UNIVERSIDAD NACIONAL DEL LITORAL

Desarrollo de algoritmos evolutivos para el descubrimiento de relaciones en datos metabólicos

Matias Fernando Gerard

FICH
FACULTAD DE INGENIERIA
Y CIENCIAS HIDRICAS

INTEC
INSTITUTO DE DESARROLLO TECNOLOGICO
PARA LA INDUSTRIA QUIMICA

Tesis de Doctorado **2014**

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
M. Gerard; "Desarrollo de algoritmos evolutivos para el descubrimiento de relaciones en datos metabólicos"
Universidad Nacional del Litoral, 2014.

UNIVERSIDAD NACIONAL DEL LITORAL
Facultad de Ingeniería y Ciencias Hídricas
Instituto de Desarrollo Tecnológico para la Industria Química

Desarrollo de algoritmos evolutivos para el descubrimiento de relaciones en datos metabólicos

Matias Fernando Gerard

Tesis remitida al Comité Académico del Doctorado
como parte de los requisitos para la obtención
del grado de
DOCTOR EN INGENIERIA
Mención Inteligencia Computacional, Señales y Sistemas
de la
UNIVERSIDAD NACIONAL DEL LITORAL

2014

Comisión de Posgrado, Facultad de Ingeniería y Ciencias Hídricas, Ciudad Universitaria,
Paraje .El Pozo", S3000, Santa Fe, Argentina.

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
M. Gerard; "Desarrollo de algoritmos evolutivos para el descubrimiento de relaciones en datos metabólicos"
Universidad Nacional del Litoral, 2014.

A mi esposa e hijas,

los soles que iluminan mis días.

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
M. Gerard; "Desarrollo de algoritmos evolutivos para el descubrimiento de relaciones en datos metabólicos"
Universidad Nacional del Litoral, 2014.

Agradecimientos

Mientras escribo estas líneas, no dejo de recordar a toda la gente que he conocido durante esta etapa. Con muchos de ellos aún continúo trabajando, pero todos han contribuido para que hoy sea la persona que soy.

Quiero expresar mi agradecimiento a mi director, Dr. Diego Milone, por compartir conmigo su experiencia y permitirme enriquecer esta tesis con sus puntos de vista. De igual manera, quiero agradecer a mi co-directora Dra. Georgina Stegmayer, por abrirme las puertas al mundo de la investigación y guiarme a lo largo de todo el trabajo con suma dedicación. A ambos, por la oportunidad de trabajar en el **sinc**(i) y el CIDISI, por las numerosas discusiones sobre la tesis, y por confiar en mi para llevar adelante este trabajo.

Quiero agradecer a todos mis compañeros del **sinc**(i) y del CIDISI, con quienes he compartido lindos momentos dentro y fuera del trabajo, por crear un espacio de trabajo ideal. Particularmente quiero agradecer a Leandro Di Persia, por su compañerismo y predisposición para atender cualquier inquietud que tuviera. A Leandro Vignolo, por compartir su experiencia y estar dispuesto siempre a brindarme su ayuda. Quiero agradecer a Federico y Carlos, quienes me acompañaron en gran parte de esta tesis, por su amistad y por los gratos momentos compartidos. También les quiero agradecer a Mariano, Jorge, Ivanna, Milton, David, Guillermo, Leonardo, César, Marcelo, Guido, Diego, Iván, José, Román, Tomás, Sebastián y Mariela, por su compañerismo y amistad. Al Dr. Leonardo Giovanini, con quien he compartido largas charlas, por sus consejos y opinión. A la Dra. María Eugenia Torres, por estar siempre dispuesta a ayudar con sus consejos y sugerencias.

Especialmente quiero agradecer a mi esposa, Luisa, por estar siempre a mi lado durante el transcurso de esta etapa, apoyándome y alentándome para seguir adelante en los momentos difíciles. Su apoyo incondicional hizo posible mi dedicación a esta carrera. También agradezco a Antonella y Olivia, que con su ternura y simpatía me han dado fuerzas para seguir adelante. No puedo dejar de agradecer a mis padres, Graciela y Carlos, por su ejemplo, su comprensión y por la ayuda que me han brindado siempre. A mis hermanos, Hernán y Andrés, quienes siempre me han brindado su apoyo. También agradezco a mi suegra, Adriana, quién también me ha aconsejado y apoyado

siempre.

Finalmente, quisiera extender mi agradecimiento a las siguientes instituciones:

- **sinc**(i): Centro de Investigación en Señales, Sistemas e Inteligencia Computacional.
- CIDISI: Centro de Investigación y Desarrollo de Ingeniería en Sistemas de Información.
- Facultad de Ingeniería y Ciencias Hídricas, Universidad Nacional del Litoral.
- Facultad Regional Santa Fe, Universidad Tecnológica Nacional.
- CONICET: Consejo Nacional de Investigaciones Científicas y Técnicas.
- Laboratorio de Cibernética (Facultad de Ingeniería, Universidad Nacional de Entre Ríos).

A todos y cada uno de ustedes, mi más sincero agradecimiento,

Matias Fernando Gerard
Santa Fe, Noviembre de 2013

Resumen

En los últimos años, el creciente volumen de datos producidos en experimentos biológicos ha llevado a la necesidad de desarrollar nuevas herramientas computacionales que sean capaces de manipular y analizar esta información para extraer conocimiento. En particular, cuando los datos analizados corresponden a compuestos biológicos se sabe que las relaciones entre ellos están determinadas por reacciones bioquímicas y por la interacción que se establece entre las mismas. Estas interacciones son sumamente importantes porque permiten transformar un compuesto particular en otro totalmente diferente mediante secuencias de reacciones que conforman vías metabólicas. Por lo tanto, conocer estas secuencias es esencial para planificar la producción de un compuesto, o para identificar si es posible sintetizar un compuesto de interés a partir de otro especificado.

En esta tesis se propone el desarrollo de dos nuevos algoritmos, basados en la computación evolutiva, que permitan encontrar vías metabólicas que relacionen un conjunto de compuestos. En particular se abordan los problemas de búsqueda de caminos lineales y ramificados. Para ambos algoritmos se propone codificar las vías metabólicas en los cromosomas como secuencias de reacciones, donde cada gen es una reacción y la posición en el cromosoma determina el orden en que cada reacción se lleva a cabo. Además, se propone el uso de operadores de variación modificados que utilizan información del cromosoma para mejorar el desempeño de la búsqueda. También se define una función de aptitud que contempla propiedades características de las cadenas de reacciones, y una nueva estrategia de inicialización para generar cromosomas de tamaño variable que codifiquen secuencias ordenadas de reacciones.

Los resultados obtenidos con el algoritmo para la búsqueda de vías metabólicas lineales muestran que la nueva estrategia de inicialización mejora significativamente la calidad de los individuos. Ésto combinado con operadores de cruza y mutación modificados reduce en forma considerable el número de generaciones requerido para encontrar una solución. El desempeño de este

algoritmo fue comparable al del algoritmo clásico de búsqueda en amplitud, empleando tiempos similares de búsqueda. Además, las soluciones encontradas con el algoritmo propuesto incluyeron a los caminos más cortos y a otros de mayor tamaño, que relacionan los compuestos empleando mecanismos alternativos para la síntesis. Una vía metabólica no reportada previamente en la literatura para *Arabidopsis thaliana* fue encontrada en pruebas realizadas sobre problemas de interés práctico. Esta vía permite relacionar dos compuestos para los cuales no se conoce hasta el momento un mecanismo para la síntesis de uno de ellos a partir del otro.

Los resultados obtenidos para el algoritmo para búsqueda de vías ramificadas también muestran una reducción en el número de generaciones requeridas para encontrar una vía metabólica al emplear los operadores modificados y la estrategia de inicialización propuesta. Particularmente, la posibilidad de emplear tamaños variables en la inicialización permite evitar el problema de definir el número adecuado de genes para la búsqueda. El algoritmo fue capaz de encontrar una amplia variedad de soluciones, con distinto tipo de interconexiones entre las reacciones, en pruebas realizadas empleando conjuntos de reacciones de diferente tamaño. Para validar el algoritmo se realizaron pruebas donde se buscó reproducir dos vías metabólicas de referencia. Las búsquedas realizadas permitieron reproducir estas vías en forma total o parcial, generando en este último caso mecanismos alternativos para relacionar los compuestos. En todos los casos, las soluciones encontradas son completamente válidas como vías metabólicas.

