ALGORITHMS FOR THE MANUSCRIPT "A NOVEL APPROACH TO CODES WITH THE MANHATTAN DISTANCE"

E. J. GARCÍA-CLARO AND ISMAEL GUTIERREZ ‡

Abstract. This document presents several SageMath implementations useful to apply the main results of the manuscript entitled "A novel approach to codes with the Manhattan distance".

Algorithms to compute $\eta_r(\mathfrak{G})$ and $\gamma_r(\mathfrak{G})$

Algorithm 1 provides the code in *SageMath* to calculate $\eta_r(\mathfrak{G})$.

To use the *Python* functions given in Algorithms 1, 2, 3, and 4, one can import them into a SageMath worksheet by writing `load("GitHub_GridCodes.py")`[1] at the beginning of the worksheet.

Data: `r` $\in \mathbb{Z}_{\geq 0}$ and the *Python* list `m` $= [m_1, ..., m_n]$ where $\mathfrak{G} = \prod_{i=1}^{n}[0, m_i - 1]$.
Result: `eta(r, m)` computes $\eta_r(\mathfrak{G})$.

```
def eta(r, m):
    if r==0:
        return 1

    n=len(m); indices = range(1, n+1); eta = 0

    for delta in range(r + 1):
        for J in Subsets(indices):
            sum_m_J = sum(m[i-1] for i in J)
            if delta - sum_m_J >= 0:
                binom_term = binomial(n + delta - sum_m_J - 1
                           , delta - sum_m_J)
                eta += (-1) ** len(J) * binom_term

    return eta
```

Algorithm 1: A *Python* function that applies Theorem 5.6 to compute $\eta_r(\mathfrak{G})$ in *SageMath*.

Let $x \in Inm(\mathfrak{G})$, $n \in \mathbb{Z}_{\geq 1}$, $\delta \in \mathbb{Z}_{>0}$, p_δ be the coefficient of degree δ of $p(t, x)$. The function in Algorithm 4 uses the functions given in Algorithms 2, 3 to provide the *SageMath* code to calculate

Date: Oct 2024.

[1]The document `"GitHub_GridCodes.py"` can be donwloaded from https://github.com/egarcia-claro/GridCodes

p_δ for $\mathfrak{G} = \prod_{i=1}^{n}[0, m_i - 1]$ when $E = \{i \in [n] : 2 \mid m_i\}$ is arbitrary.

Data: `delta` $\in \mathbb{Z}_{\geq 0}$ and the *Python* list `m` $= [m_1, ..., m_n]$ where $\mathfrak{G} = \prod_{i=1}^{n}[0, m_i - 1]$.
Result: `p_delta_o(delta, m)` computes the coefficient p_δ of degree δ of $p(t, \text{any innermost point})$ when $E = \{i \in [n] : 2 \mid m_i\} = \emptyset$.

```
def p_delta_o(delta, m):
    if delta==0:
        return 1

    n=len(m); l = [ceil((mi - 1) / 2) for mi in m]
    indices=range(1, n+1)
    E = {i for i in indices if m[i-1] % 2 == 0}
    partial=sum(li for li in l)
    p_delta = 0

    if len(E) != 0:
        raise ValueError("The m_i's must be odd")

    elif delta == partial:
        return 2**n

    P_n= Subsets(indices)
    for J in P_n:
        if len(J) > 0:
            for A in Subsets(J):
                sum_l_A = sum(l[i-1] + 1 for i in A)
                if delta - sum_l_A >= 0:
                    binom_term = binomial(len(J) + delta - sum_l_A
                                 - 1, delta - sum_l_A)
                    p_delta += (-1)**(n - len(J) + len(A))
                             * 2**len(J) * binom_term

    return p_delta
```

Algorithm 2: An algorithm that applies Theorem 5.8 (part 1) to compute p_δ in *SageMath*.

