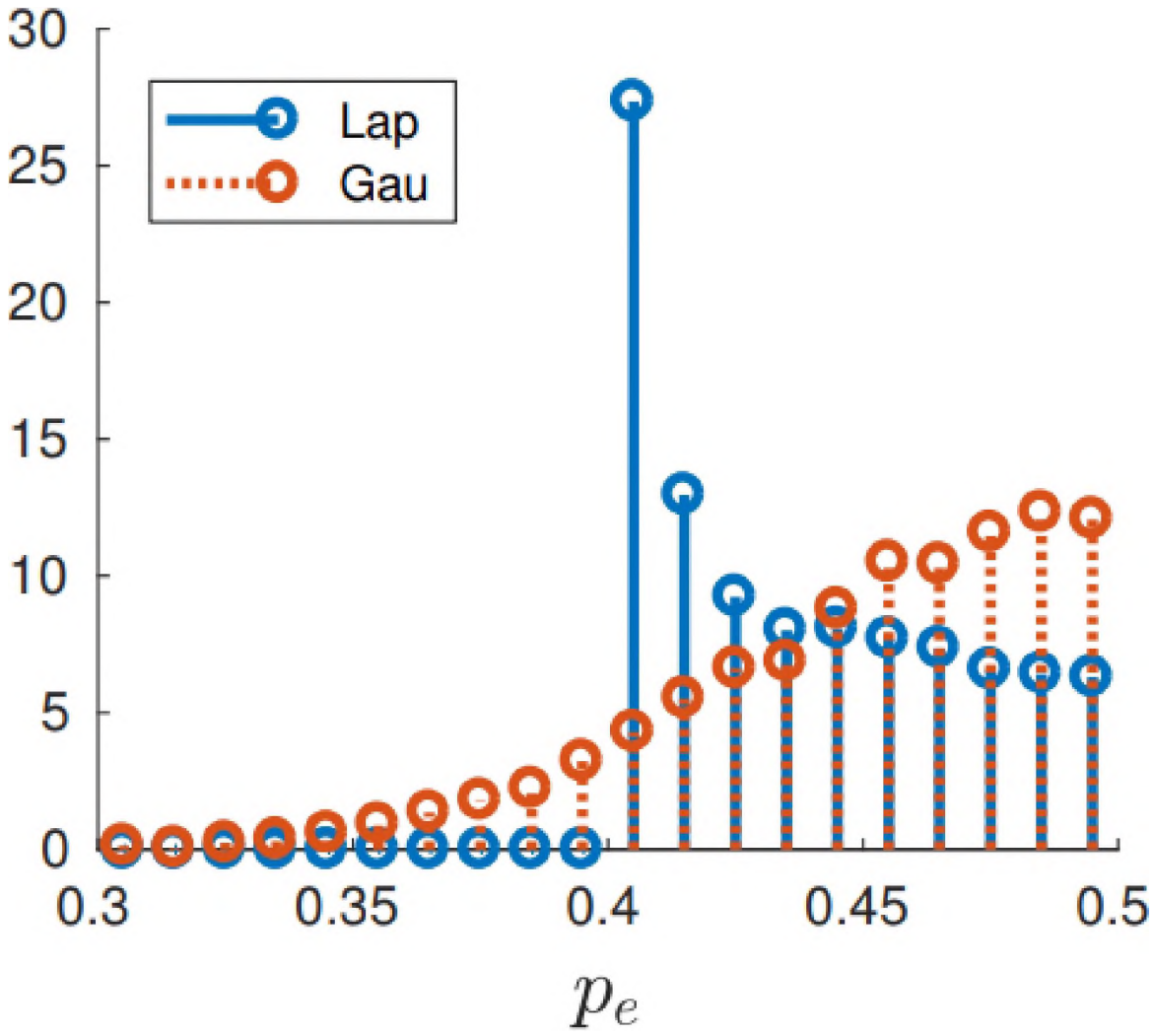
ДОДАТОК А.2


Lap, $\overline{r} = 300$
Lap, $\overline{r} = 500$
Lap, $\overline{r} = 1000$
Lap, $\overline{r} = 2000$
Gau, $\overline{r} = 300$
Gau, $\overline{r} = 500$
Gau, $\overline{r} = 1000$
Gau, $\overline{r} = 2000$
Cir, $\overline{r} = 300$
Cir, $\overline{r} = 500$
Cir, $\overline{r} = 1000$
Cir, $\overline{r} = 2000$
0.5
0.4
0.3
0.2
0.1
0
$\bar{p}_e$ (average p_e)
$p_e^* \approx 0.01$
0
500
1000
1500
$d(x, x')$ (meters)

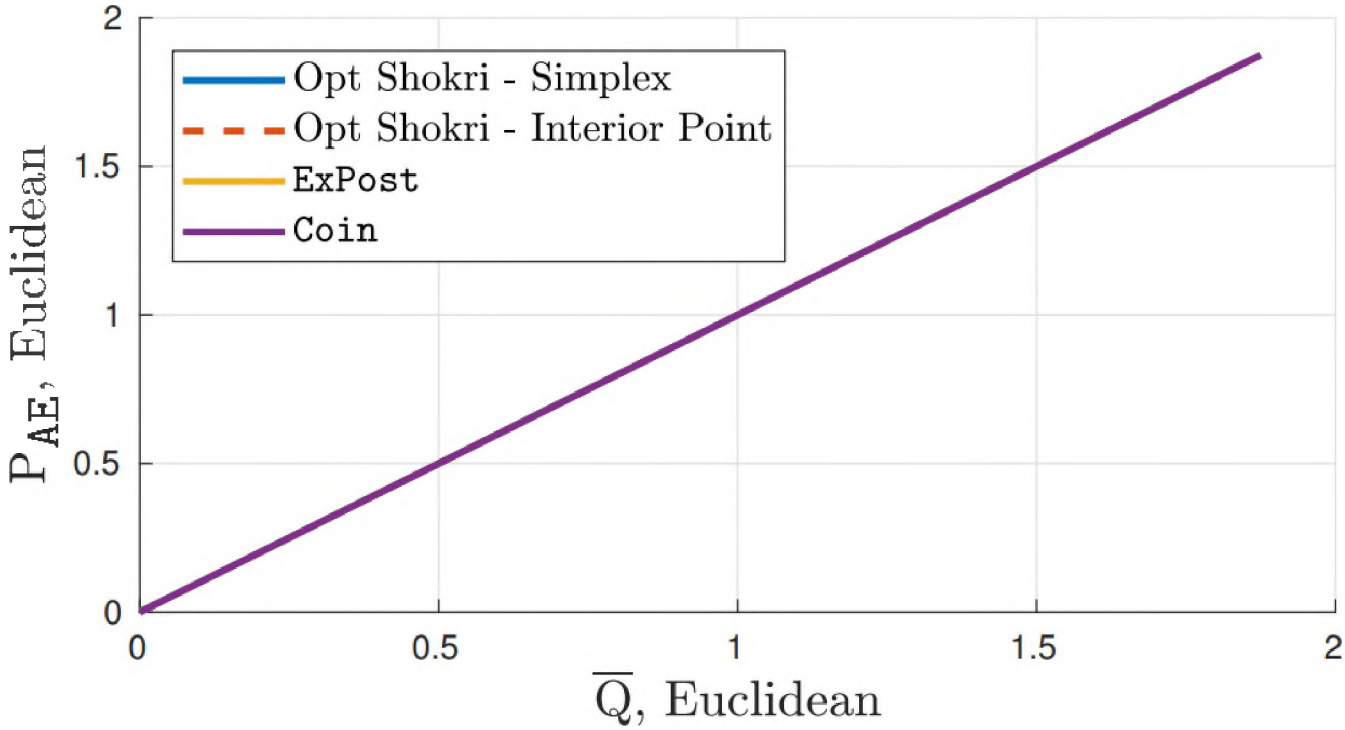
ДОДАТОК А.3


100
75
50
25
0
% better than Lap
0
500
1000
1500
$d(x, x')$ (meters)
$p_e^* \approx 0.05$
$\overline{p}_e \approx 0.16$
$p_e^* \approx 0.08$
$\overline{p}_e \approx 0.20$

