TRABAJO FIN DE MÁSTER

Máster en Ingeniería Informática

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

Resolución de problemas de optimización en la industria farmacéutica mediante algoritmos evolutivos

Curso 2016/2017

Alumno:

Savíns Puertas Martín

Director/es:

Pilar Martínez Ortigosa

Juana López Redondo

EDITADO CON LaTeX.

Plantilla original de Mathias Legrand (http://www.latextemplates.com).
Modificada por Savíns Puertas Martín.

Dedicado a mis padres

Agradecimientos

Me gustaría dedicar estas primeras líneas a mis directoras Pilar Martínez Ortigosa y Juani López Redondo. Si pudiese considerar al grupo de investigación TIC-146: Supercomputación y Algoritmos como una habitación con una puerta, Pilar sería la llave maestra que me permitió encontrar una pequeña caja mágica en la que residía Juani. No se puede entender este trabajo sin el esfuerzo, tiempo y dedicación que han consumido por y para mí. Desde mi punto de vista, su apoyo, presencia y trabajo lo veo como el yin y el yang, dos personas que se complementan a la perfección: la tranquilidad con la vivacidad, el concepto con el detalle, el respaldo con el escudo protector. Si una de ellas me hubiese faltado, hoy seguramente no estaría escribiendo esta memoria. Pero no sólo quiero agradecerles el haberme guiado, aconsejado y/o ayudado en este trabajo y en mi formación como investigador, sino también la cercanía que han tenido conmigo a nivel personal, donde han sido, más que unas directoras, unas compañeras increíbles. Además, a nivel profesional me siento sumamente orgulloso de tenerlas como espejo en mi progresión porque si en algún momento de mi vida consigo lo que ellas han alcanzado, significará que seré un gran investigador.

Tampoco me puedo olvidar de mi codirector (oficialmente de doctorado) Horacio, en la cercana Murcia, quién ha guiado mis pasos en el ámbito, desconocido por entonces para mí, de la química y los fármacos. Me ha proporcionado todo su conocimiento y recursos disponibles para conseguir estar hoy escribiendo estas palabras y cuyo trabajo, al igual que el de Pilar y Juani, continuará en el desarrollo de mi doctorado.

Existe el refrán español: “No es más rico quien más tiene, sino quien menos necesita”. Basándome en dicho refrán, puedo decir que mis padres me han hecho, y continúan haciéndolo día a día, la persona más rica del mundo. No me da tiempo a pedirles tranquilidad, apoyo, alguna necesidad material y/o consejo cuando ellos ya me lo han proporcionado y de la mejor manera posible. En ese sentido, este trabajo también es gracias a ellos por su esfuerzo, trabajo y dedicación.

Si Pilar me abrió la puerta de una habitación, ésta tenía que existir previamente y es por eso que tengo que dar las gracias al grupo TIC-146: Supercomputación y Algoritmos por acogerme, hacerme un hueco y permitirme crecer como investigador entre vosotros. Muchas gracias a Ester Martín, Leo González, Vicente González, Inma García y Eligius Hendrix.

Pertenecientes también al grupo de investigación, no me puedo olvidar de los residentes del Corredor Uno Punto: Miriam Ruiz, Nicolás Calvo, Cristóbal Medina, José Manuel García, Juanjo Moreno, Gloria Ortega, Francisco Orts, Juan Francisco Rodríguez y Gabi Barrionuevo. Estamos los que somos y somos los que estamos, compartiendo además de cada día de trabajo, alegrías y tristezas, problemas y soluciones, asistencias a congresos y ayudándonos en todo lo necesario. Y entre ellos me gustaría destacar especialmente a Miriam, cuya compañía en el despacho hace más ameno el día a día y dónde siempre he podido encontrar a una persona a la que contarle lo bien o mal que iban los trabajos o las reuniones, las últimas noticias o simplemente lo que se hizo el fin de semana anterior.

También quiero recordar el tiempo que me han dedicado otras personas, que si bien han sido situaciones puntuales, han puesto su granito de arena para que esta montaña se construyera. En ese sentido, le doy las gracias a David Llena, Jorge de la Peña, Helena den Haan, Ricardo

Rodríguez, Baldo Imbernón y Mathias Legrand.

Y por último me gustaría dar gracias a Dios por estar siempre conmigo y hacer mi vida de esta manera.

Para terminar, quiero dejar por escrito mi gratitud a las instituciones que soportan a nivel económico y material (además del equipo tecnológico del grupo TIC-146) mi investigación, ya que si bien, ésta es la memoria de un Trabajo Fin de Máster que comenzó con una beca de colaboración, la intención es convertirlo dentro de unos años en un gran trabajo de doctorado. Por tanto me gustaría decir, aunque haya sido indirectamente, que este trabajo ha sido financiado por el Ministerio de Economía y Competitividad de España (TIN2015-66680-C2-1-R), la Junta de Andalucía (P11-TIC7176 y P12-TIC301), la Fundación Séneca–Agencia de Ciencia y Tecnología de la Región de Murcia bajo los proyectos 19419/PI/14 y 18946/JLI/13, y por el Nils Coordinated Mobility bajo la subvención 012-ABEL-CM-2014A, en parte financiada por el Fondo Europeo de Desarrollo Regional (FEDER). Powered@NLHPC: La investigación realizada ha sido parcialmente soportada por la infraestructura de supercomputación del NLHPC (ECM-02). También agradezco los recursos informáticos y el soporte técnico proporcionado por la Plataforma Andaluza de Bioinformática de la Universidad de Málaga. Este trabajo ha sido parcialmente apoyado por las instalaciones informáticas del Centro Extremeño de Tecnologías Avanzadas (CETA-CIEMAT), fundado por el Fondo Europeo de Desarrollo Regional (FEDER). CETA-CIEMAT pertenece a CIEMAT y al Gobierno de España. También decir que mi directora Juani López Redondo es una beneficiaria del programa español ‘Ramón y Cajal’, cofinanciado por el European Social Fund y yo, Savíns Puertas Martín, soy un beneficiario del programa español ‘Formación de profesorado universitario’, financiado por el Ministerio de Educación, Cultura y Deporte.

Contribución científica

Este trabajo se ha difundido (o ha sido aceptado para su futura difusión) en varios eventos, que se enumeran a continuación:

S. Puertas-Martín, H. Den-Haan, J. L. Redondo, H. Perez-Sanchez and P. M. Ortigosa. **Enhancing Molecular Shape Comparison by a Global Evolutionary Algorithm**. *4th International Work-Conference on Bioinformatics and Biomedical Engineering*, University of Granada, Granada, Spain, 20-22 April 2016.

S. Puertas-Martín, J. L. Redondo, H. Den-Haan, H. Pérez-Sánchez and P. M. Ortigosa. **Multiobjective Based Scoring Function for Ligand Based Virtual Screening**. *Proceedings of the XIII Global Optimization Workshop GOW'16*, University of Minho, Braga, Portugal, pp. 105-108, 4-8 September 2016.

S. Puertas-Martín, J. L. Redondo, H. Den-Haan, H. Pérez-Sánchez y P. M. Ortigosa. **Algoritmo evolutivo global como herramienta de cribado virtual utilizando la forma molecular**. *III Jornadas Doctorales de la Universidad de Murcia*, Universidad de Murcia, Murcia, España, 30, 31 Mayo y 1 Junio 2017.

S. Puertas-Martín, M. R. Ferrández, J. L. Redondo, H. Pérez-Sánchez and P. M. Ortigosa. **Enhancing Molecular Shape Comparison by a Parallel Global Evolutionary Algorithm**. *Proceedings of the 17th International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE 2017*, Rota, Cadiz - Spain, 4-8 July 2017.

S. Puertas-Martín, M. R. Ferrández, J. L. Redondo, H. Pérez-Sánchez and P. M. Ortigosa. **Cribado virtual mediante un algoritmo evolutivo global paralelo**. *XXVIII Jornadas de Paralelismo (JP2017)*, Málaga- España, 19-22 Septiembre 2017.

El presente TFM se enmarca dentro de una las líneas de investigación del proyecto del plan nacional I+D+I Metodologías computacionales para desafíos de la sociedad (ref. TIN2015-66680-C2-1-R) del Ministerior de Economía y Competitividad que desarrolla el Grupo de Investigación Supercomputación y Algoritmos TIC-146 de la Universidad de Almería.

Índice general

Índice de figuras

Índice de tablas

1. Descripción del proyecto

Este capítulo sirve de guía para situar al lector en el marco teórico-práctico en el que este trabajo resulta interesante y se espera que sea un referente en los próximos años. De este modo, primero se realiza una introducción a los conceptos de optimización global y de cribado virtual. Seguidamente se expone la motivación que ha llevado a realizar este trabajo, la especificación del mismo y los objetivos que se pretenden alcanzar. Por último se presenta la estructura del resto de la memoria.

1.1 Introducción

Uno de los principales objetivos de la investigación es contribuir a la solución de los problemas sociales, económicos y tecnológicos de la sociedad. Para abordar de forma eficaz muchos de estos problemas se requiere la aportación multidisciplinar procedente de varios campos científicos. En este sentido, la Química y la Informática son, cada vez más, unos ingredientes esenciales en la investigación sobre nuevos retos científicos y tecnológicos, que a su vez enriquecen dicha Ciencia. En particular la modelización y la optimización se han convertido en poderosas herramientas para simular y mejorar utilizando multitud de procesos ejecutados sobre ordenadores. Esto permite, entre otras cosas, optimizar procesos y evitar un intensivo trabajo de experimentación en laboratorio, lo que se suele traducir en un considerable ahorro de tiempo y de dinero.

Este trabajo aborda algunos de los problemas a los que se enfrentan actualmente los expertos en el proceso de creación o descubrimiento de nuevos fármacos ya sea optimizando su proceso, validando nuevos modelos o simulando las posibles reacciones que puedan provocar.

El primer paso para poder estudiar y resolver este tipo de problemas reales consiste en definir e implementar un modelo matemático del problema. Tras la validación del modelo se pueden

definir los diferentes problemas de optimización que se quieren resolver. De este modo es posible aplicar algoritmos de optimización global que encuentren las soluciones óptimas que permitan disminuir el tiempo de resolución de tales problemas.

1.1.1 Optimización Global

La Optimización Global (GO) es el campo que incluye teoría, métodos y aplicaciones de las técnicas cuyo objetivo es encontrar el óptimo global en problemas de optimización donde existen (muchos) óptimos locales (son óptimos en un entorno) que no son óptimos globales (los mejores de todas las posibles alternativas). Los algoritmos de resolución disponibles son de dos tipos diferentes:

1. Algoritmos determinísticos, como los de Ramificación y Acotación (Branch and Bound), que requieren cierto conocimiento de la estructura matemática del problema a resolver y que presentan una garantía de que se ha encontrado el óptimo.
2. Algoritmos heurísticos, que están basados en generaciones aleatorias de puntos de prueba y que, a priori, no requieren conocimiento alguno de la estructura matemática del problema a resolver, si bien no se puede garantizar su convergencia hacia el óptimo.

En la literatura se han propuesto numerosos tipos de algoritmos de optimización global, aunque este trabajo, dada la complejidad de los problemas planteados, se centrará fundamentalmente en el diseño e implementación de técnicas heurísticas. En particular, en técnicas metaheurísticas híbridas para optimización mono-objetivo. En el contexto mono-objetivo la calidad de una determinada solución viene dada directamente por el valor de aptitud en la función objetivo. Teniendo en consideración la experiencia previa en optimización del grupo investigador al que pertenece el alumno que realiza este trabajo [78, 79, 80, 81, 82, 90, 91, 92, 93], las técnicas metaheurísticas que se diseñarán e implementarán serán de tipo evolutivo. La computación evolutiva está basada en las ideas de la selección natural, vista como un proceso de optimización en el que paulatinamente los individuos van mejorando para adaptarse a su medio. Para la computación evolutiva mono-objetivo, un individuo es una solución potencial a un problema; el medio donde se desenvuelve lo componen la función objetivo y las restricciones, indicando cómo de apto es el individuo para sobrevivir.

En cuanto a la carga computacional asociada a los algoritmos de optimización aplicados a problemas de bioinformática, ésta es considerable, por lo que se plantea el uso de lenguajes que permitan obtener el mejor rendimiento (C, C++).

1.1.2 Cribado Virtual

En la investigación clínica es crucial, por un lado, determinar que un determinado fármaco es seguro y eficaz y, por otro, acelerar el descubrimiento de nuevos compuestos activos. En la actualidad, la industria farmacéutica requiere una media de 12 a 20 años (ver Figura 1.1) y unos costes de aproximadamente 850 millones de euros para el desarrollo y lanzamiento de un nuevo fármaco al mercado, lo cual supone un gran gasto de tiempo y dinero. Sin embargo, la bioinformática puede acelerar drásticamente este proceso, proporcionando la predicción de la bioactividad y toxicidad de fármacos y su actividad en enfermedades nuevas, así como la

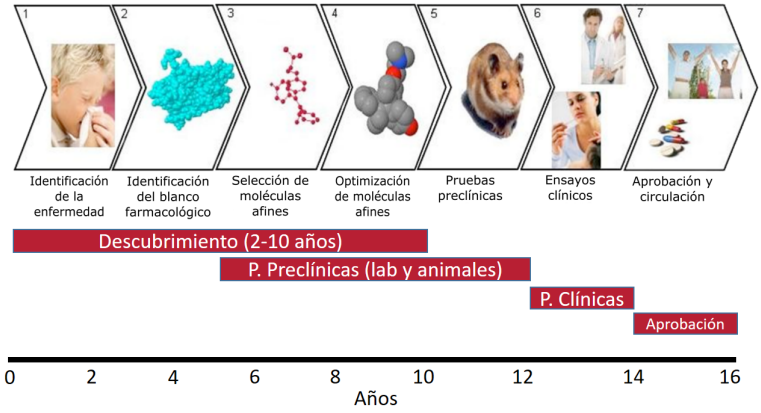


Figura 1.1: Etapas para el desarrollo de nuevos fármacos.

evolución de los compuestos activos descubiertos en ensayos clínicos.

Esto se puede lograr gracias al uso de herramientas de bioinformática y de métodos de cribado de quimiotecas (o base de datos de proteínas - PDB) virtuales (Virtual Screening), las cuales permiten probar todas las hipótesis posibles antes de realizar los ensayos clínicos. Las técnicas de cribado virtual resultan baratas, rápidas, y permiten tener en cuenta un gran número de compuestos in-silico del orden de billones, cifra impensable experimentalmente.

Existen dos metodologías para realizar un proceso de cribado virtual:

1. Si se dispone de la estructura tridimensional de la diana terapéutica, bien obtenida por métodos experimentales (cristalografía de rayos X o RMN) o bien a través de la construcción de modelos moleculares, se sigue la metodología de diseño de ligandos basada en su estructura. Se incluyen aquí las técnicas de docking (intento de encontrar el mejor acoplamiento entre dos moléculas: un receptor y un ligando).
2. En caso contrario, el cribado virtual se puede realizar mediante búsquedas basadas en ligandos, que consisten en el análisis y comparación de diversos descriptores moleculares y datos de afinidad con respecto a ligandos conocidos, sin tener en cuenta la estructura de dicho receptor. Aquí se incluyen las técnicas de búsqueda de similitud mediante descriptores 2D/3D, QSAR (Quantitative Structure-Activity) y técnicas de shape matching (comparación de la forma global o parcial entre moléculas).

1.2 Motivación

Este trabajo persigue la búsqueda de soluciones óptimas a problemas científico-tecnológicos relacionados con el descubrimiento de nuevos fármacos computacionalmente costosos mediante el desarrollo de algoritmos secuenciales. A esto se le añade el conocimiento profundo que los directores de este trabajo han adquirido durante los últimos años en los campos de la optimización global, la computación de altas prestaciones y la modelización de problemas y que es de vital importancia en este tipo de problemas puesto que éstos cada vez son más complejos y requieren de una computación mayor. Este trabajo fin de máster presenta un claro carácter multidisciplinar,

reflejo de las características propias del alumno y sus supervisores, de la investigación llevada a cabo y de las colaboraciones mantenidas, especialmente, con los miembros del grupo de BIO-HPC de la Universidad Católica San Antonio de Murcia, expertos en procesos bioinformáticos.

Los problemas que se pretenden solucionar en este trabajo son problemas de optimización global que requieren el uso de sofisticadas técnicas de Optimización Global, como las que se van a desarrollar en el presente proyecto. Dada la complejidad computacional de los algoritmos de optimización y de los problemas de optimización, se requiere del uso de computación de altas prestaciones para resolución de tales problemas, por lo que los algoritmos de optimización se diseñarán con paralelismo en mente. El grupo de investigación "Supercomputación-Algoritmos" tiene una amplia experiencia tanto en el diseño de algoritmos de optimización global y su aplicación a problemas concretos como a la aceleración de códigos mediante el uso de la computación de altas prestaciones.

La motivación en la que se sustentan los problemas a estudiar relacionados con el descubrimiento de nuevos fármacos o compuestos bioactivos es que tienen una gran relevancia en el campo de la biomedicina. Es de especial interés encontrar soluciones farmacológicas validadas y contrastadas antes de su distribución en el menor tiempo posible desde el momento de aparición del causante maligno.

1.3 Especificación del proyecto. Objetivos

En este trabajo nos centraremos en el cribado virtual de ligandos, en el que dado un conjunto de ligandos de referencia, habrá que identificar las conformaciones candidatas más similares o afines, de entre los millones existentes en una quimioteca virtual. Para ello, los algoritmos de cribado virtual, asocian una puntuación (calculada mediante una función de scoring) a cada una de las conformaciones incluidas en esa base de datos.

Las funciones a optimizar en nuestro trabajo consistirán en funciones de similitud 2D/3D que proporcionarán valores de scoring según diferentes descriptores moleculares y también funciones que comparan las formas de los compuestos y que están definidas mediante técnicas de shape matching.

Finalmente es importante destacar que para poder procesar grandes librerías con millones de conformaciones, los algoritmos de cribado deben ser lo suficientemente rápidos como para poder identificar las soluciones en un espacio de tiempo razonable. Para acelerar el proceso, se utilizan aproximaciones de las funciones objetivo, que en algunos casos pueden conducir a resultados erróneos.

De forma más específica, este Trabajo Fin de Máster se centrará en el diseño de una herramienta software, en la que dada una molécula o compuesto de referencia, sea capaz de encontrar ligandos con alto grado de similitud con la de referencia y con mayor actividad biológica aunque tengan diferente estructura química. Esta búsqueda se realizará en quimiotecas virtuales con millones de compuestos químicos y se implementará de forma eficiente mediante algoritmos metaheurísticos de optimización mono-objetivo. En cada una de las comparaciones entre la referencia y un compuesto de la base de datos, se han de considerar las propiedades espaciales

3D, lo que implica que los algoritmos de optimización tienen que explorar todas las posibilidades de rotación y traslación de la referencia para evaluar la función objetivo. Dependiendo de la información que se quiera tener al comparar los dos compuestos se puede elegir entre una gran cantidad de funciones objetivo tales como el potencial aromático, el potencial electróstatico, el potencial de desolvatación, etc. Para este trabajo se ha seleccionado la similitud de forma.

El desarrollo de este trabajo supone dar un paso hacia delante en los retos de la sociedad establecidos por el MINECO [72] y el H2020 [40]. Más concretamente en el reto 1 en Salud, cambio demográfico y Bienestar, principalmente en los puntos IV: Desarrollo de nuevas moléculas como armas terapéuticas y VI: El uso y difusión de las Tecnologías de la Información. Según se indica en [72] se favorecerán las actuaciones destinadas a maximizar:

1. El potencial de tecnologías -como la genómica, proteómica, biotecnología, nanotecnología, bioinformática y TIC, entre otras- y
2. Los retornos derivados del uso de las infraestructuras científicas y técnicas existentes.

Este trabajo también se enmarca dentro del reto de Economía y Sociedad Digital, principalmente en el punto IV de Aplicaciones y Soluciones TICs apartado (vi) salud y bienestar.

1.4 Estructura del documento

En este capítulo se ha hecho una introducción sobre el trabajo, la motivación que ha llevado a realizarlo y los objetivos que se pretenden alcanzar una vez finalizado.

En el Capítulo 2 se explican las distintas fases en las que se ha dividido el TFM, así como el tiempo aproximado dedicado en cada una de ellas. Además, se utiliza COCOMO como métrica para realizar una comparación entre el tiempo real dedicado y el teóricamente necesario.

Por su parte, en el Capítulo 3 se nombran todas las herramientas, tecnologías y aplicaciones utilizadas. El objetivo de este capítulo no es otro que hacer una breve descripción de cada una de ellas de tal forma que se conozca las bases materiales con las que se ha elaborado este trabajo.

El resto de capítulos se han dedicado a las distintas partes del algoritmo que se ha desarrollado. En ese sentido, el Capítulo 4 sirve de introducción al concepto de algoritmo evolutivo y toda la familia de éstos relacionada con su naturaleza intrínseca. Con la lectura de este capítulo se tendrá una visión general de los conceptos que permitirá afrontar con más facilidad el resto de la memoria.

En el Capítulo 5 se encuentran descritos los mecanismos específicos desarrollados en este trabajo del algoritmo. En ese sentido, este capítulo se divide en tres partes. En una primera se realiza un estudio de la función objetivo y su evolución. Posteriormente se detallan los conceptos evolutivos en los que se fundamenta el algoritmo, para dar lugar a la tercera y última parte en la que se han descrito las consideraciones específicas aplicadas al problema planteado.

Los resultados obtenidos del algoritmo así como las comparaciones realizadas con otros se muestran en el Capítulo 6. Las comparaciones se realizarán en base a dos parámetros distintos:

valor de la función objetivo y calidad de la clasificación de compuestos.

Por último, el Capítulo 7 se dedica a exponer las conclusiones obtenidas durante el desarrollo de este trabajo y deja planteado el trabajo futuro.

En las últimas páginas, se pueden encontrar las referencias bibliográficas citadas a lo largo de los capítulos de la memoria y donde el lector encontrará información adicional y más detallada de los conceptos aquí aplicados.

2. Fases del Trabajo Fin de Máster

En este capítulo se describen los distintas tareas que se han realizado para llevar a cabo el trabajo. Además, se muestra la dedicación empleada en cada una durante los dos años que se han empleado para completar este trabajo. Por último se realiza un análisis basado en COCOMO con el objetivo de realizar una comparación entre las horas dedicadas al trabajo y las horas que teóricamente se tendrían que haber dedicado.

2.1 Tareas

Como se ha podido conocer en el Capítulo 1, este trabajo tiene el objetivo de proporcionar una nueva herramienta que sea de utilidad en el proceso de desarrollo de nuevos fármacos reduciendo el tiempo de este proceso y/o mejorando la calidad de los resultados de los experimentos existentes.

Abordar todo el proyecto en su completitud como unidad puede ser una tarea agotadora e infranqueable, por lo que se decidió dividirlo en pequeñas fases. Este enfoque permite un conocimiento y profundización mayor del problema y su solución pero representado de una manera más simple e inteligible. A continuación se describirán brevemente las distintas etapas cuyo contenido se desarrollará a lo largo de los siguientes capítulos.

2.1.1 Estudio y comprensión de los algoritmos evolutivos. Técnicas

La tarea inicial de este proyecto consistió en estudiar y comprender los fundamentos de los algoritmos de optimización global y en particular la subfamilia de los evolutivos. Esta tarea se llevó a cabo en los primeros meses de ejecución de este trabajo. Se persiguieron dos objetivos,

por un lado, dar una visión global al autor sobre los diferentes tipos de algoritmos con sus ventajas y desventajas, aplicaciones, resultados esperados, y un segundo objetivo más específico, donde la asimilación de los métodos y mecanismos propios de los algoritmos evolutivos para aplicarlo al problema presente era un requisito fundamental.

2.1.2 Estudio y comprensión del procedimiento de obtención de nuevos fármacos y técnicas de cribado virtual

El siguiente estudio necesario fue el correspondiente al campo en el que se aplicaron los conceptos algorítmicos adquiridos en la tarea anterior. En ese sentido, se procedió a la lectura de las últimas publicaciones relacionadas con el cribado virtual. En este momento, tras un análisis comparativo detallado de las distintas funciones objetivo susceptibles de ser optimizadas, se seleccionó la función objetivo que se utilizaría en el resto del trabajo, en particular la de similitud de forma. De este modo, esta etapa no consistió solo en una recopilación de conocimiento sino que se hizo una introducción a los modelos matemáticos que tendrían que usarse en el futuro.

2.1.3 Diseño e implementación de la función objetivo

El objetivo de esta etapa consiste en implementar de forma eficiente y correcta la función objetivo seleccionada para este trabajo. Los modelos se conocen y están publicados en revistas de impacto y existe software de caja negra que ya los implementan. En ese sentido, para obtener un algoritmo de mejor calidad que los actuales, es necesario compararlos con las mismas herramientas y ahí es donde radica la importancia de tener implementada una función objetivo correcta que proporcione los mismos valores que los indicados en la literatura.

2.1.4 Diseño e implementación del algoritmo evolutivo e incorporación de la función objetivo

Tras desarrollar la función objetivo, el siguiente paso consistió en incorporarlo al algoritmo con el que se quiere dar solución al problema visto en el primer capítulo. Con este fin, se diseñan e implementan los métodos propios de un algoritmo evolutivos (generación, selección, evolución, etc.) y los específicos correspondientes al problema del cribado virtual (gestión de ficheros, alineaciones, filtros moleculares, etc.) formando en su conjunto un único paquete software.

2.1.5 Experimentación, incremento de características, nuevas configuraciones

En esta etapa se realizan las mismas actividades que en las dos anteriores. Sin embargo, se ha considerado oportuno extraerla para resaltar la complejidad del problema. Y es que este Trabajo Fin de Máster tiene mucho de investigación y los objetivos que se quieren alcanzar son ambiciosos. Esto implica una gran dificultad para obtener los mejores resultados que se traduce en probar muchas configuraciones del algoritmo, añadir o eliminar características y sobre todo, obtener gran cantidad de datos experimentales para poder analizarlos y actuar en consecuencia.

2.1.6 Optimización y mejora del rendimiento del algoritmo

Dado el carácter innovador del trabajo, el autor ha tenido que adquirir una gran cantidad de conocimiento y de habilidades tanto de programación como de uso de herramientas matemáticas. Todo este aprendizaje ha contribuido a que el autor, durante bastantes intervalos de tiempo, pudiera ir mejorando considerablemente las diferentes partes del algoritmo mediante técnicas nuevas aprendidas. De esta forma se redujo considerablemente el tiempo de las ejecuciones permitiendo realizar aun más experimentos y por tanto obtener una retroalimentación mayor del enfoque del problema.

2.1.7 Difusión científica

Complementando este trabajo con la parte investigadora, se presentaron distintos desarrollos en diferentes eventos científicos. Esta etapa es de especial importancia para el alumno por su preferencia continuista en la investigación durante los próximos años. Además, estas aportaciones tuvieron también el objetivo de incentivar las colaboraciones de investigación y desarrollo con personal externo al grupo de trabajo.

2.1.8 Redacción y defensa de la memoria

Este trabajo se cierra con la documentación. Su última posición no implica que no haya estado presente desde el momento que comenzó la primera etapa. Sin embargo, no es posible incluir todo el conocimiento que se ha adquirido desde entonces en este memoria. El objetivo de esta etapa es recopilar toda esa información, seleccionar la más importante y presentarla correctamente en esta memoria.

2.2 Diagrama de Gantt

Es una herramienta muy utilizada en la gestión de proyectos. Permite modelar la planificación de las tareas que deben completarse para realizar satisfactoriamente un proyecto. Fue inventada por Henry L. Gantt en 1917. En él se puede ver de una manera general el tiempo invertido en cada una de las tareas así como el orden en el que se han realizado. El diagrama de Gantt de este trabajo se puede ver en la Figura 2.1.

2.3 COCOMO

El Modelo Constructivo de Costos (COCOMO en adelante) es un modelo matemático de base empírica utilizado para estimación de costes del software. El hecho de ser empírico implica que con la experiencia y el uso se va mejorando su precisión en los resultados. Y es que éstos son estimaciones basadas en la experiencia obtenida en proyectos ya terminados.

Id	Nombre de la tarea	T4 2015			T1 2016			T2 2016			T3 2016			T4 2016			T1 2017			T2 2017		
		Oct.	Nov.	Dic.	Ene.	Feb.	Mar.	Abr.	May.	Jun.	Jul.	Ago.	Sep.	Oct.	Nov.	Dic.	Ene.	Feb.	Mar.	Abr.	May.	Jun.
2.1.1	Estudio y comprensión de los algoritmos evolutivos. Técnicas	■	■	■	■																	
2.1.2	Estudio y comprensión del procedimiento de obtención de nuevos fármacos y técnicas de cribado virtual			■	■	■																
2.1.3	Diseño e implementación de la función objetivo				■	■																
2.1.4	Diseño e implementación del algoritmo evolutivo e incorporación de la función objetivo					■	■															
2.1.5	Experimentación, incremento de características, nuevas configuraciones							■	■	■	■		■	■	■	■	■	■	■	■	■	■
2.1.6	Optimización y mejora del rendimiento del algoritmo														■	■		■				
2.1.7	Difusión científica			■	■			■			■		■					■			■	
2.1.8	Redacción y defensa de la memoria															■	■				■	■

Figura 2.1: Diagrama de Gantt del TFM.

