УДК 004.056.5 **DOI:10.30837/rt.2022.2.209.04**

Я.А. ДЕРЕВ'ЯНКО, Ю.І. ГОРБЕНКО, канд. техн. наук, О.О. КУЗНЕЦОВ, д-р техн. наук

ФАКТОРІАЛЬНА СИСТЕМА ЧИСЛЕННЯ ДЛЯ ГЕНЕРАЦІЇ НЕЛІНІЙНИХ ПІДСТАНОВОК

Вступ

Перестановки є одним із важливих елементів комбінаторики, який повсюдно використовується у різних сферах і реальних задачах. Одним із варіантів використання перестановок у криптографії є шифрування за допомогою цих повідомлень або використання перестановок високого ступеня (2^8) в різних криптографічних перетвореннях. В деяких випадках, наприклад, якщо потрібно мати доступ до певної перестановки, необхідно мати змогу швидко отримати номер перестановки за її значеннями, або навпаки значення перестановки за її номером у загальному просторі. З такою метою доцільним буде використання факторіальної системи числення, оскільки завдяки її використанню можна швидко відтворити перестановку, маючи тільки її номер, а також виконати зворотне перетворення в разі необхідності. При цьому застосовуються спеціальні математичні поняття інверсії підстановок та алгоритми кодування у факторіальній системі числення.

В даній роботі описано загальні принципи використання факторіальної системи числення для роботи з перестановками, наведено основні алгоритми переведення номеру перестановки у її значення, а також зворотного переведення. Зазначені алгоритми реалізовано мовою програмування С. Ми наводимо чисельні приклади для демонстрації та перевірки отриманих результатів. Також в роботі досліджено зміну криптографічних показників та функцій вартості для впорядкованих у факторіальній системі числення підстановок. Такі функції вартості використовуються в евристичних методах пошуку, тобто отримані результати будуть корисними для прискорення генерації нелінійних підстановок.

1. Факторіальна систем числення

У факторіальній системі числення основами є послідовність факторіалів $b_k = k!$. Кожне натуральне число x представляється у вигляді

$$x = \sum_{k=1}^{n} a_k k!, \tag{1}$$

де $0 \le a_k \le k$

Факторіальна система числення використовується при декодуванні перестановок списками інверсій, тобто маючи номер перестановки, можна відтворити її саму наступним чином: номер перестановки (починаючи з 0) записується в факторіальній системі числення, при цьому коефіцієнт при числі $i!$ буде позначати число інверсій для елементу $i+1$ в тій множині, в якій виконуються перестановки (тобто число елементів, менших за $i+1$, але які знаходяться правіше нього в шуканій перестановці).

1.1. Алгоритми представлення цілого числа в факторіальній системі числення

Для представлення цілого числа без знаку (натуральні числа) x у факторіальній системі числення потрібно знайти набір цифр a_k, $k = 1,...,n$ таких, що

$$x = \sum_{k=1}^{n} a_k k! = a_n n! + a_{n-1}(n-1)! + ... + a_2 2! + a_1 1! \tag{2}$$

де $0 \le a_k \le k$

Для такого представлення достатньо виконати послідовність наступних кроків:

Алгоритм 1:

$$\begin{aligned} &x = a_n n! + r_n \\ &r_n = a_{n-1}(n-1)! + r_{n-1} \\ &\dots \\ &r_2 = a_2 2! + r_2; \\ &r_2 = a_1. \end{aligned} \tag{3}$$

Набір $a_n, a_{n-1}, ..., a_2, a_1$ представляє ціле число без знаку в факторіальній системі числення.

Приклад 1. Представимо число 77 в факторіальній системі числення (з використанням алгоритму 1):

$$\begin{aligned} &77 = 3 \cdot 4! + 5;\ a_4 = 3; \\ &5 = 0 \cdot 3! + 5;\ a_3 = 0; \\ &5 = 2 \cdot 2! + 1;\ a_2 = 2; \\ &1 = 1;\ a_1 = 1. \end{aligned} \tag{4}$$

Вихід: число у факторіальній системі має вигляд $\{3,0,2,1,0\}$.

Недоліком такого алгоритму є необхідність обчислення основ – послідовних значень факторіалів $k!$, $k = 1,...,n$. Для великих k та n це буде доволі складною задачею.

Для усунення даного недоліку можна представити факторіальне число x з використанням схеми Горнера:

$$x = a_n n! + a_{n-1}(n-1)! + ... + a_2 2! + a_1 1! = (...(a_n n + a_{n-1})(n-1) + ... + a_2)2 + a_1 \tag{5}$$

При такому записі числа x для знаходження всіх a_k, $k = 1,...,n-1$ достатньо виконати послідовність таких кроків:

Алгоритм 2:

$$\begin{aligned} &x = q_1 2 + a_1; \\ &q_1 = q_2 3 + a_2; \\ &q_2 = q_3 4 + a_3; \\ &\dots \\ &q_{n-3} = q_{n-2}(n-1) + a_{n-2}; \\ &q_{n-2} = q_{n-1} n + a_{n-1}; \\ &q_{n-1} = a_n. \end{aligned} \tag{6}$$

Набір залишків $a_n, a_{n-1}, ..., a_2, a_1$ представляє ціле число без знаку в факторіальній системі числення.

Приклад 2. Представимо число 77 в факторіальній системі числення (з використанням алгоритму 2):

$$\begin{aligned} &77 = 38 \cdot 2 + 1;\ a_1 = 1; \\ &38 = 12 \cdot 3 + 2;\ a_2 = 2; \\ &12 = 3 \cdot 4 + 0;\ a_3 = 0; \\ &3 = 3;\ a_4 = 3. \end{aligned} \tag{7}$$

Вихід: число у факторіальній системі має вигляд $\{3,0,2,1,0\}$.

Алгоритм 2 з використанням схеми Горнера є значно простішим у застосуванні та реалізації, оскільки не вимагає обчислення основ – послідовних значень факторіалів $k!$, $k=1,...,n$.

Для представлення перестановки у факторіальній системі числення можна застосувати декілька споріднених алгоритмів. В основі цих алгоритмів лежить застосування спеціального математичного поняття інверсії.

1.2. Алгоритми представлення підстановки в факторіальній системі числення

Інверсією в дискретній математиці називається послідовність із двох чисел впорядкованих в оберненому порядку. Ми розглядаємо інверсії в перестановках множини $X=\{1,2,...,n\}$. Кожна перестановка π може бути записана як кортеж із n елементів

$$\pi=(y_1,y_2,...,y_n) \tag{8}$$

або у вигляді підстановки, наприклад:

$$\begin{pmatrix} x_1 & x_2 & x_3 & \dots & x_n \\ y_1 & y_2 & y_3 & \dots & y_n \end{pmatrix} \tag{9}$$

де

$$\{x_1,\dots,x_n\}=\{y_1,\dots,y_n\}=X,\ \pi(x_i)=y_i \tag{10}$$

Інверсією в перестановці π називається пара індексів (i,j) для пар елементів $(\pi(i),\pi(j))$ така, що $1\le i<j\le n$ та $\pi(i)>\pi(j)$.

Наприклад, в перестановці $\pi=(4,2,1,3)$ інверсії утворюють пари індексів $(1,2)$, $(1,3)$, $(1,4)$, $(2,3)$ для пар елементів $(4,2)$, $(4,1)$, $(4,3)$, $(2,1)$ відповідно.

Перестановка називається парною, якщо її елементи утворюють у сумі парне число інверсій, і непарною, якщо вони утворюють непарне число інверсій. Наприклад, в перестановці $\pi=(4,2,1,3)$ елементи утворюють чотири інверсії, тобто перестановка парна. В перестановці $\pi=(2,1,3,4)$ інверсію утворює лише пара індексів $(1,2)$, тому перестановка непарна. В перестановці $\pi=(1,2,3,4)$ немає жодної інверсії. Число інверсій дорівнює 0, тому перестановка парна.

Факторіальна система числення використовується при декодуванні перестановок списками інверсій. Тобто маючи номер перестановки, можна відтворити саму підстановку наступним чином: номер перестановки (починаючи з 0) записується в факторіальній системі числення, при цьому коефіцієнт при числі $i!$ буде позначати число інверсій для елементу $i+1$ в тій множині, в якій виконуються перестановки (тобто число елементів, менших за $i+1$, але які знаходяться правіше нього в шуканій перестановці).

Алгоритм отримання факторіального числа $a_n,a_{n-1},...,a_2,a_1$ із перестановки π множини $X=\{1,2,...,n\}$ має таку послідовність кроків.

Алгоритм 3:

$i\leftarrow 1;$
While $i<n$ do
$k\leftarrow 0;$
For $j=i+1$ to n do
If $\pi(i)>\pi(j)$ then $k\leftarrow k+1$ (11)
$a_{\pi(i)}\leftarrow k;$
$i\leftarrow i+1;$
$a_{\pi(n)}\leftarrow 0;$
Return $a_n,a_{n-1},...,a_2,a_1$.

