УДК 004.77:004.627 **doi: 10.32620/reks.2020.4.05**

Ю. С. МАНЖОС, Є. В. СОКОЛОВА

Національний аерокосмічний університет ім. М. Є. Жуковського «ХАІ», Україна

МЕТОД СТИСНЕННЯ ДАНИХ У МЕРЕЖІ ІНТЕРНЕТ РЕЧЕЙ

Інтернет речей (Internet of Things IoT) є сучасною парадигмою існування пристроїв, що надсилають та приймають повідомлення у різноманітних форматах за допомогою різних протоколів у гетерогенних мережах. Завдяки всебічному поширенню «розумних речей» виникає необхідність накопичувати великі обсяги даних, що генеруються приладами з обмеженими ресурсами, на одному чи кількох серверах. Нажаль, бездротове передавання даних є занадто «дорогим». Наприклад, у IoT, що використовують Bluetooth, надсилання даних у розумних приладах оснащених процесором STM32L, що є найекономічнішими, коштує десятки міліджоулів за секунду, у той же час повна енергія обчислення, яка необхідна для стиснення даних, складає кілька мікроджоулів. Ось чому, додаткове стиснення даних у розумних пристроях дозволяє значно зекономити енергетичні витрати. Як відомо, існують методи стиснення даних, що дозволяють стискання даних без або з інформаційними втратами. Математично доведено, що можливо побудувати як навгодно близьке наближення вхідної функції (даних з розумних IoT) до її зваженої суми узагальнених ортогональних поліномів, якими можуть бути розкладення у послідовності Чебишева та Фур'є. У статті пропонується метод стиснення інформації, особливо придатний для пристроїв IoT, що базується на паралельному використанні перетворень Чебишева та Фур'є. Для підвищення продуктивності стиснення, пропонується оптимізована модифікація алгоритму перетворення Чебишева, що дозволяє зменшити енергетичні витрати приблизно в чотири рази. Для модифікації використано метод тригонометричної оптимізації, який пряме обчислення математичної функції cos(x), що знаходиться у подвійному циклі, замінює на відповідні ітерації. Завдяки цьому алгоритм використовує одноразове обчислення функції cos(x), що дозволяє, завдяки незначному збільшенню програмного коду, значно скоротити час обчислення, тим самим зменшити ресурсні витрати. Запропоновано програмну реалізацію модифікованого алгоритму перетворення Чебишева мовою C++ та визначено енергетичну ефективність для різноманітних форм вхідних даних. Пропонований метод може бути використано не тільки у IoT, але й для накопичення даних на великих серверах.

***Ключові слова**: інтернет речей; стиснення даних зі втратами; апроксимація даних; узагальнений ортогональний поліном; перетворення Фур'є; модифіковане дискретне перетворення Чебишева; тригонометрична оптимізація; енергетична ефективність.*

Вступ

Сьогодні Інтернет речей — це глобальна мережа, що об'єднує інформаційно-комунікаційні технології інтелектуальних пристроїв, які надають широкі послуги та самостійно приймають певні рішення. Сучасні IoT мережі дозволяють накопичувати значні обсяги статистичних даних та вчасно задовольняти потреби людей. Так, у промисловості дані, що отримуються від складних виробів, дозволяють удосконалити технологічні процеси та продукцію; в медицині - вчасно діагностувати хвороби; на транспорті – знизити аварійність та пришвидшити логістику; у розумних будинках - посилити безпеку та полегшити побут.

IoT є парадигмою існування пристроїв, що обмінюються даними у гетерогенних мережах для досягнення різних цілей, використовуючи різноманітні протоколи [1].

Сучасний IoT нараховує кілька десятків мільярдів приладів, що мають унікальні ідентифікатори та дозволяють взаємодіяти у існуючій Інтернет інфраструктурі [2]. Так звані «розумні речі», тобто різноманітні електронні прилади, що мають вбудовані комп'ютери, сенсори, виконуючі органи та комунікатори, що надають цим приладам можливість передавати та обмінюватися даними [1]. Кожний з пристроїв має унікальну адресу і може взаємодіяти з іншими пристроями через Інтернет середовище. Ці пристрої мають різноманітні області застосування, що простираються від внутрішностей людського тіла до підземних та повітряних сфер.

Гетерогенність пристроїв IoT створює значні проблеми у побудові їх архітектури та протоколів взаємодії[3], які потребують використання мереж та знань розроблення вбудованого або розосередженого програмного забезпечення. Архітектура IoT має: *рівень сенсорів; мережевий рівень; рівень оброблен-*

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

ня даних; прикладний рівень.

На *сенсорному рівні*, за допомогою апаратури, що формує дані, ідентифікуються різні явища навколишнього середовища [4]. Узагальнено можливо визначити такі типи сенсорів:

– сенсори руху (лінійного чи кутового), які визначають зміни в русі чи орієнтації пристрою;

– сенсори навколишнього середовища, які вимірюють рівень світла, тиску, температури тощо.

– сенсори місцеперебування, які взаємодіють з фізичним середовищем та визначають розташування пристрою, наприклад магнітні та Global Position System (GPS) сенсори.

На *мережевому рівні* сенсори інтегруються за допомогою концентратора, що акумулює та надсилає дані до пристроїв оброблення за допомогою протоколу I2C [5] або послідовного периферійного інтерфейсу SPI [6]. Канал для передачі даних до інших сенсорів або пристроїв, побудовано за сучасними технологіями (стільниковою або супутниковою мережею; Wi-Fi [7]; Bluetooth [8]; малопотужними широкосмуговими мережами LPWAN [9] або з'єднання безпосередньо з Інтернетом).

На рівні оброблення даних головний блок обробляє й аналізує данні та приймає адекватне рішення. Інколи результати попередньо проаналізованих даних зберігаються або передаються через мережевий рівень на інші пристрої.

На рівні додатків результати роботи відтворюються за допомогою орієнтованих на користувача додатків, які можуть керувати розумними будинками, транспортом тощо.

Таким чином в IoT сенсорами формуються даних, що потім надсилаються до хмарних або інших сховищ, де потім оброблюються й інформація подається на інтерфейс користувача [10]. Для цього використовують MQTT [11] – спрощений протокол, побудований на TCP/IP, Bluetooth, UDP, який керує надсиланням даних у мережах з обмеженими ресурсами: CPU/тривалістю роботи; низькою пропускною здатністю; низькою стабільністю; значною затримкою. Префікс "MQ" успадковано від продукту IBM, що називався MQ Series та забезпечував передавання транспортної телеметрії [12]. MQTT дозволяє створювати високонадійні системи з сотень тисяч пристроїв на транспорті, виробництві, логістиці, а також у розумних будинках та містах [13]. Обмін здійснюється пакетами з простою структурою та гарантує надходження даних до адресата. За специфікацією MQTT Version 3.1.1 [14], пакет, крім унікального номеру, має рівень якості сервісу 0,1 або 2 (гарантію надходження), назву теми – текст, що може мати внутрішню ієрархію, наприклад "myhome/bathroom/temperature", власне корисне навантаження – текст, число, чи будь-які бінарні дані, а також ознаку повторного надсилання пакету – 0 або 1.

