Capítulo 3

Selección usando Algoritmos Genéticos

3.1. Introducción

En esta sección se aborda el uso de Algoritmos Genéticos (GA del inglés Genetic Algorithms) para selección de características. Así como los resultados obtenidos de aplicar dicha técnica en nuestro caso de prueba, HPV.

La sección 3.2 describe los algoritmos genéticos de manera general. Aportando la teoría básica necesaria para entenderlos. La sección 3.3 contiene la teoría necesaria para entender los detalles de la aplicación de los Algoritmos Genéticos de la manera en que se aplican en este proyecto. La sección 3.4 muestra en detalle el método de trabajo que se siguió para la obtención de resultados. Las secciones 3.5 y 3.7 muestra los avances obtenidos y las observaciones realizadas después del proceso.

3.2. Marco teórico general

Se dice que no hay área de la biología en la que la evolución no haya sido el factor de orden [22]. También que la vasta mayoría de la vida se debe a solamente unos pocos

procesos estadísticos actuando sobre y dentro de las poblaciones y especies [12]. Estos procesos son la reproducción, mutación, competencia y selección; y la evolución es el resultado de la interacción entre estos procesos estocásticos fundamentales que actúan sobre poblaciones generación tras generación.

Los algoritmos genéticos, o algoritmos evolutivos, son una técnica de búsqueda estocástica guiada inspirada por los mecanismos de la selección natural, la genética y la evolución. Las primeras investigaciones sobre el tema fueron desarrolladas por John Holland en la Universidad de Michigan en los 70's con dos objetivos: abstraer y explicar de manera rigurosa los procesos adaptativos de los sistemas naturales y diseñar sistemas de software que conservaran los mecanismos importantes de éstos [9].

Es importante aclarar que la investigación sobre los GAs continua no por simple curiosidad por explorar los puntos mencionados, sino que además porque se ha probado teórica y empíricamente que los GA son una técnica de búsqueda robusta que arroja buenos resultados en espacios complejos, lo que los hace atractivos en un sentido práctico. Los GAs son utilizados actualmente en una amplia gama de ramas incluidas, matemáticas, economía, bioinformática, industrial, etc.

Los GAs son diferentes de otras técnicas de optimización y búsqueda de 4 maneras fundamentales [9]:

1. Los GAs trabajan sobre una codificación de los parámetros a optimizar, no sobre los parámetros directamente.

2. Los GAs mejoran una población de soluciones, no una sola.

3. Los GAs usan información de *rentabilidad* de las soluciones, no derivativas ni otro tipo de información auxiliar.

4. Los GAs utilizan reglas de transición probabilísticas, no determinísticas.

Sobre la posibilidad de atacar un problema con GAs podemos decir que si éste puede ser traducido a la búsqueda de términos para la minimización de una función, entonces puede ser abordado con GAs. Esta función, a la que llamamos función objetivo, es uno de los dos puntos de partida para solucionar un problema con GAs. El segundo punto es una técnica de codificación para una posible solución que será evaluada por la función objetivo.

3.2.1. Ejemplo simple, el Algoritmo Genético Simple (Simple Genetic Algoritm, SGA)

Para ilustrar mejor el funcionamiento de los GAs explicamos ahora el SGA paso a paso. El SGA es la primera versión de un GA y es a lo que generalmente llamamos Algoritmo Genético. Las implementaciones más recientes y complejas, que son llamadas por algunos autores algoritmos *evolutivos*, siguen manteniendo en muchos casos la estructura básica del SGA.

El objetivo de cualquier GA es la búsqueda de parámetros que minimicen una función o funciones. El primer paso para resolver un problema con GAs es identificar esta función y las variables objetivo de las cuales deseamos encontrar sus valores óptimos.

Como habíamos mencionado los GAs trabajan sobre cadenas codificadas de los valores objetivos, no sobre los valores en sí. Así pues, el siguiente paso es definir una codificación para una posible representación. Esta codificación es llamada *genotipo*. A una instancia de esta codificación le llamamos *cromosoma*. Un *cromosoma* se decodifica en un conjunto de valores reales los que llamamos *fenotipo*.

El *genotipo* para un problema dado debe estar representado por una cadena de tamaño finito en un alfabeto finito. Las formas más comunes para esta representación son cadenas binarias, de valores reales o *codificación gris*, que es una forma alterna de

codificación que también utiliza valores binarios. El *genotipo* puede estar segmentado lógicamente para contener la codificación para más de una variable objetivo. Además estas variables pueden estar codificadas de manera diferente, es decir podemos definir genotipos que para algunas variables utilizan cierta representación y para otras una representación diferente [25].