Приклад 3. Покажемо на прикладі як буде відбуватися переведення перестановки $\pi = (3,5,2,1,4)$ множини $X = \{1,2,3,4,5\}$ у факторіальне число:

$i = 1,\ k = 0$

$j = 2;\ \pi(1) = 3 < \pi(2) = 5,\ k = 0;$

$i = 3;\ \pi(1) = 3 > \pi(3) = 2,\ k = 1;$

$i = 4;\ \pi(1) = 3 > \pi(4) = 1,\ k = 2;$

$i = 5;\ \pi(1) = 3 < \pi(5) = 4,\ k = 2;$

$a_3 = 2;$

$i = 2,\ k = 0$

$i = 3;\ \pi(2) = 5 > \pi(3) = 2,\ k = 1;$

$i = 4;\ \pi(2) = 5 > \pi(4) = 1,\ k = 2;$

$i = 5;\ \pi(2) = 5 > \pi(5) = 4,\ k = 3;$ (12)

$a_5 = 3;$

$i = 3,\ k = 0$

$i = 4;\ \pi(3) = 2 > \pi(4) = 1,\ k = 1;$

$a_2 = 1;$

$i = 4,\ k = 0$

$i = 5;\ \pi(4) = 1 < \pi(5) = 4,\ k = 0;$

$a_1 = 0;$

$a_4 = 0;$

Return $3, 0, 2, 1, 0$.

Для переходу від факторіальної до десяткової системи числення треба скористатися формулою (1), тобто кожен з залишків помножити на відповідну йому основу – факторіал $k!$, $k = 1,...,n$.

Отже, маємо наступне:

$$x = \sum_{k=1}^{n} a_k k! = 0 \cdot 0! + 1 \cdot 1! + 2 \cdot 2! + 0 \cdot 3! + 3 \cdot 4! = 0 + 1 + 4 + 0 + 72 = 77 \quad (13)$$

Також для даного алгоритму покажемо алгоритм зворотного переходу:

Алгоритм отримання перестановки з її номеру буде полягати в наступному.

Алгоритм 4:

1. Переводимо число з десяткової системи числення з використанням алгоритму 2.

2. Отримуємо число у факторіальній системі числення.

3. Кожен з залишків числа у факторіальній системі буде означати кількість елементів, які є меншими за $n - i$, де $i = \overline{0, n-1}$ – порядок залишку у факторіальному числі, але стоять правіше нього.

4. Отримуємо перестановку, яка відповідатиме початковому числу у десятковій СЧ і у факторіальній СЧ.

Приклад 4. Покажемо на прикладі як буде відбуватися переведення числа 77 у перестановку:

1. Переводимо число 77 у факторіальну СЧ:

$$\begin{aligned} &77 = 38 \cdot 2 + 1;\ a_1 = 1; \\ &38 = 12 \cdot 3 + 2;\ a_2 = 2; \\ &12 = 3 \cdot 4 + 0;\ a_3 = 0; \\ &3 = 3;\ a_4 = 3. \end{aligned} \quad (14)$$

2. Отримали число у ФСЧ – {3,0,2,1,0}.

3. Отже, виходячи з алгоритму 4, число у факторіальній системі числення буде означати наступне для перестановки довжини 5:

- першим залишком факторіального числа буде число елементів, які є меншими за 5, але стоять правіше нього – 3;
- другим залишком буде число елементів менших за 4, але які стоять правіше нього – 0;
- третім залишком є число елементів менших за 3, які стоять правіше нього – 2;
- четвертим залишком буде число елементів, менших за 2, які стоять правіше нього – 1.
- останній елемент перестановки завжди ставиться на вільне місце, тому значення останнього залишку завжди буде нульовим.

4. Виходячи з наведених вище правил переведення факторіального числа у перестановку, отримуємо наступну перестановку для числа 77 ({3,0,2,1,0} у ФСЧ) – (3,5,2,1,4).

Отже, як можна бачити, перетворення з десяткової системи числення в факторіальну систему числення, з факторіальної системи числення у перестановку і також зворотні дії виконані успішно, отже даний метод дозволяє виконувати такі дії.

Також, варто згадати про схожий за своїм принципом код Лемера, який полягає в наступному.

У математиці і, зокрема, у комбінаториці, код Лемера – це особливий спосіб кодування кожної можливої перестановки послідовності з n чисел. Це приклад схеми перестановок нумерації та приклад таблиці інверсії. Код Лемера використовує той факт, що існує

$$n! = n \times (n-1) \times \cdots \times 2 \times 1 \tag{15}$$

перестановок послідовності з n чисел. Якщо перестановка σ задається послідовністю $(\sigma_1,...,\sigma_n)$ її образів $1,...,n$, то вона кодується послідовністю з n чисел, але не всі такі послідовності є дійсними, оскільки потрібно використовувати кожне число тільки один раз. Отже, вибирають перше число з набору з n значень, наступне число з фіксованого набору з $n-1$ значень і так далі, зменшуючи кількість можливостей до останнього числа, для якого є лише одне фіксоване значення. Переводячи цю свободу вибору на кожному кроці в число, отримуємо алгоритм кодування, який знаходить код Лемера даної перестановки. Відповідне число для кодування кожного об'єкта σ_i – це кількість об'єктів, які були доступні на той момент (тому вони не зустрічаються до позиції i), але які є меншими за об'єкт σ_i. Отже, сам алгоритм матиме такий опис:

Алгоритм 5:

1. Для отримання коду Лемера перестановки необхідно для кожного σ_i, де $i = \overline{0, n-1}$, знайти кількість елементів, які будуть меншими за нього, але знаходитимуться правіше.

2. Отримуємо таким чином код Лемера.

Приклад 5. Покажемо на прикладі як буде відбуватися переведення перестановки {1,5,0,6,3,4,2} у код Лемера:

1. Починаючи з першого числа в перестановці знаходимо числа, які є меншими за обране число і стоять правіше нього: для перестановки {1,5,0,6,3,4,2} наприклад, число елементів, менших за 1, але правіше нього – 1, менших за 5, але правіше – 4, менших на 0, але правіше 0, менших за 6, але правіше – 3, менших за 3, але правіше – 1, менших за 4, але правіше – 1, і останній лишок завжди буде нульовим, оскільки залишається тільки 1 позиція.

2. Таким чином отримуємо код Лемера – (1,4,0,3,1,1,0).

Зворотне перетворення, тобто коду Лемера у перестановку, полягає в наступному.

Алгоритм 6:

1. Береться перестановка довжини n з елементами у чіткому порядку за зростанням – {0,1,2,3,4,5,6}.

2. Далі, згідно з кодом Лемера, ми викреслюємо з даної перестановки елемент, який відповідає числу i в коді Лемера, $i = \overline{0, n-1}$.

3. Отримуємо початкову перестановку.

Приклад 6. Покажемо на прикладі як буде відбуватися перехід від коду Ламера до перестановки:

1. Береться перестановка – {0,1,2,3,4,5,6}.

2. Перше число коду Лемера – 1, тобто починаємо від елементу під номером нуль ідемо направо та викреслюємо елемент під номером, який відповідає числу в коді Лемера – 1. Перестановка матиме вигляд {0,2,3,4,5,6}. Друге число коду дорівнює 4 – викреслюємо елемент під номером 4 в перестановці, що лишилася – це число 5. Перестановка набула вигляду {0,2,3,4,6}. Третє число коду – 0 – викреслюємо число 0. Перестановка набула вигляду {2,3,4,6}. Четверте число коду дорівнює 3 – викреслюємо число 6. Перестановка набула вигляду {2,3,4}. П'яте число коду дорівнює 1 – викреслюємо число під номером 1 в перестановці – 3. Перестановка набула вигляду {2,4}. Шосте число коду – 1. Викреслюємо число 4. Перестановка набула вигляду {2}. Останнім елементом перестановки є число, що лишилося.

3. Таким чином, використовуючи алгоритм 6, ми отримали перестановку {1,5,0,6,3,4,2}.

Отже, як можна бачити, перетворення перестановки до коду Лемера та назад успішно виконується, тому такий код може також використовуватися для кодування перестановок.

2. Програмна реалізація кодування нелінійних підстановок у факторіальній системі числення

Нами було реалізовано методи, які дозволяють виконувати описані вище дії зі звичайними типами даних (int, тощо). Реалізація доступна на сервісі GitHub за посиланням [1]: https://github.com/DereviankoYaroslav/SBoxDereviankoFactorial. Далі буде наведено лістинг коду цих методів та пояснено сутність їх роботи.

Лістинг 1. Метод перетворення числа з десяткової системи до факторіальної системи числення:

Вхідні дані методу: ціле натуральне число в десятковій СЧ, яке необхідно перевести в факторіальну СЧ – int x та значення факторіалу, в межах якого це число знаходиться int n (наприклад, для чисел 0-119 $n = 5$ і т.д.).

