УДК 004.056 DOI: 10.31673/2412-4338.2023.035158

Кир'янов А.Ю.
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

ТРІАНГУЛЯЦІЯ МІСЦЕЗНАХОДЖЕННЯ БПЛА ЗА ДОПОМОГОЮ БАЗОВИХ СТАНЦІЙ

У статті було проведено звітний аналіз методів тріангуляції в радіолокації, зокрема в контексті стільникових мереж GSM. Однією з центральних проблем, виявлених у дослідженні, була змінена структура даних, яка використовується для представлення тріангуляції.Основна мотивація для цього вивчення полягає в тому, що різні методи тріангуляції можуть вимагати різні структури даних. Наприклад, для оптимальної тріангуляції алгоритми побудови потрібні лише ребра та вузли, які повинні мати структуру даних, де вузли мають сусідів або зв'язок з ребрами. Розглянуто питання завдання зміни структури, в якій представлена тріангуляція, може виникнути, наприклад, при побудові оптимальної тріангуляції. Алгоритми їх побудови оперують тільки з ребрами та вузлами, а тому вони змушені використовувати структури даних типу «Вузли з сусідами» або «Вузли та ребра». З іншого боку, метою побудови тріангуляції може бути моделювання поверхні, для чого необхідна структура даних, наприклад, «Вузли та трикутники». Саме тому і постає завдання переходу від однієї структури даних до іншої. З іншого боку, якщо призначено моделювання поверхонь для додатків, де потрібно знати трикутники та їх параметри, необхідно використовувати структуру даних, де вузли мають взаємозв'язки з трикутниками. Завданням стало перетворення одного виду структури даних на інший у зв'язку з конкретними вимогами алгоритму та його застосування. Це завдання вимагає глибокого розуміння як алгоритмів побудови тріангуляції, так і особливостей структури даних.У результатах порівняння були переваги та обмеження різних підходів до структури даних, а також показано, що перехід між ними може вплинути на продуктивність та точність алгоритмів роботи. Деякі структури можуть бути більш ефективними для одних видів завдань, а інші – для інших. Крім того, дослідження визначило перспективи подальшого розвитку цієї області. Однією з регуляцій є можливість пошуку оптимальних структурних даних, які б дозволили ефективно вирішувати різні види завдань тріангуляції, а також розробка методів автоматизованого переходу між ними на основі вимог конкретних завдань.

Kyrianov A. Yu.
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"

UAV LOCATION TRIANGULATION USING BASE STATIONS

The article conducts a reporting analysis of triangulation methods in radar, in particular in the context of GSM cellular networks. One of the central problems identified in the study was the change in the data structure used to represent triangulation. For example, for optimal triangulation, construction algorithms only need edges and nodes, which must have a data structure where nodes have neighbors or connections to edges. The question of the problem of changing the structure in which triangulation is presented may arise, for example, when constructing optimal triangulation, is considered. Their construction algorithms operate only with edges and nodes, and therefore they are forced to use data structures such as "Nodes with neighbors" or "Nodes and edges". On the other hand, the purpose of building triangulation can be to model a surface, for which a data structure is needed, such as "Nodes and Triangles". That is

why the task of moving from one data structure to another arises. On the other hand, if surface modeling is intended for applications where you need to know triangles and their parameters, you must use a data structure where nodes have relationships with triangles. The task was to transform one type of data structure into another in connection with the specific requirements of the algorithm and its application. The results of the comparison included the advantages and limitations of different approaches to the data structure, and it was also shown that the transition between them can affect the performance and accuracy of work algorithms. Some structures may be more effective for some kinds of tasks, while others may be more effective for others. In addition, the study identified the prospects for further development of this area. One of the regulations is the ability to search for optimal structural data that would effectively solve different types of triangulation problems, as well as the development of methods for automated transition between them based on the requirements of specific tasks.

Key words: *fault tolerance, excess code, Latin square*

Актуальність.

