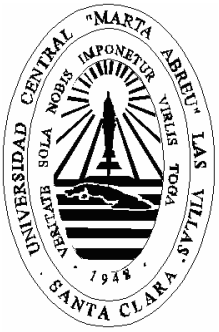
Universidad Central “Marta Abreu” de Las Villas
Facultad de Matemática, Física y Computación



Trabajo para optar por Título de
Licenciado en Ciencia de la Computación

Título

Plataforma evolutiva para la optimización de compuestos químicos modelados mediante aprendizaje automático con descriptores algebraicos.

Autor

José Ricardo Valdés Martiní

Tutores

Dr. Carlos Morell Pérez
Dr. Yovani Marrero Ponce

Inteligencia Artificial
2010

Resumen

Los métodos que emplean técnicas asistidas por ordenador usados en el descubrimiento, diseño, y optimización de compuestos con estructura y propiedades deseadas han desempeñado un rol importante en el desarrollo de fármacos que se encuentran actualmente en el mercado o en fase de estudios clínicos. El resultado de este tipo de estudio *in silico* disminuye el costo de los procesos de síntesis y bioensayos que se llevan a cabo solo después que se exploran conceptos iniciales mediante modelos computacionales.

Recientemente, la comunidad científica ha introducido el uso he la computación evolutiva en el descubrimiento de fármacos debido a su potencialidad para realizar una búsqueda efectiva en un espacio enorme.

Este trabajo describe una herramienta computacional que integra de modo natural bibliotecas de software especializadas en algoritmos genéticos, química computacional y aprendizaje automático para la optimización de compuestos químicos. Para su validación se utiliza en forma experimental la optimización de una molécula con vistas a disminuir su toxicidad.

Abstract

The methods used in the field of Computer aided Drug Discovery, Design, and Optimization of compounds with desired structures and properties, have played an important role in the development of drugs in the current market or in its experimental phase. The result of this kind of *in silico* study saves significant experimental costs of the synthesis processes and bioassays carried out after exploring initial concepts by means of computational models.

Recently, the scientific community has introduced the use of evolutionary computation in drug discovery due to its potential to achieve an effective search in a huge space.

This work describes a computational tool that integrates, in a natural way, specialized software libraries from the fields of genetic algorithms, computational chemistry and Machine Learning for the Drug Design Optimization. An experimental evaluation using real data from toxicity domain had been carried out to show the effectiveness of such approach.

Tabla de Contenidos

Introducción

La búsqueda de nuevos fármacos en la terapia de procesos patológicos como el SIDA, las enfermedades cardiovasculares y neoplásicas, la enfermedad de Alzheimer, y una inmensa variedad de infecciones de naturaleza viral y parasitaria, se encuentra entre las prioridades de la industria farmacéutica en la actualidad (PIProfile, 2009).

Las estrategias actuales para el descubrimiento/diseño de fármacos están determinadas por el desarrollo de la química combinatoria, las técnicas HTS (acrónimo de *High-Throughput Screening*), la ingeniería genética, la biología molecular y la bioinformática (Kubinyi, 2004, Dixit and Mitra, 2002).

Estos avances han permitido que en el presente, el número de compuestos químicos conocidos supere los 50 millones. Una parte considerable de ellos (más de un millón) están disponibles a través de diferentes proveedores (Service, 2009). Además, la mayoría de las compañías farmacéuticas cuentan con colecciones propias de compuestos que representan una fuente de mucho valor para el descubrimiento y optimización de compuestos líderes. Por otro lado, el empleo de la robótica y el desarrollo de las técnicas HTS han hecho posible la evaluación diaria de las propiedades biológicas de millones de compuestos contra cientos de dianas biológicas (Dekker, 2000).

Sin embargo, un estudio reciente, que incluyó aproximadamente 50 compañías e instituciones académicas, reveló que se invierten como promedio 880 millones de dólares y 15 años de investigación para el desarrollo de un nuevo fármaco, desde su descubrimiento hasta su uso terapéutico. Aproximadamente el 14% del costo se emplea en la evaluación de propiedades biológicas y farmacológicas durante la fase pre-clínica. A pesar de estos avances y las inversiones millonarias, el número de nuevos fármacos introducidos en el mercado ha permanecido constante durante los últimos años (Smith, 2002, Drews, 2000).

En este contexto, el diseño de herramientas computacionales que permitan seleccionar un número reducido de compuestos potencialmente activos a partir de bibliotecas de

compuestos químicos disponibles, surge como una alternativa a los métodos de síntesis y tamizaje reales que permitiría la reducción de los costos en términos de recursos materiales, humanos y de tiempo.

Los métodos que emplean técnicas asistidas por ordenador usados en el descubrimiento, diseño, y optimización de compuestos con estructura y propiedades deseadas han desempeñado un rol importante en el desarrollo de fármacos que se encuentran actualmente en el mercado o en fase de estudios clínicos (Estrada and Uriarte, 2001). El resultado de este tipo de estudio *in silico* puede aplicarse como estrategia de *tamizaje virtual* que pospone los costosos procesos de síntesis y bioensayos, que se llevan a cabo solo después que se exploran conceptos iniciales mediante modelos computacionales.

Entre las herramientas empleadas en el tamizaje virtual se encuentran los modelos que relacionan cuantitativamente aspectos estructurales y propiedades de las moléculas. Este tipo de estudio de *Relación Cuantitativa Estructura-Actividad/Propiedad*, es conocido ampliamente en la literatura como *estudios QSAR/QSPR* (acrónimo de *Q*uantitative *S*tructure *A*ctivity/*P*roperty *R*elationships). Este tipo de análisis proporciona una vía para estimar, con aceptable grado de precisión, la actividad/propiedad de nuevos compuestos y permite obtener una interpretación en términos estructurales de la actividad/propiedad estudiada.

El desarrollo de modelos computacionales basados en la relación estructura–actividad/propiedad requiere de la representación adecuada de la estructura molecular. Estas representaciones se logran mediante los descriptores moleculares (DMs), términos que caracterizan un aspecto específico de la molécula. Recientemente, el Dr. Marrero Ponce, ha propuesto un nuevo esquema de diseño molecular asistido por computadora, denominado **TOMOCOMD** (acrónimo de **TO**pological **MO**lecular **COM**puter **D**esign). Esta metodología se basa en la caracterización de la estructura molecular por medio de familias de descriptores moleculares 2D y 3D, denominados índices lineales y cuadráticos en analogía a las *formas lineale*s y *cuadráticas*, respectivamente. Estos enfoques se han aplicado con éxito en la identificación de compuestos químicos con actividad/propiedad de interés (Marrero-Ponce et al., 2004), la predicción de propiedades

físico-químicas de compuestos orgánicos (Marrero-Ponce, 2003), etc. Estos índices han sido extendidos a la caracterización numérica de la estructura de macromoléculas (ácidos nucleicos y proteínas) y se han empleado en la modelación de la interacción de fármacos con ácidos nucleicos y estudios de estabilidad en una serie de mutantes (Marrero-Ponce et al., 2005).

Otra de las herramientas empleadas en el *tamizaje virtual* es la búsqueda heurística en un intento por explorar un espacio de búsqueda de tamaño exponencial. Los algoritmos genéticos en particular presentan una adecuada efectividad en los espacios de búsqueda considerados "grandes" y por ello han sido empleados con éxito en el proceso de descubrimiento, diseño, y optimización de compuestos con estructura y propiedades deseadas.

En este contexto el **objetivo general** de nuestro trabajo consiste en el desarrollo de una herramienta computacional que integre de modo natural bibliotecas de software especializadas en algoritmos genéticos, informática-química y aprendizaje automático para la optimización de compuestos químicos.

En la consecución de este objetivo nos guían las siguientes **preguntas de investigación**:

¿Qué biblioteca de software para el trabajo en química computacional ofrece mejores prestaciones y facilidades para incluir el trabajo con descriptores moleculares propios?

¿Qué biblioteca de software para el trabajo con algoritmos genéticos seleccionar para su integración y extensión al trabajo en química computacional?

¿Cómo integrar la estimación de modelos de aprendizaje automático para la predicción de cierta actividad biológica de interés con las herramientas previamente seleccionadas?

Para lograr nuestro objetivo general nos hemos planteado los siguientes **objetivos específicos**:

1- Seleccionar una biblioteca líder en el campo de la química computacional desarrollada en java y de código abierto que permita trabajar con una variedad de

formatos de moléculas y haga sencilla su extensión para la inclusión de descriptores moleculares propios.

2- Extender la biblioteca química seleccionada mediante la adición de una nueva familia de índices moleculares basados en formas algebraicas.

3- Seleccionar una biblioteca líder en el campo de los algoritmos genéticos, desarrollada en java y de código abierto, que permita trabajar con optimización con uno o varios objetivos y que se pueda extender para acoplar las estructuras moleculares al sistema evolutivo.

4- Extender la biblioteca evolutiva seleccionada mediante la adición de una nueva codificación para individuos, especies y operadores genéticos

5- Proponer una solución efectiva y de fácil uso para incorporar a la función de evaluación de la calidad de un individuo, modelos que estimen una actividad biológica de interés a partir de técnicas de aprendizaje automático.

Este trabajo se ha estructurado en tres capítulos. El primero de ellos trata sobre el empleo de la computación evolutiva en el campo de la química computacional y sirve de marco teórico para nuestra investigación. En él se analizan y exponen aquellas teorías, enfoques teóricos, investigaciones y antecedentes en general válidos para el correcto encuadre del estudio. Como resultado se adoptan las herramientas necesarias para la consecución de los objetivos. El segundo capítulo está dedicado al desarrollo de nuevos descriptores moleculares y su implementación en la biblioteca seleccionada. El tercer capítulo se adentra en el campo de los algoritmos genéticos, detallando la incorporación de nuevas estructuras de software necesarias para el trabajo con representaciones moleculares. Este capítulo incluye una aplicación práctica que permite comprobar la validez de la integración propuesta. Al finalizar se proveen las conclusiones de nuestro trabajo así como propuestas para trabajos futuros.

1 Computación evolutiva y química computacional

Este capítulo presenta a los algoritmos genéticos, como mecanismos de resolución de problemas de optimización, y emprende un análisis de las aplicaciones de la química computacional para el diseño de fármacos.

1.1 Algoritmos Genéticos

Expuesto concisamente, un algoritmo genético (o AG para abreviar) es una técnica de programación que imita a la evolución biológica como estrategia para resolver problemas y constituyen una de las técnicas de computación evolutiva más difundidas en la actualidad, como consecuencia de su versatilidad para resolver un amplio rango de problemas. Al constituir un caso de técnica evolutiva, los algoritmos genéticos basan su operativa en una emulación de la evolución natural de los seres vivos, trabajando sobre una población de soluciones potenciales evoluciona de acuerdo a interacciones y transformaciones únicas. Los individuos que constituyen la población se esfuerzan por sobrevivir: una selección programada en el proceso evolutivo, inclinada hacia los individuos más aptos, determina aquellos individuos que formarán parte de la siguiente generación. El grado de adaptación de un individuo se evalúa de acuerdo al problema a resolver, mediante la definición de una función de adecuación al problema, la *función de fitness*.

Bajo ciertas condiciones, el mecanismo definido por los operadores inspirados por la genética natural y la evolución darwiniana lleva a la población a converger hacia una solución aproximada al óptimo del problema, luego de un determinado número de generaciones.

La formulación tradicional de un algoritmo genético fue presentada de excelente manera en el texto de Goldberg (Goldberg, 1989), el cual popularizó el uso de los Algoritmos Genéticos para una variada gama de problemas en las áreas de búsqueda, optimización y aprendizaje automático.

En su formulación clásica, los algoritmos genéticos se basan en el esquema genérico de un Algoritmo Evolutivo presentado en la Figura 1.1. A partir de este esquema, el algoritmo genético define "Operadores Evolutivos" que implementan la recombinación de individuos (el operador de *cruzamiento*) y la variación aleatoria para proporcionar diversidad (el operador de *mutación*), resultando el esquema presentado en la Figura 1.2.

```
Inicializar(P(0))
generación=0
mientras (no CriterioParada) hacer
       Evaluar(P(generación))
       Padres = Seleccionar(P(generación))
       Hijos = Aplicar Recombinación (Padres)
       Hijos = Aplicar Mutación (Hijos)
       NuevaPob = Reemplazar(Hijos, P(generación))
       generación ++
       P(generación) = NuevaPob
fin
retornar Mejor Solución Encontrada
```

Figura 1.2: Esquema genérico de un algoritmo genético.

La característica distintiva de los algoritmos genéticos respecto a las otras técnicas evolutivas consiste en su uso fundamental del cruzamiento como operador principal, mientras que la mutación se utiliza como operador secundario tan solo para agregar una nueva fuente de diversidad en el mecanismo de exploración del espacio de soluciones del problema.

Inclusive la mutación puede llegar a ser un operador opcional o estar ausente en algunas variantes de algoritmos genéticos que utilizan otros operadores para introducir diversidad.

En general, los algoritmos genéticos se han utilizado para trabajar con codificaciones binarias para problemas de búsqueda en espacios de cardinalidad numerable, aunque su alto nivel de aplicabilidad ha llevado a proponer su trabajo con codificaciones reales, e inclusive con codificaciones no tradicionales, dependientes de los problemas a resolver.

Algunos autores consideran que el uso del mecanismo de selección denominado *selección proporcional*, que determina la cantidad de copias de individuos a considerar

en la recombinación de forma proporcional a sus valores de *fitness*, es una característica que distingue a los algoritmos genéticos (Back et al., 1997). Pero tomando en cuenta la diversidad de mecanismos de selección utilizados en las propuestas de algoritmos genéticos en los últimos años es posible concluir que si bien en general se recurre a la selección proporcional, otros mecanismos son igualmente utilizados para modificar el proceso de exploración del espacio de soluciones de los diferentes problemas abordados.

1.1.1 Representación de soluciones

Los algoritmos genéticos no trabajan directamente sobre las soluciones del problema en cuestión, sino que lo hacen sobre una abstracción de los objetos solución, usualmente denominadas *cromosomas* por analogía con la evolución natural biológica. Un cromosoma es un vector de *genes*, mientras que el valor asignado a un gen se denomina *alelo*.

En la terminología biológica, *genotipo* denota al conjunto de cromosomas que definen las características de un individuo. El genotipo sometido al medio ambiente se denomina *fenotipo*. En términos de los algoritmos genéticos el genotipo también está constituido por cromosomas, utilizándose generalmente un único cromosoma por individuo solución al problema. Por ello suelen utilizarse indistintamente los términos *genotipo*, *cromosoma* e *individuo*. Por su parte, el *fenotipo* representa un punto del espacio de soluciones del problema.

Dado que un algoritmo genético trabaja sobre *cromosomas*, se debe definir una función de *codificación* sobre los puntos del espacio de soluciones, que mapea todo punto del espacio de soluciones en un genotipo. La función inversa de la codificación, denominada *decodificación* permite obtener el fenotipo asociado a un cromosoma.

Tomando en cuenta la observación anterior, los mecanismos de codificación de individuos solución resultan importantes para el proceso de búsqueda de los algoritmos genéticos. Habitualmente los algoritmos genéticos utilizan codificaciones *binarias* de largo fijo. Los individuos se codifican por un conjunto de cardinalidad conocida de valores binarios (ceros y unos) conocido como *string de bits* o *bitstring*. Cada *bitstring*

representa a una solución potencial del problema de acuerdo al mecanismo de codificación predefinido, en general dependiente del problema. Otros esquemas de codificación han sido utilizados con menor frecuencia en los algoritmos genéticos. En particular las codificaciones basadas en números reales son útiles para representar soluciones cuando se resuelven problemas sobre espacios de cardinalidad no numerable, como en el caso de determinación de parámetros en problemas de control o entrenamiento de redes neuronales. Los esquemas basados en permutaciones de enteros son útiles para problemas de optimización combinatoria que involucran hallar ordenamientos óptimos, como los problemas de *scheduling* o el reconocido Traveling Salesman Problem.

Codificaciones dependientes de los problemas se han propuesto con frecuencia, como un mecanismo que permite incorporar conocimiento específico en la resolución de problemas complejos.

La Figura 1.3 resume gráficamente la relación entre el espacio de soluciones de un problema, la población de cromosomas con la cual trabaja el algoritmo genético y las funciones de codificación y decodificación.

