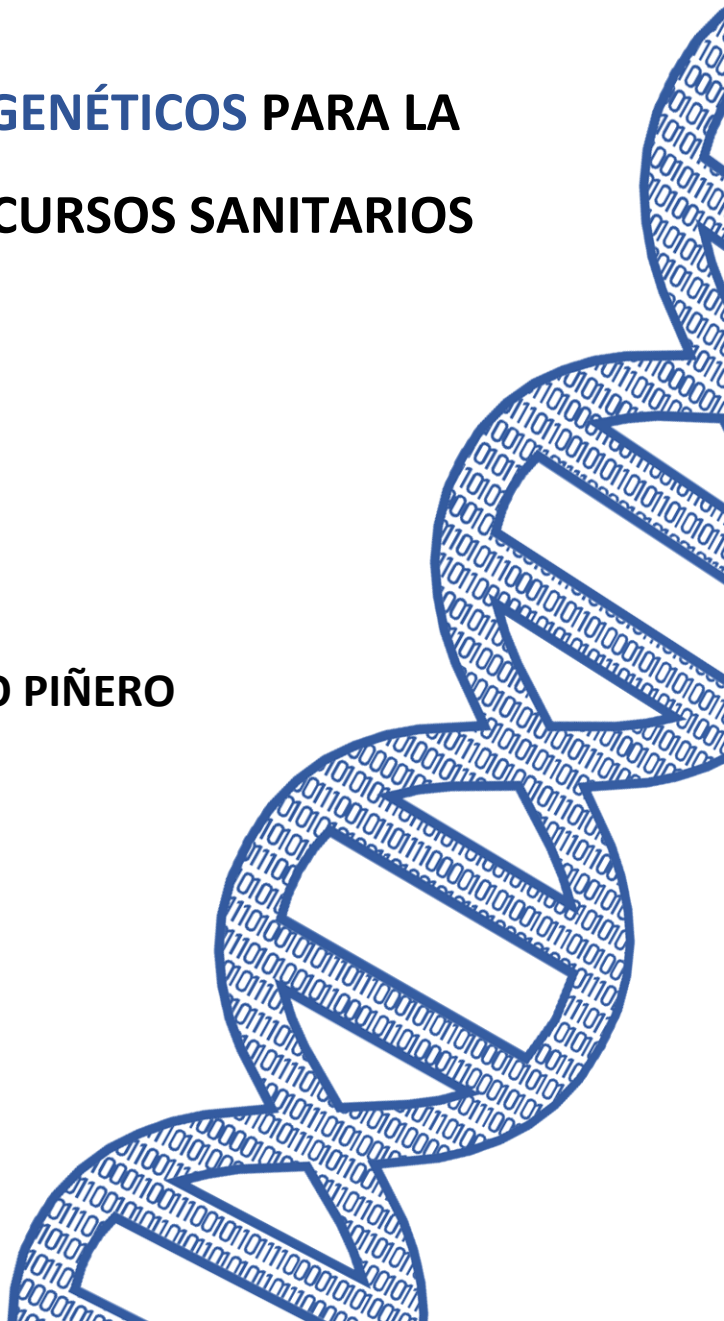
Universidad de Sevilla | Facultad de Farmacia

EL USO DE ALGORITMOS GENÉTICOS PARA LA OPTIMIZACIÓN DE LOS RECURSOS SANITARIOS



ELISA TRIGUERO PIÑERO

Universidad de Sevilla | Facultad de Farmacia

TRABAJO FIN DE GRADO

EL USO DE ALGORITMOS GENÉTICOS PARA LA OPTIMIZACIÓN DE LOS RECURSOS SANITARIOS

– Trabajo de carácter bibliográfico –

Departamento de Química Física

Autor: Elisa Triguero Piñero

Tutor: José Javier Plata Ramos

Sevilla, junio de 2020

RESUMEN

En este trabajo se presenta una revisión sobre el uso de algoritmos genéticos (AG) para la optimización de los recursos sanitarios. Los servicios sanitarios cuentan con recursos limitados, por lo que una gestión eficiente de los mismos produce un mejor resultado en términos de salud. En este trabajo se aborda la capacidad de los AG de mejorar el rendimiento de dichos recursos. Los AG siguen un método inspirado en la evolución natural para resolver problemas complejos y han demostrado ser eficaces en la mejora del rendimiento de los recursos sanitarios existentes. En este trabajo, se detalla su mecanismo cíclico de generación de individuos o soluciones a un problema planteado, seguido de una continua *selección*, *cruce* y *mutación* hasta encontrar una solución optimizada. Así mismo, se abordan sus aplicaciones más habituales en el ámbito sanitario como es la generación de turnos óptimos para pacientes, enfermeras o médicos y su utilización para encontrar ubicaciones espaciales óptimas. Finalmente, se destaca que su ámbito de aplicación es muy amplio y aún en descubrimiento, por lo que está abierta la puerta a futuras aplicaciones para optimizar los limitados recursos sanitarios.

Palabras clave: algoritmo genético, optimización, recursos sanitarios.

ABSTRACT

This paper presents a review about the use of genetic algorithms (AG) for the optimization of health resources. Health services have limited resources, so efficient management of them produces a better result in terms of health. This work addresses the ability of AG to improve the performance of these resources. AG follow a natural evolution-inspired approach to solving complex problems and have proven effective in improving the performance of existing healthcare resources. In this work, its cyclical mechanism of generation of individuals or solutions to a problem posed is detailed, followed by continuous *selection*, *crossover* and *mutation* until an optimized solution is found. Likewise, its most common applications in the health field are addressed, such as the generation of optimal shifts for patients, nurses or doctors and its use to find optimal spatial locations. Finally, it is highlighted that its scope is very wide and still in discovery, so the door is open to future applications to optimize limited healthcare resources.

Keywords: genetic algorithms, optimization, health resources.

ÍNDICE

1. Introducción

1.1 Necesidad de optimizar los recursos sanitarios

En la actualidad, los países desarrollados se caracterizan por presentar un envejecimiento de la población progresivo (Marco Ramo, 2019). Dicho cambio es fácilmente observable en España. Comparando las pirámides de población de 1979, 1999 y 2019 (Fig. 1), se percibe que con el paso de los años la base piramidal es cada vez más estrecha (población joven), mientras que el porcentaje de personas mayores va en aumento.

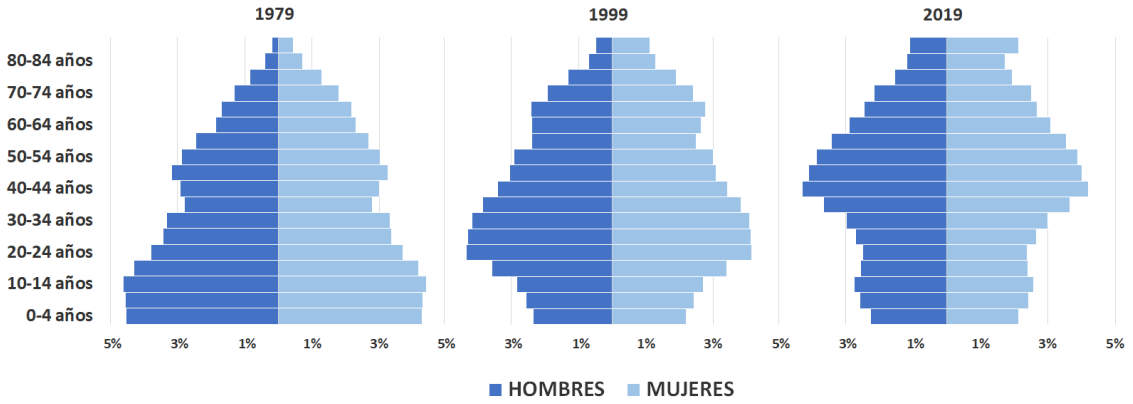


Figura 1: Pirámides de población en España para los años 1979, 1999 y 2019. Los datos utilizados para realizar los gráficos se han obtenido del Instituto Nacional de Estadística (INE, 2020).

Dicho envejecimiento carece de precedentes históricos, encendiendo desde un principio las señales de alarma sobre la sostenibilidad futura de los sistemas sanitarios (Casado Marín, 2001). Sin embargo, aunque este cambio demográfico acentúa la necesidad de maximizar la efectividad de los recursos de salud pública (Sasaki et al., 2010), el envejecimiento demográfico es, *a piori*, un factor que tenemos que asumir.

En cambio, otros factores como la intensidad de la atención, el coste de los tratamientos o el desarrollo de nuevas tecnologías, sí son susceptibles de regulación futura. De la evolución de dichos factores dependerá la cantidad de recursos que destinemos a sanidad y, más importante, lo que de ellos obtengamos en términos de una mejor salud. (Casado Marín, 2001). En otras palabras, la gestión coste-efectividad de los servicios sanitarios implica que los recursos económicos de que dispone el sistema nacional de salud, que son limitados, sirvan para producir el máximo resultado posible en términos de salud (Ribas and Portella, 2000).

En este trabajo nos centramos en el factor del desarrollo tecnológico, a nivel de sistemas de programaciones automatizados. Concretamente, proponemos el uso de **algoritmos genéticos** para agilizar diversas tareas que tienen lugar en el ámbito sanitario.

1.2 Definición de Algoritmo Genético

Los **algoritmos genéticos** (AG) son una familia de métodos computacionales inspirados en la evolución natural (Aickelin and Dowsland, 2004). A lo largo de muchas generaciones, las poblaciones de los organismos biológicos evolucionan de acuerdo con los principios de selección natural y supervivencia del más fuerte, postulados por Charles Darwin en "El origen de las especies" (Darwin, 1859). Por imitación de este proceso, los AG son capaces de ir creando soluciones óptimas o subóptimas a problemas del mundo real (Beasley et al., 1993).

Todos los AG se caracterizan por ser capaces de generar soluciones aleatorias (individuos) a un problema determinado (Ghaheri et al., 2015). Dichas soluciones estarán codificadas por cromosomas, que contienen las propiedades de la solución. Siguiendo las leyes de la genética, el AG producirá cruces y mutaciones en los cromosomas, obteniendo una segunda generación de individuos con propiedades más variadas. El algoritmo seleccionará a algunos individuos de la generación obtenida para cruzar y mutar nuevamente, de forma que las generaciones siguientes contendrán cada vez una solución más optimizada.

Estos algoritmos son, por tanto, métodos metaheurísticos. Exploran un espacio de búsqueda amplio, pero utilizan la experiencia acumulada en cada nueva búsqueda (Hassanien et al., 2016). Gracias a su adaptación, los AG proporcionan modelos adecuados cuando el número de posibles soluciones es demasiado grande para que todas sean evaluadas (Sasaki et al. 2010).

1.3 Historia de los algoritmos genéticos

En las décadas de 1950 y 1960, varios ingenieros informáticos, de forma independiente, se interesaron en los sistemas evolutivos de la naturaleza, viendo en ellos una herramienta de optimización para problemas de ingeniería (Mitchell, 1996). A raíz de esto, numerosos ingenieros desarrollaron, inspirados en la evolución, una gran variedad de algoritmos para la optimización y el aprendizaje automático. La idea de inicial en todos estos sistemas era generar una población de soluciones candidatas a un problema dado, utilizando operadores inspirados en la variación genética y la selección natural. Es en este contexto donde nacen los AG.

En la década de 1960, John Holland inventa unos algoritmos, a los que llama algoritmos genéticos. El objetivo original de Holland era estudiar formalmente el fenómeno de adaptación tal como ocurre en la naturaleza y desarrollar formas en las que los mecanismos de adaptación natural podrían importarse en sistemas informáticos (Mitchell, 1996). Estos algoritmos fueron

desarrollados por Holland, y sus estudiantes y colegas investigadores, en la Universidad de Michigan durante las décadas de 1960 y 1970.

El trabajo que realizan en relación con los algoritmos genéticos queda recogido en su libro "Adaptación en sistemas naturales y artificiales" (Holland, 1975), donde establece rigurosamente los principios básicos de los AG (Beasley et al., 1993).