OPTIPHARM	L. Implementadas	L. Reutilizadas	L. Autogeneradas
Config	39		2 735
Funciones	883		
I/O	350		
Modelo	495		
Núcleo	876	2 564	
Test	596	46	
SCRIPTS	L. Implementadas	L. Reutilizadas	L. Autogeneradas
Pymol	169		
Python	3 718		
R		100	
Total Desglosado	7 126	2 710	2 735
Total			12 571

Tabla 2.1: Líneas de código desglosadas por grupos.

Incluye tres submodelos, cada uno ofrece un nivel de detalle y aproximación cada vez mayor, a medida que avanza el proceso de desarrollo del software: básico, intermedio y detallado. Además existen dos versiones, COCOMO 87 que fue el primer modelo que se creó y COCOMO 2 que incorpora una serie de submodelos para ajustar mejor las estimaciones. Para la estimación de este trabajo se aplica COCOMO 2.

El objetivo de calcular esta estimación una vez ha finalizado el proyecto no es otra que conocer si el tiempo dedicado y los recursos humanos involucrados reales coinciden con la estimación. En ese sentido, dentro de los tres modelos que propone COCOMO 2, se va a considerar Post-Arquitectura ya que se conocen todos los datos al haber finalizado el trabajo. Las fórmulas y tablas que se aplican para obtener el esfuerzo en persona/mes, el tiempo de desarrollo y el número de personas necesarias se pueden ver en [84].

El primer dato necesario que hay que obtener para comenzar la estimación son las *KLOC* (miles de líneas de código). En la Tabla 2.1 se detallan todos los archivos que se han utilizado así como las líneas de código que se han implementado, reutilizado y autogenerado.

A continuación se va a calcular el esfuerzo persona/mes. La fórmula es:

$$PM = A \cdot [Size']^B \cdot \prod_{i=1}^{17} EM_i + PM_{AT}$$

A es una constante cuyo valor es 2.5.

$Size'$ representa el número de líneas de código ponderado según las líneas que se hayan desechado, desarrollado, reutilizado y autogenerado. Por tanto,

$$Size' = Size \cdot (1 + \frac{BRAK}{100})$$

donde *BRAK* es el porcentaje de código que se ha desechado. En este caso su valor es 0.

Finalmente, el valor de $Size'$ se obtiene mediante la siguiente expresión:

$$Size' = Size \cdot (1 + \frac{0}{100}) = Size = KNSLOC + KASLOC \cdot \frac{100 - AT}{100} * AAM$$

donde $KNSLOC$ es el número de líneas de código desarrollado (en miles), es decir 7.126 $KNSLOC$.

$KASLOC$ es el número de líneas de código reutilizado (en miles), en este caso 2.710.

AT es el porcentaje de código automáticamente generado entre el total, en este caso $\frac{2\,735}{12\,571} = 0.22$.

y AAM es un multiplicador de ajuste de adaptación que puede tomar su valor de dos posibles fórmulas dependiendo del AAF (Factor de Ajuste de Adaptación):

$$AAM = \begin{cases} \frac{AA + AAF \cdot (1 + 0.02 \cdot SU \cdot UNFM)}{100}, AAF \leq 0.05 \\ \\ \frac{AA + AAF + SU \cdot UNFM}{100}, AAF > 0.05 \end{cases}$$

donde $AAF = 0.4 \cdot DM + 0.3 \cdot CM + 0.3 \cdot IM = 0.4 \cdot 0 + 0.3 \cdot \frac{2\,710}{12\,571} \cdot 0.3 \cdot 0 = 0.3 \cdot 0.22 = 0.066$. Por tanto se utiliza la fórmula inferior de AAM:

$$AAM = \frac{AA + AAF + SU \cdot UNFM}{100}$$

donde AA representa el nivel de evaluación y asimilación. El valor que toma es 2. SU representa la comprensión del software. En ese sentido se considera un valor de 20 (Alto). Por último $UNFM$ mide el nivel de desconocimiento, cuyo valor es 1.0 (Completamente desconocido).

De esta forma, $AAM = \frac{2 + 0.066 + 20 \cdot 1.0}{100} = 0.22$.

Una vez se tienen todos los valores para calcular $Size'$, sustituimos:

$$Size' = Size = KNSLOC + KASLOC \cdot \frac{100 - AT}{100} \cdot AAM = 7.126 + 2.710 \cdot \frac{100 - 22}{100} \cdot 0.22 = 7.59$$

A continuación se va a calcular B:

$$B = 1.01 + 0.01 \cdot \sum W_i$$

W_i representa cada uno de los factores de escala:

- Procedencia ($PREC$).
- Flexibilidad en el desarrollo ($FLEX$).
- Resolución de riesgo ($RESL$).
- Cohesión de equipo ($TEAM$).

- Madurez del proceso (*PMAT*).

En ese sentido $B = 1.01 + 0.01 \cdot (PREC + FLEX + RESL + TEAM + PMAT) = 1.01 + 0.01 \cdot (6.20(VL) + 3.04(L) + 4.24(N) + 3.29(N) + 6.24(L) = 1.01 + 0.01 \cdot (23.01) = 1.24$.

$\prod_{i=1}^{17} EM_i$ representan los multiplicadores de esfuerzo, dependiendo de la madurez del proyecto puede no disponer de información de ninguno, conocer 7 o 17. En este caso dado que se conoce todo el proyecto, se aplican los 17 factores. Estos son:

- *RELY*: nivel de confiabilidad para realizar la función esperada.
- *DATA*: medida del volumen de datos.
- *CPLX*: complejidad del producto.
- *RUSE*: grado de reusabilidad requerida para otras aplicaciones.
- *DOCU*: documentación requerida de acuerdo al ciclo de vida.
- *TIME*: restricciones del tiempo de ejecución.
- *STOR*: restricciones del almacenamiento principal.
- *PVOL*: volatilidad de la plataforma *HW-SW* de base.
- *ACAP*: capacidad de los analistas para trabajar en equipo.
- *PCAP*: capacidad de los programadores para trabajar en equipo.
- *AEXP*: experiencia en las aplicaciones.
- *PEXP*: experiencia en la plataforma.
- *LTEX*: experiencia en lenguajes y herramientas.
- *PCON*: continuidad del personal.
- *TOOL*: uso de herramientas de software.
- *SITE*: desarrollo en sitios múltiples.
- *SCED*: restricciones impuestas al plan de trabajo.

El valor $\prod_{i=1}^{17} EM_i = 0.82(VH) \cdot 0.9(L) \cdot 1(N) \cdot 1.07(H) \cdot 1(N) \cdot 1.29(VH) \cdot 1(N) \cdot 1(N) \cdot 0.85(H) \cdot 0.88(H) \cdot 1.22(VL) \cdot 1.19(VL) \cdot 1.09(L) \cdot 0.81(VH) \cdot 0.78(VH) \cdot 1(N) \cdot 1(N) = 0.76$

Para obtener el valor *PM* solo falta calcular el esfuerzo persona/mes que implica el código autogenerado:

$$PM_{AT} = \frac{ASLOC \cdot \frac{AT}{100}}{ATPROD} = \frac{2\,735 \cdot \frac{22}{100}}{2\,400} = 0.25$$

Por tanto, una vez se tienen todos los valores se puede calcular el esfuerzo:

$$PM = A * [Size']^B \cdot \prod_{i=1}^{17} EM_i + PM_{AT} = 2.5 \cdot [7.59]^{1.24} \cdot 0.76 + 0.25 = 23.71$$

A continuación se calcula el tiempo de desarrollo. La fórmula es:

$TEDV = [3.67 \cdot MM^{0.28+0.2\cdot(B-1.01)}] \cdot \frac{SCED\%}{100} = 3.67 \cdot 15.05^{0.28+0.2\cdot(1.24-1.01)} \cdot \frac{100}{100} = 3.67 \cdot 15.05^{0.326} = 8.88$ meses.

Los recursos humanos necesarios se calculan con los dos últimos valores obtenidos:

$$RRHH = \frac{PM}{TEDV} = \frac{23.71}{8.88} = 2.67 \text{ personas.}$$

Los resultados teóricos que se han obtenido en la estimación indican que para realizar este trabajo son necesarias 2.67 personas trabajando durante 8.88 meses.

Este trabajo tuvo su génesis el día 1 de octubre de 2015. Considerando el mes de finalización junio de 2017 y descartando agosto de 2016, se ha trabajado en este proyecto 20 meses. Los primeros 9 meses se compaginaron con el primer curso del Máster en Ingeniería Informática, lo que implica que al día se trabajaba como máximo media jornada en el trabajo y en dos o tres de estos meses no se avanzó por dedicación plena al Máster. De este modo, finalmente restan 11 meses a tiempo completo y 6-7 meses a tiempo parcial.

Dado que he recibido ayuda de 3 supervisores, asignémosle holgadamente 1 persona (teórica) a su trabajo. En consecuencia queda como trabajo del alumno 1.67 personas y 8.88 meses. Suponiéndose un trabajo secuencial se puede reducir el número de personas a 1 produciéndose un incremento de los meses a 14.82.

De una forma más o menos aproximada se podría concluir que la estimación se ajusta aproximadamente a la realidad. No obstante, hay que tener en cuenta que el software presentado en este trabajo tiene carácter científico y esto implica que una tarea puede llevar más tiempo del estimado.

3. Herramientas

En este capítulo se detallan las herramientas y lenguajes que se han utilizado para la realización de este trabajo. Se han dividido en cinco categorías: lenguajes, comunicación, desarrollo, documentación e infraestructura. Para cada herramienta y lenguaje se hace una breve descripción de la misma y se justifican los motivos de su uso. El orden de aparición de cada una de ellas es alfabético salvo la categoría de Lenguajes que se sitúa la primera. El motivo de ello es introducir al lector el lenguaje para que tenga un conocimiento previo de él cuando se nombre en el resto de categorías.

3.1 Lenguajes

En esta categoría se van a introducir los lenguajes de programación utilizados en este trabajo.

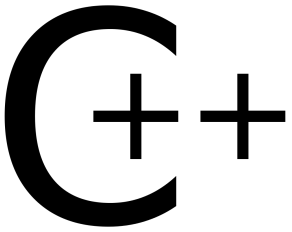
3.1.1 Bash

Bash (Bourne again shell) es un programa informático, cuya función consiste en interpretar órdenes. Está basado en el shell de Unix y es un lenguaje de programación de consola. Es el intérprete de comandos utilizado en la mayoría de las distribuciones de GNU con Linux. También se puede utilizar en Mac OS e incluso en Windows con Cygwin.

Justificación de uso

La ejecución de los programas en supercomputadores se realiza en su mayoría mediante instrucciones por consola de comandos (en su mayoría Linux). En ese sentido se ha automatizado el proceso de ejecución de miles de experimentos mediante scripts codificados en este lenguaje.

3.1.2 C++



C++ es un lenguaje de programación que fue diseñado por Bjarne Stroustrup a mediados de los años 1980. El objetivo de este lenguaje era extender la funcionalidad de C incluyéndole el paradigma orientado a objetos. Posteriormente se le añadió también el paradigma de programación estructurada, de ahí que C++ sea un lenguaje de programación multiparadigma. Se han ido validando distintos estándares y actualmente la última versión oficial es C++14, donde 14 hace referencia al año 2014.

Justificación de uso

El software desarrollado en este trabajo se ha desarrollado en C++. El intención de utilizar este lenguaje residen en el mejor rendimiento que se obtiene en las ejecuciones si se compara con otros lenguajes de más alto nivel y con una curva de aprendizaje inicial menor.

3.1.3 Latex

Si bien el lenguaje de programación real es Tex, Latex está formado por un gran conjunto de macros de este lenguaje. Se utiliza para la composición de textos. Su principal virtud es la alta calidad tipográfica que se obtiene como resultado en los documentos siendo ésta comparable con la de cualquier editorial de primera línea. Su mayor popularidad se encuentra en áreas como las matemáticas y la física por su facilidad para escribir complejas fórmulas y expresiones. No obstante se pueden hacer figuras cuando se tiene un amplio conocimiento del lenguaje. Es muy utilizado para la composición de artículos académicos, tesis y libros técnicos. LaTeX es software libre bajo licencia LPPL.

Justificación de uso

Se ha utilizado para la elaboración de la memoria de este trabajo por la calidad que se obtiene y por la facilidad de gestionar la posición de figuras y tablas y que tan problemática es con otras herramientas. Otro uso que se le ha dado ha sido para mantener un diario de trabajo donde se almacenan todos los resultados manteniendo una buena presentación con poco esfuerzo. Las contribuciones científicas derivadas de este trabajo también han sido escritas en Latex.

3.1.4 Python

Python es un lenguaje de programación interpretado cuya filosofía es mantener una sintaxis lo más sencilla y legible posible. Permite la programación orientada a objetivo, la programación imperativa y la programación funcional por lo que se puede categorizar como lenguaje de programación multiparadigma. Es un lenguaje tipado y multiplataforma. En los sistemas operativo Unix y Linux se encuentra instalado por defecto. Es administrado por la Python Software Foundation.

Justificación de uso

Dada la facilidad con la que se pueden implementar tareas complicadas en este lenguaje, se ha utilizado para crear scripts que procesen los millones de resultados que generan los experimentos. Para ello se han usado desde librerías para la lectura y escritura de archivos en texto plano o csv hasta la representación de datos en dos y tres dimensiones para su posterior análisis.

3.1.5 R

R es un lenguaje de programación utilizado principalmente para el análisis estadístico. Tiene su origen en el lenguaje S. Se trata de uno de los lenguajes más utilizados en investigación por la comunidad estadística siendo especialmente útil en todas aquellas áreas donde es necesario analizar los datos de miles o millones de registros como puede ser el caso de la minería de datos. Además, se puede ampliar su utilidad añadiéndole bibliotecas o paquetes desarrollados por terceros. R es parte del sistema GNU y se distribuye bajo la licencia GNU GPL. Está disponible para la mayoría de sistemas operativos.

Justificación de uso

Se ha utilizado R para obtener los resultados de la curva ROC y el área bajo ella. También se ha usado para incluir resultados de forma dinámica en el archivo PDF que se mantiene con todo el trabajo realizado hasta el momento. Para conseguir esto último, se ha hecho uso del paquete Knitr. Este paquete permite generar automáticamente gráficas y análisis que se incluyen en el PDF de forma sencilla y sin perder el formato ni la distribución en el documento modificando solamente el archivo origen de la fuente de datos.

3.2 Comunicación

En esta sección se exponen las herramientas utilizadas para la comunicación entre el alumno, los directores de este trabajo y el resto de personas que ha contribuido en el mismo.

3.2.1 Email

Thunderbird[1] es un cliente de correo multiplataforma y de código libre desarrollado por la Fundación Mozilla[2]. Permite gestionar varias cuentas de correo y la gran cantidad de extensiones disponibles le proporcionan una gran personalización en cuanto a diseño como a funcionalidad.

Justificación de uso

En trabajos de investigación como el que aquí se aborda se necesita de una constante comunicación ya sea para definir las pautas a seguir o para comentar los resultados obtenidos hasta el momento y proceder en base a ellos.

Debido a la cantidad de trabajo que cada componente del equipo tiene, coincidir en una fecha y hora determinada todos físicamente es complicado, sobretodo con Horacio (residente en la Región de Murcia). Es por ello que se envían los resultados por correo generando un hilo de discusión y se comparten en él las ideas de cada uno. Además queda constancia de todo lo decidido ahorrando tiempo en documentación y creando una línea del tiempo con todos los temas hablados.

3.2.2 Informes

El formato de documento portátil (PDF) permite intercambiar información de forma fácil independientemente de la plataforma utilizada. Inventado por Adobe[3], se ha convertido en un estándar oficial y abierto reconocido por la Organización Internacional para la Estandarización (ISO). Además de texto, en un PDF se pueden incluir imágenes, enlaces, vídeos, etc. También se pueden firmar electrónicamente para validar documentos. Actualmente hay multitud de programas que permiten trabajar con este tipo de documentos.

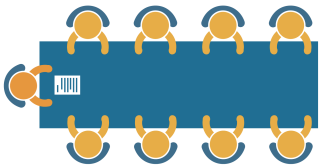
Justificación de uso

Su uso tiene dos finalidades. Por un lado, no toda la información se puede trasmitir mediante texto o voz. En ocasiones es necesario mostrar imágenes, tablas con resultados, evolución de las ejecuciones de los experimentos y es el documento PDF el que permite unir todos estos elementos en un mismo archivo. El segundo objetivo de su uso consiste en usar este

[1] https://www.mozilla.org/es-ES/thunderbird/
[2] https://www.mozilla.org/es-ES/about/
[3] http://www.adobe.com/es/

formato para mantener un historial de todas las decisiones que se han tomado, los resultados que se han obtenido y guardar todo ello en forma de diario. Además, estos informes se pueden enviar a los distintos miembros del grupo de forma que puedan seguir los avances que se están realizando sin tener que estar presencialmente en el momento en el que se obtengan.

3.2.3 Reunión



Acción donde dos o más personas quedan para discutir, debatir y/o presentar uno o varios temas. También puede ser usado para definir las futuras directrices de un trabajo.

Justificación de uso

Durante el proyecto han sido utilizadas para diversas finalidades: explicar contenido, resolver problemas, fijar nuevos objetivos, mostrar resultados, etc. Sobre todo entre los miembros de la Universidad de Almería, cuando se tiene que tratar un tema importante y de una forma rápida, en el sentido de que esa decisión afecta a los futuros avances, intercambiando opiniones de una forma fluida.

3.2.4 Skype

Es un software que permite comunicaciones de texto, voz y vídeo sobre Internet (VoIP) de forma gratuita. Es propiedad de Microsoft[4]. El código y protocolo de Skype permanecen cerrados y son privativos de la aplicación, pero los usuarios interesados pueden descargar gratuitamente la aplicación ejecutable del sitio web oficial. Es multiplataforma.

Justificación de uso

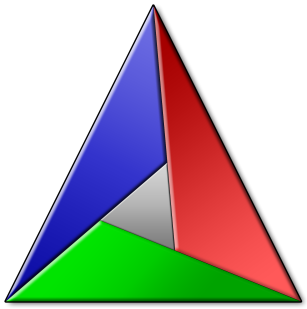
Se ha utilizado para mantener reuniones con los investigadores de la UCAM. Especialmente cuando el contacto visual era necesario y no se podía establecer una reunión física por problemas con los horarios.

3.3 Desarrollo

Las siguientes herramientas se utilizaron para el desarrollo y gestión del software en sus diferentes etapas. Algunas de ellas son específicas del ámbito de este trabajo fin de máster.

[4] `https://www.microsoft.com/es-es`

3.3.1 CMAKE



CMake es una familia de herramientas diseñada para construir, probar y empaquetar software. Para ello hace uso de ficheros de configuración sencillos e independientes de la plataforma. Estos ficheros tienen el nombre de CMakeLists.txt y mínimo debe de existir uno en el directorio raíz. Dependiendo de la complejidad del software, se pueden encontrar más archivos de configuración distribuidos en cada una de los subdirectorios del programa. CMake tiene definidos comandos propios pero también se le pueden añadir comandos personalizados por el usuario. Los archivos que genera reciben el nombre de Makefile. Existen Makefiles para Unix, Borland make, Watcom make, MinGW, MSYS y Microsoft NMake. También es posible generar ficheros de proyecto para Code::Blocks, Eclipse CDT y Microsoft Visual Studio.

Justificación de uso

Algunos de los paquetes externos utilizados hacen uso de CMake para gestionar las dependencias. Antes de compilarlo, era necesario ejecutar este programa. Por otro lado, el software que se ha desarrollado, con la intención de que se pueda ejecutar en distintas plataformas, se ha configurado con CMake. De este modo, cosas tan simples como indicar la dirección del compilador en cada uno de los equipos donde se desarrollaban las pruebas se ha automatizado.

3.3.2 Eclipse IDE

Eclipse[5] se un IDE que agrupa una amplio conjunto de herramientas. Es un software libre y está disponible para todas las plataformas ya que se ejecuta sobre una máquina virtual Java[6]. Inicialmente fue para este último lenguaje la iniciativa de crear el IDE pero tras los años se ha ido añadiendo soporte a otros lenguajes de programación como C, C++, PHP, Python, etc. Además, se le pueden añadir una gran cantidad de extensiones para adaptarlo a las necesidades finales del usuario. Estas extensiones son de lo más diversas: desde cambiarle la apariencia a la aplicación hasta cambiar el comportamiento de un funcionalidad muy específica. Eclipse es desarrollado por la Fundación Eclipse[7].

[5] https://eclipse.org/ide/
[6] https://www.java.com/es/
[7] https://eclipse.org/org/

Justificación de uso

La mayoría del software con el que se ha trabajado se encuentra escrito en C y C++ por lo que se ha usado una versión especifica del IDE para dichos lenguajes: Eclipse IDE for C/C++ Developers. Además se integra perfectamente con el control de versiones Git por lo que se puede volver a una versión anterior del código en caso de que se haya producido algún error.

3.3.3 EduPymol

PyMOL[8] es un visor molecular de código abierto. Permite generar imágenes 3D de moléculas pequeñas y de macromoléculas biológicas, como las proteínas. Permite extensiones con Python, lo cual es de gran utilidad para realizar análisis complejos de estructuras moleculares utilizando bibliotecas como NumPy o Pylab. Actualmente es comercializado por Delano Scientific LLC. Posee una versión educativa, EduPymol, que es gratuita pero tiene algunas características restringidas[9]. Permite leer y escribir moléculas en multitud de formatos, destacando el formato pse. Este formato permite salvar el estado actual del programa en el momento de guardarlo.

Justificación de uso

Uno de las principales tareas en este trabajo consiste en analizar y modificar la posición de las moléculas. Los resultados que obtiene las herramientas de optimización (y que se analizan) son devueltos en formato de texto plano. Sin embargo, a la vista del iniciado en este área esta información puede resultar insuficiente. Es por ello que se utiliza EduPymol para representar gráficamente las moléculas y la relación que guardan unas con otras. Además, no solo ayuda a representar a una molécula o conjunto de ellas sino que se puede rotar, ampliar y reducir la vista, así como hacer modificaciones en la propia moléculas permitiendo guardarlo todo en una sesión pse.

3.3.4 Gcc

El GNU Compiler Collection[10] (colección de compiladores GNU) es un conjunto de compiladores creados por el proyecto GNU. Es el compilador estándar en los sistemas operativos que tienen como origen UNIX, es decir, Linux y Mac OS X. Para su correcto funcionamiento, GCC precisa tener instalado el paquete binutils para realizar tareas como identificar archivos objeto u obtener su tamaño para copiarlos, traducirlos o crear listas, enlazarlos, o quitarles símbolos innecesarios. Originalmente solo compilaba C pero actualmente ha extendido su funcionalidad a C++, Fortran, Ada entre otros. Es software libre y lo distribuye la

[8] https://www.pymol.org/
[9] https://pymol.org/#compare3,4,5,6
[10] https://gcc.gnu.org/

Free Software Foundation (FSF)[11] bajo la licencia general pública GPL.

Justificación de uso

La mayoría del software que se ha desarrollado en este trabajo se encuentra implementado en C y C++, igual que otras librerías externas que se han utilizado. Utilizar estos lenguajes de programación obliga al uso de esta herramienta para compilar las soluciones desarrolladas.

3.3.5 Git

Git[12] es un software de control de versiones. Consiste en mantener todas las versiones de un software de una forma ligera y sin redundancia ayudando de esta forma al mantenimiento del mismo. Además se puede recuperar una versión determinada del software en cualquier momento así como crear nuevas ramas de implementación o etiquetas en versiones concretas. Puede ser usado en cualquier sistema operativo y es gratuito. Progresivamente ha ido sustituyendo a los repositorios SVN[13] y algunas web como Github[14] lo han utilizado como el núcleo de su servicio. Existen muchos proyectos, (la mayoría de software libre) que se gestionan con Git.

Justificación de uso

Se ha utilizado para mantener todas las versiones del código que se ha desarrollado así como también documentos en formato PDF o archivos comprimidos. Y es que Git no solo permite mantener un control de versiones del código sino que también sirve como gestor de copias de seguridad de otros archivos que se han considerado convenientes incluirlos junto al código fuente.

3.3.6 Make

Make es una herramienta de gestión de dependencias utilizada para automatizar el proceso de compilación de un programa. En ese sentido, una vez escritos los comandos necesarios para obtener un ejecutable en un archivo llamado Makefile y realizada la primera compilación, Make se encarga en futuras compilaciones de comprobar cuales de los archivos fuentes han sido modificados recompilando unicamente esos. De esta forma, y sobretodo para proyectos grandes, se ahorra mucho tiempo en evitar repetir acciones que no son necesarias. Make es muy usada en los sistemas operativos tipo Unix/Linux. Dependiendo de la información escrita en el archivo Makefile, puede configurarse para realizar operaciones de compilación, de limpieza, mover archivos, etc.

[11] https://www.fsf.org/?set_language=es
[12] https://git-scm.com/
[13] https://subversion.apache.org/
[14] https://github.com/

Justificación de uso

Su uso en este proyecto viene dado por la utilización de la herramienta CMake. Tras la ejecución de ésta, se generan archivos Makefile que permiten obtener los ejecutables de los programas.

3.3.7 OpenBabel

OpenBabel es un programa que se utiliza para intercambiar el formato de los archivos de moléculas. También se puede usar para filtrar moléculas o manipularlas de una forma básica. Es multiplataforma y gratuito.

Justificación de uso

Se ha utilizado principalmente para convertir las moléculas en formato Mol2 a SD y viceversa. Esto viene motivado por el hecho de que el programa desarrollado en este trabajo usa principalmente el formato Mol2 mientras que el programa comercial con el que se han realizado las comparaciones utiliza el formato SD. En situaciones puntuales se ha utilizado para convertir otros tipos de formatos menos comunes a Mol2 para experimentos muy concretos.

3.3.8 Visual Studio Code

Visual Studio Code[15] es un editor de código fuente desarrollado por Microsoft para Windows, Linux y MacOs. Tiene muchas funcionalidades incorporadas de base como la depuración de código, control de versiones Git, resaltado y autocompletado de código, entre otras. Además, esta basado en extensiones por lo que se puede añadir cualquier funcionalidad adicional simplemente obteniendo (o desarrollando) la extensión. Es gratis y de código abierto, aunque la descarga oficial es bajo una licencia propietaria.

Justificación de uso

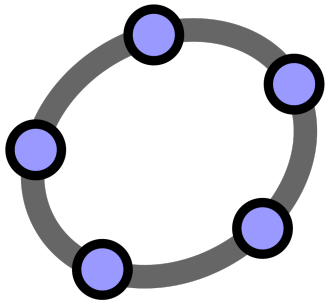
El principal motivo de utilizar Visual Studio Code y no otros editores ligeros como puede ser Notepad++[16] o Gedit[17] es su personalización. Además permite ampliar sus funcionalidades mediante extensiones que se pueden obtener desde su propio gestor interno. Algunas de las usadas están relacionadas con Python, C++, Tex y el formato PDF. Visualmente es muy agradable y se encuentra en un escalón intermedio entre los editores ya nombrados y Eclipse IDE (Subsección 3.3.2) aunque sus características lo sitúan más cerca de este último.

[15] `https://code.visualstudio.com/`
[16] `https://notepad-plus-plus.org/`
[17] `https://wiki.gnome.org/Apps/Gedit`

3.4 Documentación

En esta sección se detallarán los programas utilizados para elaborar esta memoria así como el resto de material que se ha preparado para dar difusión de los avances realizados en este trabajo de investigación.

3.4.1 GeoGebra



GeoGebra[18] es un programa matemático interactivo que incluye funciones propias de la geometría, el álgebra y el cálculo. Permite realizar representaciones gráficas en dos y tres dimensiones de puntos, lineas, planos y figuras geométricas. Además incluye una hoja de cálculo que permite la interacción con el parte gráfica permitiendo incluso la carga externa de datos. Es software libre y multiplataforma. Su sencillez le ha llevado a ser muy utilizado en colegios y universidades.

Justificación de uso

Una parte considerable del tiempo dedicado al trabajo ha consistido en el análisis de los volúmenes que ocupaban las moléculas y en las posiciones finales de los átomos después de aplicar cierta rotación sobre un eje determinado. En ese sentido, GeoGebra ha sido de gran ayuda permitiendo representar la posición de las moléculas para calcular sus volúmenes así como visualizar en dos y tres dimensiones las posiciones finales de los átomos en el espacio tras las modificaciones espaciales realizadas sobre ellos. De este modo se podía analizar el comportamiento del algoritmo.

3.4.2 GIMP

GIMP[19] (*GNU Image Manipulation Program*) es un programa de edición de imágenes digitales en forma de mapa de bits. Es un programa libre y gratuito. GIMP contiene un conjunto de herramientas que permiten modificar y editar imágenes así como añadir nuevos objetos, recortar o cambiar las propiedades físicas de las mismas. Lee y escribe en la mayoría de los formatos de ficheros gráficos teniendo incluso su propio formato de archivo, el XCF.

Justificación de uso

La representación gráfica de las moléculas con el programa EduPymol en ocasiones no permitía mostrar toda la información deseada. En ese sentido, se editaban las imágenes con

[18] `https://www.geogebra.org/`
[19] `https://www.gimp.org/`

GIMP ajustándolas a las necesidades requeridas. Un ejemplo de ello es la creación de películas donde se puede ver los valores de la función objetivo conforme se reproduce la animación.