Abstract

In the last years the growing volume of data produced in biological experiments has led to the need of new computational tools that are able to manipulate and analyze this information to extract knowledge. In particular, when the analyzed data correspond to biological compounds it is known that the relationship between them are determined by biochemical reactions and the interactions established between them. These interactions are extremely important because they allow the transformation of one particular compound into another totally different, through a sequence of reactions that are part of a metabolic pathway. Therefore, knowing how these sequences are established is essential for planning a compound production, or to identify if its possible to synthesize a compound of interest from another.

This thesis proposes new algorithms mainly based on evolutionary computation to find metabolic pathways that relate a set of compounds. In particular, we solve the problem of searching lineal and branched pathways. For both algorithms, it is proposed the encoding of metabolic pathways as reactions sequences on the chromosomes, wherein each gene is a reaction and the position on the chromosome determines the order in which each reaction is carried out. In addition, it is proposed the use of modified variation operators that use the chromosome information to improve search performance. Also, the use of a fitness function including characteristic properties of reactions chains is proposed, and a new initialization strategy to generate variable size chromosomes that encodes ordered sequences of reactions is proposed.

Results obtained with the algorithm for finding linear metabolic pathways show that the new initialization strategy improves significantly the quality of individuals. Combining this strategy with modified crossover and mutation operators considerably reduces the number of generations required to find a solution. The performance of this algorithm was comparable to the classical breadth-first search algorithm, and they use similar searching times. In addition, solutions found with the proposed algorithm included the shortest paths and larger ones. The latter allow to relate the specified compounds

using alternative mechanisms for their synthesis. Tests on practical interest problems allowed to find a metabolic pathway not previously reported in the literature for *Arabidopsis thaliana*, which would relate two compounds for which is not known a mechanism for the synthesis of one of them from the other.

The results obtained for the search algorithm for branching pathways also show a reduction in the number of generations required for finding a metabolic pathway modified by using the proposed operators and the new initialization strategy. Particularly, the possibility of using varying sizes during the initialization avoids the problem of defining the appropriate number of genes for the search. Tests performed with sets containing different number of reactions showed that the algorithm is able to find a wide variety of solutions having different types of interconnections between reactions. As one way to validate the algorithm it was proposed to reproduce two reference metabolic pathways. Performed searches reproduced these pathways in a partial or total way, finding in these last ones alternative mechanisms to relate the compounds. In all cases, the solutions are fully realizable as metabolic pathways.

Índice

Índice de figuras

Índice de tablas

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
M. Gerard; "Desarrollo de algoritmos evolutivos para el descubrimiento de relaciones en datos metabólicos"
Universidad Nacional del Litoral, 2014.

Índice de algoritmos

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
M. Gerard; "Desarrollo de algoritmos evolutivos para el descubrimiento de relaciones en datos metabólicos"
Universidad Nacional del Litoral, 2014.

Capítulo 1

Introducción

1.1. Motivación

En años recientes se ha producido una explosión en el estudio de las características estructurales y dinámicas de las redes complejas [Boccaletti et al., 2006; Sayama et al., 2013]. Estas redes se componen de elementos que al interactuar conducen a la aparición de propiedades globales que no pueden explicarse a partir de las propiedades individuales de sus elementos aislados. El interés principal que motiva su estudio es la gran abundancia de estas redes en la naturaleza y la diversidad que presentan en los niveles de organización. A nivel macroscópico encontramos las redes sociales y las redes de comunicación. A nivel microscópico, un ejemplo típico son las redes biológicas intracelulares tales como las redes de regulación genética, las redes de interacción entre proteínas y las vías metabólicas [Albert and Barabasi, 2002; Costa et al., 2011].

Una forma natural de modelar estas redes es mediante representaciones basadas en grafos. Estas estructuras modelan los objetos como *nodos* y las relaciones entre pares de nodos mediante *arcos*. También es posible incorporar información en los arcos acerca de la dirección de la relación, o sobre la relevancia de los nodos o arcos en la red [Gould, 2012]. Por ejemplo, es posible cuantificar la transferencia de información entre nodos de una red particular especificando valores de flujo sobre los arcos. Estas representacio-

nes permiten estudiar características topológicas tales como la distribución del grado de conectividad y la distancia más corta entre nodos, el diámetro de la red y la modularidad, entre otros [Ravasz et al., 2002; Aittokallio and Schwikowski, 2006].

En bioinformática, una aplicación de interés particular es la búsqueda de vías metabólicas que transforman un compuesto especificado en otro que resulta de interés [Schuster et al., 2000]. Por ejemplo, esto puede ser de utilidad para averiguar si un determinado organismo es capaz de sintetizar un compuesto particular a partir de ciertos sustratos. Tradicionalmente, el proceso de búsqueda de vías metabólicas ha sido manual, mediante búsqueda bibliográfica y en bases de datos de reacciones, como por ejemplo KEGG[1]. Sin embargo, el gran volumen de datos metabólicos disponible actualmente ha creado la necesidad de desarrollar herramientas computacionales que permitan identificar vías relevantes de forma automática. Actualmente, la búsqueda puede abordarse empleando dos estrategias diferentes: métodos basados en relaciones cuantitativas entre compuestos y métodos basados en búsqueda sobre grafos [Planes and Beasley, 2008].

Los enfoques basados en relaciones cuantitativas entre las entradas y las salidas de las reacciones [Palsson, 2006, 2011] intentan identificar vías específicas dentro de la red metabólica, en las que las concentraciones de los compuestos que la conforman se mantiene inalterada. Aunque las vías encontradas son bioquímicamente viables, el costo computacional asociado a la evaluación del sistema es muy elevado. Incluso para pequeñas redes, el número de vías potenciales que deben considerarse es muy alto y se incrementa exponencialmente con el tamaño de la red [Terzer and Stelling, 2008]. Los enfoques basados en búsqueda sobre grafos surgen como una alternativa a los métodos anteriores, principalmente para tratar con redes de gran tamaño. Estos suponen la existencia de un camino que no contiene ciclos y que relaciona los compuestos de interés. Es importante notar que se habla de vías metabólicas cuando se especifican todos los compuestos y reacciones necesarios para establecer la relación entre los compuestos de interés. El término camino metabólico se reserva para los casos en los que sólo se especifica un subconjunto de los compuestos/reacciones de la red metabólica usados para vincular los compuestos de interés.

El problema de búsqueda de caminos/vías metabólicas puede enmarcarse

[1]Kyoto Encyclopedia of Genes and Genomes. `http://www.genome.jp/kegg/`

dentro de la categoría de los problemas de enrutamiento, y enunciarse de la siguiente manera: dados el estado inicial y el estado final, encontrar una ruta o camino que satisfaga las restricciones especificadas [Russell and Norvig, 2010]. Formalmente, este problema se define a través de cinco elementos: el *estado inicial*, el conjunto de *acciones* posibles que permiten cambiar de estado, el *modelo de transición*, que indica el estado sucesor obtenido al aplicar una acción en un estado particular, la *prueba de meta*, que evalúa si se alcanza un estado objetivo, y la función *costo*, que asigna un valor al camino considerado. Hecha esta formulación, el espacio de estados queda definido y lo conforman todas las posibles secuencias de estados que, partiendo del estado inicial, pueden ser alcanzados empleando el *modelo de transición*. Una vez definido el problema, éste se resuelve mediante la construcción de un árbol de búsqueda, asociado al espacio de estados, y su exploración mediante alguna estrategia que indique la forma en que los nodos del árbol deben ser visitados. El problema principal que presentan estas estrategias es la forma en que se recorre el árbol. Si se considera, por ejemplo, el caso de un problema biológico donde cada nodo (compuesto) puede generar $b = 25$ nodos hijo en promedio, encontrar la solución para una profundidad $d = 5$ del árbol (equivalente a 5 reacciones) requerirá la exploración del orden de 10^8 estados empleando una estrategia clásica de búsqueda en amplitud. En cambio, si la solución se encuentra en el nivel $d = 7$, el número de estados se incrementará a $1.5 \cdot 10^{11}$. Suponiendo que el algoritmo sea capaz de explorar 10^5 nodos por segundo, se requerirían aproximadamente 7 horas para encontrar la solución en el primer caso, mientras que en el segundo se emplearían más de 184 días. También hay que tener en cuenta que el incremento en el número de nodos o la ramificación del árbol conduce a un fuerte incremento en el número de estados que deben ser explorados. Claramente, recorrer el árbol de búsqueda en forma exhaustiva no resulta práctico, por lo que se hace necesario desarrollar estrategias que agilicen esta tarea.