Holland consigue diseñar una herramienta eficaz para la búsqueda y optimización. Sin embargo, sus AG no se popularizaron hasta la publicación del libro escrito por Goldberg (Goldberg, 1989). Posteriormente a este trabajo, se han ido desarrollando un gran número de publicaciones (libros, revistas, informes internos de investigación…) así como de congresos y conferencias (*International Conference on Genétic Algorithms*, *Parallel Problem Solving from Nature*, *Congress on Evolutionary Computation*, *Genétic and Evolutionary Computation Conference*, etc.), convirtiendo a la computación evolutiva en general, y a los AG en particular, en una de las áreas más activas de la Inteligencia Artificial (Moujahid et al., 2018).

1.4 Por qué inspirarse en la naturaleza

Podríamos preguntarnos por qué se inspiraron los investigadores en utilizar la evolución para resolver problemas computacionales. Para facilitar la explicación, vamos a suponer que queremos diseñar un algoritmo capaz de resolver sudokus.

Supongamos que nuestro método debe ser capaz de hallar, para el sudoku planteado (Fig. 2a) la solución correspondiente (Fig. 2b). Al igual que este sudoku, muchos problemas computacionales requieren buscar entre una gran cantidad de posibles soluciones.

a

	5	1			4	2		
2	3		9	6		4		
9		7				6		1
	6		3	7				4
	9						1	3
4				1	8		5	6
				3	6			2
5		6	8		2			
		9						

b

6	5	1	7	8	4	2	3	9
2	3	8	9	6	1	4	7	5
9	4	7	5	2	3	6	8	1
1	6	5	3	7	9	8	2	4
8	9	2	6	4	5	7	1	3
4	7	3	2	1	8	9	5	6
7	8	4	1	3	6	5	9	2
5	1	6	8	9	2	3	4	7
3	2	9	4	5	7	1	6	8

Figura 2. (**a**) Problema planteado. (**b**) Solución a encontrar.

Nuestro sudoku podría resolverse, por ejemplo, de la siguiente forma: El algoritmo rellena todos los huecos vacíos con los números que faltan en el sudoku. En nuestro caso, faltan cinco *1*, cinco *2*, cinco *3*, cinco *4*, seis *5*, tres *6*, siete *7*, siete *8* y cinco *9*, porque tiene que haber nueve números de cada valor. A continuación, se observan dos posibles soluciones que podrían generarse (Fig. 3):

a

1	5	1	1	1	4	2	1	1
2	3	2	9	6	2	4	2	2
9	2	7	3	3	3	6	3	1
3	6	3	3	7	4	4	4	4
4	9	5	5	5	5	5	1	3
4	5	6	6	1	8	6	5	6
7	7	7	7	3	6	7	7	2
5	7	6	8	8	2	8	8	8
8	8	9	8	9	9	9	9	9

b

1	5	1	4	5	4	2	8	9
2	3	3	9	6	6	4	8	9
9	2	7	4	5	6	6	8	1
1	6	3	3	7	6	7	8	4
1	9	3	4	5	7	7	1	3
4	2	3	4	1	8	7	5	6
1	2	3	4	3	6	7	8	2
5	2	6	8	5	2	7	8	9
1	2	9	5	5	7	8	9	9

Figura 3: (**a**) Posible solución donde los huecos se han rellenado con los números que faltan de menor a mayor valor, ocupando los espacios de izquierda a derecha, desde la primera fila hasta la última. (**b**) Posible solución donde los huecos se han rellenado con los números que faltan de menor a mayor valor, ocupando los espacios de arriba a abajo, desde la primera columna hasta la última.

Ninguna de las dos soluciones cumple las reglas de los sudokus, por lo que el algoritmo tendría que descartarlas y generar nuevas soluciones variando la combinación de los números. Dicho proceso se repetiría una vez y otra hasta que, por azar, encontrase aquella combinación que satisfaga las reglas de este sudoku. Como es de suponer, este método, aunque finalmente encuentra la solución óptima, no es adecuado. Su ineficacia se debe al gran tiempo que requiere buscar entre todas las soluciones posibles. Entonces, ¿qué hacer cuando el espectro de búsqueda es tan amplio?

Una fuente atractiva de inspiración para abordar estos problemas es la evolución biológica. En efecto, la evolución es un método de búsqueda entre una enorme cantidad de posibles "soluciones". En biología, el gran conjunto de posibilidades es el conjunto de posibles secuencias genéticas, y las "soluciones" deseadas son organismos altamente adecuados, organismos capaces de sobrevivir y reproducirse en sus entornos (Mitchell, 1996).

Si la naturaleza ha sido capaz de generar organismos óptimos para desempeñarse en medios ambientes sumamente complejos, por qué no copiar sus métodos para resolver nuestros propios problemas y tratar de encontrarles soluciones que, de alguna manera, sean óptimas

(Kuri and Galaviz, 2002). La gran popularidad que han alcanzado los AG se debe, en buena parte, a que hacen posible abordar problemas en los que es muy difícil aplicar procedimientos matemáticos tradicionales.

Para llegar desde nuestro supuesto algoritmo que resuelve sudokus hasta el AG, vamos a ver ciertas características del comportamiento natural (Mitchell, 1996). La primera de ellas es que en la naturaleza se da un paralelismo. La evolución no trabaja en una única especie, sino que prueba y cambia millones de especies en paralelo. Nuestros problemas de búsqueda también pueden beneficiarse del uso efectivo del **paralelismo**, donde se exploran muchas posibilidades diferentes simultánea y eficientemente.

También la búsqueda de la solución (Fig. 2b) se vería acortada en el tiempo si nuestro algoritmo generase una variedad de soluciones simultáneamente, y las evaluase después.

Pero no todas las soluciones obtenidas van a ser iguales. Entre las que se generen, habrá soluciones más cercanas al óptimo, y otras más lejanas. Es decir, habrá soluciones que estén mejor adaptadas (más cerca de óptimo) y otras que lo estén peor (las que más se alejen). Por tanto, no podemos limitarnos a decir si la solución es buena o mala, sino que tenemos que encontrar una forma de aprovechar las buenas características de las soluciones que están "mejor adaptadas".

En la naturaleza, los individuos de una población compiten por distintos recursos como alimentos, agua o vivienda (Beasley et al., 1993). Además, los miembros de una misma especie compiten entre ellos para atraer a una pareja. Aquellos individuos con más éxito en sobrevivir y atraer parejas tendrán un número mayor de descendientes, mientras que aquellos con bajo rendimiento tendrán pocos o ningún descendiente. Esto significa que los genes de los individuos altamente adaptados o "en forma" se extenderán a un número creciente de individuos en cada generación sucesiva. También, a veces, la combinación de buenas características de diferentes antepasados puede producir una descendencia muy bien adaptada, cuya aptitud es mucho mayor que la de cualquiera de sus padres. De esta forma, las especies evolucionan para adaptarse cada vez mejor a su entorno.

Para que nuestro algoritmo actúe de forma análoga a la naturaleza, tendría que asignar una puntuación a cada solución. Esto es precisamente lo que ocurre en todos los AG. Cuanto más buena es la solución (por ejemplo, cuantas menos reglas del sudoku incumple cada combinación de números que obtenemos), se dice que su grado de adaptación o *"fitness"* es mayor. Esto equivaldría, en la naturaleza, a qué tan efectivo es un organismo para competir por los recursos.

A la hora de generar un nuevo conjunto de soluciones ya no partimos de cero, sino que nos guiamos por los *fitness* de las soluciones anteriores. Aquellas con alto *fitness*, tendrán la oportunidad de "reproducirse" con otras soluciones de la población, mientras que a las soluciones con peor *fitness* se les asigna una menor probabilidad de ser seleccionados para la reproducción y, por lo tanto, desaparecen (Beasley et al., 1993). Se observa entonces que, en los AG, únicamente el primer conjunto de soluciones surge del azar, puesto que los sucesivos se generan combinando las soluciones anteriores en función de su grado de adaptación o *fitness*.

El cómo se "mezclan" las soluciones de las que partimos (a las que, de ahora en adelante, llamaremos "padres") para obtener el nuevo conjunto de soluciones (a las que llamaremos "hijos"), lo explicaremos con detalle más adelante. Pero el proceso está inspirado igualmente en la naturaleza. Las soluciones se "*cruzarán*" y algunas sufrirán *"mutación*", de forma que se permita seguir ampliando el espacio de búsqueda.

Las "reglas" de la evolución son notablemente simples. Las especies evolucionan mediante variación aleatoria (mutación, recombinación y otros operadores), seguidas de una selección natural en la que los más aptos tienden a sobrevivir y reproducirse, propagando su material genético a las generaciones futuras (Mitchell, 1996). Y, sin embargo, siendo tan simples, se les responsabiliza de la extraordinaria variedad y complejidad que vemos en la biosfera.

Si estas reglas funcionan para la evolución encontrando soluciones óptimas entre una gran variedad de posibilidades, también deben funcionar en un algoritmo con el mismo fin y mecanismo, como son los AG.

1.5 Terminología biológica

Antes de entrar en el esquema general de los AG, es necesario explicar la terminología biológica que usaremos en el trabajo.

En el contexto de los algoritmos genéticos, estos términos biológicos se usan con analogía a la biología real, aunque las entidades a las que se refieren son mucho más simples que las biológicas reales (Mitchell, 1996).

1.5.1 Codificación. Cromosomas, genes y alelos

Todos los organismos vivos están formados por células. Cada célula contiene, en su interior, uno o más **cromosomas**, cadenas de ADN, con toda la información del individuo (Mitchell, 1996).

Cada cromosoma puede dividirse conceptualmente en **genes**, y cada gen codifica a una proteína en particular (Mitchell, 1996). En términos generales, un gen codifica un rasgo y los diferentes "ajustes" posibles para un rasgo se denominan **alelos**. Por ejemplo, un rasgo sería el color de ojos, y el alelo los distintos colores verde, azul o marrón (Fig. 4).

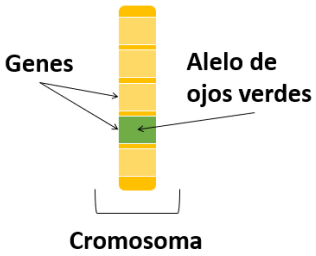


Figura 4: Cromosoma de cinco genes, con un gen para el color de ojos que contiene el alelo de ojos verdes.

En los AG, el término **cromosoma** se refiere a una solución candidata al problema (Beasley et al., 1993). El cromosoma estará a su vez formado por un conjunto de parámetros, llamados **genes**. Cada parámetro o gen podrá tener un valor u otro del alfabeto utilizado, llamado **alelo** (Fig. 5). Según Holland (Holland, 1975), el alfabeto ideal es el binario (ceros y unos), aunque se pueden utilizar otros. El requisito es que el programa que ejecute el algoritmo entienda dicho alfabeto.

(0 1 0 0 1 0 1)

Figura 5: Cromosoma o solución de un AG en alfabeto binario. Dicho cromosoma está formado por un conjunto de 7 genes o parámetros, el que hay 2 alelos posibles (0 y 1).

1.5.2 Genotipo y fenotipo

En términos biológicos, el conjunto de parámetros representados por un cromosoma particular se denomina **genotipo** (Beasley et al., 1993). El genotipo contiene la información requerida para construir un organismo, que se conoce como el **fenotipo**. En el ejemplo del color de ojos, el genotipo llevaría la información con el alelo de ojos verdes, y el fenotipo es la propia expresión de dicho color (Fig. 6).

Genotipo

Fenotipo

Figura 6: Representación de un genotipo (hebra de ADN con alelo de ojos verdes) con su correspondiente fenotipo (ojos verdes).

En el AG, el genotipo de un individuo seria la propia secuencia de bits, alelos, que hay en el cromosoma. La expresión de dicha solución sería el fenotipo del individuo. (Duarte Muñoz et al., 2007).

Aplicado al ejemplo de los sudokus (Fig. 7), la combinación de números integrada en el problema sería realmente el fenotipo. Mientras que, únicamente la secuencia de números, en orden, que hemos puesto en el sudoku, sería el genotipo.

(1 1 1 1 1 2 2 2 2 2 3 3 3
3 3 4 4 4 4 5 5 5 5 5 5 6
6 6 7 7 7 7 7 7 7 8 8 8 8
8 8 8 9 9 9 9 9)

Genotipo

1	5	1	1	1	4	2	1	1
2	3	2	9	6	2	4	2	2
9	2	7	3	3	3	6	3	1
3	6	3	3	7	4	4	4	4
4	9	5	5	5	5	5	1	3
4	5	6	6	1	8	6	5	6
7	7	7	7	3	6	7	7	2
5	7	6	8	8	2	8	8	8
8	8	9	8	9	9	9	9	9

Fenotipo

Figura 7: Fenotipo y Genotipo de la solución representada en la Fig. 3a.