В статті розглядаються зростаючі проблеми безпеки, пов’язані з дронами (БПЛА) поблизу базових станцій GSM. Методи тріангуляції можуть виявляти загрози БПЛА та реагувати на них, наголошуючи на інтеграції цієї технології з GPRS для передачі даних у реальному часі. Стаття сприяє підвищенню безпеки та стійкості мереж GSM, одночасно висвітлюючи потенційні області для майбутніх досліджень.

Структури для представлення тріангуляції. Як показує практика, вибір структури для представлення тріангуляції робить істотний вплив на теоретичну складність алгоритмів, а також на швидкість конкретної реалізації. Крім того, вибір структури може залежати від мети подальшого використання тріангуляції. У тріангуляції можна виділити 3 основних типи об'єктів: вузли (точки, вершини), ребра (відрізки) і трикутники.

У деяких алгоритмах деякі з цих операцій можуть не використовуватися. В інших алгоритмах операції з ребрами можуть відбуватися нечасто, тому ребра можна представити опосередковано у вигляді однієї зі сторін деякого трикутника. Розглянемо найпоширеніші конструкції [1].

Мета дослідження. Метою цього дослідження є дослідити й оцінити здійсненність і точність методів тріангуляції місцезнаходження БПЛА за допомогою базових станцій. Мета полягає в тому, щоб зрозуміти методи, алгоритми та технології, задіяні у визначенні точного географічного розташування БПЛА на основі сигналів, отриманих від кількох найближчих базових станцій.Вирішуючи ці завдання, дослідження має на меті надати цінну інформацію про можливості та обмеження тріангуляції розташування БПЛА за допомогою базових станцій, що може мати наслідки для широкого спектру застосувань і галузей.

Структура даних "Вузли з сусідами". У структурі "Вузли з сусідами" для кожного вузла тріангуляції зберігаються його координати на площині і список чисел (або покажчиків) сусідніх (сусідів, з якими є спільні ребра) вузлів.

По суті, список сусідів неявно визначає краї тріангуляції. Трикутники не представлені зовсім, що зазвичай є істотною перешкодою для подальшого використання тріангуляції. Крім того, недоліком є змінний розмір структури вузла, що часто призводить до марнотратного споживання пам'яті при побудові тріангуляції. Середня кількість сусідніх вузлів в тріангуляції Делоне дорівнює 6 (це доведено індукцією або з теореми планарного графа Ейлера), тому при 8-байтовому представленні координат, 4-байтових цілих числах і 4-байтових покажчиках загальний обсяг пам'яті, зайнятий цією тріангуляційною структурою, становить 44 * N байт. [2].

Структура даних вузлів та країв. У структурі «Вузли і ребра» явно вказані вузли і ребра. У структурі відсутні трикутники. Для кожного краю зберігаються покажчики на два

кінцеві вузли. Для трикутників зберігаються покажчики на три ребра, що утворюють трикутник (малюнок 2):

Ця структура часто використовується в тих випадках, коли потрібно явно представити ребра тріангуляції, але при цьому немає необхідності працювати з трикутниками. Зокрема, ця структура найкраще підходить для побудови жадібних і оптимальних тріангуляцій. Ця структура споживає зовсім небагато пам'яті: при 8-байтовому поданні координат і 4-байтових покажчиків налічується близько 40 * N байт

Структура даних "Подвійні ребра".У структурі "Подвійні ребра" основою тріангуляції є список спрямованих ребер. При цьому кожне ребро входить в тріангуляційну структуру двічі, але спрямоване в протилежні сторони:

Для кожного ребра зберігаються наступні покажчики (рисунок 1):

1) на торцевому вузлі ребра;

2) до наступного ребра за годинниковою стрілкою в трикутнику праворуч від цього ребра;

3) до «подвійного краю», з'єднавши ті ж вузли тріангуляції, що і даний, але спрямовані в зворотному напрямку;

4) до трикутника, розташованого праворуч від ребра.

Останній покажчик не потрібен для побудови тріангуляції, а тому його наявність слід визначати в залежності від мети подальшого застосування тріангуляції[4].