Процес отримання даних може бути простим (зчитування температури, тиску), або складним (запис аудіо, відео з камери відеоспостереження). Завдяки широкому застосуванню «розумних речей» стає загальним накопичення великих обсягів даних [15, 16], що IoT має надсилати на сервера з розосереджених та ресурсно-обмежених пристроїв [17, 18].

На жаль, бездротове надсилання даних надзвичайно енергетично витратне. Наприклад [19, 20], обмін даними через Bluetooth коштує десятки міліджоулів енергетично, а знаходження сучасного процесору у найдешевшому робочому режимі коштує близько одного мікроджоуля за секунду. Наприклад, мікроконтролери серії STM32L5, що характеризуються наднизьким витрачанням електроенергії і робочою температурою до 125°C, тримають оптимальний баланс між продуктивністю, споживаною енергією та безпекою мереж, що особливо важливо для IoT, медичної, промислової та побутової техніки. Наприклад найкраще споживання мікроконтролеру складає: 33 nA у черговому режимі; 3.6 μA у стоп режимі з повним доступом до пам'яті та периферійного обладнання та 5μs часом прокидання; до 60 μA/MHz у активному режимі [21].

Подальше дослідження особливостей IoT [22] дозволяє зробити висновок, що стиснення даних є досить природним кроком до використання розумних пристроїв [23].

Існуючі методи стиснення мають недоліки: 1) можуть бути застосованими тільки для запису специфічних даних - часових штампів [24, 25]; звукових [26] або медичних даних [27]; 2) використовують алгоритми, які дуже пристосовані до згенерованих сенсорами даних [28].

IoT використовують оцифровані дані, що замінюють собою аналогові сигнали, завдяки, перетворенню, що базується на теоремі відліків Найквіста-Шеннона [29]. Зп якою для запису людського голосу, потрібно 8 біт на відлік з частотою 8000 Гц, або 64 Кбіт/сек. Ось чому IoT необхідно зберігати чи надсилати значні обсяги даних. Через те, що ємність пам'яті та швидкість надсилання даних обмежені і виникає необхідність зменшення обсягу даних.

У загальному випадку стиснення даних може бути реалізовано як зі втратами так і без втрат. Стиснення без втрат дозволяє точно відновити реальні дані, але має обмежений коефіцієнт стиснення. У той же час стиснення зі втратами дозволяє досягти високого коефіцієнту стиснення з певною похибкою між оригінальними та реставрованими даними.

Тому важливою є ефективність стиснення, яка дозволяє досягти значного зменшення обсягу даних з одночасним збереженням релевантної інформації.

Зрозуміло, що для процесів, що не припускають втрату точності даних, слід використати ентропійне кодування [30]. Коли втрати точності припустимі [31], то для даних, що змінюються за гармонійними законами краще використати швидке перетворення Фур'є або дискретне косинусне перетворення. Якщо ж дані змінюються не за гармонійними законами, то більш ефективним є перетворення Чебишева [32], перевагою якого є мінімальна похибка.

Надалі, ми будемо фокусуватися на стисненні даних, що надходять від сенсорів приладів IoT, та має певні втрати, базується на рядах Фур'є та ортогональних поліномах Чебишева. Паралельне виконання перетворень Фур'є та Чебишева потребує багато ресурсів, тому пропонується оптимізоване дискретне перетворення Чебишева (ОДПЧ).

У цій статті в першому розділі описано математичні основи методу та особливості перетворення дискретних даних, для чого коротко розглядаються косинусне та чебишевське перетворення. У другому розділі описано тригонометрична оптимізація класичного методу Чебишева [22]. В третьому розділі пропонується метод стиснення даних, що базується на двох перетвореннях. Результати досліджень показані у четвертому розділі.

1. Математичні основи методу

Великий обсяг даних IoT передається дуже повільно, а пам'ять та енергетичні ресурси цих пристроїв дуже дорогі. Вихід полягає у тому, що дані, які зберігають інформацію та відповідають сигналам, мають бути стисненими, це дозволить зменшити кількість даних, що передаються. Дискретне унітарне перетворення корисне для оброблення сигналів перед зберіганням або надсиланням даних каналами зв'язку [21]. Прикладами поширених унітарних перетворень є дискретне косинусне перетворення (ДКП) та дискретне перетворення Чебишева. Однак для різних типів перетворень оптимальними є різні типи сигналів, тому актуальним є пошук інших унітарних перетворень. Нижче ми розглянемо коротко методи, засновані на ДКП та ДПЧ.

Для парних функцій $S(t)$ Фур'є апроксимація визначається як:

$$s_f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos\left(\frac{\pi n t}{L}\right), \qquad (1)$$

де $2L$ це період зміни $S(t)$ і

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} s(t)\cos(nt)dt\,. \qquad (2)$$

Вирази (1) та (2) є інверсним та прямим ДКП сигналу $S(t)$. Пряме перетворення (2) часто використовують для стиснення зі втратами вхідних даних.

Чебишевські поліноми першого виду, що визначаються як [20]:

$$T_n(t) = \frac{(-2)^n n!}{(2n)!}\sqrt{1-t^2}\,\frac{d^n}{dt^n}\left(\sqrt{1-t^2}\right)^{2n-1}, \qquad (3)$$

є ортогональними з вагою $\rho(t) = (1-t^2)^{-1/2}$.

Чебишевська апроксимація визначається як:

$$s_c(t) = c_0 + \sum_{n=1}^{\infty} c_n T_n(t) \quad -1 < t < 1, \qquad (4)$$

де

$$c_0 = \frac{1}{\pi}\int_{-1}^{1}\frac{s(t)dt}{\sqrt{1-t^2}},\ c_n = \frac{2}{\pi}\int_{-1}^{1}\frac{s(t)T_n(t)dt}{\sqrt{1-t^2}}. \qquad (5)$$

2. Геометрична оптимізація алгоритму перетворення Чебишева

Вирази (4, 5) є інверсним та прямим ДЧП сигналу $S(t)$. Головною особливістю Чебишевських поліноміальних апроксимацій є мінімальна похибка для $-1 < t < 1$. Програмний код ДЧП [21] подано на рис.1. Загальну складність цього алгоритму можливо приблизно визначити як $O(n^2\cos(x) + n\cos(x))$.

```
double bma = 0.5 * (b - a),
bpa = 0.5 * (b + a);
double sum = 0, y;
for (int k = 0; k < n; k++)
{
  y = cos(PI*(k+ 0.5) / n);
  f[k] = func(y*bma + bpa);
}
double sum = 0;
double fac = 2.0 / n;
for (int j = 0; j < n; j++)
{
  sum = 0.0;
  for (int k = 0; k < n; k++)
  {
sum += f[k]*cos(PI*j*(k + 0.5)/n);
  }
  c[j] = fac * sum;
}
```

Рис. 1. Програмний код ДПЧ:
a, b – межі інтервалу, n – кількість відліків, func – функція сигналу; c – вихідний вектор коефіцієнтів

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Для підвищення продуктивності ДПЧ запропоновано використати так звану тригонометричну оптимізацію.