En los AG, el fenotipo representa su aptitud para adaptarse al problema o *fitness* (Beasley et al., 1993). Esto se puede inferir del genotipo, es decir, se puede calcular a partir del cromosoma por lo que no es necesario que los AG representen el fenotipo para realizar la evaluación.

1.5.3 Haploides y diploides

Según la terminología biológica, los organismo pueden ser haploides o diploides (Voet and Voet, 2010). Si contienen una única copia de cada cromosoma serán haploides (Fig. 8a), mientras que si presentan dos copias por cromosoma se llaman diploides (Fig. 8b).

Figura 8: (**a**) Individuo *haploide* (una copia de cada cromosoma). (**b**) Individuo *diploide* (dos copias de cada cromosoma).

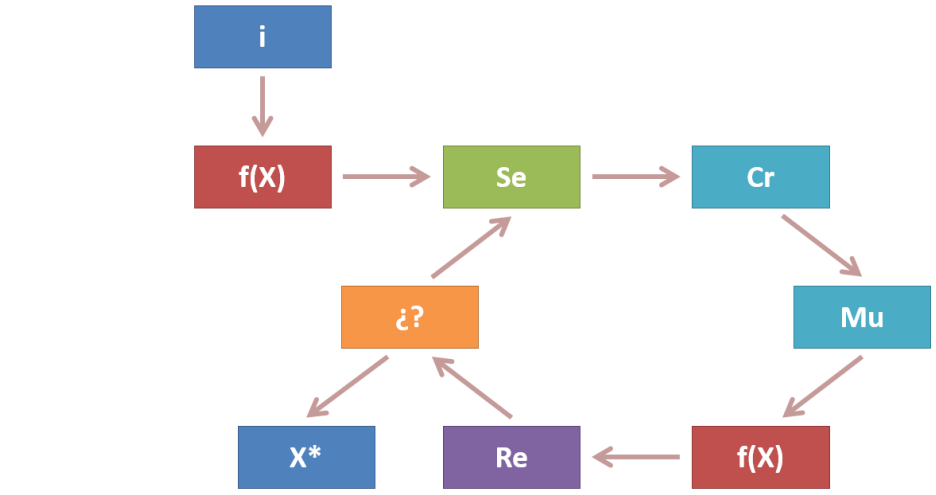
Además, cada individuo puede contener un único cromosoma en sus células, o bien que su información genética esté contenida en varios cromosomas (Voet and Voet, 2010).

Figura 9: (**a**) Individuo haploide con un único cromosoma. (**b**) Individuo haploide con más de un cromosoma (en este caso, con 5 cromosomas distintos).

Salvo contadas excepciones, **todos los AG emplean individuos haploides de un solo cromosoma**. (Mitchell, 1996).

1.6 Esquema del Algoritmo Genético

Todos los AG operan siguiendo un esquema circular (Fig. 10):



i	**Inicialización.** Se genera la población inicial.
f(X)	**Función de evaluación.** Se evalúa a la población correspondiente.
Se	**Selección.** Se seleccionan individuos de la población correspondiente.
Cr	**Cruce.** Operador genético que cruza dos individuos de la población seleccionada.
Mu	**Mutación.** Operador genético que muta un individuo.
Re	**Reemplazo.** Reemplaza en la población inicial con la población generada.
¿?	**¿Fin?** Se comprueba si se cumplen las condiciones de terminación.
X*	**Solución óptima.** Se escoge la solución de mejor *fitness* de la población final.

Figura 10: Esquema general de los Algoritmos Genéticos.

El proceso comienza con la **inicialización**, donde se generan al azar tantos individuos como se deseen tener para la población inicial (Holland, 1975; De Jong, 1975; Goldberg, 1989). A continuación, se calcula lo adaptados que están dichos individuos (soluciones) al problema planteado, mediante la **función de evaluación**. Cada individuo recibirá un valor, denominado *fitness*, en función de lo bien o mal que se adapte.

A continuación, la función de **selección** elige una serie de individuos de entre la población inicial, para ser los padres de la futura descendencia. La probabilidad de ser elegidos suele ser proporcional al *fitness* (Michalewicz and Fogel, 2000; Reeves, 2003).

Los individuos seleccionados como padres **cruzan** sus cromosomas para generar descendencia. El cruce se hace de forma que cada 2 padres, se obtienen 2 hijos. Los hijos generados están formados por fragmentos de sus padres. Sobre cada hijo obtenido se aplica un operador de **mutación**, que con una probabilidad muy pequeña produce cambios en el cromosoma.

Una vez que los hijos han pasado por el operador de mutación, son evaluados por la **función de evaluación**, que les asigna también a ellos un *fitness*. Proporcionalmente a ese *fitness*, o no, se decidirá si la descendencia **reemplaza** a alguno, todos o ninguno de los individuos que pertenecen a la población inicial. Tras aplicar la función de reemplazo, se finaliza con una población del mismo tamaño que la inicial.

Llegado a este punto, se verifica si el AG cumple o no las condiciones de **fin** o terminación, que pueden ser muy diversas. En el caso de que no se cumplan, lo más habitual, el ciclo se reinicia escogiendo como población inicial, a la final tras el remplazo. Si, por el contrario, se cumple la condición de parada, el AG devuelve el individuo de mayor *fitness* de la población final como una **solución óptima** al problema planteado.

La cantidad de variantes que se pueden presentar en los distintos pasos del AG es enorme. Existen numerosas variedades para cada uno de los pasos que componen el algoritmo. Algunos de ellos los nombraremos en este trabajo. En cualquier caso, a cada problema se le diseñará un AG único, con la idea de que sea el más adecuado para encontrarle soluciones optimizadas (Reeves, 2003).

1.6.1 Generar la población inicial

El AG comienza generando una población inicial, donde cada individuo debe estar constituido por los valores del alfabeto con probabilidad uniforme (Corral Sastre, 2018). De esta forma, la población inicial es completamente aleatoria, pues está constituida por ristras generadas al azar.

Algunos trabajos han estudiado la posibilidad de la población inicial se generase de acuerdo con alguna técnica de optimización local (Moujahid et al., 2018). En sus resultados se constata que esta inicialización no aleatoria de la población inicial tiene un desenlace indeseado. En estos casos, las soluciones generadas por el AG suelen ser muy cercanas entre sí. Habitualmente, el algoritmo se cierra sobre esa zona y finaliza con soluciones de *fitness* lejanos de ser óptimos.

Para explicarlo, podríamos suponer que queremos encontrar el pico europeo de mayor altura. Tras aplicar la técnica de optimización para generar los padres se selecciona, por ejemplo, la zona de Los Balcanes. Cuando el AG genere soluciones, podrían ser cualquiera de las siguientes (marcadas con una X en rojo sobre la Fig. 11). Evaluará las alturas de las zonas, asignando mayor puntuación a las más altas. A partir de ellas, generará la descendencia. Como los padres estaban cerca en el mapa, la descendencia estará también en puntos cercanos. El AG, con cada ciclo, encontrará picos más altos, pero en esa zona. Podrá encontrar soluciones como el pico Moldovenau o el Olimpo, pero esas soluciones, aunque sean buenos para su zona, son bajos si comparamos con otros picos como el Teide, el Mont Blanc o el Eibrus.

Figura 11: Mapa físico de Europa donde se señalan los picos de mayor altura (encuadrados) y donde se representan con una *X roja* tres posibles soluciones generadas por el AG al inicio.

De este ejemplo podemos deducir que, si queremos buscar en un mapa grande, las soluciones iniciales deben poder surgir de cualquier punto del mapa, de forma que una mayor variedad de puntos geográficos será explorada.

La población inicial tendrá un tamaño de población determinado previamente. Si la población es pequeña, se corre el riesgo de no cubrir adecuadamente el espacio de búsqueda, mientras que trabajar con poblaciones de gran tamaño pueden llevar a un costo computacional elevado, sobre todo a la hora de calcular el *fitness* de los individuos de la población (Corral Sastre, 2018). Alander (Alander, 1992), basándose en la evidencia empírica, sugirió que un tamaño de población comprendido entre *l* y 2*l* es suficiente para atacar de forma efectiva los problemas por él considerados (*l* es la longitud de la representación).

1.6.2 Función de evaluación

A partir de la población generada, se seleccionarán los individuos para reproducirse. La selección se hace en función de la aptitud o *fitness* de cada individuo, es decir, de cuán buena solución es para el problema planteado. La función que asigna un valor según el *fitness* del individuo es la **función de evaluación** o función objetivo.

La función de evaluación va a ser siempre específica de cada problema a resolver (Beasley et al., 1993). No puede ser igual la función de evaluación en el AG que resuelve sudokus que en el que busca picos altos de Europa, porque el objetivo de cada uno es diferente.

Por esta razón, la función de evaluación es el aspecto más crucial de cualquier AG. Numerosos estudios han tratado de mejorar el AG modificando distintos puntos del AG (Beasley et al., 1993). La conclusión en la mayoría de los estudios ha sido que todas aquellas modificaciones realizadas en pasos del AG, distintos a la función de evaluación, apenas conseguían mejoras en el rendimiento. Grefenstette (Grefenstette, 1986) concluyó que el mecanismo básico de un AG era tan robusto que, dentro de márgenes bastante amplios, la modificación de los demás parámetros no fueron críticos. Sin embargo, lo que es crítico en el desempeño de un AG es la función de evaluación, junto con la codificación utilizada.

Además de ser el punto más crítico, es el que supone mayor complejidad a la hora del diseño. Para cada tipo de problema, el tipo de función de evaluación adecuado variará.

Los mecanismos para el diseño de la función de evaluación son muy variados. Considerando que no es necesario entrar en ellos para este trabajo en cuestión, sí aclararé un aspecto del

mecanismo por el cual evalúa la función de evaluación: En los problemas de optimización combinatoria, la función de evaluación trabaja con restricciones (Beasley et al., 1993). Dichas restricciones se corresponden con los objetivos del AG.

El AG tiene siempre un **objetivo principal**, que cuando se cumpla, habrá encontrado una solución óptima. En el sudoku, el objetivo principal es que todas las filas, columnas y cuadrantes tengan únicamente un número de cada valor (1 al 9). Cuando dicho objetivo se cumple, se ha encontrado una solución óptima al problema.

Sin embargo, un único objetivo principal es demasiado limitante. Si la función de evaluación sólo puede evaluar una restricción (que el objetivo principal se cumpla o no), tan sólo asignará *fitness* **absolutos** (puntuación cero o máxima).

Es ello por lo que en los AG no se trabaja con restricciones absolutistas (todo o nada), sino que se trabaja con **penalizaciones**. Estas penalizaciones consisten en lo siguiente:

Además del objetivo principal, se determinan varios **objetivos secundarios**. Dichos objetivos secundarios se obtendrían de fragmentaciones del objetivo principal. Establecer estos objetivos y determinar la importancia de cada uno es el paso más laborioso del AG.

Si volviéramos al sudoku, podría haber objetivos secundarios como conseguir filas, columnas o cuadrantes que, de forma aislada, tengan únicamente un número de cada valor. Cuanto mayor sea el número de filas y/o columnas y/o cuadrantes que cumplan dichas restricciones para una solución, más cerca estará la solución de cumplir el objetivo principal. Por tanto, la función de evaluación le asignará un *fitness* mayor.

Ahora la función de evaluación puede hacer una discriminación mayor. Es decir, el rango de valores que puede asignar es más variado. De esta forma, ya no se limita a *todo o nada*, sino que diferencia entre soluciones más optimizadas de las que no.

Si las restricciones reflejan adecuadamente el camino hacia el objetivo final, la función de evaluación hará evaluaciones adecuadas. Siendo así, el AG asigna los valores de *fitness* dentro de un rango y dicho rango depende de cada problema particular.

1.6.3 Función de selección

Una vez que los individuos han sido evaluados y se les ha asignado un *fitness*, la función de selección escogerá a aquellos que vayan a cruzarse.

