Proyecto 1 Sistemas Adaptativos:

Heurística Greedy para FFMSP

Profesor Pedro Pinacho
Tomás Ying-Kit Contreras Kong - 2021420797
Constanza Fabiola Cristinich Ananías - 2021423184
Esteban Andres Chandia Cifuentes - 2019900862

Solución para FFMSP Determinístico y No-Deterministico

Algoritmo:

La ejecución del programa recibe como parámetros el nombre del archivo con el dataset, el threshold (umbral) y un valor que puede ser 0 o 1 (0 ejecuta el código determinista y 1 lo ejecuta aleatorizado).

Se guardan los strings del archivo en una lista de strings *"lista"* , declarando
m: Largo de los strings (todos poseen el mismo largo)
n: Cantidad original de strings en la lista.
dHam: Ya que el umbral se entrega como un flotante entre 0 y 1, multiplicamos el umbral por el largo del string para obtener la distancia de Hamming que se pretende igualar o superar.
solución: string solución inicializado en "", creado por el algoritmo.

Mientras el tamaño del string solución sea menor a dHam, el algoritmo itera sobre la posición i de cada string, para obtener la frecuencia de cada carácter, haciendo uso de un string contador inicializado en 0 y reseteado luego de moverse a la siguiente posición. Luego de terminar la iteración, se elige el carácter con menor frecuencia en el contador y se carga al string solución.

Cuando el tamaño del string solución es mayor o igual a dHam, se realiza una comparación entre el string solución y todos los strings en la lista.
Si la distancia de Hamming entre los strings comparados es menor a dHam, se continúa al siguiente string.
Si la distancia de Hamming entre los strings comparados es igual o mayor a dHam, se elimina de la lista y se añade a una lista de strings solución.
Una vez que se compara con toda la lista, se vuelve a elegir el carácter con menor frecuencia, se añade al string solución y se vuelve a comparar.

Cuando la lista está vacía y el tamaño del string solución es menor a m, se rellena con caracteres aleatorios.
Cuando el tamaño del string solución es igual a m y ya se comparó con todos los demás strings en la lista, el algoritmo entrega: el string solución creado, la calidad de la solución, la cantidad de strings que no cumplieron con el umbral y la cantidad de strings que sí lo hicieron, además del tiempo de demora del algoritmo.

Pseudocódigo:

```
boolean evaluar(list<string> lista, string solucion, char x, int dHam){
    aux = concatenar(solucion, x)
    para cada string en lista:
        diferencia = 0
        para j desde 0 hasta el tamaño de aux:
            si aux[j] es diferente de string[j]:
                incrementar diferencia
        si diferencia es mayor o igual a dHam:
            retornar verdadero
    retornar falso
}

int main(argc, argv){
    umbral = convertir a flotante(argv[4])
    modo = convertir a entero(argv[5])
    si argc != 5 o argv[1] != "-i" o argv[3] != "-th" o umbral fuera de rango [0, 1] o modo
!= 0 y != 1:
        retornar ERROR

    nombreArchivo = argv[2]
    lista = lista vacía
    obtener_strings(nombreArchivo, lista)
    n = cantidad_de_strings(nombreArchivo)
    m = largo_de_strings(nombreArchivo)
    solucion = string vacío
    dHam = m * umbral
    conjuntoSolucion = lista vacía
    int contador[4] = {0,0,0,0}

    tiempo = 0
    auto start = high_resolution_clock::now();

    bases = {'A', 'C', 'G', 'T'};

    para i desde 0 hasta m:
        para cada string en lista:
            caracter = obtener_caracter_en_posicion(string, i)
            segun el_caracter:
                caso 'A': incrementar contador[0]
                caso 'C': incrementar contador[1]
                caso 'G': incrementar contador[2]
                caso 'T': incrementar contador[3]
```

```
        caracteres_validos = mapa vacío

        para cada base en bases:
            caracteres_validos[base] = evaluar(lista, solucion, base, dHam)

        candidatos = lista_vacía()

        para cada par_clave_valor en caracteres_validos:
            si el valor es verdadero:
                agregar clave a candidatos

        si candidatos no está vacío:
            caracter_seleccionado = seleccionar_candidato_aleatorio(candidatos)
        de lo contrario:
            indice_minimo = encontrar_indice_del_valor_minimo_en_contador()
            segun indice_minimo:
                caso 0: caracter_seleccionado = 'A'
                caso 1: caracter_seleccionado = 'C'
                caso 2: caracter_seleccionado = 'G'
                caso 3: caracter_seleccionado = 'T'

        agregar caracter_seleccionado a solucion
        reiniciar_contador()

        si i >= (dHam - 1):
            para cada string en lista:
                condicion = 0
                para j desde 0 hasta i:
                    si caracter_en_posicion(j) en string == caracter_en_posicion(j) en
solucion:
                        incrementar condicion
                si condicion >= dHam:
                    agregar string a conjuntoSolucion
                    eliminar string de lista

    detener_reloj()

    mostrar("Tamaño de la solución:", tamaño_de_solucion)
    mostrar("Distancia Hamming:", dHam)
    mostrar("Calidad de la solución:", tamaño_de_conjuntoSolucion)
    mostrar("Elementos restantes en lista original:", tamaño_de_lista)
    mostrar("Tiempo usado:", tiempo)

    retornar 0
```