3.4.3 Microsoft Office

Microsoft Office es un conjunto de programas de ofimática que permiten realizar tareas de lo más variadas: redactar documentos con Word, trabajar con datos con Excel, realizar presentación con PowerPoint, modelar diagramas con Visio entre otras muchas funcionalidades. Se encuentra disponible para todos los sistemas operativos salvo Linux. Es un software de pago que dispone de varios planes con el objetivo de adaptarse a las necesidades del usuario final.

Justificación de uso

Su uso viene vinculado a que la Universidad de Almería llegó a un acuerdo con Microsoft para tener productos Microsoft de forma gratuita para los miembros de la propia universidad. Desde el punto de vista del autor de este trabajo, la comodidad que presenta frente a otras alternativas gratuitas es notoria. Se ha utilizado Word para redactar resúmenes para enviarlos a congresos donde solo aceptaban documentos en formato doc/docx. Para las presentaciones de distintos trabajos se ha hecho uso de PowerPoint. Excel se ha utilizado para ver en un formato distinto al de texto plano los resultados en formato de tabla que se obtenían de las ejecuciones permitiendo realizar cálculos sencillos que permitían un mejor análisis de los mismos. Por último, Visio se ha usado para modelar los diagramas de esta memoria.

3.4.4 RStudio

RStudio[20] es un entorno de desarrollo integrado (IDE) para R[21]. En su interfaz por defecto incluye una consola, un editor de código, una herramienta de depuración y un gestor del espacio de trabajo. Está disponible para Windows, Mac y Linux. RStudio aparece como una aplicación que permite trabajar con R de una forma más sencilla para el usuario.

Justificación de uso

RStudio se ha utilizado con dos finalidades. La primera es su uso con el lenguaje R. Algunas métricas como el AUC de la curva ROC se han calculado con scripts de este lenguaje. Por tanto, para depurarlos, RStudio ha sido la herramienta ideal. Por otro lado, la segunda utilidad que se le ha dado es debido a su paquete Knitr. Knitr permite usarse conjuntamente con otros lenguajes como Tex o Python y en ese sentido, para la documentación ha sido muy útil. Mismamente, RStudio se ha usado para comprobar que los resultados y los scripts eran correctos.

[20] https://www.rstudio.com/

[21] https://www.r-project.org/

3.4.5 TeXstudio

TeXstudio[22] es un IDE específico para el lenguaje TeX. Es uno de los editores más completos permitiendo corrección ortográfica interactiva, resaltado de sintaxis, control de versiones SVN, enlace entre el PDF y el código fuente, etc. Nació como una bifurcación de TexMaker[23] donde se buscaba añadir nuevas características. Es posible utilizarlo en casi cualquier plataforma.

Justificación de uso

Son varias las herramientas que permiten trabajar con Tex, sin embargo TexStudio es la más completa de todas proporcionando muchos accesos directos a ciertos comandos. Además permite usar otros lenguajes como Knitr, facilitando la gestión en algunas tareas de documentación. Su compatibilidad con Windows y Linux permite cambiar de sistema operativo sincronizando los trabajos y continuar usando la misma herramienta de documentación.

3.5 Infraestructura

Como se puede ver en los diferentes capítulos de esta memoria, la aplicación de técnicas informáticas en el proceso de desarrollo de nuevos fármacos requiere de muchos recursos para obtener resultados validados y en el menor tiempo posible. En ese sentido disponer de supercomputadores para realizar miles de experimentos con la mayor brevedad posible se hace un requisito indispensable.

Para la realización de este TFM, aparte del equipo del autor del trabajo, se ha tenido acceso a dos infraestructuras de alto rendimiento: la del grupo de la Universidad de Almería TIC-146 “Supercomputación y Algoritmos” y el “Supercomputing and Bioinnovation Center” de la Universidad de Málaga. En este último caso, el acceso ha sido posible por medio del equipo de la Universidad San Antonio de Murcia liderado por Horacio Pérez Sánchez. Las características de los equipos usados se presentan a continuación:

3.5.1 Máquina Local

TOSHIBA
Leading Innovation

Toshiba P50-B-10V. Procesador de 4ª generación Intel® Core™ i7-4710HQ con Intel® Turbo Boost Technology 2.0. Velocidad del procesador 2.50 / 3.50 Turbo GHz. 6 MB en el tercer nivel de caché. 8,192 MB de memoria RAM DDR3.

[22] http://www.texstudio.org/
[23] http://www.xm1math.net/texmaker/

Justificación de uso

Se utiliza para desarrollar el algoritmo y obtener resultados de experimentos individuales y puntuales.

3.5.2 TIC-146: Supercomputación y Algoritmos

El grupo de investigación de la Universidad de Almería TIC-146: Supercomputación y Algoritmos[24] ha disfrutado (y continúa) de varios proyectos Nacionales que han permitido adquirir de infraestructura para tareas de HPC. Las características del equipo usado se pueden ver en el Listado 3.1.

Una descripción completa de la infraestructura se puede ver en `http://www.hpca.ual.es/es/infraestructura`.

```
Cluster de 23 nodos + 54 CPUs + 11 GPUs (484 cores, 3.900 GB de RAM y
    ↪ 22.904 GB de almacenamiento):
Front-end: Bullx R423E3i. 2 Intel Xeon E5 2620 2 GHz (12 cores) y 64 GB
    ↪ RAM. Disco RAID de 16 TB.
18x Bullx R424-E3: 2 Intel Xeon E5 2650 (16 cores) y 64 GB de RAM. 128
    ↪ GB SSD.
2x Bullx R424-E3: 2 Intel Xeon E5 2650v2 (16 cores) y 128 GB de RAM. 1
    ↪ TB HDD.
2x Bullx R421-E4: 2 Intel Xeon E5 2620v3 (12 cores) y 64 GB de RAM. 1 TB
    ↪  HDD.
2x NVIDIA K80: 2 Kepler GK210 con 24 GB GDDR5 y 4.992 cores CUDA.
AMD ATI SAPPHIRE FIRE PRO S9100: 2.560 stream processors y 12 GB GDDR5.
Bullion S8: 8 Intel Xeon E7 8860v3 (16 cores) y 2,3 TB de RAM. 2x 300 GB
    ↪  SAS.
2x NextIO 2070
4x GPUs Tesla M2070 (1.792 cores)
Red de interconexión: Infiniband y Ethernet.
```

Listado 3.1: Infraestructura del TIC-146.

Justificación de uso

Los experimentos con problemas reales consisten en realizar cientos de miles de comparaciones. Para obtener resultados concluyentes es necesario ejecutar muchas ejecuciones con varias configuraciones y repetirlas para comprobar la estabilidad de los resultados. Todo ello se debe de realizar en el menor tiempo posible para analizar los resultados y actuar según se requiera. Esta infraestructura ha permitido conseguir este objetivo.

[24] `http://www.hpca.ual.es/es/`

3.5.3 Universidad San Antonio de Murcia

Gracias al equipo de colaboración de la Universidad San Antonio de Murcia se ha podido hacer uso del Supercomputing and Bioinnovation Center. El Supercomputing and Bioinnovation Center[25] de la Universidad de Málaga brinda apoyo científico y recursos a los grupos de investigación UMA, instituciones públicas y empresas privadas, especialmente en los campos de la informática, la nanotecnología y la biología. El equipo usado para los experimentos se puede ver en el Listado 3.2.

Una descripción completa de la infraestructura se puede ver en `http://www.scbi.uma.es/site/scbi/hardware`.

```
Cluster HP Intel E5-2670
48 HP-SL230G8 x 2 x 160 GLOPS = 16 TFLOPS
32 GPUS M2075 x 1030 GFLOPS = 33 TFLOPS
48 x 2 E5-2670 processors x 8 cores = 768 cores
2.60GHz per core
48 * 64 GB RAM = 3 TB total RAM
Infiniband FDR network at 54.54 Gbit/s
Shared scratch in lustre filesystem
```

Listado 3.2: Infraestructura del Supercomputing and Bioinnovation Center.

Justificación de uso

Véase la justificación de uso de la Subsección 3.5.2.

[25] `http://www.scbi.uma.es/site/`

4. Algoritmos Evolutivos

Este capítulo introduce al lector en el campo de la optimización y los algoritmos de búsqueda heurísticos. Concretamente se explicará de forma breve todos los conceptos necesarios para entender las bases sobre las que se ha desarrollado el algoritmo propuesto en este trabajo y que ocupará el siguiente capítulo.

4.1 Optimización global

El objetivo de la optimización global es encontrar la mejor solución (global) de los modelos, en presencia de múltiples soluciones óptimas locales y globales. Formalmente, la optimización global busca la solución global de un modelo de optimización delimitado por restricciones. Los modelos no lineales están omnipresentes en muchas aplicaciones, por ejemplo, en diseños de ingeniería, biotecnología, análisis de datos, gestión ambiental, planificación financiera o control de procesos, entre otras. Su solución a menudo requiere de un enfoque de búsqueda global.

La representación matemática de un problema de optimización global, en su forma de maximización, viene dada por:

$$\begin{aligned} &\text{máx } f(x) \\ &\text{t. q. } x \in X \end{aligned} \tag{4.1}$$

donde X es un conjunto cerrado no vacío en $\mathbb{R}^n$ y f es una función continua. Por tanto, el objetivo es encontrar el valor máximo f^* en todos los puntos $x^* \in X$ tal que:

$$f^* = f(x^*) \geq f(x) \quad \forall x \in X \tag{4.2}$$

La conversión de un problema de minimización en uno de maximización es sencillo por lo que minimizar $f(x)$ es equivalente a maximizar $-f(x)$ (Ecuación 4.3). En este capítulo, el

problema de optimización será tratado como un problema de maximización.

$$\text{mín}\, f(x)|x \in X = -\,\text{máx} - f(x)|x \in X \tag{4.3}$$

4.2 Algoritmos de búsqueda

En general, las técnicas de optimización clásicas (locales) tienen dificultades al tratar con problemas de optimización global. Una de las principales razones por las que fallan los métodos de optimización local es que se quedan atrapados fácilmente en óptimos locales. Además, las técnicas de optimización local no pueden generar ni usar la información global necesaria para encontrar el óptimo global para una función con múltiples óptimos locales. Para encontrar el óptimo global, es necesario un esfuerzo en el enfoque de la búsqueda. El campo de la optimización global estudia este tipo de problemas (Sección 4.1) que tienen uno o más óptimos globales.

Hay muchas taxonomías de estrategias de optimizadores globales. Un ejemplo puede encontrarse en [61], donde las estrategias de optimización se clasifican, de acuerdo al grado de rigor con el que se aborda al objetivo, en:

- *Métodos incompletos*, donde se usan heurísticas intuitivas inteligentes, pero que no pueden garantizar escapar de un óptimo local.
- *Métodos asintóticamente completos*. La probabilidad de obtener el óptimo global existe si se ejecuta indefinidamente. Sin embargo, no podemos asegurar que el óptimo global sea alcanzado.
- *Métodos completos*. Pueden alcanzar el óptimo global con certeza cuando se utilizan cálculos exactos y se ejecutan indefinidamente. También pueden encontrar un óptimo global aproximado, dentro de las tolerancias prescritas, después de un tiempo finito.
- *Métodos rigurosos*, donde el óptimo global es alcanzado con certeza, dentro de las tolerancias dadas, incluso con cálculos aproximados.

Otras clasificaciones de los métodos de optimización global pueden ser consultadas en [76, 88]. No obstante, actualmente es muy difícil proponer una clasificación de los algoritmos de búsqueda, debido a que se mezclan distintas técnicas para obtener el óptimo global.

Inicialmente, los métodos de optimización global se dividen en exactos y heurísticos, dependiendo de si pueden o no dar garantía de encontrar una solución óptima.

En los métodos exactos (tales como la optimización de Lipschitzian o Ramificación y Acotación (Branch and Bound)), apenas se incluyen factores aleatorios. Algunos de ellos convergen en el óptimo global bajo ciertas condiciones, pero cuando el algoritmo se detiene después de un número finito de iteraciones, la precisión de la solución puede que no se conozca con exactitud.

Los métodos exactos con terminación finita necesita de acceso detallado a la información global sobre el problema. Los métodos completos más eficientes generalmente combinan técnicas de ramificación con una o varias técnicas de optimización local, análisis convexo, análisis de intervalos o programación de restricciones.

Además, los algoritmos exactos garantizan encontrar, para cualquier instancia de tamaño finito de un problema, una solución óptima. Sin embargo, para problemas NP-complejos, todavía no existen algoritmos con tiempo polinomial. Para propósitos prácticos, los métodos exactos son muy costosos desde el punto de vista computacional por lo que los métodos heurísticos han recibido una atención incremental en los últimos años. En este tipo de métodos, la garantía de encontrar una solución óptima es sacrificada a cambio de obtener una buena solución en un periodo de tiempo suficientemente corto.

La heurística puede proporcionar soluciones útiles y prácticas para un amplio rango de problemas y dominio de aplicaciones. El poder de las heurísticas viene de su habilidad de tratar con problemas complejos cuando no hay conocimiento del problema o éste es escaso. Se adaptan particularmente bien a un amplio rango de aplicaciones de optimización computacionalmente intratables y de toma de decisiones.

Los métodos heurísticos pueden ser considerados algoritmos que realizan búsquedas aleatorias parciales de soluciones factibles, óptimas o cercanas a óptimas, para un problema hasta que se cumple una condición de parada concreta o después de que se alcance un número predefinido de iteraciones. Los algoritmos heurísticos no pueden garantizar que el conjunto de óptimos encontrados incluya al optimo global. Normalmente, estos algoritmos pueden encontrar una solución cuyo valor objetivo es cercano al optimo global y lo encuentran rápidamente y de forma sencilla. A veces, estos algoritmos pueden ser precisos, lo que significa que encuentra la mejor solución, pero siguen siendo heurísticos por lo que no se puede probar que la solución encontrada sea la mejor.

Usualmente, las heurísticas imitan estrategias exitosas encontradas en la naturaleza. Por ejemplo, las técnicas evolutivas copian los principios aplicados a las especies para desarrollar cualidades superiores durante las generaciones; el enfriamiento simulado está basado en como los cristales se unen cuando los materiales se enfrían y las partículas encuentran una estructura que minimiza el balanceo de energía; algunos métodos basados en poblaciones replican la inteligencia de grupos o el comportamiento social colectivo de animales, donde el todo es más que la suma de las habilidades de sus miembros, etc.

En este trabajo se ha seguido la taxonomía propuesta en [77], y que se encuentra representada en la Figura 4.1. En las siguientes secciones se describirá la rama de los algoritmos heurísticos y más concretamente el camino descendente hacia los algoritmos genéticos.

4.3 Algoritmos Heurísticos

En ciencias de la computación, hay dos objetivos fundamentales que los algoritmos tienen que alcanzar: demostrar tener una calidad buena u óptima en la solución y obtener buenos tiempos de ejecución. Una heurística es un algoritmo que abandona uno o ambos objetivos. Por ejemplo, una heurística encuentra normalmente soluciones buenas, pero esto no prueba que la solución pueda volverse arbitrariamente mala; o por lo general se ejecuta razonablemente rápido, pero esto no es argumento de que siempre sea así.

Como se mencionó anteriormente, puede que no estemos interesados en encontrar la solución

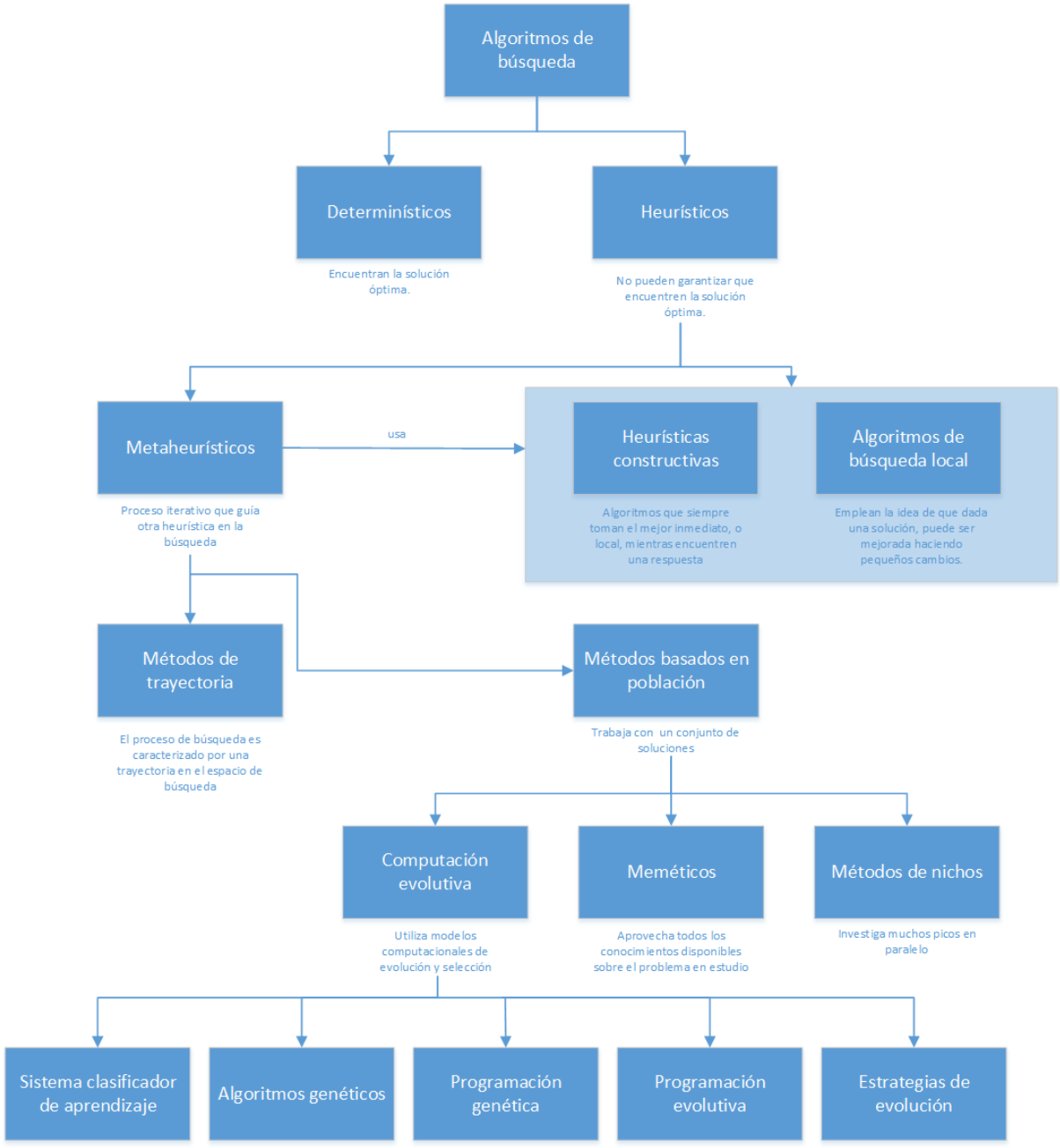


Figura 4.1: Una taxonomía de los algoritmos evolutivos.

óptima porque:

- El tamaño del problema a resolver supera los limites computacionales de los algoritmos óptimos conocidos en un tiempo de computación limitado.
- No vale la pena tanto esfuerzo (tiempo, dinero, etc.) para encontrar la solución óptima.

En tales casos, se puede usar un algoritmo heurístico que pueda encontrar una solución factible cercana al óptimo en términos del valor de la función objetivo. De hecho, ocurre con frecuencia que un algoritmo heurístico bien diseñado puede dar resultados con buena calidad (cercanos al óptimo). Las principales ventajas de los algoritmos heurísticos frente a los algoritmos exactos se basan en que son (a menudo) conceptualmente más simples y (en la mayoría de los casos) menos costosos computacionalmente.

Entre los métodos heurísticos, es posible distinguir entre heurísticas constructivas y métodos de búsqueda local [6]. Este trabajo se va a centrar en los métodos de búsqueda local.

4.3.1 Métodos de búsqueda local

Un método de búsqueda local se basa en la idea de que dada una solución S, ésta pueda ser mejorada haciendo pequeños cambios. Estas soluciones obtenidas modificando una solución S, se llaman *vecinos de* S, y la aplicación de un operador que produce un vecino S' se llama comúnmente *movimiento*.

Un algoritmo de búsqueda local comienza con una solución inicial y se mueve de vecino a vecino mientras el valor de la función objetivo se incremente. El principal inconveniente de esta estrategia es su dificultad de escapar de óptimos locales donde la solución no se encuentra en ninguno de los vecinos que mejora el valor de la función objetivo.

En el Algoritmo 1 se muestra una búsqueda local de mejora iterativa donde cada movimiento solo es realizado si el resultado de la solución es mejor que la solución actual. El algoritmo se detiene tan pronto como encuentre un óptimo local.

Algoritmo 1 Búsqueda local de mejora iterativa (IILS)

1: Generar una solución inicial S
2: **while** un vecino S' de S existe si $f(S') > f(S)$ **do**
3: Se selecciona el vecino mejorado S'
4: Se asigna S a S'
5: **output:** S

4.4 Algoritmos metaheurísticos

El término metaheurístico, introducido por primera vez en [28], tiene su origen en la composición de dos palabras griegas. Heurístico proviene del verbo *heuriskein* que significa “encontrar”, mientras que el prefijo *meta* significa “más allá, en el nivel superior”. Algunos algoritmos muy conocidos son considerados metaheurísticos: optimización aleatoria, búsqueda local, algoritmos

genéticos, enfriamiento simulado, búsqueda Tabú, optimización de colonia de hormigas, GRASP, búsqueda de difusión estocástica, optimización extrema generalizada y búsqueda de Harmony. Se han propuesto innumerables variantes e híbridos de estas técnicas, y se han publicado muchas más aplicaciones de metaheurísticas para problemas específicos. Éste es un campo activo de la investigación, con una considerable bibliografía, una gran comunidad de investigadores y usuarios, y un amplio rango de aplicaciones.

Una metaheurística puede definirse como un marco de trabajo algorítmico de alto nivel o un enfoque que puede estar especializado en solucionar problemas de optimización. Consiste en un proceso iterativo, que guía a otras heurísticas en la búsqueda de soluciones factibles. Las metaheurísticas son aproximadas y normalmente no determinísticas. Generalmente son aplicadas a problemas para los que no hay un algoritmo específico del problema satisfactorio o heurístico, o cuando no es práctico implementar dicho método. No son específicos del problema, pero pueden hacer uso de conocimientos específicos del dominio en forma de heurísticas que son controladas por una estrategia de nivel superior.

Una metaheurística exitosa tiene que proporcionar una equilibrio entre la explotación de la experiencia de la búsqueda acumulada y la exploración en el espacio de búsqueda para identificar regiones con mayor nivel de calidad de las soluciones. El equilibrio entre exploración y explotación es muy importante, por un lado para identificar regiones rápidamente en el espacio de búsqueda con soluciones de gran calidad y por otro lado para no perder demasiado tiempo en las regiones de búsqueda del espacio que ya han sido exploradas y no proporcionan soluciones de gran calidad [9]. La principal diferencia entre las metaheurísticas existentes se basa en la forma particular en que tratan de lograr este equilibrio. Los diferentes enfoques metaheurísticos pueden caracterizarse por diferentes aspectos basados en el camino de búsqueda que siguen, cómo se explota la memoria, el uso de poblaciones de soluciones, el número de vecindarios considerados y, por qué no, las fuentes inspiradas. En [7] se presenta una discusión sobre estos aspectos.

4.4.1 Métodos basados en poblaciones

En secciones anteriores, se ha hablado sobre métodos de resolución de problemas clásicos, incluyendo métodos exactos, y algoritmos de búsqueda local. Para estos métodos, dado un espacio de búsqueda y una función objetivo, algunos de ellos siempre devolverían la misma solución (por ejemplo, métodos determinísticos), mientras que otros podrían generar diferentes soluciones basadas en la configuración inicial o punto de partida.

Hay una interesante observación compartida por todas estas técnicas: cada una se basa en una única solución como base para la exploración futura con cada iteración. O bien procesan soluciones completas en su totalidad (búsqueda local), o bien construyen la solución final a partir de bloques de construcción más pequeños (ramificación y acotación). A pesar de sus diferencias, cada uno de estos algoritmos trabaja o construye una sola solución a la vez.

Las metaheurísticas basadas en poblaciones (PBM) [9] son algoritmos que trabajan con un conjunto de soluciones (es decir, una población) al mismo tiempo en vez de con una única solución. A primera vista, quizás parezca que esta idea no proporciona realmente nada nuevo, ya que los algoritmos anteriores se podrían ejecutar k veces para incrementar la probabilidad de encontrar el óptimo global. Pero hay un componente adicional que puede hacer que los

algoritmos basados en poblaciones sean esencialmente diferentes de otros métodos de resolución: el concepto de competencia entre soluciones de una población. Es decir, simular el proceso evolutivo de la competencia y selección y dejar que las soluciones candidatas en la población luchen por tener espacio en las generaciones futuras. De esta manera, los algoritmos basados en poblaciones proporcionan una forma natural e intrínseca de explorar el espacio de búsqueda.

Algunos de los métodos basados en población más estudiados son la computación evolutiva y la optimización de colonias. El primero se estudiará a continuación. El último está fuera del enfoque de este trabajo, pero el lector puede acudir a [2, 18, 19, 99] para una descripción en profundidad del algoritmo y sus aplicaciones.

4.5 Computación evolutiva

La computación evolutiva (EC) es una técnica moderna de búsqueda que usa modelos computacionales de procesos de evolución y selección [46, 51]. Los conceptos y mecanismos de la evolución darwiniana y la selección natural están codificados en algoritmos evolutivos (EAs) y se utilizan para resolver problemas en muchos campos de la ingeniería y la ciencia [13].

El fuerte parecido a los procesos biológicos, así como sus aplicaciones iniciales para el modelado de sistemas adaptativos complejos influyó en la terminología utilizada por los investigadores de la EC. Tiene mucho de genética, de teoría evolutiva y de biología celular, tanto es así que una solución candidata a un problema se llama *individuo*, mientras que el conjunto entero (superconjunto) de soluciones actuales se llama *población*. Para algunos dominios de problemas, una población puede dividirse en varias *subpoblaciones* o *especies*. La representación real (codificación) de un individuo se llama *genoma* o *cromosoma*. Cada genoma consiste en una secuencia de genes, es decir, atributos que describen a un individuo. Un valor de un gen se llama *alelo*. Cuando las soluciones individuales son modificadas para producir nuevas soluciones candidatas, se dice que se reproducen y la nueva solución candidata se denomina *descendencia* o *hijo*. Durante la evaluación de una solución candidata, ésta recibe una calificación denominada *aptitud*, que indica la calidad de la solución en el contexto del problema dado. Cuando la población actual es reemplazada por sus descendientes, la nueva población se llama *nueva generación*. Finalmente, todo el proceso de búsqueda de una solución óptima se denomina *evolución* [37].

El Algoritmo 2 describe la estructura básica de un EA. Inicialmente, se crea una población de individuos generados aleatoriamente (o individuos obtenidos de otras formas tales como heurísticas constructivas). La aptitud es usada para determinar el mérito relativo de cada individuo. Una vez que la población inicial se ha obtenido, se lleva a cabo un proceso iterativo. En cada iteración, se genera una nueva descendencia usando un operador de recombinación, normalmente un cruce entre dos o varios padres [5, 21]. Otras técnicas usan estadísticas de población para generar hijos [60, 87].

Para introducir algo de ruido en el proceso de búsqueda con el fin de evitar la convergencia en un óptimo local, se aplica un operador de mutación a los individuos progenitores. En algunas aplicaciones, se utilizan pequeños cambios aleatorios como mecanismo de mutación, pero en otros, resulta ser bastante beneficioso usar métodos de mejora para aumentar la aptitud de los individuos. Los algoritmos evolutivos que aplican un algoritmo de búsqueda local para cada

individuo de una población se denominan algoritmos meméticos. Éstos se basan en la idea de que mientras que el uso de una población asegura una exploración del espacio de búsqueda, el uso de técnicas de búsqueda local ayuda a identificar rápidamente áreas buenas en el espacio de búsqueda. Sin embargo, cuando se aplica búsqueda local puede ocurrir una convergencia prematura hacia soluciones subóptimas. Con el fin de evitar este inconveniente, existen, además del uso de un operador de mutación aleatoria, numerosas maneras de mantener la diversidad de la población. Algunas de estas formas son la agrupación [16], compartir aptitud [32] y usar nichos [53]. Finalmente, los individuos compiten cada uno entre el resto y también contra los padres para mantenerse en la población en la siguiente iteración. Esto es hecho mediante un esquema de selección.

Algoritmo 2 Algoritmos evolutivos (EA)

1: Generar una población inicial
2: Se evalúa la población
3: **while** la condición de parada no se cumpla **do**
4: Se recombina la población para obtener una nueva generación
5: Se muta la generación para obtener una nueva población
6: Se evalúa la nueva población
7: Se seleccionan los individuos de la nueva población que serán considerados para la siguiente generación
8: **output:** mejor solución encontrada

Existen técnicas similares que difieren en detalles de implementación y en la naturaleza del problema, como la programación genética [47, 48], la programación evolutiva [22], la estrategia evolutiva [6] y el sistema clasificador de aprendizaje [49]. Sin embargo, el tipo más popular de algoritmos evolutivos es el algoritmo genético, que se describirá brevemente en la siguiente sección [30, 37].