Una vez definida la función objetivo y la codificación para las posibles soluciones (*genotipo*) aplicamos el algoritmo, cuya forma se muestra en la figura 3.1.

Iniciar la población significa crear un conjunto de soluciones iniciales. Estas soluciones están en la forma definida por nosotros. Un punto a decidir es el número N de individuos en esta población. Este valor se mantendrá constante durante toda la ejecución y es importante asignarlo a través de experimentación preliminar y observar el comportamiento del algoritmo, ajustando cuando el tamaño de la población afecte negativamente en tiempo de ejecución o amplitud de la búsqueda.

Una vez iniciada la población inicial entramos en el ciclo generacional que producirá mejores soluciones en cada iteración. Este ciclo generacional consiste en la aplicación de diferentes procedimientos sobre la población actual. Estos operadores son llamados *operadores genéticos*, y son el equivalente a los procesos naturales de selección, reproducción y mutación. El ciclo generacional se repite mientras una condición de terminación no se haya cumplido. La condición más común es que el número de iteraciones, o generaciones, alcance cierto valor.

El primer paso del ciclo generacional es evaluar la aptitud de las soluciones presentes en nuestra población actual. Esto consiste en aplicar la función objetivo sobre cada uno de los valores decodificados de la población.

El siguiente paso es la selección de individuos para el apareamiento. Este paso se relaciona con los principios de *selección natural*. La idea es escoger individuos que nos permitan acercarnos a la solución óptima. Existen diferentes maneras de hacer esta

hb

```
1  iniciar la población P con N individuos al azar
2  while GENERATION ≤ MAX_GENERATIONS
3     evaluar la aptitud de los individuos de P
4     Aplicar selección sobre P
5     Aplicar crossover
6     Aplicar mutación
7     GENERATION += 1
8  endwhile
```

Figura 3.1: The Simple Genetic Algoritm

selección y la forma que usemos en este paso afectará de manera drástica el comportamiento del GA. En secciones posteriores hablaremos de las diferentes técnicas de selección que fueron consideradas para este proyecto.

Los individuos seleccionados forman el conjunto de individuos que se reproducirán. El operador de reproducción es llamado *crossover*. La idea detrás de este operador es utilizar información de los padres para producir hijos. El operador es *ciego* a las propiedades de la cadena decodificada, pues trabaja solamente con la información codificada. Además los procedimientos que generan los nuevos hijos también son estocásticos.

Una vez generados los hijos se procede a aplicar el operador de mutación. Este operador también trabaja sobre las cadenas codificadas. Su funcionamiento es simple, introduce cambios aleatorios a los hijos generados en el paso anterior. El propósito de este operador es permitir que se exploren regiones del espacio solución diferentes a las que no es posible llegar a través de la selección y reproducción. La tasa de mutación es un factor definido por el usuario.

Por último es necesario insertar la nueva descendencia a nuestra población. Recor-

demos que el número de individuos en nuestra población permanece estable así que será necesario seleccionar los individuos que serán insertados y los que serán desechados. Al igual que en las demás etapas del ciclo evolutivo, en ésta existen diferentes formas de implementación que serán discutidas posteriormente.

En resumen, comenzamos con una población aleatoria, cuyos individuos contienen características que combinados podrían contribuir a encontrar individuos mejores. A través del tiempo, el ciclo generacional explora el espacio solución buscando individuos cada ves más cercanos a la solución óptima; guiado por los mecanismos de selección, combinando características de los individuos existentes a través de la reproducción y generando individuos con características no existentes en la población actual a través de la mutación.

3.3. Marco teórico para el diseño del experimento

Como explicamos anteriormente, el SGA es la base para los GAs más avanzados. Para cada uno de los pasos del SGA existen diferentes variantes y métodos de los que se puede escoger para adaptarse mejor al problema en cuestión. De ser necesario también es posible modificar en detalle cualquiera de los procedimientos para explorar diferentes caminos de búsqueda.

En el caso de esta investigación existen ciertas características del problema que hacen desear cierto comportamiento de parte del GA, por lo que investigamos sobre diferentes técnicas de cada una de las partes del algoritmo para ajustarlo mejor al problema. En esta sección explicamos las diferentes técnicas con las que decidimos trabajar.

3.3.1. Generación de la población inicial

La aproximación clásica para la generación de la población inicial es la generación al azar. En este tipo de inicialización se generan N cromosomas los cuales son generados completamente al azar. En el caso de las representaciones con cadenas binarias, cada posición tiene la misma posibilidad de ser 0 o 1.