1.2. Estado del arte

Como se discutió en la Sección anterior, explorar completamente el árbol de búsqueda puede requerir una elevada carga computacional. Por ejemplo, empleando una estrategia de búsqueda en profundidad limitada a caminos

con hasta nueve reacciones, Küffner et al. [2000] realizaron la búsqueda de todos los posibles caminos metabólicos entre *glucosa* y *piruvato*. Como resultado encontraron aproximadamente $5 \cdot 10^5$ caminos, de los cuales una gran proporción no era biológicamente posible. Este trabajo puso en evidencia dos dificultades importantes. En primer lugar, la enumeración de todos los caminos entre dos compuestos es poco práctica. Además, disponer de todos los posibles caminos entre estos compuestos puede dificultar la interpretación los resultados. Una forma de simplificar el problema consiste en limitar la búsqueda a un pequeño subconjunto de soluciones. Siguiendo esta dirección se puede limitar la búsqueda a los $k-$caminos más cortos o los $k-$caminos de menor costo, esto último cuando se utilizan grafos ponderados. De esta manera, la búsqueda del camino más corto corresponde al valor $k = 1$. Cuando $k > 1$, la búsqueda procede hasta encontrar una solución, ésta se almacena y la búsqueda del siguiente camino continúa ignorando el último nodo explorado. En segundo lugar, la presencia de compuestos denominados secundarios o abundantes (tales como agua y adenosil trifosfato (ATP)) que poseen una elevada conectividad debido a que participan en un gran número de reacciones, son usados como atajos y conducen a soluciones sin validez biológica. Para salvar esta dificultad fue necesario desarrollar estrategias más sofisticadas. Una primera aproximación analizada por algunos autores fue la remoción de los compuestos altamente conectados, buscando mantener la cohesión de la vía metabólica [Huss and Holme, 2007; Gerlee et al., 2009]. Sin embargo, la elección de estos compuestos depende fuertemente de la red considerada, ya que puede ser que el compuesto sea removido de la vía donde se sintetiza. Un ejemplo típico es el ATP, el cual actúa como fuente de energía de un gran número de reacciones biológicas y posee una elevada conectividad. Si éste es descartado, se elimina la posibilidad de estudiar su síntesis. En consecuencia, se hace necesario considerar otras aproximaciones. La necesidad de encontrar un reducido número de caminos que contengan la mayor relevancia biológica posible, orientó los esfuerzos hacia el desarrollo de algoritmos que incorporaran mejoras en la definición de la función *sucesor* y de la función *costo*. Actualmente pueden identificarse, en forma general, cuatro estrategias diferentes.

La primera estrategia se enfoca en la función *costo* y consiste en asignar un peso a cada nodo del grafo, generalmente determinado por la conectividad del nodo. Con esto se logra que los nodos altamente conectados sean

fácilmente identificables y evitados si se aplican estrategias para buscar los $k-$caminos de menor costo. Aplicando esta propuesta en el algoritmo *Metabolic Pathfinding*, Croes et al. [2006] reprodujeron la síntesis del compuesto *lisina* a partir del compuesto *homocitrato* mediante la búsqueda de caminos sobre un grafo ponderado, asignando a cada compuesto un peso equivalente al número de reacciones en las que interviene.

La segunda estrategia consiste en seleccionar los sucesores empleando mapeos atómicos, los cuales identifican los átomos que son transferidos entre pares sustrato-producto en cada reacción. De esta forma, un sustrato puede estar relacionado con varios productos debido a que sus átomos son transferidos a más de un compuesto. Empleando este enfoque, [Arita, 2000] fueron capaces de realizar la búsqueda de los caminos más cortos entre dos compuestos, de manera de transferir al menos un átomo del compuesto inicial al compuesto final. En forma similar, Blum and Kohlbacher [2008] emplearon este enfoque combinado con una estrategia de asignación de pesos para realizar la búsqueda de los $k-$caminos de menor costo. Estos mapeos pueden ser divididos en cinco categorías, denominadas *pares reactantes*, de acuerdo a la función que cumplen en la reacción [Kotera et al., 2004a,b]. Por ejemplo, los pares sustrato-producto que experimentan las mayores transferencias se denominan *pares reactantes principales* (compuesto cuya estructura es más parecida al sustrato). Algunos algoritmos, tales como *PathComp* [Ogata et al., 1998] y *Linked Metabolites* [Easton et al., 2010] utilizan en forma directa los *pares reactantes principales* para dirigir la búsqueda hacia caminos que podrían ser biológicamente relevantes, ya que de esta forma se evitan los compuestos secundarios. Los *pares reactantes* también han sido utilizados por Faust et al. [2009], pero incorporando distintas estrategias de asignación de pesos. La combinación estos dos enfoques permitió la reconstrucción del 93 % de la red metabólica de la bacteria *Escherichia coli*. Estos autores extendieron su trabajo para realizar la búsqueda de vías metabólicas que relacionen un conjunto de compuestos. Basados en un enfoque similar, realizan la búsqueda de caminos entre todos los pares de compuestos a relacionar hasta una longitud máxima permitida, y los combinan para generar una vía metabólica que los relaciona [Faust et al., 2010].

La tercera estrategia es una extensión directa de la estrategia anterior, ya que utiliza el mapeo atómico para seguir el flujo de átomos entre los compuestos inicial y final. Este enfoque encuentra su fundamento en que

el compuesto que desea producirse debe contener al menos un átomo del compuesto inicial para que ambos presenten algún grado de conexión. Esta estrategia fue explotada por Pitkänen et al. [2009] en su algoritmo *ReTrace*. Este método busca caminos entre dos compuestos seleccionando los sucesores que maximizan la transferencia de átomos entre los compuestos a relacionar. Dado que este enfoque contempla el seguimiento del flujo de átomos, también es posible construir caminos ramificados mediante la combinación de caminos lineales. Basándose en esta misma idea Heath et al. [2010] también proponen un método que busca los caminos más cortos que maximizan la transferencia de átomos. Su propuesta garantiza que los caminos conservan un número mínimo de átomos entre los compuestos a relacionar y también permite seguir el flujo de átomos del compuesto inicial para identificar caminos ramificados.

La cuarta estrategia se basa en la selección de los sucesores a través de la comparación de las estructuras moleculares de los compuestos. De esta forma, la estructura del compuesto inicial es comparada con la de los productos de las reacciones que lo emplean como sustrato y se selecciona la que maximiza/minimiza alguna medida de similaridad/distancia. El algoritmo *PathMiner* [McShan et al., 2003] utiliza una representación de los compuestos basada en descriptores (vectores binarios que identifican conjuntos de átomos que caracterizan el compuesto) y una función para elegir los sucesores. La función se compone de una heurística que calcula la distancia entre el compuesto inicial y el compuesto actual, y una función de costo que estima la distancia al compuesto final, en ambos casos empleando la distancia de Manhatan. El algoritmo en *Pathway Hunter Tool* [Rahman et al., 2005] emplea una medida de similaridad para seleccionar el compuesto siguiente en la búsqueda. Esta medida usa la información estructural de los compuestos expresada como vectores binarios y realiza el cálculo contemplando la información contenida en el par de compuestos comparados y la contribución relativa de éstos en la reacción.

Estas cuatro estrategias mejoran sustancialmente los resultados de las búsquedas, proporcionando caminos biológicamente relevantes. Sin embargo, estos enfoques trabajan bajo dos suposiciones: los caminos más cortos son los que tienen mayor relevancia y los compuestos secundarios deben ser evitados. Aunque los caminos más cortos probablemente estén optimizados y sean los más eficientes, el estudio de caminos de mayor longitud podría brindar información sobre nuevas interconexiones entre procesos no conoci-

das previamente. Además, la incorporación de los compuestos secundarios permite un modelado más realista del problema, facilitando la extracción de conclusiones. Tampoco debe olvidarse que algunas de las aproximaciones descriptas previamente requieren de la estructura molecular de los compuestos para guiar la búsqueda. Esto puede ser un problema ya que no siempre se dispone de dicha estructura para todos los compuestos.

1.3. Objetivos de la tesis

Tomando en consideración lo presentado previamente, en esta tesis se plantea la búsqueda de caminos y vías metabólicas que relacionen un grupo de compuestos sin el uso de información *a priori*, tal como las enzimas que participan en el proceso o las vías metabólicas de referencia donde los compuestos desempeñan alguna función.