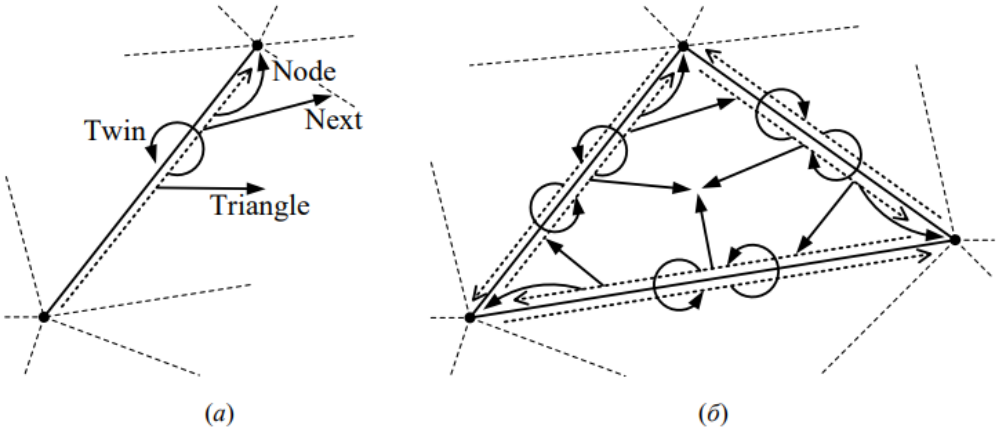

Рис. 1. Крайові з'єднання (a) та неявне визначення трикутників (b) у структурі "Подвійні ребра"

Недоліками такої структури є подання трикутників в неявному вигляді, а також велика витрата пам'яті, який при 8-байтовому поданні координат і 4-байтових покажчиків становить не менше 64 * N байт (не враховуючи витрата пам'яті на представлення додаткових даних в трикутниках) [5].

Структура даних вузлів і трикутників. У структурі «Вузли і трикутники» для кожного трикутника зберігаються три покажчика на утворюють його вузли і три покажчика на сусідні трикутники.

Точки і сусідні трикутники нумеруються за годинниковою стрілкою, в той час як навпроти точки з номером i{1, 2, 3} знаходиться ребро, відповідне сусідньому трикутнику з таким же номером. Краї в цій тріангуляції явно не зберігаються. При необхідності їх зазвичай представляють у вигляді покажчика на трикутник і номер ребра всередині нього. З 8-байтовим поданням координат і 4-байтовими покажчиками ця структура тріангуляції вимагає приблизно 64 * N байт. Незважаючи на те що дана структура поступається Nodes with Neighbors, вона найчастіше використовується на практиці завдяки відносній простоті і легкості програмування алгоритмів на її основі.∈

Структура даних вузлів, ребер і трикутників. У структурі "Вузли, ребра і трикутники" явно вказані всі об'єкти тріангуляції: вузли, ребра і трикутники. Для кожного

ребра зберігаються покажчики на два торцеві вузли і два сусідніх трикутника. Для трикутників зберігаються покажчики на три ребра, що утворюють трикутник (рисунок 2):

Точки і сусідні трикутники нумеруються за годинниковою стрілкою, при цьому навпроти точки з числом i{1, 2, 3} знаходиться ребро, відповідне сусідньому трикутнику з таким же номером (рисунок 2). Ця структура часто використовується на практиці, особливо в задачах, де потрібно явно представити краї тріангуляції.∈