La función de selección realiza su elección al azar, pero en un procedimiento que favorezca a los individuos mejor adaptados y desfavorezca a los que tienen peor *fitness*. De esta forma, aunque los individuos de alto *fitness* sean probablemente escogidos más veces, no se impide que los de baja adaptación puedan ser elegidos.

Dar posibilidad a los individuos de bajo *fitness* de ser elegidos es algunas veces crucial puesto que, al cruzarse con otro individuo, puede generar uno de *fitness* mayor que si se cruzaran dos de *fitness* altos. Es decir, que los peor adaptados contienen a veces información necesaria para llegar a la solución óptima (Kuri and Galaviz, 2002).

Existen numerosos procedimientos para realizar la selección. La más habitual es la llamada **selección proporcional o "ruleta"** (Goldberg, 1989)**.** En ella, los individuos tienen una probabilidad de ser elegidos proporcional a su *fitness*.

Esta selección es llamada "ruleta" porque funciona como una ruleta de casino (Fig. 12). En la ruleta, cada individuo ocupa una porción de forma proporcional a su *fitness*. Se "lanzaría" la bola y la zona en donde caiga determina el individuo que se elegirá. Por tanto, aunque la bola puede caer en cualquiera de las porciones de la ruleta, lo hará con más frecuencia en las porciones que correspondían a individuos de *fitness* mayor, al presentar mayor área.

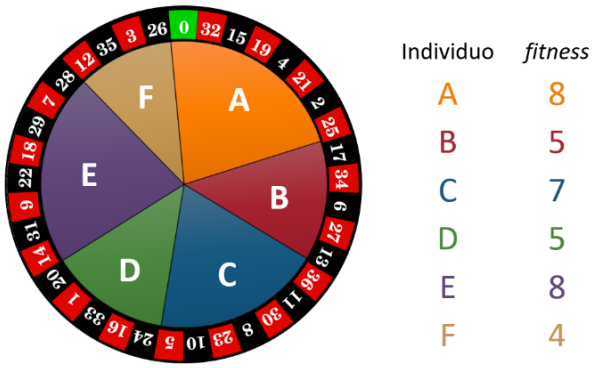


Figura 12: Analogía de la selección proporcional con una ruleta de casino. Cada individuo ocupa una porción de la ruleta proporcional a su *fitness*.

La ruleta se hace girar tantas veces como individuos queremos que sean padres (Goldberg, 1989). Otra opción es usar la ruleta una única vez, y en el sitio donde "caiga la bola" se corresponde a un padre, y los demás padres se van seleccionando al ir alejándose una distancia equidistante del punto donde ha caído (Baker, 1987). En ambos mecanismos, se permite que un mismo individuo sea seleccionado más de una vez como padres.

1.6.4 Operadores genéticos

Una vez que los padres han sido seleccionados, sus cromosomas se recombinan para producir descendencia. Esto ocurre a través de los operadores de cruce. Además, la descendencia generada podrá sufrir mutaciones, por los operadores de mutación (Fig. 13). Ambos operadores simulan los procesos de entrecruzamiento y mutación que se dan en la naturaleza.

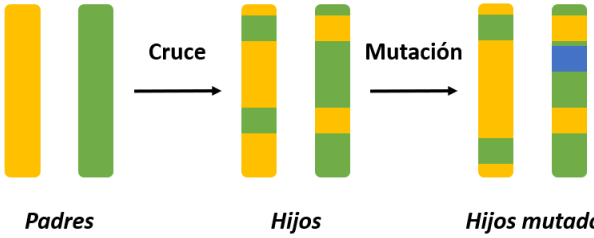


Figura 13: Padres que se cruzan generando una descendencia. Posteriormente, ambos hijos de la descendencia sufren mutación

Cualquier algoritmo de optimización eficiente debe ser capaz de combinar la **explotación** (utilizar el conocimiento previo a la hora de buscar mejores zonas en el espacio de búsqueda) y la **exploración** (investigar áreas nuevas y desconocidas del espacio de búsqueda) (Beasley et al., 1993).

Ya explicamos que la selección de los padres se hacía en base al *fitness* de cada individuo. Por tanto, favorecía la explotación del espacio del espacio de búsqueda. Ahora, los operadores genéticos, al producir nuevas soluciones (descendencia), van a favorecer su exploración. Es importante recalcar que la explotación y la exploración deben mantener un cierto equilibrio, pues el exceso de uno reduce necesariamente al opuesto. Se jugará con dichos parámetros (selección y operadores genéticos) según mejor convenga en cada problema.

Antes de explicar ambos operadores, hay que resaltar que la descendencia generada puede representar soluciones no factibles al problema (Corral Sastre, 2018). En dicho caso, será nuestra elección corregirlos, eliminarlos, o no hacer nada, en base al problema determinado para el que diseñemos el AG.

OPERADORES DE CRUCE

Los operadores de cruce están basados en la naturaleza. Al igual que las nuevas generaciones en cada especie se generan tras el intercambio de los cromosomas de los padres, en los AG los cromosomas seleccionados como padres se cruzarán para producir una descendencia.

En el cruce se seleccionan dos padres para generar dos hijos. Existen variaciones del AG donde se seleccionan de forma simultánea dos o más parejas de padres para que produzcan, a su vez, dos o más generaciones de hijos.

Antes de explicar las formas más conocidas de realizar un cruce, es necesario aclarar que, aunque cromosomas se representan como secuencias cerradas, los operados de crucen los **interpretan como un circuito,** y no como si estuvieran cerrados con principio y fin (Fig. 14). (Moujahid et al., 2018)

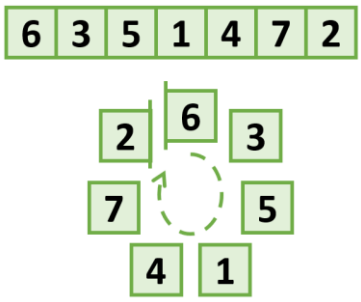


Figura 14: Individuo visto como un circuito.

Las formas más habituales de realizar el cruce de los cromosomas son:

<u>Operador de cruce basado en *n* puntos</u>: Se seleccionan *n* puntos (siendo *n* un número natural) y se intercambian los genes que se encuentran entre los puntos.

Cuanto mayor es *n*, el cruce tiene un efecto más destructivo porque, aunque amplía el espacio de búsqueda, aumenta la probabilidad de ruptura de buenas soluciones (Corral Sastre, 2018).

De Jong (De Jong, 1975) investigó el comportamiento de este operador, concluyendo que *n*=2 era el valor que mejores resultados obtenía en los AG. Por ello, en general se utiliza el cruce basado en 2 puntos.

En el **cruce basado en 2 puntos**, se seleccionan 2 puntos al azar en los padres, y los genes comprendidos entre ambos puntos se intercambiarán para generar la descendencia (Figs. 15 y 16).

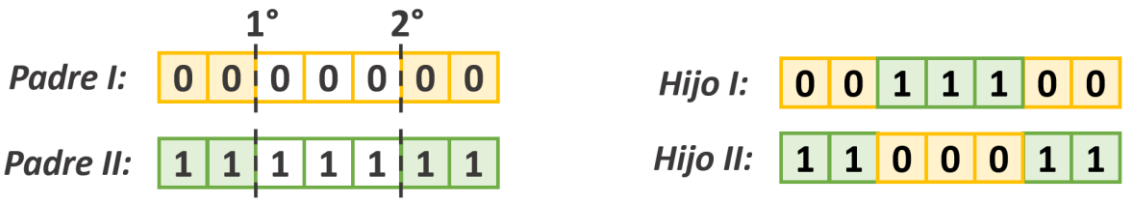


Figura 15: Cruce basado en dos puntos de cromosomas codificados en binario.

Figura 16: Cruce basado en dos puntos de cromosomas codificados en binario.

Cruce uniforme: En el cruce uniforme (Syswerda, 1991) se genera primero un descendiente, y después el opuesto.

Cada gen del primer hijo se escogerá a partir de un padre u otro, por un procedimiento similar al de lanzar una moneda al aire. Si saliese cara, se escoge de un padre, si saliese cruz, se escoge del otro. De esta forma, "lanzando una moneda" para cada posición, se completa el primer hijo. A continuación, el segundo hijo se forma con los genes no escogidos de los padres en el primer descendiente, manteniendo sus posiciones iniciales (Fig. 17).

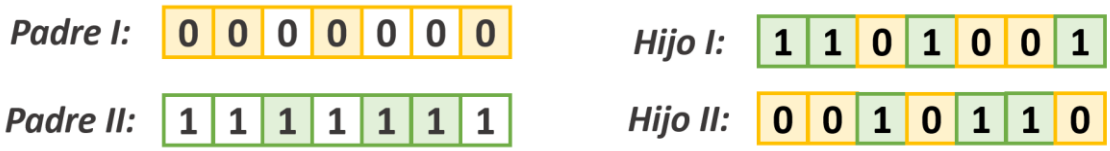


Figura 17: Cruce uniforme de cromosomas codificados en binario.

Aunque los operadores clásicos, como son el cruce basado en *n* puntos y el cruce uniforme, son los más habituales, existen otros más complejos. Estos operadores complejos se distancian del entrecruzamiento que se da en la naturaleza y son más utilizados que los clásicos cuando la codificación no es binaria. Están limitados a cromosomas que no presenten alelos repetidos. Destacamos el cruce parcialmente mapeado, de entre los operadores de mutación complejos.

Cruce parcialmente mapeado: En cruce parcialmente mapeado (Goldberg and Lingle, 1985), se seleccionan dos puntos al azar en los padres A y B. Los genes comprendidos entre dichos puntos se copian, en los hijos contrarios (padre A en hijo B, y viceversa). Después, a cada hijo se le copia los genes su padre correspondiente. Únicamente van a copiarse las posiciones que no tienen genes vacíos (es decir, todos aquellos que no fueron copiados en el hijo por el padre contrario).

En el primer gen se copia el alelo del primer gen de su padre correspondiente. El proceso sería el mismo para todos los genes vacío salvo una excepción: Si en la posición a rellenar del hijo, el padre presenta un alelo que el hijo ya posee, se busca dicho alelo en el padre contrario.

Entonces, se vuelve desde esa nueva posición al padre inicial y el alelo que presenta el padre en su nueva posición se copia en el hijo, pero manteniendo en él su posición de partida.

Dicho mecanismo puede observarse en el ejemplo siguiente (Fig. 18), donde 2 padres generan 2 hijos por dicho mecanismo:

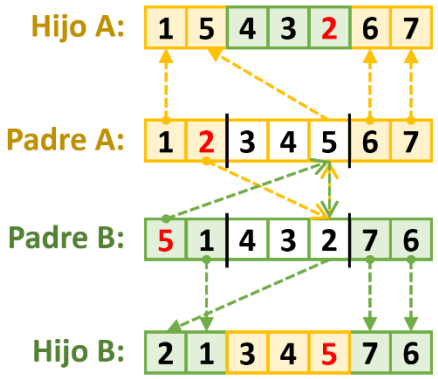


Figura 18: Cruce parcialmente mapeado

OPERADORES DE MUTACIÓN

Los operadores de mutación son aquellos que van a producir cambios en los hijos generados tras el cruce. Su objetivo es aumentar la exploración.

Dichos operadores, a diferencia de los de cruce, actúan sobre cromosomas individuales y no sobre pares de ellos. Actúan en base a una determinada probabilidad, denominada probabilidad de mutación. La probabilidad de mutación es siempre baja (normalmente de 0,001) (Beasley et al., 1993), para que no se pierda la capacidad de explotación del algoritmo.

Cuando se trabaja con un alfabeto binario, que es el más habitual se da este tipo de mutación, llamada **mutación estándar** o clásica (Fig. 19). En ella, la probabilidad de mutación determina si se produce o no el cambio de alelo sobre el gen del cromosoma en el que está actuando en ese momento.

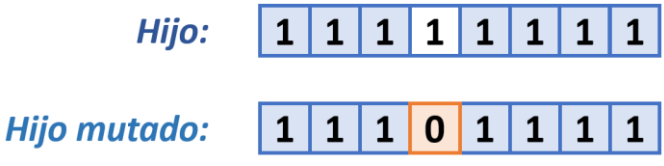


Figura 19: Mutación estándar que muestra la selección del cuarto gen de la descendencia, y su correspondiente mutación.