La principal contribución realizada consiste en el desarrollo de métodos de búsqueda basados en algoritmos evolutivos que permiten encontrar relaciones entre dos o más compuestos empleando sólo la información de las reacciones que componen el espacio de búsqueda. Además, estos métodos son capaces de buscar vías metabólicas con diferentes topologías, seleccionando de forma automática el compuesto con el que iniciar la búsqueda. A diferencia de los métodos que existen actualmente, los algoritmos propuestos encuentran soluciones que pueden contener mayor número de reacciones que las soluciones más cortas posibles, favoreciendo la exploración de mecanismos alternativos para la síntesis de compuestos.

En base a esto, los objetivos perseguidos en esta tesis se resumen en los siguientes puntos:

- Desarrollar nuevos algoritmos evolutivos para búsqueda de relaciones entre conjuntos de compuestos metabólicos.
- Estudiar distintas codificaciones para los cromosomas, contemplando relaciones tales como secuencias, árboles y/o redes entre compuestos.
- Diseñar operadores genéticos que incorporen conocimiento del dominio del problema.
- Explorar diferentes medidas para construir la función de aptitud, evaluando características específicas de las soluciones buscadas.

- Proponer medidas apropiadas para el análisis y comparación de vías metabólicas.
- Evaluar las propuestas empleando redes metabólicas conocidas extraídas de bases de datos biológicas de dominio público.
- Implementar métodos de visualización que provean una fácil interpretación de las relaciones descubiertas.

1.4. Organización de la tesis

Esta tesis está estructurada de acuerdo a los siguientes capítulos:

- En el presente capítulo se describió la motivación de esta investigación y el problema que será abordado. Se introducen algunos conceptos biológicos necesarios para entender el problema y se presenta una revisión del estado del arte de las técnicas actualmente utilizadas. Finalmente, se detallan los objetivos perseguidos en esta tesis y se describe la estructura de capítulos.
- El Capítulo 2 presenta el marco teórico para las técnicas empleadas en esta tesis. Inicialmente se define de manera formal el problema de búsqueda y optimización. Luego se introducen las técnicas clásicas de búsquedas, estableciendo las diferencias entre las búsquedas informadas y las no informadas. A continuación, se presenta la familia de algoritmos de computación evolutiva, describiendo un marco teórico que los sustenta y sus elementos característicos. El capítulo finaliza haciendo una comparación entre las técnicas clásicas de búsqueda y los algoritmos evolutivos.
- El Capítulo 3 presenta el primer aporte de esta tesis, proponiendo el uso de un algoritmo evolutivo para la búsqueda de caminos metabólicos. A partir de una estructura típica de algoritmo evolutivo se introducen modificaciones en los operadores de cruza y mutación, y se discuten las diferencias frente a los operadores clásicos. Luego se analiza la función de aptitud, diseñada para contemplar aspectos característicos de una secuencia de reacciones químicas, en términos de las medidas que la componen. En la parte final del capítulo se presentan los resultados

alcanzados durante la evaluación del desempeño del algoritmo y la comparación realizada con dos algoritmos clásicos de búsqueda. También se analizan dos casos de estudio.

- El Capítulo 4 presenta una propuesta para la búsqueda de vías metabólicas ramificadas empleando algoritmos evolutivos. Primero se introduce una propuesta para modelar la codificación de las vías metabólicas en cromosomas lineales, mediante el uso de conjuntos de compuestos. También se describe el algoritmo evolutivo propuesto y los diferentes operadores empleados, señalando en cada caso las diferencias con los operadores clásicos. Luego se realiza el análisis y justificación de las medidas empleadas en la función de aptitud. A continuación se describe el conjunto de datos empleado y las medidas propuestas para evaluar el desempeño y las propiedades de las vías metabólicas en los diferentes experimentos. El capítulo finaliza presentando los experimentos realizados y discutiendo los resultados conseguidos, incluyendo también dos casos de estudio.

- El Capítulo 5 presenta las conclusiones particulares y generales de los algoritmos desarrollados, discutiendo diferentes propuestas para trabajos futuros. Además, se listan las publicaciones logradas como fruto de esta tesis.

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
M. Gerard; "Desarrollo de algoritmos evolutivos para el descubrimiento de relaciones en datos metabólicos"
Universidad Nacional del Litoral, 2014.

Capítulo 2

Marco teórico

En este capítulo se describen los elementos de un problema de búsqueda y se presentan las características de las técnicas de búsqueda clásicas y de las técnicas metaheurísticas provenientes de la computación evolutiva. En una primera parte se presentan los métodos clásicos de búsqueda, la formulación del problema y una descripción general del proceso de exploración. En particular, se presentan las estrategias de búsqueda sin uso de información del problema y se analizan dos estrategias características de esta familia de algoritmos. Además se presentan los métodos que emplean información, analizando sus propiedades y analizando brevemente sus dificultades. En una segunda parte se presentan las técnicas asociadas a la computación evolutiva. Primero se enumeran las características de estas técnicas y sus ventajas frente a otros métodos de búsqueda. Luego se describe el algoritmo general para este conjunto de técnicas y se analizan los elementos que lo conforman, brindando algunos criterios para su elección.

2.1. Métodos clásicos de búsqueda

El primer paso cuando se aborda un problema de búsqueda consiste en la correcta formulación del objetivo que se persigue. Así es posible concentrar los recursos disponibles en aquellas opciones que resultan más factibles para satisfacer el objetivo planteado. Una vez definido el objetivo, el siguiente

paso consiste en realizar la formulación del problema [Russell and Norvig, 2010]. Formalmente, un problema puede definirse mediante cinco elementos:

- **Estado**: especifica un conjunto de valores particulares para todas las variables que definen el problema que se está abordando. Dos instancias particulares de este elemento son necesarias en la definición del problema. El estado *inicial* (s) indica el estado a partir del cual comienza la búsqueda, mientras que el estado *objetivo* o *final* indica uno o más estados que cumplen con algún requisito particular.

- **Acciones**: describe el conjunto de todas las transiciones entre estados posibles para el problema bajo estudio. En particular, es importante conocer el subconjunto de acciones factibles A que pueden aplicarse a un estado s particular.

- **Modelo de transición**: especifica el resultado de aplicar cada acción factible $a \in A$. Este modelo se caracteriza a través de una función $\mathcal{R}(s, a)$ que devuelve el estado alcanzado luego de aplicar una acción a estando en el estado s.

- **Prueba de meta**: evalúa si un estado dado corresponde a un estado objetivo del problema. En algunos casos, se dispone de un conjunto explícito de estados objetivos y la *prueba de meta* sólo evalúa si un estado particular pertenece a ese grupo. En otros casos, la *prueba de meta* simplemente evalúa si el estado cumple con alguna condición especificada.

- **Función de costo**: asigna un costo a cada secuencia de acciones del estado inicial al final, basado en el costo individual de cada estado o en el costo asociado a la transición entre dos estados.

Una vez presentadas estas definiciones, el espacio de estados queda especificado de forma automática a partir de los tres primeros elementos, como el conjunto de todos los posibles estados que pueden ser alcanzados a partir del estado inicial mediante cualquier secuencia factible de acciones. En este contexto, un *camino* es una secuencia de acciones factibles que llevan del estado inicial al estado objetivo. Así, el espacio de estados puede ser explorado analizando todas las posibles secuencias de acciones que parten del estado inicial. Sin embargo, explorar el espacio de estados en forma exhaustiva no

es la forma más eficiente para encontrar un camino entre los estados inicial y final.

Una forma de abordar esta tarea es mediante la construcción de un árbol de búsqueda, cuya raíz sea el nodo inicial, las ramas son las acciones posibles y los nodos restantes son los estados alcanzables mediante la combinación de diferentes acciones [Poole and Mackworth, 2010]. La búsqueda de una solución consiste en un proceso de dos etapas que se suceden en forma iterativa. La primera consiste en seleccionar un nodo del árbol empleando alguna *estrategia de búsqueda* y evaluar si dicho nodo corresponde a un estado objetivo. En caso de no satisfacer esta condición, la etapa siguiente consiste en aplicar sobre este nodo todas las acciones posibles que conduzcan a un nuevo estado (expansión del nodo) y adicionar los nuevos nodos al árbol para continuar la búsqueda. El conjunto de todos los nodos hoja del árbol conforman la *frontera de búsqueda*. Este proceso es realizado hasta encontrar una solución o hasta que no es posible continuar expandiendo la frontera (árbol explorado completamente).