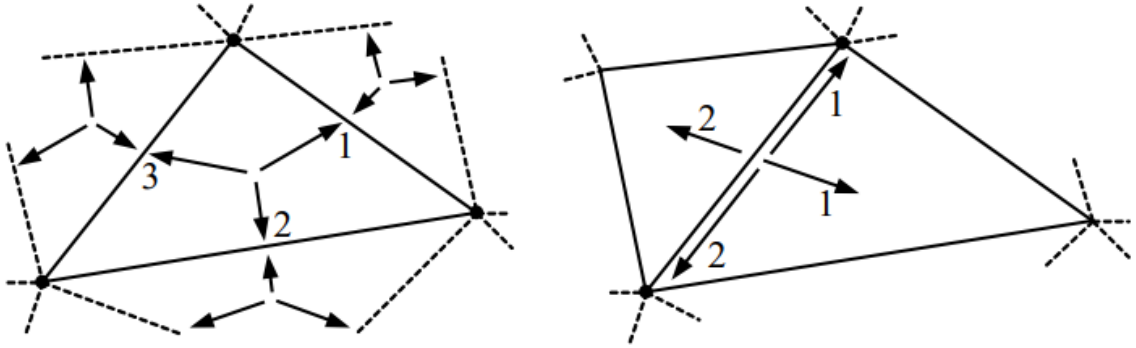

Рис. 2. З'єднання трикутників (ліворуч) і ребер (праворуч) структури «Вузли, ребра і трикутники»

Недоліком такої структури є велика витрата пам'яті, що становить приблизно 88 * N байт з 8-байтовим поданням координат і 4-байтовими покажчиками[6].

Вузли, прості ребра і трикутники структури даних. У структурі "Вузли, прості ребра та трикутники" чітко вказані всі об'єкти тріангуляції: вузли, ребра та трикутники. Для кожного ребра зберігаються покажчики на два торцеві вузли і два сусідніх трикутника. Особливої інформації для країв немає. Для трикутників зберігаються покажчики на три вузли і три ребра, що утворюють трикутник, а також покажчики на три сусідніх трикутника.

Ця структура часто використовується на практиці, особливо в задачах, де потрібно явно представити краї тріангуляції.

Недоліком такої структури є відносно велика витрата пам'яті, що становить близько 88 * N байт з 8-байтовим поданням координат і 4-байтовими покажчиками. На завершення цього розділу таблиця 1 підсумовує характеристики даних даних структур, включаючи вартість пам'яті і ступінь представлення різних елементів тріангуляції ("–" - елемент відсутній, "+" - присутній, "±" - присутній, але немає посилань на інші елементи тріангуляції). В цілому можна відзначити, що структура «Вузли з сусідами» менш зручна, ніж інші, так як явно не представляє ребер і трикутників. Серед інших структура вузлів і трикутників досить зручна в програмуванні. Однак деякі алгоритми тріангуляції вимагають явного представлення ребер, тому там можна рекомендувати структуру вузлів, ребер і трикутників.

Таблиця 1.

Основні характеристики структур даних

Ім'я структури даних	Пам'ять	Вузлів	Ребра	Трикутники
«Вузли з сусідами»	44*N	+	-	-
«Вузли і краї»	40*N	±	+	-
«Подвійні ребра»	64*N	±	+	±
«Вузли і трикутники»	64*N	±	-	+
"Вузли, ребра і трикутники"	88*N	±	+	+
"Вузли, прості ребра і трикутники"	88*N	±	±	+

Перетворення структур даних. Проблема зміни структури, в якій представлена тріангуляція, може виникнути, наприклад, при побудові жадібної або оптимальної тріангуляції. Алгоритми їх побудови оперують тільки ребрами і вузлами, а тому змушені використовувати структури даних типу «Вузли з сусідами» або «Вузли і ребра». З іншого боку, метою побудови тріангуляції може бути поверхневе моделювання, яке вимагає структури даних, такої як «Вузли і трикутники». Саме тому виникає проблема переходу від однієї структури даних до іншої. Для початку розглянемо досить простий алгоритм переходу від структури «Вузли і ребра» до структури «Вузли з сусідами». Основною метою даного алгоритму є обчислення ребер, прилеглих до вузлів. Алгоритм перетворення структури даних "Вузли і ребра" в структуру "Вузли з сусідами" Структури даних. Початкові структури представлені масивами вузлів і ребер. В результаті ми повинні отримати масив NewNodes. Алгоритм зажадає тимчасового масиву R довжини N для підрахунку кількості ребер, прилеглих до відповідних вузлів[7].