Розглянемо перший цикл, де аргумент косинусу $\cos(\pi/2n+\pi k/n)$ має початкове значення $\pi/2n$ та постійний приріст π/n. Трудомісткість першого циклу можливо зменшити завдяки використанню ітеративних обчислень замість виконання тригонометричної функції, як показано умовно на рис. 2.

```
cosk=cos(PI*0.5/n);
sink=sin(PI*0.5/n);
cosdfk= cos(PI/n);
sindfk =sin(PI/n);
cosk1=cosk*cosdfk-sink*sindfk;
sink1=sink*cosdfk+sink*sindfk;
cosk=cosk1;
sink=sink1;
```

Рис. 2. Ітерації першого циклу, що визначають значення початкових косинусів

Таким чином, програмний код оптимізованого першого циклу перетворюється на код, зображений на рис. 3.

```
double bma = 0.5 * (b - a),bpa = 0.5 * (b + a);
double sum = 0, y;
double cosk=cos(PI*0.5/n), cosk1;
double sink=sin(PI*0.5/n), sink1;
double cosdfk= cos(PI/n), sindfk =sin(PI/n);
for (int k = 0; k < n; k++)
{
//double y = cos(PI * (k + 0.5) / n);
cosk1=cosk*cosdfk-sink*sindfk;
sink1=sink*cosdfk+cosk*sindfk;
cosk=cosk1;
sink=sink1;
y = cosk;
f[k] = func(y * bma + bpa);
}
```

Рис. 3. Оптимізований програмний код першого циклу ДПЧ:
a, b – межі інтервалу, n – кількість відліків, func – функція сігналу

Якщо згадати, що

$$\cos(\pi/n)=\cos^2(\pi/(2n))-\sin^2(\pi/(2n)),$$
$$\sin(\pi/n)=2\cos(\pi/(2n))\sin(\pi/(2n)),$$

то маємо подальшу оптимізацію коду, що зображена на рис. 4.

```
double bma = 0.5 * (b - a),bpa = 0.5 * (b + a);
double sum = 0, y;
double cosk=cos(PI*0.5/n), cosk1;
double sink=sin(PI*0.5/n), sink1;
double cosdfk = cosk*cosk-sink*sink;
double sindfk = 2* sink* cosk;
for (int k = 0; k < n; k++)
{
//double y = cos(PI * (k + 0.5) / n);
cosk1=cosk*cosdfk-sink*sindfk;
sink1=sink*cosdfk+cosk*sindfk;
cosk=cosk1;
sink=sink1;
y = cosk;
f[k] = func(y * bma + bpa);
}
```

Рис. 4. Подальша оптимізація коду ДПЧ

Розглянемо уважно подвійний цикл з рис. 1. Спробуємо виділити постійну та ітеративну частину аргументу $\cos(\pi jk/n+\pi j/2n)$.

Будемо надалі вважати, що аргумент косинусу має початкове значення та приріст, які позначимо відповідно $\varphi_0=\pi j/2n$, $\Delta\varphi=2\varphi_0$ та визначаються першим циклом та змінною j.

Тоді внутрішній цикл за k буде змінювати значення аргументу. Позначимо частину аргументу як ψ, початкове значення якої $\psi_0=0$.

Таким чином, маємо ітерацію:

$$\cos(\psi+\varphi_0)=\cos(\psi)\cos(\varphi_0)-\sin(\psi)\sin(\varphi_0),$$
$$\sin(\psi+\varphi_0)=\sin(\psi)\cos(\varphi_0)+\cos(\psi)\sin(\varphi_0).$$

Причому $\cos(\psi)$ $\sin(\psi)$ будуть ітеративно підраховуватися у циклі, за схемою:

$$\cos(\psi+\Delta\varphi)=\cos(\psi)\cos(\Delta\varphi)-\sin(\psi)\sin(\Delta\varphi),$$
$$\sin(\psi+\Delta\varphi)=\sin(\psi)\cos(\Delta\varphi)+\cos(\psi)\sin(\Delta\varphi).$$

Загалом, після проведеної оптимізації, що наведена вище, програмна реалізація перетворюється на зображену на рисунку 5:

У програмному коді, зображеному на рис. 5 маємо оптимізоване дискретне перетворення Чебишева (ОДПЧ), що має значно довший, але більш швидкий програмний код.

ОДПЧ та дискретне косинусне перетворення були використані для розроблення складного методу стиснення даних, що базується на паралельному перетворенні вхідних сигналів $s(t_i)$, наприклад, у розумних будинках. Після перетворення отримуємо апроксимовані дані $\hat{s}(t_i)$, що відповідають інформаційному змісту вхідної інформації.

Загальна кількість обчислювання функції cos(x) скоротилася з $n+n^2$ до двох. При цьому час обчислювання перетворення Чебишева скоротився майже у п'ять разів.

```
double bma = 0.5 * (b - a), bpa = 0.5 * (b + a),
sum, y,
cospi2n=cos(PI*0.5*n), sinpi2n=sin(PI*0.5*n),
cosk=cospi2n, cosk1,  sink=sinpi2n, sink1,
cosdfk = cospi2n*cospi2n-sinpi2n*sinpi2n,
sindfk = 2.*sinpi2n*cospi2n;
for (int k = 0; k < n; k++)
{
  cosk1=cosk*cosdfk-sink*sindfk
  sink1=sink*cosdfk+cosk*sindfk
  cosk=cosk1
  sink=sink1
  y = cosk;
  f[k] = func(y * bma + bpa);
}
cosdfij=cospi2n; sindfij=sinpi2n;
cosdfijk=1 sindfijk=0, cpj=1, spj=0;
cosdfj=cosdfij*cosdfij - sindfij*sinfij;
sindfj=2.*cosdfij*sindfij;
for (int j = 0; j < n; j++)
{
 sum = 0.0;
 PIJ=PI*J*0.5/n;
// CPJ=cos(PIJ); SPJ=sin(PIJ)
cpj1 = cosdfij * cpj – sindfij*spj
spj1 = sindaij * cpj + cosdfij*spj
cpj = cpj1;
spj = cpj1;
//cosdfijk=cos(PI *j/n) sindfijk=cos(PI *j/n)
  cosdfijk1 = cosdfj * cosdfijk – sindfj * sindfijk;
  sindfijk1 = sindfj * cosdfijk + cosdfj * sindfijk;
  cosdfijk = cosdfijk1;
  sindfijk = sindfijk1;
  for (int k = 0; k < n; k++)
  {
    cosk1 = cosdfijk * cosk – sindfijk*sink
    sink1  = sindfijk * cosk + cosdfijk*sink
    sum += f[k] * cosk1;
    cosk=cosk1
    sink=sink1
  }
  c[j] = fac * sum;
}
```