4.6 Algoritmo genético

Los algoritmos genéticos (GAs) son una subfamilia de los algoritmos evolutivos (EAs) basados en principios evolutivos y genéticos. Los GAs son aplicados en biogenética, ciencias de la computación, ingeniería, economía, química, manufacturación, matemáticas, física y otros campos.

Un algoritmo genético funciona modificando repetidamente una población de estructuras artificiales mediante la aplicación de operadores genéticos. Usan técnicas de herencia, mutación, selección y recombinación. Los GAs son normalmente métodos de caja negra que usan exclusivamente la información de aptitud; no necesitan conocimiento interno del problema. En optimización, el objetivo es encontrar la mejor solución posible o soluciones a un problema, con respecto a uno o más criterios.

Un GA necesita normalmente:

- De una estructura adecuada para representar soluciones en el dominio de búsqueda , y
- Una función de aptitud para evaluar el dominio de la solución.

La función de aptitud se define sobre la representación genética, mide la calidad de la solución representada y es siempre dependiente del problema.

Tradicionalmente, una estructura de datos de un algoritmo genético consiste en una cadena de 0s y 1s. El término *cromosoma* se usa con frecuencia para referirse a la cadena. Sin embargo, los GAs no están restringidos a una representación de una cadena de bits, sino que pueden usar otras codificaciones, como por ejemplo vectores de números reales [15, 24, 29, 31] o programas informáticos de alto nivel [47]. Aunque las estructuras de longitud variable se utilizan en muchos problemas, para este trabajo nos centraremos en las estructuras con cadenas de l bits.

Cada cromosoma (cadena) es una concatenación de un número de subcomponentes llamados genes. Los genes se encuentran en varias posiciones o lugares del cromosoma, y toman valores llamados alelos. En la representación de cadena de bits, un gen es un bit, un lugar es su posición en la cadena, y un alelo es el valor (0 o 1). El término biológico genotipo se refiere a la composición genética general de un individuo, y corresponde a una estructura en el GA. El término fenotipo se refiere a las características exteriores de un individuo, y corresponde a una estructura decodificada en el GA.

Como se dijo anteriormente, para optimizar una estructura usando un GA, uno debe ser capaz de definir una medida de calidad de cada estructura en el espacio de búsqueda. La función de aptitud es la responsable de esta tarea. En funciones de maximización, la función objetivo a menudo actúa como la función de aptitud. Las funciones de aptitud también existen en dominios de aplicación distintos de la optimización de funciones matemáticas. Para problemas de optimización combinatoria como el problema del viajante [30, 38], la distancia de la ruta es una buena elección como función de aptitud. Para la optimización del peso en redes neuronales [96, 97], la suma de errores cuadrados en un conjunto de ejemplos de entrenamiento puede servir como función de aptitud.

Una vez que la representación genética y la función de aptitud han sido definidas, el GA procede a inicializar una población de forma aleatoria (soluciones candidatas de un problema). A continuación, la población se mejora mediante la aplicación repetida de operadores de mutación, cruce y selección (Figura 4.2). Normalmente, el algoritmo termina cuando se ha producido un número máximo de generaciones o se ha alcanzado un nivel satisfactorio de aptitud para la población. Si el algoritmo ha terminado debido a un número máximo de generaciones, una solución satisfactoria puede o no haber sido alcanzada. A continuación, se describen en detalle los componentes del bucle del GA.

4.6.1 Selección

Durante cada generación sucesiva, una proporción de la población existente se selecciona para criar a una nueva generación. Esta selección de los individuos se realiza a través de un proceso basado en la aptitud, donde las soluciones más adecuadas (calculado mediante una función de aptitud o función objetivo) tienen normalmente más probabilidades de ser seleccionadas. Ciertos métodos de selección valoran la aptitud de cada solución y seleccionan las mejores soluciones. Otros métodos califican sólo a una muestra aleatoria de la población, haciendo que el proceso consuma menos tiempo.

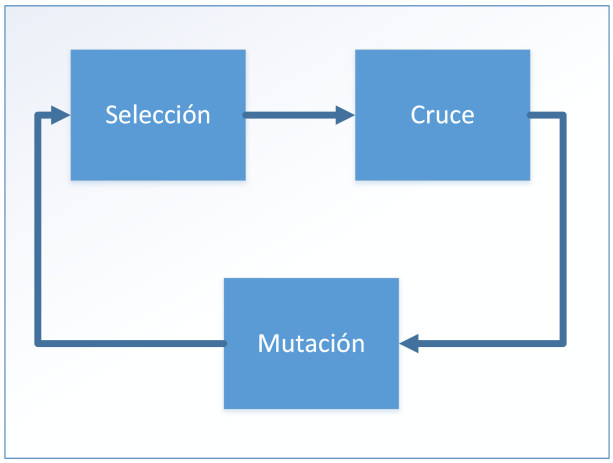


Figura 4.2: Bucle básico de un GA.

La mayoría de las funciones de selección están diseñadas de tal manera que se fuerza a que se escoja también una pequeña proporción de soluciones menos aptas. Esto ayuda a mantener una amplia diversidad de la población, evitando la convergencia prematura en soluciones pobres. Los métodos de selección populares y bien estudiados son:

Selección por ruleta

En la selección por ruleta (también conocida como selección proporcional de aptitud), se calcula una probabilidad de selección para cada individuo. Esta probabilidad está dada por:

$$p_s(i) = \frac{f(i)}{\sum_{j=1}^{n} f(j)} \tag{4.4}$$

donde $f(i)$ es la aptitud del individuo i y n es el número de individuos en la población. Las soluciones candidatas con mayor aptitud tendrán menos probabilidad de ser eliminadas. Sin embargo, con la selección proporcional de aptitud, existe la posibilidad de que las soluciones más débiles sobrevivan al proceso de selección; esto es una ventaja puesto que, aunque una solución puede ser débil, puede incluir algunos componentes que podrían resultar útiles después del proceso de recombinación.

La analogía con la rueda de una ruleta consiste en imaginar una rueda de una ruleta en la que cada solución candidata representa una casilla de la rueda. El tamaño de las casillas es proporcional a la probabilidad de selección de la solución. Seleccionar n cromosomas de la población equivale a jugar n juegos en la ruleta, ya que cada candidato se elije independientemente (Figura 4.3).

Selección por torneo

Se usan n torneos para elegir n individuos. Cada torneo consiste en muestrear k elementos de la población (el tamaño del torneo) y elegir el más apto. Véase [8] para un análisis detallado de la selección de torneos.

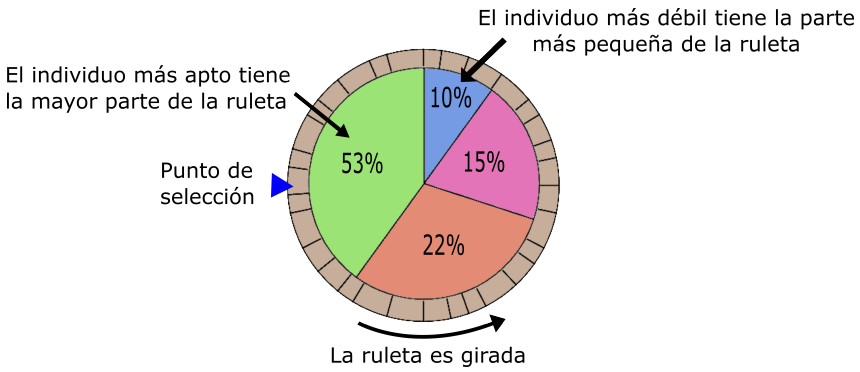


Figura 4.3: Selección por ruleta.

La presión de selección se puede ajustar fácilmente cambiando el tamaño del torneo. Si el tamaño del torneo es muy grande, los individuos débiles tienen una menor probabilidad de ser seleccionados. La relación entre el tamaño del torneo y la presión de selección también se muestra en [8].

Selección elitista

Asegura que el mejor elemento o elementos de la población sobrevivan de generación en generación. La estrategia elitista más básica copia el mejor elemento de la población actual a la siguiente, si ese elemento no ha sido transferido a través del proceso normal de selección, cruce y mutación. Cualquier método de selección estándar puede hacerse elitista.

Selección por clasificación

A cada individuo de la población se le asigna un número basado en la aptitud, y la selección se basa en esta clasificación en lugar de diferencias absolutas en la aptitud. La ventaja de este método es que puede evitar que los individuos muy aptos ganen dominio temprano a expensas de los menos aptos, lo que reduciría la diversidad genética de la población y podría obstaculizar los intentos de encontrar una solución aceptable.

Selección de estado estacionario

Los descendientes de los individuos seleccionados de cada generación se introducen en el grupo de sus progenitores, reemplazando a algunos de los miembros menos aptos de la generación anterior. Algunos individuos son retenidos entre generaciones. Algunas aplicaciones del proceso de selección en estado estacionario se pueden encontrar en [23, 42].

Selección generacional

Los descendientes de los individuos seleccionados de cada generación se convierten en la próxima generación. Ningún individuo es retenido entre generaciones.

Selección hereditaria

Los individuos pasan por múltiples rondas de selección en cada generación. Las evaluaciones del nivel inferior son más rápidas y menos discriminatorias, mientras que las que sobreviven a niveles más altos se evalúan con mayor rigor. La ventaja de este método es que reduce el tiempo de cómputo global utilizando una evaluación más rápida y menos selectiva para eliminar a la mayoría de los individuos que muestran poca o ninguna promesa y sólo retener a los individuos que sobreviven a esta prueba inicial.

4.6.2 Cruce

El cruce es un operador genético que combina dos cromosomas (padres) para producir un nuevo cromosoma (hijo). La idea detrás del cruce es que el nuevo cromosoma puede ser mejor que ambos padres si toma las mejores características de cada uno de ellos. El cruce ocurre durante la evolución de acuerdo con una probabilidad de cruce definida por el usuario. Existen muchas técnicas de cruce, pero en esta sección limitamos nuestra atención a las que se pueden aplicar a cadenas individuales de l bits.

- *Cruce de un punto*. Como muestra la Figura 4.4, en esta técnica de cruce, los cromosomas de los padres se cortan en algún punto común elegido al azar y los subcromosomas resultantes son intercambiados [73].
- *Cruce de dos puntos*. Se seleccionan al azar dos sitios de cruce y los cromosomas progenitores intercambian el segmento que se encuentra entre los dos sitios de cruce. Un ejemplo de este tipo de cruce se puede ver en la Figura 4.5.

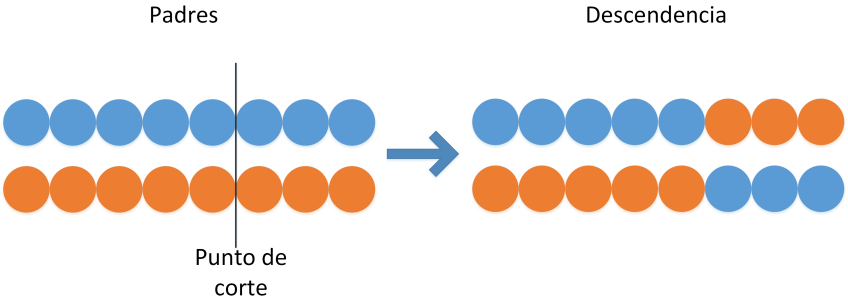


Figura 4.4: Cruce de un punto.

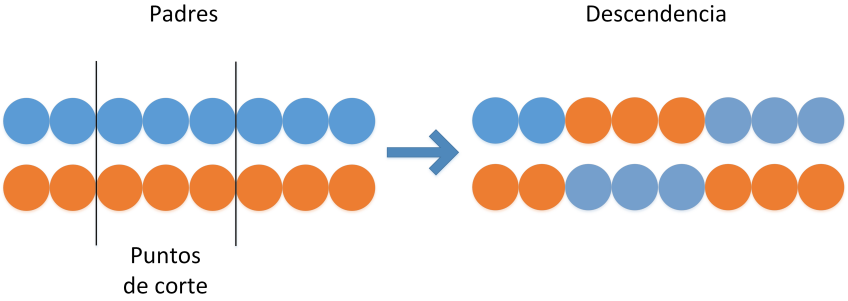


Figura 4.5: Cruce de dos puntos.

Cruce uniforme

Cada gen del descendiente se selecciona aleatoriamente de los genes correspondientes de los padres. Téngase en cuenta que un cruce de un punto y dos puntos producen dos descendientes, mientras que el cruce uniforme produce sólo uno. Ver Figura 4.6.

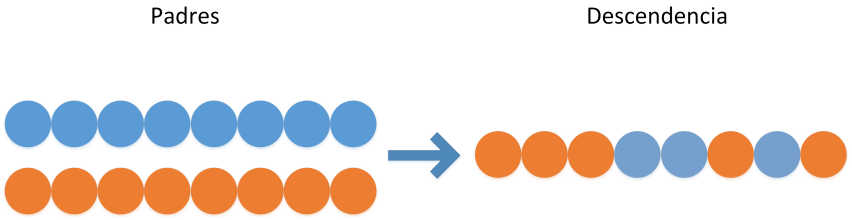


Figura 4.6: Cruce uniforme.

4.6.3 Mutación

La mutación es un operador genético que altera uno o más valores de los genes en un cromosoma desde su estado inicial. Esto puede producir valores de genes totalmente nuevos que se agregan al conjunto de genes. Con estos nuevos valores de genes, el algoritmo genético puede ser capaz de aumentar la diversidad de la población y por tanto la probabilidad de escapar de un óptima local. La mutación ocurre durante la evolución según una probabilidad de mutación definida por el usuario. En los cromosomas binarios, consiste en invertir bits aleatorios del genotipo.

4.6.4 Probabilidad de aplicación de los operadores genéticos

Como se mencionó anteriormente, tanto el operador de cruce como de mutación se aplican de acuerdo con una probabilidad definida por el usuario. Esta sección se dedica a explicar estas probabilidades.

Probabilidad de cruce (P_c). Su valor normalmente es elegido en el rango {0.5,...,0.8}. Esto significa que el cruce es aplicado a individuos con una probabilidad de P_c y clonación con una probabilidad de $1 - P_c$.

Probabilidad de mutación (P_m). Ésta es la probabilidad de cambiar cualquier bit de forma individual. En los GAs, P_m es normalmente muy bajo, generalmente en el rango [0.001, 0.01]. Si el valor es muy alto, la búsqueda se convierte en una búsqueda primitiva aleatoria. Una buena regla podría ser $P_m = 1/l$, donde l es el número de bits del cromosoma.

4.7 Fortalezas y debilidades de los GAs

En [54], están descritas algunas de las ventajas e inconvenientes de los GAs. Esta sección simplemente las resume.

Fortalezas de los GAs

- Los algoritmos genéticos son intrínsecamente paralelos. Muchos algoritmos sólo pueden explorar el espacio de la solución a un problema en una dirección a la vez. Sin embargo, dado que los GA tienen descendencia múltiple, pueden explorar el espacio de la solución en múltiples direcciones a la vez.
- Los algoritmos genéticos son adecuados para resolver problemas en los que el espacio de todas las soluciones potenciales es extremadamente grande - demasiado extenso para buscar exhaustivamente en un tiempo razonable. La mayoría de los problemas que se conocen de este tipo son *no lineales*. En un problema lineal, la aptitud de cada componente es independiente, por lo que cualquier mejora de una parte dará lugar a una mejora del sistema como un todo. Sin embargo, en uno no lineal el cambio de un componente puede tener efectos alternos en todo el sistema, y múltiples cambios que, individualmente, son perjudiciales, pueden conducir a mayores mejoras en la aptitud cuando se combinan.
- Los algoritmos genéticos funcionan bien en problemas en los que la función de aptitud es compleja (discontinua, ruidosa, cambia con el tiempo, o tiene muchos óptimos locales). Muchos algoritmos de búsqueda pueden quedar atrapados en los óptimos locales. Los algoritmos evolutivos, y en particular los GAs, han demostrado ser eficaces para escapar de los óptimos locales y descubrir el óptimo global, incluso en un relieve generado por una función de aptitud muy accidentada y compleja. Normalmente no hay manera de saber si una solución dada a un problema es el óptimo global o simplemente un óptimo local muy alto, pero es posible decir que un GA casi siempre puede ofrecer una solución muy buena.
- Los algoritmos genéticos pueden manipular muchos parámetros simultáneamente. Muchos problemas del mundo real no se pueden expresar en términos de un valor único que debe maximizarse, sino que debe expresarse en términos de objetivos múltiples, generalmente con balanceos involucrados: uno sólo puede ser mejorado a expensas de otro. Los GAs son muy buenos para resolver estos problemas.
- Los GAs no saben nada sobre los problemas que están resolviendo. En lugar de utilizar información específica de dominio previamente conocida para guiar cada paso y hacer cambios con sentido hacia la mejora, van a ciegas, hacen cambios aleatorios a sus soluciones candidatas y luego usan la función de aptitud para determinar si esos cambios producen una mejora.

Limitaciones de los GAs

- La primera consideración en la creación de un algoritmo genético es la definición de una representación para el problema actual. Esto no es trivial, ya que no existe una metodología para hacerlo. La representación seleccionada tiene que tolerar cambios aleatorios en las soluciones y evitar, al mismo tiempo, tanto errores graves como resultados absurdos, que no satisfagan las limitaciones del problema.
- El problema de cómo escribir la función de aptitud debe ser cuidadosamente considerado para que la aptitud más alta sea alcanzable y realmente sea equivalente a una mejor solución al problema dado. Si la función de aptitud se elige mal o se define de manera imprecisa, el algoritmo genético puede ser incapaz de encontrar una solución al problema, o puede terminar resolviendo el problema de forma equivocada.
- Además de hacer una buena elección de la función de aptitud, los otros parámetros de un GA (el tamaño de la población, la tasa de mutación y cruce, el tipo y la fuerza de la selección) también deben elegirse con cuidado. Si el tamaño de la población es demasiado

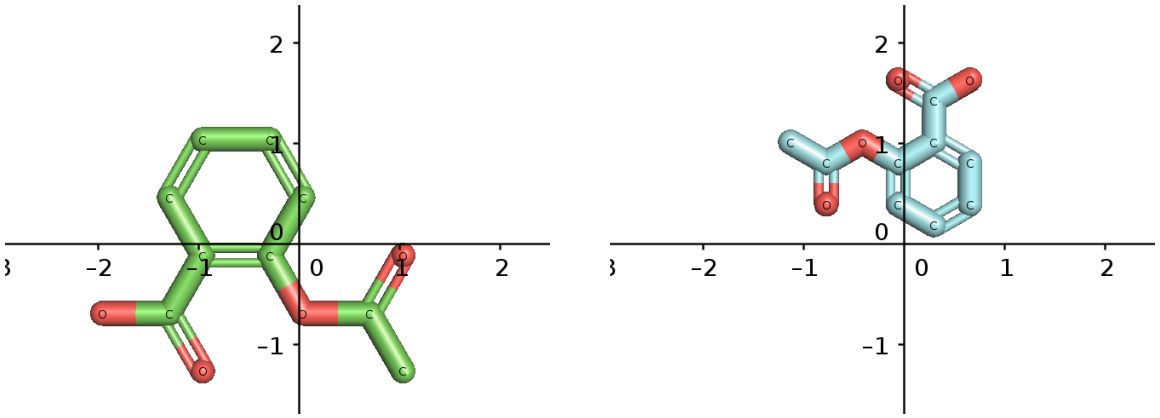
pequeño, el algoritmo genético puede no explorar el espacio de la solución suficiente para encontrar consistentemente buenas soluciones. O si la tasa de cambio genético es demasiado alta o el esquema de selección es elegido vagamente, puede que la población cambie con demasiada frecuencia impidiendo la convergencia hacia una solución.

- Un problema bien conocido que puede ocurrir con un GA es la convergencia prematura. Si un individuo, que tiene mejor valor de aptitud que la mayoría de sus competidores, emerge temprano, puede reproducirse demasiado disminuyendo la diversidad de la población demasiado pronto. Esto significa que el algoritmo converge hacia el óptimo local que representa el individuo, en lugar de buscar en profundidad en toda la función de aptitud para encontrar el óptimo global.

5. Maximización del solapamiento

La forma molecular es un concepto muy importante en el diseño de fármacos asistido por ordenador. La base que fundamenta esta idea es que si varios compuestos (ligandos) reaccionan por una misma zona con otra molécula dada (receptor), entonces existe la posibilidad de que la forma de la molécula esté influyendo en esa reacción [35]. En las últimas décadas se han diseñado varios métodos para comparar las formas moleculares y se han presentado numerosas aplicaciones para el cribado virtual, el *scaffold hopping* y el alineamiento molecular basado en la forma característica [1, 26, 57, 58, 59, 75, 94]. No obstante, existen algunos detalles que tienen que ser considerados a la hora de analizar los resultados obtenidos por un método de optimización de la forma molecular:

- *Radio de van der Waals*. Es un radio imaginario asignado a cada átomo cuyo valor es distinto para cada elemento. Se utiliza para la representación de los átomos como esferas sólidas. El trabajo realizado por A. Bondi [10] sobre el cálculo de los radios es actualmente el más respaldado científicamente, aplicándose a varios trabajos [11, 55, 63, 65, 95, 98].
- *Posición de los átomos en la molécula*. Los átomos que constituyen una molécula no permanecen fijos durante la existencia de ésta sino que dependiendo de diferentes factores modifican su proximidad, ángulos de los enlaces, etc. Esto es un detalle muy importante en el momento de seleccionar los compuestos de las bases de datos. En caso de que se utilicen varias fuentes, la misma molécula se puede encontrar espacialmente en distinta posición y la distancia entre sus átomos puede ser diferente. En la Figura 5.1 se pueden ver dos moléculas de ácido acetilsalicílico obtenidas de dos bases de datos diferentes. La Figura 5.1a pertenece a Chemspider [71], mientras que la Figura 5.1b pertenece DrukBank [50]. Se trata del mismo compuesto, pero el tamaño y la posición son completamente distintos. Para solucionar este inconveniente se pueden utilizar distintos métodos, desde el sencillo uso de una única fuente de moléculas hasta técnicas avanzadas como la isomería conformacional.



(a) Molécula obtenida de ChemSpider.

(b) Molécula obtenida de DrugBank.

Figura 5.1: Dos moléculas de ácido acetilsalicílico obtenido de dos bases de datos diferentes.

5.1 Evolución del cálculo de la similitud de forma

Para realizar una comparación de dos compuestos mediante la forma, es fundamental definir un modelo lo más realista posible, aunque la exactitud no es el único factor relevante en este proceso. Como se introdujo en el Capítulo 1, el cribado virtual se utiliza para seleccionar una cantidad reducida de moléculas (compuestos, proteínas, etc.) de entre millones, que posteriormente serán tratadas en el laboratorio. Esto requiere que los tiempos computacionales sean los más bajos posibles. Por tanto, es necesario alcanzar un equilibrio entre calidad y velocidad con el modelo.

Los dos modelos más usados actualmente son el de esferas sólidas [12, 56] y el de la representación gaussiana de las moléculas [35, 36]. Ambos se detallarán en las siguientes subsecciones.

5.1.1 Modelo de esferas sólidas

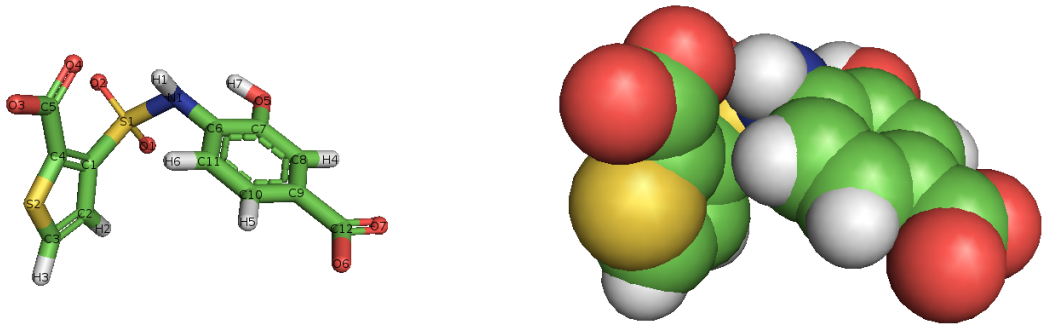
Este modelo consiste en representar cada uno de los átomos de las moléculas como si de esferas se tratase. El radio de éstas viene definido por el radio de van der Waals. En la Figura 5.2 se puede ver una molécula representada con este modelo. A la izquierda (Figura 5.2a) aparece su representación con enlaces y en la derecha (Figura 5.2b), con esferas sólidas. Nótese la diferencia de tamaño de los distintos elementos como en el caso del azufre (amarillo) y el hidrógeno (blanco).

La representación matemática de la función de densidad de la molécula se corresponde con la Ecuación 5.1.

$$H(r) = \sum_i h_i(r) - \sum_{i<j} h_i(r)h_j(r) + \sum_{i<j<k} h_i(r)h_j(r)h_k(r) - \sum_{i<j<k<l} h_i(r)h_j(r)h_k(r)h_l(r) + \ldots \tag{5.1}$$

donde $h_i(r)$ toma el siguiente valor:

$$h_i(r) = \begin{cases} 1, (|r - r_i| \leq \sigma_i) \\ 0, (|r - r_i| > \sigma_i) \end{cases} \tag{5.2}$$



(a) Molécula representada mediante enlaces. (b) Molécula representada mediante esferas solidas.

Figura 5.2: Distinta representación de las moléculas.

σ_i es el radio de van der Waals del átomo i, r la posición a evaluar en el espacio y r_i la posición de dicho átomo.

A modo gráfico, el valor que toma $h_i(r)$ queda representado en la Figura 5.3. A_1 y A_2 son los centros de dos átomos cualesquiera con sus respectivas esferas. El resto de puntos se corresponden con r, es decir, el punto evaluado en el espacio. Entre ellos se diferencian tres grupos: los puntos verdes D_x, que se encuentran dentro de las esferas de los átomos, para ellos $h_i(r) = 1$; los puntos morados O_x, que se encuentran fuera de cualquier esfera por lo que $h_i(r) = 0$. Los puntos rojos I_x, pertenecen al grupo de los verdes ($h_i(r) = 1$), pero debido a que se encuentran en la intersección de dos esferas generan un cierto problema cuya solución explica los signos alternos en la Ecuación 5.1.

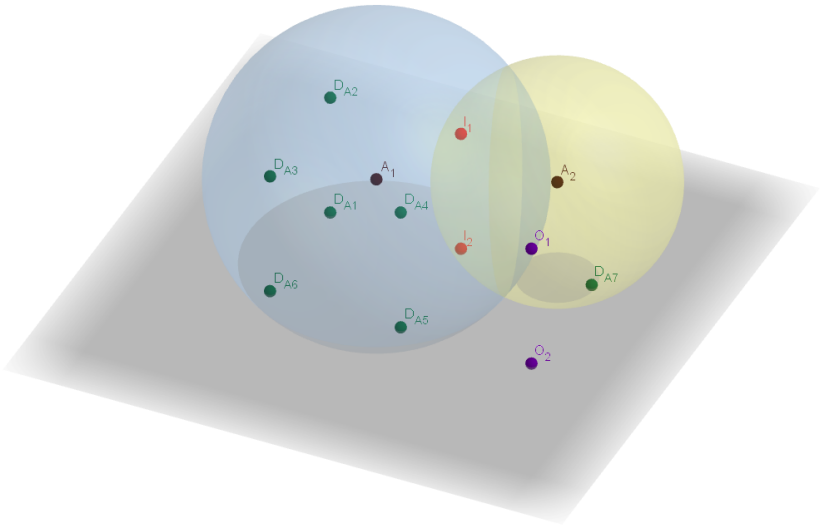
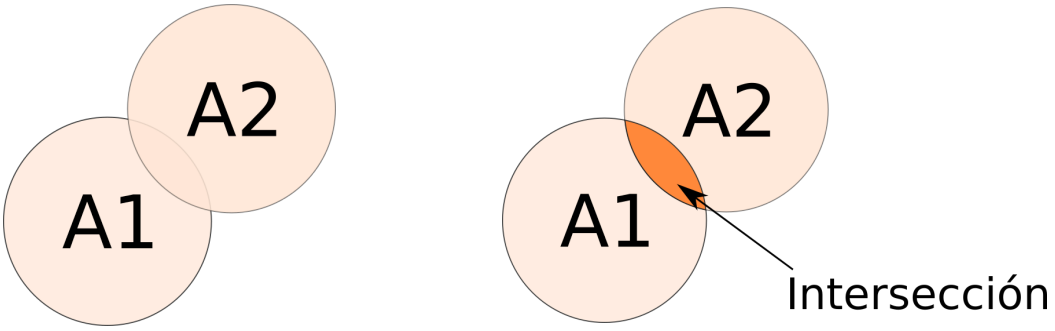


Figura 5.3: Evaluación de $h_i(r)$.

La Ecuación 5.1 se compone de tantos términos como átomos tiene la molécula evaluada. La suma y resta alterna de términos tiene su fundamento en la corrección del cálculo de la sobreestimación del volumen. Para entender este último concepto, considérese solamente los dos primeros términos de la serie. El primero obtiene un volumen de la molécula igual a la

suma de todos los volúmenes de los átomos suponiendo que éstos están aislados, sin embargo, la situación real de los átomos es que se encuentran solapados (Figura 5.4a). El segundo término precisamente corrige este valor, eliminando las intersecciones de los volúmenes de los pares de átomos (5.4b). El resto de la serie seguiría el mismo procedimiento.



(a) Primer término. (b) Segundo término.

Figura 5.4: Superposición de moléculas.