Aunque el espacio de estados puede contener un número finito de estados, el árbol de búsqueda puede ser infinito debido a que un mismo estado puede ser explorado más de una vez (caminos cíclicos). Los caminos cíclicos forman parte de un conjunto más amplio denominado *caminos redundantes*, que son típicos para los casos en donde existe más de un camino que vincula dos estados, y se caracterizan por contener un número de pasos mayor que el mínimo necesario para producir la transición entre los estados.

Durante la búsqueda, es necesario conservar cierta información en cada nodo del árbol para poder reconstruir la secuencia que conduce del estado inicial al objetivo una vez que es alcanzado. Para cada nodo es necesario preservar la información siguiente: *estado*, *nodo padre*, *acción* y *costo acumulado*. El *estado* preserva la información del estado específico que el nodo representa. El *nodo padre* indica el nodo que fue expandido para producir el nodo actual, mientras que *acción* contiene la información acerca de la acción específica aplicada al *nodo padre* para generar el nodo actual. Finalmente, el *costo acumulado* almacena el costo asociado a la secuencia de acciones que conducen del nodo raíz al nodo actual. Es importante remarcar que nodos y estados no son lo mismo y no deben ser confundidos. Si bien cada nodo representa un estado, dos nodos diferentes del árbol pueden representar un mismo estado, alcanzado en cada caso mediante diferentes secuencias de

acciones [Russell and Norvig, 2010].

Una estructura de datos adecuada para almacenar en forma ordenada los nodos de la frontera durante el proceso de búsqueda es la *lista*. De acuerdo a la forma de extraer los elementos almacenados se distinguen tres variantes más comunes: *cola* (*FIFO*, del inglés First In - First Out), *pila* (*LIFO*, del inglés Last In - First Out) y *lista con prioridad*. Al emplear una *cola*, los elementos se extraen de la lista comenzando por el que posee mayor antigüedad. En cambio, cuando se emplea una *pila*, los elementos son extraídos de forma cronológica pero comenzando por el más reciente. Cuando se emplea una *lista con prioridad*, los elementos se almacenan en la lista de forma ordenada siguiendo alguna función o criterio de prioridad, y los elementos se extraen de acuerdo a la prioridad que tienen asignada.

La forma en que pueden elegirse los nodos a expandir produce un fuerte efecto en el comportamiento del algoritmo, por lo que se hace necesario contar con algún criterio para evaluar su desempeño. Comúnmente se tienen en cuenta cuatro elementos:

- *Completitud*: ¿El algoritmo es capaz de encontrar la solución?
- *Optimalidad*: ¿Proporciona la solución óptima (camino con menor costo de entre todas las posibles alternativas)?
- *Complejidad temporal*: ¿Cuánto tiempo requiere el algoritmo para encontrar la solución?
- *Complejidad espacial*: ¿Cuánta memoria es necesaria para realizar la búsqueda?

Cuando el espacio de estados se representa implícitamente a través de la definición de un estado inicial, acciones y un modelo de transición, el tamaño del árbol modelado puede ser infinito. Cuando el espacio de estados se recorre mediante un árbol, la complejidad temporal y espacial puede relacionarse directamente con el número total de nodos y enlaces del mismo. De esta forma, la complejidad puede expresarse en término de tres cantidades: b, que representa el factor de ramificación (número máximo de sucesores para cualquier nodo); d, profundidad mínima a la que se encuentra un nodo asociado a un estado objetivo; m, el número de pasos necesarios para alcanzar el nodo objetivo más profundo. Por su parte, el tiempo se evalúa en términos

del número de nodos expandidos durante la búsqueda y el espacio en función del máximo número de nodos que deben ser almacenados en memoria.

En las siguientes subsecciones se describen las estrategias de búsqueda informada y no informada, exponiendo algunos de los algoritmos que caracterizan a estas estrategias. En cada caso se presentan los pasos seguidos en cada algoritmo, se analizan sus características y particularidades, y se detalla el cálculo de la complejidad.

2.1.1. Búsqueda no informada

Las estrategias de búsqueda no informada sólo disponen de la información provista en la definición del problema y no disponen de información adicional que permita identificar caminos potencialmente mejores [Kumar, 2008]. En esta familia, sus miembros se diferencian por el orden en que expanden los nodos durante la búsqueda. Dos de las estrategias más comúnmente empleadas son las de amplitud y profundidad.

2.1.1.1. Búsqueda en amplitud

La estrategia de *búsqueda primero en amplitud* (*BFS*, del inglés Breadth First Search) consiste en expandir primero todos los nodos para un mismo nivel d del árbol antes de expandir los del nivel siguiente $d+1$. Una forma sencilla de implementar esta estrategia es mediante el uso de una *cola*, que permite mantener los nodos de la frontera que se encuentran a menor profundidad en la parte superior para ser expandidos primero. De esta forma, BFS siempre conserva los caminos más cortos a cada nodo de la frontera [Cormen et al., 2001].

El Algoritmo 2.1 presenta los pasos seguidos para generar el árbol de búsqueda utilizando BFS. El algoritmo comienza con la inicialización de la frontera como una cola vacía. Luego se comprueba si el nodo inicial corresponde a un estado objetivo y, si se verifica, el algoritmo se detiene ya que se habrá encontrado la solución. A continuación se ingresa en un ciclo que se repite hasta que se encuentra un camino entre el estado inicial y el final o hasta que no es posible seguir expandiendo nodos del árbol. El primer paso de este ciclo consiste en extraer de la cola el nodo más superficial. A partir

Algoritmo 2.1: Algoritmo de búsqueda en amplitud.

```
1  n ⟵ nodo inicial
2  f ⟵ inicializar frontera como una cola vacía
3  if n es un estado objetivo then
4  |  c ⟵ n
5  else
6  |  f ⟵ incorporar n a la frontera
7  |  while f no esté vacía and c esté vacío do
8  |  |  n ⟵ extraer de f el nodo n con menor profundidad
9  |  |  q ⟵ expandir nodo n para generar los nodos hijo
10 |  |  while q no esté vacía and c esté vacío do
11 |  |  |  m ⟵ extraer un nodo de q
12 |  |  |  if m no pertenece a la frontera then
13 |  |  |  |  if m es un estado objetivo then
14 |  |  |  |  |  c ⟵ reconstruir camino de m al nodo inicial
15 |  |  |  |  else
16 |  |  |  |  |  f ⟵ incorporar m a la frontera
17 return c
```

de este punto el algoritmo puede proceder siguiendo dos secuencias de pasos diferentes, donde una variante presenta mayor complejidad que la otra. Una variante consiste en evaluar el nodo extraído de la frontera y expandirlo en caso de que no corresponda a un estado objetivo. La otra variante consiste en expandir primero el nodo y luego evaluar si alguno de los hijos corresponde a un estado objetivo antes de incorporarlos a la frontera. Esta última es la variante presentada en el Algoritmo 2.1. El efecto que estas variantes introducen sobre su complejidad algorítmica se describirán en detalle cuando se analice la complejidad espacial y temporal.

La Figura 2.1 presenta un árbol de búsqueda y la secuencia en que se expanden/exploran los nodos de la frontera (línea de trazos) utilizando BFS. Inicialmente la frontera contiene al nodo raíz A, con profundidad $d = 0$. Luego de ser evaluado, se expanden sus nodos hijos en el nivel $d = 1$, correspondientes a las hojas B y C. En este punto, ambos nodos forman parte de la frontera de búsqueda. Primero se expande el nodo B y sus hijos, con profundidad $d = 2$, son evaluados. Luego se procede de la misma manera con el nodo C. Una vez evaluados los hijos de C y no habiendo alcanzado un estado objetivo, la frontera queda conformada sólo por los nodos D, E,

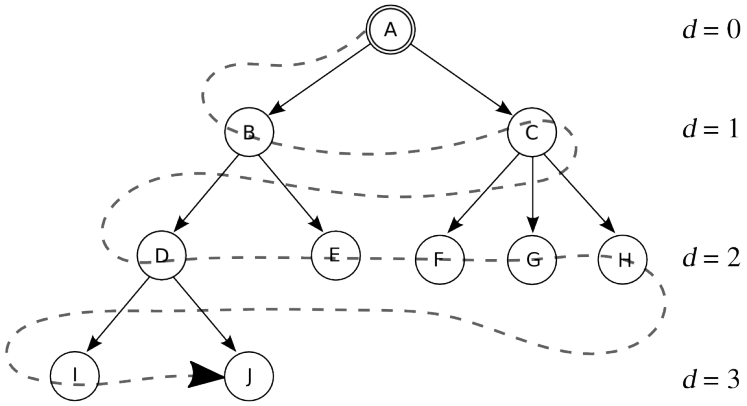


Figura 2.1: Recorrido de un árbol de búsqueda empleando la estrategia de búsqueda en amplitud. La línea de trazos indica el orden en que se generan los nodos del árbol.

ya que F, G, H no generan hijos (no pueden ser expandidos) y son retirados de la frontera. Siguiendo esta lógica se podría continuar con la exploración del espacio de soluciones para el nivel $d = 3$.