Рис. 5. Програмний код оптимізованого алгоритму ДПЧ де a, b – інтервал вхідних даних, n – кількість відліків сигналу, func – функція, що формує вхідні дані; c- вихідний вектор коефіцієнтів, що відповідають поліномам Чебишева

Приведену вище реалізацію доцільно використовувати для пристроїв, яким необхідно зберігати дані про різноманітні процеси, що дозволить зменшити обсяг даних і енерговитрати на перетворення і зберігання або надсилання даних.

Нехай n буде загальною кількістю відліків сигналів. У більшості алгоритмів стиснення, використовується багато метрик продуктивності, наприклад:

Відносна максимальна помилка (ВМП, або англійською RME) визначатиметься як:

$$\mathrm{RME} = \max_{i} \left| \frac{s(t_i) - s_c(t_i)}{s(t_i)} \right|. \qquad (7)$$

Коефіцієнт стиснення (КС), що визначається як відношення кількості бітів, необхідних для подання вхідних даних до кількості бітів, необхідних для подання стиснених даних (сигналів).

3. Метод стиснення даних

Для досягнення оптимального КС, на підставі ОДПЧ та ДКП, було розроблено функціональну архітектуру, зображену на рис. 6. Кожне з цих перетворень має свої переваги застосування на певних вхідних даних, а їх комплексне використання дозволяє підвищити КС.

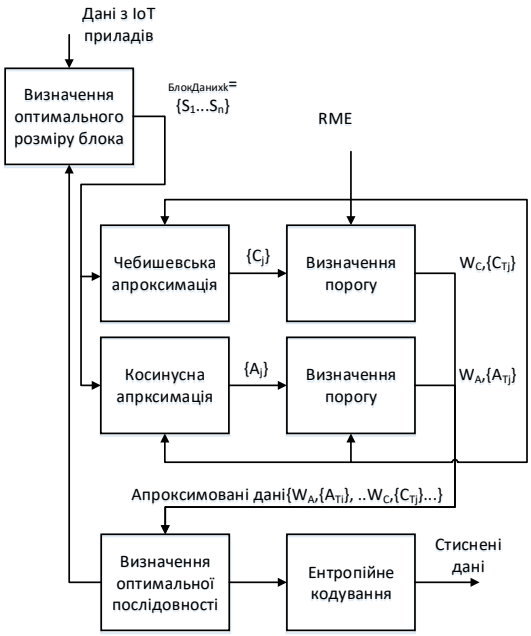


Рис. 6. Функціональна архітектура адаптивної стиснення сигналів

На першому кроці, для досягнення максимального КС процесор підраховує ДКП та ОДПЧ для мінімального блоку даних (наприклад, довжиною вісім відліків). Максимальний розмір даних визначається найвищим степенем узагальненого поліному Чебишева (або максимальною продуктивністю процесора та розміром пам'яті), наприклад максимум може складати 64. Після цієї операції, маємо дві множини коефіцієнтів апроксимації A_i, C_i.

Для апроксимації Чебишева та косинусної апроксимації, пропонується два алгоритми.

Розглянемо алгоритм чебишевської апроксимації ChebApprox:

Вхідні дані: n – кількість відліків сигналу $s(t_i)$ у блоці даних; [a; b] – інтервал апроксимації, – помилка апроксимації, яка задається користувачем, наприклад $\varepsilon = 10^{-2}$;

Вихідні дані: $\{C_i\}$ - вектор коефіцієнтів чебишевської апроксимації.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Початок алгоритму

1. Фіксуємо порядок m=n для чебишевської апроксимації.

2. Перетворюємо вхідний часовий ряд у інтервалі [a; b]. Для цього використовуємо (5) і знаходимо C_i використовуючи інтерполяцію вхідних даних.

3. Будуємо функцію, що апроксимує $\hat{s}(t_i)$ Для цього використовуємо (4).

5. Підраховуємо по (7) значення RME.

6. Якщо $RME < \varepsilon$, то призначаємо $m = m - 1$ і рухаємося до пункту 2.

Кінець алгоритму чебишевської апроксимації.

Алгоритм косинусної апроксимації Фур'є $(n, \varepsilon, s(t_i), [a;b], \{A_i\})$

Вхідні дані: n – кількість відліків сигналу $s(t_i)$ у блоці даних; [a; b] – інтервал апроксимації, ε – потрібна помилка апроксимації, наприклад $\varepsilon = 10^{-2}$.

Вихід: $\{A_i\}$ – вектор косинусної апроксимації.

ПОЧАТОК алгоритму

1. Фіксуємо порядок m=n косинусного перетворення.

2. Перетворюємо вхідний часовий ряд в інтервалі [a; b]. Для цього використовуємо вхідні дані та (2). Знаходимо коефіцієнти $\{A_i\}$.

3. На підставі (1) будуємо функцію $\hat{s}(t_i)$, що апроксимує вхідні дані.

5. Використовуємо (7) для знаходження RME.

6. Якщо $RME > \varepsilon$, то призначаємо $m = m + 1$ і ідемо до пункту 2.

7. Якщо $RME < \varepsilon$, то призначаємо $m = m - 1$ і ідемо до пункту 2.

Кінець алгоритму косинусної апроксимації.

На другому кроці процесор, відповідно до рівня порогу (точності апроксимації вхідних даних) створює відповідно (9) дві підмножини коефіцієнтів $\{A_{Ti}\}$, $\{C_{Ti}\}$ та контрольні слова W_A та W_c.

$$C_i = \begin{cases} 0, & |C_i| < \delta \\ C_i, & |C_i| \geq \delta \end{cases}, \quad A_i = \begin{cases} 0, & |A_i| < \delta \\ A_i, & |A_i| \geq \delta \end{cases}. \qquad (9)$$

Кількість елементів цих множин менша кількості елементів у апроксимуючих множинах коефіцієнтів. Ненульовий біт у контрольних словах визначає позицію ненульового елементу у вхідних множинах. Перший біт у контрольних множинах дорівнює одиниці для ДКП та нулю для ОДПЧ.