Вихідні дані методу: масив, розміру n, кожним елементом якого будуть залишки факторіального числа за відповідною основою $k!$, $k = 1,...,n$, весь масив в цілому представляє число у факторіальному вигляді.

```
int *numberToFactorial(int x, int n){
  int q = 2;
  int counter = n-2;
  int *positions = calloc (n,sizeof(int));
  positions[n-1] = 0;
  while (counter > 0){
    long long temp = floor(x/q);
    long long val = x – (temp*q);
    positions[counter] = val;
    x = temp;
    ++q;
    counter--;
  }
  positions[counter] = x;
  printf("\nPOSITIONS VECTOR\n");
  for (int i = 0; i < n ; ++i){
    printf("%d ",positions[i]);
  }
```

```
    int *positionsRev = calloc (n,sizeof(int));
    for (int r = 0,t = n-1; r < n, t>=0; ++r, t--){
        positionsRev[t] = positions[r];
    }
    free(positions);
    return positionsRev;
}
```

Лістинг 2. Метод перетворення факторіального числа до перестановки (за алгоритмом 3):

Вхідні дані методу: масив, розміру n, кожним елементом якого будуть залишки факторіального числа за відповідною основою $k!$, $k = 1,...,n$, розмір масиву – $size = n$.

Вихідні дані методу: перестановка довжини n, яка відповідатиме вхідному факторіальному числу.

```
    int *numberToSubstitution(int *number, int size){
    int *S = calloc (size,sizeof(int));
    int *result = calloc (size,sizeof(int));
    int newSize = size;
    int counter = 0;
    int coeffNum = 0;
    int innerCounter = 0;
    while(counter < size){
        int *emptyPos = calloc (newSize,sizeof(int));
        for (int i = 0; i < size; ++i){
            if (S[i]==0){
                emptyPos[innerCounter] = i;
                innerCounter++;
            }
        }
        for (int q = 0; q < newSize; ++q){
            if (q==number[newSize-1]){
                S[emptyPos[q]] = 1;
                result[coeffNum] = emptyPos[q];
                ++coeffNum;
            }
        }
        innerCounter = 0;
        ++counter;
        newSize--;
        free(emptyPos);
    }
    int numberInArr = size;
    int *sub = calloc (size,sizeof(int));
    for (int u = 0; u < size; ++u){
        sub[result[u]] = numberInArr;
        numberInArr--;
    }
    free(result);
    int *subRev = calloc (size,sizeof(int));
    for (int r = 0,t = size-1; r < size, t>=0; ++r, t--){
        subRev[t] = sub[r];
    }
```

```
    free(sub);
    free(S);
    return subRev;
}
```

Лістинг 3. Метод зворотного перетворення перестановки у факторіальне число:

Вхідні дані методу: перестановка довжини n ($size = n$), розмір перестановки $size = n$

Вихідні дані методу: масив, розміру n, кожним елементом якого будуть залишки факторіального числа за відповідною основою $k!$, $k = 1,...,n$, весь масив в цілому представляє число у факторіальному вигляді.

```
    int *substitutionToFactorial(int *sub, int size) {
    int *result = calloc(size,sizeof(int));
    int value = size;
    int flag = 0;
    int innerCounter = 0;
    int counter = 0;
    while (counter<size) {
        for (int i = 0; i < size; ++i) {
            if (flag == 1 && sub[i] < value) {
                ++innerCounter;
            }
            if (sub[i] == value) {
                flag = 1;
            }
        }
        result[counter] = innerCounter;
        innerCounter = 0;
        flag = 0;
        ++counter;
        value--;
    }
    return result;
}
```

Лістинг 4. Метод зворотного перетворення факторіального числа у десяткове число:

Вхідні дані методу: масив, розміру n, кожним елементом якого будуть залишки факторіального числа за відповідною основою $k!$, $k = 1,...,n$, розмір масиву $size = n$.

Вихідні дані методу: ціле натуральне число у десятковій СЧ – int result.

```
    long long factorialNumberToNumber(int *number, int size){
    int *numberRev = calloc (size,sizeof(int));
    long long result = 0;
    for (int r = 0,t = size-1; r < size, t>=0; ++r, t--){
        numberRev[t] = number[r];
    }
    for (int i = 0; i < size; ++i){
        result += numberRev[i]*factorialCounting(i);
    }
    free(numberRev);
    return result;
}
```

2.1. Приклади переведення чисел у перестановки і зворотні перетворення

Покажемо результати виконання послідовного перетворення десяткового числа у факторіальне число, факторіального числа у перестановку, перестановку назад у факторіальне число і далі – факторіальне число у десяткове число. Для наочної демонстрації результатів покажемо спочатку переведення у факторіальну систему числення числа з прикладу – 77, а потім перетворення числа 100 з прикладу та ще декількох чисел на перестановки та назад.

Переведення числа 77 з десяткової СЧ у факторіальну буде відбуватися з використанням схеми Горнера, так наприклад, спочатку число буде ділитися на 2, і першим числом факторіального числа буде залишок від такого ділення, далі на 3, і так далі, поки дільник не буде дорівнювати $n-1$. Дільником на кожному кроці буде виступати ціла частина від такого ділення. Така дія матиме наступний вигляд:

```
START NUMBER
77

a_n = 1
a_n = 2
a_n = 0
a_n = 3

POSITIONS VECTOR
3, 0, 2, 1, 0,
```

Рис. 1. Приклад переведення числа 77 у ФСЧ

Перше значення буде залишком від ділення на 2, друге на 3 і т.д. аж до $n-1$. Таким чином, ці залишки будуть утворювати факторіальне число. Останній залишок факторіального числа завжди буде нульовим. Далі буде показано результати перетворення різних чисел з десяткової СЧ у ФСЧ, із ФСЧ у перестановку, і результати зворотних дій. Деталі перетворень описані у коді алгоритмів:

```
START NUMBER
77

FACTORIAL NUMBER
3, 0, 2, 1, 0,

SUBSTITUTION
3, 5, 2, 1, 4,

FACTORIAL NUMBER
3, 0, 2, 1, 0,

FINAL NUMBER
77
```

Рис. 2. Переведення числа 77 у перестановку і назад (n=5)

```
START NUMBER
100

FACTORIAL NUMBER
4, 0, 2, 0, 0,

SUBSTITUTION
5, 3, 1, 2, 4,

FACTORIAL NUMBER
4, 0, 2, 0, 0,

FINAL NUMBER
100
```

Рис. 3. Переведення числа 100 у перестановку і назад (n=5)

```
START NUMBER
2999999

FACTORIAL NUMBER
8, 2, 3, 1, 3, 4, 3, 2, 1, 0,

SUBSTITUTION
5, 10, 4, 6, 3, 8, 2, 9, 7, 1,

FACTORIAL NUMBER
8, 2, 3, 1, 3, 4, 3, 2, 1, 0,

FINAL NUMBER
2999999
```

Рис. 4. Переведення числа 2999999 у перестановку і назад (n=10)

3. Експерименти

Для демонстрації та перевірки правильності застосування розроблених алгоритмів в роботі проведено низку експериментів із представлення нелінійних підстановок відомих стандартів симетричного криптоперетворення. Зокрема, ми розглядаємо найпростіший приклад 4-бітних S-boxes із відомого алгоритму шифрування ГОСТ 28147-89 (приклади підстановок описані в українському стандарті). Також ми розглядаємо 8-бітну підстановку сучасного американського стандарту симетричного шифрування AES.

Ми наводимо приклади переходу від перестановок до факторіальних чисел і від них, до чисел у десятковій системі числення. Також ми наводимо заміри швидкості виконання операцій такого переходу. Для кожного прикладу пораховано нелінійність підстановки та «вартість» з використанням різних цінових функції. Зокрема, ми розглядаємо цінові функції WHS PCF та WCF з [4]. Такі евристичні функції вартості застосовують в ітераційних алгоритмах генерації S-Boxes, наприклад у HC [5], SA [6] та інших евристичних алгоритмах пошуку.

3.1. Аналіз перестановок (S-box'ів), що використовуються в українській версії ГОСТ 28147-89 (приклади з українського стандарту ДСТУ 4145-2002)

Покажемо демонстрацію роботи з нелінійними вузлами заміни на прикладі S-boxes стандарту ГОСТ 28147-89 (приклади підстановок наведено в українському стандарті ДСТУ 4145-2002).

Блоки для українського варіанту ГОСТ 28147-89 мають вигляд, як у табл. 1. Тут і надалі S-boxes подаються у шістнадцятирічному вигляді.

Таблиця 1

Нелінійні вузли заміни алгоритму ГОСТ 28147-89

S-Box	Шістнадцятирічне представлення S-Box'у																NL	Cost
1	A	9	D	6	E	B	4	5	F	1	3	C	7	0	8	2	4	0
2	8	0	C	4	9	6	7	B	2	3	1	F	5	E	A	D	4	0
3	F	6	5	8	E	B	A	4	C	0	3	7	2	9	1	D	4	0
4	3	8	D	9	6	B	F	0	2	5	C	A	4	E	1	7	4	0
5	F	8	E	9	7	2	0	D	C	6	1	5	B	4	3	A	4	0
6	2	8	9	7	5	F	0	B	C	1	D	E	A	3	6	4	4	0
7	3	8	B	5	6	4	E	A	2	C	1	7	9	F	D	0	4	0
8	1	2	3	E	6	D	B	8	F	A	C	5	7	9	0	4	4	0

Тепер покажемо (табл. 2) перехід від такого представлення до числа у факторіальній системі числення за допомогою алгоритму 3, псевдокод якого наведено у лістингу 2.