Cuando la codificación es más compleja, se suelen utilizar operadores de mutación más intrincados, que siguen operando en base a la probabilidad de mutación. Los más habituales son:

Operador de mutación basado en el desplazamiento (Michalewicz, 1999):

Se seleccionan aleatoriamente dos puntos de corte. La subristra comprendida entre los puntos se extrae y se inserta en un lugar aleatorio. En la Fig. 20, el operador de mutación ha seleccionado la subristra (345) y ha escogido el alelo (7) para insertarla a continuación.

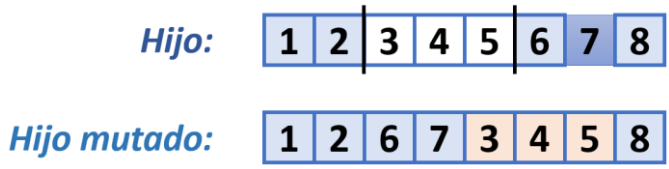


Figura 20: Mutación a través de un operador basado en el desplazamiento.

Operador de mutación basado en cambios (Banzhaf, 1990):

Se seleccionan al azar dos alelos y los intercambia de posición. En la Fig. 21, el operador de mutación ha escogido los alelos (3) y (5) para intercambiarlos.

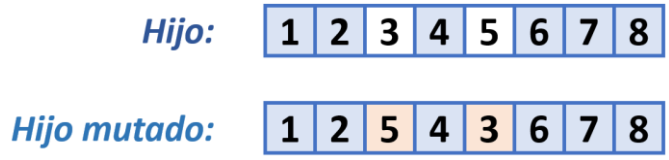


Figura 21: Mutación a través de un operador basado en cambios.

Operador de mutación basado en el cambio (Syswerda, 1991):

Se seleccionan aleatoriamente dos puntos de corte. A continuación, se intercambia al azar las posiciones de los alelos que contiene la subristra generada. En la Fig. 22, el operador de mutación ha escogido la subristra (45678), sobre la que altera el orden de los alelos.

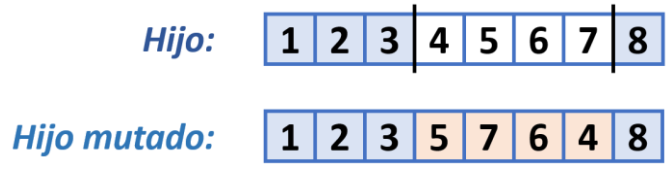


Figura 22: Mutación a través de un operador basado en el cambio.

Operador de mutación basado en la inserción (Fogel, 1988; Michalewicz, 1999):

Se selecciona al azar un alelo. Dicho alelo se extrae y se inserta en otro lugar seleccionado al azar. En la Fig. 23, el operador de mutación ha seleccionado el alelo (4) y ha escogido la séptima posición para insertarlo a continuación.

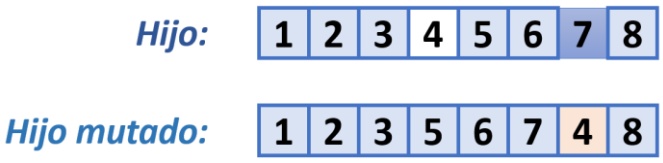


Figura 23: Mutación a través de un operador basado en la inserción.

Operador de mutación basado en la inversión simple (Holland, 1975; Grefenstette et al., 1985):

Se seleccionan aleatoriamente dos puntos de corte. A continuación, se invierten las posiciones de los alelos que contiene la subristra generada. En la Fig. 24, el operador de mutación ha seleccionado la subristra (3456), cuyos alelos invierte de orden.

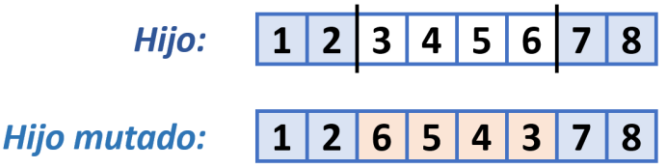


Figura 24: Mutación a través de un operador basado en la inversión simple.

Operador de mutación basado en la inversión (Fogel, 1988; Fogel, 1993):

Se seleccionan aleatoriamente dos puntos de corte. La subristra comprendida entre los puntos se extrae y se inserta en un lugar aleatorio, pero invirtiendo el orden de los alelos. En la Fig. 25, el operador de mutación ha seleccionado la subristra (345), y ha escogido el alelo (7) para insertarla a continuación, pero invirtiendo de orden sus alelos.

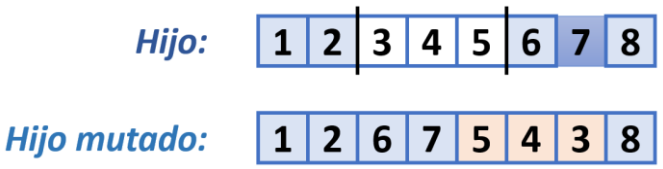


Figura 25: Mutación a través de un operador basado en la inversión.

1.6.5 Función de reemplazo o reducción

Tras la aplicación de los operadores genéticos, los individuos generados (descendientes) son calificados por la función de evaluación, que les asigna un *fitness*. Llegados a este punto tendremos que decidir qué individuos queremos que conformen la población.

El procedimiento más habitual es el llamado **generacional**, que consiste en reemplazar toda la población en cada generación. Cada individuo sobrevive solo una generación porque todos los hijos reemplazan a todos los padres (Corral Sastre, 2018). Este procedimiento está respaldado por las investigaciones de Grefenstette (Grefenstette, 1986).

Una tendencia más reciente propone que en cada generación se reemplacen sólo unos pocos individuos (Beasley et al., 1993). Dicho proceso se asemejaría a las especies de vida larga, en la naturaleza, donde los padres y su descendencia viven al mismo tiempo, generando competencia entre ellos.

El procedimiento más habitual que sigue este mecanismo es el de tipo elitista. En una reducción **elitista**, se escogen de entre la población inicial más sus descendientes aquellos con mejor *fitness* (Corral Sastre, 2018). Se continúa el proceso de selección hasta completar el número de individuos que son necesarios para formar la población inicial.

Creo necesario aclarar que no siempre se van a generar tantos individuos como población inicial había. Si la población inicial era de 10 individuos, pero hemos determinado en la función de selección que sólo sean seleccionados 2 individuos como padres, únicamente se generarán 2 hijos. En este caso, la población total que tendríamos sería de 12 que, tras aplicar la función de reducción, sería una población de 10 individuos, como habíamos empezado.

Se entiende entonces que, en el reemplazo generacional, los 10 individuos finales sean los 8 que no se eligieron, más los 2 descendientes. Mientras que, en la reducción elitista, los 10 individuos finales serán los 10 con mejor *fitness* de entre los 12. El reemplazo generacional, sin embargo, no es habitual en estos casos, sino que es propio de aquellos procesos donde se generan el mismo número de descendientes que progenitores había.

1.6.6 Condición de parada

Una vez se ha realizado la reducción de la población, es el momento de comprobar si el AG debe parar o volver a repetir todo el ciclo.

Las condiciones que determinan si el AG continúa o termina, pueden ser variadas. Puede haber una única condición o varias posibles. Dichas condiciones se establecen para el AG cuando se plantea el tipo de problema a resolver.

Entre las condiciones de parada, las más habituales son: tras llegar a un número fijo de generaciones (p.ej., tras 100 generaciones), tras un tiempo de cómputo fijo (p. ej., tras 2 horas desde el inicio), cuando se llegue a un determinado *fitness* (p. ej., al aparecer un individuo de *fitness* 9) o tras varias generaciones sin variación del *fitness* (p. ej., si tras varias vueltas, la población mantiene un *fitness* constante de 6) (Corral Sastre, 2018).

Si no se cumplen las condiciones, el AG reinicia el ciclo utilizando como población inicial a la población final tras la función de reducción.

Si las condiciones de parada se cumplen, el individuo de mejor *fitness* de la población final es la que se proporciona como solución óptima al problema planteado (Corral Sastre, 2018).

1.6.7 Solución óptima

Una vez finalizado el AG, es necesario recordar que los AG proporcionan una solución óptima para el problema planteado, que no es necesariamente *la mejor* de todas las soluciones que pueden existir.

Antes de proseguir, quiero aclarar que los AG sí dan la mejor solución en aquellos problemas donde hay **una única solución** válida, como cuando se resuelve un Sudoku. Pero esto es porque, a lo que se le llama *mejor solución*, es realmente la única. En dichos casos, la condición de terminación del AG sería encontrar la única solución válida.

Pero, en la vida real, pocas veces los problemas tienen una única *mejor* solución, sino que existen **numerosas soluciones** que pueden satisfacer al problema.

Por ejemplo, si se quiere hacer un horario de clases para el profesorado de la Facultad de Farmacia, existen numerosas combinaciones que pueden valer para que los horarios de unos y otros estén coordinados. Para dicho problema se puede diseñar un AG que encuentre una de esas combinaciones válidas.

En ese AG, se indicarán distintos objetivos, con distintos grados de importancia. Algunos de ellos serán obligatorios o primarios, como que un mismo profesor no tenga asignadas dos clases diferentes a la misma hora; otros tendrán una prioridad menor o secundaria (como la preferencia de un profesor de dar clases en horario de mañana o tarde), pero que también se

desean satisfacer. Los objetivos secundarios pueden, y suelen, ser contradictorios, lo que abre un campo de muchas posibles soluciones.

El AG irá produciendo soluciones, a las que asignará un *fitness* en función del grado en que cumpla los objetivos (dando mayor puntuación a los objetivos llamados obligatorios). Dichas soluciones, siguiendo el esquema de los AG, serán seleccionadas, cruzadas y algunas mutadas. Además, se irán reemplazando unas a otras hasta que el AG finalice.

Una vez que el algoritmo termina, proporciona como solución a la de más alto *fitness*. Es decir, aquella que cumple en mayor grado los objetivos que se exigían. Es, por tanto, una solución óptima.

En el caso de los profesores, puede haber muchas soluciones que cumplan todos los objetivos primarios y el mismo número, alto, de objetivos secundarios. El AG encontrará, al menos, una de dichas soluciones, que dará por óptima. Como vemos, no es la mejor de todas (habría varias "mejores"), pero sí soluciona el problema.

Como vemos, la solución óptima es aquella que satisface el problema planteado (cubre los objetivos primarios) con un alto cumplimiento de los objetivos secundarios.

2. Metodología

La búsqueda de información para esta revisión bibliográfica se ha realizado entre los meses de febrero y abril de 2020.

Para la realización de la introducción, se han utilizado especialmente libros generalistas de literatura especializada para aquellos conceptos más generales como los relacionas con el esquema general de los AG. Principalmente, estos libros han sido consultados en la biblioteca de la Universidad de Sevilla a través del buscador FAMA. También se han buscado documentos relacionados a través del buscador Google Académico.

Para la parte correspondiente a resultados, se han utilizado diversas fuentes de datos para obtener la información necesaria lo más actualizada posible. Para ello, se han buscado los estudios más relevantes sobre el tema en Scopus, PubMed o ScienceDirect. Del mismo modo, también se ha utilizado Google Académico, que suele actualizar su base de datos con mayor frecuencia.

Se ha reducido el número de artículos en base al año de publicación utilizando solo los que han sido publicados en los últimos años, así mismo se han obviado artículos en idiomas distintos a inglés o español. También se han discriminado aquellos artículos o fuentes de información no revisadas a través de revisión por pares o aquellos cuyo contenido puede ser manipulado o modificado por cualquier tipo de usuario.

En la búsqueda de información, se ha dado prioridad a aquellos resultados que se correspondían con revisiones bibliográficas, publicaciones con alto índice de impacto, publicaciones con alto número de citas y a las publicaciones más recientes (10 a 15 años).

También se han utilizado, como fuente de información, los artículos de revisión (*reviews*) encontrados en la bibliografía para encontrar otros artículos cuando era necesario profundizar en algún ejemplo.

Las palabras clave utilizadas para la búsqueda de información en las bases de datos señaladas anteriormente fueron: *genetics algorithm*, *metaheuristics*, *optimization*, *health resources*; debiendo estar al menos dos de las palabras en el título o en el resumen. Para la búsqueda combinada de varias palabras claves se utilizó el operador boleano “AND”.