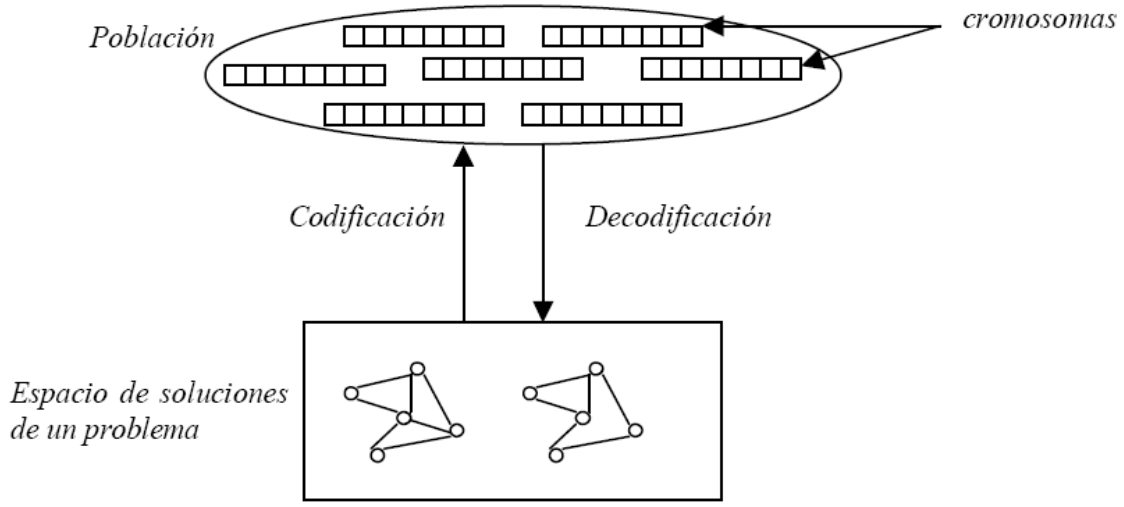


Figura 1.3: Codificación de soluciones en un algoritmo genético.

1.1.2 Función de evaluación

Todo cromosoma tiene un valor asociado de *fitness* que evalúa aptitud del individuo para resolver el problema en cuestión. La función de fitness tiene el mismo tipo que la función objetivo del problema, lo cual implica que el cálculo del valor de fitness se realiza sobre el fenotipo correspondiente al cromosoma, como se presenta en la Figura 1.4.

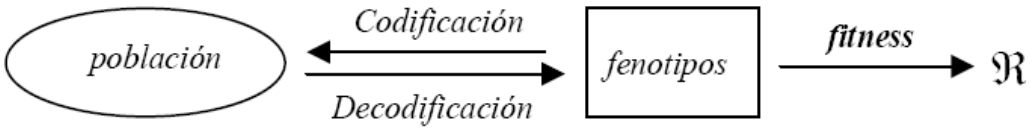


Figura 1.4: Codificación y función de fitness.

La función de fitness tiene una influencia importante en el mecanismo del AG. Si bien actúa como una caja negra para el proceso evolutivo, la función de fitness guía el mecanismo de exploración, al actuar representando al entorno que evalúa la bondad de un individuo solución para la resolución del problema.

Varios puntos han sido identificados como importantes al momento de construir la función de fitness. De acuerdo a (Alba and Cotta, 2003), es posible indicar que:

- La función de fitness debe contemplar el criterio del problema de optimización (minimización o maximización de un objetivo) y las restricciones presentes en el problema de optimización. En caso de surgir soluciones no factibles del problema, la función de fitness deberá asignarle valores adecuados que garanticen que tales individuos no se perpetúen durante el proceso evolutivo (valores muy pequeños en el caso de un problema de maximización y muy elevados en el caso de un problema de minimización). A tales efectos diversas transformaciones a aplicar sobre funciones de fitness han sido definidas para mapear problemas de minimización a maximización y aplicar penalizaciones a soluciones no factibles (Goldberg, 1989).

- Deben contemplarse los casos en que el entorno presente problemas para la evaluación de la función de fitness, utilizando evaluaciones parciales cuando existen valores de fitness no definidos. Asimismo, debe considerarse el uso de funciones de fitness multivaluadas que asignen diferentes valores a un mismo individuo para simplificar la labor del operador de selección.

- El caso de funciones de fitness que varían dinámicamente durante la evolución del algoritmo genético debe tenerse en cuenta los mecanismos complejos como las representaciones múltiples, que son útiles para introducir memoria en la operativa del algoritmo genético.

- En caso de problemas con objetivos múltiples, todos ellos deben estar contemplados en la función de fitness. La utilización de algoritmos genéticos para la resolución de problemas multiobjetivo constituye en sí misma una subárea de investigación con complejidades inherentes.

- Para resolver problemas de dominancia de soluciones muy adaptadas en generaciones tempranas de la evolución, y para evitar el estancamiento en poblaciones similares al final de la evolución, deben considerarse mecanismos de *escalado* de los valores de fitness (Michalewicz, 1992).

En general, la función de fitness será compleja de evaluar, en particular demandará un esfuerzo computacional mucho mayor que el requerido para realizar los operadores evolutivos. Inclusive podría ocurrir que el proceso de evaluación sea tan complejo que solamente valores aproximados pudieran obtenerse en tiempos razonables. Este aspecto será importante al momento de proponer técnicas de alto desempeño para mejorar la eficiencia de los algoritmos genéticos.

1.1.3 Operadores

La gran mayoría de las variantes de algoritmos genéticos utiliza como operadores a la selección, la recombinación y la mutación. El mecanismo de selección determina el modo de perpetuar *buenas* características, que se asumen son aquellas presentes en los

individuos más adaptados. El mecanismo de *selección proporcional* o *selección por ruleta* elige aleatoriamente individuos utilizando una ruleta sesgada, en la cual la probabilidad de ser seleccionado es proporcional al fitness de cada individuo. Otros mecanismos de selección introducen diferentes grados de elitismo, conservando un cierto número prefijado de los mejores individuos a través de las generaciones. En el caso de la *selección por torneo* se escogen aleatoriamente un determinado número de individuos de la población, los cuales compiten entre ellos para determinar cuáles se seleccionarán para reproducirse, de acuerdo a sus valores de fitness. El mecanismo de *selección basado en el rango* introduce el mayor grado de elitismo posible, al mantener entre generaciones un porcentaje generalmente elevado, de los mejores individuos de la población.

Estas diferentes políticas de selección, conjuntamente con políticas similares utilizadas para determinar los individuos reemplazados por los descendientes generados posibilitan el diseño de diferentes modelos evolutivos para los algoritmos genéticos.

Los esquemas de codificación binaria de largo fijo tienen como ventaja principal que resulta sencillo definir operadores evolutivos simples sobre ellos. En la formulación clásica de un algoritmo genético, denominada por Goldberg como *Algoritmo Genético Simple* (Goldberg, 1989), se propone como operador de recombinación el *cruzamiento de un punto*, que consiste en obtener dos descendientes a partir de dos individuos padres seleccionando un punto al azar, cortando los padres e intercambiando los trozos de cromosoma, tal como se presenta en la Figura 1.5.

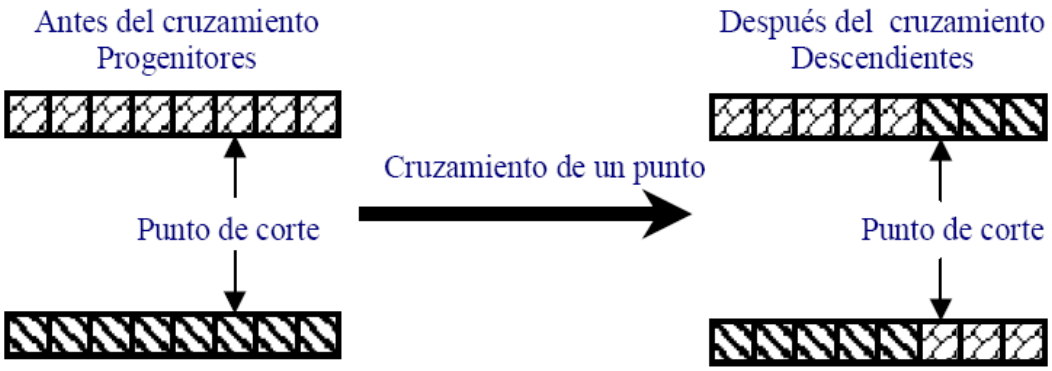


Figura 1.5: Esquema del cruzamiento de un punto.

El operador tradicional de mutación introduce diversidad en el mecanismo evolutivo, simplemente modificando aleatoriamente uno de los valores binarios del cromosoma. Sobre un esquema de codificación binaria la modificación consiste en invertir el valor binario de un alelo, y por ello recibe el nombre de mutación de inversión del valor de un bit (*flip bit*); su operativa se ejemplifica en la Figura 1.6.

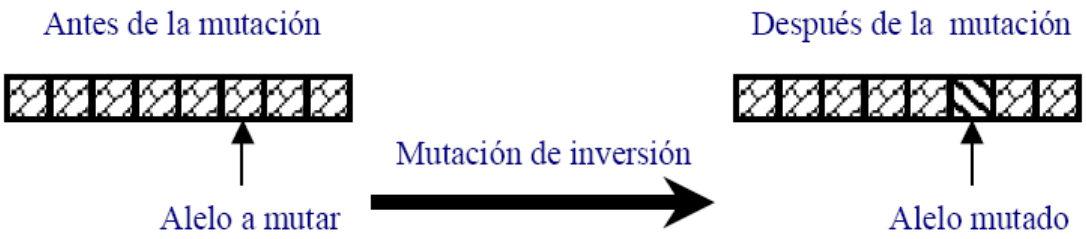


Figura 1.6: Esquema de la mutación de inversión del valor de un alelo.

Tanto la recombinación como la mutación son operadores probabilísticos, en el sentido en que se aplican o no, teniendo en cuenta una tasa de aplicación del operador. Generalmente la tasa de aplicación del operador de cruzamiento es elevada en un algoritmo genético simple (entre 0,5 y 0,9) mientras que la tasa de aplicación del operador de mutación es muy baja, del orden de 0,001 para cada bit en la representación.

Operadores evolutivos más complejos han sido propuestos como alternativas para modificar el comportamiento del mecanismo de exploración del espacio de soluciones. Es habitual encontrar operadores de cruzamiento *multipunto* en donde se utilizan dos o más puntos de corte (la operativa del cruzamiento de dos puntos se ejemplifica en la Figura 1.7) o *uniformes* en donde para cada posición en el cromosoma se decide intercambiar material genético de acuerdo a una probabilidad prefijada (su operativa se ejemplifica en la Figura 1.8).

Figura 1.7: Esquema del cruzamiento de dos puntos.

Figura 1.8: Esquema del cruzamiento uniforme.

1.1.4 Modelos de evolución

En el modelo tradicional de algoritmos genéticos el paso básico de evolución consiste en el reemplazo de la totalidad de la población por sus descendientes en cada generación (*modelo generacional*). Otros modelos diferentes de evolución han sido propuestos por los investigadores en el área. Entre ellos, es posible destacar el modelo *de estado estacionario* (Whitley, 1991). En este paradigma se crea un único nuevo individuo en cada generación, tratando de balancear el compromiso entre exploración (dada por el conjunto de individuos que forman la población) y explotación (realizada por el mecanismo de generación del único individuo creado en cada generación, generalmente utilizando técnicas de elitismo) para la resolución del problema.

Un modelo intermedio entre el tradicional y el de estado estacionario es el *modelo de gap generacional* (De Jong and Sarma, 1993), en el cual un porcentaje de la población (denominado *gap*) se genera en cada paso evolutivo.

Otros modelos de evolución más complejos han sido formulados para los algoritmos genéticos (modelos jerárquicos, no heterogéneos, híbridos, inspirados en los métodos de las estrategias de evolución). Estos modelos se diferencian de los tradicionales, incluyendo complejidades inherentes al mecanismo evolutivo que proponen.

1.1.5 Formalización del mecanismo de los algoritmos genéticos

La teoría tradicional de exploración del espacio de soluciones por parte del mecanismo evolutivo de los algoritmos genéticos fue formalizada por Holland en 1975. Holland definió el concepto de *esquema*, para representar a conjuntos de individuos con características comunes de codificación y enunció el *teorema de los esquemas* para formalizar el muestreo de hiperplanos del espacio de soluciones. Como consecuencia de este teorema se propuso la denominada *hipótesis de los building blocks* que fundamenta el éxito del mecanismo de exploración de los algoritmos genéticos en el número exponencial de muestras que reciben aquellos esquemas cortos, con pocas posiciones definidas, pero que tienen un valor promedio de fitness superior al del resto de esquemas de la población. El mecanismo de recombinación, fundamental en el proceso evolutivo de un algoritmo genético, tenderá a crear individuos cada vez más aptos, perpetuando aquellas características adecuadas y combinándolas con otras de similar calidad (Holland, 1975). Trabajos teóricos posteriores han complementado las estimaciones originales de Holland (Whitley, 2002), e inclusive se han presentado formalizaciones matemáticas exactas del mecanismo (Poloni, 1999), aunque también ofrecido puntos de vista alternativos y críticos a la *hipótesis de los building blocks* (Thornton, 1998).

En (Goldberg et al., 1991) se demostró que el tiempo que un algoritmo genético emplea en converger hacia una solución única depende del tamaño de la población considerada. Utilizando los mecanismos de selección adecuados –aquellos que favorecen la

reproducción de los mejores individuos–, el tiempo de convergencia del algoritmo genético es del orden de O(*n log n*), siendo *n* el tamaño de la población utilizada.

Adicionalmente a sus estudios sobre la formación de bloques básicos de construcción de soluciones, Holland se percató que el mecanismo evolutivo de los algoritmos genéticos tiene una característica denominada *paralelismo intrínseco*, mediante la cual el ciclo evolutivo procesa un número mayor de esquemas que la cantidad de individuos presentes en la población. En efecto, trabajando sobre una población de *n* individuos, y por tanto requiriendo un esfuerzo computacional para realizar solamente *n* evaluaciones de la función de fitness, el algoritmo genético procesa útilmente O (n^3) esquemas en cada generación. Por este motivo, al examinar un elevado número de secciones del espacio de soluciones, el mecanismo de los algoritmos genéticos logra compensar la disrupción de esquemas de mayor largo y de alto número de posiciones definidas cuando se aplican los operadores evolutivos de cruzamiento y mutación.

1.1.6 ¿Cuándo se pueden aplicar los Algoritmos Genéticos?

En primer lugar, en el problema a resolver la meta ha de poder ser observada en grados cualitativamente comparables. Por otra parte, las principales dificultades a la hora de implementar un algoritmo genético son:

- Definir una estructura de datos que pueda contener patrones que representen, la solución óptima buscada (desconocida) y todas las posibles alternativas de aproximaciones a la solución.

- Definir un tipo de patrón tal que si un patrón es seleccionado positivamente, que esto no sea debido a la interacción de los distintos segmentos del patrón, sino que existan segmentos que por sí solos provocan una selección positiva.

- Definir una función de evaluación que seleccione los mejores individuos.

Las condiciones que debe cumplir un problema para ser abordable por algoritmos evolutivos son:

- El espacio de búsqueda deber ser acotado.

- Debe existir un procedimiento relativamente rápido que asigne un grado de utilidad a cada solución propuesta, de forma que este grado de utilidad asignado corresponda, o bien directamente con la calidad de la solución en cuanto al problema a resolver, o bien con un valor de calidad relativo al resto de la población que permita obtener en el futuro mejores soluciones.

- Debe existir un método de codificación de soluciones que admita la posibilidad de que los cruzamientos combinen las características positivas de ambos progenitores. Este método debe permitir también aplicar algún mecanismo de mutación que sea capaz de conseguir tanto soluciones muy dispares respecto de la solución sin mutar, como muy parecidas a ésta.

1.1.7 Comparación de los algoritmos genéticos con otras técnicas heurísticas

Si bien no existe una opinión unánime sobre la aplicabilidad de los algoritmos genéticos, la amplia gama de problemas resueltos exitosamente en diversas áreas han popularizado a esta técnica evolutiva como una de las más versátiles y robustas. Las opiniones de los investigadores en el área concuerdan en señalar a los algoritmos genéticos como altamente útiles para resolver problemas con espacio de soluciones de dimensión elevada o no muy bien comprendido de antemano, donde el diseño de operadores específicos de *hill-climbing* no sea sencillo, en problemas multimodales donde los métodos heurísticos tradicionales pueden quedar atrapados en óptimos locales o en casos donde no es necesario obtener una solución óptima al problema, sino que una buena solución aproximada sería suficiente.