Таблиця 2

Факторіальне представлення нелінійних вузлів заміни алгоритму ГОСТ 28147-89

S-Box	Факторіальне представлення S-Box'у															
1	7	10	11	4	8	10	9	1	2	6	4	4	2	0	1	0
2	4	2	0	10	5	0	6	8	4	4	0	3	1	1	0	0
3	15	11	0	6	8	7	1	6	2	6	5	4	2	1	0	0
4	9	2	11	4	7	3	7	7	0	5	2	1	3	1	0	0
5	15	13	8	7	3	0	8	8	7	4	2	1	0	2	0	0
6	10	4	4	5	5	3	7	7	6	1	4	0	0	2	0	0
7	2	8	1	4	9	5	1	7	1	4	4	3	3	2	1	0
8	7	11	9	5	7	5	2	4	2	3	2	0	1	1	1	0

Коли всі блоки показані у вигляді факторіальних чисел, отримаємо з них числа у десятковій системі числення, тобто їхні номери у одновимірному просторі перестановок (табл. 3).

Таблиця 3

Представлення нелінійних вузлів заміни алгоритму ГОСТ 28147-89
у вигляді номеру у одновимірному просторі

S-Box	Номерне представлення S-Box'у
1	10096275666829
2	5410046177360
3	20577296088710
4	12014132259884
5	20801725497628
6	13452973260244
7	3321296141615
8	10171418435529

Як видно з таблиць, ми отримали номерне представлення блоків у одновимірній множині усіх блоків і тепер можемо робити будь-які маніпуляції з цими номерами для пошуку,

наприклад, нових блоків з хорошими показниками, але відмінними від тих, що використовуються в стандарті.

Нами проведено заміри швидкодії операцій перетворення перестановки у номер, а також зворотного перетворення. Для цього було проведено тестування, шляхом виконання операцій над одним і тим же блоком 1000000 разів, щоб порахувати середню швидкість виконання операції в мілісекундах, оскільки ця швидкість має дуже маленьке значення і не відображається коректно шляхом меншої кількості тестів.

Лістинг коду для проведення тестування наводиться нижче:

Лістинг 5. Вимірювання швидкодії операцій переходу від перестановки до номеру:

```
long t;
t = mtime();
for (int y = 0 ; y < 100000000; ++y) {
      int *factNum = substitutionToFactorial(sub2, n);
      long long number = factorialNumberToNumber(factNum, n);
      free(factNum);
}
t = mtime() – t;
printf ("%ld milliseconds\n",t);
```

Лістинг 6. Вимірювання швидкодії операцій переходу від номеру до перестановки:

```
t = mtime();
   for (int y = 0 ; y < 100000000; ++y) {
      int *arr = numberToFactorial(a, n);
      int *sub = numberToSubstitution(arr, n);
      free(arr);
      free(sub);
   }
t = mtime() – t;
printf ("%ld milliseconds\n",t);
```

Результати вимірювання швидкодії операції наведено у табл. 4.

Таблиця 4

Результати вимірювання швидкодії операції, в мілісекундах

Перехід від перестановки до номеру	0,001217 мс
Перехід від номеру до перестановки	0,003457 мс

Також для виконання програми з заданими розмірами блоків (4×4) було виміряне використання оперативної пам’яті під час роботи. Показник використання незалежно від кількості блоків коливався від 352Кб до 364 Кб (рис. 5). Тобто, програмі виділяється потрібна їй пам’ять, далі програма виконує необхідні операції і звільняє пам’ять для наступних операцій.

Рис. 5. Результати тестування витрат пам’яті

Отже, виходячи з отриманих результатів, виконання всіх необхідних операцій є доволі швидким (1-2 мікросекунди для переходу від перестановки до номеру і 3-4 мікросекунди – для переходу від номеру до перестановки) і при цьому не навантажує оперативну пам’ять комп’ютера – використання менше ніж половина мегабайта.

3.2. Аналіз S-box алгоритму AES

У даному розділі продемонструємо роботу з нелінійними вузлами заміни на прикладі AES S-box'у. Для роботи з подібними вузлами проект було адаптовано на мову програмування C++ , також було використано бібліотеку NTL для роботи з нескінченно великими числами. Реалізація доступна на сервісі GitHub за посиланнями [2, 3]:

https://github.com/DereviankoYaroslav/FactorialPSO-WHS та
https://github.com/DereviankoYaroslav/FactorialPSO-CUBA.

Таблиця 5

Нелінійний вузол заміни алгоритму AES

63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Показники у даного блоку будуть наступними (табл. 6):

Таблиця 6

Показники AES-Sbox

S-Box	NL	Cost WHS	Cost WCF	Cost PCF
AES	112	4151216	0	7491.62

Тепер покажемо (табл. 7) перехід від такого представлення до числа у факторіальній системі числення за допомогою алгоритму 3, псевдокод якого наведено у лістингу 2.

Таблиця 7

Факторіальне представлення нелінійного вузла заміни алгоритму AES

130	242	221	169	155	231	149	30	212	40	133	67	126	238	203	220
150	99	161	120	182	66	20	52	73	10	193	73	159	175	28	85
15	87	48	93	83	113	23	179	204	152	63	192	72	107	143	130
130	17	105	173	133	189	187	65	164	46	191	96	160	69	27	170
10	126	40	101	1	48	26	73	161	96	33	40	126	137	119	2
157	45	157	58	161	109	49	97	81	39	141	147	93	147	10	117
93	21	87	141	15	125	4	18	77	123	48	14	131	81	47	60
79	14	42	9	27	26	8	53	10	14	77	89	98	119	54	100
73	60	114	123	121	31	37	29	118	111	93	25	51	105	100	22
109	85	32	30	106	73	11	5	100	18	25	46	99	28	14	41
46	35	41	27	61	68	86	57	12	23	8	1	58	61	45	46
34	21	47	48	15	47	23	13	68	27	41	32	41	4	3	35
58	11	27	33	40	20	34	31	16	50	8	48	30	18	45	48
31	12	2	32	42	18	28	3	30	37	10	11	32	15	20	21
8	3	4	6	18	18	11	21	11	0	20	9	9	14	2	9
0	1	0	5	4	3	5	1	4	1	3	3	0	0	1	0

Тепер отримаємо з цього факторіального числа число у десятковій системі числення, тобто номер цієї перестановки у одновимірному просторі перестановок (табл. 8).

Таблиця 8

Представлення нелінійного вузла заміни алгоритму AES у вигляді номеру у одновимірному просторі

AES-SBox	438802080077461506931487886587859279629566435895940576091249389663118690638511176335757277833257114168214620949945405264375192303618393662335277010380153387264179445016571705252302920569441228045303327567040823942746123930949356593784189672702597268903394924732421379919535170324637836064806689559503327871102084486758924065072351307405825431157286059632497461568460660658015651849289360349395308511707577545803238296080881647265855676077166079014716466333599431175990239379323828746367933202518757906752833

Як видно з таблиць, так як і для блоків ГОСТ 28147-89, для блоку AES ми отримали номерне представлення у одновимірній множині усіх блоків і тепер можемо робити будь-які маніпуляції з цими номерами для пошуку, наприклад, нових блоків з хорошими показниками, але відмінними від тих, що використовуються в стандарті. Далі буде показано можливе практичне застосування цього.

Для блоків розміром 256 елементів також було виконано заміри швидкодії переходу до різних систем числення, а також пам’ять, що використовується при роботі програми. Заміри відбувалися так, як показано в пункті 4.1.

Результати вимірювання швидкодії операції наведено у табл. 9.

Таблиця 9

Результати вимірювання швидкодії операції, в мілісекундах, для блоків розміру 256 елементів

Перехід від перестановки до номеру	2,86940 мс
Перехід від номеру до перестановки	0,60700 мс

Також для виконання програми з заданими розмірами блоків (256 елементів) було виміряно використання оперативної пам’яті під час роботи. Показник використання в залежності від кількості блоків, задіяних для пошуку відрізнявся від 10,5 Мб для 10000 блоків до 101 Мб для 100000 блоків (рис. 6 та 7). Тобто програмі виділяється потрібна для роботи з певною кількістю блоків пам’ять, далі програма виконує необхідні операції і звільняє пам’ять для наступних операцій.