Крок 1. Обчисліть кількість сусідніх ребер R[i], включених в кожен і-й вузол тріангуляції. Для цього спочатку присвоюємо і: R[i]:=0. Потім, в циклі над і, ми переглядаємо всі ребра і для кожного ребра Ribs[i] з'єднуємо вузли з числами a=Ribs[i]. Вузли[1] та b=Ребра[i]. Вузли[2], збільшуємо лічильники ребер у вузлах: R[a]++, R[b]++.∀

Крок 2. Розподіліть пам'ять для кожного вузла вузлів зі структурою Neighbors, використовуючи R[i] як кількість вузлів у масиві вузлів вузлів. В результаті отримуємо нові записи NewNodes[i]. У NewNodes[i] ми копіюємо поля з координатами X,Y з відповідних полів вузлів[i] вихіднихструктур даних "Вузли та ребра". Поки що встановіть нуль інших полів: NewNodes[i]. Лічильник:=0, j: Нові вузли[i]. Вузли[j]:=0.∀

Крок 3. У циклі над і, ми переглядаємо всі ребра і для кожного ребра R[i] з'єднуючи вузли з числами a і b, додаємо до вузлів ланки до вузла, що примикає через це ребро і збільшуємо лічильники сусідніх вузлів в нових вузлах:

Нові вузли[a]. Вузли[NewNodes[a]. Count]:=b; Нові вузли[a]. Лічба++;

Нові вузли[b]. Вузли[Нові вузли[b]. Count]:=b; Нові вузли[b]. Лічильник++.

Кінець алгоритму.

Складність описаного алгоритму лінійна за кількістю вузлів тріангуляції. Наступний алгоритм переходу від структури «Вузли і ребра» до структури «Вузли і трикутники» більш складний. Вона вимагає створення взаємопов'язаних конструкцій трикутника. Перші 3 кроки цього алгоритму практично збігаються з попереднім алгоритмом: в ньому створюється спеціальна тимчасова структура даних для кожного вузла. Алгоритм перетворення структури даних "Nodes and Edges" в структуру "Nodes and Triangles". Структури даних. Початкові структури представлені масивами вузлів і ребер. В процесі роботи алгоритму потрібен тимчасовий масив R довжини N для підрахунку кількості ребер, прилеглих до вузлів. Крім того, нам знадобиться тимчасова змінена структура даних "Вузли з сусідами" (розширена списком сусідніх ребер і трикутників, але без координат, так як ми можемо отримати їх з вихідного масиву Nodes), яка буде представлена в масиві TmpNodes.

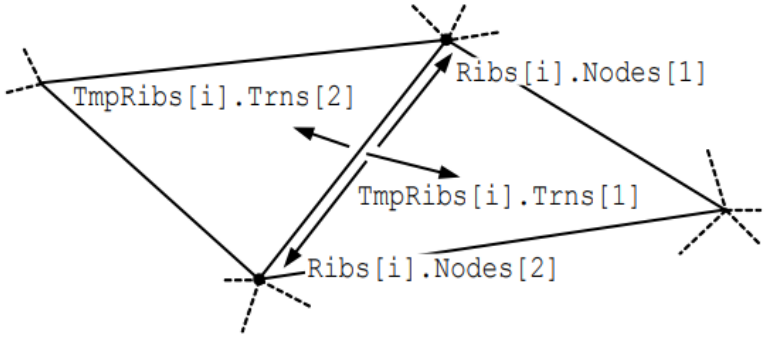

Рис. 3. Тимчасові зв'язки ребер в алгоритмі перетворення структури дані "Вузли і ребра" в структуру "Вузли і трикутники"[15].

В результаті роботи алгоритму ми повинні отримати масив NewNodes.