Los algoritmos genéticos presentan ciertas características que los hacen ventajosos cuando se los compara con otras técnicas de resolución de problemas. La operativa del mecanismo evolutivo de los algoritmos genéticos es independiente de particularidades del dominio, permitiendo definir esquemas genéricos capaces de abordar diferentes clases de

problemas. Esta característica, conjuntamente con el hecho de que el mecanismo evolutivo se aplique sobre representaciones, hace a los algoritmos genéticos aplicables a una amplia gama de problemas.

Además, los algoritmos genéticos no son sensibles a efectos de no linearidad de la función a optimizar o características del problema que afectan a algoritmos estándar de optimización que utilizan información adicional, como las técnicas de hill-climbing o algoritmos que asumen aspectos de linearidad, convexidad, diferenciabilidad u otras características sobre la función a optimizar. Estas características brindan a los algoritmos genéticos una significativa robustez de funcionamiento y de aplicabilidad.

Desde el punto de vista de la resolución del problema, el algoritmo genético funciona como una caja negra que solamente utiliza como dato de entrada la función de fitness definida para el problema para evaluar a los individuos en el ciclo evolutivo. No utiliza información adicional sobre las características del problema, aunque ciertas particularidades de la codificación pueden complementar la operativa simple de los operadores evolutivos y dirigir la búsqueda. La independencia del mecanismo de búsqueda evolutivo de las características del problema permite a los algoritmos genéticos evitar, en ocasiones, el estancamiento en mínimos locales del problema en los cuales son proclives a caer ciertos algoritmos tradicionales basados en gradientes u otras búsquedas locales dirigidas.

Una descripción detallada del mecanismo operativo de los algoritmos genéticos, que complemente los breves detalles que se resumen en este capítulo, puede encontrarse en los textos de Goldberg (Goldberg, 1989) o Mitchell (Mitchell, 1996).

1.1.8 Herramientas y bibliotecas evolutivas

La gran expansión que está sufriendo la Computación Evolutiva indica que las herramientas que se utilizan para el desarrollo deben ser cada día más flexibles y potentes, permitiendo a los investigadores construir cualquier tipo de estructura de datos que se desee implementar. Con esta flexibilidad, se facilita el hecho de que un investigador trabaje siempre con la misma herramienta aunque el problema que estén

resolviendo en cada momento sea totalmente diferente al anterior, ahorrando tiempo de comprensión de nuevas herramientas (Arenas et al.) y permitiendo la reutilización del trabajo ya terminado.

Aunque Java está considerado como un Lenguaje que cubre todos los ámbitos no existe ninguna biblioteca comparable en todos los puntos a *Evolving Objects* (EO) (Merelo et al., 2000) que tiene como principal norma de diseño la capacidad de poder hacer evolucionar cualquier tipo de estructura de datos (Merelo et al.) o a GAlib (Wall, 1995) ambas desarrolladas en su totalidad en C++.

Existen herramientas como pueden ser JGAP (Rotstan and Meffert, 2008) (*Java Genetic Algorithms Package*) y JDEAL (Costa et al.), *(Java Distributed Evolutionary Algorithms Library)*, que incluye muchas de las características de EO, incluyendo distribución de código y muchos tipos diferentes de cromosomas y operadores. Sin embargo, JDEAL no presenta la generalidad que EO posee, en el sentido de que sólo ofrece dos tipos de operadores: mutación y cruce y no parece estar diseñadas para ser totalmente ampliable puesto que su diseño está totalmente basado en clases no en interfaces. Esto limita al usuario la posibilidad de que el cromosoma o individuo que él necesita pueda ser, a la vez, de cromosoma y de operador, puesto que en Java no está permitido la herencia múltiple. Sería totalmente imposible que una misma clase heredara de algún tipo de cromosoma y algún tipo de operador a la vez. Pero no estaría de más decir que esta característica es simplemente una limitación, no un defecto.

Bastante recomendable en el caso que nos concierne, es utilizar una herramienta como ECJ. De este modo separaríamos claramente la parte del experimento de los detalles de implementación para la programación genética.

1.1.9 ECJ Características Generales

ECJ siglas en ingles para *A Java-based Evolutionary Computation and Genetic Programming Research System* (Luke et al.) es una plataforma de desarrollo de algoritmos evolutivos desarrollada en el Laboratorio de Computación Evolutiva ECLab

de la George Mason University (ECLab, 2010), casi todas sus clases y parámetros se determinan en tiempo de ejecución mediante archivos de parámetros jerárquicos.

Diseñada para ser flexible, hereda todas las ventajas de Java, como multiplataforma, multihilo, manejo de excepciones, *garbage collection*, incluye funciones de logging y checkpoint independientes para varias técnicas evolutivas. Brinda compatibilidad con Algoritmos Genéticos y Programación Genética de estado estable y/o generacional, con o sin elitismo para evolución de tipo: *mu*, *lambda* y *mu+lambda* sobre una arquitectura de reproducción (*breeding*) bastante flexible y modelos de islas asincrónicos sobre TCP/IP. Es capaz de manejar gran número de operadores de selección para múltiples subpoblaciones y especies, permitiendo intercambios entre subpoblaciones de individuos con genes de longitud fija o variable y soporta coevolución de población simple y múltiple, serialización de poblaciones en archivos persistentes y optimización multiobjetivo SPEA2, facilitando la posibilidad de extender la librería para otros métodos de optimización multiobjetivo.

1.2 Estrategias QSPR

Lo modelos que relacionan la propiedad y la estructura cuantitativa (QSPR) describen una relación matemática entre las características estructurales y una propiedad de una serie de compuestos químicos. El uso de tales relaciones matemáticas para predecir la propiedad objeto de interés en una sustancia química resulta seductor en lugar de mediciones experimentales laboriosas que conllevan un gran consumo de tiempo y recursos materiales muy costosos.

Los estudios QSPR constituyen un enfoque que permite entender como la variación estructural afecta la propiedad de un conjunto de compuestos. En estos estudios, los descriptores moleculares (X) se correlacionan con una variable respuesta (Y). Es decir, este análisis puede definirse como una aplicación de métodos matemáticos y estadísticos al problema de encontrar una ecuación empírica de la forma $Y_i=f_i(X_1, X_2,\ldots X_N)$, donde Y_i son las propiedades de la molécula y $X_1, X_2\ldots X_N$, son propiedades estructurales experimentales o calculadas (descriptores moleculares) de los compuestos. En este

sentido, cada compuesto puede representarse como un punto en un espacio multidimensional, en los cuales los descriptores X_1, $X_2 \ldots X_N$ son coordenadas independientes del compuesto. El objetivo más usual de este análisis es predecir la propiedad estudiada a un objetivo (compuesto) no utilizado en la obtención del modelo.

La principal característica de estas estrategias es el enfoque multivariado al problema, la búsqueda de información relevante, la validación de los modelos para generar modelos con poder predictivo, comparación de los resultados obtenidos por diferentes métodos, y la definición y el uso de índices capaces de medirla calidad de la información extraída. Esta es la herramienta más usada en los estudios QSPR ya que brinda una sólida base para el análisis y la modelación de datos proporcionando una batería de diferentes métodos para este fin. Varias técnicas estadísticas fueron aplicadas para predecir propiedades, usando descriptores moleculares, pero las más utilizadas son el análisis de Cluster, la regresión lineal múltiple y las redes neuronales (Chen, 2008).

Un aspecto medular de esta rama es la atención que se le presta al poder predictivo del modelo, su complejidad y calidad. Así que, para que un estudio QSPR sea adecuado a los fines regulatorios dados debe cumplir con las siguientes condiciones fundamentales: (1) Estimar por un modelo validado; (2) Aplicar el modelo al interés químico, con suficiente confiabilidad y (3) el punto final del modelo debe ser relevante según las regulatorios reportadas para estos fines (Worth, 2005). Con el objetivo de unificar estos criterios en todas las investigaciones del tema surgen los principios de la Organización para la Económica, Cooperación y el Desarrollo (OECD, siglas en inglés), que por su gran interés fueron considerados durante el desarrollo del trabajo. Estos principios de validación de la OECD (OECD, 2004, OECD, 2007), se crean para facilitar las consideraciones de los modelos obtenidos y están asociados con la siguiente información:

1. Propósito final definido
2. Algoritmo no ambiguo
3. Dominio de aplicabilidad definido

4. Mediciones apropiadas de bondad de ajuste, robustez y predictividad.

5. Interpretación mecanicista

1.2.1 Metodología general empleada en estudios QSPR/QSAR

Los modelos QSPR para ser confiables y predictivos, deben: (1) ser significativos estadísticamente y robustos, (2) ser validados por marcar las predicciones exactas para la data externa que no es usada en el desarrollo de modelos, y (3) tienen su límite definido de aplicación (Tropsha, 2003). Solo los modelos QSPR validados pueden ofrecer una interpretación significativa, especialmente en el contexto de diseñar o descubrir nuevos agentes químicos con propiedad deseada. Los principios de la metodología QSAR/QSPR puede describirse mediante los siguientes pasos comunes (Van de Waterbeemd, 1995):

1) Formulación del problema, se determina el objeto de análisis y nivel de información requerido

2) Parametrización cuantitativa de la estructura molecular de los compuestos químicos orgánicos

3) Medición de la propiedad de interés (efectos biológicos, índices de retención, toxicidad)

4) Escoger el tipo de modelo QSAR/QSPR que se va a desarrollar

5) Selección de los compuestos

6) Análisis matemático de los datos y validación interna y externa de los modelos obtenidos

7) Definición del dominio de aplicación del modelo obtenido.

1.2.2 Validación de modelos

Independientemente del tipo de análisis QSAR empleado el resultado final va a ser un modelo matemático. Para estar completamente seguros de que el modelo obtenido no resulta de un ordenamiento a azar de los datos, es común verificar la calidad del mismo según las normativas internacionales.

Otro problema es que el modelo no sea predictivo, es decir que la función propuesta no se cumpla para productos no incluidos en la serie de exploración. Esta problemática suele solucionarse realizando una validación cruzada (*crossvalidation*) con el procedimiento conocido como *leave-one-out* al modelo, generando los valores promedios de los coeficientes de correlación (r^2 ó r_{cross}) y de las desviaciones estándar obtenidos (S_{press}).

Para que el estudio de relaciones estructura-actividad se concluya con éxito debe obtenerse un modelo significativo y predictivo. Aún así, la validez del modelo estará siempre limitada al rango de parámetros explorados por la serie de exploración, fuera del cual nunca debe considerarse válido. De hecho, esta es la principal limitación a la hora de buscar el óptimo de actividad predicho por un modelo.

1.2.3 Estimación de modelos QSAR con Aprendizaje Automático

Los Métodos de clasificación se utilizan para la asignación de objetos a una de varias clases basado en una regla de clasificación. Son métodos de aprendizaje supervisado, pues se "aprende" a partir de una serie de casos con variables predictivas y función objetivo o variable dependiente como también es conocida. Tal es el caso también del clásico método de regresión (para valores numéricos); solo que en los problemas de clasificación la función objetivo es discreta (nominal u ordinal). El objetivo de tales técnicas es calcular una regla de clasificación (y posiblemente, límites de clases, o probabilidades de pertenencia a una clase), basada en los objetos de la serie de entrenamiento y aplicar esta regla para asignar una de estas clases, a objetos de clases previamente desconocidas (Hand, 1981, Frank and Friedman, 1989). Los métodos de clasificación son apropiados para modelar varias respuestas QSAR, como por ejemplo:

compuestos activos/no-activos, compuestos de toxicidad baja/mediana/alta, compuestos mutagénicos/no mutagénicos, etc.

Los métodos de clasificación de origen estadístico más populares son: Análisis Discriminante Lineal o LDA, de sus siglas en inglés, *Linear Discriminant Análisis*; Análisis Discriminante Cuadrático o QDA, de sus siglas en inglés, *Quadratic Discriminant Analysis*; Analisis Discriminante Regularizado o RDA, de sus siglas en inglés, *Regularized Discriminant Analysis*; *K*-ésimos Vecinos más Cercanos o KNN, de sus siglas en inglés, K^{th} *Nearest Neighbours*; Métodos de Árboles de Clasificación o CTM, de sus siglas en inglés, *Classification Tree Methods* (también conocidos en la literatura como DT, acrónimo de *Decision Trees*); *Regression Decision Trees* potentes árboles de decisión cuyas hojas finales no definen una clase para un segmento de la población, sino una ecuación de regresión, por ejemplo logística para clasificar a los elementos de ese segmento, los Clasificadores de Función Potencial o PFC, de sus siglas en inglés, *Potencial Function Clasifiers*; los Clasificadores Medios más Cercanos o NMC, de sus siglas en inglés, *Nearest Mean Classifiers* y los Clasificadores Medios más Cercanos Ponderados o WNMC, de sus siglas en inglés, *Weighted Nearest Mean Classifiers*.

La calidad de los modelos de clasificación se evalúa por los parámetros de clasificación, para ambos, propósitos de ajuste y predicción (Todeschini, 2000). Existe toda una teoría, y técnicas concretas para el análisis y la comparación de diferentes modelos de clasificación ante problemas específicos (Vanhoof, 2006).

Además de las técnicas de aprendizaje supervisado, se utilizan técnicas de aprendizaje no supervisado, en la cual se conforman "clases" a partir de variables predictivas conocidas sin que haya un conocimiento previo sobre la clase a que pertenece cada instancia, a veces, ni siquiera, sobre el número de clases a considerar. Se trata de "agrupar" o más bien "separar en grupos", instancias similares entre sí y diferentes de otros grupos. Las técnicas estadísticas clásicas fueron denominadas técnicas de detección de "agrupamientos" o de "conglomerados", en inglés, *clustering*.

Presentamos un conocido software para aprendizaje automático y minería de datos escrito en Java (Witten and Frank, 2000), que concentra todos los métodos de clasificación que usaremos para generar los modelos QSAR: Weka (Waikato Environment for Knowledge Analysis - Entorno para Análisis del Conocimiento) desarrollado en la Universidad de Waikato en Nueva Zelanda. WEKA es un software libre distribuido bajo licencia GNU-GPL, que contiene una extensa colección de técnicas para preprocesamiento de datos y modelado.

Weka soporta varias tareas estándar de minería de datos, especialmente, preprocesamiento de datos, clustering, clasificación, regresión, visualización, y selección. Todas las técnicas de Weka se fundamentan en la asunción de que los datos están disponibles en un fichero plano (*flat file*) o una relación, en la que cada registro de datos está descrito por un número fijo de atributos (normalmente numéricos o nominales, aunque también se soportan otros tipos).

Entre las razones a favor de Weka, se destaca que es muy portable porque está completamente implementado en Java y puede correr en casi cualquier plataforma, que contiene una extensa colección de técnicas para preprocesamiento de datos y modelado y sobre todo es muy fácil de utilizar por un principiante gracias a su interfaz gráfica de usuario.

1.3 Algoritmos genéticos en la química computacional

Adentrándonos en la química computacional, nos encontramos con un número considerable de aplicaciones, que en su mayoría presentan un espacio de búsqueda de tamaño exponencial directamente proporcional a las dimensiones del problema, por tanto estos problemas no pueden ser resueltos por métodos tradicionales de búsqueda exhaustiva. Paradójicamente los algoritmos genéticos (AG) presentan una adecuada efectividad en los espacios de búsqueda considerados "grandes". En los últimos años la aplicación de los AG en estos complejos problemas se ha convertido en un amplio tema de desarrollo y existe una creciente colección de publicaciones que así lo demuestran. Como podemos apreciar en Clark and Gestead (Clark and Westhead, 1996), con el diseño

de moléculas auxiliado por computadoras (*computer-aided molecule design*), Willett (Willett, 1995) con el reconocimiento de moléculas, Parrill (Parrill, 1996) en el descubrimiento de fármacos, Pedersen and Moult (Pedersen and Moult, 1996) con la predicción de las estructuras en proteínas, finalmente podemos encontrar en Devillers (Devillers, 1996b) un estudio general de las aplicaciones sobre el modelado de moléculas, entre otros.

1.3.1 Diseño Molecular

El diseño y la generación de moléculas con cualidades específicas constituyen una actividad muy importante en las investigaciones tanto químicas, como farmacéuticas. El diseño de moléculas guiado por computadoras ofrece una atractiva alternativa comparado con los métodos tradicionales.