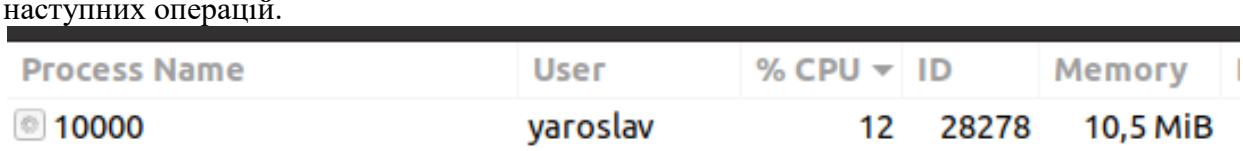

Рис. 6. Результати тестування витрат пам’яті при роботі алгоритму пошуку з кількістю 10000 блоків розміру 256

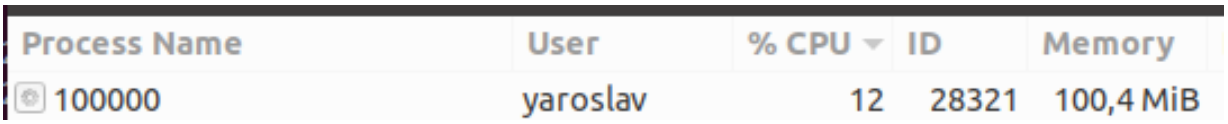

Рис. 7. Результати тестування витрат пам’яті при роботі алгоритму пошуку з кількістю 100000 блоків розміру 256

Нами також було проведено дослідження простору блоків поруч з AES-Sbox для виявлення показників блоків, що знаходяться поруч із ним. Це може допомогти в подальшому краще розуміти простір блоків та оптимізувати пошуки як цим, так і іншими алгоритмами

пошуку. Дані нелінійності та функцій вартості [4] для блоків 100 номерів до та 100 номерів після S-box'у алгоритму AES наводяться у табл. 10.

Таблиця 10

Результати вимірювання нелінійності та функцій вартості блоків у просторі навколо AES-Sbox

AES NUMBER (+/-): -100	Cost WHS	Cost WCF	Cost PCF	NL
AES NUMBER (+/-): -99	4244476	32247990190080	1975.76	108
AES NUMBER (+/-): -98	4249994	54222284390400	2008.22	108
AES NUMBER (+/-): -97	4232112	45565744250880	2003.28	108
AES NUMBER (+/-): -96	4224624	29299058933760	1972.38	108
AES NUMBER (+/-): -95	4278078	36718950481920	1974.98	108
AES NUMBER (+/-): -94	4228678	39001994035200	2002.33	108
AES NUMBER (+/-): -93	4300764	48324421877760	2005.2	108
AES NUMBER (+/-): -92	4196718	41950925291520	1997.79	108
AES NUMBER (+/-): -91	4215350	43853461585920	1998.06	108
AES NUMBER (+/-): -90	4233146	56219947499520	2021.2	108
AES NUMBER (+/-): -89	4251950	67159531192320	1000.43	106
AES NUMBER (+/-): -88	4220900	56029693870080	2028.86	108
AES NUMBER (+/-): -87	4266858	58217610608640	2029.28	108
AES NUMBER (+/-): -86	4218474	65352121712640	1001.52	106
AES NUMBER (+/-): -85	4245628	56600454758400	2023.21	108
AES NUMBER (+/-): -84	4255832	49085436395520	2024.04	108
AES NUMBER (+/-): -83	4312198	64971614453760	1003.94	106
AES NUMBER (+/-): -82	4239532	39192247664640	2001.74	108
AES NUMBER (+/-): -81	4304420	44424222474240	2005.43	108
AES NUMBER (+/-): -80	4223506	69062067486720	1004.6	106
AES NUMBER (+/-): -79	4232028	58407864238080	2029.32	108
AES NUMBER (+/-): -78	4242676	51653860392960	2019.49	108
AES NUMBER (+/-): -77	4280238	56600454758400	2023.04	108
AES NUMBER (+/-): -76	4290560	44614476103680	1998.27	108
AES NUMBER (+/-): -75	4301994	42426559365120	1999.36	108
AES NUMBER (+/-): -74	4276960	65161868083200	2031.77	108
AES NUMBER (+/-): -73	4250832	58027356979200	2029.31	108
AES NUMBER (+/-): -72	4254212	44519349288960	2002.02	108
AES NUMBER (+/-): -71	4200800	30821087969280	1971.74	108
AES NUMBER (+/-): -70	4197658	48704929136640	1995.49	108
AES NUMBER (+/-): -69	4153708	16932573020160	3889.49	110
AES NUMBER (+/-): -68	4219674	43282700697600	1997.28	108
AES NUMBER (+/-): -67	4229136	25208605900800	1971.72	108
AES NUMBER (+/-): -66	4301116	26540381306880	1972.54	108
AES NUMBER (+/-): -65	4240594	16932573020160	3902.09	110
AES NUMBER (+/-): -64	4222232	38621486776320	1996.08	108
AES NUMBER (+/-): -63	4218076	25303732715520	1970.26	108
AES NUMBER (+/-): -62	4247938	47182900101120	2001.29	108
AES NUMBER (+/-): -61	4304304	43472954327040	2002.08	108
AES NUMBER (+/-): -60	4254024	30821087969280	1968.48	108
AES NUMBER (+/-): -59	4196730	11034710507520	3841.33	110
AES NUMBER (+/-): -58	4231694	38716613591040	1998.56	108
AES NUMBER (+/-): -57	4221304	33294385152000	1972.31	108
AES NUMBER (+/-): -56	4210396	28347790786560	1969.27	108
AES NUMBER (+/-): -55	4257300	42711939809280	1995.62	108
AES NUMBER (+/-): -54	4215518	27777029898240	1973.42	108
AES NUMBER (+/-): -53	4218746	17598460723200	3905	110
AES NUMBER (+/-): -52	4203988	45565744250880	2003.24	108
AES NUMBER (+/-): -51	4247010	53841777131520	2007.28	108
AES NUMBER (+/-): -50	4156984	26635508121600	1968.75	108
AES NUMBER (+/-): -49	4196778	45090110177280	1998.77	108
AES NUMBER (+/-): -48	4265426	30440580710400	1974.95	108

AES NUMBER (+/-): -47	4203034	17693587537920	3904.35	110
AES NUMBER (+/-): -46	4168880	34150526484480	1969.22	108
AES NUMBER (+/-): -45	4134586	11034710507520	3841.33	110
AES NUMBER (+/-): -44	4180802	26730634936320	1970.48	108
AES NUMBER (+/-): -43	4208900	16361812131840	3900.12	110
AES NUMBER (+/-): -42	4302216	43092447068160	1997.97	108
AES NUMBER (+/-): -41	4241694	26730634936320	1971.04	108
AES NUMBER (+/-): -40	4215210	42807066624000	1997.84	108
AES NUMBER (+/-): -39	4219576	30440580710400	1971.41	108
AES NUMBER (+/-): -38	4240916	45375490621440	2002.8	108
AES NUMBER (+/-): -37	4305804	49370816839680	2003.29	108
AES NUMBER (+/-): -36	4233768	47373153730560	1992.81	108
AES NUMBER (+/-): -35	4176474	16171558502400	3888.69	110
AES NUMBER (+/-): -34	4243308	43377827512320	1998.41	108
AES NUMBER (+/-): -33	4222804	26540381306880	1970.91	108
AES NUMBER (+/-): -32	4222010	25208605900800	1968.46	108
AES NUMBER (+/-): -31	4258800	39572754923520	1994.41	108
AES NUMBER (+/-): -30	4185168	43948588400640	1996.97	108
AES NUMBER (+/-): -29	4188396	29108805304320	1970.45	108
AES NUMBER (+/-): -28	4206622	53556396687360	2005.33	108
AES NUMBER (+/-): -27	4248510	49465943654400	2005.69	108
AES NUMBER (+/-): -26	4159618	32818751078400	1970.42	108
AES NUMBER (+/-): -25	4198278	43568081141760	1997.31	108
AES NUMBER (+/-): -24	4211454	17408207093760	3904.26	110
AES NUMBER (+/-): -23	4149062	11034710507520	3852.16	110
AES NUMBER (+/-): -22	4129464	27016015380480	1971.95	108
AES NUMBER (+/-): -21	4139788	16361812131840	3900.94	110
AES NUMBER (+/-): -20	4141386	32057736560640	1975.61	108
AES NUMBER (+/-): -19	4214102	27777029898240	1974.82	108
AES NUMBER (+/-): -18	4201340	48039041433600	1999.53	108
AES NUMBER (+/-): -17	4147928	33960272855040	1971.05	108
AES NUMBER (+/-): -16	4151220	45755997880320	1999.98	108
AES NUMBER (+/-): -15	4160410	27396522639360	1970.32	108
AES NUMBER (+/-): -14	4173236	46802392842240	2003.9	108
AES NUMBER (+/-): -13	4235838	42521686179840	2002.73	108
AES NUMBER (+/-): -12	4192066	40428896256000	1996.81	108
AES NUMBER (+/-): -11	4181676	29489312563200	1971.26	108
AES NUMBER (+/-): -10	4223936	28538044416000	1972.34	108
AES NUMBER (+/-): -9	4203432	17598460723200	3902.97	110
AES NUMBER (+/-): -8	4227212	47087773286400	2001.23	108
AES NUMBER (+/-): -7	4217098	47087773286400	2000.85	108
AES NUMBER (+/-): -6	4150576	43187573882880	2000.74	108
AES NUMBER (+/-): -5	4193598	46326758768640	2003.66	108
AES NUMBER (+/-): -4	4183560	35006667816960	1979.26	108
AES NUMBER (+/-): -3	4225448	30440580710400	1979.06	108
AES NUMBER (+/-): -2	4164820	52510001725440	2004.23	108
AES NUMBER (+/-): -1	4163686	44804729733120	2001.12	108
AES NUMBER (+/-): 0	4185510	11700598210560	3848.05	110
AES NUMBER (+/-): 1	4151216	0	7491.62	112
AES NUMBER (+/-): 2	4200066	17598460723200	3909.85	110
AES NUMBER (+/-): 3	4210390	11985978654720	3857.61	110
AES NUMBER (+/-): 4	4143540	11320090951680	3852.37	110
AES NUMBER (+/-): 5	4188158	17408207093760	3902.36	110
AES NUMBER (+/-): 6	4194032	42807066624000	1969.83	108
AES NUMBER (+/-): 7	4150082	11034710507520	3841.33	110
AES NUMBER (+/-): 8	4200466	36338443223040	1976.81	108
AES NUMBER (+/-): 9	4209656	18359475240960	3904.74	110
AES NUMBER (+/-): 10	4175390	17027699834880	3904.83	110
AES NUMBER (+/-): 11	4228530	30821087969280	1971.69	108