Data: `delta` $\in \mathbb{Z}_{\geq 0}$ and the *Python* list `m` $= [m_1, ..., m_n]$ where $\mathfrak{G} = \prod_{i=1}^{n} [0, m_i - 1]$.
Result: `p_delta_o(delta, m)` computes the coefficient p_δ of degree δ of $p(t,$ any innermost point) when $E = \{i \in [n] : 2 \mid m_i\} = [n]$.

```
def p_delta_e(delta, m):
    if delta==0:
        return 1

    n=len(m); l = [ceil((mi - 1) / 2) for mi in m]
    indices=range(1, n+1)
    E = {i for i in indices if m[i-1] % 2 == 0}
    partial=sum(li for li in l)
    p_delta = 0

    if len(E) != n:
        raise ValueError("The m_i’s must be even")

    if delta == partial:
        return 1

    P_n = Subsets(indices)
    for J in P_n:
        if len(J) > 0:  # Ensure J is non-empty
            J_c = set(indices) - set(J)
            u_J_delta = (-1)**(len(J)) if delta ==
                        sum(l[i-1] for i in J_c) else 0

            term_sum = u_J_delta

            for A in Subsets(J):
                if len(A) > 0:
                    for B in Subsets(A):
                        sum_l_B_J_c = sum(l[i-1] for i in B)
                                    + sum(l[i-1] for i in J_c)
                        if delta - sum_l_B_J_c >= 0:
                            binom_term = binomial(len(A) + delta
                                       - sum_l_B_J_c - 1, delta
                                       - sum_l_B_J_c)
                            term_sum += (-1)**(len(J) - len(A)
                                      + len(B))  *  2**len(A)
                                      *  binom_term

            p_delta += term_sum

    return p_delta
```

Algorithm 3: An algorithm that applies Theorem 5.8 (part 2) to compute p_δ in *SageMath*.

Data: `delta` $\in \mathbb{Z}_{\geq 0}$ and the *Python* list `m` $= [m_1, ..., m_n]$ where $\mathfrak{G} = \prod_{i=1}^{n}[0, m_i - 1]$.
Result: `p_delta(delta, m)` computes the coefficient p_δ of degree δ of $p(t, \text{any innermost point})$ when $E = \{i \in [n] : 2 \mid m_i\}$ is arbitrary.

```
def p_delta(delta, m):
    if delta==0:
        return 1

    n=len(m); indices=range(1, n+1)
    m_e=[m[i-1] for i in indices if m[i-1] % 2 == 0]
    partial_e=sum(ceil((m - 1) / 2) for m in m_e)
    m_o=[m[i-1] for i in indices if m[i-1] % 2 != 0]
    p_delta = 0

    return sum(p_delta_e(j, m_e) *  p_delta_o(delta-j, m_o)
               for j in range(partial_e + 1))
```

Algorithm 4: An algorithm that applies Theorem 5.8 (part 3) to compute p_δ in *SageMath*. It works even when $E = \emptyset$ or $E = [n]$, because `p_delta_e(0, [])=p_delta_o(0, [])=1`.

Data: `r` $\in \mathbb{Z}_{\geq 0}$ and the *Python* list `m` $= [m_1, ..., m_n]$ where $\mathfrak{G} = \prod_{i=1}^{n}[0, m_i - 1]$.
Result: `gamma(r, m)` computes $\gamma_r(\mathfrak{G})$.

```
def gamma(r, m):
    if r==0:
        return 1

    return 1 + sum(p_delta(delta, m) for delta in range(1, r+1))
```

Algorithm 5: Since $\gamma_r(\mathfrak{G}) = p(1, x)_{\leq r} = 1 + \sum_{\delta=1}^{r} p_\delta$ (by Lemma 5.1 and Theorem 5.4) and $p_\delta =$ `p_delta(delta, m)` (where `p_delta(delta, m)` is given as in Algorithm 4), this function computes $\gamma_r(\mathfrak{G})$.

Departamento de Matemáticas, Universidad Autónoma Metropolitana, Unidad Iztapalapa, Código postal 09340, Ciudad de México - México
Email address: eliasjaviergarcia@gmail.com

Department of Mathematics and Statistics, Universidad del Norte, Km 5 via a Puerto Colombia, Barranquilla - Colombia
Email address: isgutier@uninorte.edu.co