# Algoritmo memético paralelo: Avance II

Pablo Zapata

# CSP: Closest string problem

- Encontrar el centro geométrico de un set de strings.
- En palabras más simples, encontrar una string donde la distancia hamming entre ella y cada una de un conjunto sea de a lo más x. Objetivo: minimizar x.
- NP-Hard.
- Útil para encontrar señales en secuencias de ADN.

# CSP: Closest string problem

Instancia: C = {$s_1$, $s_2$, ..., $s_n$}, n strings de tamaño m sobre un alfabeto A.

Objetivo: Encontrar una string x de tamaño m sobre A tal que $d_H(x, s_i) \leq d_c$ para todo $s_i$ en C, donde $d_c$ es mínimo.

C = {ACGT, TTAC, CCGC}, x = TCGC

$d_H(x, s_1) = 2$ $d_H(x, s_2) = 2$ $d_H(x, s_3) = 1$

# Avances

Cálculo de la distancia hamming usando cuda.

```
TAATCGT ... GCGGCCC
TCTTCGG ... ACCCATA
0110001 ... 1011111
```

Reducción

```
__global__ void kernelHamming(char *s1, char *s2, int *dist){
    extern __shared__ int temp[];
    int offset = blockIdx.x*blockDim.x;
    thread_group g = this_thread_block();

    int tid = g.thread_rank();

    if(s1[tid + offset] != s2[tid + offset]) temp[tid] = 1;
    else temp[tid] = 0;
    g.sync();

    for(int i=512; i>0; i/=2){
        if(tid < i && i < blockDim.x) temp[tid] += temp[tid + i];
        g.sync();
    }

    if(threadIdx.x == 0) atomicAdd(dist, temp[0]);
}
```

# Avances

```
void procesarCUDA(){
    datasetDst = vector<char*>(n);
    dist = vector<int*>(n);

    cudaMalloc(&solDst, m*sizeof(char));
    for(int i=0; i<n; i++){
        cudaMalloc(&datasetDst[i], m*sizeof(char));
        cudaMallocHost(&dist[i], sizeof(int));
    }

    stream = vector<cudaStream_t>(n);
    for(int i=0; i<n; i++) cudaStreamCreate(&stream[i]);

    bloques = m/1000;
    hebras = 1000;
    sharedBytes = hebras * sizeof(int);

    for(int i=0; i<n; i++)
        cudaMemcpyAsync(datasetDst[i], &dataset[i][0], m*sizeof(char), cudaMemcpyHostToDevice, stream[i]);
}
```

```
vector<string> dataset;
char *solDst;
vector<char*> datasetDst;
vector<int*> dist;
vector<cudaStream_t> stream;
int bloques, hebras, sharedBytes;
```

# Avances

```
vector<int> calidadCUDA(string sol){
    cudaMemcpyAsync(solDst, &sol[0], m*sizeof(char), cudaMemcpyHostToDevice);
    for(int i=0; i<n; i++) *dist[i] = 0;

    for(int i=0; i<n; i++) kernelHamming<<<bloques, hebras, sharedBytes, stream[i]>>>(solDst, datasetDst[i], dist[i]);
    cudaDeviceSynchronize();

    vector<int> distHamming(n);
    for(int i=0; i<n; i++) distHamming[i] = *dist[i];

    return distHamming;
}
```

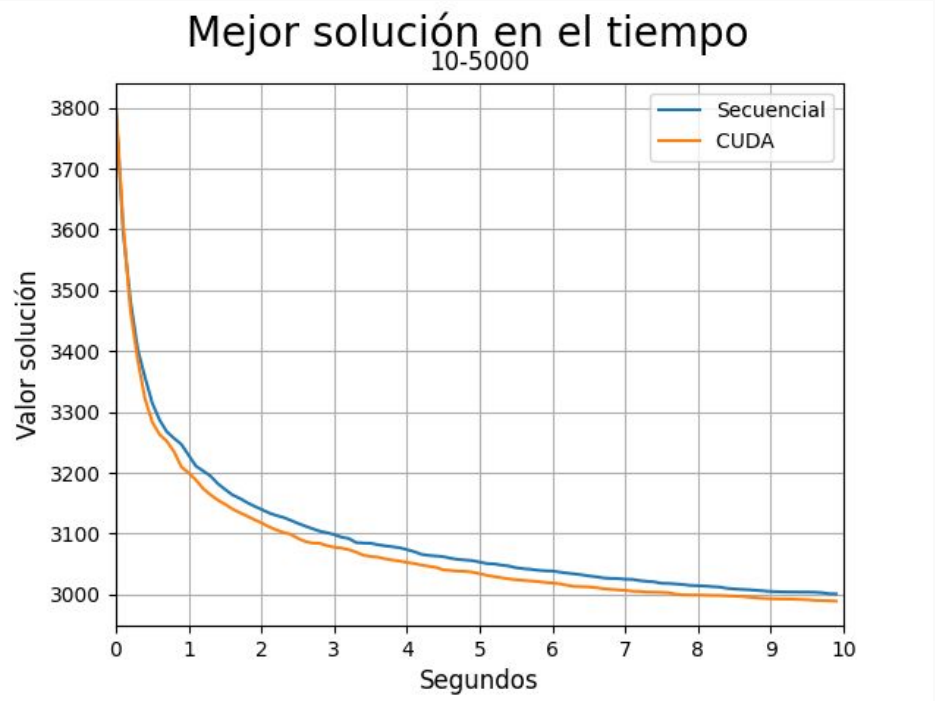
# Análisis experimental

- Cálculo de distancia hamming secuencial vs CUDA.