Otra técnica comúnmente aplicada es sembrar la población inicial con puntos que se creen cercanos a la solución óptima con el propósito de acelerar la convergencia del algoritmo [15]. Esta técnica tiene la desventaja de aumentar la posibilidad de que el algoritmo converja a un óptimo local en caso de que el óptimo global se encuentre lejos del punto sembrado.

3.3.2. Selección de la población para reproducción

Existen diferentes técnicas para la selección de individuos para la reproducción y la reinserción. A continuación mostramos las dos técnicas evaluadas en esta investigación. Ambas son ampliamente usadas para diferentes fines, por lo que exploraremos su desempeño en el problema que atacamos en esta investigación.

Selección por Ruleta

Descrita por Hassoun [11] como "la versión estocástica de la supervivencia del más apto", el Muestreo Estocástico con Remplazo, o selección por ruleta, es la técnica de selección más usada. Esta técnica consiste, en palabras coloquiales, en asignar un segmento de la ruleta a los individuos en base a la aptitud de éstos y la aptitud total de la población actual, y girar la ruleta tantas veces como selecciones se requieran. El procedimiento es el siguiente:

1. Calcular el valor objetivo $f(x_i)$ para cada cromosoma x_i

2. Calcular el valor objetivo total para la población:

$$F = \sum_{I}^{i=1} f(x_i) \qquad \text{i=1,2,...,I donde I = tamaño de la población} \tag{3.1}$$

3. Calcular la probabilidad de selección p_k para cada cromosoma x_i:

$$p_i = \frac{f(x_i)}{F} \qquad \text{i=1,2,...,I} \tag{3.2}$$

4. Calcular la probabilidad cumulativa p_i para cada cromosoma x_i

$$q_i = \sum_{i}^{l=1} p_l \qquad \text{i=1,2,...,I} \tag{3.3}$$

Después la selección se hace de la siguiente manera I repeticiones:

1. Generar un número al azar ρ en un rango $[0, 1]$.

2. Escoger el i-ésimo cromosoma x_i tal que $q_{i-1} < \rho \leq q_i$

Aunque, como ya dijimos, ésta técnica es una de las más usadas, esto no significa que sea una de las mejores. Esta técnica tiene diversos problemas, uno de los más graves es posiblemente el hecho de que un cromosoma con segmento de tamaño > 0 pudiera *dominar* nuestras selecciones [4].

Muestreo Estocástico Universal

El muestreo estocástico universal o SUS por su nombre en ingles (Stochastic Universal Sampling) es un algoritmo de muestreo que se implementa en una sola fase. Surgió con el fin de corregir algunos de los problemas del algoritmo de muestreo de ruleta anteriormente. Fue desarrollado por Baker en 1987 [2].

El algoritmo SUS es simple y eficiente [29]. Dado un conjunto de n individuos y sus valores objetivos asociados, SUS los acomoda en una ruleta donde el tamaño de los cortes asignados a cada individuo es proporcional al valor objetivo (como en el algoritmo

de ruleta). Después, una segunda ruleta, es marcada con y *marcadores* igualmente espaciados entre si, donde y es el número de selecciones que deseamos efectuar. Por ultimo se gira la ruleta y se selecciona un individuo por cada marcador. Las posiciones de los marcadores indican los individuos seleccionados. Si l marcadores caen sobre el mismo individuo, éste es seleccionado l veces. Esto garantiza que ningún individuo sea seleccionado ni más ni menos veces que las esperadas debido a que no menos de $\lfloor v \rfloor$ ni mas de $\lceil v \rceil$ marcadores pueden caer en corte de tamaño v.

3.3.3. Reproducción

La reproducción, o crossover, es el operador genético principal. A través de él se generan nuevas soluciones a partir de las soluciones actuales, desplazándose en el espacio solución hacia un óptimo [25].

Opera sobre dos cromosomas cada vez y genera descendencia combinando algunas de las características de ambos cromosomas. El caso más simple, *crossover* de un solo punto, escoge un punto de corte al azar en las dos cadenas de los cromosomas padres para formar dos subcadenas en cada una, una a la izquierda del punto de corte y otra a la derecha. Después se *pega* la subcadena izquierda de un padre con la subcadena derecha del otro para forma una cadena hija. De manera similar se genera el segundo hijo, pero con las subcadenas restantes. Por ejemplo, si tenemos los cromosomas c_1 y c_2:

$$c_1 = [000|00000]$$
$$c_2 = [111|11111]$$

Y se genera un punto de corte como el marcado por el símbolo |, generamos dos hijos de la siguiente manera:

$$c_3 = [000|11111]$$

$$c_4 = [111|00000]$$

Ahora, hagamos que p_c sea el llamado *factor de reproducción* o *crossover rate*. Este define el proporción, en base al J tamaño de la población, del número de descendencia producida en cada generación. Este factor controla el número esperado $p_c \times J$ de cromosomas que participarán en un *crossover*.