El análisis del desempeño de este algoritmo presenta resultados interesantes. Fácilmente puede apreciarse que esta estrategia cumple con el requisito de completitud, ya que si un nodo solución se encuentra a una profundidad finita, esta estrategia expandirá todos los nodos más superficiales hasta alcanzarlo. Además, dado que cada nodo es evaluado cuando es generado, la solución también será óptima para el caso en que la función de costo del camino sea no decreciente con la profundidad. Los escenarios más comunes contemplan el mismo costo para todas las acciones, por lo que en estos casos está garantizada la optimalidad.

La complejidad temporal está asociada al valor de ramificación b y la profundidad d a la cual se encuentra la solución, debido a que deberán expandirse todos los nodos hasta el nivel d y generarán, en el peor caso, $b + b^2 + b^3 + \cdots + b^d$ nodos durante la búsqueda. Como consecuencia, la complejidad temporal será $\mathcal{O}(b^d)$. Como se mencionó antes, la ventaja de evaluar los nodos cuando son generados evita tener que generar los nodos del nivel $d + 1$ antes de encontrar la solución, lo que en otro caso tendría asociada una complejidad $\mathcal{O}(b^{d+1})$. La complejidad espacial también está dominada por el tamaño de la frontera siendo $\mathcal{O}(b^d)$, ya que cada nodo generado es conservado en memoria. Como consecuencia, la principal limitación práctica

Algoritmo 2.2: Algoritmo de búsqueda en profundidad.

```
1  n ⟵ nodo inicial
2  f ⟵ inicializar frontera como una pila vacía
3  if n es un estado objetivo then
4  |  c ⟵ n
5  else
6  |  f ⟵ incorporar n a la frontera
7  |  while f no esté vacía and c esté vacío do
8  |  |  n ⟵ extraer de f el nodo n con mayor profundidad
9  |  |  if n es un estado objetivo then
10 |  |  |  c ⟵ reconstruir el camino de n al nodo inicial
11 |  |  else
12 |  |  |  q ⟵ expandir nodo n para generar los nodos hijo
13 |  |  |  foreach m en q do
14 |  |  |  |  if m no forma parte del camino actual then
15 |  |  |  |  |  f ⟵ incorporar m a la frontera
16 return c
```

de esta estrategia es el rápido incremento en el consumo de memoria para almacenar los nodos, incluso para búsquedas poco profundas.

2.1.1.2. Búsqueda en profundidad

La estrategia de *búsqueda en profundidad* (*DFS*, del inglés Depth First Search) prioriza la expansión del nodo más profundo de la frontera. Cuando éste es alcanzado y no se dispone de sucesores para continuar la expansión, el nodo es retirado de la frontera y la estrategia continúa la expansión del siguiente nodo más profundo. Esta estrategia emplea una *pila* para mantener en la parte superior de la lista los nodos más recientes (de mayor profundidad) que serán expandidos [Cormen et al., 2001]. Además, es común incluir en la implementación de esta estrategia una etapa de verificación para evitar estados repetidos que podrían conducir a caminos con ciclos infinitos. Así, primero se determinan los sucesores del nodo y luego se evalúan dichos sucesores para descartar aquellos nodos asociados a estados que ya forman parte del camino. Si bien esta modificación permite evitar caminos cíclicos, no elimina la posibilidad de encontrar caminos redundantes.

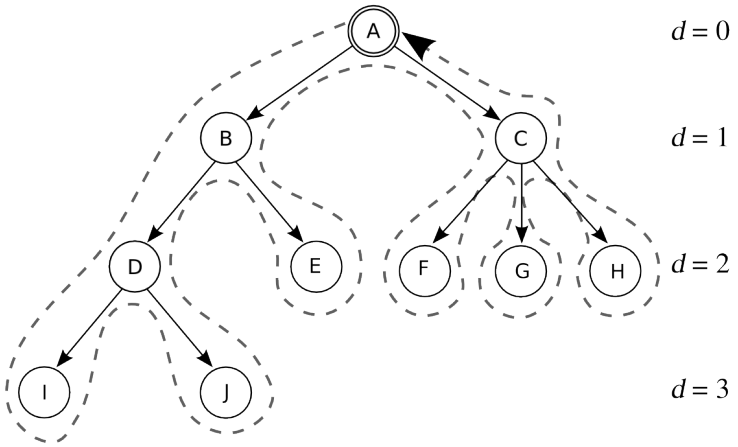


Figura 2.2: Recorrido de un árbol de búsqueda empleando la estrategia de búsqueda en profundidad. La línea de trazos indica el orden en que se generan los nodos del árbol.

El Algoritmo 2.2 presenta los pasos seguidos para generar el árbol de búsqueda utilizando DFS. El algoritmo comienza con la inicialización de la frontera como una pila vacía. Luego se comprueba si el nodo inicial corresponde a un estado objetivo y, si se verifica, el algoritmo se detiene ya que se habrá encontrado la solución. A continuación se ingresa en un ciclo que se detiene cuando no es posible seguir expandiendo nodos del árbol o se encuentra un camino entre el estado inicial y el final. En cada iteración se extrae el nodo más profundo del árbol (que forma parte de la frontera de búsqueda), se lo evaluá para verificar si es un estado objetivo y se lo expande en caso de no serlo. Los nodos hijos generados, que no forman parte del camino entre el nodo raíz y el nodo padre que los generó, son almacenados en la frontera.

La Figura 2.2 presenta el árbol de búsqueda y la secuencia en que se expanden/exploran los nodos de la frontera (línea de trazos) utilizando la estrategia de DFS. Desde el inicio, la búsqueda procede inmediatamente hasta el nodo I que es el nodo más profundo ($d = 3$) del árbol, evaluando y expandiendo sólo un nodo en cada nivel pero preservando los restantes como parte de la frontera de búsqueda. Debido a que I no puede ser expandido, éste se remueve de la frontera y se selecciona el siguiente nodo con mayor profundidad. En este ejemplo, el nodo J se encuentra al mismo nivel que I, por lo que es seleccionado para continuar la búsqueda. Una vez que este

nodo es evaluado y habiendo determinado que no puede ser expandido es retirado de la frontera junto con el nodo D ya explorado, dejando al nodo E con profundidad $d = 2$ como el nodo más profundo en la lista. Nuevamente, luego de evaluarlo y retirarlo de la frontera (no puede ser expandido), el subárbol izquierdo con raíz en el nodo A ya ha sido explorado completamente (el nodo B también es retirado de la frontera), por lo que el siguiente nodo a expandir es C, con profundidad $d = 1$. Continuando con la lógica de exploración seguida hasta el momento, los siguientes nodos a explorar serán en forma ordenada F, G y H. Para este ejemplo dichos nodos no pueden ser expandidos por lo que la búsqueda concluye cuando la frontera queda vacía.

Esta estrategia presenta la desventaja de no cumplir con el criterio de completitud. Si el espacio de estados es finito, la verificación de estados ya explorados en el camino evita ciclos pero no los caminos redundantes. En espacios de estado infinitos, la estrategia falla si la búsqueda conduce a un camino infinito que no contiene una solución. Nuevamente, teniendo en cuenta que una solución óptima posee el menor costo de entre todas las soluciones, esta estrategia no es óptima. Esto se debe a que las soluciones son buscadas en el nivel más profundo del árbol antes que en niveles más cercanos a la superficie.