Після цього встановлюється подвоєний розмір блоку даних. Якщо ж новий розмір блоку менше за максимальний розмір, ми переходимо на перший крок. Після кількох повторів досягається оптимальний КК, що відповідає оптимальному розміру блока даних.

На третьому кроці, за допомогою ентропійного кодування, досягається додаткове стиснення і ми можемо надіслати дані до інших приладів, або зберегти їх у пам'яті.

Операції кроків 1-3 повторюються для повної множини даних, які треба зберігати або надіслати.

4. Оцінювання енергетичної ефективності

Після реалізації ОДПЧ мовою С++ була визначена відносна продуктивність (ВП) як відношення часу виконання дискретного перетворення Чебишева до часу виконання ОДПЧ для різноманітних вхідних даних.

Для унеможливлення впливу кеш-пам'яті, кожне з перетворень виконувалося 5000 разів. Час перетворення було визначено як середній час для кожного з вхідних даних (сигналів). Результати цих досліджень показані на рис.7.

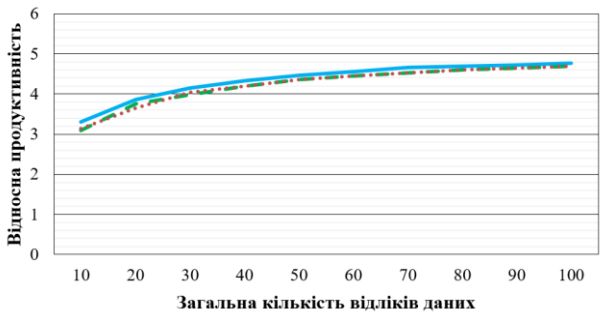


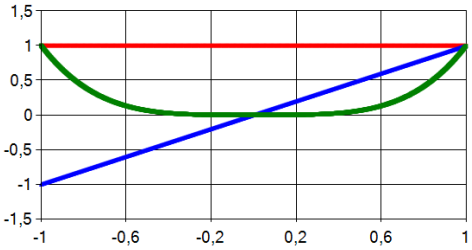
Рис. 7. Відносна продуктивність ОДПЧ

Ми бачимо, що відносна продуктивність монотонно зростає із загальною кількістю даних та має асимптотичне значення 4.7, при цьому дисперсія зростає. Таким чином, енергетична ефективність, як відношення енергії, що необхідна для класичного перетворення Чебишева до енергії, що необхідна для модифікованого перетворення Чебишева перевищує значення 4.7.

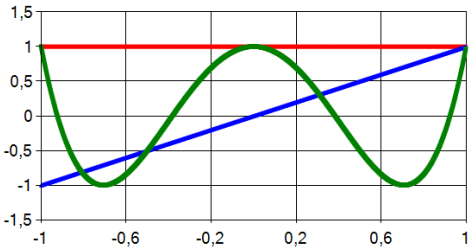
Отримані результати для різноманітних сигналів для приладів IP подано на рис. 8.

У всіх випадках використання запропонованого методу спостерігалося підвищення КС (рис. 9) та його девіації (рис. 10).

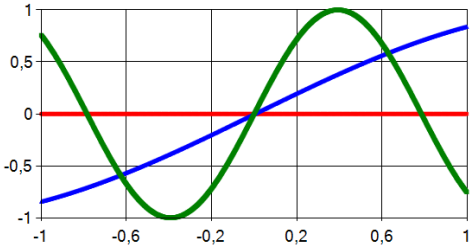
Девіація та коефіцієнт компресії значно залежить від форми сигналів. Якщо сигнали мають значну гармонічну складову, то досягається значна компресія за допомогою перетворення Фур'є (косинусного перетворення).



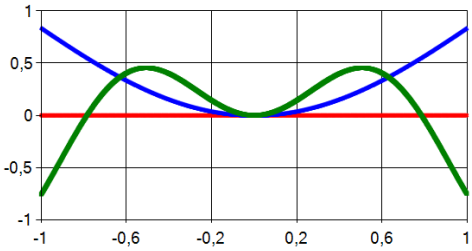
(а) $s_n(t) = t^n$,



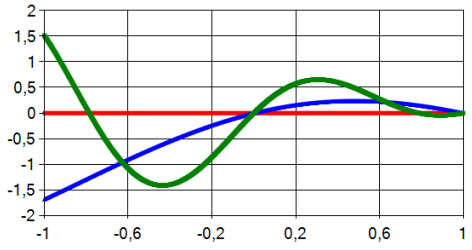
(б) $s_n(t) = T_n(t)$,



(в) $s_n(t) = \sin(nt)$,



(г) $s_n(t) = t\sin(nt)$,



(д) $s_n(t) = (1-t)\sin(nt)$,

Рис. 8. Вхідні дані для $t \in [-1;1]$, $n = \{0,1,2,5\}$, відповідно: суцільна, пунктирна, точкова, пунктирно-точкова лінії

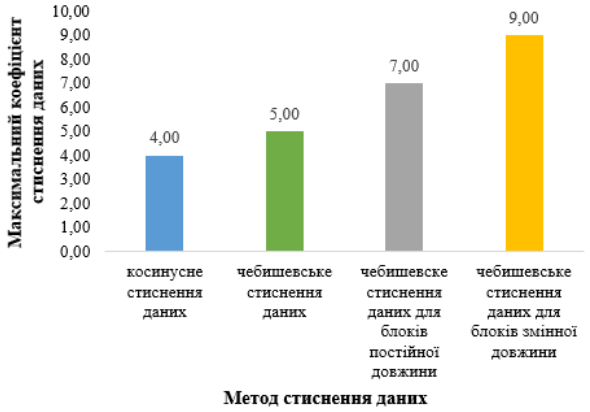


Рис. 9. Максимальний коефіцієнт стиснення

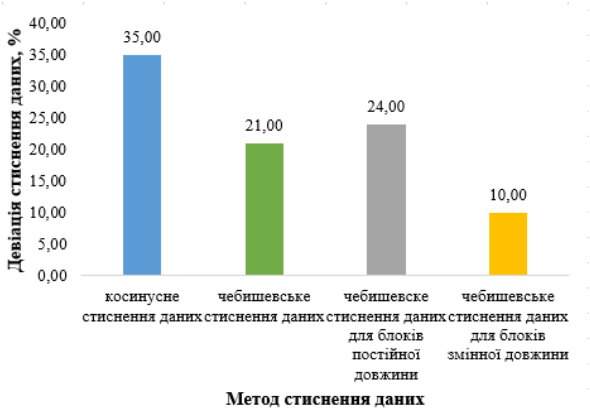


Рис. 10. Девіація відношення компресій

Якщо ж сигнал відповідає ступеневому закону, то в такому випадку перетворення Чебишева забезпечує ефективну компресію. Комплексне використання методів дозволяє досягти оптимуму у обох випадках.