Dando lugar a esta investigación los AG han mostrado una fuente fiable para este campo de estudio que prácticamente recién comienza, por ejemplo (Venkatasubramanian et al., 1994, Venkatasubramanian et al., 1995b, Venkatasubramanian et al., 1995a) ha desarrollado un AG para el diseño de moléculas, las cuales son codificadas como cadenas usando un alfabeto simbólico de grupos químicos y luego optimizadas de acuerdo a las características físicas deseadas.

Por otra parte Glen and Payne (Glen and A. W. R. Payne, 1995) diseñaron un AG para la generación de moléculas mediante restricciones, donde los individuos se codifican como estructuras en tres dimensiones (3D) y el operador de cruzamiento (*crossover*) realiza el intercambio de sub-estructuras entre las estructuras padres. La mutación puede tener varios efectos como son: mueve la estructura, cambia la conformación molecular o altera la composición química de la estructura.

Hasta ahora se han expuesto dos estrategias que garantizan la convergencia a una solución óptima aproximada mediante AG, como se puede apreciar desde un primer enfoque hacia las técnicas de que abarcan los algoritmos evolutivos, los AG muestran superioridad a la hora de incursionar en el amplio espacio de búsqueda de la química computacional, vale la pena mencionar que solamente estamos abordando el diseño de

moléculas. Estas conclusiones condujeron varias investigaciones de avanzada que incorporaran exclusivas características, podemos mencionar algunas, como por ejemplo el trabajo realizado por Devillers en (Devillers, 1996a) donde utiliza un sistema hibrido de comunicación entre un AG y una red neuronal, pero deseamos hacer mayor énfasis en una de las estrategias lideres en el diseño de moléculas, la que también forma parte fundamental como base de esta investigación, sin dudas nos referimos a DND, abreviaturas en inglés para "*de novo design*" y al trabajo realizado por Westhead et al (Westhead et al., 1995) para el refinamiento de estructuras usando AG y DND. Nuevamente la codificación utilizada es la estructura química en sí y la población inicial consiste en un número de componentes producidos por un programa DND. Como se ha podido apreciar el diseño de moléculas con rasgos fijados mediante algoritmos evolutivos es un objetivo que perseguimos.

1.4 Consideraciones finales.

Los métodos que emplean técnicas asistidas por ordenador usados en el descubrimiento, diseño, y optimización de compuestos con estructura y propiedades deseadas han desempeñado un rol importante en el desarrollo de fármacos que se encuentran actualmente en el mercado o en fase de estudios clínicos. Entre estos se destacan los modelos que relacionan cuantitativamente aspectos estructurales y propiedades de las moléculas (*estudios QSAR*). Otra herramienta computacional empleada con éxito es la búsqueda heurística y en específico los algoritmos genéticos.

Numerosas bibliotecas para algoritmos genéticos están disponibles para el desarrollador de nuevas aplicaciones. Entre ellas se destaca ECJ, escrita en el lenguaje de programación Java, de código abierto y con una comunidad de usuarios y desarrolladores amplia y en pleno auge.

2 Modelación de la actividad biológica molecular

En el campo de la química computacional los programas especializados en apoyar las tareas cotidianas de los investigadores generalmente son comerciales y muy costosos, por tanto es poco probable que podamos encontrar los códigos fuentes o versiones de código abierto (en inglés *OpenSource*) (OSI).

Un gran número de bibliotecas escritas en Java están disponibles en forma binaria, pero pocas incluyen acceso a los códigos fuentes y facilidades para extenderlas (Rzepa and Tonge, 1998, Csizmadia, 2000, Blauch, 2002), razones suficientes para orientar este trabajo hacia las necesidades de la comunidad de investigadores, que en muchas ocasiones deben de compartir los resultados entre ellos, actividad imposible utilizando software privativo.

En este capítulo se presenta una poderosa herramienta de software que servirá de plataforma común para representar las estructuras químicas y sus funcionalidades, también proponemos su extensión para incorporarle índices moleculares basados en formas algebraicas

2.1 Descriptores moleculares

2.1.1 Definición

Los índices o descriptores moleculares (DMs) representan la vía por la cual la estructura química se transforma en números permitiendo el tratamiento matemático de la información química contenida en la molécula y pueden definirse como representaciones matemáticas de las moléculas que se obtienen al aplicar algoritmos específicos sobre una representación molecular definida o a partir de procedimientos experimentales específicos.

La utilidad de un DM debe analizarse con doble sentido: el número puede brindar una interpretación más profunda en términos estructurales de la propiedad molecular y/o es capaz de tomar parte en un modelo para la predicción de propiedades moleculares de interés (Todeschini, 2000). Incluso si la interpretación del DM es débil o carente, este podría estar estrechamente correlacionado con algunas propiedades moleculares permitiendo obtener modelos con alta capacidad predictiva. Por otro lado, DMs con baja capacidad predictiva pueden ser mantenidos en el modelo cuando están correctamente fundamentados por la teoría y son interpretables debido a su capacidad para codificar información de la estructura química.

Aunque hasta el momento el número de índices moleculares reportados supera el millar, el desempeño de estos en la predicción de determinadas propiedades no siempre es totalmente satisfactorio. Por esa razón, la definición de DMs se mantiene como un área de intensa actividad en el campo de la química computacional. La aplicación de conceptos de la Teoría de Grafos (TG) en el desarrollo de métodos teóricos para la representación de estructuras químicas ha tenido un enorme impacto, entre las aplicaciones más importantes de la TG a la química se encuentra la caracterización numérica de la estructura molecular a partir de invariantes grafo-teóricas, que se emplean como DMs de compuestos químicos en estudios de estructura-propiedad (QSAR).

La próxima sección introduce un enfoque innovador basado en nociones de la TG y en conceptos básicos de álgebra matricial y vectorial aplicados en formas lineales y cuadráticas.

2.1.2 Descriptores para formas algebraicas

Metodología basada en la caracterización de la estructura molecular por medio de familias de descriptores moleculares, denominados índices lineales y cuadráticos, en analogía a las *formas algebraicas lineale*s y *cuadráticas*, respectivamente.

Descriptores algebraicos “no estocásticos”

En orden de obtener descriptores con estas características, se construye un vector molecular (X), donde sus componentes son valores numéricos de una propiedad que caracteriza cada tipo de átomo presente en la molécula.

Dado una molécula constituida por n átomos (vector de $\Re$ n), los kth índices cuadráticos moleculares, qk(x) son calculados como una forma cuadrática (q: $\Re$ n$\rightarrow$ $\Re$) en las bases canónicas como se muestra en la ecuación 2.1:

$$q_k(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} {}^{k}a_{ij} X_i X_j \qquad (2.1)$$

donde $ka_{ij} = ka_{ji}$ (matriz cuadrada y simétrica), n es el número de átomos de la molécula y $X_1,\ldots,X_n$ son las coordenadas del vector molecular X en un sistema de vectores bases de $\Re$ n. Los valores de las coordenadas del vector dependen de la base escogida (Browder, 1996, Axler, 1996, Maltsev, 1976). En la llamada base canónica (natural) las coordenadas de cualquier vector coinciden con los componentes del vector, por esta razón las coordenadas de los vectores pueden ser consideradas como pesos o etiquetas de átomos.

Los coeficientes ${}^{k}a_{ij}$ son los elementos a_{ij} de la k^{th} potencia de la matriz de adyacencia entre vértices ($\mathbf{M}^{k}$) del seudografo molecular (G), la cual es utilizada como la matriz de la forma con respecto a la base canónica. Luego, $\mathbf{M}$ (G) $\equiv \mathbf{M} = [a_{ij}]$, donde n es el número de vértices, y los elementos a_{ij} se definen como sigue:

$a_{ij} = P_{ij}$ si $i \neq j$ y $\exists$ $e_k \in E / e_k \sim v_i, v_j$ (2.2)

$= L_{ii}$ si $i = j$

= 0 de otra forma

donde E(G) representa el conjunto de las aristas. P_{ij} es el número de aristas entre los vértices v_i y v_j. L_{ij} es el número de lazos en v_i.

Los elementos a_{ij} ($a_{ij} = P_{ij}$) de la matriz M representan los enlaces entre un átomo v_i y otro v_j. La matriz $\mathbf{M}^{k}$ provee el número de *caminos de longitud k* que une los vértices de v_i y v_j.

Por esta razón, cada arista representa dos electrones del enlace covalente entre dos átomos v_i y v_j; y esto puede ser apreciado en las entradas a_{ij} y a_{ji} igual a 1, 2 y 3 de la matriz **M** (k = 1) cuando entre los átomos (vértices) v_i y v_j existe un simple, doble o triple enlace, respectivamente.

Los anillos aromáticos con una sola estructura canónica tales como el furano, tiofeno, pirrol, entre otros, son representados como multigrafos. Además, los $\boldsymbol{q_k}(x)$ pueden ser obtenidos mediante la expresión matricial representada en la ecuación. 2.3:

$$q_k(x) = \begin{bmatrix} X_1 & \Lambda & X_n \end{bmatrix} \begin{bmatrix} a_{11} & \Lambda & a_{1n} \\ \mathrm{M} & & \mathrm{M} \\ a_{n1} & \Lambda & a_{nn} \end{bmatrix}^k \begin{bmatrix} X_1 \\ \mathrm{M} \\ X_n \end{bmatrix} \qquad (2.3)$$

o de forma más sencilla,

$$\boldsymbol{q_k}(x) = [\mathrm{X}]^t \mathbf{M}^{\boldsymbol{k}} [\mathrm{X}] \qquad (2.4)$$

donde [X] es el vector columna (una matriz de nx1) de coordenadas de **X** en la base canónica de $\Re^n$, $[\mathrm{X}]^t$ es la matriz transpuesta de [X] (una matriz de 1xn) y $\mathbf{M}^{\boldsymbol{k}}$ la k^{th} potencia de la matriz **M** de el seudografo molecular G (matriz de la forma cuadrática).

Los índice lineales atómicos son definidos como una transformación lineal $\boldsymbol{f_k}(x_i)$ sobre un espacio vectorial molecular, $\Re^n$. Esta aplicación, es una correspondencia que asigna a cualquier vector **X** en $\Re^n$ un vector $\boldsymbol{f}(x)$ de forma tal que:

$$f(\lambda_1 \mathrm{X}_1 + \lambda_2 \mathrm{X}_2) = \lambda_1 f(\mathrm{X}_1) + \lambda_2 f(\mathrm{X}_2) \qquad (2.5)$$

Para todo λ_1,λ_2 número reales y cualquier vector X_1, X_2 en $\Re^n$. En otras palabras, $\boldsymbol{f_k}(x_i)$ es una aplicación lineal dado que la imagen de la combinación lineal de dos vectores X_1 y X_2, $\lambda_1\mathrm{X}_1 + \lambda_2\mathrm{X}_2$ es igual a la combinación lineal de las imágenes $f(\mathrm{X}_1)$ y $f(\mathrm{X}_2)$, $\lambda_1 f(\mathrm{X}_1) + \lambda_2 f(\mathrm{X}_2)$. Esta condición se denomina *condición de linealidad* (Browder, 1996, Maltsev, 1976). La ecuación de definición (2.5) para $\boldsymbol{f_k}(x_i)$ puede ser escrita como una simple ecuación matricial:

$$f_k(x_i) = \begin{bmatrix} X'_1 \\ \mathrm{M} \\ X'_n \end{bmatrix}^k = \begin{bmatrix} a_{11} & \Lambda & a_{1n} \\ \mathrm{M} & & \mathrm{M} \\ a_{n1} & \Lambda & a_{nn} \end{bmatrix}^k \begin{bmatrix} X_1 \\ \mathrm{M} \\ X_n \end{bmatrix} \qquad (2.6)$$

o de forma más sencilla,

$$\boldsymbol{f_k}(x_i) = [\mathrm{X'}]^k = \mathbf{M}^{\boldsymbol{k}}[\mathrm{X}] \qquad (2.7)$$

Donde [X] es el vector columna (una matriz de *n*x1) de coordenadas de X en la base canónica de $\Re^n$, $[X]^t$ es la matriz transpuesta de [X] (una matriz de 1x*n*) y M^k la k^{th} potencia de la matriz M del seudografo molecular G (Matriz de la aplicación lineal).

Este enfoque es similar al método **LCAO-MO,** siglas acrónimas de **L**inear **C**ombinations of **A**tomic **O**rbitals-**M**olecular **O**rbitals. Realmente nuestro enfoque (para $k = 1$) es una aproximación muy similar al método de Hückel extendido, dado que nuestra matriz considera tanto electrones sigma como pi.

Los índices lineales totales constituyen funciones lineales (algunos matemáticos usan el término *formas lineales*) sobre $\Re^n$ (Browder, 1996, Axler, 1996, Maltsev, 1976). Es decir, los k^{th} índices lineales totales constituyen aplicaciones lineales de $\Re^n$ a escalares $\Re$ [$f_k(x)$: $\Re^n \rightarrow \Re$].

Descriptores algebraicos "estocásticos"

Los descriptores TOMOCOMD-CARDD pueden ser calculados en términos de probabilidades si utilizamos como matriz de las aplicaciones las matrices estocásticas de cada una de la potencias de la matriz de adyacencia entre átomos del seudografo molecular. Estos descriptores se han denominados como índices cuadráticos y lineales estocásticos, y presentan las mismas propiedades descritas para sus homólogos no estocásticos. Es decir, los *k-ésimos* descriptores moleculares estocásticos totales se calculan según las mismas técnicas definidas anteriormente (ver ecuaciones 2.4 y 2.7), pero usando la matriz estocástica de adyacencia entre átomos del pseudografo molecular, $\mathbf{S}^k$(G), como matriz de las formas.

Las matrices estocásticas $\mathbf{S}^k$(G) pueden ser obtenidas dividiendo cada elemento ${}^k a_{ij}$ de $\mathbf{M}^k$(G) (electrones que comparte el átomo '*i*' con el átomo '*j*') entre el número de electrones que el átomo '*i*' comparte con todos los demás átomos en la molécula a *k*-distancias, incluido el mismo.

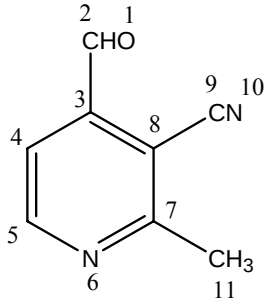
Los elementos ${}^k\mathbf{s}_{ij}$ se definen como se muestra en la ecuación 2.8:

$$ {}^k s_{ij} = \frac{{}^k a_{ij}}{{}^k SUM_i} = \frac{{}^k a_{ij}}{{}^k \delta_i} \qquad (2.8) $$

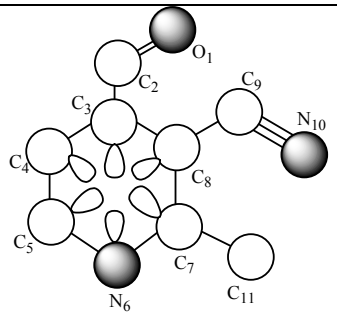
donde ${}^k a_{ij}$ son los elementos de la *k*-ésima potencia de **M**, y ${}^k\mathrm{SUM}_i$ es la suma de la fila *i-ésima* de $\mathbf{M}^k$ o grado del vértice de orden *k* del átomo *i*, ${}^k\delta_i$. Esta transformación normaliza cada fila de la matriz original [las matrices con estas propiedades y con elementos no negativos se denominan estocásticas (Edwards and Penney, 1988)] y por tanto, sus *k*-ésimos elementos constituyen las probabilidades de transición con las cuales un electrón se mueve de un átomo *i* a otro *j* en un período de tiempo discreto t_k..

En la columna derecha de la Tabla 2.1 se muestra, a modo de ejemplo, las matrices estocásticas de orden 0-2 para la molécula del 2-formil-6-metilbenzonitrilo. Nótese que los *k*-ésimos elementos $\boldsymbol{s}_{ij}$ toman en consideración la información de la topología molecular en *k* pasos a través de todo el esqueleto covalente. Así por ejemplo, los valores de ${}^2\mathbf{s}_{ij}$ pueden distinguir entre las diferentes formas híbridas de cada átomo. En este sentido, en la Tabla 3.2 (columna derecha) puede observarse que los electrones tienen una mayor probabilidad de regresar a un átomo de nitrógeno sp [$p(\mathrm{N}_{10}) = 0.75$] que a un átomo de nitrógeno sp^2 [$p(\mathrm{N}_6) = 0.33$] en t_2 ($k = 2$). Un comportamiento similar puede observarse entre los diferentes "estados híbridos" de los átomos de carbono en la molécula 2-formil-6-metilbenzonitrilo (ver Tabla 3.2): Csp^3 [$p(\mathrm{C}_{11}) = 0.25$]; Csp^2 [$p(\mathrm{C}_2) = 0.625$]; Csp^2_{arom} [$p(\mathrm{C}_3) = 0.285$, $p(\mathrm{C}_4) = 0.3$, $p(\mathrm{C}_5) = 0.33$, $p(\mathrm{C}_7) = 0.33$, $p(\mathrm{C}_8) = 0.25$]; y Csp [$p(\mathrm{C}_9) = 0.769$]. Esto es un resultado lógico si tomamos en cuenta las propiedades electrónicas (por ejemplo su escala de electronegatividad) de cada una de las diferentes hibridaciones de estos átomos.