3. Resultados

En este apartado se van a comentar diferentes artículos donde se utilizan los AG como herramienta de optimización de diversos recursos sanitarios. La selección de los artículos se ha hecho con un doble objetivo: exponer situaciones donde los AG han demostrado su utilidad y dar a conocer algunas de las modificaciones más frecuentes que se operan sobre los AG para favorecer su adaptación a cada problema concreto.

3.1 Utilización de los AG para generar cronogramas

El uso más extendido de los AG en la optimización de recursos sanitaros es la generación de horarios. Concretamente, son ampliamente utilizados en los centros sanitarios para crear turnos eficientes. Dichos turnos pueden establecerse tanto para el orden de los pacientes, como los horarios del propio personal sanitario como enfermeras o médicos.

3.1.1 Pacientes

La programación de los turnos en que los diversos pacientes son atendidos desempeña un papel importante en el sistema de atención médica (Azadeh et al., 2014). Una buena organización reduce el tiempo de espera de los pacientes y facilita que los tratamientos se efectúen de forma consecutiva, sin interrupción.

En la mayoría de los casos, los pacientes que acuden a un hospital deben pasar por distintas áreas de este, con su consiguiente personal sanitario y equipo. Dicha utilización de los recursos por parte de cada paciente nos lleva a buscar un horario óptimo que garantice que todos los recursos estén ocupados al máximo de su capacidad y que los pacientes esperen lo mínimo en ser atendidos (Podgorelec and Kokol, 1997).

Dado que dicha programación del paciente es un problema altamente complejo, resulta casi imposible hacer un cronograma cualitativo a mano. Son situaciones con numerosos factores y restricciones, por lo que es preferible desarrollar un método de programación automatizado, como los AG (Podgorelec and Kokol, 1997).

A continuación, expondré el trabajo de Azadeh y colaboradores (Azadeh et al., 2014), como ejemplo de la aplicación de un AG clásico para resolver el problema de la creación de turnos adecuados en el ambiente hospitalario.

Azadeh y colaboradores (Azadeh et al., 2014) diseñaron un **AG clásico** para programar el turno de los pacientes que deben someterse a una o varias pruebas en los laboratorios del departamento de Urgencias de un hospital real. La rapidez con que cada paciente es atendido depende habitualmente del estado de salud del paciente, siendo considerados de mayor urgencia aquellos de mayor gravedad.

En su AG, cada **cromosoma** estaba constituido por una permutación de números enteros. Cada número se correspondía con un paciente concreto y un laboratorio determinado. Al conjunto de paciente con laboratorio lo denominaron *operación*. Cada cromosoma tendría tantos **genes** como *operaciones* se daban en el problema. Es decir, la longitud de la solución o cromosoma dependía en cada caso del número de pacientes y de los laboratorios por los que debían pasar.

Para hacer más visual la explicación anterior, suponemos que el AG debe crear un cronograma para 5 pacientes (I al V) y 3 laboratorios (A, B y C), donde 2 pacientes tienen pruebas en 3 laboratorios, 1 paciente en dos de ellos y los 2 restantes en 1 sólo laboratorio. El número asignado a cada *operación* viene representado en la tabla 1. Una posible solución a dicho problema podría ser generada por el AG como la Fig. 26.

Tabla 1: Asignación de un número entero, empezando por el 1, a cada *operación*.

		Laboratorios		
		A	B	C
Pacientes	I	**1**	**2**	**3**
	II	**4**	**5**	**6**
	III	**7**		**8**
	IV	**9**		
	V		**10**	

Figura 26: Posible cromosoma generado, que contiene todas las *operaciones*.

Las soluciones de estos AG contenía una secuencia ordenada de operaciones. Es decir, el cronograma generado por el AG seguía el orden de turnos que contenía en sus genes. Aquí, primero se llamaría al *paciente II* para el *laboratorio A* (primer gen con valor 4 de operación), luego al *paciente III* para el *laboratorio C* (segundo gen con valor 8 de operación), a continuación, *el paciente V* para el *laboratorio B* (tercer gen con valor 10 de operación)… y así sucesivamente.

Una vez que los cromosomas eran generados por el algoritmo, ya sea al generar la población inicial, bien tras finalizar el proceso de selección, cruce y mutación, eran evaluados por la **función de evaluación** que les asignaba un valor de *fitness*. La función de evaluación puntuaba cada cromosoma según el número de objetivos que cumplía y en su grado de importancia. La función de evaluación tenía en cuenta factores como la gravedad del paciente, que su tiempo de espera sea mínimo o que el mayor número de laboratorios estuviesen siendo utilizados, entre otros.

En este trabajo, Azadeh y colaboradores establecieron el **tamaño de población** en 51 individuos. La **selección** de los padres se hizo proporcional a su *fitness*, siguiendo el modelo habitual de "ruleta". El **cruzamiento** tenía lugar con un operador de cruce basado en un punto, con una probabilidad de 0,29. Para la **mutación**, siguieron el modelo de mutación estándar, donde el operador de mutación realizaba cambios con una probabilidad de 0,29. Finalmente, establecieron como **condición de parada** que el AG produjera 60 generaciones.

Una vez que se determinan todos los parámetros del AG, se procede a la evaluación del AG. Es decir, se comprueba si el AG cumple con el objetivo para el que se ha diseñado. La evaluación consta de dos fases.

Primero, se compara el AG diseñado con un algoritmo de optimización *patrón*. El patrón, que no es genético, encuentra soluciones adecuadas en problemas de optimización, pero con el inconveniente de que se ralentiza según aumenta el tamaño de población. Por tanto, el AG propuesto se considera adecuado cuando encuentra soluciones similares al patrón (óptimas), pero en menor tiempo.

Para ello, resolvieron 10 instancias de problemas generados aleatoriamente con diferentes tamaños. Los problemas fueron resueltos por el AG y por el algoritmo patrón. Se estableció un tiempo máximo de ejecución de 10 horas. Los resultados mostraron que en los problemas donde la población era pequeña, el AG y el patrón producían soluciones parecidas y en tiempos similares. Sin embargo, a medida que aumentaba el tamaño de población, el patrón se ralentizaba considerablemente (dando como solución óptima la mejor encontrada tras 10 horas de ejecución), mientras que el AG continuaba encontrando soluciones parecidas, pero en un tiempo apenas aumentado (Fig. 27). En conclusión, el AG demostró ser superior al patrón, por lo que su aplicación en este problema suponía una mejora respecto a lo que ya había.

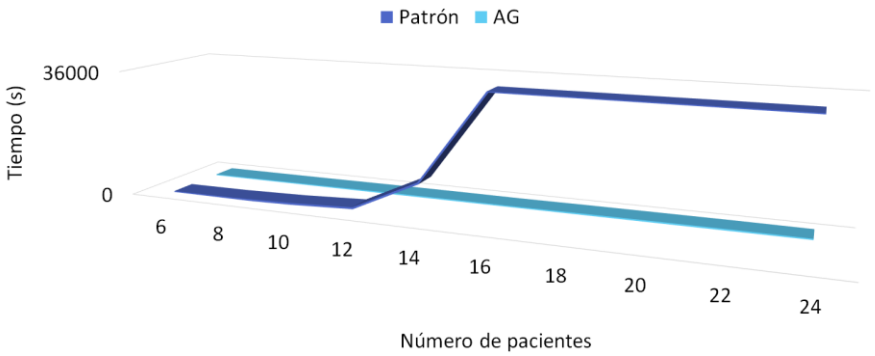


Figura 27: Comparación del tiempo de ejecución requerido por el patrón y el AG para encontrar la solución.

Para todos los AG, la segunda fase de evaluación consiste en la aplicación del AG a un caso real. Para ello, Azadeh y colaboradores realizaron una simulación del funcionamiento real del departamento de Urgencias de un hospital. Tras comprobar que el simulador reflejaba fielmente el funcionamiento diario del hospital, se estableció un caso. Dicho caso se introdujo en el AG, que generó una solución optimizada. Para comprobar si era efectivo, se hizo actuar al simulador del hospital dos veces para cada problema: la primera vez, el simulador se ejecutó siguiendo los turnos de pacientes del modelo basado en el hospital real; la segunda vez, siguió el orden de pacientes que determinó el AG para ese caso. Los resultados de la simulación mostraron que el AG propuesto mejoró significativamente la eficiencia del departamento de Urgencias al reducir el tiempo de espera total de los pacientes priorizados.

El esquema que se ha expuesto en este ejemplo es común a todos los artículos que utilizan AG clásicos para resolver cronogramas de centros sanitarios, con pequeñas variaciones. Sin embargo, es muy habitual que el AG no se utilice de forma aislada, sino que sea combinado con otros algoritmos que mejoren su búsqueda.

AG COMBINADO CON UN ALGORITMO DE APRENDIZAJE AUTOMÁTICO O AAA PARA PRODUCIR SOLUCIONES SIEMPRE FACTIBLES

Un ejemplo del uso de AG combinados con otros algoritmos es el caso de Podgorelec y Kokol (Podgorelec and Kokol, 1997). Ellos desarrollaron un método para programar la atención de pacientes con distintas necesidades de fisioterapia cuando el número de terapeutas y dispositivos era muy limitado, haciendo más difícil que el AG encontrase soluciones factibles. Para resolver dicho inconveniente, optaron por combinar un AG clásico con un **algoritmo de aprendizaje automático** (**AAA**).

En su caso, el algoritmo de aprendizaje actuaba sobre el AG con el fin de transformar todas las soluciones para que fueran siempre factibles. El AAA intervenía en el AG cada vez que éste producía soluciones no factibles, descartando dichas soluciones y forzando al AG a repetir el proceso hasta que generase el total de la población con soluciones factibles. Dicho proceso podía ser el de *inicialización*, o el de *cruce y mutación*.

El esquema clásico del AG sigue siendo necesario porque, recordemos, que una solución sea factible no significa necesariamente que sea lo suficientemente buena como para ser utilizada. Los resultados mostraron que, al realizar dicha combinación, el AG fue capaz de finalizar con soluciones optimizadas para ese problema.

COMBINACIÓN DE TRES AAA PARA CONFORMAR LA FUNCIÓN DE EVALUACIÓN DEL AG

En el trabajo de Yousefi y colaboradores (Yousefi et al., 2018) encontramos la misma idea de combinar un AG con otros AAA. En su estudio, buscaron un método que garantizase la óptima planificación de los recursos en los departamentos de Urgencias en tiempo real, desde la perspectiva de mejorar los turnos en que los pacientes son atendidos. Para ello, propusieron la combinación de tres AAA con el algoritmo genético.

En este caso, los tres AAA actuaban simultáneamente para conformar la *función de evaluación* del AG. El triple AAA realizaba la evaluación en base a su experiencia acumulada. Esta experiencia no se debía únicamente a la adquirida durante el funcionamiento del AG, sino que utilizaba además un conocimiento previo que había obtenido de una simulación anterior del propio departamento de Urgencias del hospital en que actuaba. De esta forma, el triple AAA asignaba el valor de *fitness* a cada solución en base al parecido que tuviese con aquellas que ya conocían. Las soluciones que se parecían a otras antiguas que se tradujeron en una optimización del proceso, recibían ahora una mejor puntuación. A partir de dicho valor de *fitness*, el AG continuaba su funcionamiento habitual.

El método que desarrollaron se probó en el departamento de Urgencias de dicho hospital. Al aplicar el AG combinado con los tres AAA, los resultados mostraron una optimización en el funcionamiento de dicho departamento. Esta optimización resultó especialmente evidente en aquellas áreas de Urgencias donde habitualmente se producían congestiones, porque el AG combinado consiguió mejorar el flujo de pacientes.

3.1.2 Enfermeras

A la hora de mejorar el rendimiento del funcionamiento del hospital, se puede optar también por optimizar los turnos del propio personal sanitario. El departamento de Urgencias es la primera línea de atención de emergencias en un hospital. Desafortunadamente, es también el primero en sufrir el problema de escasez de personal, especialmente de enfermeras, siendo criticado habitualmente por la disminución de la calidad del servicio (Wong et al., 2014).

Sin embargo, como Yeh y Lin demuestran en su informe (Yeh and Lin, 2007), se puede mejorar la calidad del servicio en los departamentos de Urgencias sin contratar personal adicional al utilizar un algoritmo genético (AG) que ajuste adecuadamente los horarios de las enfermeras.