Por ejemplo, si tuviéramos un $p_c = 0{,}25$ esperaríamos que en promedio, un 25 % de los cromosomas participaran en una reproducción. Si tuvieramos un conjunto de cromosomas seleccionados para reproducción de tamaño 10, esperariamos que 2 o 3 cromosomas se reproducieran. El procedimiento es: para cada cromosoma en la población seleccionada para reproducción se genera un número ρ en el rango [0,1], si $\rho < p_c$ ese cromosoma es seleccionado para participar en un cruzamiento.

Multipoint Crossover

Es fácil imaginar operadores de *crossover* que corten las cadenas padres en más de un punto. Los puntos de corte se generarían al azar y se mezclaría la información de los padres intercalando la información de sus respectivas cadenas. A este tipo de operador se le llama, crossover multipunto. A continuación se muestra el procedimiento de un crossover multipunto de 2 puntos, el símbolo | indica los puntos de corte.

$$Padre_1 = [000|00000|0000]$$
$$Padre_2 = [111|11111|1111]$$

genera

$$Hijo_1 = [000|11111|0000]$$
$$Hijo_2 = [111|00000|1111]$$

De Jong [14] estudió este problema y concluyó que el cruzamiento multipunto decrementa la efectividad de los algoritmos genéticos y este decremento aumenta mientras más puntos de corte se utilicen. El *crossover* multipunto acerca al algoritmo genético a un búsqueda al azar simple. Sin embargo, en algunos casos puede generar mejores resultados [25].

Cruzamiento de un punto y multipunto con sustituto reducido (Reduced Surrogate)

Las variedades del crossover de un punto y multipunto que implementan la versión de "sustituto reducido (reduced surrogate)"tienen el fin de garantizar que la reproducción genere nuevos individuos [4].

Cuando se escoge un punto de corte al azar, es posible que algunas de las subcadenas de los padres sean iguales. Debido a esto, al hacer el cruzamiento no generaríamos nuevos individuos. Esto se traduce a pérdida de eficiencia debido a la re-evaluación de espacios ya explorados. Por ejemplo, las siguientes cadenas padres con el punto de corte marcado generarían las mismas cadenas como hijos:

$$Padre_1 = [000|000000010]$$
$$Padre_2 = [000|010100000]$$

Para evitar estos casos, la variante "sustituto reducido"del operador de reproducción garantiza que los puntos de corte se realizen en puntos de las cadenas donde las cadenas padre difieren.

3.3.4. Mutación

La mutación es un operador que produce cambios espontáneos en la población de cromosomas lo que introduce variabilidad extra que permite la exploración de espacios

de solución nuevos y ayuda a evitar la caída en mínimos locales.

La mutación es controlada por el factor p_m al que llamamos *tasa de mutación*. Por cada bit de los cromosomas de nuestra población actual generamos un numero ρ al azar entre [0,1]. Si $\rho < p_m$ entonces ese bit mutó e invertimos su valor. Si el bit era 0 ahora será 1 y viceversa.

Si la codificación del gen no es binaria debe de modificarse el operador de mutación para adaptarlo a la codificación en cuestión.

3.3.5. Algoritmos Genéticos para problemas multi-objetivo

Recordando la definición 1, en cuanto a la evaluación, el objetivo de la selección de características es maximizar cierta propiedad deseada. En el caso de la mayoría de los problemas de selección de características, esta propiedad deseada se divide en dos: minimizar el número de características seleccionadas y maximizar el número de clases distinguibles. Esto hace que nuestro algoritmo genético entre dentro de la categoría de *multiobjetivo*.

Así pues tenemos un problema con más de una función objetivo. Existen diferentes maneras de atacar un problema multiobjetivo. La más común es la suma objetivos ponderados. Para este método se crea una función objetivo de la forma:

$$F(x) = w_1 f_1(x) + w_2 f_2(x) + \ldots + w_i f_i(x) \tag{3.4}$$

tal que $w_1 + w_2 + \ldots + w_i = 1$ y donde $f_i(x)$ es la función del objetivo i y w_i es el *peso* asociado con ésta. Para los i objetivos que deseamos minimizar.

Otra práctica recomendable en el caso de la suma de objetivos ponderados es normalizar las salidas de las funciones objetivo. Debido a que cada función mide diferentes factores, pueden estar en diferente escala; mantener estos valores en una escala similar es conveniente para lograr ajustar adecuadamente los pesos w_i asociados.