Крок 1. Обчисліть кількість сусідніх ребер R[i], включених в кожен і-й вузол тріангуляції. Для цього спочатку присвоюємо i: R[i]:=0. Потім, в циклі над і, ми переглядаємо всі ребра і для кожного ребра Ribs[i] з'єднуємо вузли з числами a=Ribs[i]. Вузли[1] та b=Ребра[i]. Вузли[2], збільшуємо лічильники ребер у вузлах: R[a]++, R[b]++.∀

Крок 2. Створіть тимчасові записи TmpNodes[i] для кожного вузла, використовуючи R[i] як довжини масивів Nodes, Ribs та Trns. Інші поля поки що заповнені нулями та порожніми посиланнями: TmpNodes[i]. Лічильник:=0, j: TmpNodes[i]. Вузли[j]:=0, j: TmpNodes[i]. Ribs[j]:=0, j : TmpNodes[i]. Trns[j]:=0.∀∀∀

Крок 3. Дивимося через всі ребра і для кожного ребра R з'єднуємо вузли з номерами a і b, додаємо до вузлів ланки R і вузол, що примикає через це ребро R, і збільшуємо лічильники сусідніх вузлів в нових вузлах:

TmpNodes[a].Nodes[TmpNodes[a].Count]:=b;

TmpNodes[a].Ribs[TmpNodes[a].Count]:=R; TmpNodes[a].Count++;

TmpNodes[b].Nodes[TmpNodes[b].Count]:=b;

TmpNodes[b].Ribs[TmpNodes[b].Count]:=R; TmpNodes[b].Count++.

Крок 4. На кожному вузлі тимчасової структури TmpNodes, сортуйте сусідні вузли та ребра за годинниковою стрілкою (одночасно в TmpNodes[i]. Вузли та TmpNodes[i]. Ребра масивів). Для всіх ребер тріангуляції масиву TmpRibs[i]. Trns заповнюються порожніми посиланнями (нульові числа трикутників): i,j: TmpRibs[i]. Trns[j]:=0.∀

Крок 5. У циклі над і для кожного вузла робимо вкладений цикл над j над усіма сусідніми ребрами. Для кожного ребра визначте номер k вузла в ребрі і встановіть TmpRibs[TmpNodes[i]. Ребра[j]]. IndexInNode[k]:=j.

Крок 6. У циклі і для кожного ребра ми робимо вкладений цикл j через два трикутники, прилеглі до ребра, і намагаємося створити новий трикутник з TmpRibs[i]. Trns[j], якщо цей трикутник ще не створений (якщо TmpRibs[i] . Trns[j]=0). Цей трикутник з'єднає кінцеві вузли поточного ребра (Ribs[j]. Вузли[j], ребра[j]. Nodes[3-j]) та інший вузол, який можна визначити, використовуючи список суміжних вузлів у вузлі Ribs[j]. Вузли[j] за допомогою TmpRibs[i]. IndexInNode[j]. Крім того, потрібно визначити 3 ребра, які утворюють трикутник і виставляти зв'язки з цих ребер на новий трикутник.

Крок 7. Встановлюємо взаємні зв'язки трикутників між собою. Для цього робимо цикл уздовж і уздовж кожного внутрішнього краю тріангуляції (по всіх і-х ребрах з j: TmpRibs[i]. Trns[j]≠0) і для нього встановлюємо взаємні зв'язки сусідніх трикутників TmpRibs[i] . Trns[0] і TmpRibs[i]. Трн[1].∀ [8].

Кінець алгоритму. За умови, що на кроці 4 використовується сортування з лінійною складністю (наприклад, цифрова), то складність цього алгоритму лінійна по відношенню до загальної кількості вузлів в тріангуляції. В цілому, навіть якщо використовувати нелінійну сортування, складність алгоритму в середньому також буде лінійною - O (N). Це випливає з того, що середня кількість ребер, прилеглих до вузла в тріангуляції, є константою,

незалежною від N, і, отже, сортування буде виконувати O(1) час в середньому. Висновок: Вищевказаний алгоритм можна легко модифікувати для отримання інших структур даних, в яких явно представлені трикутники.

Тріангуляція місцезнаходження БПЛА за базовими станціями.