Una vez definida la función de densidad de la molécula (Ecuación 5.1) se puede calcular el volumen. Debido a que la forma y el volumen son dos conceptos estrechamente relacionados, el cálculo del volumen a partir de la función de densidad consiste en integrar este última en todo el espacio (Ecuación 5.3).

$$V = \int H(r)dr = \sum_{i} v_i - \sum_{i<j} v_{ij} + \sum_{i<j<k} v_{ijk} - \ldots \tag{5.3}$$

donde v_i representa el volumen de un átomo y su valor se obtiene de $v_i = \frac{4\pi\sigma_i^3}{3}$.

La intersección de dos volúmenes de un par de átomos viene dada por la Ecuación 5.4.

$$v_{ij} = \int h_i(r)h_j(r)dr \tag{5.4}$$

El principal problema con este método es su implementación, ya que aunque la expresión analítica para el volumen y sus derivadas son conocidas [27, 74], las fórmulas para la intersección de esferas múltiples no son triviales. Por esta razón, el método gaussiano propuesto por Good y Richards [33, 34] se ha utilizado más ampliamente.

5.1.2 Modelo gaussiano

Este modelo sustituye las esferas sólidas por funciones gaussianas. En consecuencia, cada átomo ahora está representado por una función gaussiana [35, 36]. La representación de la Ecuación 5.1 se convierte en la Ecuación 5.5.

$$G(r) = \sum_{i} g_i(r) - \sum_{i<j} g_i(r)g_j(r) + \sum_{i<j<k} g_i(r)g_j(r)g_k(r) - \ldots = 1 - \prod_{i}[1 - g_i(r)] \tag{5.5}$$

donde $g_i(r)$ es igual a:

$$g_i(r) = pe^{-(\frac{3p\pi^{1/2}}{4\sigma^3})^{2/3}(r-r_i)^2} \tag{5.6}$$

Véase la similitud con la expresión genérica de Gauss (Figura 5.5). Con este modelo se obtiene mejor aproximación que con el de las esferas sólidas (donde el valor era 1 dentro de la esfera y 0 fuera) de la distribución de las partículas subatómicas dentro del átomo. La gráfica de la función es simétrica con forma de campana, conocida como campana de Gauss. El parámetro a (p, valor obtenido empíricamente) es la altura de la campana centrada en el punto b (r_i), determinando c (σ_i, el radio de van der Waals) el ancho de la misma.

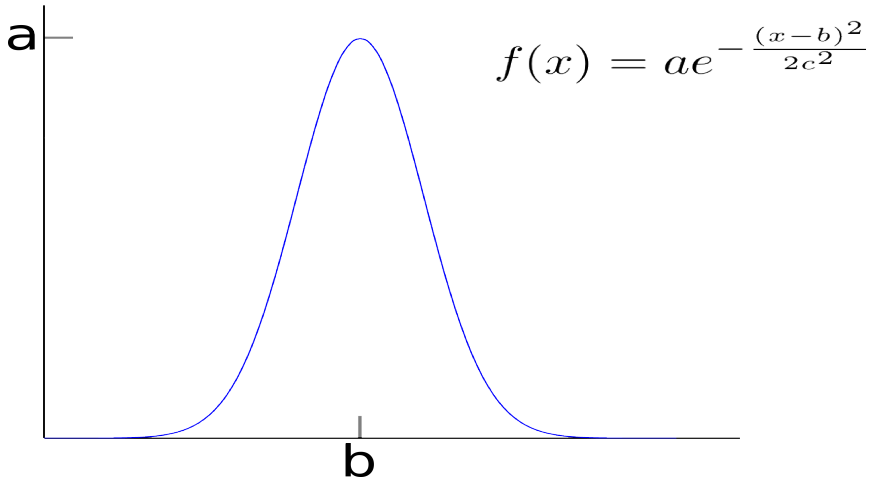


Figura 5.5: Función gaussiana genérica.

Por último, igual que en el modelo anterior, la integral de la función de densidad de dos átomos permite obtener el volumen intersectado de ambos átomos (Ecuación 5.7).

$$v_{ij}^g = \int g_i(r)g_j(r)dr \tag{5.7}$$

Conociendo ya el valor de $G(r)$ y haciendo uso de la propiedad de las funciones gaussianas que permite obtener una gaussiana a partir del producto de otras dos, si consideramos la función de densidad de dos moléculas $G_A(r)$ y $G_B(r)$, su integral (al igual que la Ecuación 5.7) nos permite conocer el volumen intersección entre ambas moléculas.

En la expresión original se recomienda la inclusión de seis términos de la serie de la Ecuación 5.5, aunque en la práctica, el primer término es suficiente para obtener una buena precisión [62].

Este modelo representó una gran mejora en la representación de las moléculas y el cálculo de la similitud de forma y muchas herramientas lo han implementado, siendo la más conocida de ellas ROCS [64]. Pero todavía a este modelo era necesario añadirle más información sobre las moléculas. Esto se explicará en la siguiente subsección.

5.1.3 WEGA (Weighted Gaussian Functions, Funciones gaussianas ponderadas)

El modelo gaussiano no recibe información del estado de los átomos ni diferencia entre la posición de los átomos en distintas sustancias. Y éste es un concepto a considerar puesto que existen sustancias en las que los átomos se encuentran más juntos que otras. Por ejemplo, en el Adamantano, los átomos de carbono se superponen fuertemente con sus vecinos, en consecuencia, esta molécula tiene una sobreestimación del solapamiento. En el lado contrario, se encuentra el gas noble Argón, una molécula de átomo único donde el volumen de solapamiento es exactamente igual al volumen de la esfera sólidas [98].

Para ajustar mejor el valor de este solapamiento entre átomos, Yan et al. [98] han propuesto añadir un factor de peso a cada átomo en las moléculas evaluadas. Por tanto, el volumen de la forma de una molécula se expresa como una combinación lineal de funciones gaussianas ponderadas en base a cada átomo. La expresión es la siguiente:

$$G(r) = \sum_i w_i g_i(r) = \sum_i w_i p e^{-(\frac{3p\pi^{1/2}}{4\sigma_i^3})(r-r_i)^2} \tag{5.8}$$

donde $w_i = \frac{v_i^g}{v_i^g + k\sum_{j\neq i} v_{ij}^g}$, siendo k una constante universal.

El volumen de solapamiento de dos moléculas viene dado por:

$$V_{AB}^g = \sum_{i\in A, j\in B} w_i w_j v_{ij}^g \tag{5.9}$$

Actualmente éste es el modelo matemático que mejor resultados obtiene y por tanto se ha seleccionado como función objetivo para este trabajo fin de máster.

La aportación del artículo de Yan et al. [98] no solo es la adición de un parámetro de ajuste en la función objetivo, sino que incorporan dicha función a un nuevo algoritmo denominado WEGA. WEGA es un algoritmo determinístico multistart que utiliza un optimizador local basado en el BFGS. Este algoritmo obtiene mejores resultados que, hasta el momento la considerada la mejor herramienta de comparación de forma, ROCS [64]. Es por este motivo que todas las comparaciones que se realicen con el algoritmo desarrollado en este trabajo sean contra WEGA.

5.1.4 Normalización de los resultados

El valor de solapamiento obtenido en la Ecuación 5.9 no se encuentra acotado por ningún límite superior (el límite inferior es 0). Cuanto mayor sea el número de átomos de las moléculas, mayor espacio ocuparán y por tanto el valor de solapamiento posiblemente sea mayor. Esto no permite realizar una comparación de resultados ni contrastar la calidad con otros experimentos y/o algoritmos. Por tanto, para resolver esta problemática se utiliza la métrica de Tanimoto (Ecuación 5.10). Esta métrica requiere del cálculo de solapamiento de las dos moléculas consigo mismas, y el solapamiento entre ambas. A partir de estos datos, realiza una ponderación devolviendo un valor entre 0 y 1. Si el valor devuelto es 0, significa que no existe solapamiento entre ambas

moléculas; sin embargo, si el valor es 1, el solapamiento es total. De esta forma se obtiene un valor normalizado e independiente del tamaño de las moléculas que se puede utilizar para los estudios y comparaciones pertinentes.

$$S_{AB} = \frac{V_{AB}}{V_{AA} + V_{BB} - V_{AB}} \tag{5.10}$$

5.2 Optipharm. Características generales

A continuación, se presentará Optipharm, un algoritmo que se ha desarrollado en este trabajo para dar solución al problema planteado del cribado virtual basado en la similitud de forma de las moléculas. En la presente sección se tratarán los temas que conciernen a la naturaleza del algoritmo.

5.2.1 Conceptos básicos

Optipharm encuentra sus orígenes en UEGO [45], en consecuencia, es un algoritmo multimodal que es capaz de resolver problemas de optimización multimodales (donde la función objetivo tiene múltiples óptimos locales) y descubrir la estructura de estos óptimos [43, 66, 68]. En el dominio multimodal, cada pico puede ser considerado como un nicho. La analogía con la naturaleza es que dentro de un área hay diferentes subespacios, nichos, que pueden soportar diferentes tipos de vida (especies u organismos). Los *métodos de niching*, *clustering* o *especiación* son técnicas que promueven la formación y mantenimiento de subpoblaciones en los Algoritmos Evolutivos (EA). Un método de niching debe ser capaz de formar y mantener múltiples soluciones diversas con valores de aptitud idénticos o distintos. Estos mecanismos permiten a un EA identificar múltiples óptimos en un dominio multimodal, y no sólo un único óptimo global. Optipharm es un método de clustering que ofrece una solución al llamado problema del radio de nicho, que es un problema común de muchas técnicas de niching como el *fitness sharing* [17], *simple iteration* o *sequential niching* [4]. Este problema está relacionado con funciones que tienen múltiples óptimos locales y cuyas óptimos se distribuyen de manera desigual por todo el espacio de búsqueda. En tales funciones, el radio del nicho no se puede establecer correctamente ya que si es demasiado pequeño la búsqueda resulta ineficaz y si es demasiado grande no se pueden distinguir los óptimos locales que están demasiado cerca entre sí. La solución de Optipharm implica una técnica de *enfriamiento* que permite a la búsqueda centrarse en las regiones prometedoras del espacio, comenzando con un radio relativamente grande que disminuye a medida que la búsqueda avanza.

Esta técnica de enfriamiento de radio tiene algunas similitudes con el concepto de mutación variable utilizado en el Algoritmo Genético Orientado a Clústeres (vmCOGA) que no necesita establecer ningún radio de nicho [69]. Este algoritmo utiliza diferentes tasas de mutación durante la búsqueda, por lo que en las etapas iniciales, la tasa de mutación es alta para asegurar un muestreo eficiente de todo el espacio de diseño; y en etapas posteriores, la tasa de mutación es menor para promover la convergencia de la búsqueda y la formación de conjuntos.

En los EA multimodales, el concepto de subpoblación, el cual es fundamental en Optipharm, es sustituido por especie en muchos métodos de niching o especiación [3, 43, 44, 67]. Así que

una especie sería equivalente a una subpoblación o grupo en un algoritmo evolutivo basado en subpoblaciones.

Otra característica importante de Optipharm es que no es necesario conocer con antelación el número de especies. Esto no es algo trivial, y aunque algunos algoritmos no necesitan conocer el número total de conjuntos tampoco [52, 69], la mayoría de algoritmos de niching sí requieren conocer esta información. Por ejemplo, en el algoritmo genético basado en islas [39] los grupos evolucionan de forma independiente e intercambian cierta información. Por lo general se implementa en entornos distribuidos, donde cada conjunto reside en un procesador independiente, y por lo tanto el número de conjuntos está determinado por el número de procesadores. En Optipharm, como en vmCOGA [69], la información de la especie se extrae dinámicamente durante la búsqueda y se crea una lista de especies encontradas.

En Optipharm todas las especies están destinadas a ocupar un máximo local de la función de aptitud, sin conocer el número total de máximos locales existentes en el relieve de la función. Esto significa que cuando el algoritmo comienza no sabe cuántas especies habrá al final. Para ello, Optipharm utiliza un conjunto de especies que pueden superponerse y que definen subdominios del espacio de búsqueda. A medida que avanza el algoritmo, el proceso de búsqueda puede dirigirse hacia regiones más pequeñas creando nuevos conjuntos de especies que definen subdominios más pequeños. Este proceso guarda similitud con el enfriamiento simulado. Una especie en particular no es una parte fija del dominio de búsqueda sino que puede moverse a través del espacio mientras que la búsqueda continúa. Además, Optipharm es un algoritmo memético que introduce un optimizador local en el proceso de evolución. De esta manera, en cada generación Optipharm realiza una operación de optimización local en cada especie, y estas soluciones óptimas localmente reemplazan a las especies que estaban al inicio del proceso. Optipharm es abstracto en el sentido de que la gestión de especies y el mecanismo de enfriamiento han sido separados lógicamente del algoritmo de optimización local. Por lo tanto, es posible implementar cualquier tipo de optimizador para trabajar dentro de una especie.

La gestión de las especies es una de las partes centrales de Optipharm. Consiste en procedimientos para crear, seleccionar y eliminar especies durante todo el proceso de optimización. Una especie puede ser considerada como una *ventana* en todo el espacio de búsqueda (ver Figura 5.6). Esta ventana está definida por su centro y su radio. El centro es una solución, y el radio es un número positivo. En particular, para el problema de la similitud de forma, una especie es un array de la forma $(x, f(x), R)$ (también almacenamos información sobre el valor objetivo en el centro de la especie). Teniendo en cuenta que el centro es una solución, una especie podría ser equivalente al concepto de individuo en un algoritmo evolutivo estándar. Sin embargo, una especie tiene una zona de atracción, definida por su radio, que cubre una región del espacio de búsqueda y por lo tanto múltiples soluciones.

El papel principal de la ventana es definir la región de búsqueda para el optimizador local. Si el valor de una nueva solución encontrada por el optimizador es mejor que el del centro antiguo, la nueva solución se convierte en el centro y la ventana se mueve manteniendo el mismo valor del radio.

Como se ha mencionado anteriormente, en Optipharm todas las especies están destinadas a determinar un máximo local de la función de aptitud, sin conocimiento del número total de máximos locales existentes en el relieve de la función de aptitud. Esto significa que cuando

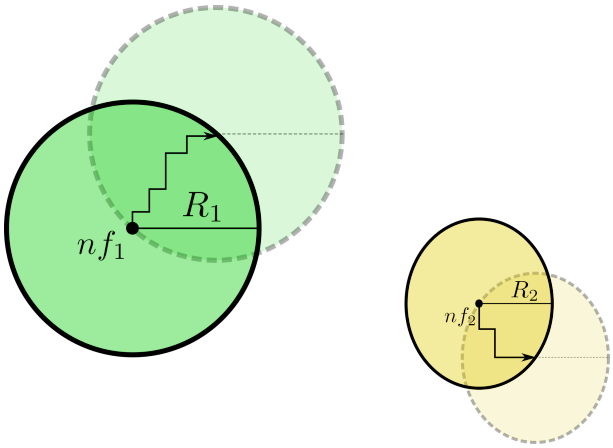


Figura 5.6: Proceso de optimización de especies.

el algoritmo comienza no sabe cuántas especies habrá al final. Esto implica que el número de especies no se fija por adelantado, y la introducción de los conceptos de creación y selección de especies es necesario. Al principio existen cinco especies y, a medida que el algoritmo evoluciona y se aplican operadores genéticos, pueden crearse nuevas especies y algunas de las especies existentes eliminarse. Dados dos padres (puntos aleatorios en el área de atracción de una especie), una nueva especie es creada cuando es probable que los padres están en diferentes colinas. Es interesante observar que una ubicación en el espacio de búsqueda puede pertenecer a diferentes especies y que los operadores de reproducción, es decir, la optimización y creación de nuevas especies, se aplican dentro del área cubierta por la especie, que evoluciona de forma independiente. Esto también le sucede a otros algoritmos evolutivos multimodales, como los GA basados en islas [39] y los GA ecológicos [14]. ECOGA fue el primer EA espaciado estructuralmente en la que cada individuo de la población global ocupa una sola ubicación en el espacio y se forman subpoblaciones o agrupaciones recopilando a los individuos de lugares cercanos. Las operaciones genéticas están limitadas dentro del grupo y una localización (individuo) puede pertenecer a más de un conjunto, como en Optipharm, donde una posición puede pertenecer a más de una especie.

Otra característica especial de Optipharm es el radio de la especie, que no es constante a lo largo de la ejecución del algoritmo ni el mismo para cada especie. Este radio es una función monótona que disminuye a medida que aumenta el *nivel* (o ciclos o generaciones). El parámetro *nivel (L)* indica el número máximo de niveles en el algoritmo, es decir, el número de diferentes etapas de enfriamiento. La función del radio se mantiene constante en todos los niveles. La función del radio se define de tal manera que coincide con el diámetro del dominio inicial en el primer nivel y converge a cero cuando el número de niveles tiende hacia el infinito.

La función de radio se utiliza para controlar la velocidad de enfriamiento. De hecho, dado un nivel, da la temperatura del sistema en ese nivel. El radio de una especie creada en el nivel i, viene dado por la siguiente función exponencial, donde R_L y R_1 son los radios dados (parámetros de entradas) más pequeño y más grande, respectivamente:

$$R_i = R_1(\frac{R_L}{R_1})^{\frac{i-1}{L-1}} \quad (i = 2,...,L) \tag{5.11}$$

Cada vez que se crea una nueva especie, se asociará con un radio, cuyo valor depende del *nivel* actual. De esta manera, las especies que se han creado en diferentes niveles, tienen radios diferentes. Las especies creadas en los niveles mayores tienen el radio más pequeño. Por lo tanto, las especies con pequeños radios se comportan como si estuvieran más frías, examinan un área relativamente pequeña, su movimiento en el espacio es más lento pero son capaces de diferenciar entre óptimos locales que están relativamente cerca el uno del otro.

Durante el proceso de optimización, una lista de especies es mantenida por Optipharm. Este concepto, *species_list*, sería equivalente al término de población en un algoritmo evolutivo. Optipharm es de hecho un método para manejar esta *species_list* (es decir, crear, eliminar y optimizar especies). La longitud máxima de la lista de especies está dada por el parámetro de entrada *max_spec_num* (tamaño máximo de la población).

Como se mencionó anteriormente, este algoritmo no sólo crea nuevas especies, sino que también selecciona algunas de ellas, eliminando el resto. El mecanismo de selección consiste en elegir de todo el listado *species_list*, aquellas que son más aptas eliminando tantas especies como número de especies existentes exceda el parámetro de entrada *max_spec_num*.

5.2.2 Parámetros de Optipharm

En Optipharm, al igual que en UEGO, los parámetros más importantes son los que se definen en cada nivel: el radio (R_i) y el número máximo de evaluaciones para la creación de especies (new_i) y para la optimización (n_i). Estos parámetros se calculan a partir de los siguientes parámetros de entrada proporcionados por el usuario:

- *evals (N)*: El número máximo de evaluaciones para todo el proceso de optimización. Nótese que el número real de evaluaciones es normalmente menor que este valor.
- *levels (L)*: El número máximo de niveles (es decir, etapas de enfriamiento).
- *max_spec_num*(M): Longitud máxima del listado de especies o el tamaño máximo de la población.
- *min_r*(R_L): El radio asociado con el nivel máximo.

Como se mencionó anteriormente, existe una relación estrecha entre los parámetros de entrada (N, L, M y R_L) y los parámetros en cada nivel (R_i, new_i y n_i). Las ecuaciones que unen estos parámetros se detallan en [43, 67], donde también se formulan algunas definiciones y principios. A efectos de exhaustividad, todas las cosas relacionadas con todo el conjunto de parámetros, se detallará a continuación.

En primer lugar, se introduce la noción de la *velocidad del optimizador*. El optimizador local sólo puede mover una especie en la región de búsqueda definida por su ventana, la cual está determinada por un radio. Para ciertos optimizadores que utilizan relieves ideales (como funciones lineales) con la ayuda de modelos matemáticos o resultados experimentales, la distancia que una especie se puede mover durante el proceso de optimización puede ser aproximada en función de su radio asociado. Esto conduce naturalmente a una noción de velocidad que caracterizará un dominio dado (suponiendo, por ejemplo, un relieve lineal) y dependerá del radio de la especie. La velocidad se denota por $v(R)$ y para un nivel i se calcula como:

$$v(R_i) = \frac{\binom{dim}{\frac{dim-1}{2}}}{2^{dim+1}} \cdot R_i \quad (i = 2, ..., L) \tag{5.12}$$

donde *dim* indica la dimensión del problema a resolver. El método de parametrización está basado en los siguientes principios [43, 67]:

- *Principio de la igualdad de oportunidades*. Este principio asegura que cada especie reciba un cierto número de evaluaciones, lo que bastará para moverlo desde su centro original, al menos, a una distancia fija. Esta distancia común se define por $R_1 v$, donde v es un umbral que controla directamente la distancia que una especie puede cubrir. En realidad, controla la probabilidad de que una especie eventualmente represente un óptimo local: cuanto más lejos pueda ir una especie, mayor es la probabilidad de alcanzar un óptimo local (y más costosa es la optimización). Recuerde que R_1 es siempre el diámetro del espacio de búsqueda. Por lo tanto, el principio puede ser formalizado:

$$\frac{v(R_i)n_i}{M} = R_i v \quad (i = 2, ..., L). \tag{5.13}$$

- *Principio de disminución del radio exponencial*. Dado el radio más pequeño y el más grande (R_L y R_1), los radios restantes pueden ser expresados con la siguiente función exponencial:

$$R_i = R_1 (\frac{R_L}{R_1})^{\frac{i-1}{L-1}} \quad (i = 2, ..., L). \tag{5.14}$$

- *Principio de oportunidad de creación de especies constante*. Este principio asegura que incluso si la longitud de la lista de especies es máxima, existe la posibilidad de crear, al menos, dos especies más para cada especie antigua. También hace una simplificación fuerte, a saber, que todas las evaluaciones deben fijarse al mismo valor constante.

$$new_i = 3M \quad (i = 2, ..., L). \tag{5.15}$$

$$\sum_{i=1}^{L} (new_i + n_i) = (L-1)3M + \sum_{i=1}^{L} n_i = N \tag{5.16}$$

Ahora es posible desarrollar un método para determinar el parámetro n_i a partir de los parámetros dados por el usuario (N, L, M y R_L). Primero vamos a expresar el número máximo de evaluación de funciones (N) usando los parámetros del algoritmo. Supongamos $new_1 = 0$ para simplificar, ya que new_1 nunca es usado por Optipharm. Por lo tanto, la ecuación puede escribirse haciendo uso de (5.15) en el proceso. Otra simplificación es posible estableciendo $n_1 = 0$ cuando $l > 1$. Observe que si $l = 1$, entonces Optipharm se reduce al optimizador local. Expresando n_i de (5.13) y sustituyéndolo en (5.16), es posible escribir

$$(L-1)3M + \sum_{i=2}^{L} \frac{MR_1 v}{v(R_i)} = N \tag{5.17}$$

Usando la Ecuación 5.14, se puede observar que los parámetros desconocidos en la Ecuación 5.17 son sólo los parámetros dados por el usuario y, debido a la monotonía de esta ecuación en cada variable, cualquiera de los parámetros puede ser dado usando métodos numéricos efectivos. Los parámetros importantes restantes (n_i, new_i y R_i) se

pueden obtener usando los principios anteriores. Téngase en cuenta, sin embargo, que algunas de las configuraciones establecidas por el usuario pueden ser inviables.

5.2.3 Descripción algorítmica de Optipharm

En el Algoritmo 3 se da una descripción global de Optipharm. A continuación se describen las diferentes etapas clave del algoritmo:

- *Init_species_list*: Se crea una nueva lista de especies que consiste en cinco especies en el nivel 1. Teniendo en cuenta que el radio asociado en el nivel 1 es el diámetro del espacio de búsqueda, estas especies siempre cubren todo el espacio de búsqueda.
- *Create_species(create_evals)*: Para cada especie en la lista, se crean soluciones candidatas aleatorias en la ventana de la especie, y por cada par de soluciones candidatas, se computa el punto medio del segmento que las conecta. Se evalúan tanto el punto medio como los extremos y si alguna de estas soluciones tiene un mejor valor de aptitud que el centro de la especie, entonces ese punto será el nuevo centro, manteniendo el mismo valor de radio. Además, si el valor de la aptitud en un punto medio es peor que en los miembros correspondientes de su par, entonces los miembros del par se insertan en la lista de especies (ver Figura 5.7). La motivación detrás de este método es crear especies que se encuentran en diferentes *colinas*, asegurando así que haya un valle entre las nuevas especies. A cada nueva especie insertada se le asigna el valor del *nivel* actual (i). Como resultado de este procedimiento, la lista de especies contendrá finalmente varias especies con diferentes niveles (y por tanto, radios diferentes).
  En términos de la Computación Evolutiva, este procedimiento puede ser interpretado como un mecanismo para crear descendencia, donde una especie genera nuevas especies que están dentro de su área de atracción, aunque más tarde la nueva especie puede moverse hacia nuevas regiones.
  El parámetro *create_evals* está dividido internamente por el número actual de especies existentes ($length(species\_list_i)$). Esto significa que cada especie en la lista tiene un número fijo de evaluaciones para la creación de nuevos puntos, igual a:

$$budget\_per\_species = new_i / length(species\_list_i). \tag{5.18}$$

  Por lo tanto, el número NP de los nuevos puntos que se pueden generar por especie está limitado por:

$$NP + \binom{NP}{2} \leq budget\_per\_species \tag{5.19}$$

  (recordar que para cada par de puntos, el punto medio del segmento que conecta el par también debe ser evaluado).
  Ampliando la ecuación anterior, la cantidad total de nuevos puntos que se pueden crear por especie está dada por:

$$NP = \frac{\sqrt{(1 + 8 \cdot budget\_per\_species)} - 1}{2} \tag{5.20}$$

  Obsérvese que cuando el número de especies en la lista de especies es pequeño, entonces se permite que cada especie genere más puntos de prueba, mientras que el número de puntos de prueba es menor cuando la lista de especies contiene más especies.

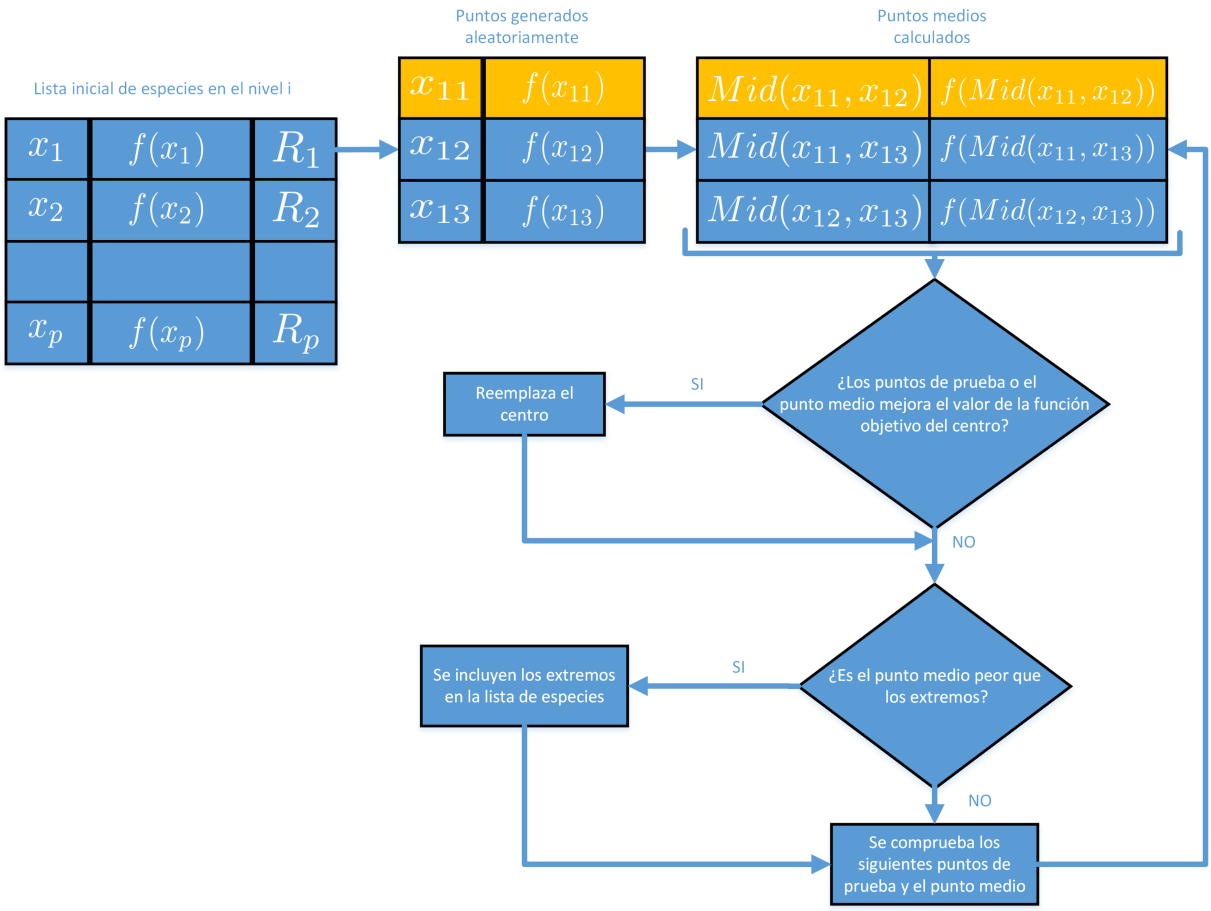


Figura 5.7: Proceso de creación de especies.