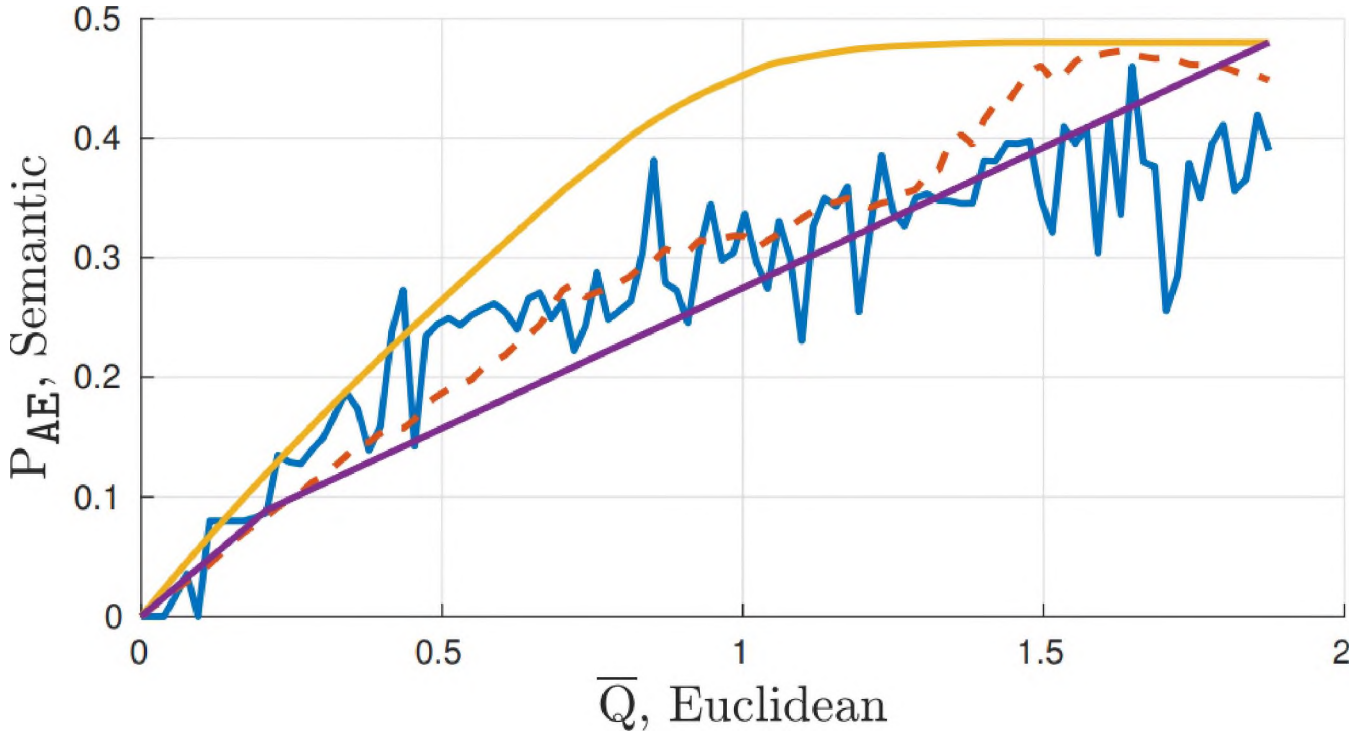
ДОДАТОК А.4



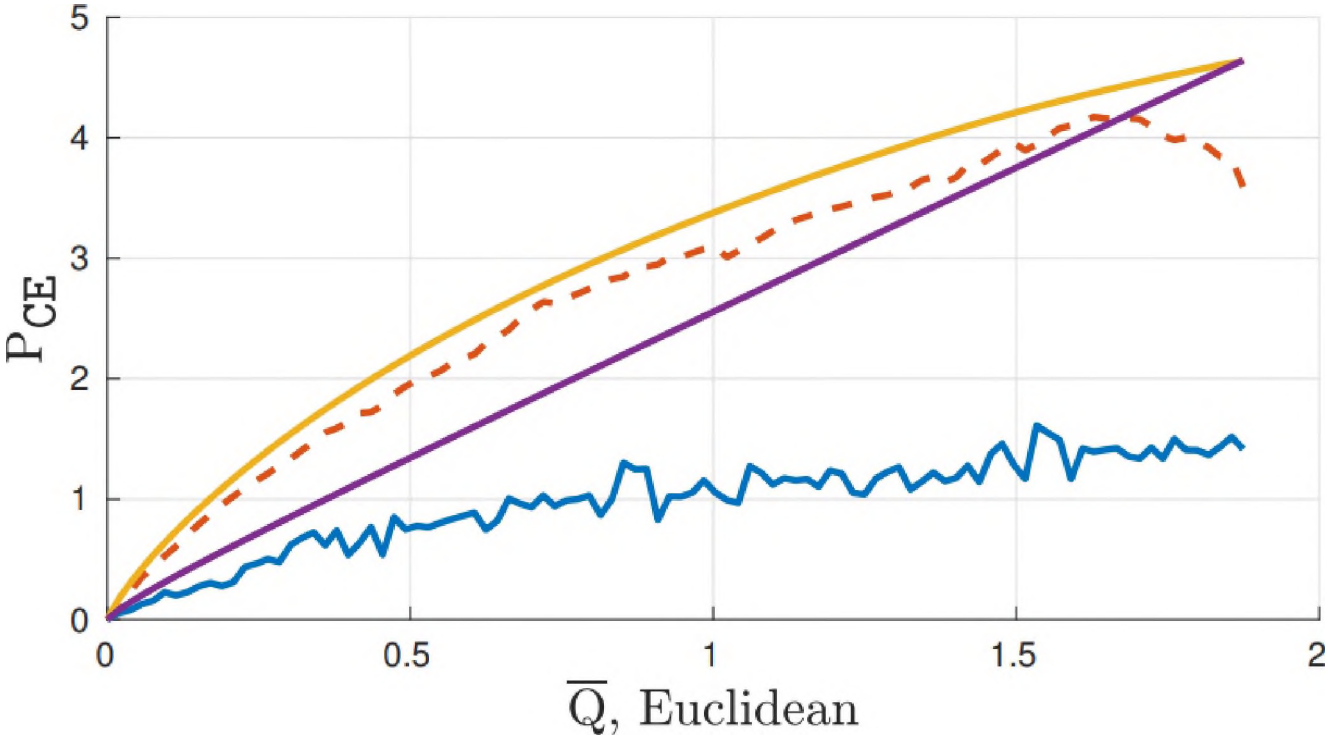
ДОДАТОК Б.1


Opt Shokri - Simplex
Opt Shokri - Interior Point
ExPost
Coin
P_{AE}, Euclidean
$\overline{Q}$, Euclidean
0
0.5
1
1.5
2

