WEB-СКМ SAGE У ЗАДАЧАХ ТЕОРІЇ КОДУВАННЯ

О.П. Поліщук[α], С.В. Шокалюк[β], І.С. Закарлюка
Україна, м. Кривий Ріг, Криворізький державний педагогічний університет
[α] apol@cabletv.dp.ua
[β] ksv_ipm@mail.ru

Новий «системний інтегратор» SAGE [1] має інтерфейси для підключення як широко поширених пакетів загального призначення Maple, Mathematica, Matlab, Maxima (так званих «великих *M*»), так і для спеціалізованих систем. Одна з них – GAP, ПЗ для алгебраїчних досліджень – включає в себе пакет GUAVA, що реалізує в собі основні об'єкти та методи теорії кодування. Застосування цього пакету дозволяє частково розв'язати проблему практичної підтримки курсу «Теорія кодування» у випадку відсутності лабораторних занять.

Розглянемо загальні функції теорії кодування, що надає SAGE.

1. Клас `LinearCode` та функція `LinearCodeFromVectorspace`.

```
sage: MS = MatrixSpace(GF(2),4,7)
sage: G = MS([[1,1,1,0,0,0,0], [1,0,0,1,1,0,0],
              [0,1,0,1,0,1,0], [1,1,0,1,0,0,1]])
sage: C = LinearCode(G) sage: C Linear code of length 7, dimension 4 over
Finite Field of size 2
sage: C.base_ring()
Finite Field of size
sage: C.dimension()
4
sage: C.length()
7
sage: C.minimum_distance()
3
sage: C.spectrum()
[1, 0, 0, 7, 7, 0, 0, 1]
sage: C.weight_distribution()
[1, 0, 0, 7, 7, 0, 0, 1]
```

Наведемо приклад застосування для створення власної кодової функції, що повертає ерміттів гексакод [6, 3, 4] типу IV над GF(4):

```
def Hexacode():
  F = GF(4,"z")
  z = F.gen()
  MS = MatrixSpace(F, 3, 6)
  G = MS([[1, 0, 0, 1, z, z ],\
          [0, 1, 0, z, 1, z ],\
          [0, 0, 1, z, z, 1 ]])
  return LinearCode(G)
```

2. spectrum (ваговий розподіл), minimum_distance, characteristic_function та кілька реалізації дзета-функції Івана Дуурсма (zeta_polynomial, zeta_function, chinen_polynomial):

```
sage: C = HammingCode(3,GF(2))
sage: C.zeta_polynomial()
2/5*T^2 + 2/5*T + 1/5
sage: C = best_known_linear_code(6,3,GF(2))
sage: C.minimum_distance()
3
sage: C.zeta_polynomial()
2/5*T^2 + 2/5*T + 1/
```

3. gen_mat, check_mat, decode, dual_code, extended_code, binomial_moment для класу лінійних кодів:

```
sage: C = HammingCode(3,GF(2))
sage: C.binomial_moment(2)
0
sage: C.binomial_moment(3)
0
sage: C.binomial_moment(4)
35
sage: C = HammingCode(3,GF(2))
sage: MS = MatrixSpace(GF(2),1,7)
sage: F = GF(2); a = F.gen()
sage: v1 = [a,a,F(0),a,a,F(0),a]
sage: C.decode(v1)
(1, 0, 0, 1, 1, 0, 1)
```

4. Предикати is_self_orthogonal, is_permutation_automorphism, "==", is_self_dual.