Pareto optimalidad

Los problemas multiobjetivo se caracterizan porque generalmente no tienen una solución única, sino un conjunto de soluciones llamadas *pareto-óptimas*.

Definición 2. [25] Un punto $x^* \in X$ es llamado Pareto optimo si y solo si no existe ningun $x \in X$ tal que $f_i(x) \leq f_i(x^*)$, para $i = 1, 2, \ldots, I$ y existe al menos un i donde $f_i(x) < f_i(x^*)$.

Esta definición viene del conocimiento intuitivo de que un punto x^* es escogido como óptimo si ningún criterio puede ser mejorado sin empeorar al menos otro criterio.

La definición anterior tiene como consecuencia que generalmente no se tiene una sola solución, sino un conjunto de soluciones pareto-óptimas a veces también llamadas no-inferiores o no-dominadas. En algunos problemas también pueden ocurrir soluciones débilmente pareto-óptimas.

Definición 3. [25] Un punto $x^* \in X$ es llamado débilmente pareto óptimo si y solo si no existe ningún $x \in X$ tal que $f_i(x) \leq f_i(x^*)$, para $i = 1, 2, \ldots, I$.

Es claro que una solución pareto-óptima es también débilmente pareto-óptima, más no al contrario.

3.4. Algoritmos Genéticos para la Selección de Características, caso HPV

Para resolver el problema de selección de características primero debemos definir las dos partes iniciales del algoritmo genético, la codificación y la evaluación.

3.4.1. Codificación

La codificación es muy simple. Se utiliza un cromosoma $\overline{X}$ en forma de cadena binaria de tamaño n, donde n es el número de características en el conjunto total. Cada bit de la cadena binaria está relacionado con una característica. Si el correspondiente bit es igual a 1, la característica relacionada estará seleccionada y viceversa. Para el caso de la optimización de la clasificación del HPV, las características corresponden directamente a enzimas.

3.4.2. Evaluación

En cuanto a la evaluación, este es un problema de selección de características, y como tal tiene implícitos 2 objetivos. La minimización del número de éstas y la maximización de la distinción entre clases (tipos virales en el caso del HPV). También, en el caso de HPV, decidimos agregar un objetivo más, maximizar la calidad de las enzimas contenidas en la solución. Dicha calidad está en función del número de fragmentos que dicha enzima genera en promedio. Así pues tenemos que nuestra función objetivo queda de la siguiente manera:

$$F(\overline{x}) = w_{count} f_{count}(\overline{x}) + w_{qlty} f_{qlty}(\overline{x}) + w_{sep} f_{sep}(\overline{x}) \tag{3.5}$$

donde $\overline{x}$ es un cromosoma que es una instancia de $\overline{X}$, $f_{count}(\overline{x})$ es el número de características seleccionadas, $f_{qlty}(\overline{x})$ es la calidad de dichas características, $f_{sep}(\overline{x})$ es la separabilidad de las clases a distinguir.

Función $f_{count}(\overline{x})$, número de características seleccionadas

La implementación de $f_{count}(\overline{x})$ simplemente cuenta el número de características seleccionadas en el cromosoma $\overline{x}$, que es directamente el número de 1s en la cadena

binaria. En el caso del HPV, donde tenemos 205 enzimas, $f_{count}(\overline{x})$ devuelve un valor entre [0,205].

Función $f_{sep}(\overline{x})$, capacidad de separación entre clases

Para la capacidad de separación entre clases, $f_{sep}(\overline{x})$ devuelve un valor relacionado con el número de clases no distinguibles utilizando las enzimas seleccionadas en el cromosoma $\overline{x}$. El procedimiento se muestra en la figura 3.2.

Ahí, la función $formarOutputs(\overline{x})$ devuelve una matriz $OUTPUTS$ de tamaño $MAXCLASSES$ donde cada renglón contiene un vector

$$\overline{OUTPUTS_i} = [output(i, feature_{\overline{x}_1})|output(i, feature\overline{x}_2)|\ldots \\ |output(i, feature_{\overline{x}_h})]$$

donde la función $output(j, h)$ devuelve el valor de la característica h para la clase i. Y $feature_{\overline{x}_h}$ es la característica h cuyo bit relacionado en el cromosoma $\overline{x}$ es igual a 1.

En el mismo algoritmo, la función $verifyUnique(\overline{output_m})$ funciona de la siguiente manera:

$$verifyUnique(\overline{output_m}) = \begin{cases} TRUE & \neg\exists\overline{output_{m^*}} \in OUTPUTS \\ & \text{tal que } \overline{output_m} == \overline{output_{m*}} \text{ y } m \neq m^* \\ FALSE & \text{de otra manera} \end{cases}$$

Por último, multiplicamos el porcentaje de fallo calculado por el número de características seleccionables para mantener la salida de $f_{sep}(\overline{x})$ y la salida de $f_{count}(\overline{x})$ en la misma escala [0,205].