Висновки

В цій статті ми запропонували оптимізацію програмного коду алгоритму дискретного перетворення Чебишева, та новий метод стиснення даних, що базується на паралельному використанні дискретного косинусного та оптимізованого дискретного Чебишевського перетворення вхідних даних, що забезпечило підвищення коефіцієнту стиснення, а тим самим загальної продуктивності як для передавання, так і для зберігання даних.

Запропонований метод стиснення даних доцільно використовувати у бездротових IoT мережах, на нижньому рівні архітектури, коли формуються великі обсяги даних, пряме надсилання яких за протоколом MQTT є неефективним, що обумовлює значні витрати енергії на роботу передавача, який у випадку зовнішніх спотворень, змушений повторно надсилати пакети. Завдяки попередньому обробленню зменшується обсяг даних, що надсилаються, що

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

не тільки зменшує час роботи передавача, але й знижує рівень перешкод для інших приладів. Загальні витрати енергії також зменшуються через дуже низьке споживання сучасних мікроконтролерів під час стиснення та завдяки модифікації алгоритму перетворення Чебишева, що збільшує його продуктивність майже у чотири рази.

Наша подальша робота полягає у експериментуванні різноманітних пристроїв для визначення універсальної ефективності запропонованої схеми з релевантними втратами під час стиснення даних.

Подяка. Це дослідження було підтримано проектом STARC (Methodology of SusTAinable Development and InfoRmation Technologies of Green COmputing and Communication), яке засновано Міністерством Освіти та Науки України.

Література

1. Bai, L. S. Archetype-based design: Sensor network programming for application experts, not just programming experts [Text] / L. S. Bai, R. P. Dick, P. A. Dinda // Proceedings of the 2009 International Conference on Information Processing in Sensor Networks. - Washington, IEEE Computer Society 2009. – P. 85–96.

2. IoT: number of connected devices worldwide 2012–2025 [Online]. - Available: https:// https://camrojud.com/%E2%80%A2-iot-number-of-connected-devices-worldwide-2012-2025/. – 10.11.2019.

3. Internet of Things: vision, applications and research challenges [Text] / D. Miorandi, S. Sicari, F. De Pellegrini, I. Chlamtac // Ad Hoc Networks, 2012. – No. 10(7). – P. 1497–1516.

4. A Survey on Sensor-based Threats to Internet-of-Things (IoT) Devices and Applications [Online]. – Available: https://arxiv.org/pdf/1802.02041.pdf. – 01.06.2019.

5. Basics of the I2C communication protocol [Online]. – Available: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol. - 12.02.2016.

6. Serial Peripheral Interface [Online]. – Available: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface. – 29.08.2019.

7. Wi-Fi certified 6 [Online]. - Available: https://www.wi-fi.org/discover-wi-fi/wi-fi-certified-6. – 3.06.2020.

8. Bluetooth Core Specification Version 5.0 Feature Overview [Online]. - Available: https://www.bluetooth.com/bluetooth-resources/bluetooth-5-go-faster-go-further/. – 30.04.2018.

9. Understanding the limits of LoRaWAN [Online] / F. Adelantado, X. Vilajosana, P. Tuset-Peiro and other. // IEEE Communications. - Available: https://arxiv.org/abs/1607.08011. - 13.02.2017.

10. IoT Explained - How Does an IoT System Actually Work? [Online]. - Available: https://medium.com/iotforall/iot-explained-how-does-an-iot-systemactually-work-e90e2c435fe7. - 01.06.2019.

11. ISO/IEC 20922:2016 Information technology - - MQ Telemetry Transport (MQTT) v3.1.1 [Online]. - Available: https://www.iso.org/standard/69466.html - 08.06. 2016.

12. MQTT: The Standard for IoT Messaging [Online]. - Available: https://mqtt.org – 12.01.2015.

13. Obermaier D. MQTT Monday Is Back - Introducing the MQTT Essentials Video Series [Online]. - Available: https://www.hivemq.com/blog/mqtt-essentials-video-series - 03.08.2020.

14. MQTT Version 5.0 OASIS Standard Specification URIs. [Online]. - Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html. – 07.03.2019.

15. Respawn: a distributed multi-resolution time-series datastore [Text] / M. Buevich, A. Wright, R. Sargent, A. Rowe // IEEE 34th Real-Time Systems Symposium. - 2013. - P. 288–297.

16. Blalock, D. Sprintz: Time Series Compression for the Internet of Things [Text] / D. Blalock, S. Madden, J. Guttag // Proceedings of the ACM Interactive, Mobile, Wearable and Ubiquitous Technologies. - 2018. - Vol. 2, No. 3. – P. 93:1-93:23.

17. Littletable: a time-series database and its uses [Text] / S. Rhea, E. Wang, E. Wong, E. Atkins, N. Storer // Proceedings of the 2017 ACM International Conference on Management of Data, 2017. – P. 125–138.

18. Sokolov, I. Resource efficient data warehouse optimization [Text] / I. Sokolov, I. Turkin // Dependable Systems, Services and Technologies Proceedings of the 9th International IEEE Conference. – Kyiv : IEEE, 2018. - P. 491-495.

19. T. Instruments. 2.4-GHz Bluetooth low energy system-on-chip [Online]. - Available: http://www.ti.com/lit/ds/symlink/cc2540.pdf - 01.06.2013

20. T. Instruments. CC2640 simplelink bluetooth wireless MCU [Online]. - Available: https://www.ti.com/lit/ds/symlink/cc2640.pdf?ts=1603975816936&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FCC2640 – 02.08.2016

21. STM32L562xx [Online]. – Available: https://www.st.com/resource/en/datasheet/stm32l562qe.pdf - 01.03.2020.

22. Signal characteristics on sensor data compression in IoT- An investigation [Text] / T. Bose, S. Bandyopadhyay, S. Kumar, A. Bhattacharyya, A. Pal // 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), 2016. – P. 1–6.

23. Ukil, A. Iot data compression: Sensor-agnostic approach [Text] / A. Ukil, S. Bandyopadhyay, A. Pal // In Data Compression Conference (DCC), 2015. – P. 303–312.