5. Функції перестановки: standard_form, automorphism_group_binary_code, is_permutation_automorphism, module_composition_factors:

```
sage: C = HammingCode(3,GF(2))
sage: G = C.automorphism_group_binary_code(); G
Permutation Group with generators [(2,3)(5,7), (2,5)(3,7),
                                   (2,3,7,5)(4,6), (2,4)(6,7), (1,2)(3,4)]
sage: G.order()
168
sage: C = HammingCode(3,GF(2))
sage: C.gen_mat()
[1 0 0 1 0 1 0]
[0 1 0 1 0 1 1]
[0 0 1 1 0 0 1]
[0 0 0 0 1 1 1]
sage: C.redundancy_matrix()
[1 1 0]
[1 1 1]
[1 0 1]
```

```
[0 1 1]
sage: C.standard_form()[0].gen_mat()
[1 0 0 0 1 1 0]
[0 1 0 0 1 1 1]
[0 0 1 0 1 0 1]
[0 0 0 1 0 1 1]
sage: MS = MatrixSpace(GF(2),4,8)
sage: G = MS([[1,0,0,0,1,1,1,0],[0,1,1,1,0,0,0,0],
              [0,0,0,0,0,0,0,1],[0,0,0,0,0,1,0,0]])
sage: C = LinearCode(G)
sage: gp = C.automorphism_group_binary_code()
sage: C.module_composition_factors(gp)
[ rec(
  field := GF(2),
  isMTXModule := true,
  dimension := 1,
  generators := [ [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ],
                [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ] ],
  smashMeataxe := rec(
      algebraElement :=
        [ [ [ 5, 3 ], [ 5, 3 ] ], [ Z(2)^0, Z(2)^0, 0*Z(2), Z(2)^0,
                  0*Z(2), Z(2)^0, Z(2)^0, Z(2)^0 ] ],
      algebraElementMatrix := [ [ 0*Z(2) ] ],
      characteristicPolynomial := x_1,
      charpolFactors := x_1,
      nullspaceVector := [ Z(2)^0 ],
      ndimFlag := 1 ),
  IsIrreducible := true ), rec(
  field := GF(2),
  isMTXModule := true,
  dimension := 1,
  generators := [ [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ],
                [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ] ],
  smashMeataxe := rec(
      algebraElement := [ [ [ 5, 2 ], [ 1, 2 ] ], [ 0*Z(2), 0*Z(2), 0*Z(2),
                          0*Z(2), Z(2)^0, 0*Z(2), Z(2)^0, 0*Z(2) ] ],
      algebraElementMatrix := [ [ 0*Z(2) ] ],
      characteristicPolynomial := x_1,
      charpolFactors := x_1,
      nullspaceVector := [ Z(2)^0 ],
      ndimFlag := 1 ),
  IsIrreducible := true ), rec(
  field := GF(2),
  isMTXModule := true,
  dimension := 1,
  generators := [ [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ],
                [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ] ],
  smashMeataxe := rec(
      algebraElement := [ [ [ 4, 2 ], [ 7, 4 ] ], [ 0*Z(2), Z(2)^0, Z(2)^0,
```

```
                          0*Z(2), Z(2)^0, Z(2)^0, Z(2)^0, Z(2)^0 ] ],
      algebraElementMatrix := [ [ 0*Z(2) ] ],
      characteristicPolynomial := x_1,
      charpolFactors := x_1,
      nullspaceVector := [ Z(2)^0 ],
  ndimFlag := 1 ),
  IsIrreducible := true ), rec(
  field := GF(2),
  isMTXModule := true,
  dimension := 1,
  generators := [ [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ],
               [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ] ],
  smashMeataxe := rec(
      algebraElement := [ [ [ 4, 6 ], [ 1, 6 ] ], [ 0*Z(2), Z(2)^0, Z(2)^0,
                        0*Z(2), Z(2)^0, 0*Z(2), Z(2)^0, Z(2)^0 ] ],
      algebraElementMatrix := [ [ Z(2)^0 ] ],
      characteristicPolynomial := x_1+Z(2)^0,
      charpolFactors := x_1+Z(2)^0,
      nullspaceVector := [ Z(2)^0 ],
  ndimFlag := 1 ),
  IsIrreducible := true ) ]
```

6. `assmus_mattson_designs` (реалізація теореми Ассмуса-Меттсона):

```
sage: C = ExtendedBinaryGolayCode() # приклад 1
sage: C.assmus_mattson_designs(5)
['weights from C: ',
[8, 12, 16, 24],
'designs from C: ',
[[5, (24, 8, 1)], [5, (24, 12, 48)], [5, (24, 16, 78)], [5, (24, 24, 1)]],
'weights from C*: ',
[8, 12, 16],
'designs from C*: ',
[[5, (24, 8, 1)], [5, (24, 12, 48)], [5, (24, 16, 78)]]]
sage: C.assmus_mattson_designs(6)
0
sage: X = range(24) # приклад 2
sage: blocks = [c.support() for c in C if hamming_weight(c)==8]
sage: len(blocks)
759
```

Основні класи кодів, що надає SAGE: БЧХ (`BCHCode`), Хеммінга (`HammingCode`), Уолша (`WalshCode`), Ріда-Соломона (`ReedSolomonCode`), Голея (`BinaryGolayCode`, `ExtendedBinaryGolayCode`, `ExtendedTernaryGolayCode`, `TernaryGolayCode`), торичні (`ToricCode`), дуадічні (`DuadicCodeEvenPair`, `DuadicCodeOddPair`), лінійні (`LinearCodeFromCheckMatrix`, `RandomLinearCode` та ін.), квадратичні лишкові (`QuadraticResidueCodeEven[Odd]Pair`), циклічні (`CyclicCodeFromGeneratingPolynomial`, `CyclicCodeFromCheckPolynomial`).

Література:

1. Stein, W. Sage Reference Manual. – 2008. – XII+3460 p.