Función $f_{qlty}(\overline{x})$, calidad de las enzimas seleccionadas

Esta función es específica para el caso de la selección de enzimas para la tipificación de HPV.

```
1  función f_sep(x̄)
2  OUTPUTS = formarOutputs(x̄)
3  CLASES_NO_DISTINGUIBLES = 0;
4  for CLASS = 1 to MAXCLASS
5     if verifyUnique(OUTPUTS_CLASS) == FALSE then
6        CLASES_NO_DISTINGUIBLES + 1
7     end(if)
8  end(for)
9  PORCENTAJE = CLASES_NO_DISTINGUIBLES/MAXCLASS
10 RESULT = PORCENTAJE * NUMERO_CARACTERISTICAS
11 return result
```

Figura 3.2: Algoritmo para el calculo de $f_{sep}(\overline{x})$

Llamamos enzima de baja calidad a aquella enzima que genera muchos fragmentos de restricción para la mayoría de los tipos virales a clasificar. La razón es que en el laboratorio, a mayor cantidad de fragmentos generados, aumenta la susceptibilidad de fallo debido a mutaciones en el virus o variación de las condiciones de prueba. Entonces deseamos que las enzimas seleccionadas generen, en promedio, un número reducido de fragmentos.

La función $f_{qlty}(\overline{x})$ queda de la siguiente manera:

$$f_{qlty}(\overline{x}) = max(meanFrag(enzime_1), meanFrag(enzime_2), \ldots, meanFrag(enzime_h))^5$$

para toda enzima $enzime_h$ que este seleccionada en el cromosoma $\overline{x}$. Donde la función $meanFrag(h)$ devuelve el promedio de fragmentos que genera la enzima h.

La salida de esta función sin ser elevada a la 5a. potencia está en un rango de [1,2.9]. Elevando esta salida a la 5a potencia extendemos la escala de y aumentamos la presión

para que se seleccione enzimas con un valor bajo en este factor. La escala final esta en el rango [1,205.11].

3.5. Procedimiento y Resultados

Como pudimos ver, existen muchos parámetros que influyen en el comportamiento del algoritmo y que deben ser ajustados. El procedimiento para este ajuste consitió en tomar el parámetro que se deseara ajustar y fijar los demás. Cambiar el valor de dicho parámetro buscando mejoras en el desempeño del algoritmo, escoger el valor que devuelve mejores resultados y continuar con otro parámetro.

Debido a que en el algoritmo interviene el azar, una sola ejecución no es suficiente para saber si un parámetro es adecuado. Se decidió ejecutar el algoritmo 10 veces por cada ajuste en cualquier parámetro

Para su análisis se organizó la salida de las pruebas en tablas como la 3.1. Para entender la tabla recordemos que ejecutamos 10 veces el algoritmo por cada configuración de los parámetros de ejecución. De cada una de estas 10 poblaciones escogemos el mejor individuo según la función objetivo. Además anotamos los casos particularmente mejores aunque no hayan sido los mejores de su población. Esta información es la que se muestra en dichas tablas. Para los parámetros se utilizan las siguientes abreviaturas:

- Tamaño de la población = POPS
- Número de generaciones = NGEN
- Crossover rate = XOVRATE
- Mutation rate = MUTRATE
- Tipo de selección = SELTYPE

- Tipo de inicialización = INITTYPE
- Tipo de reproducción = REPTYPE

Las opciones para estos parámetros son:

- Tamaño de la población = $n \in Z^+$
- Número de generaciones = $n \in Z^+$
- Crossover rate = $n \in [0, 1]$
- Mutation rate = $n \in [0, 1]$
- Tipo de selección = Ruleta (rws, Roulete Weel Selection), Muestreo Estocástico Universal (sus).
- Tipo de inicialización = Al azar uniforme (ur, Uniform Random), Selección Vacia (sv)
- Tipo de reproducción = Cruzamiento en un punto (spxov, Single Point Crossover), Cruzamiento en dos puntos (dbxov, Double Point Crossover), Cruzamiento en un punto con sustituto reducido(spxovrs, Single Point Crossover with Reduced Surrogate), Cruzamiento en dos puntos con sustituto reducido (dbxovrs, Double Point Crossover with Reduced Surrogate).