Існує поширена помилка, що географічне розташування будь-якого БПЛА GSM можна визначити з достатньою точністю шляхом тріангуляції на трьох базових станціях. Зазвичай це описується наступним чином: наприклад, якщо є можливість визначити відстань від базової станції до БПЛА стандартними засобами, то за відстанями від трьох базових станцій можна отримати точні координати пристрою, а за відстанню від двох базових станцій - дві точки, в одній з яких буде розташовуватися потрібний БПЛА. Як правило, народна поголоска наділяє кримінальні елементи або правоохоронні органи можливістю використовувати таку технологію для пошуку потрібних їм людей. Частково це твердження вірно. Стандартними засобами іноді можна визначити відстань від БПЛА до однієї базової станції. Цим, можливо, пояснюється стійкість віри в те, що тріангуляція можлива. Насправді це не так. Перш ніж приступити до детального розбору випадку тріангуляції, варто зробити істотне застереження. Необхідно провести чітку межу між помилкою, що базові станції будь-якої мережі GSM завжди тріангулюють місцезнаходження зазначеного БПЛА і можливістю визначення місцезнаходження іншими способами в рамках єдиної мережі [9].

Послуги на основі місцезнаходження. Для надання послуг на основі визначення місцезнаходження (LBS) існує безліч способів, заснованих на наявності додаткового програмного і апаратного забезпечення на всіх базових станціях конкретної мережі. Приклад таких сервісів: показати абоненту карту міста і його місця на ній, повідомити адресу найближчого ресторану або магазину, повідомити місцезнаходження іншого абонента, прокласти маршрут до заданої точки. Для ряду додатків досить приблизно знати одну базову станцію, в зоні покриття якої знаходиться абонент. Це можна зробити в будь-якій мережі GSM. Результат: коло радіусом до 32 км з центром в місці установки який-небудь базової станції. У міських умовах радіус може бути зменшений, так як зони покриття базових станцій зазвичай невеликі. Варто зазначити, що інформація про «поточну» базову станцію оновлюється з кожним дзвінком / SMS або приблизно раз на годину, тому для підвищення точності виявлення абоненту безпосередньо перед «виміром» надсилається SMS або самому абоненту пропонується відправити SMS із запитом на кшталт «де я/де знаходиться найближчий ресторан/готель/метро/...». Цей результат можна поліпшити за допомогою методики під назвою «час прибуття». Всі базові станції в мережі потребують модернізації. Результат: коло радіусом 100-500 метрів по центру місця установки базової станції. Використання ще більш просунутих методів (їх опис можна знайти в Мережі за допомогою ключових слів «кут прильоту», «висхідна різниця часу прильоту», «GPS», «допоміжний GPS») дозволяє ще більше зменшити радіус кола або перемістити його центр в реальне місце розташування абонента. Наявність будь-якої складної системи визначення місцезнаходження абонента в мережі оператора визначити дуже просто - оператор буде продавати відповідні послуги, ніхто не буде вкладати кошти в створення необхідної інфраструктури просто так. Часто побіжного погляду на рекламні матеріали послуги достатньо, щоб визначити тип технології, яку використовує оператор, просто на основі даних точності виявлення [16].

Тріангуляція. Почнемо з аналогії. Розглянемо наступне твердження: «за допомогою утиліти ping можна визначити час проходження TCP-пакетів з одного комп'ютера на інший, а значить, оцінити відстань між ними. Потім на відстані від трьох комп'ютерів, знаючи їх координати, можна отримати координати потрібного комп'ютера». Якщо ми візьмемо чотири комп'ютера, то з'єднаємо їх мережевими кабелями між собою безпосередньо, без використання проміжних мереж, а дроти будемо прокладати строго по прямій лінії. Чи зможемо ми визначити координати центрального комп'ютера в цьому випадку, знаючи координати периферійних і використовуючи тільки пінг? Ми можемо. Чи означає це, що цим методом завжди можна скористатися? Звичайно, ні. По-перше, дроти рідко з'єднують два

комп'ютера прямо і строго по прямій, а по-друге, ми, як правило, не знаємо точних координат «еталонних» комп'ютерів. Продовжити цей список буде нескладно [10].