AES NUMBER (+/-): 12	4253206	30916214784000	1972.28	108
AES NUMBER (+/-): 13	4249050	17122826649600	3905.77	110
AES NUMBER (+/-): 14	4245084	18549728870400	3907.39	110
AES NUMBER (+/-): 15	4249450	12461612728320	3857.38	110
AES NUMBER (+/-): 16	4248272	37479964999680	1978.98	108
AES NUMBER (+/-): 17	4256794	45185236992000	1978.69	108
AES NUMBER (+/-): 18	4152730	17788714352640	3900.44	110
AES NUMBER (+/-): 19	4192524	35767682334720	1974.61	108
AES NUMBER (+/-): 20	4185714	12461612728320	3857.38	110
AES NUMBER (+/-): 21	4224374	18074094796800	3909.61	110
AES NUMBER (+/-): 22	4213978	50702592245760	1983.7	108
AES NUMBER (+/-): 23	4212844	38336106332160	1980.03	108
AES NUMBER (+/-): 24	4180910	16361812131840	3902.3	110
AES NUMBER (+/-): 25	4209008	11034710507520	3852.16	110
AES NUMBER (+/-): 26	4195466	28347790786560	1977.02	108
AES NUMBER (+/-): 27	4268182	35482301890560	1979	108
AES NUMBER (+/-): 28	4127018	17027699834880	3899.93	110
AES NUMBER (+/-): 29	4171636	29489312563200	1971.7	108
AES NUMBER (+/-): 30	4189432	49085436395520	1996.44	108
AES NUMBER (+/-): 31	4198894	31391848857600	1970.98	108
AES NUMBER (+/-): 32	4195866	49465943654400	2003.76	108
AES NUMBER (+/-): 33	4258468	45185236992000	2004.48	108
AES NUMBER (+/-): 34	4148774	29108805304320	1970.33	108
AES NUMBER (+/-): 35	4201914	42521686179840	1996.52	108
AES NUMBER (+/-): 36	4248606	41094783959040	2000.2	108
AES NUMBER (+/-): 37	4304972	43663207956480	2002.63	108
AES NUMBER (+/-): 38	4240484	29489312563200	1976.59	108
AES NUMBER (+/-): 39	4305372	33009004707840	1978.42	108
AES NUMBER (+/-): 40	4217966	45851124695040	2002.01	108
AES NUMBER (+/-): 41	4226488	46802392842240	2001.4	108
AES NUMBER (+/-): 42	4189620	25208605900800	1969.16	108
AES NUMBER (+/-): 43	4236524	45470617436160	1997.06	108
AES NUMBER (+/-): 44	4221490	16932573020160	3900.06	110
AES NUMBER (+/-): 45	4258280	25779366789120	1971.52	108
AES NUMBER (+/-): 46	4246064	53651523502080	2003.73	108
AES NUMBER (+/-): 47	4235950	42236305735680	2002.47	108
AES NUMBER (+/-): 48	4278342	64781360824320	1003.52	106
AES NUMBER (+/-): 49	4277002	41380164403200	1996.66	108
AES NUMBER (+/-): 50	4243018	70679223336960	1001.67	106
AES NUMBER (+/-): 51	4232336	29013678489600	1969.73	108
AES NUMBER (+/-): 52	4262608	56410201128960	2022.57	108
AES NUMBER (+/-): 53	4253266	38145852702720	1998.79	108
AES NUMBER (+/-): 54	4325246	39097120849920	1998.62	108
AES NUMBER (+/-): 55	4288150	25684239974400	1972.3	108
AES NUMBER (+/-): 56	4289628	53841777131520	2026.6	108
AES NUMBER (+/-): 57	4290094	38621486776320	2001.52	108
AES NUMBER (+/-): 58	4290872	58978625126400	2027.03	108
AES NUMBER (+/-): 59	4328434	57171215646720	2028.27	108
AES NUMBER (+/-): 60	4280580	42997320253440	2000.18	108
AES NUMBER (+/-): 61	4223286	17313080279040	3901.7	110
AES NUMBER (+/-): 62	4280286	51844114022400	2031.31	108
AES NUMBER (+/-): 63	4269896	42616812994560	2001.98	108
AES NUMBER (+/-): 64	4257638	40333769441280	2002.86	108
AES NUMBER (+/-): 65	4304542	56790708387840	2027.87	108
AES NUMBER (+/-): 66	4263074	39192247664640	1999.28	108
AES NUMBER (+/-): 67	4242876	26920888565760	1971.26	108
AES NUMBER (+/-): 68	4280190	57456596090880	2028.1	108
AES NUMBER (+/-): 69	4271140	71630491484160	1004.53	106
AES NUMBER (+/-): 70	4256298	40809403514880	1999.21	108

AES NUMBER (+/-): 71	4267446	67254658007040	1004.04	106
AES NUMBER (+/-): 72	4298478	42331432550400	2001.25	108
AES NUMBER (+/-): 73	4293794	27016015380480	1973.91	108
AES NUMBER (+/-): 74	4287346	48039041433600	1999.52	108
AES NUMBER (+/-): 75	4257306	18739982499840	3904.46	110
AES NUMBER (+/-): 76	4267308	40238642626560	1997.66	108
AES NUMBER (+/-): 77	4241952	26255000862720	1972.68	108
AES NUMBER (+/-): 78	4335268	60595780976640	1002.17	106
AES NUMBER (+/-): 79	4298172	39763008552960	1998.52	108
AES NUMBER (+/-): 80	4326178	58407864238080	2030.25	108
AES NUMBER (+/-): 81	4300516	40714276700160	2003.26	108
AES NUMBER (+/-): 82	4327422	61356795494400	2030.38	108
AES NUMBER (+/-): 83	4338856	64495980380160	1004.64	106
AES NUMBER (+/-): 84	4298780	74389169111040	1002.39	106
AES NUMBER (+/-): 85	4241486	29679566192640	1970.74	108
AES NUMBER (+/-): 86	4300822	66493643489280	1005.79	106
AES NUMBER (+/-): 87	4280318	38906867220480	2002.2	108
AES NUMBER (+/-): 88	4278174	41475291217920	2003.14	108
AES NUMBER (+/-): 89	4314964	58598117867520	1003.49	106
AES NUMBER (+/-): 90	4241646	65352121712640	1001.59	106
AES NUMBER (+/-): 91	4221448	41475291217920	1995.88	108
AES NUMBER (+/-): 92	4297382	73723281408000	1005.72	106
AES NUMBER (+/-): 93	4281562	61451922309120	2029.32	108
AES NUMBER (+/-): 94	4273490	45755997880320	2002.79	108
AES NUMBER (+/-): 95	4277868	57361469276160	2028.12	108
AES NUMBER (+/-): 96	4233710	29489312563200	1971.53	108
AES NUMBER (+/-): 97	4229026	18454602055680	3905.33	110
AES NUMBER (+/-): 98	4224954	38716613591040	2000.18	108
AES NUMBER (+/-): 99	4251712	27491649454080	1974.74	108
AES NUMBER (+/-): 100	4204916	42046052106240	2001.16	108