| | Milisegundos | |
|---|---|---|
| | Secuencial | CUDA |
| 10-1000 | 0.161 | 0.087 |
| 10-2000 | 0.323 | 0.083 |
| 10-3000 | 0.487 | 0.084 |
| 10-4000 | 0.645 | 0.085 |
| 10-5000 | 0.809 | 0.083 |

| | Milisegundos | |
|---|---|---|
| | Secuencial | CUDA |
| 30-1000 | 0.492 | 0.218 |
| 30-2000 | 0.971 | 0.219 |
| 30-3000 | 1.462 | 0.22 |
| 30-4000 | 1.938 | 0.206 |
| 30-5000 | 2.427 | 0.205 |

# Análisis experimental

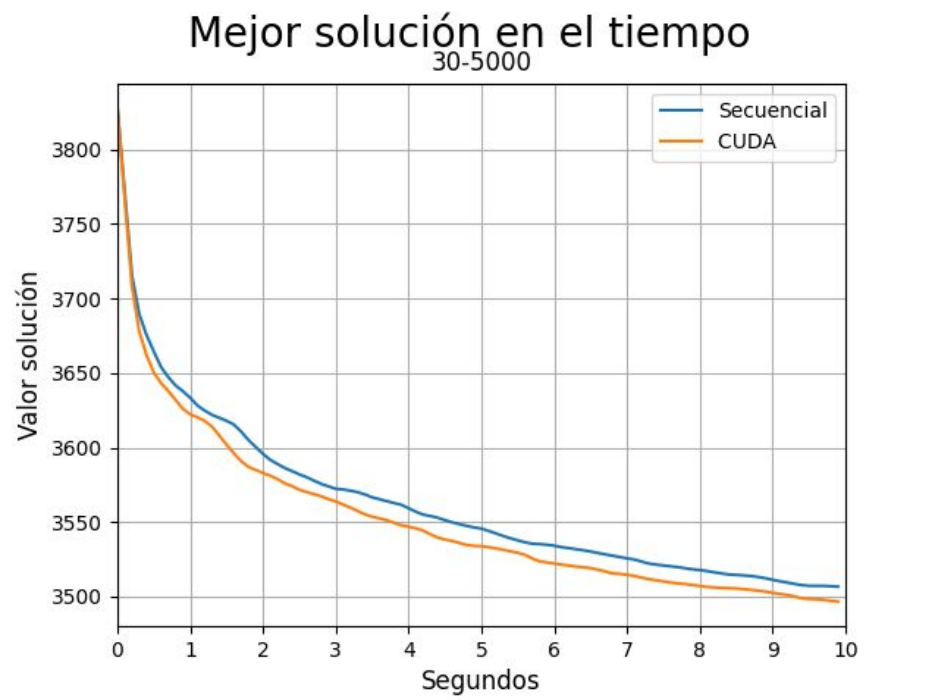



n: 10
m: 5000

| Calidad | Tiempo (Segundos) | |
|---|---|---|
| | Secuencial | CUDA |
| 3200 | 1.24 | 1 |
| 3100 | 2.95 | 2.36 |
| 3000 | 10 | 7.73 |

Reducción de tiempo: 20.7%

# Análisis experimental




n: 10
m: 5000

| Calidad | Tiempo (Segundos) | |
|---|---|---|
| | Secuencia | CUDA |
| 3600 | 1.91 | 1.53 |
| 3550 | 4.58 | 3.78 |
| 3510 | 9.13 | 7.53 |

Reducción de tiempo: 18.3%

# Análisis teórico

**Calculo distancia hamming secuencial:**
O(n*m)

**Calculo distancia hamming CUDA:**
O(n + b)

p: población

| **Pseudocódigo** | **Secuencial** | **CUDA** |
|---|---|---|
| crearPoblacion() | O(p*n*m) | O(p*n*m) |
| while(condiciones): | | |
| crearAntibiotico() | O(1) | O(1) |
| mutacion() | O(p*m) | O(p*m) |
| busquedaLocal() | O(p*n*m) | O(p*n*m) |
| evaluarBacterias() | O(p*n*m) | O(p) * O(n + b) |
| clasificacion() | O(p) | O(p) |
| actualizarFeromonas() | O(m) | O(m) |
| administrarAntibiotico() | O(p) | O(p) |
| Complejidad del ciclo: | O(p*n*m) | O(p*n*m) |

# Planificación

- Entrega final: Asignación de cada hilo a un grupo de la población junto a la integración del avance 2.

- Cada entrega tendrá análisis teórico y experimental.