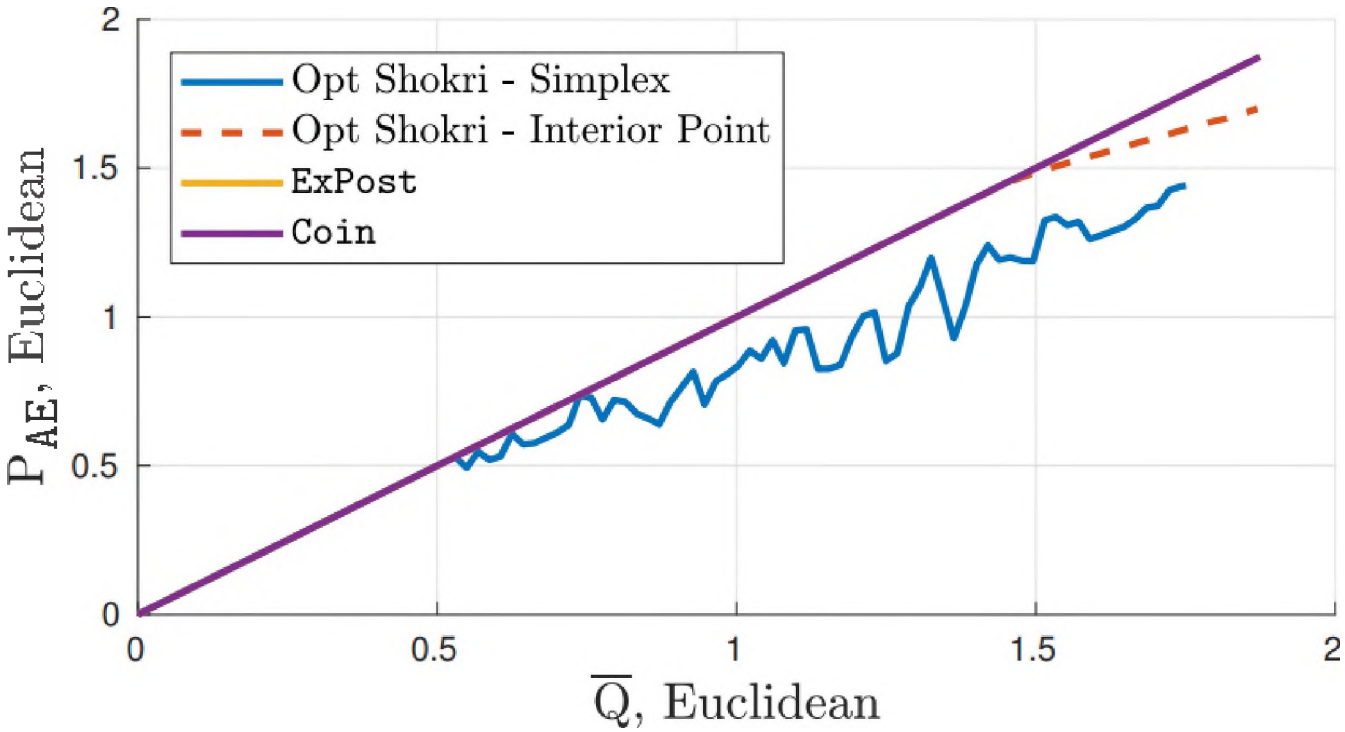
ДОДАТОК Б.2



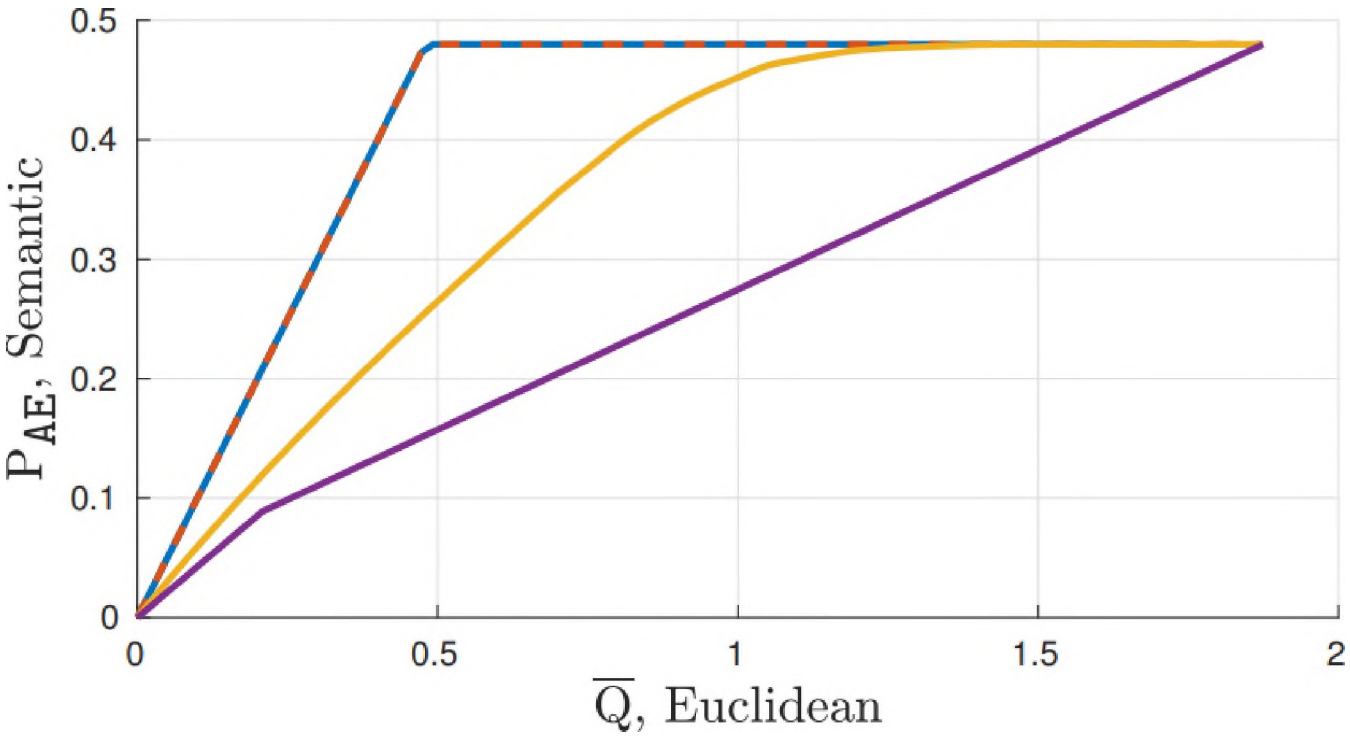
ДОДАТОК Б.3



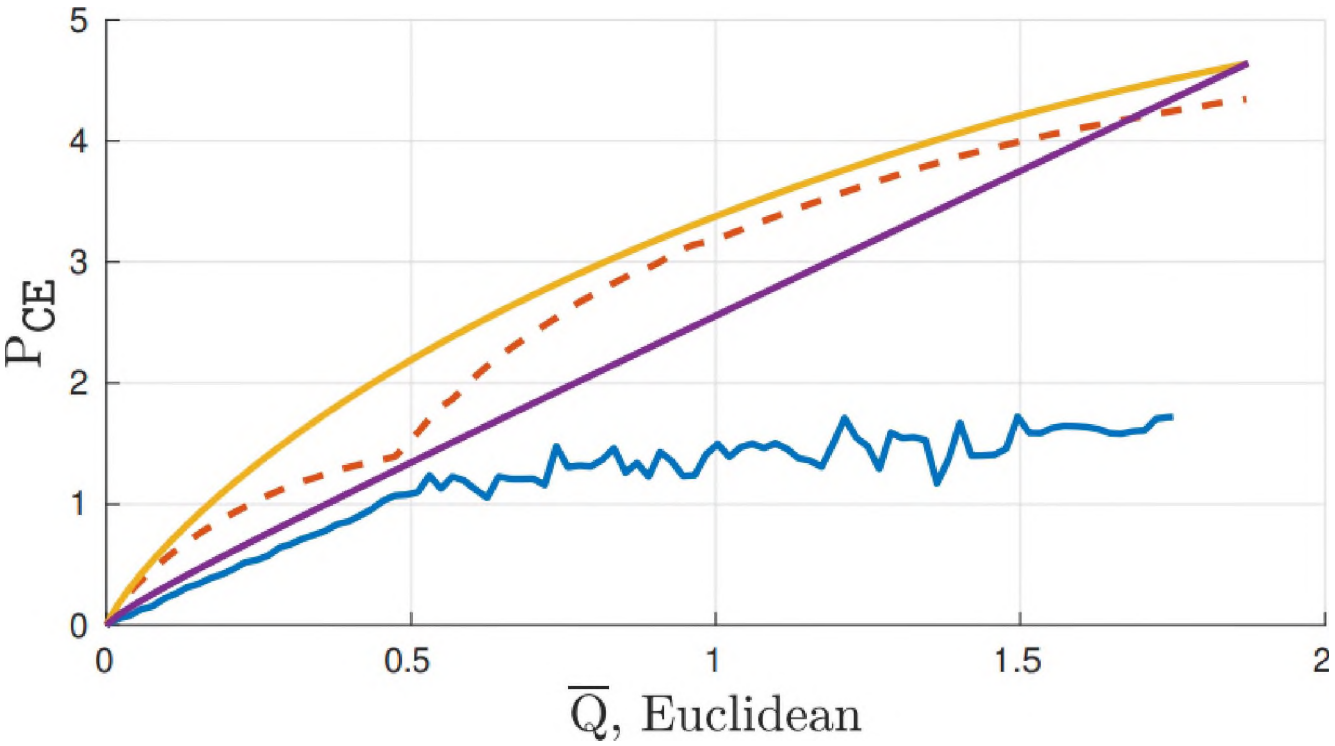
ДОДАТОК Б.4


Opt Shokri - Simplex
Opt Shokri - Interior Point
ExPost
Coin
P_{AE}, Euclidean
$\overline{Q}$, Euclidean
0
0.5
1
1.5
2