Aunque cada par (y su correspondiente punto medio) se evalúa independientemente de los restantes, el número total de evaluaciones por especie es limitado de antemano. Por lo tanto, este proceso de generación está siempre bajo control.

- *Elitist_selection (max_list_length)*: se eliminan especies para reducir la longitud de la lista al valor *max_list_length*. Se ordenan las especies de mayor a menor valor de aptitud y se eliminan las especies menos aptas hasta que el número de éstas sea inferior o igual al del valor *max_list_length*.
- *Optimize_species(opt_evals)*: Ejecuta el optimizador para cada especie con un número determinado de evaluaciones (*budget_per_species*) (ver Figura 5.6). Este presupuesto se calcula dividiendo el número máximo de evaluaciones asignadas al procedimiento de optimización en el nivel *i* (*opt_evals*), por el número máximo de especies o el tamaño máximo de la población (*max_spec_num*). Esto significa que si el número actual de especies ($length(species\_list_i)$) en el nivel *i* es menor que el máximo permitido, el número de evaluaciones de la función es menor que *opt_evals*. Además, el optimizador llamado por una especie no puede utilizar más evaluaciones que el máximo pasado como argumento (*budget_per_species*). En este trabajo se ha utilizado el optimizador SASS [83].

Tenga en cuenta que Optipharm termina cuando ha ejecutado todos sus niveles. Por lo tanto, el número final de evaluaciones depende de la complejidad del problema en cuestión.

Algoritmo 3 Algoritmo Optipharm

```
1: Init_species_list
2: Optimize_species(n_1)
3: for i = 2 to L do
4:     Determine R_i, new_i, n_i
5:     Create_species(new_i)            ▷ # budget_per_species = new_i/length(species_list_i)
6:     Elitist_selection(max_spec_num)
7:     Optimize_species(n_i)            ▷ # budget_per_species = n_i/max_spec_num
```

5.3 Optipharm. Características específicas

Después de explicar los fundamentos en los que se basa Optipharm y que mayormente provienen de UEGO, se van a detallar todos los conceptos específicos y propios del cribado virtual utilizando la forma molecular que se han incorporado al algoritmo. Y es que si bien se ha analizado ya la función objetivo que se pretende maximizar, el problema que se afronta en este trabajo se encuentra constituido de cientos de miles de subproblemas individuales e independientes entre sí, pero que deben de producir un resultado como unidad, siendo ésta la principal problemática de este tipo de problemas. A continuación se explican las decisiones específicas que se han propuesto para este trabajo.

5.3.1 Dimensiones del problema

Para el correcto funcionamiento de Optipharm, es necesario definir las dimensiones que tendrá el problema para, de esta forma, encontrar el mejor valor de cada una y que en su conjunto se obtenga el mejor valor de la función objetivo. Uno de los principales inconvenientes de este tipo de problemas es que es necesario realizar modificaciones en el espacio sobre una de las moléculas que se está comparando. Estas modificaciones consisten en trasladar y girar la molécula. Para ello son necesarios 10 parámetros: 1 para definir el ángulo que se va a rotar la molécula (θ), 6 para definir los dos puntos que forman el eje ($x1$, $y1$, $z1$, $x2$, $y2$ y $z2$) y 3 más para definir la traslación en el espacio (Δ_x, Δ_y y Δ_z).

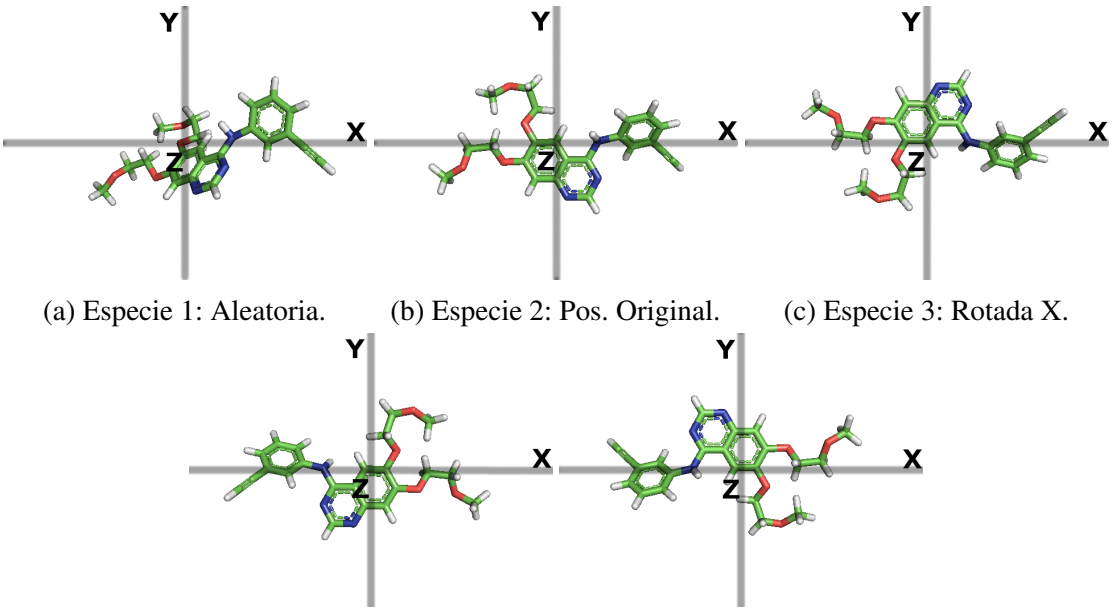
5.3.2 Especies iniciales

Como se dijo en la método de inicialización de especies, en la segunda parte de este capítulo, cinco son las especies iniciales de Optipharm. Los valores de sus centros son mostrados en la Tabla 5.1. La primera especie siempre toma valores aleatorios definidos cada uno de ellos entre su respectivo límite superior e inferior. El resto de especies tienen valores fijos, independientemente del par de moléculas a comparar. Estas especies representan la molécula variable en su posición inicial, rotada 180º en el eje X, rotada 180º en el eje Y y rotada 180º en el eje Z, respectivamente. En la Figura 5.8 se muestra un ejemplo de las cinco especies iniciales para el receptor celular del factor de crecimiento epidérmico (EGFR).

La elección de estas cinco especies se encuentra fundamentada en los resultados empíricos que se han obtenido con cientos de miles de comparaciones. La experimentación realizada fue la siguiente: se fijaron los valores $x1$, $x2$, $x3$, $x4$, $x5$ y $x6$, y se optimizó el ángulo y la traslación.

Especie	θ	x1	y1	z1	x2	y2	z2	Δ_x	Δ_y	Δ_z
1	random	random	random	random	random	random	random	random	random	random
2	0	1	0	0	0	0	0	0	0	0
3	π	1	0	0	0	0	0	0	0	0
4	π	0	1	0	0	0	0	0	0	0
5	π	0	0	1	0	0	0	0	0	0

Tabla 5.1: Centro de las cinco especies iniciales.



(a) Especie 1: Aleatoria. (b) Especie 2: Pos. Original. (c) Especie 3: Rotada X.

(d) Especie 4: Rotada Y. (e) Especie 5: Rotada Z.

Figura 5.8: Un ejemplo de especies iniciales de Optipharm para el EGFR.

Los resultados que se obtuvieron mostraban que las rotaciones en el eje X, Y o Z eran las más repetidas.

5.3.3 Limitación de movimiento

Optipharm es un algoritmo evolutivo global. Esto significa que puede encontrar todos los óptimos locales en todo el espacio de búsqueda. Este espacio viene determinado por los límites superiores e inferiores de cada uno de los parámetros (que en sí mismos componen la dimensión del espacio). Teniendo esto en cuenta, reducir al máximo este espacio ayudará a reducir el espacio de búsqueda y las especies podrán afinar mejor los valores para esos parámetros.

Aquí se presenta uno de los principales problemas que se han tenido que afrontar en este trabajo. Se realizan millones de comparaciones entre pares de moléculas y cada una tiene un tamaño específico. En consecuencia, no se puede establecer un valor fijo común a todas las comparaciones ya que posiblemente sean muchos más los casos en los que el valor preestablecido no se adapte a las dimensiones de las moléculas, ya sea por grande o pequeño. Debido a esto, se ha considerado un espacio de movimiento variable y calculado de forma específica para cada configuración. Con ello se consigue el objetivo de reducir el espacio de búsqueda y por otro lado, se eliminan zonas del espacio físico que obtendrían malas soluciones por estar demasiado lejos

del centro.

El espacio que puede desplazarse una molécula de otra se encuentra definido de la siguiente forma. Para facilitar la visualización, las figuras se han representado en dos dimensiones. Superpuestas dos moléculas (Figura 5.9) representadas mediante puntos azules y rojos, se obtienen las coordenadas que contienen, en forma de caja, a cada una de ellas (Figura 5.10).

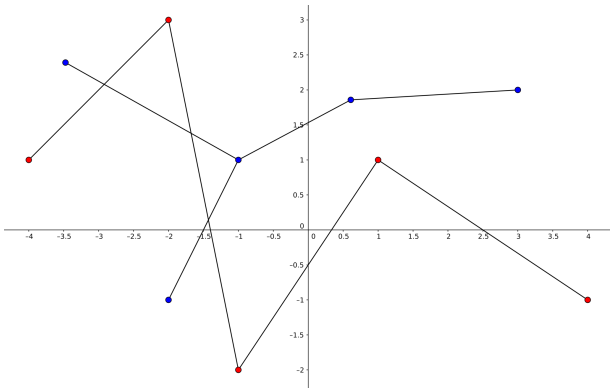


Figura 5.9: Dos moléculas superpuestas.

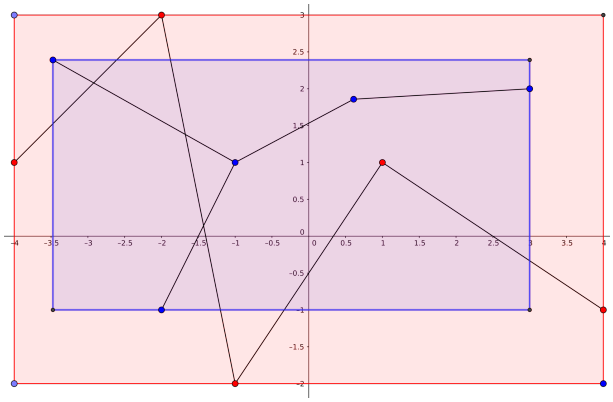


Figura 5.10: Se calculan las cajas que contienen a cada molécula.

A continuación, se restan las dimensiones de ambas cajas en los tres ejes (Figura 5.11).

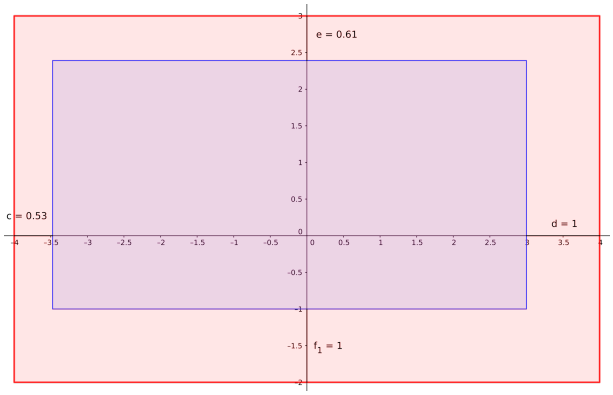


Figura 5.11: Se obtiene la diferencia entre las dimensiones de cada caja.

Los valores para este caso son 0.61 en el eje Y superior y 1 en el inferior; 0.53 en la parte negativa de X y 1 en la positiva. Por último, dado que se tienen dos diferencias por cada eje (la positiva y la negativa), se obtiene el valor absoluto. Este valor será el máximo desplazamiento de la molécula en el eje en el que se ha calculado. El valor absoluto máximo es 1 en cada eje, por lo que el movimiento que puede realizar la molécula azul sobre la roja consiste en un desplazamiento de una unidad en ambos ejes del espacio (Figura 5.12). Este movimiento se encuentra representado mediante la caja verde.

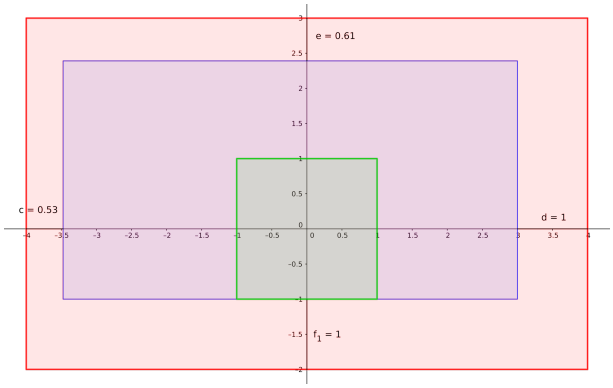


Figura 5.12: Se selecciona el mayor valor absoluto de cada eje.

5.3.4 Puntos para construir un eje de rotación

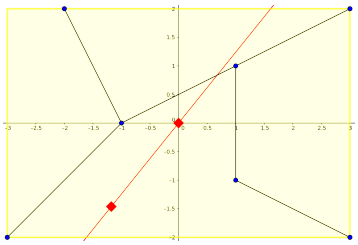
Optipharm necesita rotar la molécula variable para asemejarla lo más posible a la fija. Para ello utiliza un parámetro que es el ángulo, cuyo valor está comprendido en el intervalo [0,360), y un eje sobre el que realizar el giro. Este eje se construye a partir de dos puntos. Establecer las coordenadas de éstos, así como restringirlos a unos valores límites no es sencillo. Aquí se presenta el mismo problema que en la limitación del movimiento, sin embargo, dado que solo afecta a una molécula, no es necesario considerar ambas para obtener el espacio.

El procedimiento es similar al caso anterior, consiste en definir la caja que engloba la molécula variable. Las dimensiones que se obtengan se trasladarán a los límites superiores e inferiores de los respectivos parámetros.

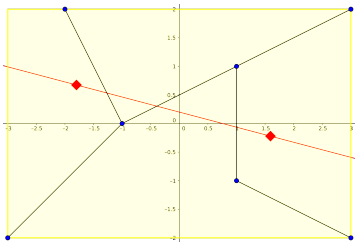
No obstante, aunque este proceso es sencillo de calcular, existe una problemática intrínseca. Para formar un eje son necesarios dos puntos, no siendo posible que coincidan en la misma posición ya que, de darse el caso, no se podría definir un eje. Asociado al concepto del eje, su creación se puede producir en base a distintas restricciones:

- Considerar un punto en el centro de masas de la molécula y permitir que el otro se genere en cualquier lugar del espacio, siempre que esté dentro de la caja precalculada.
- Ambos puntos se generan de forma aleatoria dentro del espacio de la caja precalculada.
- Uno de los puntos se sitúa en uno de los átomos de la molécula y el otro es generado aleatoriamente en el espacio de búsqueda definido por la caja precalculada.

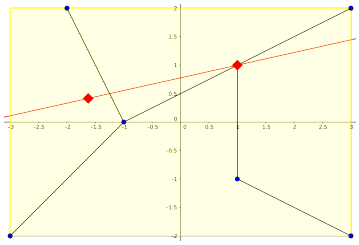
En la Figura 5.13 se pueden visualizar las tres técnicas comentadas anteriormente para la creación de ejes. En Optipharm se han implementado las dos primeras restricciones. Dependiendo de la casuística del problema a resolver se puede utilizar una u otra configuración.



(a) Un punto es aleatorio y el otro se sitúa en el centro de masas de la molécula.



(b) Los dos puntos son generados aleatoriamente dentro de la caja.



(c) Un punto es aleatorio y el otro se sitúa en la posición de un átomo.

Figura 5.13: Técnicas de generación de puntos para el eje de rotación.

5.3.5 Reconfiguración de límites

Optipharm se ha declarado como un gestor de especies, es decir, se encarga de la generación, selección y optimización de las mismas. En especial, en el proceso de generación y optimización se ha visto que estas operaciones se llevan a cabo dentro de los límites de cada especie, que se define por el radio y que es distinto para cada especie dependiendo del nivel en el que se cree.

El radio mínimo para el último nivel es proporcionado por el usuario, sin embargo, el radio de las especies de nivel 1 se calcula a partir de los límites superiores e inferiores de cada parámetro. Para este tipo de problemas, no se puede considerar unos límites constantes para todo par de moléculas a ser comparadas ya que los tamaños pueden ser muy dispares (Figura 5.14) implicando, por ejemplo, desplazar 3 unidades hacia la izquierda puede ser suficiente para ocupar todo el cuerpo de la molécula pequeña pero insuficiente para la grande.

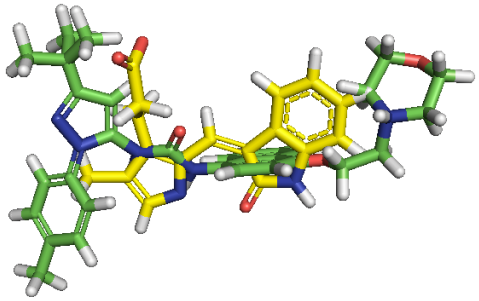
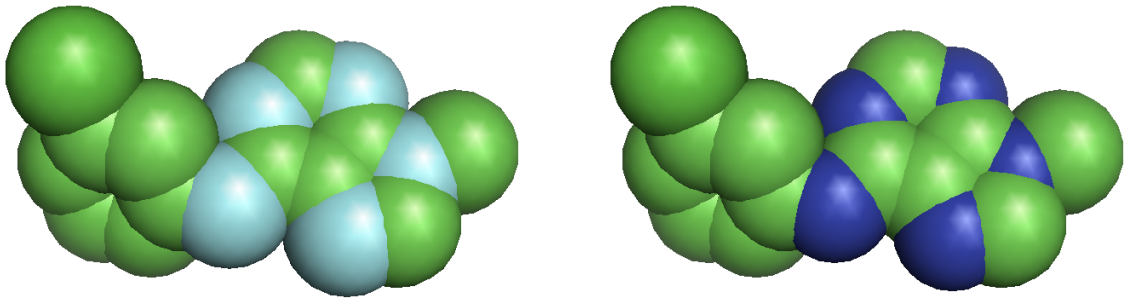


Figura 5.14: Moléculas con distintos tamaños.

5.3.6 Radios de Van del Waals

Como se ha visto en la expresión matemática utilizada para el cálculo de la similitud de forma, el radio considerado para cada elemento es uno de los parámetros que influye en su valor final. Si bien, en el bibliografía se encuentran experimentos en los que el radio para todos los átomos es el mismo, puede resultar interesante y más realista asignar un radio distinto a cada elemento. En ese sentido, en Optipharm se han considerado las dos opciones: para experimentos con mismo radio para todos los elementos, el valor elegido ha sido 1.8 mientras que para experimentos con radios distintos, los valores utilizados pueden ser consultados en [10].

Visualmente, en la Figura 5.15 se puede observar la proteína cdk2 sin hidrógenos con las dos configuraciones.



(a) Proteína cdk2 con mismo radio. (b) Proteína cdk2 con distinto radio.

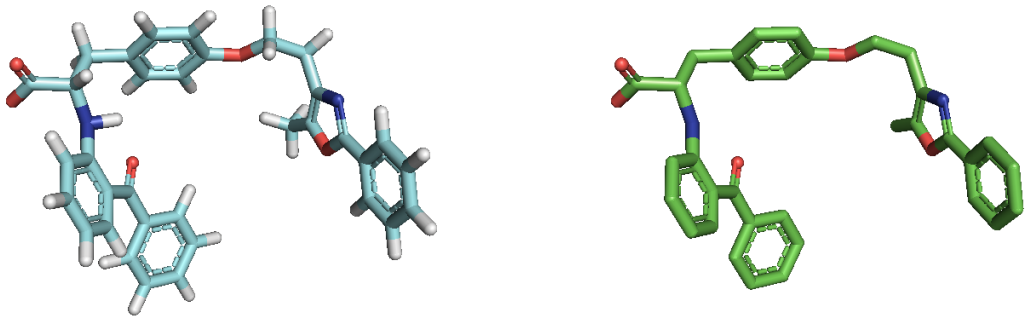
Figura 5.15: Proteína cdk2, distinta configuración de radios de van der Waals.

En lo referente a los valores empíricos, el volumen de la Figura 5.15a es 213.095 mientras que en la Figura 5.15b, el valor es 186.679. En este ejemplo con pocos átomos queda reflejada la importancia de los radios en el volumen y como puede afectar su valor de forma significativa.

5.3.7 Hidrógenos

El hidrógeno es el primer elemento de la tabla periódica con todo lo que ello conlleva: masa molecular mínima, mínimo número de protones y electrones, etc. En las técnicas de cribado virtual donde se tienen en cuenta las propiedades físicas de las moléculas, suelen eliminarse los hidrógenos. Las principales razones por las que se toma esta decisión es que el volumen que aportan no es significativo; y segundo y más importante, es muy complicado determinar la posición de un hidrógeno respecto al otro átomo con el que está enlazado. En ese sentido, antes de aportar una información errónea se toma la decisión de eliminarlos en las comparaciones.

En la Figura 5.16 se muestra el receptor peroxisoma-proliferador-activado gamma (ppar_gamma) con (Figura 5.16a) y sin hidrógenos (Figura 5.16b).



(a) Receptor ppar_gamma con hidrógenos. (b) Receptor ppar_gamma sin hidrógenos.

Figura 5.16: Receptor ppar_gamma con y sin hidrógenos.

Empíricamente, el volumen que se pierde al eliminar los hidrógenos es 222.936. Resulta de restar los volúmenes del receptor con hidrógenos (718.958) y sin hidrógenos (496.022).

La decisión de considerar los hidrógenos o no, para la experimentación se basa, al igual que en los radios, en la uniformidad de la decisión. Siempre que se tome la misma decisión para todo el conjunto de experimentos, los resultados serán similares. Lo que no está permitido es realizar ejecuciones con configuraciones dispares. Esto produciría un falta de credibilidad en los resultados y cualquier decisión en base a ellos posiblemente sería errónea.

5.3.8 Gestión de archivos

Las sustancias, compuestos y/o moléculas con los que trabajan algoritmos como el desarrollado en este trabajo se encuentran representados en muy diversos formatos. Algunos de ellos son mol, mol2, sd, pdb, xyz y gpr. Excluyendo la opción ideal donde todos los formatos fuesen aceptados por el algoritmo, se ha decidido usar dos de ellos. La primera opción y sobre la que se implementan todas las mejoras es el formato mol2. Este formato es considerado por muchos un estándar debido a la buena documentación que se dispone de él y dada su modularidad. Permite almacenar solo la información que se necesite evitando de este modo trabajar con archivos de gran tamaño. Un ejemplo de este formato se puede ver en el Listado 5.1.

En este ejemplo se pueden diferenciar tres partes gracias a las etiquetas *MOLECULE*, *ATOM* y *BOND*. Cada apartado proporciona una información. *MOLECULE* da los detalles generales de la molécula: nombre de la molécula según la base de datos de referencia, número de átomos, número de enlaces, tamaño, etc. *ATOM* da la información de cada átomo: nombre, coordenadas XYZ, etc. Y por último *BOND* índica que enlaces existen y qué átomos afecta. Existen más etiquetas que se pueden consultar en [89].

Además de todas sus ventajas teóricas, es el formato representado para la mayoría de bases de datos disponibles de moléculas, entre ellas dos que se usan en la experimentación de este trabajo: FDA[1] y DUD [41].

[1] `https://www.fda.gov/`

```
@<TRIPOS>MOLECULE
DB01154
   35    35     0     0     0
SMALL
USER_CHARGES

@<TRIPOS>ATOM
      1 S1       4.0110    3.0907    0.5933 S.2     1 <0>     -0.3800
      2 O1       0.3914    0.3180    2.3367 O.2     1 <0>     -0.5700
    ...
     34 H17      4.0323   -3.0526    0.0368 H       1 <0>      0.1500
     35 H18      2.7808   -2.9310   -1.3303 H       1 <0>      0.1500
@<TRIPOS>BOND
     1    1   15 2
     2    2   11 2
     3    3   12 2
     ...
    34   17   34 1
    35   17   35 1
```

Listado 5.1: Formato Mol2.

Sobre el formato mol2 se ha realizado un exhaustivo trabajo considerando las diferentes partes de las que se compone el documento e incluyendo esa información dentro de Optipharm. Y lo que es más importante, estas bases de datos se encuentran generadas por distintos programas externos en distintos sistemas operativos lo que implica por ejemplo a tener en cuenta la lectura de los saltos de línea, que son distintos para sistemas operativos Microsoft y Unix.

Por otro lado, el segundo formato que se ha considerado es el Formato SDF. El motivo por el que incluir este formato reside en su uso por parte del algoritmo WEGA [98], muy conocido en la literatura y que se utiliza para comparar los resultados de Optipharm. Este formato es muy utilizado cuando solo necesita información estructural de las moléculas, de ahí su nombre: Structure-Data File. En el Listado 5.2 se muestra la misma molécula que el Listado 5.1.

```
DB01154
 OpenBabel12221612453D

 35 35  0  0  1  0  0  0  0  0999 V2000
    4.0110    3.0907    0.5933 S   0  0  0  0  0  0  0  0  0  0  0  0
    0.3914    0.3180    2.3367 O   0  0  0  0  0  0  0  0  0  0  0  0
   ...
    4.0323   -3.0526    0.0368 H   0  0  0  0  0  0  0  0  0  0  0  0
    2.7808   -2.9310   -1.3303 H   0  0  0  0  0  0  0  0  0  0  0  0
  1 15  2  0  0  0  0
  2 11  2  0  0  0  0
  3 12  2  0  0  0  0
  ...
 17 34  1  0  0  0  0
 17 35  1  0  0  0  0
M  END
$$$$
```

Listado 5.2: Formato SDF.

5.3.9 Evaluación de una solución candidata

Un concepto importante en Optipharm es la conexión entre la solución de una especie (su centro) y el valor de la función objetivo asociado a ella. A continuación se mostrará cada uno de los pasos a seguir para una evaluación de un par de moléculas.

Para el ejemplo se van a utilizar las dos moléculas que se muestran en la Figura 5.17. La molécula verde mantendrá su posición fija mientras que la azul será la que se modifique en base a una especie de Optipharm. El centro de la especie para este caso de estudio es [3, -3, 8.7, -1.5, 0, 0, 0, 1, 1.75, -1]. El valor de la función objetivo en la posición inicial es 0.18.

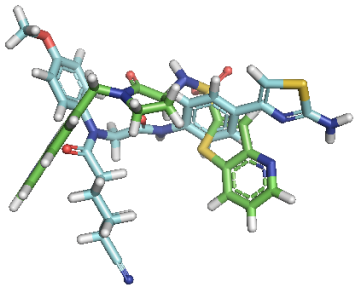


Figura 5.17: Posición inicial de las moléculas.

La primera modificación que se realiza sobre la molécula es la rotación. Existen principalmente dos métodos para realizar rotaciones, los ángulso de Euler y los cuaterniones. Se ha decidido utilizar el segundo método dada la eficiencia de las operaciones, aunque conceptualmente es más difícil de entender.

La rotación se aplica átomo a átomo dado un eje y un ángulo. Para definir estos dos elementos se utilizan los siete primeros parámetros de la solución de la especie. El primero se corresponde con el ángulo y los seis siguientes con las coordenadas X, Y y Z de los dos puntos que definirán el eje. En este momento, la situación de las moléculas se puede ver en la Figura 5.18. Se ha definido el eje, pero todavía no se ha aplicado la rotación.

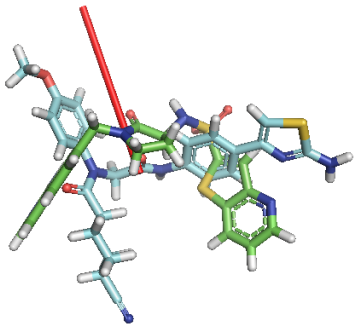


Figura 5.18: Definición del eje de rotación.

Una vez definido el eje, se aplica la rotación. En este caso el valor es 3 radianes. Como 180^o es equivalente a π radianes, la rotación que se realizará sobre el eje será aproximadamente de 171^o (Figura 5.19). El valor actual de la función objetivo es 0.35 (este valor se ha calculado por tratarse de un ejemplo, Optipharm no lo calcula). Se puede apreciar en rosa como se encuentra rotada 171^o la molécula respecto al eje en comparación con su posición original (azul).

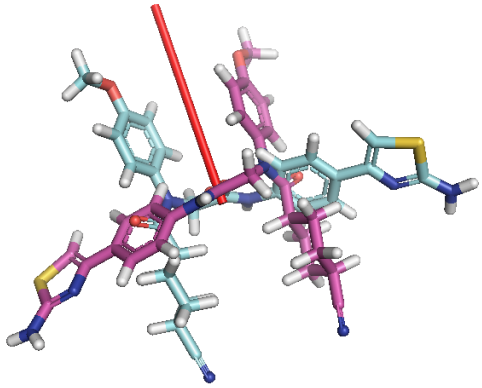


Figura 5.19: Rotación de la molécula.

El último paso que se realiza es la traslación. Para ellos utilizan los últimos 3 parámetros de la solución de la especie. De la misma forma que en el proceso de rotación, la traslación se realiza átomo a átomo. En este ejemplo, la traslación consiste en desplazar 1 unidad en el eje X, 1.75 en el Y y -1 en el Z. La distancia desplazada se puede ver en la Figura 5.20, donde la molécula rosa se corresponde con la molécula únicamente rotada, y la amarilla, con la rotada y trasladada.