24. Andersen, M. BTrDB: Optimizing storage system design for timeseries processing [Text] / M. P. Andersen, D. E. Culler // In 14th USENIX

Conference on File and Storage Technologies (FAST'16), USENIX Association, *2016. – P. 39–52.*

25. Gorilla: A fast, scalable, in-memory time series database [Text] / T. Pelkonen, S. Franklin, J. Teller, P. Cavallaro, Q. Huang, J. Meza, K. Veeraraghavan // Proceedings of the VLDB Endowment, 2015. – V. 8, No. 12. – P. 1816–1827.

26. ISO/IEC 13818-7:2006. Information technology – generic coding of moving pictures and associated audio information [Online]. - Available: https://www.iso.org/standard/43345.html - 15.01.2006

27. A micro-power EEG acquisition SoC with integrated feature extraction processor for a chronic seizure detection system [Text] / N. Verma, A. Shoeb, J. Bohorquez, J. Dawson, J. Guttag, A. P. Chandrakasan // IEEE Journal of Solid-State Circuits, 2010. – Vol. 45, No. 4. – P. 804–816.

28. An evaluation of model-based approaches to sensor data compression [Text] / N. Q. V. Hung, H. Jeung, K. Aberer // IEEE Transactions on Knowledge and Data Engineering, 2013. – Vol. 25, No. 11. – P. 2434–2447.

29. Shannon, C. E. Communication in the presence of noise [Text] / C. E. Shannon // Proceedings of the IRE, 1949. - Vol. 37. – No. 1. - P. 10-21.

30. Huffman, D. A. Method for the Construction of Minimum-Redundancy Codes [Text] / D. A Huffman // Proceedings of the IRE, 1952. - Vol. 40. - No. 9. – P. 1098-1101.

31. Internet-of-things data aggregation using compressed sensing with side information [Text] / E. Zimos, J. F. Mota, M. Rodrigues, N. Deligiannis // Proceedings of the 23rd International Conference on Telecommunications (ICT), 2016. – P. 1-5.

32. Numerical Recipes: The Art of Scientific Computing [Text] / W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery. - Cambridge University Press, 2007. - 1262 p.

References

1. Bai, L. S., Dick, R. P., Dinda, P. A. Archetype-based design: Sensor network programming for application experts, not just programming experts. *International Conference on Information Processing in Sensor Networks*, Washington, IEEE Computer Society Publ., 2009, pp. 85–96.

2. *IoT: number of connected devices worldwide 2012–2025* Available at: https://camrojud.com/%E2%80%A2-iot-number-of-connected-devices-worldwide-2012-2025/ (accessed 10.11.2019).

3. Miorandi, D., Sicari, S., Pellegrini, F., Chlamtac I. Internet of Things: Vision, applications and research challenges. *Ad Hoc Networks*, 2012, no. 10 (7), pp. 1497–1516.

4. *A Survey on Sensor-based Threats to Internet-of-Things (IoT) Devices and Applications* Available at: https://arxiv.org/pdf/1802.02041.pdf (accessed 01.06.2019).

5. *Basics of the I2C communication protocol* Available at: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol (accessed 12.02.2016).

6. *Serial Peripheral Interface* Available at: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface (accessed 29.08.2019).

7. *Wi-Fi certified 6* Available at: https://www.wi-fi.org/discover-wi-fi/wi-fi-certified-6 (accessed 3.06.2020).

8. Bluetooth Core Specification Version 5.0 Feature Overview Available at: https://www.bluetooth.com/bluetooth-resources/bluetooth-5-go-faster-go-further/ (accessed 30.04.2018).

9. Adelantado F., Vilajosana X., Tuset-Peiro P., Martinez B., Melia J., Watteyne T. Understanding the limits of LoRaWAN. *IEEE Communications Magazine* Available at: https://arxiv.org/abs/1607.08011 (accessed 13.02.2017).

10. *IoT Explained - How Does an IoT System Actually Work?* Available at: https://medium.com/iotforall/iot-explained-how-does-an-iot-systemactually-work-e90e2c435fe7 (accessed 01.06.2019).

11. *ISO/IEC 20922:2016 Information technology - - MQ Telemetry Transport (MQTT) v3.1.1* Available at: https://www.iso.org/standard/69466.html (accessed 08.06. 2016).

12. *MQTT: The Standard for IoT Messaging* Available at: https://mqtt.org. (accessed 12.01.2015).

13. Obermaier, D. MQTT Monday Is Back - Introducing the MQTT Essentials Video Series Available at: https://www.hivemq.com/blog/mqtt-essentials-video-series/ (accessed 03.08.2020).

14. *MQTT Version 5.0 OASIS Standard Specification URIs.* Available at: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html (accesed 07.03.2019).

15. Buevich, M. Respawn: a distributed multi-resolution time-series datastore. *IEEE 34th Real-Time Systems Symposium*, 2013, pp. 288–297.

16. Blalock, D., Madden, S., Guttag, J. Sprintz: Time Series Compression for the Internet of Things. *Proceedings of the ACM Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 3, Article 93, ACM Publ., 2018, pp. 1-23.

17. Rhea, S., Wang, E., Atkins, E., Storer N. Littletable: a time-series database and its uses. *Proceedings of the 2017 ACM International Conference on Management of Data*, ACM Publ., 2017, pp. 125–138.

18. Sokolov, I., Turkin, I. Resource efficient data warehouse optimization. *Dependable Systems, Services and Technologies Proceedings of the 9th International IEEE Conference*, Kyiv, IEEE Publ. 2018, pp. 491-495.

19. *T. Instruments. 2.4-GHz Bluetooth low energy system-on-chip*. Available at: http://www.ti.com/lit/ds/symlink/cc2540.pdf (accessed 01.06.2013).

20. *T. Instruments. CC2640 simplelink bluetooth wireless MCU*. Available at: https://www.ti.com/lit/ds/symlink/cc2640.pdf?ts=1603975816936&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FCC2640 (accessed 09.12.2020).

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

21. *STM32L562xx.* Available at: https://www.st.com/resource/en/datasheet/stm32l562qe.pdf (accessed 01.03.2020).

22. Bose, T., Bandyopadhyay, S., Kumar, S., Bhattacharyya, A., Pal A. Signal characteristics on sensor data compression in IoT- An investigation. *13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON),* IEEE Publ. 2016, pp. 1-6.

23. Ukil, A., Bandyopadhyay, S., Pal, A. Iot data compression: Sensor-agnostic approach. *In Data Compression Conference (DCC),* IEEE Publ. 2015, pp. 303–312.

24. Andersen, M. P., Culler, D. E. BTrDB: Optimizing storage system design for timeseries processing, *14th USENIX Conference on File and Storage Technologies (FAST'16),* USENIX Association Publ. 2016, pp. 39–52.