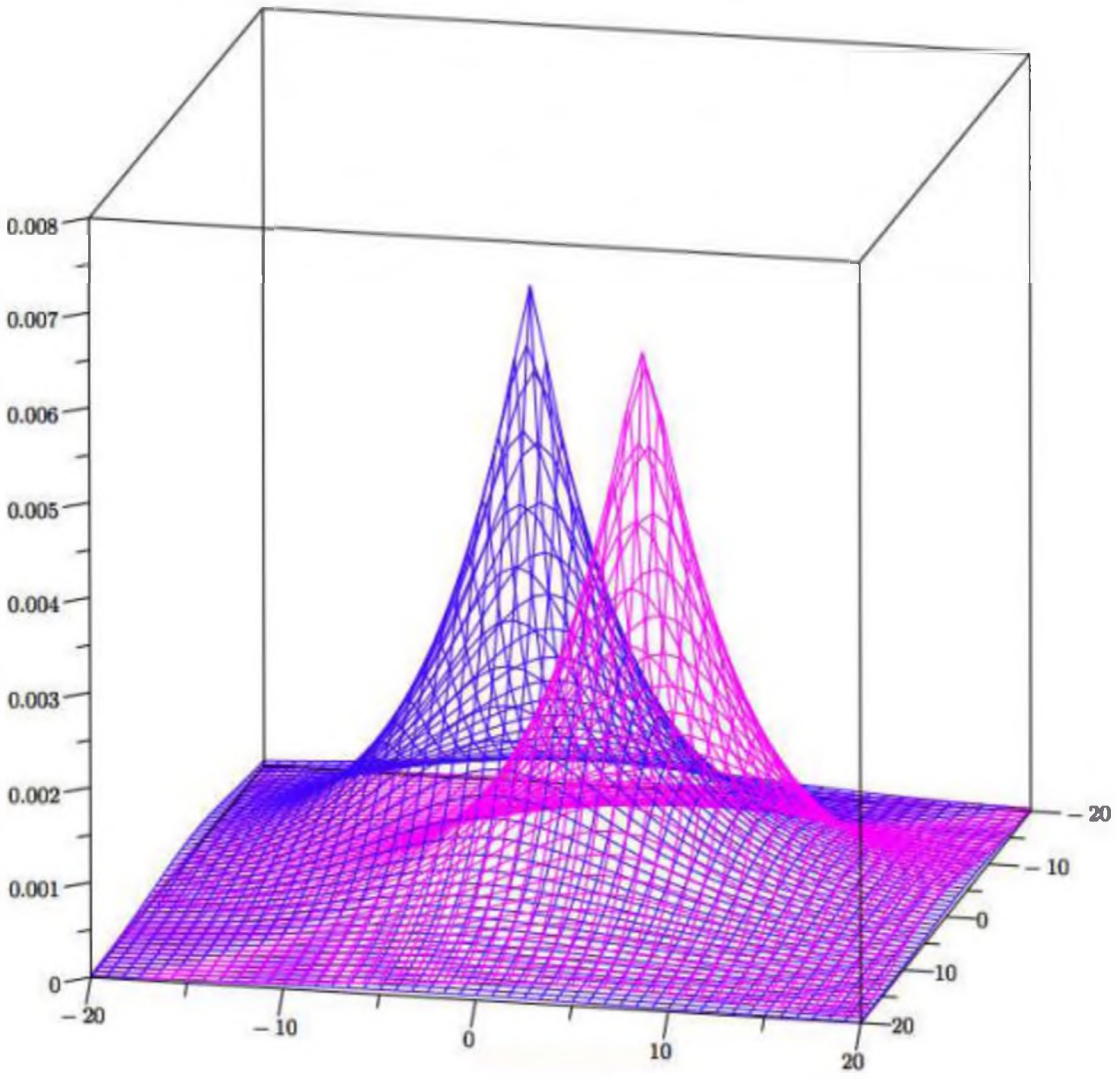
ДОДАТОК Б.5



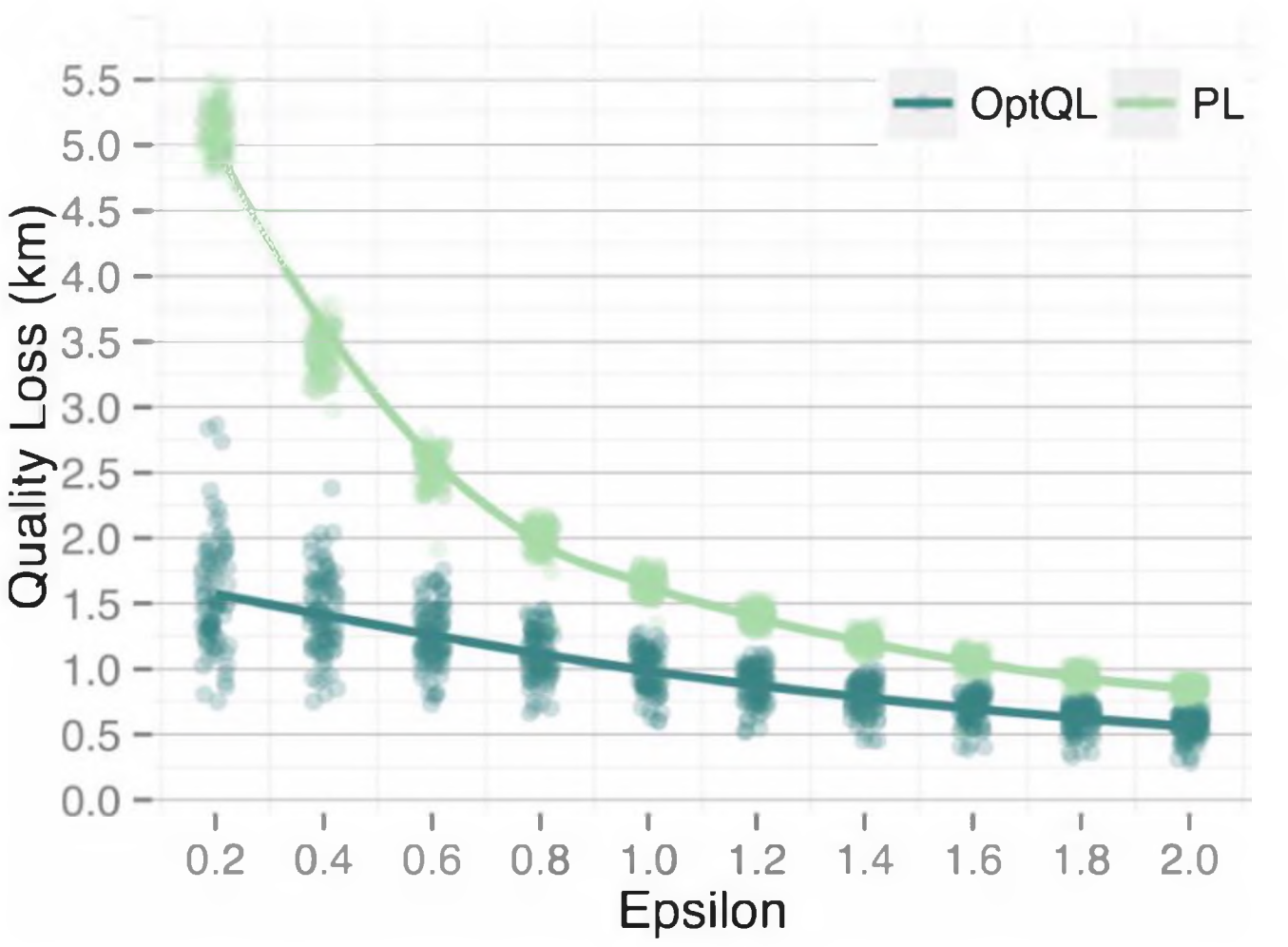
ДОДАТОК Б.6