La complejidad temporal depende fuertemente del tamaño del espacio de estados. Esta estrategia es capaz de generar los $\mathcal{O}(b^m)$ nodos para encontrar la solución. Debe tenerse en cuenta que m puede ser mucho mayor que la profundidad de la solución d más superficial (cada posible transición entre estados genera un nuevo nodo del árbol) y en el caso de espacios de estados infinitos la profundidad del árbol no estará acotada. Como contrapartida, esta estrategia posee una baja complejidad espacial ya que sólo se requiere almacenar los nodos del camino que conduce del nodo raíz al nodo hoja más profundo considerado, y los nodos aún no expandidos en cada nivel. Además, una vez que se ha alcanzado un nodo que no puede expandirse, éste es removido directamente de memoria. De esta forma, la cantidad de nodos que DFS debe conservar en memoria, para un espacio de estados con una ramificación b y una profundidad máxima d, está acotado a $\mathcal{O}(bm)$.

2.1.2. Búsqueda informada

Las estrategias de búsqueda informada se caracterizan por disponer de algún método para evaluar la calidad de los caminos y expandir el nodo que tiene más posibilidades de conducir al objetivo buscado. En general, estas estrategias emplean conocimiento específico del problema, adicional a la información provista durante la definición del problema, para realizar la búsqueda de manera más eficiente [Cormen et al., 2001]. El enfoque más general se denomina estrategia de *primero el mejor*. La información se incorpora en la estrategia mediante una función de evaluación $f(n)$, que estima el costo del camino generado al expandir el nodo n. Estos nodos pueden así ser almacenados empleando una *lista ordenada* de acuerdo a su costo asociado, de manera de mantener un registro ordenado de los nodos de acuerdo a su costo.

La elección de la función de evaluación determina el tipo de estrategia que será empleada. Muchos de los algoritmos basados en la estrategia de *primero el mejor* emplean una función heurística $h(n)$ para estimar el costo del camino desde el nodo actual n hasta el nodo objetivo. Esta función sólo requiere conocer el estado asociado al nodo evaluado, y es la forma más habitual de incorporar información del dominio del problema al algoritmo de búsqueda. En forma general, es necesario que ésta sea no negativa, diseñada específicamente para cada problema y para la cual su valor sea cero cuando se alcanza un nodo objetivo.

Existen dos variantes comunes dentro de las estrategias de búsqueda *primero el mejor*, denominadas *búsqueda avara* y *búsqueda* A^*. Ambas se diferencian en la manera en que la función de evaluación está compuesta, pero comparten el objetivo de optimizar el proceso de búsqueda mediante la elección del nodo a expandir que minimice el valor de la función elegida [Russell and Norvig, 2010].

La *búsqueda avara* emplea $f(n) = h(n)$ como función de evaluación, y busca minimizar esta heurística expandiendo el nodo más cercano a un nodo objetivo. Debido a que sólo se estima el valor del camino para alcanzar el nodo objetivo, los caminos encontrados pueden ser redundantes (con mayor valor que el de otros caminos posibles). La complejidad espacial y temporal para esta estrategia en el peor caso es simplemente $\mathcal{O}(b^m)$ y corresponde a la situación donde debe recorrerse el árbol en forma completa. Aunque

esta estrategia puede reducir sustancialmente la complejidad si se elige una heurística apropiada para el problema, la búsqueda es incompleta. Incluso para un espacio de estados finito el algoritmo podría explorar un camino infinito que contiene un bucle, por ejemplo, si alcanza un nodo no expandible que presenta la menor distancia al objetivo que cualquier otro nodo en la frontera.

La estrategia de búsqueda A* es una versión mejorada de la estrategia anterior, ya que redefine la función de evaluación para incorporar un término adicional. Ésta función queda definida como $f(n) = g(n) + h(n)$, donde $h(n)$ estima el valor del camino más barato al nodo objetivo y el nuevo término $g(n)$ evalúa el costo acumulado del camino desde el nodo raíz hasta el nodo considerado. Esta estrategia es óptima y completa cuando se emplean árboles y la función heurística $h(n)$ nunca sobreestima el valor del camino que conduce a la solución. Aunque este algoritmo es completo, se requiere de la existencia de un número finito de nodos cuyo costo $f(n)$ sea menor o igual al del camino óptimo. Esto se cumple cuando el factor de ramificación y el costo de cada acción tienen valores finitos, limitando así el número de estados que deben ser explorados. Como consecuencia de esto, el algoritmo es capaz de podar el árbol de búsqueda descartando aquellos sub-árboles cuyo nodo raíz tiene un costo mayor que el camino óptimo. Si bien una heurística más precisa mejora el proceso de búsqueda, la complejidad está fuertemente influenciada por las hipótesis hechas acerca del espacio de estados (uno o múltiples estados objetivo). Una importante limitación que posee este algoritmo es el consumo de memoria ya que mantiene en memoria todos los nodos generados. Esto lo hace poco práctico para problemas donde deben explorarse espacios de gran tamaño. Además, es necesario disponer de información que permita caracterizar adecuadamente el estado objetivo para estimar correctamente el costo de alcanzarlo.

2.2. Computación evolutiva

La Computación Evolutiva (CE) es una rama de la Inteligencia Computacional compuesta por un conjunto de técnicas de optimización metaheurística inspiradas en la evolución. De acuerdo a la *Teoría de la selección natural* postulada por Charles Darwin, la evolución de los organismos está dada por la supervivencia y reproducción de los individuos que mejor se adaptan al

medio en el que se encuentran, dado que continuamente éstos compiten por el espacio y los recursos disponibles. Cada organismo contiene en sus genes información que lo caracteriza y que le permite desarrollar sus funciones. Esta información es la que define la capacidad de supervivencia y que es transmitida a la descendencia. Cuando un gen proporciona una ventaja evolutiva, éste es transmitido de una generación a la siguiente de manera que se perpetúa su presencia en el tiempo. Adicionalmente, la información contenida en los genes sufre en algunas ocasiones pequeñas modificaciones durante la producción de la descendencia. Estas modificaciones ocurren en forma aleatoria y si se traducen en un aumento en la capacidad de supervivencia del organismo, éstas se conservan y también son transmitidas a la descendencia [Holland, 1975; Engelbrecht, 2007].

Las técnicas englobadas por la CE se emplean para la resolución de una amplia variedad de problemas de optimización y búsqueda. Se basan en la evolución de una población de individuos, donde cada uno representa una solución potencial del problema, guiada por una función de aptitud que evalúa la capacidad de supervivencia de cada individuo. El proceso de evolución es simulado mediante la aplicación de operadores de cruza y mutación. El operador de cruza combina el material genético de los padres para generar un nuevo individuo, mientras que el operador de mutación introduce pequeñas modificaciones que permiten añadir ventajas potenciales en la descendencia. El Algoritmo 2.3 describe las etapas generales de los algoritmos evolutivos (AE). El primer paso consiste en la generación e inicialización de las soluciones potenciales del problema. Luego, cada individuo es evaluado para determinar su capacidad de supervivencia. A continuación, se comienza un proceso cíclico que se detiene cuando se cumple algún criterio de finalización preestablecido de acuerdo a las soluciones buscadas. En primer lugar se seleccionan los individuos que serán los padres de la nueva generación. Posteriormente se realiza la cruza y se genera la descendencia, la cual es sometida a mutaciones con una baja probabilidad. Finalmente se produce la población que conformará la siguiente generación y sus individuos son evaluados para determinar la aptitud.

La evolución de una dada población mediante la selección de sus individuos puede interpretarse como un proceso de búsqueda a través de un espacio de soluciones definido a partir de todos los posibles valores almacenados en los cromosomas. De esta manera, los AE pueden verse como procesos

Algoritmo 2.3: Estructura general de un algoritmo evolutivo.

```
1  Inicializar el contador G de generaciones en cero (G = 0)
2  A^ϵ ← mínima aptitud tolerada
3  P^0 ← inicializar la población
4  A(c_i)_{∀c_i∈P^0} ← evaluar la aptitud de los individuos en P^0
5  while A(c_i)_{∀c_i∈P^G} < A^ϵ do
6  |  seleccionar los padres para la nueva generación (P^{G+1})
7  |  generar la descendencia
8  |  P^{G+1} ← seleccionar los individuos para la nueva población
9  |  G ← G + 1
10 |  A(c_i)_{∀c_i∈P^G} ← evaluar la aptitud de los individuos en P^G
```

estocásticos de búsqueda para recorrer el espacio de soluciones. El comportamiento de estos algoritmos está fuertemente influenciado por los siguientes elementos:

- **Representación**: determina la forma en que las soluciones del problema serán representadas en los cromosomas ($\mathbf{c}_i$).
- **Población inicial** ($\mathcal{P}^0$): realiza el muestreo inicial del espacio de soluciones.
- **Función de aptitud** ($\mathcal{A}(\cdot)$): evalúa la calidad o la capacidad de supervivencia de los individuos.
- **Selección**: determinan los individuos que producirán descendencia y los que conformarán la población de la siguiente generación.
- **Reproducción y reemplazo**: determinan la información que es transmitida a la descendencia.
- **Criterios de finalización**: especifican las condiciones que deben cumplirse para finalizar la ejecución del algoritmo.