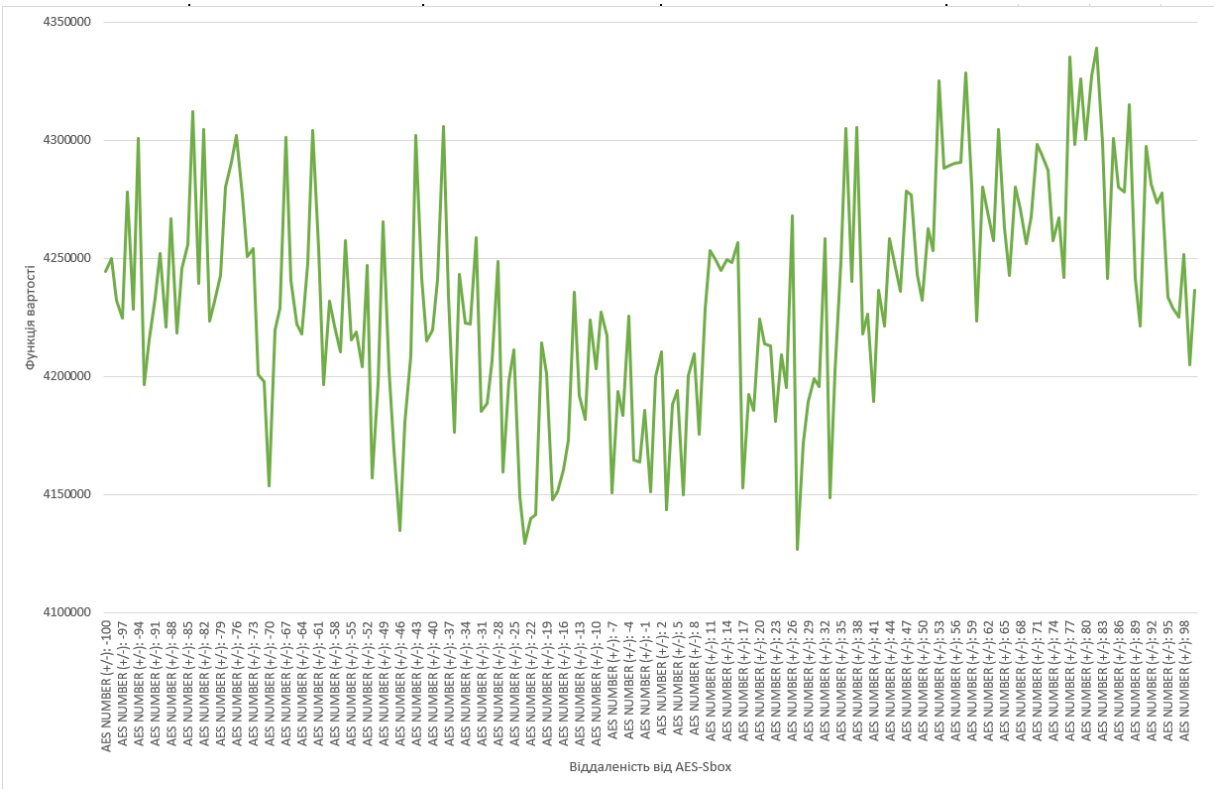

Рис. 8. Графік вартості блок Графік вартості блоків у просторі навколо AES-Sbox для цінової функції WHS

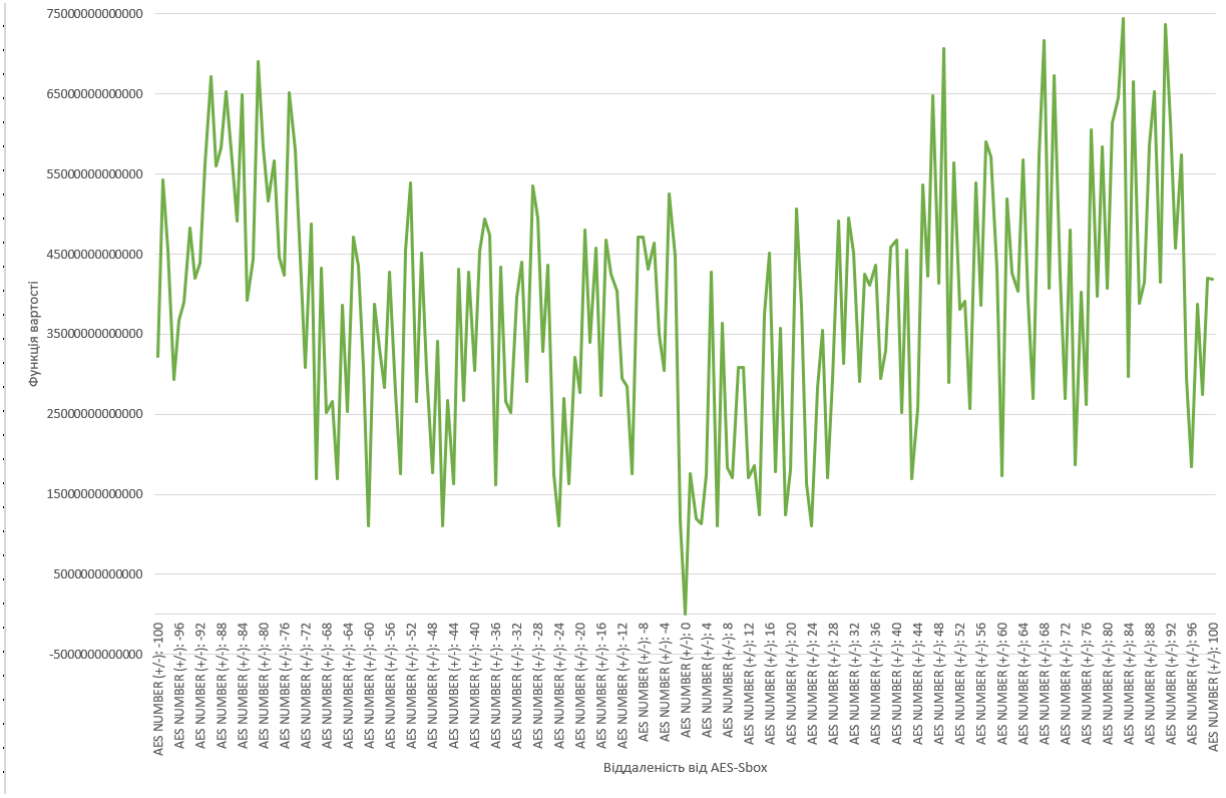

Рис. 9. Графік вартості блоків у просторі навколо AES-Sbox для цінової функції WCF

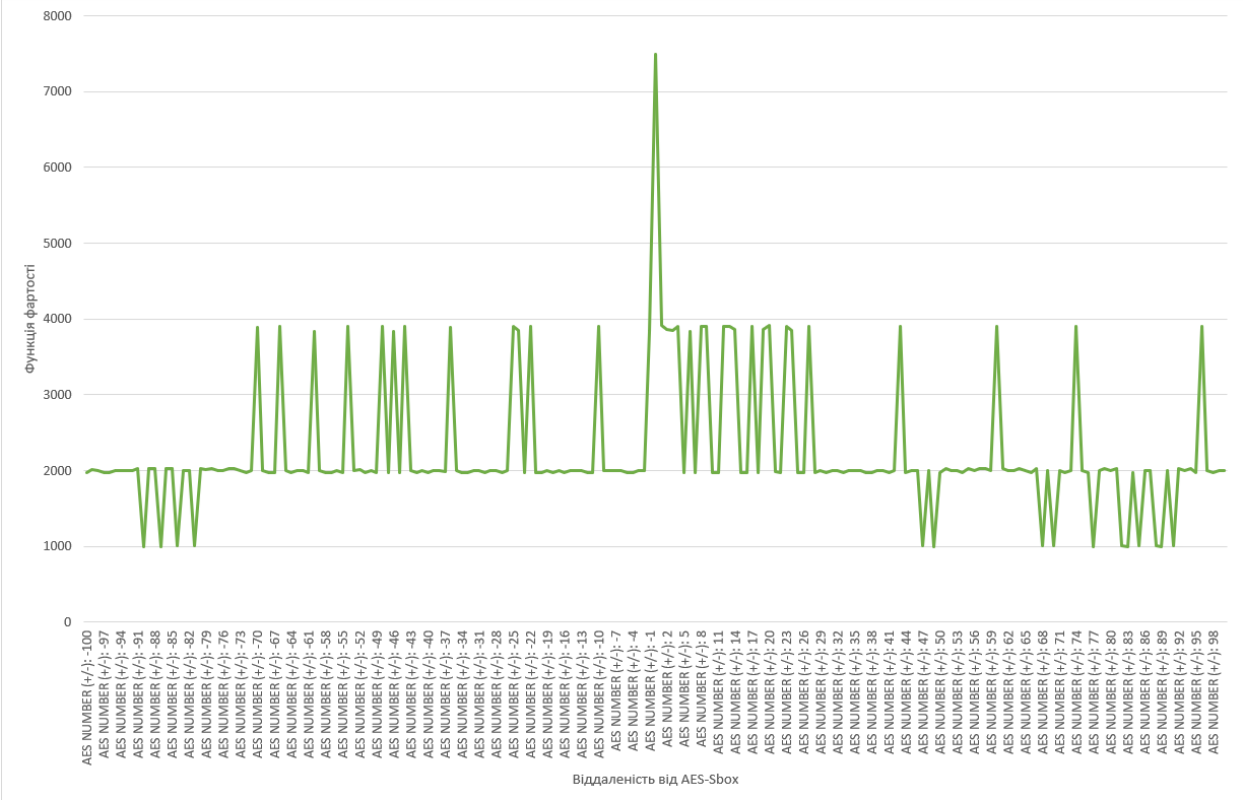

Рис. 10. Графік вартості блоків у просторі навколо AES-Sbox для цінової функції PCF