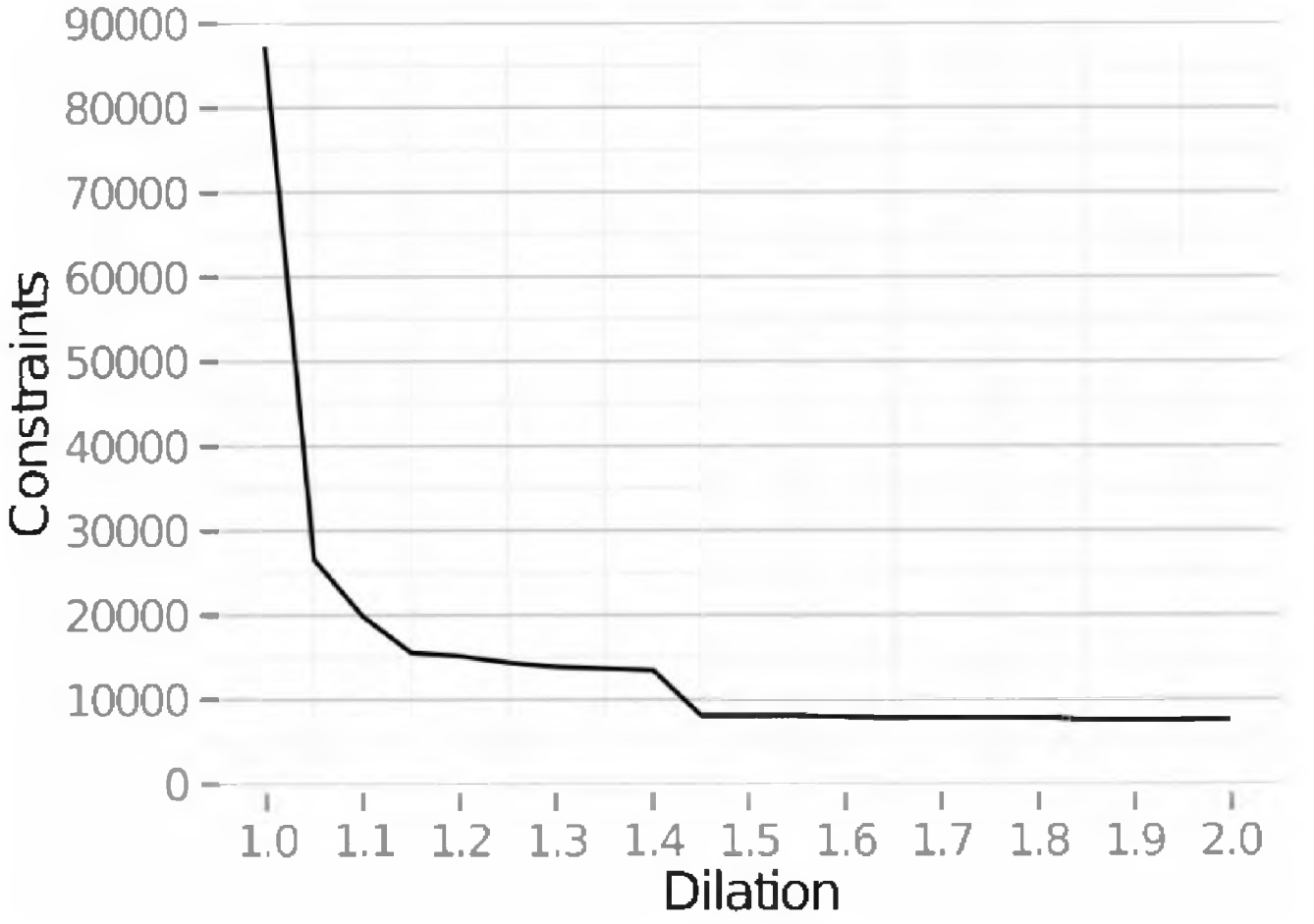
ДОДАТОК В.1


0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0
−20
−10
0
10
20
−20
−10
0
10
20