Ellos desarrollan un AG clásico para obtener un programa de enfermería casi óptimo, similar al desarrollado por Azadeh y colaboradores (Azadeh et al., 2014) para crear turnos de pacientes. Los resultados computacionales, para los que también utilizaron una simulación, mostraron que los horarios de enfermería generados por el AG significaron una reducción del tiempo promedio de espera de los pacientes casi a la mitad. Sus resultados implican que la atención de calidad en el servicio de Urgencias puede ser realmente mejorado haciendo ajustes a los horarios de las enfermeras sin aumentar su número en el sistema (Yeh and Lin, 2007).

Sin embargo, aunque Yeh y Lin optan por un AG clásico, no es lo habitual en este tipo de problemas. En la mayoría de los hospitales, los tipos de contratos de trabajo para las enfermeras genera una gran variedad de objetivos con distintos grados, de restricciones complejas, que complican la búsqueda de parámetros adecuados para el AG.

AG INDIRECTO ACOPLADO A UN DECODIFICADOR COMPLEJO

Debido a la cantidad de variables que hay que tener en cuenta, el problema del horario de enfermeras es realmente complejo. Como prioritario se encuentra que el cronograma debe cumplir con los contratos de trabajo, satisfacer la demanda de un número determinado de enfermeras de diferentes grados en cada turno y tiene que ser considerado justo por las propias enfermeras (Aickelin and Dowsland, 2004). Además, hay otros objetivos, secundarios pero deseables, como que se tenga en cuenta la mayor cantidad de solicitudes de preferencias de enfermeras o que se garantice que los turnos impopulares se distribuyan de manera uniforme.

El problema se complica aún más porque las enfermeras de alto grado o calificación pueden sustituir, si fuera necesario, a las de grado inferior; pero no a la inversa (Aickelin and Dowsland, 2004). Además, la mayoría de las enfermeras están contratadas para trabajar días o noches en una semana, pero no ambas. Y debido a los contratos de trabajo, la cantidad de días para trabajar no suele ser igual a la de noches.

Esta variedad de objetivos y restricciones con las que los hospitales deben contar al hacer los cronogramas de las enfermeras resulta uno de los mayores inconvenientes a la hora de encontrar un AG adecuado (Aickelin and Dowsland, 2004) . Esto es porque, realmente, no existe una forma predefinida de incluir restricciones en los AG.

En su trabajo, Aickelin y Dowsland (Aickelin and Dowsland, 2004) deciden desarrollar un AG flexible, de forma que pueda ser fácilmente adaptado a cualquier hospital para generar horarios de enfermeras.

Puesto que las restricciones de cada AG son propias del problema concreto que están tratando de solucionar, deciden que *el AG funcione sin ninguna restricción específica* pero que se acople a un decodificador propio del problema.

El ***decodificador*** es siempre un traductor del AG, que convierte el genotipo al fenotipo. Interviene cada vez que se ejecuta la función de evaluación y al finalizar el algoritmo. Lo habitual en los AG es tener un decodificador simple, es decir, que traduce sin más consideraciones. El *decodificador simple* funcionaría, por ejemplo, transformando un genotipo binario a un fenotipo decimal (Fig. 28).

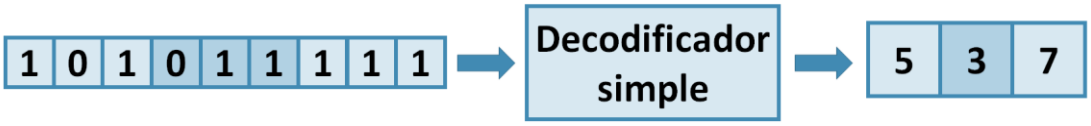


Figura 28: Decodificador simple que traduce un código binario a decimal.

Sin embargo, el decodificador que proponen Aickelin y Dowsland es novedoso porque, aunque traduce el genotipo, lo hace con ciertas restricciones. En lugar de hacer la habitual traducción de genotipo a su fenotipo propio, dirige la conversión del genotipo a ciertos grupos de fenotipos (aquellos que cumplan las restricciones indicadas en el decodificador). Es decir, proponen un *decodificador degenerado*, donde distintos genotipos podrían corresponderse a un mismo fenotipo.

Para explicar dicho decodificador complejo, lo asemejaremos al proceso de traducción celular. En la traducción celular, el ARNt actúa como un decodificador degenerado porque existen más tripletes o codones que aminoácidos (Fig. 29). De forma análoga, el decodificador degenerado de Aickelin y Dowsland permite que existan más combinaciones de números que posibles soluciones.

Figura 29: Decodificador degenerado que traduce en base a unas restricciones. En este caso, decodificador genético que traduce codones a aminoácidos.

Además del decodificador degenerado, para que el AG sea flexible debe utilizar operadores de cruce y mutación de los que llamamos *complejos* (algunos mencionados en la introducción de este trabajo).

Con ello, el AG resuelve un problema sin restricciones dejando el manejo de restricciones al decodificador, que las usa para sesgar directamente la búsqueda. De esta forma, el AG actúa intentando encontrar el mejor orden posible de las enfermeras, y dicho orden sería luego traducido por el decodificador con sus restricciones generando así la solución real.

Al probar la combinación del AG con el decodificador degenerado, obtienen resultados de la suficiente calidad como para considerarlo una buena opción para resolver el problema de la programación de horarios de enfermeras. Además, la flexibilidad que otorga el decodificador al AG permite que pueda ser adaptado en otros problemas complejos con enfoques similares, y deja abierta la posibilidad a investigaciones futuras para otras áreas.

Como hemos visto, un buen horario puede contribuir a mejorar la eficiencia del trabajo y la experiencia del paciente, al mejorar el bienestar de la enfermera y la distribución de la carga de trabajo. Además, permite a los encargados de la toma de decisiones del departamento de Urgencias gestionar mejor los escasos recursos del departamento (Wong et al., 2014).

3.1.3 Médicos

Aunque el problema de diseñar horarios para el personal sanitario del departamento de Urgencias hospitalario se ha estudiado ampliamente en los colectivos de enfermeras, no se ha abordado de manera profusa dentro del personal médico, a pesar de la creciente carga de trabajo y el consiguiente aumento del personal en estos servicios hospitalarios (Puente et al., 2009).

Observando la necesidad creciente de optimizar los recursos de dichos departamentos en los hospitales y la carencia de estudios referidas a la rotación de turnos del personal médico, Puente y colaboradores (Puente et al., 2009) deciden crear un AG que automatice la generación de un cronograma para los médicos de Urgencias.

El AG que diseñan difiera del AG clásico en ciertas singularidades. La generación de la población inicial no es aleatoria, sino que está sesgada para que sólo considere genere soluciones factibles. Además, diseñan un operador de cruce específico para este tipo de problemas, que sólo intercambia aquellos fragmentos del cromosoma que incluyan semanas de trabajo completas, no pudiendo intercambiarse días sueltos. Por último, incluyen una ***función de reparación*** que transforma aquellas soluciones, obtenidas tras el cruce y mutación, que no sean factibles para que sí lo sean.

Al aplicar este modelo en el departamento de Urgencias de un hospital de España, los resultados muestran que el AG genera soluciones óptimas que se traducen en una asignación de turnos más equilibrada entre los médicos, cumpliendo los requisitos de los contratos de trabajo. Por ello, su implementación actual en dicho departamento se ha realizado con un alto grado de satisfacción.

Sin embargo, finalizan su estudio dejando abierta la puerta a futuras investigaciones. Remarcan la necesidad de continuar la investigación en encontrar nuevos operadores de cruce y mutación que sean aún específicos del problema de cronogramas médicos. Del mismo modo, señalan que una reestructuración del AG que proponen puede satisfacer las crecientes necesidades de este tipo de servicios hospitalarios, como opción a su pronta aplicación para la optimización de los recursos.

3.2 Utilización de los AG para encontrar ubicaciones óptimas en un espacio geográfico

La principal aplicación de los AG para optimizar recursos hospitalarios es, como hemos desarrollado, la automatización de generar cronogramas adecuados que permitan el mayor aprovechamiento de los recursos limitados que los hospitales poseen, especialmente necesario en el departamento de Urgencias.

Sin embargo, los AG también suelen utilizarse para encontrar localizaciones óptimas en un espacio geográfico. Basándose en esta idea, Sasaki y colaboradores (Sasaki et al., 2010) deciden crear un AG que sea capaz de encontrar ubicaciones potencialmente eficientes de ambulancias dentro de una ciudad. Con ello, aprovechan una aplicación habitual de los AG para un problema de optimización de recursos sanitarios.

Los estudios sobre el efecto de la reducción de los tiempos de respuesta de la ambulancia sobre las tasas de supervivencia han argumentado que el riesgo de mortalidad es sensible a los tiempos de respuesta de la ambulancia en menos de *5 minutos* (Blackwell and Kaufma, 2002), siendo crucial para la supervivencia el tiempo de respuesta en dichas situaciones. En su trabajo, Sasaki y colaboradores (Sasaki et al., 2010) proponen un AG para la reasignación de ambulancias hacia ubicaciones óptimas, con el objetivo de reducir su tiempo de llegada a 5 minutos.

Realizan su estudio sobre la ciudad de Niigata, capital de la prefectura de Niigata, en Japón. En este país, las ambulancias suelen ubicarse en las estación de bomberos. El despliegue de una ambulancia está controlado por un sistema de comando central. Cuando se recibe una llamada de emergencia, la ambulancia disponible más cercana a la escena se despliega en el incidente.

En la ciudad de Niigata hay 35 estaciones de bomberos (ubicaciones potenciales de ambulancias) (Fig. 27a) y un total de 27 ambulancias, que, en el momento del estudio, se encontraban en las estaciones indicadas en la Fig. 27b. Para la Fig. 27b, se ha representado en *azul* las estaciones de bomberos que cuentan con ambulancia y en *naranja* aquellas que no.

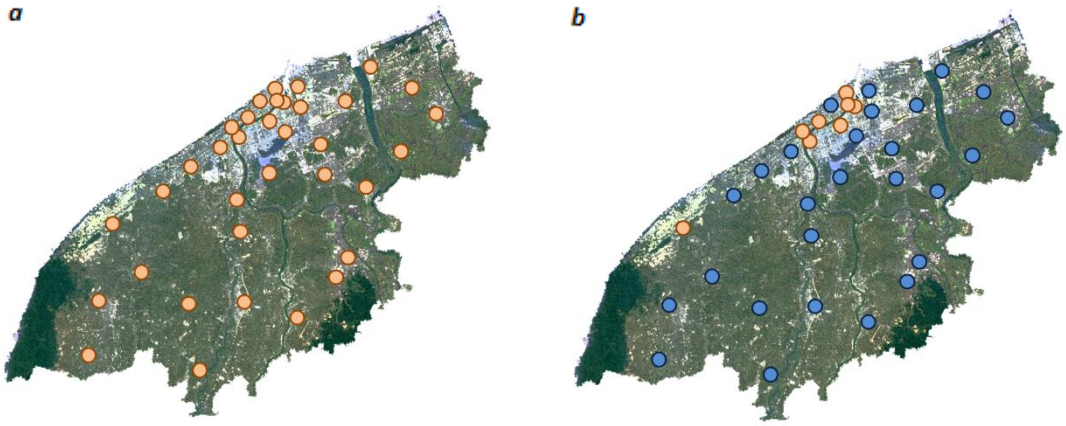


Figura 27: (**a**) Distribución de las 35 estaciones de bomberos (en *naranja*). (**b**) Ubicación actual de las 27 ambulancias (en *azul*).

Realizaron un análisis de red para calcular las distancias y tiempos de viaje entre ubicaciones o puntos en redes lineales o carreteras y generar así una matriz de distancias. También utilizaron un registro de varios meses de 2007 sobre emergencias médicas prehospitalarias (donde actúan las ambulancias) para conocer la situación real. Además, utilizaron dicho registro para realizar un pronóstico, en base a datos del denso demográfico y un programa probabilístico, sobre las posibles demandas de emergencia futuras.