3.6. Resultados HPV

Los pesos w_{count}, w_{qlty} y w_{sep} de la función objetivo también fueron ajustados en el proceso. Después de algunas pruebas preliminares, se escogieron los siguientes valores iniciales para estas variables:

- $w_{count} = 0{,}5$
- $w_{qlty} = 0{,}35$
- $w_{sep} = 0{,}15$

La idea detrás de estos valores es presionar para que el número de enzimas sea factor que guíe la búsqueda en las primeras generaciones, para después refinar los resultados con los demás objetivos.

Estos pesos se observaron durante el ajuste de los demás parámetros y se hicieron cambios conforme fue necesario. Conservando siempre la idea expuesta en el párrafo anterior.

Iniciamos con los siguientes valores:

- Tamaño de la población = 100
- Número de generaciones = 300
- Crossover rate = 0.7
- Mutation rate = 0.001
- Tipo de selección = Ruleta
- Tipo de inicialización = Random uniforme
- Tipo de reproducción = Single-Point Crossover

La razón principal para escoger estos valores fue, mantenerlos en rangos *típicos* para no errar las decisiones desde el comienzo. Además de algunas pruebas de velocidad que se realizaron anteriormente.

Se probaron 37 configuraciones, para y un total de 370 poblaciones de 100 individuos cada una.

3.6.1. Ajustes en la función objetivo

Durante el proceso se observaron individualmente los resultados de las funciones f_{count},f_{qlty} y f_{sep} buscando casos que pudieran indicar la necesidad de cambio radical en alguno de estos parámetros, principalmente en los pesos de la función objetivo. Se observaron dos casos que condujeron a un ajuste, el primero se muestra en la tabla 3.1. Esta es la configuración #8, marcados en negrita se encuentran los individuos que provocaron el cambió. En este caso el mejor individuo de la población 5 fue evaluado mejor por nuestra función objetivo que el mejor individuo de la población 2. A juicio del usuario esto no es correcto por lo que decidimos ajustar los pesos de la función.

De

- $w_{count} = 0{,}5$
- $w_{qlty} = 0{,}35$
- $w_{sep} = 0{,}15$

a

- $w_{count} = 0{,}6$
- $w_{qlty} = 0{,}25$
- $w_{sep} = 0{,}15$

Posteriormente, en la configuración #11 observamos un caso similar. Dicha configuración y sus resultados se muestran en la tabla 3.2.

En dicha tabla una vez más podemos observar dos casos en el que el juicio de $F(x)$ y el juicio del usuario difieren. Los casos marcados en negrita son los mejores casos de las poblaciones 3 y 10. Donde según $F(x)$ el individuo de la población 3 es mejor que

Tabla 3.1: Ejecución 8. POPS = 100, NGEN = 150, XOVRATE = 0.7, MUTRATE = 0.001, GGAP = 0.05, SELTYPE = sus, REPTYPE = spxov, INITTYPE = sv

Población#	$f_{count}(\overline{x})$	$f_{qlty}(\overline{x})$	$f_{sep}(\overline{x})$
1	10	19.56	0
2	**5**	**18.46**	**0**
3	7	7.59	4.27
4	8	18.46	0
5	**9**	**10.62**	**0**
6	7	23.17	0
7	12	9.95	12.81
8	7	9.95	8.54
9	11	16.41	29.89
10	11	18.46	0

Tabla 3.2: Ejecución 8. POPS = 100, NGEN = 150, XOVRATE = 0.7, MUTRATE = 0.001, GGAP = 0.20, SELTYPE = sus, REPTYPE = spxov, INITTYPE = sv

Población#	$f_{count}(\overline{x})$	$f_{qlty}(\overline{x})$	$f_{sep}(\overline{x})$
1	8	12.86	0
2	8	16.41	0
3	**7**	**9.31**	**0**
4	10	9.31	0
5	7	17.41	0
6	9	9.95	0
7	7	17.41	0
8	7	12.86	0
9	8	9.31	0
10	**5**	**23.17**	**0**

el de la población 10, que es mejor a juicio del usuario. Se ajustó la función objetivo a los siguientes valores:

- $w_{count} = 0{,}8$
- $w_{qlty} = 0{,}05$
- $w_{sep} = 0{,}15$

Ambos cambios mejoraron la calidad de de soluciones generadas por el algoritmo. La última configuración permaneció fija durante el resto de las pruebas.

3.6.2. Progresión de mejoras

Una vez ajustada la función objetivo se procedió a continuar con el ajuste de los parámetros del algoritmo. El ajuste se hizo manualmente, y en algunos casos la decisión de fijar un parámetro en cierto valor fue subjetiva debido a que los cambios que a veces se producían eran casi despreciables. Por ejemplo en el efecto de SUS en lugar de RWS para la selección, o SPXOV o MPXOV para la reproducción.