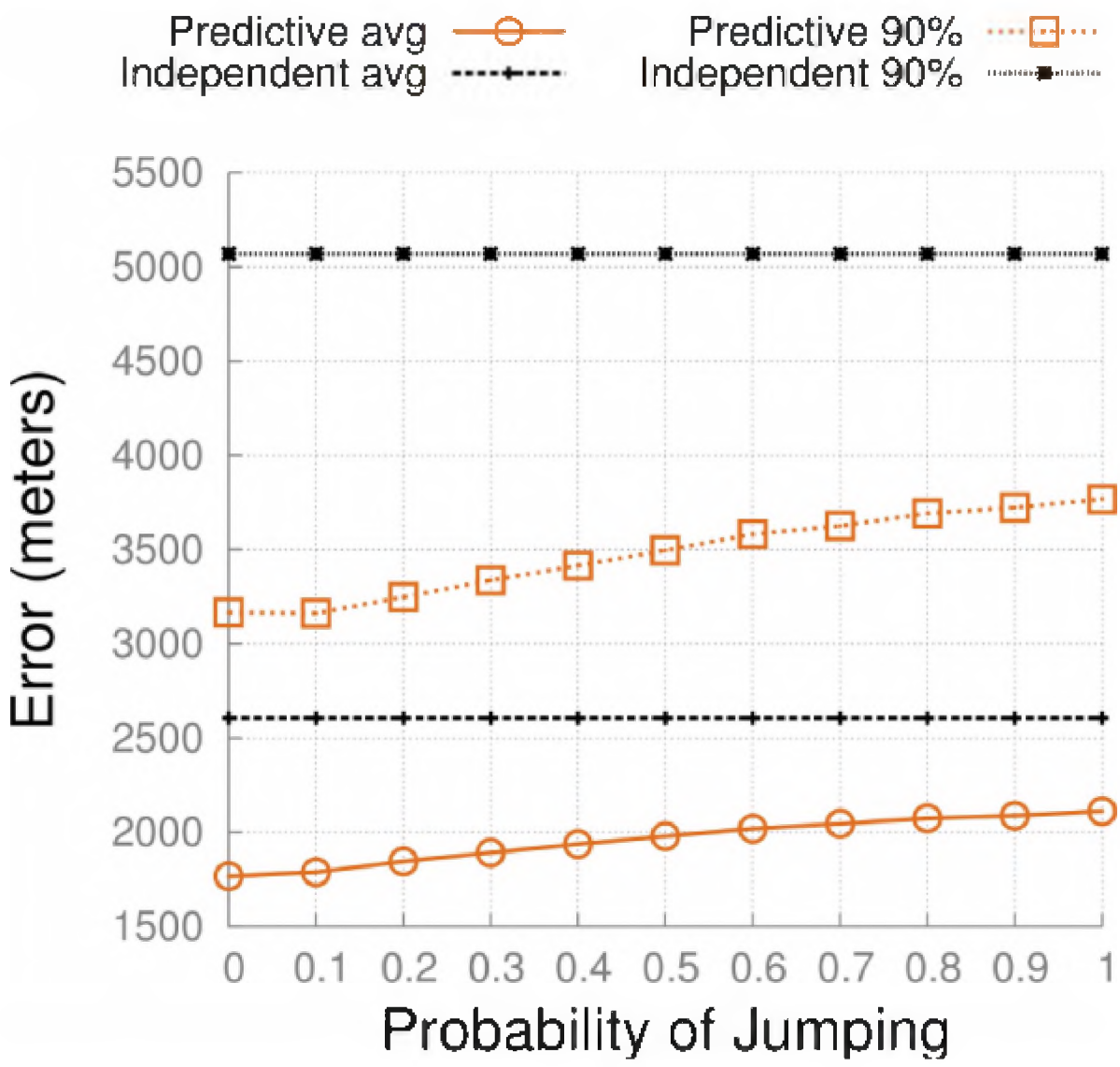
ДОДАТОК В.2


OptQL
PL
Quality Loss (km)
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0
Epsilon

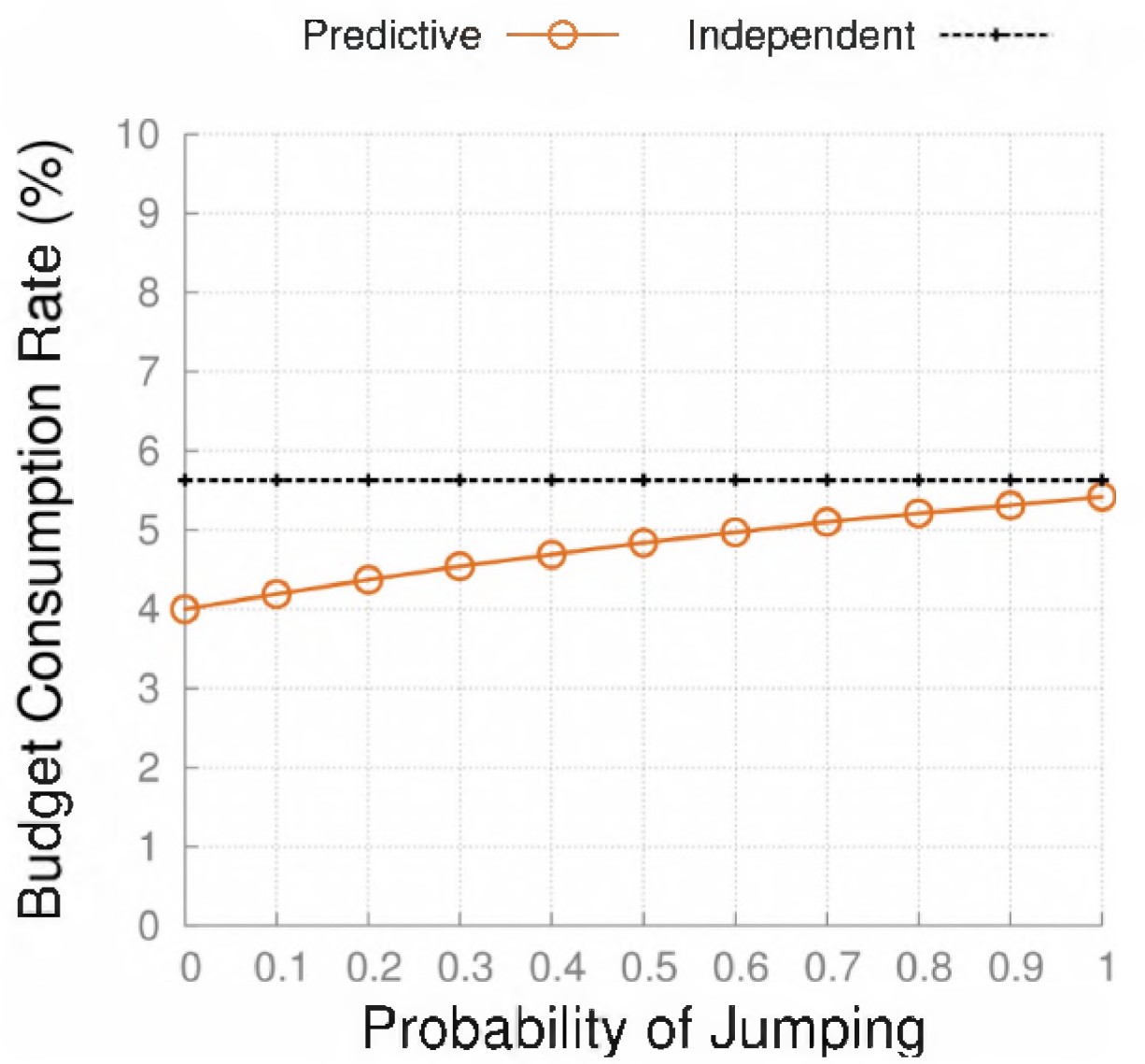
ДОДАТОК В.3


90000
80000
70000
60000
50000
40000
30000
20000
10000
0
Constraints
1.0
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
2.0
Dilation

ДОДАТОК В.4


Predictive avg
Independent avg
Predictive 90%
Independent 90%
Error (meters)
5500
5000
4500
4000
3500
3000
2500
2000
1500
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Probability of Jumping

ДОДАТОК В.5


Predictive
Independent
Budget Consumption Rate (%)
10
9
8
7
6
5
4
3
2
1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Probability of Jumping