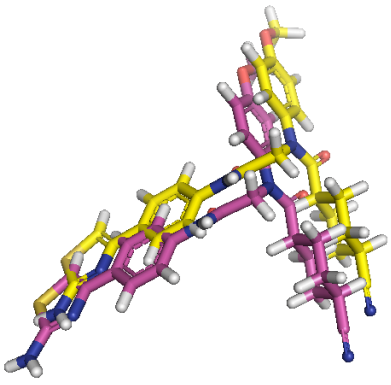
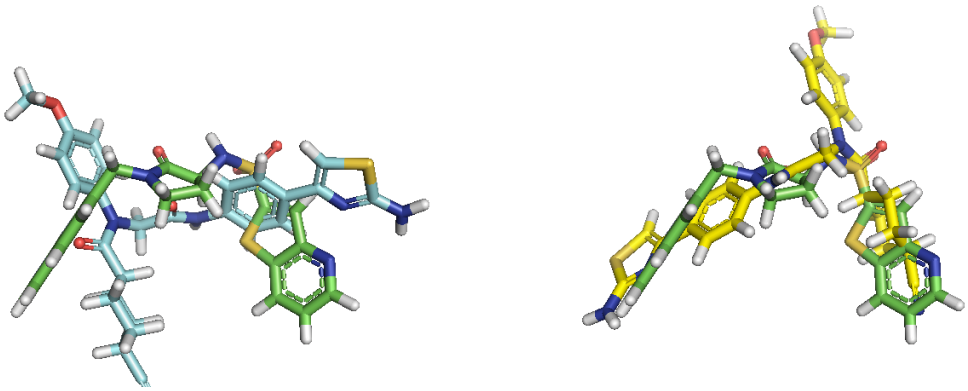


Figura 5.20: Traslación de la molécula.

Por último, en la Figura 5.21 se muestra la posición inicial o punto de partida de ambas moléculas, donde el valor de la función objetivo era 0.18 (Figura 5.21a), y la posición final tras las modificaciones donde el valor de la función objetivo es de 0.53 (Figura 5.21b).



(a) Posición original del par de moléculas.

(b) Posición final del par de moléculas.

Figura 5.21: Posición inicial y final tras el proceso de rotación y traslación de una molécula.

6. Resultados

Este capítulo recoge los experimentos que se han realizado con Optipharm. Se ha utilizado WEGA como software de referencia para comparar los resultados obtenidos por Optipharm. Las métricas seleccionadas para la comparativa han sido dos: el valor de la función objetivo, en este caso, la similitud de forma y el valor de AUC. Los resultados obtenidos se mostraran en las siguientes secciones.

6.1 Bases de datos

Para la evaluación de las funciones objetivo y por tanto, el comportamiento de los algoritmos, es necesario tener una fuente de moléculas (proteínas, compuestos, ligandos, etc.) que evaluar. En ese sentido, los expertos en el área de la bioinformática han dedicado parte de sus esfuerzos a crear y almacenar distintas fuentes de compuestos. Algunas de ellas solo tienen como fin almacenar el conocimiento o agrupar ciertos compuestos interesantes. Otras por el contrario se han creado con el objetivo de evaluar la calidad de un algoritmo, una función objetivo, etc. Algunas de ellas son DrugBank[1], Mcule[2], ChemSpider[3], PubMed[4], ChEBI[5] y Pubchem[6]. Para este trabajo se han seleccionado dos: la base de datos de US Food and Drug Administration (FDA)[7] y el DUD (Directory of Useful Decoys) [41].

[1] https://www.drugbank.ca/
[2] https://mcule.com/
[3] http://www.chemspider.com
[4] https://www.ncbi.nlm.nih.gov/pubmed/
[5] http://www.ebi.ac.uk/chebi/
[6] https://pubchem.ncbi.nlm.nih.gov/
[7] https://www.fda.gov/

6.2 Función objetivo

Para realizar la comparativa de la función objetivo, el análisis se va a realizar con la base de datos FDA.

6.2.1 FDA Database

La FDA (Food and Drug Administration, Administración de Medicamentos y Alimentos) es la agencia del gobierno de los Estados Unidos responsable de la regulación de alimentos (tanto para personas como para animales), medicamentos (humanos y veterinarios), cosméticos, aparatos médicos (humanos y animales), productos biológicos y derivados sanguíneos. DrugBank incluye 1 751 moléculas pequeñas que se corresponden con medicamentos ya aprobados en Estados Unidos para su uso en humanos, y cuya búsqueda de moléculas similares a éstas es muy frecuente entre muchos investigadores.

Para realizar la experimentación, en primer lugar se ha realizado un estudio del tamaño de las moléculas. La distribución de los tamaños de las moléculas se puede ver en la Figura 6.1. Como se puede apreciar, la mayoría de las moléculas tienen un tamaño de entre 20 y 70 átomos. En consecuencia, se ha elegido el doble de moléculas fijas de este intervalo que del resto. No sé puede descartar el resto debido a que puede ser interesante conocer como se comporta el algoritmo cuando tiene que comparar dos moléculas con tamaños muy dispares.

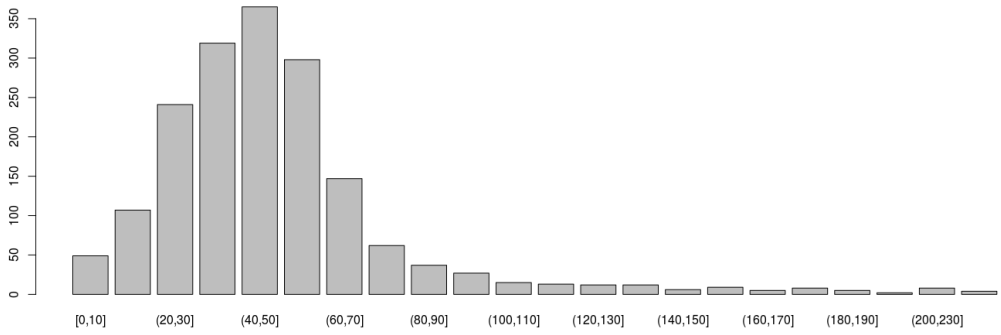


Figura 6.1: Distribución de los tamaños de los átomos.

En la Tabla 6.1 se muestran el identificador de las moléculas seleccionadas así como el número de átomos que cada una de ellas tiene. Cada una de estas moléculas se comparará con las 1 751 de la base de datos, o dicho de otra manera, se compara con los 1 750 compuestos restantes y con él mismo. Además, antes de realizar la evaluación de la base de datos con cualquiera de los algoritmos, se ha centrado cada molécula en el punto (0, 0, 0) del eje de coordenadas respecto al centro de masas de cada una utilizando la técnica del análisis de componentes principales. De esta forma, el eje más largo de la molécula quedaba paralelo al eje X, el siguiente eje más largo, paralelo al eje Y y por último, el eje más corto, paralelo al eje Z.

Id. (FDA)	nº Átomos	Id. (FDA)	nº Átomos
1040	10	869	80
880	20	1297	91
178	30	83	100
931	35	857	110
524	40	341	120
48	45	706	130
377	50	1671	140
466	55	267	151
46	60	1301	169
215	70	1085	194

Tabla 6.1: Selección de moléculas fijas para realizar las comparaciones.

6.2.2 Resultados

La configuración de ambos algoritmos ha sido idéntica. Se han eliminado los hidrógenos, asignado el mismo radio de van der Waals a todos los elementos y permitido valores aleatorios a ambos puntos del eje. En cuanto a los parámetros de entrada propios de la naturaleza evolutiva de Optipharm, el número de evaluaciones ha sido 200 000, 5 han sido los niveles, 5 las especies y al radio mínimo se le ha asignado el valor 1.

Los resultados obtenidos se muestran en la Tabla 6.2. En esta tabla se diferencian tres partes, la primera columna muestra los compuestos elegidos, cuyo valor se encuentra relacionado con la Tabla 6.1. Los otros dos grandes bloques se corresponden con los resultados de Optipharm y de WEGA. En cada uno de ellos se puede ver el valor medio de la función objetivo obtenido tras comparar todos los compuestos con el de referencia, el valor de la función objetivo del compuesto con menor valor y la última columna hace referencia al tiempo. Se ha obviado el valor máximo encontrado para cada compuesto ya que en todos los experimentos ha sido 1 (se encontraba a sí misma). La última fila muestra el valor medio de cada columna.

Analizando los resultados en base al valor obtenido de la función objetivo, Optipharm encuentra la misma calidad de la solución que WEGA.

6.3 Área bajo la Curva ROC (ROC AUC)

Una de las principales aplicaciones del cribado virtual es filtrar, entre millones de compuestos, aquellos que, dada una molécula de consulta, sean los más favorables o susceptibles de poder utilizarse como molécula alternativa. Y es que uno de los objetivos del algoritmo es clasificar correctamente los compuestos favorables como tal y lo propio con los no favorables. En el ámbito estadístico esto se conoce como tabla de contingencia, cuyo término apareció por primera vez en [70]. En esta sección se ha resumido el concepto de curva ROC y el cálculo de su área, para una comprensión mayor véase [25].

	Optipharm			WEGA		
Id. (FDA)	**F. Obj. (Media)**	**F. Obj. Mín.**	**T(min)**	**F. Obj. (Media)**	**F. Obj. Mín.**	**T(min)**
1040	0.337	0.016	14.348	0.336	0.016	0.413
880	0.471	0.025	25.964	0.472	0.025	0.399
178	0.507	0.031	37.485	0.509	0.032	0.403
931	0.550	0.046	65.230	0.554	0.046	0.426
524	0.512	0.055	78.159	0.517	0.055	0.419
48	0.523	0.056	88.096	0.527	0.058	0.432
377	0.481	0.063	98.810	0.486	0.061	0.459
466	0.509	0.064	108.018	0.513	0.065	0.447
46	0.490	0.065	97.899	0.495	0.068	0.443
215	0.384	0.078	131.569	0.388	0.077	0.542
869	0.423	0.080	134.826	0.426	0.076	0.526
1297	0.466	0.082	116.559	0.469	0.079	0.481
83	0.357	0.067	149.782	0.358	0.066	0.542
857	0.379	0.065	148.478	0.380	0.060	0.547
341	0.210	0.039	240.313	0.195	0.037	0.808
706	0.280	0.054	148.017	0.281	0.059	0.565
1671	0.286	0.053	206.407	0.282	0.053	0.655
267	0.272	0.047	168.714	0.270	0.048	0.690
1301	0.221	0.036	223.947	0.215	0.038	0.788
1085	0.197	0.032	226.139	0.197	0.031	0.790
Media	0.393	0.053	125.438	0.393	0.053	0.539

Tabla 6.2: Comparación de la función objetivo de Optipharm y WEGA con la base de datos FDA.

6.3.1 Cálculo del área bajo la curva ROC

Un gráfico ROC (de sus siglas en inglés, Receiver Operating Characteristics, Característico Operativo del Receptor) es una técnica de visualización, organización y clasificación basada en los valores obtenidos de un experimento. Su uso comenzó con la teoría de detección de señales intentando diferenciar entre las que eran reales y las falsas [20, 86]. A partir de entonces se extendió su aplicación a otras disciplinas. De este modo, comenzó a usarse en las pruebas de diagnóstico para la toma de decisiones médicas [100]. Sin embargo, no fue hasta [85] cuando se comenzó a usar las curvas ROC para el aprendizaje automático y la evaluación y comparación de algoritmos. Desde entonces, son muchos los casos en los que se utiliza esta métrica para comparar la calidad de los algoritmos.

El clasificador

La base en la que se sostiene la curva ROC es que se parte de un conjunto de elementos favorables, o similares según ciertas propiedades, y de elementos no favorables. A priori, se desconoce su valor y se requiere de un clasificador que prediga si cada uno de los elementos se considera favorable o verdadero, o bien no favorable o falso. En base a esta predicción generamos una tabla de contingencia o matriz de confusión (Figura 6.2).

Como se puede observar, se han generado 4 categorías:

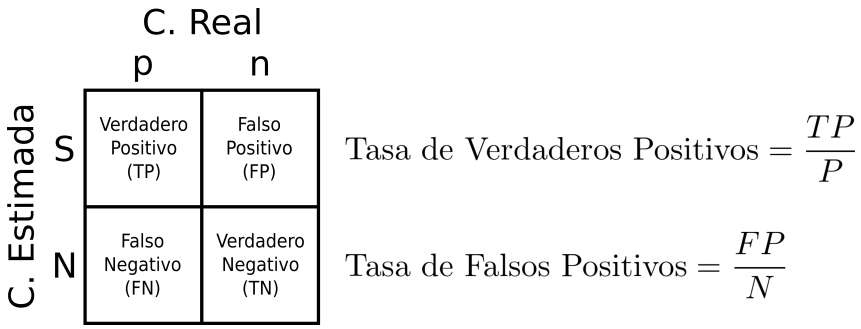


Figura 6.2: Matriz de confusión.

- *Verdadero positivo*: elemento que es verdadero y se ha clasificado como verdadero.
- *Verdadero negativo*: elemento falso que se ha clasificado como falso.
- *Falso positivo*: elemento falso clasificado como verdadero.
- *Falso negativo*: elemento verdadero clasificado como falso.

A partir de la combinación de estas cuatro categorías, se generan el resto de expresiones matemáticas (valores) permitiendo obtener distinta información según lo que se desee analizar.

Como se puede comprobar, los 4 valores principales son complementarios dos a dos conociéndose el número total de elementos evaluados, por lo que se suelen utilizar dos de ellos para realizar las comparaciones. Las categorías seleccionadas para crear la curva ROC son los verdaderos positivos y los falsos positivos (Figura 6.3). Las dos dimensiones del gráfico en el que la curva es representada toman valores en el intervalo [0,1]. En este cuadro se representa el ratio de verdaderos positivos y falsos positivos.

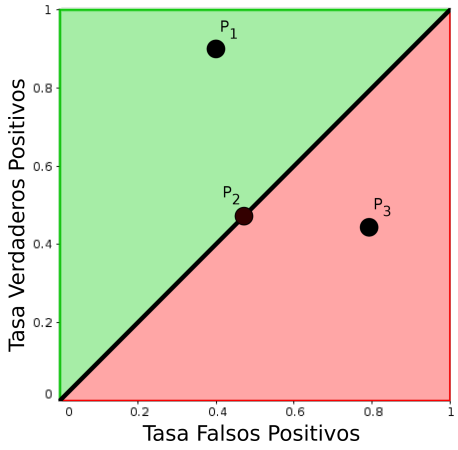


Figura 6.3: Un grafo ROC mostrando 3 valores discretos.

Adicionalmente se añade una diagonal desde el punto 0,0 al punto 1,1. Esta línea indica que todo resultado que se obtenga y se sitúe en algún punto sobre ella significará que el algoritmo evaluado clasificará de forma azarosa. Un valor en el triangulo derecho inferior (zona roja) implica un mal clasificador y por tanto sería mejor clasificar al azar; y un valor en la matriz de confusión en el triángulo izquierdo superior (zona verde) implica que se tiene un buen clasificador.

Ahora bien, tal y como está planteado ahora mismo el clasificador, solo nos devuelve una clase

con un resultado (lo que se corresponde con un punto en el gráfico). Existen otros clasificadores que generan varios puntos o una probabilidad. En el problema que se presenta, el clasificador obtiene en número de verdaderos positivos que hay en las primeras posiciones barriendo los resultados desde su inicio al fin. Es por ello que se necesitan de distintos puntos y analizar como se comportan.

Para la generación de la curva ROC se utiliza el Algoritmo 4. Un ejemplo de su ejecución se puede ver en la Figura 6.4 [25]. En la Figura 6.4a se muestran los datos de entrada utilizados. Éstos consisten en 20 valores equitativamente divididos en positivos y negativos conocidos ordenados por el valor de la función objetivo. A su derecha (Figura 6.4b) se muestra la curva ROC generada por la ejecución del algoritmo con esos elementos. La generación ha consistido en lo siguiente: se evalúa cada elemento de la lista ordenada, si la clase es positiva se incrementa la tasa de verdaderos positivos; si es negativa, se incrementa la tasa de falsos positivos. La clasificación perfecta consistiría en que los valores positivos ocupasen las 10 primeras posiciones.

Algoritmo 4 Método eficiente de generación de puntos ROC

Input: L, el conjunto de muestras; $f(i)$, la estimación del clasificador probabilístico de que la muestra i sea positiva; P y N, número de muestras positivas y negativas.

Output: R, una lista de puntos ROC que incrementan por tasa de falsos positivos.

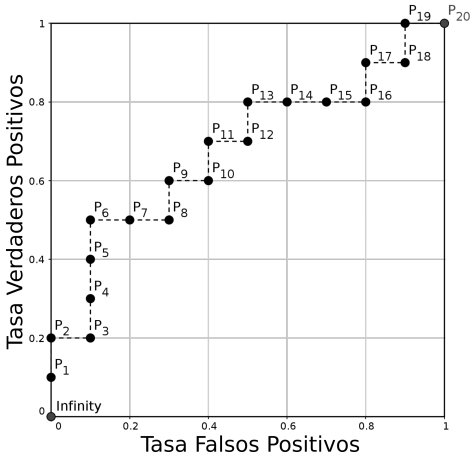
Required: $P > 0$ y $N > 0$.

1: $L_{sorted} = L$ ordenados por valores de f decrecientes
2: $FP = TP = 0$
3: $R = ()$
4: $f_{prev} = -$ínf
5: $i = 1$
6: **while** $i \leq |L_{sorted}|$ **do**
7: **if** $f(i) \neq f_{prev}$ **then**
8: insertar($\frac{FP}{N}, \frac{TP}{P}$) en R
9: $f_{prev} = f(i)$
10: **if** $L_{sorted}[i]$ es una muestra positiva **then**
11: $TP = TP + 1$
12: **else** ▷ Es una muestra negativa
13: $FP = FP + 1$
14: $i = i + 1$
15: insertar($\frac{FP}{N}, \frac{TP}{P}$) en R ▷ Esto es (1,1)

Una vez se ha generado la curva, visualmente se pueden comparar los resultados, pero para realizar un análisis con gran cantidad de experimentos, es recomendable obtener un valor único. Este valor consiste en calcular el área que existe debajo de la curva obtenida. Si el valor es cercano a 1, significará que se diferencian muy bien los verdaderos positivos de los verdaderos negativos. Si el valor es cercano a 0.5 (sobre la diagonal [(0,0), (1,1)]), se han clasificado al azar los elementos; y un valor cercano a 0 significa que la calidad del clasificador es pésima. El Algoritmo 5 calcula este valor.

ID	Clase	Valor	ID	Clase	Valor
P_1	p	.9	P_{11}	p	.4
P_2	p	.8	P_{12}	n	.39
P_3	n	.7	P_{13}	p	.38
P_4	p	.6	P_{14}	n	.37
P_5	p	.55	P_{15}	n	.36
P_6	p	.54	P_{16}	n	.35
P_7	n	.53	P_{17}	p	.34
P_8	n	.52	P_{18}	n	.33
P_9	p	.51	P_{19}	p	.30
P_{10}	n	.505	P_{20}	n	.1

(a) Elementos de entrada.



(b) Curva ROC generada.

Figura 6.4: Un ejemplo de generación de una curva ROC.

Algoritmo 5 Cálculo del área bajo la curva ROC

Input: L, el conjunto de muestras; $f(i)$, la estimación del clasificador probabilístico de que la muestra i sea positiva; P y N, número de muestras positivas y negativas.

Output: A, el área bajo la curva ROC.

Required: $P > 0$ y $N > 0$.

$L_{sorted} = L$ ordenados por valores de f decrecientes
$FP = TP = 0$
$A = 0$
$f_{prev} = -$ínf
$i = 1$
while $i \leq |L_{sorted}|$ **do**
    if $f(i) \neq f_{prev}$ **then**
        $A = A + TRAPEZOID\_AREA(FP, FP_{prev}, TP, TP_{prev})$
        $f_{prev} = f(i)$
        $FP_{prev} = FP$
        $TP_{prev} = TP$
    if i es una muestra positiva **then**
        $TP = TP + 1$
    else ▷ Es una muestra negativa
        $FP = FP + 1$
    $i = i + 1$
$A = A + TRAPEZOID\_AREA(FP, FP_{prev}, TP, TP_{prev})$
$A = A/(P \times N)$ ▷ Se escala a la unidad cuadrada
procedure TRAPEZOID_AREA($X1$, $X2$, $X3$, $X4$)
    $Base = |X1 - X2|$
    $Height_{avg} = (Y1 + Y2)/2$
    return $Base \times Height_{avg}$

6.3.2 DUD: Directory of Useful Decoys (Directorio de señuelos útiles)

DUD es una base de datos de moléculas creada para comprobar la calidad clasificatoria de los algoritmos. Tiene 3 961 ligandos distribuidos entre 40 objetivos farmacológicos con los que se comparan. Además, a partir de cada ligando, se han generado 36 señuelos. En este contexto, un señuelo se corresponde con moléculas que tienen propiedades físicas similares al ligando del que se creó, pero estructura química distinta y por tanto el clasificador lo debería de detectar como falso obteniéndose un falso positivo. En consecuencia, se tiene un total de 3 961 ligandos y 124 413 señuelos. Antes de realizar la evaluación de la base de datos con cualquiera de los algoritmos, se ha centrado cada molécula en el punto (0, 0, 0) del eje de coordenadas respecto al centro de masas de cada una utilizando la técnica del análisis de componentes principales. De esta forma, el eje más largo de la molécula quedaba paralelo al eje X, el siguiente eje más largo, paralelo al eje Y y por último, el eje más corto, paralelo al eje Z.

El interés de esta base de datos para este trabajo reside en que se conocen que compuestos se deberían clasificar como verdaderos positivos (ligandos) y cuales como verdaderos negativos (señuelos) dada un compuesto de referencia. De este modo, una vez se ha obtenido el valor de la función objetivo de cada comparación, se aplica a cada una de las 40 proteínas el método de la curva AUC, obteniendo como resultado lo bien o mal que se han clasificado los ligandos y señuelos.

6.3.3 Resultados

Para este experimento, al igual que en el anterior, la configuración de ambos algoritmos ha sido idéntica. Se han eliminado los hidrógenos, asignado el mismo radio de van der Waals a todos los elementos y permitido valores aleatorios a ambos puntos del eje. En cuanto a los parámetros de entrada de Optipharm, el número de evaluaciones ha sido 200 000, 5 han sido los niveles, 5 las especies y al radio mínimo se le ha asignado el valor 1. Los resultados obtenidos se puede ver en la Tabla 6.3.

Sobre estos resultados se puede comprobar que la calidad clasificatoria de ambos algoritmos es prácticamente la misma. Hay que tener en cuenta el detalle de que estos resultados se han obtenido analizando 128 374 comparaciones.

Un caso particular es la proteína FXA (y sus respectivos ligandos y señuelos). En las otras 39 proteínas los resultados son similares, obteniendo en algunos mejor valor WEGA y otros Optipharm, pero siempre con valores cercanos. Sin embargo, el valor de AUC de WEGA en FXA es 0.1 superior al valor de Optipharm en la misma proteína. Cabe destacar que incluso incluyendo ese resultado en las medias, el valor de WEGA solo es 0.08 unidades superior a Optipharm, siendo esta diferencia inapreciable.

Estos resultados han sido obtenidos por Optipharm con la misma configuración de parámetros de entrada (Número de evaluaciones (N), especies (M), niveles (L) y radio mínimo (R_L). Sin embargo, Optipharm, a diferencia de WEGA, permite modificar estos parámetros para obtener mayor o menor precisión en la solución dependiendo de la naturaleza y el relieve que presente la función objetivo y por tanto el espacio de búsqueda. En ese sentido, se elaboró un test con distintas configuraciones de Optipharm. Los parámetros de entrada considerados se puede ver en

nombre	WEGA	Optipharm	nombre	WEGA	Optipharm
ace	0.333	0.356	hivrt	0.752	0.738
ache	0.719	0.718	hmga	0.773	0.735
ada	0.663	0.667	hsp90	0.665	0.709
alr2	0.216	0.235	inha	0.6	0.531
ampc	0.709	0.698	mr	0.838	0.842
ar	0.72	0.731	na	0.854	0.834
cdk2	0.59	0.569	p38	0.47	0.504
comt	0.373	0.426	parp	0.491	0.478
cox1	0.484	0.494	pde5	0.745	0.726
cox2	0.951	0.945	pdgfrb	0.462	0.442
dhfr	0.653	0.628	pnp	0.628	0.625
egfr	0.571	0.547	ppar_gamma	0.703	0.654
er_agonist	0.789	0.788	pr	0.609	0.611
er_antagonist	0.72	0.739	rxr_alpha	0.907	0.851
fgfr1	0.396	0.43	sahh	0.894	0.854
fxa	0.68	0.584	src	0.302	0.336
gart	0.273	0.312	thrombin	0.548	0.516
gpb	0.842	0.83	tk	0.579	0.528
gr	0.617	0.61	trypsin	0.259	0.289
hivpr	0.756	0.721	vegfr2	0.606	0.594
		WEGA	Optipharm		
	Media	0.619	0.611		

Tabla 6.3: Comparativa de resultados AUC de Optipharm y WEGA.

la Tabla 6.4. El número de configuraciones evaluadas distintas ha sido $6 \cdot 5 \cdot 4 \cdot 2 = 240$.

Tras realizar toda la experimentación, se seleccionó el mejor valor de AUC obtenido para cada proteína. Los resultados muestran que Optipharm es mejor clasificador que WEGA ya que de manera individual, obtiene mejor valor de AUC en 30 de las 40 proteínas, con una media total superior en 0.037. Estos resultados se pueden analizar de forma visual en la Figura 6.5 y de forma numérica, y por tanto exacta, en la Tabla 6.5.

N	M	L	R_L
15 000	15	12	10
12 000	12	10	5
10 000	10	8	
7 500	8	5	
5 000	5		
1 000			

Tabla 6.4: Batería de experimentos.

Nombre	WEGA	Optipharm	Conf. de parámetros			
			N	M	L	R_L
ace	0.333	0.425	1 000	12	10	10
ache	0.719	0.844	5 000	12	5	10
ada	0.663	0.752	1 000	15	10	10
alr2	0.216	0.353	1 000	12	10	5
ampc	0.709	0.725	5 000	8	10	5
ar	0.720	0.734	15 000	15	10	10
cdk2	0.590	0.624	10 000	15	12	10
comt	0.373	0.507	7 500	15	5	5
cox1	0.484	0.519	5 000	15	12	10
cox2	0.951	0.944	10 000	12	8	10
dhfr	0.653	0.649	15 000	8	10	5
egfr	0.571	0.570	15 000	8	10	10
er_agonist	0.789	0.808	15 000	8	8	5
er_antagonist	0.720	0.726	10 000	12	8	10
fgfr1	0.396	0.452	5 000	15	10	10
fxa	0.680	0.598	10 000	10	10	5
gart	0.273	0.494	15 000	10	12	5
gpb	0.842	0.850	10 000	10	10	10
gr	0.617	0.678	12 000	8	12	10
hivpr	0.756	0.754	10 000	12	8	10
hivrt	0.752	0.756	10 000	12	8	10
hmga	0.773	0.763	10 000	12	8	10
hsp90	0.665	0.776	1 000	15	10	5
inha	0.600	0.582	10 000	12	8	10
mr	0.838	0.864	12 000	8	5	10
na	0.854	0.846	10 000	12	8	10
p38	0.470	0.505	10 000	12	8	10
parp	0.491	0.500	12 000	8	10	10
pde5	0.745	0.758	10 000	5	10	5
pdgfrb	0.462	0.475	5 000	12	10	5
pnp	0.628	0.681	7 500	12	12	10
ppar_gamma	0.703	0.713	5 000	8	10	5
pr	0.609	0.662	10 000	5	12	5
rxr_alpha	0.907	0.904	5 000	8	10	10
sahh	0.894	0.901	7 500	8	12	10
src	0.302	0.378	10 000	15	8	10
thrombin	0.548	0.610	7 500	8	8	10
tk	0.579	0.569	5 000	15	10	10
trypsin	0.259	0.359	7 500	15	5	10
vegfr2	0.606	0.617	15 000	15	10	5
Media	0.619	0.656				

Tabla 6.5: Mejor AUC de Optipharm combinando distintas configuraciones.