En otros casos el efecto de los cambios fue notorio, fue para los parámetros XOVRATE y MUTRATE.

En el caso de la inicialización, el uso de inicialización *al azar uniforme* o nuestra variante de inicialización con conjuntos vacios, no pareció afectar la eficacia del algoritmo pero si su eficiencia. La inicialización con conjuntos vacios resulto más eficientes pues arrojó resultados equivalentes a los arrojados usando inicialización *al azar uniforme* pero con tiempos de procesamiento mucho menores. Como ilustración, la configuración del caso #32, de la tabla 3.3 con inicialización *al azar uniforme* tardó un tiempo aproximado de ejecución de 1:30 horas. La misma configuración usando inicialización con conjuntos vacios se ejecutó en aproximadamente 15 minutos. La razón es simple, en el

Tabla 3.3: Ejecución 32. POPS = 100, NGEN = 150, XOVRATE = 0.9, MUTRATE = 0.01, GGAP = 0.20, SELTYPE = sus, REPTYPE = spxovrs, INITTYPE = sv

Población#	$f_{count}(\overline{x})$	$f_{qlty}(\overline{x})$	$f_{sep}(\overline{x})$
1	5	19.56	4.27
2	**4**	**33.70**	**0**
3	5	19.56	0
4	5	18.46	0
5	6	23.17	0
6	5	18.46	4.27
7	5	32.00	0
8	5	33.70	0
9	5	28.80	0
10	5	28.80	0

primer caso se comienza con soluciones con un número relativamente grande de enzimas seleccionadas, lo que hace que la función objetivo tarde más en ejecutarse.

La tabla 3.3 muestra la ejecución que arrojó el mejor resultado del total de ejecuciones del algoritmo. Los parámetros del algoritmo usados para este caso se consideran la mejor combinación obtenida. Las enzimas seleccionadas en esta solución son las número 56,173,192 y 193, con respecto a los dato de entrada.

Las tablas correspondientes a las 36 ejecuciónes del algoritmo se encuentran en el apéndice B.

3.7. Conclusiones

Observamos el efecto de los diferentes parámetros sobre el algoritmo y logramos ajustarlos a lo que creemos, es una forma muy efectiva para resolver nuestro problema. Los parámetros finales permiten alcanzar soluciones muy cercanas al óptimo con la ventaja de ser una solución robusta fácilmente aplicable a otros problemas de selección de características.

A pesar de la incertidumbre inicial sobre el planteamiento de la función objetivo, se

observó que el uso que el uso de la suma de objetivos ponderados resulta efectiva para este tipo de problemas por su alta maleabilidad.

De los parámetros del algoritmo genético, *tamaño de población*, *número de generaciones* y *tasa de cruzamiento*, podemos decir que trabaja de una manera similar, todos estos parámetros ensanchan o estrechan el espacio de búsqueda. Valores elevados para estos parámetros aumentan la probabilidad de encontrar soluciones muy cercanas a la óptima, siempre y cuando se logre mantener la diversidad poblacional. Si esto no se garantiza entonces ocurre la dominación poblacional, donde un individuo fuerte y soluciones muy cercanas a éste poblan por completo el espacio solución y la búsqueda resulta limitada.

Los parámetros *tipo de selección* y *tipo de reproducción* sirven garantizan la diversidad poblacional. Pudimos observar que la selección de tipo *SUS* permite que las poblaciones tarden más generaciones en ser dominadas. El algoritmo *cruzamiento en dos punto con sustituto reducido* resultó más efectivo.

También observamos que debido al gran número de variables en nuestra función objetivo y a, probablemente, la alta redundancia en nuestro conjunto de datos, dependemos mucho de mutaciones con suerte y reproducciones con suerte para alcanzar la solución óptima. Esto es, una vez que se alcanzó un conjunto de características *pequeño*, se vuelve difícil continuar refinando la solución debido a que las mutaciones que agregan características generalmente no aportan información extra, y las mutaciones que eliminan características eliminan información importante. Dependemos de que en algún momento sean seleccionadas las características que nos permiten alcanzar este óptimo, y que durante el proceso evolutivo no se pierdan debido a los efectos de la selección, mutación y reproducción. Esta es una consecuencia de que los algoritmos genéticos están *ciegos* a las características específicas del problema que están resolviendo; es a la vez una debilidad y su fortaleza, puesto que esto también los hace robustos.

Como mencionamos anteriormente, también resulta necesario asegurar la diversidad poblacional. Aunque *SUS* resultó efectivo y obtuvimos buenos resultados, es importante probar con otros algoritmos como *la selección por torneo* o *selección con rango lineal o no lineal* para evaluar de nuevo el desempeño del algoritmo.