Dicha matriz, junto con los recuentos reales y pronosticados de emergencias sanitarias donde hiciera falta una ambulancia, se usaron como entrada en el AG, para las 35 ubicaciones potenciales de ambulancias. Al ejecutar el AG, se obtienen como localizaciones óptimas de ambulancias en las estaciones de bomberos, las indicadas en las Figs. siguientes, para el año del estudio (Fig. 28a) y la situación futura pronosticada (Fig. 28b). Se ha utilizado el color *azul* para indicar aquellas estaciones ya contaban con una ambulancia, el color *verde* para indicar aquellas estaciones de bomberos donde no había ambulancia, pero el AG ha considerado que debía haber; y en *rojo* aquellas estaciones que contaban con ambulancia, pero el AG ha descartado como ubicación óptima de las mismas.

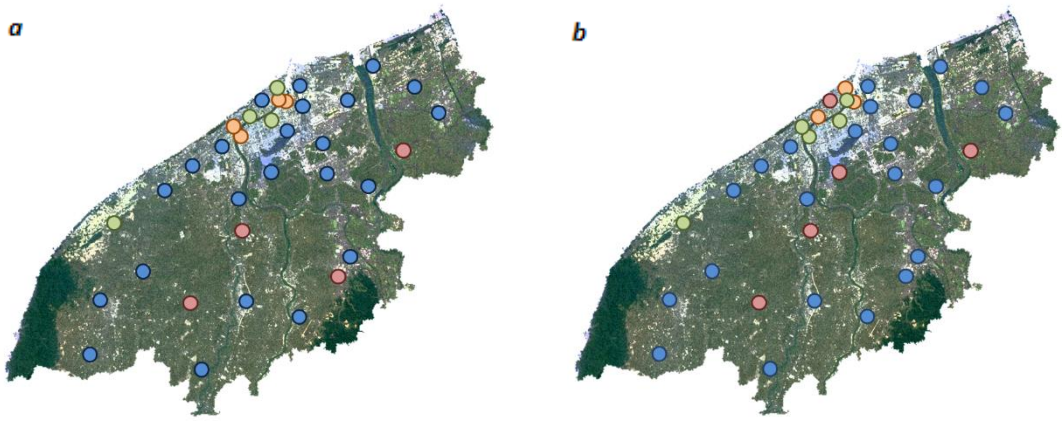


Figura 28: (**a**) Ubicación óptima de 27 ambulancias según casos de emergencia en 2007. (**b**) Ubicación óptima de 27 ambulancias según casos pronosticados para 2030

Es decir, el AG ha reasignado las ambulancias que ya había, en las posiciones que ha considerado óptimas para cubrir la demanda de emergencias prehospitalarias. Los resultados del estudio muestran que la reasignación de las 27 ambulancias a su localización óptima redujo los tiempos de respuesta promedio estimados de 5 minutos 21 segundos a 4 minutos 24 segundos. Además, a cada punto donde se genera una emergencia le corresponde la ambulancia de la estación de bomberos más cercana. Si dicha ambulancia se encuentra atendiendo otra emergencia, esta nueva emergencia es atendida por la ambulancia de otra estación. Con el AG se consiguió aumentar los porcentajes de despliegue de la ambulancia más cercana a la escena del 67,8 % al 83,0 %. Los porcentajes de despliegue de la ambulancia más cercana y los tiempos de respuesta reducidos indican una utilización más eficiente de la ambulancia.

4. Conclusiones

Los AG son un método adecuado para resolver problemas complejos donde el número de posibles soluciones es demasiado grande como para que todas ellas sean evaluadas.

Su eficacia se debe a su inspiración en la evolución natural. Al imitar cíclicamente los procesos de selección, cruce y mutación, favoreciendo a aquellos individuos o soluciones con mejor adaptación, el AG es capaz de finalizar encontrando una solución mejor adaptada u óptima al problema planteado.

Los AG poseen una gran versatilidad, pudiendo sufrir pequeñas modificaciones en su esquema de funcionamiento para adaptarse al problema que se quiere resolver.

Varios estudios han demostrado que el uso de los AG en el ámbito sanitario es adecuado para optimizar problemas de fácil planteamiento, pero difícil resolución.

Su uso más común es de la generación de horarios. En la optimización de recursos sanitarios, son empleados con frecuencia para generar los turnos de pacientes, del personal de enfermería o del propio personal médico para agilizar el funcionamiento del centro hospitalario donde se apliquen.

También se emplean con frecuencia para encontrar puntos óptimos de un espacio geográfico. En el ámbito sanitario, se ha utilizado para encontrar localizaciones óptimas donde ubicar ambulancias en una ciudad y garantizar así una mejor cobertura.

En dichos ejemplos, los AG han mantenido su esquema clásico o bien han sufrido pequeñas modificaciones. Las modificaciones más habituales son las aplicadas sobre los operadores de cruce y mutación o sobre la propia función de evaluación. Sin embargo, cualquiera de los pasos del AG es susceptible de ser adaptado.

Por tanto, aunque las aplicaciones más frecuentes de los AG para optimizar los recursos sanitarios son los aquí presentados, los AG son susceptibles de ser aplicados a cualquier problema de optimización. Es decir, los AG presentan un campo de posibilidades abierto a futuras investigaciones.

5. Bibliografía

Aickelin U, Dowsland K. An indirect Genetic Algorithm for a nurse–scheduling problem. Comput Oper Res. 2004; 31(5): 761–778.

Alander JT. On optimal population size of genetic algorithms. In: CompEuro 1992 Proceedings Computer Systems and Software Engineering; 1992 May 4–8; The Hague, Netherlands. IEEE; 1992. p. 65–70.

Azadeh A, Hosseinabadi Farahani M, Torabzadeh S, Baghersad M. Scheduling prioritized patients in emergency department laboratories. Comput Methods Programs Biomed. 2014; 117(2): 61–70.

Baker JE. Reducing bias and inefficiency in the selection algorithm. In: John J. Grefenstette ed. Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application; 1987 Oct; Cambridge, MA. Broadway Hillsdale, NJ: L. Erlbaum Associates Inc.; 1987. p. 14–21.

Banzhaf W. The 'molecular' Traveling Salesman. Biol Cybern. 1990; 64(1): 7–14.

Beasley D, Bull DR, Martin RR. An overview of Genetic Algorithms: Part 1, fundamentals. Univ. Comput. 1993; 15(2): 58–69.

Blackwell TH, Kaufman JS. Response Time Effectiveness: Comparison of response time and survival in an Urban Emergency Medical Services System. Academic Emergency Medicine. 2002; 9(4): 288–295.

Casado Marín D. Los efectos del envejecimiento demográfico sobre el gasto sanitario: mitos y realidades. Gac Sanit. 2001; 15(2): 154–163.

Corral Sastre A. Desarrollo y evaluación de Algoritmos Genéticos Multiobjetivo. Aplicación al problema del viajante [Trabajo Fin de Grado]. Escuela Técnica Superior de Ingeniería Informática, Universidad Politécnica de Valencia; 2018.

Darwin C. On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life. London: John Murray; 1859.

De Jong KA. An analysis of the behaviour of a class of genetic adaptative systems [Thesis]. Computer and Comunication Sciences Department, University of Michigan; 1975.

Duarte Muñoz A, Pantrigo Fernández J, Gallego Carrillo M. Metaheurísticas. Móstoles, Madrid: Dykinson; 2007: 81–84.

Duran I. Introducción a los Algoritmos Genéticos [Apuntes de clase]. Universidad Autónoma de Barcelona; apuntes proporcionados en las clases dadas del 1–11 de mayo de 2016.

Fogel DB. An evolutionary approach to the Traveling Salesman Problem. Biol Cybern. 1988; 60(2): 139–144.

Fogel DB. Applying evolutionary programming to selected Traveling Salesman Problems. Cybern Syst. 1993; 24(1): 27–36.

Ghaheri A, Shoar S, Naderan M, Hoseini SS. The applications of Genetic Algorithms in Medicine. Oman Med J. 2015; 30(6): 406–416.

Goldberg DE, Lingle R. Alleles Loci and the Traveling Salesman Problem. In: John J. Grefenstette ed. Proceedings of the 1st International Conference on Genetic Algorithms; 1985 July; Pittsburgh. Broadway Hillsdale: L. Erlbaum Associates Inc.; 1985. p. 154–159.

Goldberg DE. Genetic Algorithms in search, optimization, and machine learning. Boston: Addison–Wesley; 1989.

Grefenstette JJ, Gopal R, Rosmaita BJ, Gucht DV. Genetic Algorithms for the Traveling Salesman Problem. In: Grefenstette JJ, eds. Proceedings of the 1st International Conference on Genetic Algorithms; 1985 July; Pittsburgh. Broadway Hillsdale: L. Erlbaum Associates Inc.; 1985. p. 160–168.

Grefenstette JJ. Optimization of control parameters for Genetic Algorithms. IEEE Trans Syst Man Cybern. 1986; 16(1): 122–128.

Hassanien AE, Grosan C, Tolba FM. Applications of Intelligent Optimization in Biology and Medicine. Cham: Springer International Publishing; 2016.

Holland JH. Adaptation in natural and artificial systems: An introductory analysis with applications to Biology, Control, and Artificial Intelligence. Ann Arbor, Mich.: University of Michigan Press; 1975.

INE. Principales series desde 1971: Subelementos Resultados Nacionales: Población residente por fecha, sexo y edad [en línea]. [Consultado en febrero de 2020]. Disponible en: https://www.ine.es/dynt3/inebase/index.htm?padre=1949&capsel=1950#.

Kuri Morales A, Galaviz Casas J. Algoritmos Genéticos. México D. F: Instituto Politécnico Nacional; 2002.

Marco Ramo E. Recursos sociales y/o sanitarios disponibles para los cuidadores no profesionales de mayores dependientes en Soria [Trabajo Fin de Grado]. Facultad de Enfermería de Soria, Universidad de Valladolid; 2019.

Michalewicz Z. Genetic Algorithms + Data structures = Evolution programs. Berlin: Springer-Verlag; 1999.

Michalewicz Z, Fogel DB. How to solve it: Modern Heuristics. Berlin: Springer; 2000.

Mitchell M. An introduction to Genetic Algorithms. Cambridge, Mass.: MIT Press; 1996.

Podgorelec V, Kokol P. Genetic Algorithm based system for patient scheduling in highly constrained situations. J Med Syst. 1997; 21(6): 417–427.

Puente J, Gómez A, Fernández I, Priore P. Medical doctor rostering problem in a hospital emergency department by means of Genetic Algorithms. Comput Ind Eng. 2009; 56(4): 1232–1242.

Reeves C. Genetic Algorithms. In: Glover FW, Kochenberger GA, eds. Handbook of Metaheuristics. 1st ed. New York: Springer; 2003. p. 55–82.

Ribas E, Portella E. Optimizar los recursos y la gestión de los servicios sanitarios. En: Álvarez C, Peiró S, eds. Informe SESPAS 2000: La salud pública ante los desafíos de un nuevo siglo. Granada: Escuela Andaluza de Salud Pública; 2000: 357–361.

Sasaki S, Comber AJ, Suzuki H, Brunsdon C. Using genetic algorithms to optimise current and future health planning – the example of ambulance locations. Int J Health Geogr. 2010; 9(1): 4.

Syswerda G. Schedule optimization using genetic algorithms. In: Davis L, ed. Handbook of Genetic Algorithms. New York: Van Nostrand Reinhold; 1991. p. 332–349.

Moujahid A, Iñaki Inza I, Larrañaga P. Tema 2: Algoritmos Genéticos [Apuntes de clase]. Universidad del País Vasco – Euskal Herriko Unibertsitatea, Departamento de Ciencias de la Computación e Inteligencia Artificial; apuntes proporcionados en febrero de 2018.

Voet D, Voet J. Biochemistry. 4th ed. Hoboken, N.J.: John Wiley and Sons; 2010.

Wong TC, Xu M, Chin KS. A two–stage heuristic approach for nurse scheduling problem: A case study in an emergency department. Comput Oper Res. 2014; 51: 99–110.

Yeh JY, Lin WS. Using simulation technique and genetic algorithm to improve the quality care of a hospital emergency department. Expert Syst Appl. 2007; 32(4): 1073–1083.

Yousefi M, Yousefi M, Ferreira RPM, Kim JH, Fogliatto FS. Chaotic genetic algorithm and Adaboost ensemble metamodeling approach for optimum resource planning in emergency departments. Artif Intell Med. 2018; 84: 23–33.