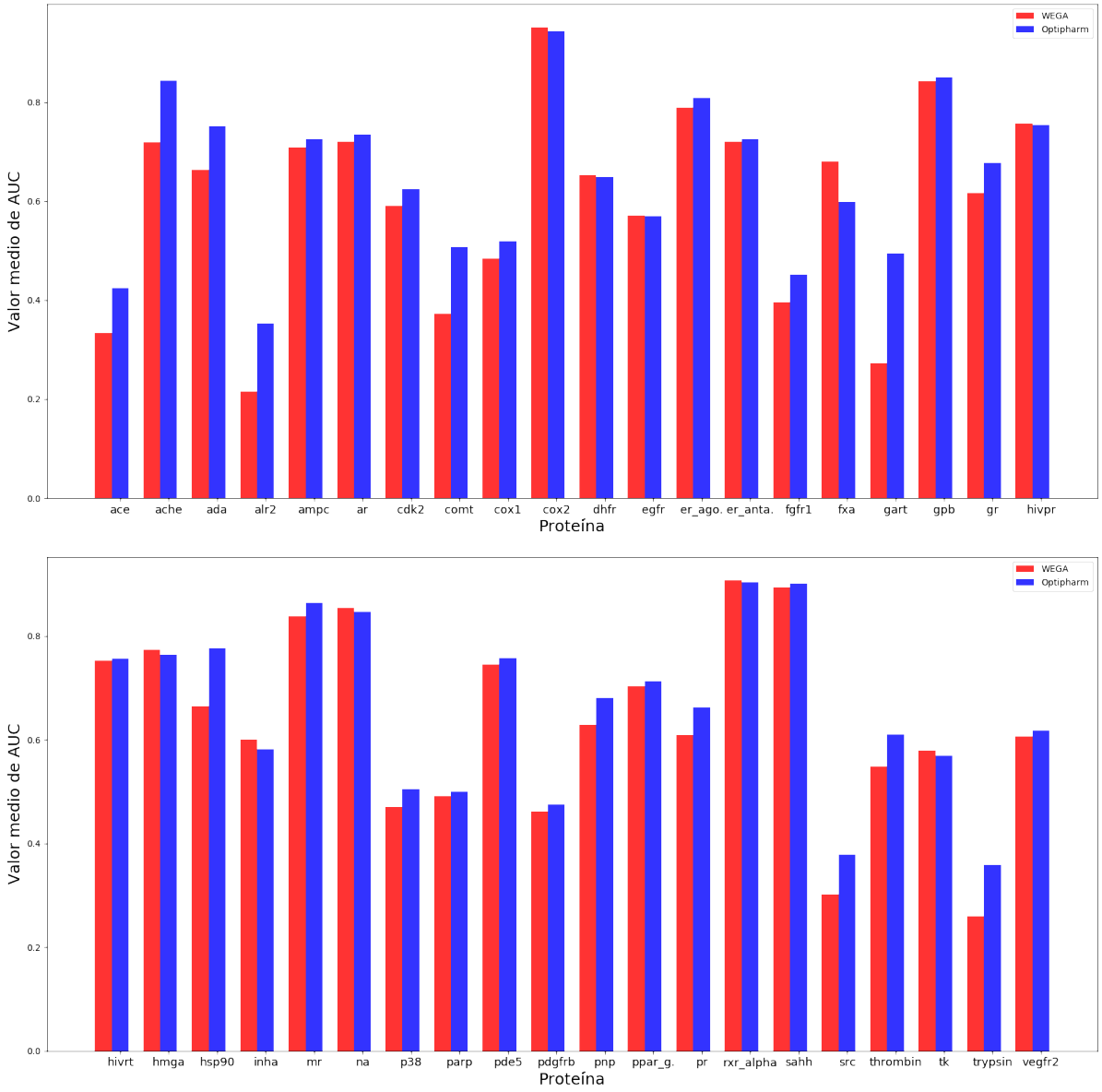


Figura 6.5: Comparativa de valores de AUC de WEGA y la mejor combinación de configuraciones de Optipharm.

7. Conclusiones y trabajo futuro

Éste es el último capítulo de la presente memoria. Se ha destinado a mostrar al lector las conclusiones que se han obtenido de este trabajo así como las propuestas futuras que se pretenden realizar para obtener mejoras y ampliar la variedad y cantidad de los problemas en los que la solución aquí propuesta puede ser aplicada, y es que este trabajo no es sino la primera parte de un proyecto más ambicioso: el doctorado del autor de esta memoria.

7.1 Conclusiones

En este trabajo se ha presentado un algoritmo evolutivo de optimización multimodal denominado Optipharm para resolver un problema de comparación molecular de ligandos basado en la similitud de su forma física. Para obtener el valor de similitud dado un par de moléculas se ha utilizado un modelo matemático gaussiano ponderado. El objetivo del algoritmo consiste en modificar espacialmente una de las moléculas, mediante traslación y rotación, para ajustarla a la posición más parecida de la molécula que se mantiene fija. El número de comparaciones a resolver asciende a miles de cientos en los problemas más pequeños, ampliándose a millones en los más complejos. Esto ha implicado realizar un ajuste dinámico de la configuración donde cada comparación tiene los óptimos parámetros para encontrar la mejor solución en el menor tiempo posible y con la mejor calidad.

Para comparar la calidad de los resultados de Optipharm, se ha seleccionado un algoritmo determinista denominado WEGA. WEGA es actualmente el mejor algoritmo que soluciona el problema planteado en esta memoria. Las bases sobre las que se asienta la propuesta de un algoritmo heurístico son que permite encontrar todos los óptimos locales con un coste menor, debido a la casuística del problema donde no existe una solución fija y explorar de forma exhaustiva todo el espacio de búsqueda requiere de gran cantidad de tiempo, que considerando el volumen de comparaciones que se tienen que realizar, puede resultar inviable.

Los experimentos realizados han demostrado que las soluciones de Optipharm tienen la misma calidad que las de WEGA en lo referente al valor de la función objetivo y como clasificador, considerando por parte del nuestro algoritmo una misma configuración de parámetros de entrada. Si se considera una batería de configuraciones distintas, Optipharm obtiene mejor calidad de la solución que WEGA en valor de AUC.

Muchas han sido las configuraciones que se han estudiado mediante un exhaustivo estudio computacional, esto ha permitido obtener ciertas conclusiones que han conllevado a la versión final del algoritmo Optipharm:

- Es necesario considerar las propiedades físicas de las moléculas antes de su comparación debido a la diversidad de éstas. Un rango de valores puede ser suficiente para una molécula mientras que para otras puede resultar excesivo.
- El valor de la función objetivo es muy sensible por lo que encontrar la mejor solución de forma muy precisa se ha hecho fundamental.
- Se debe de encontrar una buena solución común de parámetros de configuración que se adapte a todas las comparaciones, y es que uno de los principales problema que se ha tenido que afrontar en este trabajo es que no se resuelve un único problema sino cientos de miles y encontrar la configuración común para esa cantidad de experimentos ha sido uno de los handicap de este trabajo.
- Se requiere de un estudio previo del problema a resolver. Existen distintos caminos para obtener el mismo resultado, por lo que dependiendo de las necesidades se desarrolla una solución u otra.

7.2 Trabajo Futuro

Si bien el algoritmo desarrollado ha demostrado un buen comportamiento, el carácter investigador de este trabajo permite tener diversos objetivos en el horizonte todavía por afrontar.

Centrándose en la versión presentada del algoritmo, están propuestas técnicas que reduzcan la tiempo de cómputo, ya sea mediante optimización de código, implementación de aproximaciones matemáticas que proporcionen la misma calidad de la solución en un tiempo muy inferior o utilizando técnicas de computación de altas prestaciones.

Otro objetivo que se tiene marcado es la incorporación de nuevas funciones objetivo. Como se vio en el Capítulo 1, existen otras propiedades de las moléculas (compuestos químicos, proteínas, ligandos, etc.) que pueden ser utilizadas para encontrar similitudes o por el interés de sustituir compuestos que proporcionen los mismos beneficios que otras pero sin tener efectos negativos.

Por último, y en relación al concepto del párrafo anterior, queda construir un algoritmo multiobjetivo donde varias de estas funciones objetivo estén contempladas en los resultados y ya sea un agente superior quien decida cual de los compuestos analizados puede ser interesante para la elaboración de fármacos. Mismamente, a esta idea también se le pueden aplicar técnicas de computación de altas prestaciones para reducir los tiempos.

Bibliografía

[1] R. Abagyan, M. Totrov y D. Kuznetsov. "ICM-a new method for protein modeling and design: applications to docking and structure prediction from the distorted native conformation". En: *Journal of computational chemistry* 15.5 (1994), páginas 488-506.

[2] E. Alba. *Parallel Metaheuristics: A New Class of Algorithms*. Wiley-Interscience, 2005.

[3] E. Alba y M. Tomassini. "Parallelism and evolutionary algorithms". En: *IEEE Transactions on Evolutionary Computation* 6.5 (2002), páginas 443-462.

[4] D. Beasley, D. R. Bull y R. R. Martin. "A sequential niche technique for multimodal function optimization". En: *Evolutionary computation* 1.2 (1993), páginas 101-125.

[5] H. Bersini y G. Seront. "In search of a good evolution-optimization crossover". En: *Parallel Problem Solving from Nature - PPSN II* (1992), páginas 479-488.

[6] H.-G. Beyer y H.-P. Schwefel. "Evolution Strategies &Ndash;A Comprehensive Introduction". En: 1.1 (mayo de 2002), páginas 3-52.

[7] M. Birattari, L. Paquete, T. Stüztle y K. Varrentrapp. *Classification of Metaheuristics and Design of Experiments for the Analysis of Components*. Informe técnico. Technical report, Darmstadt University of Technology, Germany, 2001.

[8] T. Blickle y L. Thiele. *A Comparison of Selection Schemes used in Genetic Algorithms*. Informe técnico. Gloriastrasse 35, CH-8092 Zurich: Swiss Federal Institute of Technology (ETH) Zurich, Computer Engineering y Communications Networks Lab (TIK, 1995.

[9] C. Blum y A. Roli. "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison". En: *ACM Computing Surveys* 35.3 (sep. de 2003), páginas 268-308.

[10] A. Bondi. "van der Waals Volumes and Radii". En: *The Journal of Physical Chemistry* 68.3 (mar. de 1964), páginas 441-451.

[11] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan y M. Karplus. “CHARMM: A program for macromolecular energy, minimization, and dynamics calculations”. En: *Journal of Computational Chemistry* 4.2 (1983), páginas 187-217.

[12] M. L. Connolly. “Computation of molecular volume”. En: *Journal of the American Chemical society* 107.5 (1985), páginas 1118-1124.

[13] C. Darwin. *On the Origin of Species by Means of Natural Selection*. London: Murray, 1859.

[14] Y. Davidor. “A naturally occurring niche and species phenomenon: The model and first results.” En: *Proceedings of the 4th International Conference on Genetic Algorithms*. Editado por R.K. Belew y L.B. Booker. Morgan Kaufmann, 1991, páginas 257-263.

[15] L. Davis, edicíon. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[16] K. A. De Jong. “An Analysis of the Behavior of a Class of Genetic Adaptive Systems.” AAI7609381. Tesis doctoral. Ann Arbor, MI, USA, 1975.

[17] K. Deb. *Genetic algorithms in multimodal function optimization*. TCGA report no. 89002. The University of Alabama, Dept. of Engineering Mechanics, 1989.

[18] M. Dorigo. “Ant colony optimization”. En: *Scholarpedia* 2.3 (2007), página 1461.

[19] M. Dorigo y G. Di Caro. “New Ideas in Optimization”. En: editado por D. Corne, M. Dorigo, F. Glover, D. Dasgupta, P. Moscato, R. Poli y K. V. Price. Maidenhead, UK, England: McGraw-Hill Ltd., UK, 1999. Capítulo The Ant Colony Optimization Meta-heuristic, páginas 11-32.

[20] J. P. Egan. *Signal detection theory and ROC-analysis*. English. Includes bibliographies and index. New York : Academic Press, 1975.

[21] A. E Eiben, P. E. Raue y Z. Ruttkay. “Genetic algorithms with multi-parent recombination”. En: *Parallel Problem Solving from Nature - PPSN III* 866 (1994), páginas 78-87.

[22] A. E. Eiben y J. E. Smith. *Introduction to Evolutionary Computing*. SpringerVerlag, 2003.

[23] R. Enache, B. Sendhoff, M. Olhofer y M. Hasenjäger. “Comparison of Steady-State and Generational Evolution Strategies for Parallel Architectures”. En: *Parallel Problem Solving from Nature - PPSN VIII: 8th International Conference, Birmingham, UK, September 18-22, 2004. Proceedings*. Editado por X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. E. Rowe, P. Tiňo, A. Kabán y H.-P. Schwefel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, páginas 253-262.

[24] L. J. Eshelman y J. D. Schaffer. “Real-Coded Genetic Algorithms and Interval-Schemata.” En: *Foundations of genetic algorithms*. Editado por L. D. Whitley. Morgan Kaufmann, 1992, páginas 187-202.

[25] T. Fawcett. “An introduction to ROC analysis”. En: *Pattern Recognition Letters* 27.8 (jun. de 2006), páginas 861-874.

[26] R. A. Friesner, J. L. Banks, R. B. Murphy, T. A. Halgren, J. J. Klicic, D. T. Mainz, M. P. Repasky, E. H. Knoll, M. Shelley, J. K. Perry y col. “Glide: a new approach for rapid, accurate docking and scoring. 1. Method and assessment of docking accuracy”. En: *Journal of medicinal chemistry* 47.7 (2004), páginas 1739-1749.

[27] K.D. Gibson y H.A. Scheraga. “Exact calculation of the volume and surface area of fused hard-sphere molecules with unequal atomic radii”. En: *Molecular Physics* 62.5 (1987), páginas 1247-1265.

[28] F. Glover. “Future Paths for Integer Programming and Links to Artificial Intelligence”. En: *Computers and Operational Research* 13.5 (mayo de 1986), páginas 533-549.

[29] D. Goldberg. “The theory of virtual alphabets”. En: *Parallel problem solving from nature* (1991), páginas 13-22.

[30] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[31] D. E. Goldberg. “Real-coded genetic algorithms, virtual alphabets, and blocking”. En: *Complex systems* 5.2 (1991), páginas 139-167.

[32] D. E. Goldberg y J. Richardson. “Genetic Algorithms with Sharing for Multimodal Function Optimization”. En: *Genetic Algorithms on Genetic Algorithms and Their Application*. Hillsdale, NJ, USA: Lawrence Erlbaum Associates Inc., 1987, páginas 177-183.

[33] A. C. Good, E. E. Hodgkin y W. G. Richards. “Utilization of Gaussian functions for the rapid evaluation of molecular similarity”. En: *Journal of Chemical Information and Computer Sciences* 32.3 (1992), páginas 188-191.

[34] A. C. Good y W. G. Richards. “Rapid evaluation of shape similarity using Gaussian functions”. En: *Journal of Chemical Information and Computer Sciences* 33.1 (1993), páginas 112-116.

[35] J. A. Grant, M. A. Gallardo y B. T. Pickup. “A Fast Method of Molecular Shape Comparison: A Simple Application of a Gaussian Description of Molecular Shape”. En: *Journal of computational chemistry* 17.14 (1996), páginas 1653-1666.

[36] J. A. Grant y B. T. Pickup. “A Gaussian description of molecular shape”. En: *The Journal of Physical Chemistry* 99.11 (1995), páginas 3503-3510.

[37] J. H. Holland. *Adaptation in Natural and Artificial Systems*. Ann Arbor, Michigan: The University of Michigan Press, 1975.

[38] A. Homaifar, S. Guan y G. E. Liepins. “A New Approach on the Traveling Salesman Problem by Genetic Algorithms”. En: *Proceedings of the 5th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, páginas 460-466.

[39] H. Homayounfar, S. Areibi y F. Wang. “An advanced island based GA for optimization problems”. En: *Proc. Int. DCDIS Conf. on Engineering Applications and Computational*. 2003, páginas 46-51.

[40] *Horizon 2020. The framework programme for research and innovation.* URL: `http://www.ficyt.es/progeur/Horizon2020.asp`.

[41] N. Huang, B. K. Shoichet y J. J. Irwin. “Benchmarking Sets for Molecular Docking”. En: *Journal of Medicinal Chemistry* 49.23 (nov. de 2006), páginas 6789-6801.

[42] C. Igel, T. Suttorp y N. Hansen. “Steady-State Selection and Efficient Covariance Matrix Update in the Multi-objective CMA-ES”. En: *Evolutionary Multi-Criterion Optimization: 4th International Conference, EMO 2007, Matsushima, Japan, March 5-8, 2007. Proceedings*. Editado por S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu y T. Murata. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, páginas 171-185.

[43] M. Jelasity. “The shape of evolutionary search: Discovering and representing search space structure”. Tesis doctoral. Leiden University, 2001.

[44] M. Jelasity y J. Dombi. “GAS, a concept on modeling species in genetic algorithms”. En: *Artificial Intelligence* 99.1 (1998), páginas 1-19.

[45] M. Jelasity, P. M. Ortigosa y I. García. “UEGO, an Abstract Clustering Technique for Multimodal Global Optimization”. En: *Journal of Heuristics* 7.3 (2001), páginas 215-233.

[46] R. Kicinger, T. Arciszewski y K. De Jong. “Evolutionary Computation and Structural Design: A Survey of the State-of-the-art”. En: *Comput. Struct.* 83.23-24 (sep. de 2005), páginas 1943-1978.

[47] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[48] J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge, MA, USA: MIT Press, 1994.

[49] P. L. Lanzi, W. Stolzmann y S. W. Wilson, edicíon. *Learning Classifier Systems: From Foundations to Applications*. Volumen 1813. LNAI. Springer, Berlin, Heidelberg, New York, 2000.

[50] V. Law, C. Knox, Y. Djoumbou, T. Jewison, A. C. Guo, Y. Liu, A. Maciejewski, D. Arndt, M. Wilson, V. Neveu, A. Tang, G. Gabriel, C. Ly, S. Adamjee, Z. T. Dame, B. Han, Y. Zhou y D. S. Wishart. “DrugBank 4.0: shedding new light on drug metabolism”. En: *Nucleic Acids Research* 42.D1 (2014), página D1091.

[51] S. Luke. “Issues in Scaling Genetic Programming: Breeding Strategies, Tree Generation, and Code Bloat”. Tesis doctoral. A. V. Williams Building, University of Maryland, College Park, MD 20742 USA: Department of Computer Science, University of Maryland, 2000.

[52] P. C. H. Ma, K. C. C. Chan, X. Yao y D. K. Y. Chiu. “An evolutionary clustering algorithm for gene expression microarray data analysis”. En: *IEEE Transactions on Evolutionary Computation* 10.3 (jun. de 2006), páginas 296-314.

[53] S. W. Mahfoud. “Niching methods for genetic algorithms”. Tesis doctoral. University of Illinois at Urbana-Champaign, Urbana, IL, 1995.

[54] A. Marczyk. *Genetic algorithms and evolutionary computation*. URL: `http://www.talkorigins.org/faqs/genalg/genalg.html`.

[55] A. V. Marenich, R. M. Olson, C. P. Kelly, C. J. Cramer y D. G. Truhlar. “Self-Consistent Reaction Field Model for Aqueous and Nonaqueous Solutions Based on Accurate Polarized Partial Charges”. En: *Journal of Chemical Theory and Computation* 3.6 (2007). PMID: 26636198, páginas 2011-2033.

[56] B. B. Masek, A. Merchant y J. B. Matthew. “Molecular shape comparison of angiotensin II receptor antagonists”. En: *Journal of medicinal chemistry* 36.9 (1993), páginas 1230-1238.

[57] M. R. McGann, H. R. Almond, A. Nicholls, J. A. Grant y F. K. Brown. “Gaussian docking functions”. En: *Biopolymers* 68.1 (2003), páginas 76-90.

[58] E. C. Meng, B. K. Shoichet y I. D. Kuntz. “Automated docking with grid-based energy evaluation”. En: *Journal of computational chemistry* 13.4 (1992), páginas 505-524.

[59] M. D. Miller, S. K. Kearsley, D. J. Underwood y R. P. Sheridan. “FLOG: a system to select ’quasi-flexible’ ligands complementary to a receptor of known three-dimensional structure”. En: *Journal of computer-aided molecular design* 8.2 (1994), páginas 153-174.

[60] H. Muhlenbein y H. Voigt. “Gene pool recombination in genetic algorithms”. En: *Proceeding of the Metaheuristics Conference*. Editado por I.H. Osman y J.P. Kelly. Volumen 1. Kluwer Academic Publishers, 1995, páginas 53-62.

[61] A. Neumaier. “Complete search in continuous global optimization and constraint satisfaction”. En: *Acta numerica* 13 (2004), páginas 271-369.

[62] A. Nicholls, N. E. MacCuish y J. D. MacCuish. “Variable selection and model validation of 2D and 3D molecular descriptors”. En: *Journal of Computer-Aided Molecular Design* 18.7 (2004), páginas 451-474.

[63] R. M. Olson, A. V. Marenich, C. J. Cramer y D. G. Truhlar. “Charge Model 4 and Intramolecular Charge Polarization”. En: *Journal of Chemical Theory and Computation* 3.6 (2007), páginas 2046-2054.

[64] *ROCS*. Santa Fe, NM, USA, 2008. URL: `www.eyesopen.com`.

[65] M. Orozco y F. J. Luque. “Molecular interaction potential: A new tool for the theoretical study of molecular reactivity”. En: *Journal of Computational Chemistry* 14.5 (1993), páginas 587-602.

[66] P. M. Ortigosa. “Métodos Estocásticos de Optimización Global. Procesamiento Paralelo.” Tesis doctoral. Universidad de Málaga, 1999.

[67] P. M. Ortigosa, I. García y M. Jelasity. “Reliability and Performance of UEGO, a Clustering-based Global Optimizer”. En: *Journal of Global Optimization* 19.3 (2001), páginas 265-289.

[68] P. M. Ortigosa, J. L. Redondo, I. García y J. J. Fernández. “A population global optimization algorithm to solve the image alignment problem in electron crystallography”. En: *Journal of Global Optimization* 37.4 (2007), páginas 527-539.

[69] I. C. Parmee y C. R. Bonham. “Improving cluster oriented genetic algorithms for high-performance region identification”. En: *Proceedings US United Engineering Foundation’s ‘Optimisation in industry’ conference*. 2002.

[70] K. Pearson. *Mathematical contributions to the theory of evolution*. Mathematical Contributions to the Theory of Evolution v. 13-17. Dulau y co., 1904.

[71] H. E. Pence y A. Williams. “ChemSpider: An Online Chemical Information Resource”. En: *Journal of Chemical Education* 87.11 (2010), páginas 1123-1124.

[72] *Plan Estatal de investigación científica, técnica y de innovación*. URL: `http://www.idi.mineco.gob.es/stfls/MICINN/Investigacion/FICHEROS/Plan_Estatal_Inves_cientifica_tecnica_innovacion.pdf`.

[73] R. Poli y W. B. Langdon. *Genetic programming with one-point crossover and point mutation*. Technical Report CSRP-97-13. Birmingham, B15 2TT, UK: School of Computer Science, The University of Birmingham, 1997.

[74] Timothy J. R. “Solvent accessible surface area and excluded volume in proteins”. En: *Journal of Molecular Biology* 178.1 (1984), páginas 63-89.

[75] M. Rarey, B. Kramer, T. Lengauer y G. Klebe. “A fast flexible docking method using an incremental construction algorithm”. En: *Journal of molecular biology* 261.3 (1996), páginas 470-489.

[76] H. Ratschek y J. Rokne. *New Computer Methods for Global Optimization*. New York, NY, USA: Halsted Press, 1988.

[77] J. L. Redondo. *Solving competitive location problems via memetic algorithms. High performance computing approaches*. Universidad de Almería, 2009.

[78] J. L. Redondo, J. Fernández, I. García y P. M. Ortigosa. "Parallel algorithms for continuous competitive location problems". En: *Optimization Methods & Software* 23.5 (2008), páginas 779-791.

[79] J. L. Redondo, J. Fernández, I. García y P. M. Ortigosa. "Parallel algorithms for continuous multifacility competitive location problems". En: *Journal of Global Optimization* 50.4 (2011), páginas 557-573.

[80] J. L. Redondo, J. Fernández, I. García y P. M. Ortigosa. "Solving the facility location and design (1||1)-centroid problem via parallel algorithms". En: *Journal of Supercomputing* 58.3 (2011), páginas 420-428.

[81] J. L. Redondo, I. García y P. M. Ortigosa. "Parallel evolutionary algorithms based on shared memory programming approaches". En: *Journal of Supercomputing* 58.2 (2011), páginas 270-279.

[82] J. L. Redondo, I. García, P. M. Ortigosa, B. Pelegrín y P. Fernández. "Parallelization of an algorithm for finding facility locations for an entering firm under delivered pricing". En: *Parallel Computing: Current and Future Issues of High-End Computing*. Editado por G. R. Joubert, W. E. Nagel, F. J. Peters, O. Plata, P. Tirado y E. Zapata. Volumen 33. NIC series. John von Neumann Institute for Computing, 2006, páginas 269-276.

[83] F.J. Solis y R. J. B. Wets. "Minimization by Random Search Techniques". En: *Math. Oper. Res.* 6.1 (feb. de 1981), páginas 19-30.

[84] University of Southern California. *COCOMO II. Model Definition Manual. Version 1.4*.

[85] K. A. Spackman. "Signal detection theory: Valuable tools for evaluating inductive learning". En: *Proceedings of the sixth international workshop on Machine learning*. Morgan Kaufmann Publishers Inc. 1989, páginas 160-163.

[86] J. A. Swets, R. M. Dawes y J. Monahan. "Better decisions through science." En: *Scientific American* 283.4 (2000), páginas 82-87.

[87] G. Syswerda. "Simulated crossover in genetic algorithms". En: *Proceeding of the Second Workshop on Foundations of Genetic Algorithms*. Editado por L.D. Whitley. Morgan Kaufmann Publishers, 1993, páginas 239-255.

[88] A. Torn y A. Zilinskas. *Global Optimization*. New York, NY, USA: Springer-Verlag New York, Inc., 1989.

[89] *Tripos Mol2 File Format*. URL: `http://chemyang.ccnu.edu.cn/ccb/server/AIMMS/mol2.pdf`.

[90] F. Vázquez, J. J. Fernández y E. M. Garzón. "A new approach for sparse matrix vector product on NVIDIA GPUs". En: *Concurrency and Computation: Practice and Experience* 23 (2011), páginas 815-826.

[91] F. Vázquez, J. J. Fernández y E. M. Garzón. "Automatic tuning of the sparse matrix vector product on GPUs based on the ELLR-T approach". En: *Parallel Computing* 38.8 (2012), páginas 408-420.

[92] F. Vázquez, E. M. Garzón y J. J. Fernández. "A matrix approach to tomographic reconstruction and its implementation on GPUs". En: *Journal of Structural Biology* 170 (2010), páginas 146-151.

[93] F. Vázquez, E. M. Garzón y J. J. Fernández. “Matrix Implementation of Simultaneous Iterative Reconstruction Technique (SIRT) on GPUs”. En: *The Computer Journal* 55.11 (2011), páginas 1861-1868.

[94] M. L Verdonk, J. C. Cole, M. J. Hartshorn, C. W. Murray y R. D. Taylor. “Improved protein–ligand docking using GOLD”. En: *Proteins: Structure, Function, and Bioinformatics* 52.4 (2003), páginas 609-623.

[95] P. K. Weiner y P. A. Kollman. “AMBER: Assisted model building with energy refinement. A general program for modeling molecules and their interactions”. En: *Journal of Computational Chemistry* 2.3 (1981), páginas 287-303.

[96] D. Whitley y T. Hanson. “Optimizing Neural Networks Using Faster, More Accurate Genetic Search”. En: *Proceedings of the Third International Conference on Genetic Algorithms*. George Mason University, USA: Morgan Kaufmann Publishers Inc., 1989, páginas 391-396.

[97] D. Whitley, T. Starkweather y C. Bogart. “Genetic algorithms and neural networks: optimizing connections and connectivity"”. En: *Parallel Computing* 14.3 (1990), páginas 347-361.

[98] X. Yan, J. Li, Z. Liu, M. Zheng, H. Ge y J. Xu. “Enhancing Molecular Shape Comparison by Weighted Gaussian Functions”. En: *Journal of Chemical Information and Modeling* 53.8 (ago. de 2013), páginas 1967-1978.

[99] L. Yu, K. Liu y K. Li. “Ant colony optimization in continuous problem”. En: *Frontiers of Mechanical Engineering in China* 2.4 (2007), páginas 459-462.

[100] K. H. Zou. “Receiver operating characteristic (ROC) literature research”. En: (2002).

UNIVERSIDAD DE ALMERÍA

Resumen/Abstract

La industria farmacéutica precisa de una media de 12 a 20 años y unos costes de aproximadamente 850 millones de euros para el desarrollo y lanzamiento de un nuevo fármaco al mercado, por lo que son necesarias herramientas software que reduzcan los gastos y el tiempo.

Para solucionar parte de este problema, este trabajo aborda el desarrollo de una herramienta que permite seleccionar, de entre millones de compuestos, aquellos que son más afines a un blanco farmacológico dado comparando su forma molecular.

Concretamente, se presenta una algoritmo evolutivo de optimización global denominado Optipharm. Esta herramienta incorpora diversas técnicas que permite adaptarse dinámicamente a las diferentes propiedades físicas que presenta cada molécula en particular.

Los experimentos se realizan en base a la evaluación de la similitud de la forma molecular y la clasificación de moléculas afines mediante el cálculo del AUC ROC. Los resultados muestran una calidad similar e incluso mejor que la mejor referencia bibliográfica.

The pharmaceutical industry needs an average of 12 to 20 years and costs of approximately 850 million euros for the development and approval of a new drug to the market, so that software tools are necessary to reduce costs and time.

To solve part of this problem, this work approaches the development of a tool that allows to select, among millions of compounds, those that are more related to a pharmacological target comparing its molecular shape.

Specifically, an evolutionary optimization algorithm called Optipharm is presented. This tool incorporates diverse techniques that allow to adapt dynamically to the different physical properties that each single molecule.

The experiments are performed based on the evaluation of the similarity of the molecular shape and the classification of related molecules by the calculation of the AUC ROC. The results show a similar quality and even better than the best literature reference.