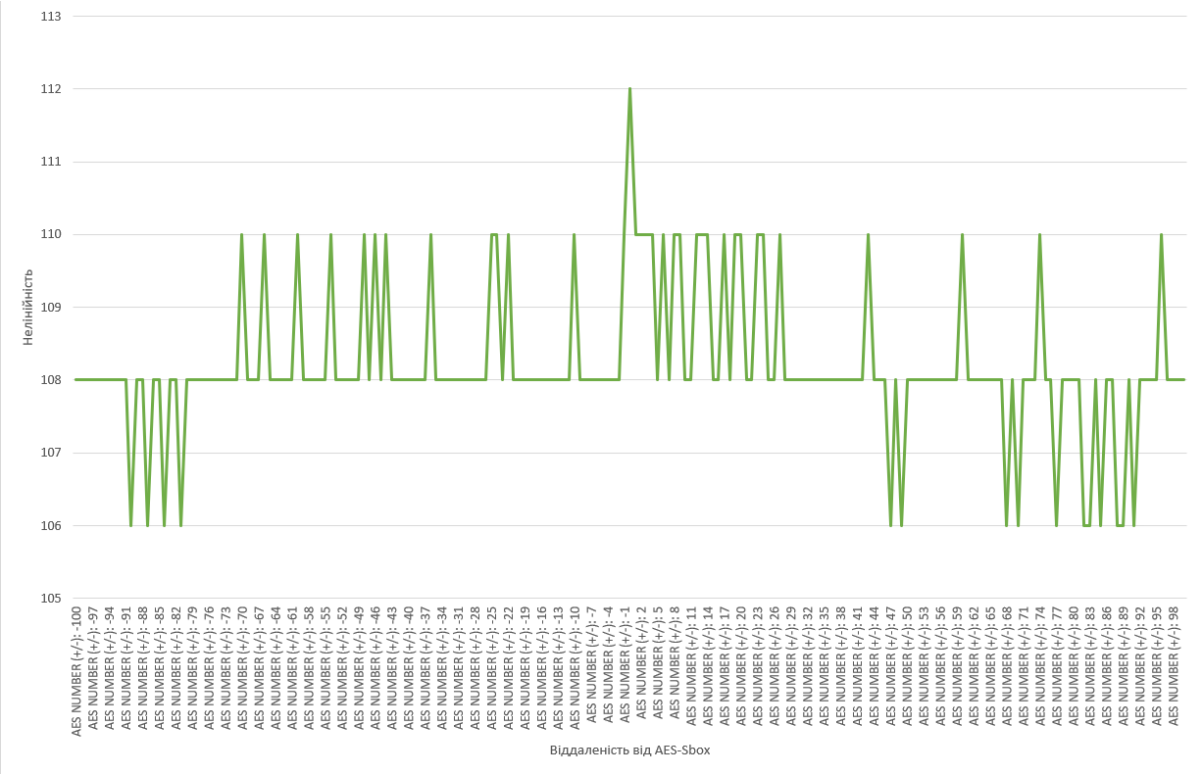

Рис. 11. Графік нелінійності простору блоків навколо AES-Sbox

Як видно з табл. 10 та графіків, простір навколо AES містить у собі блоки з дуже гарними показниками нелінійності, проте всі інші показники, такі як, наприклад, алгебраїчний імунітет, у цих блоків не дуже хороші. Також з результатів видно, що нелінійності блоків несуть в собі дуже малу кількість інформації щодо цінності кожного з блоків, проте при використанні цінових функцій, чітко помітно різницю між «хорошим» та «поганим» блоком, тобто локальні мінімуми чи максимуми (в залежності від функції), в яких і знаходяться «хороші» блоки. Отже, використання цінових функцій має значно покращити пошукові можливості алгоритму, оскільки алгоритм більш чітко «бачитиме» місця, в яких знаходяться «хороші» і «погані» блоки. Це означатиме, що при пошуку він не буде відволікатися на блоки, які є гарними тільки за нелінійністю, тобто враховуватиме тільки блоки, які є кращими за інші за повним спектром, а не тільки за найбільшим з його коефіцієнтів. За результатами тестування можна сказати, що найбільш оптимальною буде функція WHS, оскільки в ній найбільше виражені мінімуми та максимуми простору.

Висновки

1. Оскільки робота з перестановками є важливою задачею в багатьох сферах сучасної діяльності, застосування факторіальної системи числення з метою доступу до певної перестановки за її номером або навпаки отримання номеру за її значеннями є доволі важливою задачею.

2. Детально розглянуто та пояснено загальні принципи використання факторіальної системи числення для роботи з перестановками. В ній пропонуються алгоритми переведення номеру перестановки у її значення, а також зворотного переведення мовою програмування С. Також досліджуються нелінійні перестановки, що використовуються в алгоритмах ГОСТ 28147-89 та AES. Практично отримано номери цих перестановок з використанням факторіальної СЧ, визначено параметри даних блоків, а також надано результати вимірювання швидкодії операцій переходу від перестановок до номерів і зворотних перетворень.

3. Проведено дослідження простору блоків поруч з AES-Sbox, що знаходяться поруч із ним. Це може допомогти в подальшому краще розуміти простір блоків та оптимізувати пошуки як запропонованим, так і іншими алгоритмами пошуку.

4. Визначено показники блоків у просторі – нелінійності та функції вартості. Результати дослідження підтвердили, що простір навколо AES не є оптимальним пошуковим місцем, оскільки блоки тут є «хорошими» тільки за показником нелінійності, тому алгоритм слід адаптувати виключно під випадковий пошук, а не на основі якихось уже відомих блоків.

5. Для ефективного пошуку необхідно застосовувати функцію вартості. Це дозволить в процесі пошуку алгоритму зациклюватися тільки на блоках з повністю «хорошим» спектром, відкидаючи блоки з тільки одним «хорошим» коефіцієнтом.

Список літератури:

1. Програмна реалізація Факторіальної системи числення від Дерев'янка Я. А. URL: https://github.com/DereviankoYaroslav/SBoxDereviankoFactorial.

2. Програмна реалізація алгоритму PSO з використанням факторіальної системи числення та цінової функції WHS від Дерев'янка Я. А. URL: https://github.com/DereviankoYaroslav/FactorialPSO-WHS.

3. Програмна реалізація алгоритму PSO з використанням факторіальної системи числення та цінової функції WCF від Дерев'янка Я. А. URL: https://github.com/DereviankoYaroslav/FactorialPSO-CUBA.

4. Alejandro Freyre-Echevarr´ıa, Ismel Mart´ınez-D´ıaz. A new cost function to improve nonlinearity ofbijective S-boxes. URL: https://www.researchgate.net/publication/343699912_A_new_cost_function_to_improve_nonlinearity_of_bijective_S-boxes.

5. Alexandr Kuznetsov, Luca Romeo, Nikolay Poluyanenko, Sergey Kandiy, Kateryna Kuznetsova. Optimizing Hill Climbing Algorithm Parameters for Generation of Cryptographically Strong S-Boxes. URL: https://assets.researchsquare.com/files/rs-1657863/v1_covered.pdf?c=1653408505.

6. John A. Clark, Jeremy L. Jacob, Susan Stepney. The Design of S-Boxes by Simulated Annealing. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.123.7114&rep=rep1&type=pdf.

Надійшла до редколегії 15.02.2022

Відомості про авторів:

Дерев'янко Ярослав Андрійович – студент кафедри безпеки інформаційних систем і технологій факультету комп'ютерних наук, Харківський національний університет імені В.Н. Каразіна, Україна; e-mail: yarik0009258@gmail.com; ORCID: https://orcid.org/0000-0002-3290-3373

Горбенко Юрій Іванович – канд. техн. наук, Харківський національний університет імені В.Н. Каразіна, старший науковий співробітник кафедри безпеки інформаційних систем і технологій, факультет комп'ютерних наук; Україна; e-mail: gorbenkou@iit.kharkov.ua, ORCID: https://orcid.org/0000-0002-0652-8629

Кузнецов Олександр Олександрович – д-р техн. наук, професор, Харківський національний університет імені В.Н. Каразіна, професор кафедри безпеки інформаційних систем і технологій, факультет комп'ютерних наук; Україна; e-mail: kuznetsov@karazin.ua, ORCID: https://orcid.org/0000-0003-2331-6326