ДОДАТОК Г

```
import ast, getopt, sys, copy, os
from fractions import Fraction

clear = lambda: os.system('cls' if os.name == 'nt' else 'clear')


class SimplexSolver():

    # Table for converting inequality list to LaTeX
    latex_ineq = {'=': '=',
                  '<=': r'$\leq$',
                  '>=': r'$\geq$'}

    def __init__(self):
        self.A = []
        self.b = []
        self.c = []
        self.tableau = []
        self.entering = []
        self.departing = []
        self.ineq = []
        self.prob = "max"
        self.gen_doc = False
        self.doc = ""

    def run_simplex(self, A, b, c, prob='max', ineq=[],
                    enable_msg=False, latex=False):
        ''' Run simplex algorithm.
        '''
        self.prob = prob
        self.gen_doc = latex
        self.ineq = ineq

        # Create the header for the latex doc.
        self.start_doc()

        # Add slack & artificial variables
        self.set_simplex_input(A, b, c)

        # Are there any negative elements on the bottom (disregarding
        # right-most element...)
        while (not self.should_terminate()):
            # ... if so, continue.
            if(enable_msg):
                clear()
                self._print_tableau()
                print("Current solution: %s\n" %
                      str(self.get_current_solution()))
                self._prompt()

            # Attempt to find a non-negative pivot.
            pivot = self.find_pivot()
            if pivot[1] < 0:
                if (enable_msg):
                    print ("There exists no non-negative pivot. "
                           "Thus, the solution is infeasible.")
                self.infeasible_doc()
                self.print_doc()
                return None
            else:
                self.pivot_doc(pivot)
                if (enable_msg):
                    clear()
```

```
                print ("There exists no non-negative pivot. "
                       "Thus, the solution is infeasible.")
            self.infeasible_doc()
            self.print_doc()
            return None
        else:
            self.pivot_doc(pivot)
            if (enable_msg):
                clear()
                self._print_tableau()
                print("\nThere are negative elements in the bottom row, "
                      "so the current solution is not optimal. "
                      "Thus, pivot to improve the current solution. The "
                      "entering variable is %s and the departing "
                      "variable is %s.\n" %
                       (str(self.entering[pivot[0]]),
                        str(self.departing[pivot[1]])))
                self._prompt()
                print("\nPerform elementary row operations until the "
                      "pivot is one and all other elements in the "
                      "entering column are zero.\n")

        # Do row operations to make every other element in column zero.
        self.pivot(pivot)
        self.tableau_doc()

    solution = self.get_current_solution()
    self.final_solution_doc(solution)
    if (enable_msg):
        clear()
        self._print_tableau()
        print("Current solution: %s\n" % str(solution))
        print("That's all folks!")
    self.print_doc()
    return solution

def set_simplex_input(self, A, b, c):
    ''' Set initial variables and create tableau.
    '''
    # Convert all entries to fractions for readability.
    for a in A:
        self.A.append([Fraction(x) for x in a])
    self.b = [Fraction(x) for x in b]
    self.c = [Fraction(x) for x in c]
    if not self.ineq:
        if self.prob == 'max':
            self.ineq = ['<='] * len(b)
        elif self.prob == 'min':
            self.ineq = ['>='] * len(b)

    self.update_enter_depart(self.get_Ab())
    self.init_problem_doc()

    # If this is a minimization problem...
    if self.prob == 'min':
        # ... find the dual maximum and solve that.
        m = self.get_Ab()
        m.append(self.c + [0])
```

```
            m = [list(t) for t in zip(*m)] # Calculates the transpose
            self.A = [x[:(len(x)-1)] for x in m]
            self.b = [y[len(y) - 1] for y in m]
            self.c = m[len(m) -1]
            self.A.pop()
            self.b.pop()
            self.c.pop()
            self.ineq = ['<='] * len(self.b)

        self.create_tableau()
        self.ineq = ['='] * len(self.b)
        self.update_enter_depart(self.tableau)
        self.slack_doc()
        self.init_tableau_doc()

    def update_enter_depart(self, matrix):
        self.entering = []
        self.departing = []
        # Create tables for entering and departing variables
        for i in range(0, len(matrix[0])):
            if i < len(self.A[0]):
                prefix = 'x' if self.prob == 'max' else 'y'
                self.entering.append("%s_%s" % (prefix, str(i + 1)))
            elif i < len(matrix[0]) - 1:
                self.entering.append("s_%s" % str(i + 1 - len(self.A[0])))
                self.departing.append("s_%s" % str(i + 1 - len(self.A[0])))
            else:
                self.entering.append("b")

    def add_slack_variables(self):
        ''' Add slack & artificial variables to matrix A to transform
            all inequalities to equalities.
        '''
        slack_vars = self._generate_identity(len(self.tableau))
        for i in range(0, len(slack_vars)):
            self.tableau[i] += slack_vars[i]
            self.tableau[i] += [self.b[i]]

    def create_tableau(self):
        ''' Create initial tableau table.
        '''
        self.tableau = copy.deepcopy(self.A)
        self.add_slack_variables()
        c = copy.deepcopy(self.c)
        for index, value in enumerate(c):
            c[index] = -value
        self.tableau.append(c + [0] * (len(self.b)+1))

    def find_pivot(self):
        ''' Find pivot index.
        '''
        enter_index = self.get_entering_var()
```

```
    depart_index = self.get_departing_var(enter_index)
    return [enter_index, depart_index]

def pivot(self, pivot_index):
    ''' Perform operations on pivot.
    '''
    j,i = pivot_index

    pivot = self.tableau[i][j]
    self.tableau[i] = [element / pivot for
                       element in self.tableau[i]]
    for index, row in enumerate(self.tableau):
       if index != i:
          row_scale = [y * self.tableau[index][j]
                       for y in self.tableau[i]]
          self.tableau[index] = [x - y for x,y in
                                 zip(self.tableau[index],
                                     row_scale)]

    self.departing[i] = self.entering[j]

def get_entering_var(self):
    ''' Get entering variable by determining the 'most negative'
        element of the bottom row.
    '''
    bottom_row = self.tableau[len(self.tableau) - 1]
    most_neg_ind = 0
    most_neg = bottom_row[most_neg_ind]
    for index, value in enumerate(bottom_row):
        if value < most_neg:
            most_neg = value
            most_neg_ind = index
    return most_neg_ind


def get_departing_var(self, entering_index):
    ''' To calculate the departing variable, get the minimum of the ratio
        of b (b_i) to the corresponding value in the entering collumn.
    '''
    skip = 0
    min_ratio_index = -1
    min_ratio = 0
    for index, x in enumerate(self.tableau):
        if x[entering_index] != 0 and x[len(x)-1]/x[entering_index] > 0:
            skip = index
            min_ratio_index = index
            min_ratio = x[len(x)-1]/x[entering_index]
            break

    if min_ratio > 0:
        for index, x in enumerate(self.tableau):
            if index > skip and x[entering_index] > 0:
                ratio = x[len(x)-1]/x[entering_index]
                if min_ratio > ratio:
```

```
                        min_ratio = ratio
                        min_ratio_index = index

        return min_ratio_index

    def get_Ab(self):
        ''' Get A matrix with b vector appended.
        '''
        matrix = copy.deepcopy(self.A)
        for i in range(0, len(matrix)):
            matrix[i] += [self.b[i]]
        return matrix

    def should_terminate(self):
        ''' Determines whether there are any negative elements
            on the bottom row
        '''
        result = True
        index = len(self.tableau) - 1
        for i, x in enumerate(self.tableau[index]):
            if x < 0 and i != len(self.tableau[index]) - 1:
                result = False
        return result

    def get_current_solution(self):
        ''' Get the current solution from tableau.
        '''
        solution = {}
        for x in self.entering:
            if x is not 'b':
                if x in self.departing:
                    solution[x] =
self.tableau[self.departing.index(x)]\

[len(self.tableau[self.departing.index(x)])-1]
                else:
                    solution[x] = 0
        solution['z'] = self.tableau[len(self.tableau) - 1]\
                        [len(self.tableau[0]) - 1]

        # If this is a minimization problem...
        if (self.prob == 'min'):
            # ... then get x_1, ..., x_n  from last element of
            # the slack columns.