Tabla 2.1. Cálculo de $\mathbf{M}^k$(G) y $\mathbf{S}^k$(G) para la Molécula del 2-formil-6-metilbenzonitrilo Cuando *k* varía entre 0 y 2.



Estructura Molecular



Pseudografo Molecular (G)

a_{ij}	O_1	C_2	C_3	C_4	C_5	N_6	C_7	C_8	C_9	N_{10}	C_{11}	${}^k\delta_i$	O_1	C_2	C_3	C_4	C_5	N_6	C_7	C_8	C_9	N_{10}	C_{11}
	$\mathbf{M}^0$(G)												$\mathbf{S}^0$(G)										
O_1	1	0	0	0	0	0	0	0	0	0	0	*1*	1	0	0	0	0	0	0	0	0	0	0
C_2	0	1	0	0	0	0	0	0	0	0	0	*1*	0	1	0	0	0	0	0	0	0	0	0
C_3	0	0	1	0	0	0	0	0	0	0	0	*1*	0	0	1	0	0	0	0	0	0	0	0
C_4	0	0	0	1	0	0	0	0	0	0	0	*1*	0	0	0	1	0	0	0	0	0	0	0
C_5	0	0	0	0	1	0	0	0	0	0	0	*1*	0	0	0	0	1	0	0	0	0	0	0
N_6	0	0	0	0	0	1	0	0	0	0	0	*1*	0	0	0	0	0	1	0	0	0	0	0
C_7	0	0	0	0	0	0	1	0	0	0	0	*1*	0	0	0	0	0	0	1	0	0	0	0
C_8	0	0	0	0	0	0	0	1	0	0	0	*1*	0	0	0	0	0	0	0	1	0	0	0
C_9	0	0	0	0	0	0	0	0	1	0	0	*1*	0	0	0	0	0	0	0	0	1	0	0
N_{10}	0	0	0	0	0	0	0	0	0	1	0	*1*	0	0	0	0	0	0	0	0	0	1	0
C_{11}	0	0	0	0	0	0	0	0	0	0	1	*1*	0	0	0	0	0	0	0	0	0	0	1
	$\mathbf{M}^1$(G)												$\mathbf{S}^1$(G)										
O_1	0	2	0	0	0	0	0	0	0	0	0	*2*	0	1	0	0	0	0	0	0	0	0	0
C_2	2	0	1	0	0	0	0	0	0	0	0	*3*	0.66	0	0.33	0	0	0	0	0	0	0	0
C_3	0	1	1	1	0	0	0	1	0	0	0	*4*	0	0.25	0.25	0.25	0	0	0	0.25	0	0	0
C_4	0	0	1	1	1	0	0	0	0	0	0	*3*	0	0	0.33	0.33	0.33	0	0	0	0	0	0
C_5	0	0	0	1	1	1	0	0	0	0	0	*3*	0	0	0	0.33	0.33	0.33	0	0	0	0	0
N_6	0	0	0	0	1	1	1	0	0	0	0	*3*	0	0	0	0	0.33	0.33	0.33	0	0	0	0
C_7	0	0	0	0	0	1	1	1	0	0	1	*4*	0	0	0	0	0	0.25	0.25	0.25	0	0	0.25
C_8	0	0	1	0	0	0	1	1	1	0	0	*4*	0	0	0.25	0	0	0	0.25	0.25	0.25	0	0
C_9	0	0	0	0	0	0	0	1	0	3	0	*4*	0	0	0	0	0	0	0	0.25	0	0.75	0
N_{10}	0	0	0	0	0	0	0	0	3	0	0	*3*	0	0	0	0	0	0	0	0	1	0	0
C_{11}	0	0	0	0	0	0	1	0	0	0	0	*1*	0	0	0	0	0	0	1	0	0	0	0
	$\mathbf{M}^2$(G)												$\mathbf{S}^2$(G)										
O_1	4	0	2	0	0	0	0	0	0	0	0	*6*	0.66	0	0.33	0	0	0	0	0	0	0	0
C_2	0	5	1	1	0	0	0	1	0	0	0	*8*	0	0.625	0.125	0.125	0	0	0	0.125	0	0	0
C_3	2	1	4	2	1	0	1	2	1	0	0	*14*	0.143	0.071	0.287	0.143	0.071	0	0.071	0.143	0.071	0	0
C_4	0	1	2	3	2	1	0	1	0	0	0	*10*	0	0.1	0.2	0.3	0.2	0.1	0	0.1	0	0	0
C_5	0	0	1	2	3	2	1	0	0	0	0	*9*	0	0	0.111	0.222	0.333	0.222	0.111	0	0	0	0
N_6	0	0	0	1	2	3	2	1	0	0	1	*10*	0	0	0	0.1	0.2	0.3	0.2	0.1	0	0	0.1
C_7	0	0	1	0	1	2	4	2	1	0	1	*12*	0	0	0.083	0	0.083	0.166	0.333	0.166	0.083	0	0.083
C_8	0	1	2	1	0	1	2	4	1	3	1	*16*	0	0.063	0.125	0.063	0	0.063	0.125	0.25	0.063	0.188	0.063

C_9	0	0	1	0	0	0	1	1	10	0	0	*13*	0	0	0.077	0	0	0	0.077	0.077	0.769	0	0
N_{10}	0	0	0	0	0	0	0	3	0	9	0	*12*	0	0	0	0	0	0	0	0.25	0	0.75	0
C_{11}	0	0	0	0	0	1	1	1	0	0	1	*4*	0	0	0	0	0	0.25	0.25	0.25	0	0	0.25

2.2 Implementación de los descriptores algebraicos

Partimos de una biblioteca gratuita, de código abierto y disponible en Java, orientada fundamentalmente para la química y la bioinformática, nos referimos a CDK (acrónimo del inglés *Chemistry Development Kit*) (Steinbeck, 2003), para representar tanto el comportamiento de las estructuras moleculares, como las transformaciones que se aplicarán sobre ellas.

Los códigos fuentes en Java están organizados mediante colecciones de clases nombradas, denominadas paquetes, estos por lo general presentan un estilo propio en sus nombres que se corresponde, en cierta forma, con direcciones de Internet invertidas (Steinbeck, 2003). Desde que la biblioteca CDK forma parte del proyecto OpenScience (OpenScienceProject) mantiene una disposición de paquetes para sus códigos fuentes encabezada por *org.openscience.cdk.*

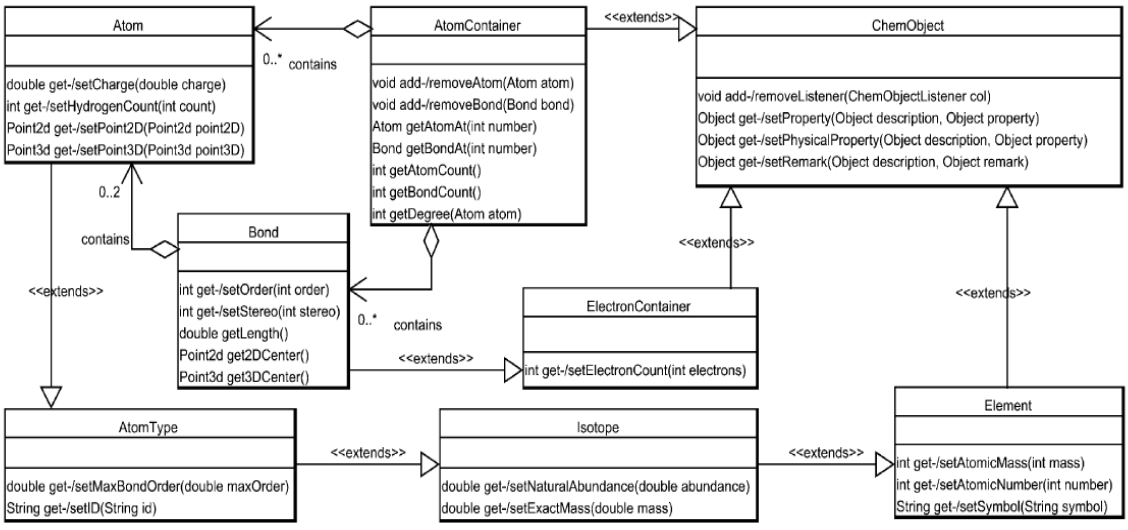


Figura 2.1 Diagrama UML con las principales dependencias entre clases en CDK

Las clases contenidas en la sección raíz de la jerarquía de paquetes son representaciones formales de conceptos básicos de química, como átomos, enlaces, moléculas, etc. La figura 2.1 muestra un diagrama en UML con las relaciones y dependencias entre las clases fundamentales de CDK (Steinbeck, 2003), donde podemos apreciar la clase *ChemObject* que se presenta como la superclase medular. Este diseño es lógico desde el punto de vista químico y sirve como base de un mecanismo simple para la creación de átomos y moléculas.

Las clases orientadas al cálculo de índices moleculares están agrupadas en el paquete *org.openscience.cdk.qsar.descriptors.molecular* y poseen entradas de especificación definidas mediante OWL (acrónimo en inglés para *Web Ontology Language*) un lenguaje de representación de conocimiento (Bechhofer et al., 2009), ubicadas en el fichero *descriptor-algorithms.owl* dentro del paquete *org.openscience.cdk.dict.data* con el objetivo de coordinar un identificador unívoco para cada descriptor implementado.

Para desarrollar la implementación de los descriptores algebraicos en el entorno de trabajo de CDK se propone hacer uso de la flexibilidad que nos brinda el lenguaje Java en los espacios de nombres de las clases, para incorporar nuevas clases en paquetes ya existentes, mediante archivos independientes.

Para añadir un descriptor algebraico que cumpla con los requisitos expuestos tenemos que crear en el paquete *org.openscience.cdk.qsar.descriptors.molecular* un subpaquete que agrupará la colección de clases para los nuevos índices, definido como *org.openscience.cdk.qsar.descriptors.molecular.algebraic*. Este paquete representará el comportamiento de los índices algebraicos dentro de CDK, extendemos de la clase *IMolecularDescriptor*, que asume el esquema principal de un descriptor molecular, para conformar las clases *LinearDescriptor* y *QuadraticDescriptor*, como se ilustra en la figura 2.2.

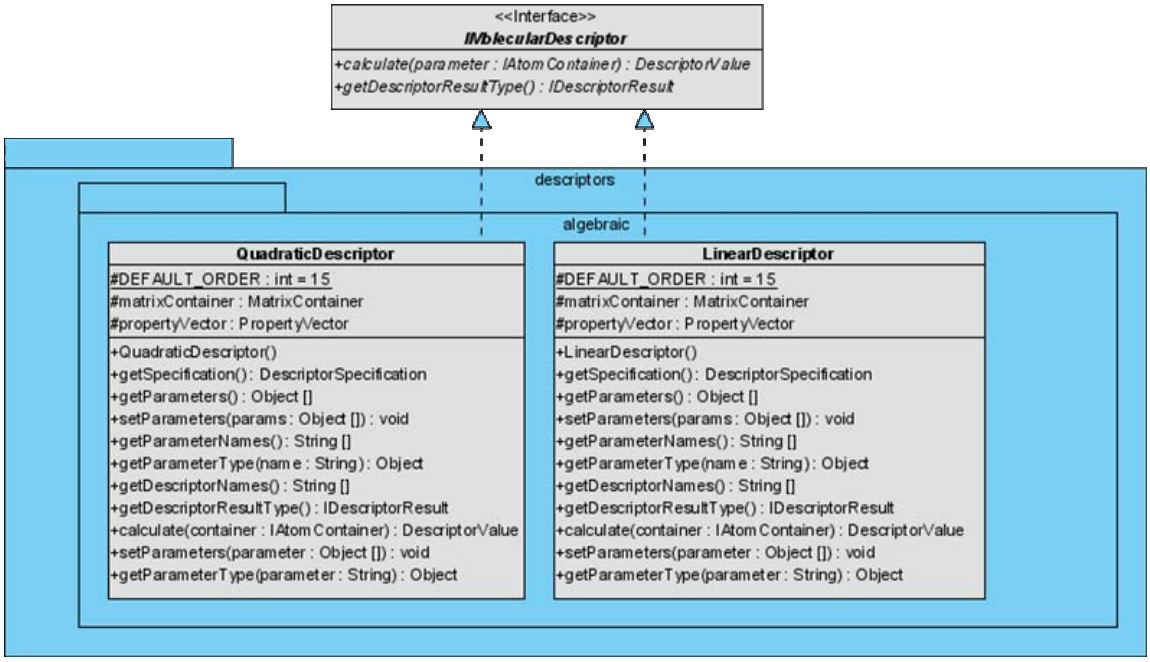


Figura 2.2 Diagrama de clases para descriptores algebraicos

Para completar el esquema del cálculo en forma algebraica de índices moleculares, introducimos un conjunto de clases representativas para cada una de las posibles ponderaciones atómicas (entiéndase, cada propiedad define una clase) y para cada una de los posibles procedimientos de construir la matriz. Por ejemplo en el (anexo) se muestra la jerarquía de herencias para un descriptor lineal (*LinearDescriptor*) formado por matrices no estocásticas (*NStochLDescriptor*) que utiliza como propiedad atómica la electronegatividad de Pauling (*ElectronegativityNSLDescriptor*). Consecuente a la definición de este nuevo descriptor definido por la clase *ElectronegativityNSLDescriptor*, incluimos la declaración en el lenguaje OWL que le corresponde, ver figuras 2.3 y 2.4 respectivamente.