Tablas de Tiempo y Calidad

Tiempo					
N	M	Umbral	Media Greedy	Media A-Greedy	Desviación Estándar A-Greedy
100	300	75%	0.310668	0.154962	0.009547731168
		80%	0.343292	0.176855	0.01324089568
		85%	0.380848	0.183978	0.0130204935
	600	75%	1.277153	0.654199	0.02964810501
		80%	1.427232	0.729370	0.03576097537
		85%	1.478290	0.727050	0.03086904735
	800	75%	2.244550	1.185578	0.050473125
		80%	2.534686	1.298966	0.05933587523
		85%	2.587097	1.299414	0.05170118203
200	300	75%	0.618411	0.311923	0.02095137391
		80%	0.699263	0.354616	0.02957904657
		85%	0.732450	0.373735	0.02634347797
	600	75%	2.561683	1.328995	0.06848074935
		80%	2.892328	1.465577	0.06897427078
		85%	2.920631	1.458615	0.07343987026
	800	75%	4.594842	2.366468	0.1101224264
		80%	5.182541	2.609270	0.1148934033
		85%	5.190200	2.605217	0.1073051325

Calidad					
N	M	Umbral	Media Greedy	Media A-Greedy	Desviación Estándar A-Greedy
100	300	75%	97.700000	82.230000	4.882156751
		80%	49.550000	13.470000	4.562594955
		85%	1.720000	0.110000	0.3450955072
	600	75%	99.910000	90.770000	3.754741111
		80%	48.600000	6.760000	3.169042171
		85%	0.090000	0.000000	0
	800	75%	100.000000	94.570000	2.667632401
		80%	49.090000	4.030000	2.240377854
		85%	0.040000	0.000000	0
200	300	75%	183.930000	147.050000	11.61188437
		80%	55.500000	17.210000	5.554677708
		85%	0.720000	0.040000	0.1969463856
	600	75%	195.180000	165.680000	7.341428039
		80%	39.450000	5.540000	2.587538122
		85%	0.000000	0.000000	0
	800	75%	198.260000	172.070000	6.751363499
		80%	32.430000	2.750000	1.701900305
		85%	0.000000	0.000000	0

Link de la spreadsheet con datos.

Análisis de Tabla

Ya que el algoritmo presentado está construido a partir de una heurística Greedy, su velocidad es considerablemente rápida en problemas que entregan calidades altas

con umbrales relativamente bajos. Y, a medida que los umbrales van subiendo, las calidades disminuyen proporcionalmente.