            bottom_row = self.tableau[len(self.tableau) - 1]
            for v in self.entering:
                if 's' in v:
                    solution[v.replace('s', 'x')] =
bottom_row[self.entering.index(v)]

        return solution

    def start_doc(self):
        if not self.gen_doc:
            return
        self.doc = (r"\documentclass{article}"
                    r"\usepackage{amsmath}"
```

```
                r"\begin{document}"
                r"\title{Simplex Solver}"
                r"\maketitle"
                r"\begin{flushleft}"
                r"\textbf{Problem}"
                r"\end{flushleft}")

    def init_problem_doc(self):
        if not self.gen_doc:
            return
        # Objective function.
        self.doc += (r"\begin{flushleft}"
                     r"Given the following linear system and objective
"
                     r"function, find the optimal solution."
                     r"\end{flushleft}"
                     r"\begin{equation*}")
        func = ""
        found_value = False
        for index, x in enumerate(self.c):
            opp = '+'
            if x == 0:
                continue
            if x < 0:
                opp = ' - '
            elif index == 0 or not found_value:
                opp = ''
            if x == 1 or x == -1:
                x = ''
            func += (r"%s %sx_%s "  % (opp, str(x), str(index+1)))
            found_value = True
        self.doc += (r"\max{%s} \\ "
                     r"\end{equation*}" % func)
        self.linear_system_doc(self.get_Ab())
        self.doc += (r"\begin{flushleft}"
                     r"\textbf{Solution}"
                     r"\end{flushleft}")

    def linear_system_doc(self, matrix):
        if not self.gen_doc:
            return
        self.doc += (r"\["
                     r"\left\{"
                     r"\begin{array}{c}")
        for i in range(0, len(matrix)):
            found_value = False
            for index, x in enumerate(matrix[i]):
                opp = '+'
                if x == 0 and index != len(matrix[i]) - 1:
                    continue
                if x < 0:
                    opp = '-'
                elif index == 0 or not found_value:
                    opp = ''
                if index != len(matrix[i]) - 1:
```

```
                    if x == 1 or x == -1:
                        x = ''
                    self.doc += (r"%s %s%s "  % (opp, str(x),

str(self.entering[index])))
                else:
                    self.doc += (r"%s %s"  %
(self.latex_ineq[self.ineq[i]],str(x)))
                found_value = True
                if (index == len(matrix[i]) - 1):
                    self.doc += r" \\ "
        self.doc += (r"\end{array}"
                     r"\right."
                     r"\]")

    def slack_doc(self):
        if not self.gen_doc:
            return
        self.doc += (r"\begin{flushleft}"
                     r"Add slack variables to turn "
                     r"all inequalities to equalities."
                     r"\end{flushleft}")
        self.linear_system_doc(self.tableau[:len(self.tableau)-1])

    def init_tableau_doc(self):
        if not self.gen_doc:
            return
        self.doc += (r"\begin{flushleft}"
                     r"Create the initial tableau of the new linear
system."
                     r"\end{flushleft}")
        self.tableau_doc()

    def tableau_doc(self):
        if not self.gen_doc:
            return
        self.doc += r"\begin{equation*}"
        self.doc += r"\begin{bmatrix}"
        self.doc += r"\begin{array}{%s|c}" % ("c" *
(len(self.tableau[0])-1))
        for index, var in enumerate(self.entering):
            if index != len(self.entering) - 1:
                self.doc += r"%s &" % var
            else:
                self.doc += r"%s \\ \hline" % var
        for indexr, row in enumerate(self.tableau):
            for indexv, value in enumerate(row):
                if indexv != (len(row)-1):
                    self.doc += r"%s & " % (str(value))
                elif indexr != (len(self.tableau)-2):
                    self.doc += r"%s \\" % (str(value))
                else:
                    self.doc += r"%s \\ \hline_" % (str(value))
        self.doc += r"\end{array}"
        self.doc += r"\end{bmatrix}"
        self.doc += (r"\begin{array}{c}"
                     r"\\")
        for var in self.departing:
```

```
        self.doc += r"\\"
        self.doc += r"\end{array}"
        self.doc += r"\end{equation*}"

    def infeasible_doc(self):
        if not self.gen_doc:
            return
        self.doc += (r"\begin{flushleft}"
                     r"There are no non-negative candidates for the
pivot. "
                     r"Thus, the solution is infeasible."
                     r"\end{flushleft}")

    def pivot_doc(self, pivot):
        if not self.gen_doc:
            return
        self.doc += (r"\begin{flushleft}"
                     r"There are negative elements in the bottom row,
"
                     r"so the current solution is not optimal. "
                     r"Thus, pivot to improve the current solution.
The "
                     r"entering variable is $%s$ and the departing "
                     r"variable is $%s$."
                     r"\end{flushleft}" %
                     (str(self.entering[pivot[0]]),
                     str(self.departing[pivot[1]])))
        self.doc += (r"\begin{flushleft}"
                     r"Perform elementary row operations until the "
                     r"pivot element is 1 and all other elements in
the "
                     r"entering column are 0."
                     r"\end{flushleft}")

    def current_solution_doc(self, solution):
        if not self.gen_doc:
            return
        self.doc += r"\begin{equation*}"
        for key,value in sorted(solution.items()):
            self.doc += r"%s = %s" % (key,
self._fraction_to_latex(value))
            if key != 'z':
                self.doc += r", "
        self.doc += r"\end{equation*}"

    def final_solution_doc(self, solution):
        if not self.gen_doc:
            return
        self.doc += (r"\begin{flushleft}"
                     r"There are no negative elements in the bottom
row, so "
                     r"we know the solution is optimal. Thus, the
solution is: "
                     r"\end{flushleft}")
        self.current_solution_doc(solution)

    def print_doc(self):
        if not self.gen_doc:
```

```
    def _fraction_to_latex(self, fract):
        if fract.denominator == 1:
            return str(fract.numerator)
        else:
            return r"\frac{%s}{%s}" % (str(fract.numerator),
str(fract.denominator))

    def _generate_identity(self, n):
        ''' Helper function for generating a square identity matrix.
        '''
        I = []
        for i in range(0, n):
            row = []
            for j in range(0, n):
                if i == j:
                    row.append(1)
                else:
                    row.append(0)
            I.append(row)
        return I

    def _print_matrix(self, M):
        ''' Print some matrix.
        '''
        for row in M:
            print '|',
            for val in row:
                print '{:^5}'.format(str(val)),
            print '|'

    def _print_tableau(self):
        ''' Print simplex tableau.
        '''
        print ' ',
        for val in self.entering:
            print '{:^5}'.format(str(val)),
        print ' '
        for num, row in enumerate(self.tableau):
            print '|',
            for index, val in enumerate(row):
                print '{:^5}'.format(str(val)),
            if num < (len(self.tableau) -1):
                print '| %s' % self.departing[num]
            else:
                print '|'

    def _prompt(self):
        raw_input("Press enter to continue...")

if __name__ == '__main__':
    clear()

    ''' COMMAND LINE INPUT HANDLING '''
    A = []
    b = []
```

```
    c = []
    p = ''
    argv = sys.argv[1:]
    try:
        opts, args =
getopt.getopt(argv,"hA:b:c:p:",["A=","b=","c=","p="])
    except getopt.GetoptError:
        print('simplex.py -A <matrix> -b <vector> -c <vector> -p
<type>')
        sys.exit(2)
    for opt, arg in opts:
        if opt == '-h':
            print('simplex.py -A <matrix> -b <vector> -c <vector> -p
<obj_func_type>')
            print('A: Matrix that represents coefficients of
constraints.')
            print('b: Ax <= b')
            print('c: Coefficients of objective function.')
            print('p: Indicates max or min objective function.')
            sys.exit()
        elif opt in ("-A"):
            A = ast.literal_eval(arg)
        elif opt in ("-b"):
            b = ast.literal_eval(arg)
        elif opt in ("-c"):
            c = ast.literal_eval(arg)
        elif opt in ("-p"):
            p = arg.strip()
    if not A or not b or not c:
        print('Must provide arguments for A, b, c (use -h for more
info)')
        sys.exit()
    ''' END OF COMMAND LINE INPUT HANDLING '''

    # Assume maximization problem as default.
    if p not in ('max', 'min'):
        p = 'max'
```