25. Pelkonen, T., Franklin, S., Teller, J., Cavallaro, P., Huang, Q., Meza, J., Veeraraghavan, K. Gorilla: A fast, scalable, in-memory time series database. *Proceedings of the VLDB Endowment*, Vol. 8, № 12, VLDB Endowment Publ. 2015, pp.1816–1827.

26. *ISO/IEC 13818-7:2006. Information technology – generic coding of moving pictures and associated audio information.* Available at: https://www.iso.org/standard/43345.html (accessed -15.01.2006).

27. Verma, N., Shoeb, A., Bohorquez, J., Dawson, J., Guttag, J., Chandrakasan A. P. A micro-power EEG acquisition SoC with integrated feature extraction processor for a chronic seizure detection system. *IEEE Journal of Solid-State Circuits*, vol. 45, iss. 4, IEEE Publ. 2010, pp. 804–816.

28. Hung, N. Q. V., Jeung, H., Aberer, K. An evaluation of model-based approaches to sensor data compression. *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, iss. 11, IEEE Publ. 2013, pp. 2334–2447.

29. Shannon, C. E. Communication in the presence of noise. *Proceedings of the IRE*, vol. 37, iss. 1, Publ. 1949, pp. 10-21.

30. Huffman, D. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE*, vol. 40, iss. 9, Publ. 1952, pp. 1098-1101.

31. Zimos, E., Mota, J. F., Rodrigues, M., Deligiannis, N. Internet-of-things data aggregation using compressed sensing with side information. *Proceedings of the 23rd International Conference on Telecommunications* (ICT), IEEE Publ. 2016, pp. 1-5.

32. Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press Publ., 2007. 1262 p.

Надійшла до редакції 6.09.2020, розглянута на редколегії 16.11.2020

МЕТОД СЖАТИЯ ДАННЫХ В СЕТИ ИНТЕРНЕТ ВЕЩЕЙ

Ю. С. Манжос, Е. В. Соколова

Интернет вещей (ИВ) является современной парадигмой существования устройств, которые посылают и принимают сообщения в различных форматах с помощью различных протоколов в гетерогенных сетях. Благодаря всестороннему распространению умных вещей возникает необходимость накапливать большие объемы данных, генерируемых приборами с ограниченными ресурсами, на одном или нескольких серверах. К сожалению, беспроводная передача данных является слишком ресурсоемкой. Например, в ИВ, использующих Bluetooth, передача данных в умных вещах оснащенных процессором STM32L, являющимся самым экономичным, стоит десятки миллиджоулей в секунду, в то же время полная энергия вычисления, необходимая для сжатия данных, составляет только один микроджоуль. Поэтому, дополнительное сжатие данных в умных вещах позволяет значительно сэкономить энергетические затраты. Как известно, существуют методы сжатия данных, которые позволяют сжатие данных без или с информационными потерями. Математически доказано, что возможно построение функции имеющей наиболее близкое приближение к входной функции (данных из ИВ), представляющей взвешенную сумму обобщенных ортогональных полиномов, например, разложения в последовательности Чебышева и Фурье. В статье предлагается метод сжатия информации, особенно подходящий для сетей ИВ, основанный на одновременном использовании преобразований Чебышева и Фурье. Для повышения продуктивности сжатия, предлагается оптимизированная модификация алгоритма преобразования Чебышева, что позволяет уменьшить энергетические затраты примерно в четыре раза. Для модификации использован метод тригонометрической оптимизации, который прямое вычисление математической функции cos(x), что находится в двойном цикле классического алгоритма, заменяет на соответствующие итерации. Благодаря этому, алгоритм использует однократное вычисление функции cos(x), что позволяет, благодаря незначительному увеличению программного кода, значительно сократить вычисления и, тем самым, уменьшить ресурсные затраты. Предложено программную реализацию модифицированного алгоритма преобразования Чебышева на языке C ++ и определена энергетическую эффективность для различных форм входных данных. Предлагаемый метод может быть использован не только в ИВ, но и для накопления данных на больших серверах.

Ключевые слова: интернет вещей; сжатие данных с потерями; аппроксимация данных; обобщенный ортогональный полином; преобразования Фурье; модифицированное дискретное преобразование Чебышева; тригонометрическая оптимизация; энергетическая эффективность.

THE METHOD OF DATA COMPRESSION IN INTERNET OF THINGS COMMUNICATION

Y. Manzhos, Y. Sokolova

The Internet of Things (IoT) is a modern paradigm consisting of heterogeneous intercommunicated devices that sending and receiving messages in various formats through different protocols. Thanks to the everywhere use of smart things, it is becoming common to collect large quantities of data generated by resource-constrained, distributed devices at one or more servers. However, the wireless transmitting of data is very expensive. For example in IoT, using Bluetooth Low Energy costs tens of millijoules per connection, while computing at full energy costs only tens of microjoules, and sitting idle costs close to one microjoule per second for STM processors. That is why additional data compression for smart devices can decrease the energy costs of IoT. There are methods of data compression without or with information loss. It is mathematically proved, that it is possible to construct as arbitrarily close approximations of a weighted sum of generalized orthogonal polynomials to an input function (IoT data). In this article, we are researching the Chebyshev and Fourier sequences as an approximation of source data. For a different type of data in the different sequences, we have a different compression for Chebyshev and Fourier approximation. Concurrent use of transformations allows selecting a maximal compression for different sequences. This article proposes a compression method especially suited for IoT devices. The proposed method is based on the simultaneous use of Chebyshev and Fourier transforms. To improve the compression performance was used a trigonometric optimization. The modification of Chebyshev transformation allows reducing energy costs by about four times. Trigonometric optimization replaces the direct use of the mathematical function cos(x) in a double loop by iteration expressions. A modified algorithm uses a one-time calculation of the cos(x) function. As a result, we have a slight increase of the source code and decrease of the computation time, and increasing energy effectiveness. The software implementation in C ++ of the modified Chebyshev transformation algorithm was proposed. The proposed method can be used not only in IoT but also for the accumulation of data on big servers.

Keywords: Internet of Things; lossy signal compression; data approximation; general orthogonal polynomials; Fourier transformation; modification of Chebyshev discrete transformation; trigonometric optimization; energy effectiveness.

Манжос Юрій Семенович – канд. техн. наук, доцент, доцент кафедри інженерії програмного забезпечення, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Соколова Євгенія Віталіївна – канд. техн. наук, доцент, доцент кафедри інженерії програмного забезпечення, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Yuriy Manzhos – PhD, Assistant Professor of Department of Software Engineering Department, National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine,
e-mail: y.manzhos@khai.edu, ORCID: 0000-0002-4910-7285.

Yevheniiay Sokolova – PhD, Assistant Professor of Department of Software Engineering Department, National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine,
e-mail: y.sokolova@khai.edu, ORCID: 0000-0002-1497-4987.