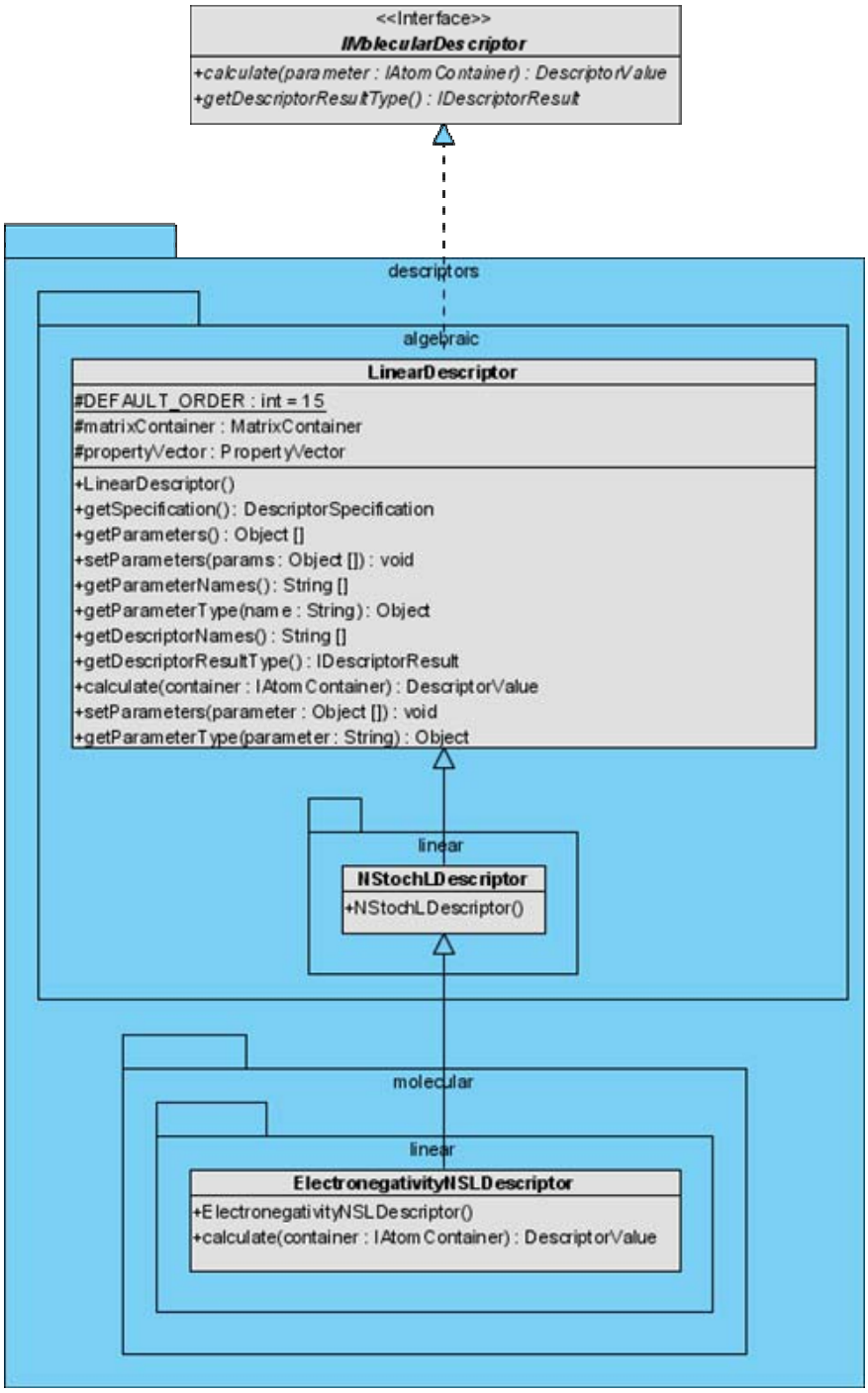


Figura 2.3 Herencia de clases para un descriptor lineal no estocástico ponderado con electronegatividad

```
<Descriptor rdf:ID="enslinear">
        <rdfs:label>
            LinearFormNonStochastic(Electronegativity)
        </rdfs:label>
        <dc:contributor rdf:resource="#jvmartini"/>
        <dc:date>2010-02-07</dc:date>
        <definition rdf:parseType='Literal'>
            The LinearNonSthochastic descriptor using atomic weight
        </definition>
        <isClassifiedAs rdf:resource="#algebraicFormDescriptor"/>
        <isClassifiedAs rdf:resource="#molecularDescriptor"/>
</Descriptor>
```

Figura 2.4 Definición en OWL para el descriptor de la figura 2.3

Con el objetivo de facilitar el proceso de incorporar un nuevo descriptor molecular para formas algebraicas completamente funcional y compatible con los requisitos de CDK, se diseñó e implementó una colección de clases que representa todas las posible alternativas, aislando solo dos pasos para satisfacer esta tarea, los que se exponen a continuación:

1. Crear una clase para especificar la propiedad que se desea evaluar en el descriptor y asignarla al *vector de propiedad*, esta clase tendrá como superclase una representación de su configuración sea el descriptor lineal o cuadrático y a la método de construir la matriz.

2. Definir la especificación del nuevo descriptor en el lenguaje OWL e incorporarla al fichero *descriptor-algorithms.owl*

Como anexo se exponen todas las posibles combinaciones de formas algebraicas y técnicas para construir las matrices, así como una lista ordenada de los descriptores algebraicos incorporados a la biblioteca CDK.

2.3 Una aplicación visual para calcular descriptores

Presentamos la aplicación BMD-DescriptorCalculator como extensión del proyecto desarrollado por Rajarshi Guha disponible en su página Web (Guha), este permite evaluar de forma sencilla los descriptores implementados en las bibliotecas CDK (Steinbeck et

al., 2003) y JOELib (Wegner, 2004), todos los descriptores serán evaluados en conjunto o se puede determinar una configuración específica seleccionando un descriptor individual. Entre sus principales características encontramos:

- Detecta automáticamente las clases definidas en el diccionario de descriptores en la biblioteca CDK
- Se pueden seleccionar los descriptores individualmente o por categorías.
- Compatible con los formatos de entrada SDF y SMI de representación de moléculas.
- El archivo de salida se puede generar en texto plano delimitado, SDF acotado o en ARFF.

En el anexo 2 se expone una lista detallada de los descriptores incorporados y sus nombres. Estos son cargados mediante la arquitectura de *plugins* que esta aplicación presenta, con solo agregar en el *classpath* el archivo que contiene los descriptores compatibles con el diseño estructural de la biblioteca CDK, se incorporan los descriptores algebraicos, desde un archivo *.jar* independiente, los que nos proporciona un apropiado esquema el caso que se desee extender las funcionalidades o incrementar el número de descriptores algebraicos.

2.4 Estimar un modelo QSAR con WEKA

En esta sección se presenta una metodología para desarrollar modelos de aprendizaje automático orientados a cuantificar relaciones estructura/actividad (*QSAR*) de la estructura de una molécula, descritas sin ambigüedades por cadenas ASCII cortas, denominadas SMILES (acrónimo en inglés para *Simplified Molecular Input Line Entry Specification*) (Anderson et al., 1987).

Varias cadenas SMILES pueden ser importadas dentro de un único archivo para representar más de una estructura molecular, generalmente estos de archivos en texto plano se identifican con la extensión *.smi*.

El primer paso para construir el modelo, parte de un par de archivos *.smi* que contienen las moléculas que intervienen en la predicción. Genéricamente asociaremos nombres a estos archivos para diferenciarlos entre el conjunto de entrenamiento y prueba, estos son *train.smi* y *test.smi* respectivamente.

Previamente conformados por las moléculas intencionalmente seleccionadas los archivos *train.smi* y *test.smi*, se procesan con la aplicación BMD-DescriptorCalculator, para caracterizar las estructuras moleculares mediante los descriptores implementados en ella, obteniendo consecuentemente los archivos de datos *train.arff* y *test.arff* plenamente compatibles con Weka.

En aras de lograr un modelo de regresión con datos numéricos, se agrega en ambos archivos de datos un nuevo atributo, que nombraremos *class* y le hacemos corresponder los valores indicativos asociados a cada molécula.

Utilizamos Weka para visualizar y procesar el archivo *train.smi* con un algoritmo para el filtrado de atributos, por ejemplo *CFSSubSetEval*, una vez determinados cuales son los atributos menos significativos y redundantes pasamos a eliminarlos de ambos archivos (*train.smi y test.smi*, entrenamiento y prueba respectivamente), tarea que se puede llevar a cabo con el filtro no supervisado para atributos *Remove*, surgiendo dos nuevos archivos de datos *train_reduced.arff* y *test_reduced.arff*.

Se calcula un modelo de regresión para el conjunto de entrenamiento especificado en el archivo *train_reduced.arff* con los algoritmos conocidos y empleando el archivo de prueba *test_reduced.arff* para la validación externa del modelo. Una vez determinado el modelo que mejor ajusta (minimizando el RMSE) se aplica un algoritmo para selección de atributos orientado a un clasificador específico (mejor modelo del paso anterior), por ejemplo *WrapperSubsetEval* evalúa el conjunto de atributos usando un esquema de aprendizaje y estima su precisión mediante validación cruzada. Análogamente eliminamos los atributos menos precisos mediante el filtro *Remove*, obteniendo los archivos *train_tuned.arff* y *test_tuned.arff*.

Entrenamos finalmente un modelo con el conjunto de atributos resultante en el archivo *train_tuned.arff* y se almacena en forma binaria serializando directamente el objeto que instancia el clasificador obtenido.

2.5 Conclusiones parciales

Se realizó un estudio de las facilidades que brinda la herramienta CDK a nivel de usuario y de la implementación en Java de su código fuente, comprobando que el diseño e implementación de las clases que las componen se concibe para que su extensión no sea una tarea compleja. Se implementó para CDK, una nueva familia de índices moleculares basados en formas algebraicas, colocados en un archivo independiente de fácil distribución y portabilidad. Estos índices están disponibles para el investigador a través de un software para el cálculo de descriptores moleculares desarrollado reutilizando el código disponible.

3 Algoritmos genéticos para la optimización de compuestos químicos

En este capítulo se presenta una de las variantes no tradicionales de algoritmos genéticos, codificando un genotipo muy exclusivo para el problema en cuestión, en las próximas secciones se puntualiza cada unos de sus rasgos, así como un ejemplo ilustrativo en concreto.

3.1 Extendiendo ECJ

La selección de la plataforma de desarrollo para algoritmos evolutivos ECJ se justifica en la sección 1.1.8 del capítulo primero. Explotando su diseño altamente flexible escrito en Java, con la mayoría de las clases y ajustes determinados dinámicamente durante la ejecución mediante un archivo de parámetros proporcionado por el usuario, ECJ concluye en una estructura preparada para ser fácilmente modificable.

Este epígrafe presenta las clases añadidas en ECJ, que implementadas en un archivo *.jar* independiente, precisan un espacio de nombres dentro del paquete *ec.molecular*, otorgando al algoritmo genético la habilidad de evolucionar compuestos químicos, y en el paquete *ec.app.molecular* que contiene el archivo de parámetros (*molecular.params*) con su correspondiente clase que representa al evaluador.

3.1.1 Individuo

El diseño que comprende la biblioteca evolutiva ECJ para codificar los individuos, deja bien claro las posibilidades de extenderla hacia otros entornos más complejos.

Se proyecta una analogía desde la representación de una molécula proporcionada por la biblioteca CDK, como fenotipo del cromosoma que identifica singularmente a cada individuo, de manera que la población agrupa varias moléculas, por decirlo de alguna forma.

Los nuevos individuos moleculares se introdujeron en ECJ respetando el patrón de diseño utilizado, como muestra la figura 3.1, especificando su comportamiento mediante la clase *ec.molecular.MolecularIndividual*. Esta clase tiene asociado un valor de fitness, un atributo que indica la especie a la que pertenece y una marca para determinar si el individuo ha sido evaluado, contiene también dos métodos que vale la pena mencionar: *defaultCrossover(...)* y *defaultMutate(...),* que definen los operadores de cruzamiento y mutación respectivamente. Las clases *MolecularCrossoverPipeline* y *MolecularMutationPipeline* situadas en el paquete *ec.molecular.breed* usan estos métodos para hacer las transformaciones sobre el material genético.

Los individuos son creados por primera vez en el método *newIndividual(...)* definido en la clase *ec.Species* e implementado en *ec.molecular.MolecularSpecies* para este problema en específico.

Parámetro que define los individuos en la especie:

```
pop.subpop.0.species.ind = ec.molecular.MolecularIndividual
```

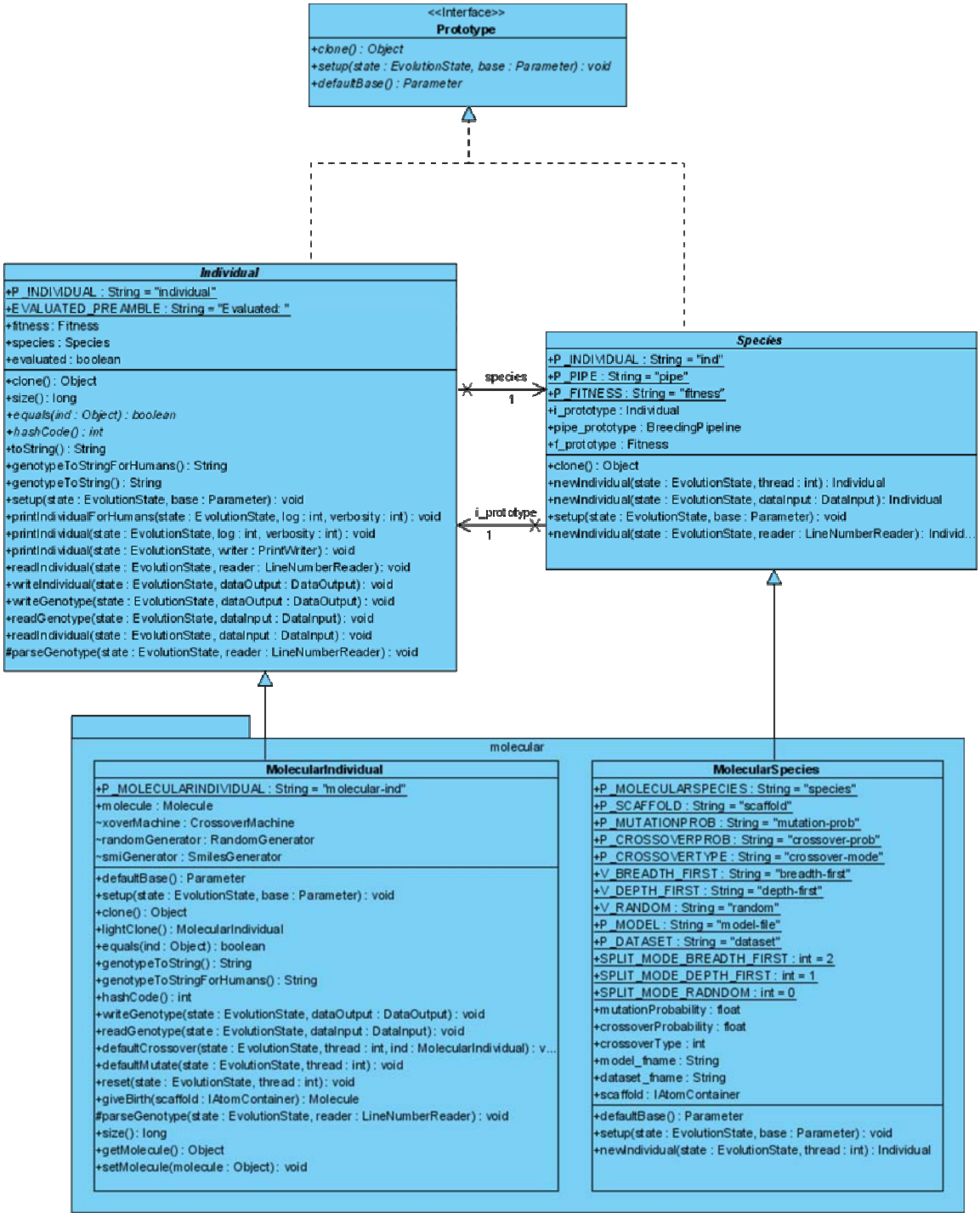


Figura 3.1 Diseño de clases incorporadas en ECJ

3.1.2 Especie

Una colección de individuos moleculares con características similares, define un espacio de información denominado "especie", donde comparten conocimientos y ajustan parámetros que controlan el algoritmo genético. Una vez creado el paquete *ec.molecular* incorporamos la clase *MolecularSpecies* favorable a garantizar la perpetuación de la especie durante la supervivencia, en esta clase podemos encontrar:

- las probabilidades de mutación y cruce para cada operador,
- el tipo de cruzamiento,
- el modelo que se usará para construir la función de evaluación, con sus respectivos datos de aprendizaje y
- el archivo con las moléculas a sembrar en la población inicial, entre otros.

Parámetro que define la especie:

```
pop.subpop.0.species            = ec.molecular.MolecularSpecies
```

3.1.3 Población inicial

Se genera aleatoriamente la población inicial a partir de un conjunto de tamaño constante constituido por cromosomas con características predeterminadas, es decir, se siembra una cantidad fija de moléculas con cualidades y efectos conocidos, que representan posibles soluciones parciales del problema, para completar el número de individuos requerido en el proceso evolutivo se propone incrementar este conjunto inicial establecido, aplicando transformaciones en sus ejemplares mediante el operador de mutación (*RandomGenerator*), detallado en la sección 3.1.5.

Es importante garantizar que la población base inicial, mantenga diversidad estructural sobre el espacio de soluciones, para que lo represente significativamente y evite la convergencia prematura.

Utilizar el fichero de configuración parametrizado que nos brinda ECJ para especificar las moléculas que se van a sembrar, es la vía más eficaz para introducir la base de la

población inicial en el sistema evolutivo, explícitamente podemos definirlo de la siguiente forma:

```
pop.subpop.0.species.smi-file = drugbank31.smi
```

alojado en la definición de la especie en la población, se asocia un nombre de archivo al parámetro **`smi-file`**, este archivo contiene la representación en formato SMILES (Anderson et al., 1987) de las moléculas a sembrar, el número de moléculas debe ser menor o igual que la cantidad límite para el tamaño de población.