Висновки

Якщо мова йде не про конкретного оператора, а йдеться про GSM як технології в цілому, то можна стверджувати, що:

1. Стандартні можливості мережі GSM дозволяють будувати системи визначення місця розташування абонента на основі вимірювання параметрів радіосигналу, але стандарту GSM Phase 2+ для таких систем технологій немає.
2. Якщо така технологія не реалізована в мережі оператора, то за допомогою самої мережі можна визначити тільки останнє відоме місце розташування абонента БПЛА з точністю до базової станції, яка має реєстрацію в мережі.
3. На базі стандартного GSM можна побудувати систему визначення його місця розташування, але тільки при наявності даних про координати установки базових станцій.

Список літератури

1. Kushwaha, Kanchan, et al. Automatic Street Lighting System Utilising the Internet of Things. International Journal of Analysis of Electrical Machines, 2022, 8.2: 30-36.
2. Batra A. Street Lighting and Urban Design. 2017.
3. Agarwal T. Know About Different Types of Lights in Lighting System. 2014.
4. Archibong, Ekaette Ifiok; Ozuomba, Simeon; Ekott, Etinamabasiyaka. Internet of things (IoT)-based, solar powered street light system with anti-vandalisation mechanism. In: 2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS). IEEE, 2020. p. 1-6.
5. Umoette, Anyanime Tim; Ubom, Emmanuel A.; Festus, Mbetobong Udo. Design of stand alone floating PV system for Ibeno health centre. Sci. J. Energy Eng, 2016, 4.6: 56.
6. Nwabuokei, F. I.; Awili, C. P. N.; Ikuebebe, B. C. Design of A Stand-Alone Photovoltaic Power System: Case Study of A Residence in Ogwashi-Ukwu, Delta State. INT J, 2014, 7: 1-17.
7. Technolabs, Silicon. IR Proximity Sensor–Product Datasheet. 2018.
8. Hussain, Syed Asad, et al. Positioning a mobile subscriber in a cellular network system based on signal strength. IAENG International Journal of Computer Science, 2007, 34.2: 1-7.
9. Archibong, Ekaette Ifiok; Ozuomba, Simeon; Ekott, Etinamabasiyaka. Internet of things (IoT)-based, solar powered street light system with anti-vandalisation mechanism. In: 2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS). IEEE, 2020. p. 1-6.
10. Liu, Huiyu, et al. Mobile localization based on received signal strength and Pearson's correlation coefficient. International Journal of Distributed Sensor Networks, 2015, 11.8: 157046.
11. Ozuomba, Simeon; Archibong, Ekaette Ifiok; Ekott, Etinamabasiyaka Edet. Development Of Microcontroller-Based Tricycle Tracking Using Gps And Gsm Modules. Journal of Multidisciplinary Engineering Science and Technology (JMEST), 2020, 7.1.
12. Archibong, Ekaette Ifiok; Ozuomba, Simeon; Ekott, Etinamabasiyaka Edet. Design And Construction Of The Circuits For An Iot-Based, Stand-Alone, Solar Powered Street Light With Vandalisation Monitoring And Tracking Mechanism. Science and Technology Publishing (SCI & TECH), 2020, 4.7.
13. Brain, M. Lithium-ion Battery–Life and Death. 2006.
14. Buchmann, I. Learning the basics about batteries. Battery University, Cadex Electronics Inc. http://www. batteryuniversity. com/learn/,(Retrieved May 4th, 2019), 2019.
15. Archibong, Ekaette Ifiok; Ozuomba, Simeon; Ekott, Etinamabasiyaka Edet. Design And Construction Of The Circuits For An Iot-Based, Stand-Alone, Solar Powered Street Light With Vandalisation Monitoring And Tracking Mechanism. Science and Technology Publishing (SCI & TECH), 2020, 4.7.
16. A. Hazleton, “How to wire LEDs for 12V”, https://sciencing.com/wire-50leds-together-6072438.htm, (Retrieved November 14th, 2019), April 2017.