Otros parámetros necesarios:

```
pop                                 = ec.Population
pop.subpops                         = 1
pop.subpop.0                        = ec.Subpopulation
pop.subpop.0.duplicate-retries      = 0
pop.subpop.0.size                   = 50
```

3.1.4 Operador de Cruzamiento

Una de las partes más exquisitas dentro del algoritmo genético es el operador de cruzamiento, que divide dos moléculas en fragmentos y los combina en cada molécula original. Descrito por la clase *CrossoverMachine* localizada bajo el paquete *org.openscience.cdk.structgen.stochastic.operator* en la biblioteca CDK, el operador de cruce falla aproximadamente el 3% de todos los casos, debido a la naturaleza aleatoria de su algoritmo, el cual se puntualiza a continuación:

a) Crear copias de las moléculas padres
b) Dividir aleatoriamente cada copia en dos fragmentos, seleccionando un conjunto-corte de enlaces al azar y eliminándolos en cada copia, logrando descomponer la molécula en dos fragmentos aislados. El conjunto de enlaces se determina con el siguiente procedimiento:
    1. Escoger un enlace al azar.
    2. Encontrar el camino más corto entre los átomos del enlace. Un camino es una lista ordenada de enlaces que comparten un átomo en cada vecindad.
    3. Seleccionar un enlace de este camino, excluirlo de la molécula y agregarlo al conjunto-corte de enlaces.
    4. Regresar al paso 2 si no se encontró el conjunto de corte.
c) Combinar un fragmento de la copia del padre con otro fragmento de la copia de la madre, de la siguiente forma:
    1. Seleccionar un enlace de corte.

```
2. Si existe algún enlace de corte compatible en los
   fragmentos de la otra copia de los padres, escoger uno al
   azar.
3. Si no se encontró ninguno, seleccionar aleatoriamente un
   enlace de corte en la otra copia del fragmento y conectar
   los átomos con nuevos enlaces, cumpliendo con la valencia.
4. Si no quedan enlaces de cortes, conectar el enlace con
   cualquier átomo del otro fragmento copia.
5. Ir al paso 1 hasta que todos los enlaces hayan sido
   procesados exactamente una vez.
```

Esta estrategia puede abrir y cerrar anillos moleculares usando solamente cruzamientos, incluso puede llegar hasta estructuras moleculares de alta dimensión mientras que existan suficientes anillos en la población. Desafortunadamente si la población no tiene anillos o está compuesta solamente por anillos no se logrará generar estructuras completas. No obstante, esta táctica es la más general de las examinadas o encontradas en la literatura (Nachbar, 1998, Weininger, 1995).

Parámetros que definen el cruzamiento:

```
pop.subpop.0.species.crossover-type   = breadth-first
pop.subpop.0.species.crossover-prob   = 0.86
pop.subpop.0.species.pipe.source.0    =
                  ec.molecular.breed.MolecularCrossoverPipeline
```

3.1.5 Operador de Mutación

Podemos encontrar algunos generadores de estructuras simples que fueron usados en el sistema computacional SENECA para la elucidación de estructuras (Steinbeck, 2001) en el paquete *org.openscience.cdk.structgen* de CDK, por ejemplo la clase *SingleRandomStructureGenerator* pretende generar estructuras totalmente aleatorias para un espacio constitucional de una formula molecular dada y *RandomGenerator* selecciona aleatoriamente cuatro átomos para alterar los enlaces entre ellos, siguiendo las reglas descritas por Jean-Loup Faulon (Faulon, 1996), lo que realiza pequeños movimientos al azar dentro del espacio constitucional (con dimensión 10^{32}).

Para complementar al operador de cruzamiento definido en la sección anterior, se incluirá en la definición del algoritmo genético este generador de estructuras al azar (*RandomGenerator*) como operador de mutación.

Parámetros que definen la mutación:

```
pop.subpop.0.species.mutation-prob    = 0.05
pop.subpop.0.species.pipe             =
                  ec.molecular.breed.MolecularMutationPipeline
```

3.1.6 Función de evaluación

El problema de cómo escribir la función de evaluación debe considerarse cuidadosamente para que se pueda alcanzar una mayor aptitud y verdaderamente signifique una solución mejor para el problema dado. Si se define de manera inexacta, puede que el algoritmo genético sea incapaz de encontrar una solución al problema, o puede acabar resolviendo el problema equivocado.

La función que se propone en este trabajo es compleja, pues emplea la representación molecular brindada por CDK para calcular ciertos descriptores algebraicos, los datos obtenidos formarán una instancia que será estimada por un modelo creado con técnicas de aprendizaje automático en Weka. El procedimiento de evaluar una molécula se definirse como:

1. Verificar que el individuo es un *MolecularIndividual* y no fue evaluado previamente.

2. Decodificar el genotipo en una molécula de CDK.

3. Caracterizar de la estructura molecular con la familia de descriptores algebraicos.

4. Construir una instancia de Weka con estos datos.

5. Estimar el grado de pertenencia al modelo previamente concebido con Weka.

6. Asignar el valor conseguido como fitness para este individuo.

7. Marcar el individuo como ya evaluado.

La implementación de este procedimiento se encuentra en el método *evaluate(…)* de la clase *ec.app.molecular.ProblemDef*, la cual concreta propiamente el evaluador para el

algoritmo genético, que actúa sobre valor de fitness definido por una subclase de *ec.Fitness*, en el caso que nos ocupa, se representa mediante la clase *ec.simple.SimpleFitness*, que indica oportunamente el resultado de calcular la función de evaluación lograda y determina el ritmo de ser seleccionado.

Parámetros significativos para la función de evaluación:

```
pop.subpop.0.species.fitness                = ec.simple.SimpleFitness
pop.subpop.0.species.model-file             = regression.model
pop.subpop.0.species.dataset                = dataset.arff
```

3.1.7 Definiendo el Evaluador

Hasta ahora hemos concebido un proceso evolutivo de alto nivel, administrando detalles, representaciones y un mecanismo de reproducción sin escribir una sola línea de código.

Proponemos escribir un objeto que sea responsable de evaluar el valor de fitness para nuestros individuos. Este objeto es llamado "Problem", y es especificado también como parámetro, de la siguiente forma:

```
eval.problem            = ec.app.molecular.ProblemDef
```

Estableciendo una base común para los problemas evolutivos que pretendan optimizar compuestos químicos, se incluye en ECJ una subclase de *Problem* denominada *ec.app.molecular.ProblemDef*, que tomará un individuo molecular miembro de la especie *MolecularSpecies*, lo evaluará y lo regresará a la población.

La mayor parte de la funcionalidad del programa que controla el algoritmo genético se determinó mediante un fichero de configuración que se compuso mediante tutorial ofrecido en la página en Internet de ECJ (ECLab, 2010) y una búsqueda entre los ficheros de configuración por defecto repartidos en distintos paquetes de la biblioteca.

3.2 Ejemplo ilustrativo

La toxicidad química está asociada a muchos efectos biológicos peligrosos tales como daño genético, carcinogénesis, o inducción de enfermedades letales en humanos o

animales. Los protocolos experimentales estándares (llamados métodos *in vivo*) han sido establecidos por industria química, compañías farmacéuticas, y agencias estatales para probar el potencial tóxico de los productos químicos. Más de 120.000 compuestos deben registrarse durante los 10 años próximos en la Unión Europea para la puesta en práctica de la nueva legislación referente al registro, la evaluación, la autorización y a la restricción de los productos químicos (REACH, del inglés *registration, evaluation, authorization and restriction of chemicals*) (Hofer et al., 2004). En el caso específico del instituto federal alemán para la estimación de riesgo estima que esta legislación podría conducir a una demanda para de hasta 45 millones de animales de laboratorio.

Las predicciones computacionales a partir de las estructuras de las moléculas, llamadas predicciones *in silico*, pueden proporcionar una alternativa a la prueba animal, ahorrando costos experimentales significativos y salvando la vida de millones de animales. La predicción de la toxicidad en animales, sin embargo, es una tarea muy compleja por lo que en los últimos tiempos se trata de forma indirecta. Para ello se considera la predicción de la toxicidad ambiental de productos químicos contra *pyriformis del T* (Zhu et al., 2008). La inhibición del crecimiento del protozoario *T. pyriformis* medido a través del $\log(IGC50^{-1})$ es una herramienta comúnmente aceptada para estimar la toxicidad de los nuevos compuestos químicos

El desarrollo de una herramienta computacional para la predicción de la toxicidad en componentes químicos es una actividad retadora debido a la alta dimensión del espacio de búsqueda (estimada entre 10^{80}-10^{100}). Por ello, emplearemos este problema para comprobar la validez de la herramienta resultante de este trabajo. Ante todo debemos construir un modelo para la estimación de la toxicidad de una molécula a partir de una muestra de aprendizaje formada por 644 moléculas (disponible en http://www.cadaster.eu/node/67) y su nivel de toxicidad ambiental. Posteriormente, se empleará tal modelo para estimar la calidad de las moléculas resultantes del proceso de evolución. Ello nos debe conducir a una parte del espacio donde se encuentren moléculas de baja toxicidad. Nótese que esta prueba pretende medir la factibilidad de nuestro enfoque para explorar regiones del espacio de las moléculas que tengan cierto comportamiento y no alcanzar resultados relevantes en el campo de la química.

Corresponde a un especialista en el dominio del problema el uso y validación de nuestra propuesta en la búsqueda y optimización de un compuesto químico con una actividad biológica deseada.

En la construcción del modelo para la estimación de la toxicidad se siguen los pasos descritos en el epígrafe cuarto del capítulo anterior sobre la estimación de modelos QSAR con Weka. El resultado de esta etapa consiste en un archivo done se ha hecho persistente el objeto que representa el mejor modelo previamente entrenado con Weka.

```
19  ...
20
21  #configure evolution
22  generations                     = 10
23  quit-on-run-complete            = true
24  checkpoint                      = false
25
26  pop.subpops                     = 1
27  pop.subpop.0                    = ec.Subpopulation
28  pop.subpop.0.size               = 50
29
30  pop.subpop.0.species            = ec.molecular.MolecularSpecies
31  pop.subpop.0.species.ind        = ec.molecular.MolecularIndividual
32  pop.subpop.0.species.fitness    = ec.simple.SimpleFitness
33
34  #settings the probs
35  pop.subpop.0.species.crossover-type = breadth-first
36  pop.subpop.0.species.crossover-prob = 0.86
37  pop.subpop.0.species.mutation-prob  = 0.33
38
39  #loading file for stored model
40  pop.subpop.0.species.model-file     = regression.model
41  pop.subpop.0.species.dataset        = dataset.arff
42
43  ## Use our own custom breeding class
44  pop.subpop.0.species.pipe               = ec.molecular.breed.MolecularMutationPipeline
45  pop.subpop.0.species.pipe.source.0      = ec.molecular.breed.MolecularCrossoverPipeline
46
47  #set the file name containing the smiles represetation for seed
48  pop.subpop.0.species.smi-file = drugbank01.smi
49
50  #molecular problem definition
51  eval.problem            = ec.app.molecular.ProblemDef
```

Figura 3.2 Archivo de parámetros para ECJ

Posteriormente se necesita configurar los parámetros del proceso evolutivo que tendrá lugar en ECJ. En esta prueba se configura el archivo de parámetros de la forma especificada en la figura 3.2. La línea 22 indica el límite de diez generaciones, la línea 26 especifica un tamaño de población de 50. Las próximas tres líneas indican las clases

encargadas de modelar la especie, el individuo y el tipo de función de aptitud, en este caso *SimpleFitness* debido a que se estima un único objetivo. Las líneas 36 y 37 indican la tasa de cruzamiento y mutación respectivamente. A continuación se especifican los archivos que almacenan en modelo y la muestra de aprendizaje, seguido por la definición de las clases responsables de realizar transformaciones de los individuos que representan moléculas (mutación y cruce).

Por último especificamos la moléculas propuestas como base de la población inicial, mediante un archivo .*smi*, y terminado de configurar los parámetros de ECJ con la especificación de el evaluador definido exclusivamente para este problema.

La ejecución de esta configuración de ECJ arroja que en un tiempo breve se obtiene una molécula, en la última generación, con una de toxicidad de 0.31. Esto representa un logro si se tiene en cuenta que la misma medida de la molécula sembrada al inicio del proceso es de 0.52.

3.3 Conclusiones parciales

Se realizó un estudio de las facilidades que brindan la herramienta ECJ comprobando que su extensión no es na tarea compleja. Se incorporó un nuevo tipo de individuos con su especie correspondiente denominados *MolecularIndividual* y *MolecularSpecies* respectivamente, que acoplan las estructuras moleculares al sistema evolutivo, y se implementaron los operadores genéticos capaces de transformar estas representaciones.

Conclusiones

Se desarrolló una herramienta computacional que integra de modo natural bibliotecas de software especializadas en algoritmos genéticos (ECJ), química computacional (CDK) y aprendizaje automático (Weka) para la optimización de compuestos químicos. Esta herramienta se utiliza de forma experimental en la optimización de una molécula para obtener una nueva con valor mínimo de su toxicidad.

La plataforma resultante de combinar bibliotecas desarrolladas en Java siguiendo el estándar de código abierto incluye los siguientes aportes:

- Se realizó un estudio de las facilidades que brindan las herramientas ECJ y CDK a nivel de usuario y de la implementación en Java de su código fuente, comprobando que el diseño e implementación de las clases que las componen se concibe para que su extensión no sea una tarea compleja.

- Se incorporó un nuevo tipo de individuos con su especie correspondiente denominados *MolecularIndividua*l y *MolecularSpecies* respectivamente, que acoplan las estructuras moleculares al sistema evolutivo

- Se propuso una metodología para la implementación de nuevos problemas con solo establecer la función de evaluación para el modelo con su conjunto de aprendizaje.

- Se implementó para CDK, una nueva familia de índices moleculares basados en formas algebraicas, colocados en un archivo independiente de fácil distribución y portabilidad. Estos índices están disponibles para el investigador a través de un software para el cálculo de descriptores moleculares desarrollado reutilizando el código disponible.

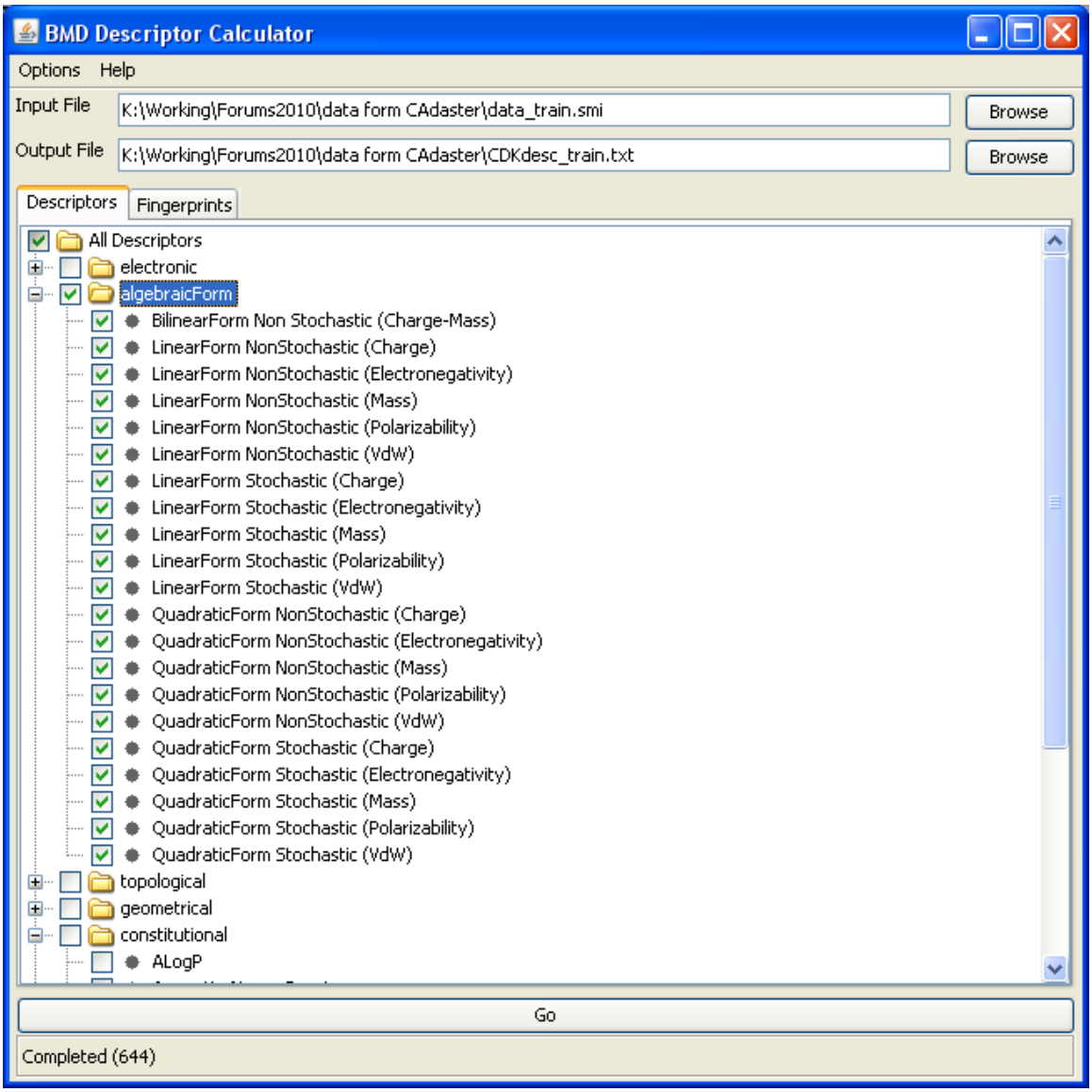
Recomendaciones

Este trabajo constituye el inicio de una herramienta computacional de extrema utilidad para los investigadores en el diseño de nuevos compuestos químicos. Desde el punto de vista computacional se ha comprobado la validez de nuestra integración de técnicas especializadas en cada uno de los componentes que intervienen en la optimización de nuevos compuestos. Queda ahora someter a la herramienta a una validación experimental utilizando datos de una aplicación real por parte de un equipo multidisciplinario donde intervenga un especialista en el diseño computacional de fármacos.

Por otra parte, la modelación que se ha hecho de los descriptores lineales y cuadráticos puede ser extendida mediante el uso de nuevas propiedades moleculares, otras formas algebraicas y nuevos métodos de construir la matriz con información de la topología de la molécula. Los nuevos descriptores creados de esta forma pueden considerar aspectos de interés para una aplicación específica que ahora no se tengan en cuenta por el límite en tiempo que impone la realización del presente trabajo.

Anexos

1.Ventana de la aplicación para el cálculo de descriptores



2. Relación de los nombres para los descriptores algebraicos implementados

1. ChargeNSLDescriptor.java
2. ChargeNSQDescriptor.java
3. ChargeSSLDescriptor.java
4. ChargeSSQDescriptor.java
5. ChargeXAtomsNSLDescriptor.java
6. ElectronegativityAAtomsNSLDescriptor.java
7. ElectronegativityAAtomsNSQDescriptor.java
8. ElectronegativityAAtomsSSLDescriptor.java
9. ElectronegativityAAtomsSSQDescriptor.java
10. ElectronegativityDAtomsNSLDescriptor.java
11. ElectronegativityDAtomsNSQDescriptor.java
12. ElectronegativityDAtomsSSLDescriptor.java
13. ElectronegativityDAtomsSSQDescriptor.java
14. ElectronegativityGAtomsNSLDescriptor.java
15. ElectronegativityGAtomsNSQDescriptor.java
16. ElectronegativityGAtomsSSLDescriptor.java
17. ElectronegativityGAtomsSSQDescriptor.java
18. ElectronegativityNSLDescriptor.java
19. ElectronegativityNSQDescriptor.java
20. ElectronegativitySSLDescriptor.java
21. ElectronegativitySSQDescriptor.java
22. ElectronegativityXAtomsNSLDescriptor.java
23. ElectronegativityXAtomsNSQDescriptor.java

24. ElectronegativityXAtomsSSLDescriptor.java
25. ElectronegativityXAtomsSSQDescriptor.java
26. MassNSLDescriptor.java
27. MassNSQDescriptor.java
28. MassSSLDescriptor.java
29. MassSSQDescriptor.java
30. PolarizabilityNSLDescriptor.java
31. PolarizabilityNSQDescriptor.java
32. PolarizabilitySSLDescriptor.java
33. PolarizabilitySSQDescriptor.java
34. VdWNSLDescriptor.java
35. VdWNSQDescriptor.java
36. VdWSSLDescriptor.java
37. VdWSSQDescriptor.java

Referencias

ALBA, E. & COTTA, C. 2003. Tutorial on evolution computation.

ANDERSON, E., VEITH, G. D. & WEININGER, D. 1987. SMILES: A line notation and computerized interpreter for chemical structures. *U.S. EPA, Environmental Research Laboratory-Duluth*.

ARENAS, M. G., FOUCART, L., SCHOENAUER, M. & MERELO, J. J. Computación Evolutiva en Java: JEO. *Universidad de Granada. Facultad de Ciencias*.

AXLER, S. 1996. Linear Algebra Done Right. *Springer-Verlag: New York***,** 37-70.

BACK, T., FOGEL, D. B. & MICHALEWICZ, Z. 1997. Handbook of Evolutionary Computation. *OUP/IOP*.

BECHHOFER, S., HARMELEN, F. V., HENDLER, J., HORROCKS, I., MCGUINNESS, D. L., PATEL-SCHNEIDER, P. F. & STEIN, L. A. 2009. OWL Web Ontology Language *W3C Semantic Web Activity*.

BLAUCH, D. 2002. Java Classes for Managing Chemical Information and Solving Generalized Equilibrium Problems. *J. Chem. Inf. Comput. Sci.,* 42**,** 143-146.

BROWDER, A. 1996. Mathematical Analysis. An Introduction. *Springer-Verlag, New York, Inc*.

CHEN, H. F. 2008 Quantitative predictions of gas chromatography retention indexes with support vector machines, radial basis neural networks and multiple linear regresión. *Analytica chimica acta.*

CLARK, D. E. & WESTHEAD, D. R. 1996. *J. Comput.-Aided Mol. Des.,* 10**,** 337-358.

COSTA, J., LOPES, N. & SILVA, P. JDEAL, The Java Distributed Evolutionary Algorithms Library. *http://laseeb.ist.utl.pt/sw/jdeal*.

CSIZMADIA, F. 2000. JChem: Java Applets and Modules Supporting Chemical Database Handling from Web Browsers. *J. Chem. Inf. Comput. Sci.,* 40**,** 323-324.

DE JONG, K. A. & SARMA, J. 1993. Generation Gaps Revisited. *Foundations of Genetic Algotihms 2.***,** 19-28.

DEKKER, M. 2000. High Throughput Screening.. *New York: McGraw Hill*.

DEVILLERS, J. 1996a. *J. Chem. Inf. Comput. Sci,* 36**,** 1061-1066.

DEVILLERS, J. 1996b. Genetic Algorithms in Molecular Modelling. *Academic Press, London*.

DIXIT, K. S. & MITRA, S. N. 2002. CRIPS.

DREWS, J. 2000. *Science,* 287**,** 1960.

ECLAB. 2010. Available: http://cs.gmu.edu/~eclab/projects/ecj/ [Accessed Junio 2010].

EDWARDS, C. H. & PENNEY, D. E. 1988. Elementary Linear Algebra. *Prentice-Hall, Englewood Cliffs: New Jersey, USA*.

ESTRADA, E. & URIARTE, E. 2001. Curr Med Chem. 8**,** 1573.

FAULON, J.-L. 1996. Stochastic Generator of Chemical Structure. 2. Using Simulated Annealing To Search the Space of Constitutional Isomers. *J. Chem. Inf. Comput. Sci.,* 36**,** 731-740.

FRANK, I. E. & FRIEDMAN, J. H. 1989. *J. Chemom.,* 3**,** 463.

GLEN, C. & A. W. R. PAYNE, J. 1995. *Comput.-Aided Mol. Des,* 9**,** 181-202.

GOLDBERG, D. E. 1989. Genetic Algorithms in Search Optimization and Machine Learning. *Addison-Wesley, Reading, MA*.

GOLDBERG, D. E., DEB, K. & CLARK, J. H. 1991. Genetic Algorithms, Noise, and the Sizing of Populations. *IlliGAL Report No. 91010. – University of Illinois, Urbana-Champaign*.

GUHA, R. Available: http://rguha.net/code/java/cdkdesc.html [Accessed junio 2010].

HAND, D. J. 1981. Discrimination and Classification. *Chichester (UK): Wiley*.

HOFER, T., GERNER, I., GUNDERT-REMY, U., LIEBSCH, M., SCHULTE, A., SPIELMANN, H., VOGEL, R. & WETTIG, K. 2004. Animal testing and alternative approaches for the human health risk assessment under the proposed new European chemicals regulation. *Arch. Toxicol.,* 78**,** 549-564.

HOLLAND, J. H. 1975. Adaptation in Natural and Artificial Systems. *The University of Michigan Press, Ann Arbor*.

KUBINYI, H. 2004. Changing paradigms in drug discovery. *The Chemical Theatre of Biological Systems: Bozen, Italy.*

LUKE, S., PANAIT, L., BALAN, G., PAUS, S., SKOLICKI, Z., POPOVICI, E., SULLIVAN, K., HARRISON, J., BASSETT, J., HUBLEY, R., CHIRCOP, A., COMPTON, J., HADDON, W., DONNELLY, S., JAMIL, B. & O'BEIRNE, J. A Java-based Evolutionary Computation Research System. *Evolutionary Computation Journal*.

MALTSEV, A. I. 1976. Fundamentos del Álgebra Lineal. *Mir: Moscow* 68-262.

MARRERO-PONCE, Y. 2003. *Molecules,* 8**,** 687.

MARRERO-PONCE, Y., CASTILLO-GARIT, J. A., OLAZABAL, E., SERRANO, H. S., MORALES, A., CASTAÑEDO, N., IBARRA-VELARDE, F., HUESCA-GUILLEN, A., JORGE, E., DEL VALLE, A., TORRENS, F. & CASTRO, E. A. J. 2004. *Comput. Aided Mol. Des.,* 18**,** 615.

MARRERO-PONCE, Y., MEDINA-MARRERO, R., CASTILLO-GARIT, J. A., ROMERO-ZALDIVAR, V., TORRENS, F. & CASTRO, E. A. 2005. *Bioorg. Med. Chem.,* 13**,** 3003.

MARRERO PONCE, Y., CASTILLO GARIT, J. A. & NODARSE, D. 2005. *Bioorg. Med. Chem.,* 13**,** 3397.

MERELO, J. J., ARENAS, M. G., CARPIO, J., CASTILLO, P., RIVAS, V. M., ROMERO, G. & SCHOENAUER, M. 2000. Evolving Objects. *Proc. JCIS 2000 (Joint Conference on Information Sciences), P.P. Wang, Ed,* 1**,** 1083-1086.

MERELO, J. J., KEIJZER, M. & SCHOENAUER, M. EO evolutionary computation framework. *http://eodev.sourceforge.net*.

MICHALEWICZ, Z. 1992. Genetic Algorithms + Data Structures = Evolution Programs. *Springer Verlag, Berlin*.

MITCHELL, M. 1996. An Introduction to Genetic Algorithms. *MIT Press, Cambridge (MA)*.

NACHBAR, R. B. 1998. Molecular evolution: a hierarchical representation for chemical topology and its automated manipulation. *Proceedings of the Third Annual Genetic Programming Conference, University of Wisconsin, Madison, Wisconsin***,** 246-253.

OECD 2004. The Report from the Expert Group on (Quantitative) Structure–Activity Relationships [(Q)SARs] on the Principles for the Validation of

(Q)SARs. ENV/JM/TG(2004)27/REV. *In:* DEVELOPMENT, O. F. E. C. A. (ed.). Paris.

OECD 2007. Guidance Document on the Validation of (Quantitative) Structure–Activity Relationships [(Q)SAR] Models. ENV/JM/ MONO(2007)2. *In:* DEVELOPMENT, O. F. E. C. A. (ed.). Paris

OPENSCIENCEPROJECT. *The OpenScience Project* [Online]. Available: http://www.openscience.org [Accessed Junio 2010].

OSI. 2010. *The Open Source Initiative (OSI)* [Online]. Available: http://www.opensource.org [Accessed Junio 2010].

PARRILL, A. L. 1996. *Drug Discovery Today,* 1**,** 514-521.

PEDERSEN, J. T. & MOULT, J. 1996. *Curr. Opin. Struct. Biol,* 6.

PIPROFILE. 2009. *Pharmaceutical Industry Profile* [Online]. Available: http://www.phrma.org/profiles%26reports [Accessed].

ROTSTAN, N. & MEFFERT, K. 2008. JGAP: Java Genetic Algorithms Package. *http://jgap.sourceforge.net/index.html*.

RZEPA, H. & TONGE, A. 1998. VChemLab: A Virtual Chemistry Laboratory. The Storage, Retrieval, and Display of Chemical Information Using Standard Internet Tools. *J. Chem. Inf. Comput. Sci.,* 38**,** 1048-1053.

SERVICE, C. A. 2009. 50 Millionth Unique Chemical Substance Recorded in CAS REGISTRY.

SMITH, A. 2002. *Nature,* 418**,** 453.

STEINBECK, C. 2001. SENECA: A Platform-Independent, Distributed and Parallel System for Computer-Assisted Structure Elucidation in Organic Chemistry. *J. Chem. Inf. Comput. Sci.,* 41**,** 1500-1507.

STEINBECK, C., HAN, Y., KUHN, S., HORLACHER, O., LUTTMANN, E. & WILLIGHAGEN, E. 2003. The Chemistry Development Kit (CDK): An Open-Source Java Library for Chemo and Bioinformatics. *Journal of Chemical Information and Computer Sciences,* 43**,** 493-500.

STEINBECK, C. A. H., Y. Q. AND KUHN, S. AND HORLACHER, O. AND LUTTMANN, E. AND WILLIGHAGEN, E.L. 2003. The Chemistry Development Kit (CDK): An open-source Java library for chemo- and bioinformatics. *Journal of Chemical Information and Computer Sciences* 43**,** 493-500.

TODESCHINI, R., CONSONI, V. 2000. *Handbook of Molecular Descriptors,* Weinheim.

TROPSHA, A., GRAMATICA, P., GOMBAR, V. K. 2003. The importance of being earnest: Validation is the absolute essential for successful application and interpretation of QSPR models. *QSAR Comb. Sci.*

VAN DE WATERBEEMD, H. 1995. *Chemometric Methods in Molecular Design,* New York.

VANHOOF, K. 2006. Conceptual Course on Data Mining. *Universidad Central "Martha Abreu" de Las Villas: Santa Clara, Cuba,* 101.

VENKATASUBRAMANIAN, V., CHAN, K. & CARUTHERS, J. M. 1994. Comput. Chem. Eng. 18**,** 833-844.

VENKATASUBRAMANIAN, V., CHAN, K. & CARUTHERS, J. M. 1995a. *J. Chem. Inf. Comput. Sci,* 35**,** 188-195.

VENKATASUBRAMANIAN, V., CHAN, K. & CARUTHERS, J. M. 1995b. *ACS Symp. Ser,* 389**,** 396-411.

WALL, M. 1995. Overview of Matthew's genetic algorithm library. *http://lancet.mit.edu/ga,1995*.

WEGNER, D. C. J. R. K. 2004. JOELib Tutorial: A Java based cheminformatics/computational chemistry package. *Computer Architecture, University of TÃ¼bingen*.

WEININGER, D. 1995. Method and apparatus for designing molecules with desired properties by evolving successive populations. *U.S. patent US5434796, Daylight Chemical Information Systems, Inc.* .

WESTHEAD, D. R., CLARK, D. E., FRENKEL, D., LI, J., MURRAY, C. W., ROBSON, B. & WASZKOWYCZ, B. 1995. *J. Comput.-Aided Mol. Des.,* 9**,** 139-148.

WHITLEY, D. L. 1991. Fundamental principles of deception in genetic search. *Foundations of genetic algorithms. - San Mateo, CA: Morgan Kaufmann***,** 221-241.

WHITLEY, D. L. 2002. Genetic Algorithms and Evolutionary Computing. *Van Nostrand's Scientific Encyclopedia*.

WILLETT, P. 1995. *Trends Biotechnol,* 13**,** 516-512.

WITTEN, I. H. & FRANK, E. 2000. Data Mining: Practical machine learning tools with Java implementations. *Morgan Kaufmann, San Francisco.*

WORTH, A. P., BASSAN, A., GALLEGOS, A., NETZEVA, T.I., PATLEWICZ, G.,PAVAN, M., TSAKOVSKA, I., VRACKO, M. 2005. The Characterisation of (Quantitative) Structure–Activity Relationships: Preliminary Guidance. *In:* EUR (ed.) *21866.* http://ecb.jrc.it/.

ZHU, H., TROPSHA, A., FOURCHES, D., VARNEK, A., PAPA, E., GRAMATICA, P., OBERG, T., DAO, P., CHERKASOV, A. & TETKO, I. V. 2008. Combinatorial QSAR Modeling of Chemical Toxicants Tested against Tetrahymena pyriformis J. *Chem. Inf. Model.,* 48**,** 766-784.