La manera en que cada uno de estos elementos es implementada conduce a diferentes paradigmas dentro de la CE. Las cuatro familias más representativas son los Algoritmos Genéticos [Holland, 1975; Goldberg, 1997], las Estrategias Evolutivas [Rechenberg, 1973], la Programación Genética [Koza, 1992] y la Programación Evolutiva [Fogel et al., 1966]. Cada una nació impulsada por diferentes motivaciones y se diferencian principalmente en los

esquemas de representación y en los operadores de selección y reproducción [Hammel and Schwefel, 1997]. Los algoritmos genéticos se caracterizan por emplear cromosomas que utilizan codificación binaria, los cuales poseen un elevado número de genes pero pocos alelos. Las estrategias evolutivas utilizan una codificación fenotípica del problema, representando valores reales en cada gen. Esta representación utiliza pocos genes y un elevado número de alelos. Además, la convergencia es fuertemente dependiente de los operadores empleados. La programación genética codifica los problemas utilizando árboles, y las variaciones son introducidas mediante intercambio de ramas y modificación de la información contenida en los nodos. Por su parte, la programación evolutiva se utiliza, por ejemplo, para evolucionar autómatas de estados finitos.

Estos algoritmos se caracterizan principalmente por ser técnicas conceptualmente simples y de propósito general, que no requieren conocimiento específico del problema para ser aplicadas. Esta sencillez brinda un desempeño robusto frente a entornos complejos y cambios dinámicos que pueden producirse durante la evolución [Sivanandam and Deepa, 2008]. El uso de una población de soluciones candidatas les permite explorar en forma simultánea diferentes puntos del espacio de soluciones, haciéndolos una alternativa interesante para problemas asociados a espacios con elevadas dimensiones. Adicionalmente, el uso de una población también permite manipular distintas familias de soluciones en problemas multimodales.

Otro elemento que caracteriza a estas técnicas es la capacidad de encontrar en tiempos razonables soluciones suficientemente buenas en términos de alguna medida de error, que se aproximan a la solución óptima principalmente cuando el problema presenta un elevado número de máximos locales. En todos los casos, la exploración se realiza empleando reglas probabilísticas para explotar la información almacenada en cada población. Además, la evolución es guiada por una función de aptitud que requiere pocos requisitos formales y que puede formularse para que la búsqueda esté orientada a satisfacer múltiples objetivos simultáneamente [Rutkowski, 2008].

Habiendo presentado en forma general los elementos que componen un algoritmo evolutivo, en las siguientes subsecciones se describirán en detalle cada uno de sus componentes, junto con algunas instancias de éstos empleadas por los distintos miembros de CE.

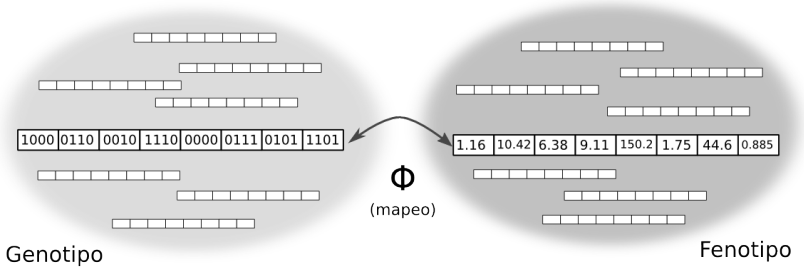


Figura 2.3: Mapeo entre el espacio de los genotipos y de los fenotipos para un algoritmo genético.

2.2.1. Representación

Los organismos poseen conjuntos de características que determinan su habilidad para sobrevivir y para reproducirse, las cuales se almacenan en estructuras que conforman largas cadenas de información denominadas cromosomas. Estas estructuras están organizadas en genes, que corresponde a paquetes de información acerca de alguna característica heredable. A su vez, cada una de las formas alternativas para un dado *gen* se conocen como *alelo*. La composición genética o *genotipo* de cada individuo es única y determina sus características morfológicas y fisiológicas (fenotipo) [Rothlauf, 2006].

En el contexto de la CE, cada individuo representa una solución potencial y es la composición genética la que se evoluciona para encontrar la solución. Dependiendo del dominio o las características del problema, cada *alelo* tomará uno de los posibles valores asociados a un gen particular para determinar el *genotipo* del individuo. El *fenotipo* asociado corresponde al comportamiento reflejado por ese *genotipo* en un entorno particular. La Figura 2.3 presenta un ejemplo de la relación entre el *genotipo* y el *fenotipo* de un individuo para un algoritmo genético. El espacio de todos los posibles *genotipos* está formado por cromosomas conteniendo 8 genes y cada gen corresponde a una secuencia de 4 bits. El mapeo Φ transforma cada gen en un valor real (*fenotipo*) asociado a la secuencia binaria. Es importante destacar que un mapeo correcto asigna un único *fenotipo* a cada *genotipo*.

Una etapa importante en el diseño de AE es la elección de una representación adecuada al problema, ya que la eficiencia y complejidad del algoritmo de búsqueda dependerán fuertemente de la representación empleada. La mayoría de los AE emplean vectores de algún tipo específico de dato para construir los cromosomas. Una excepción es la programación evolutiva, que

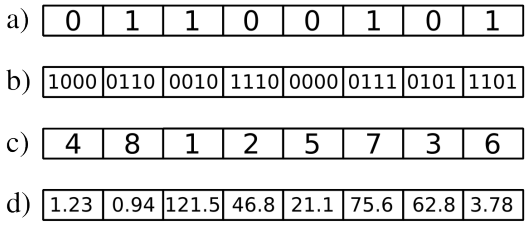


Figura 2.4: Ejemplos de representaciones empleadas comúnmente por los algoritmos evolutivos. a) y b) codificación binaria, c) codificación entera para un problema de permutaciones, d) codificación real.

codifica las soluciones potenciales empleando estructuras de datos tipo árbol.

La Figura 2.4 presenta algunos ejemplos de representaciones usadas habitualmente. Los dos cromosomas de la parte superior corresponden a representaciones empleadas habitualmente en los algoritmos genéticos. El cromosoma ubicado en la parte superior de la figura podría codificar una solución para un problema donde se busca un subconjunto de variables que satisfacen las restricciones de un problema. Se dispone de ocho elementos, cada uno asociado a una posición del cromosoma, y la asignación de 1 o 0 al mismo indica que el elemento debe o no ser considerado para la solución. En el segundo cromosoma (cromosoma b), cada gen corresponde a una secuencia binaria que podría representar a una variable real dentro un rango dado. El tercer cromosoma (cromosoma c) emplea una codificación basada en números enteros para un problema de permutaciones. El orden con que se disponen los genes determina las características de la solución. El cuarto cromosoma corresponde a una codificación empleando valores reales (cada gen corresponde a un valor), como es habitual en las estrategias evolutivas.

2.2.2. Población inicial

El primer paso que debe realizarse cuando se emplean estos algoritmos consiste en generar la población inicial. La forma más común de realizar esta tarea consiste en asignar en forma aleatoria uno de los posibles valores que puede tomar cada gen en cada cromosoma. Esta etapa es sumamente importante ya que aquellas regiones del espacio de búsqueda que no sean contempladas durante la inicialización probablemente serán descartadas du-

rante el proceso de búsqueda [Rutkowski, 2008]. Sólo mediante la acción de algún operador que introduzca fuertes modificaciones en los cromosomas, como por ejemplo el operador de mutación, será posible reconsiderar dichas regiones.

El tamaño de esta población inicial repercute fuertemente en la complejidad computacional y en la capacidad de exploración de estos algoritmos. El uso de una población con una elevada cantidad de individuos favorece la diversidad y mejora la capacidad de explorar el espacio de búsqueda, aunque la complejidad computacional para cada generación será mayor. Si bien la carga computacional por generación será mayor, es probable que se requiera un reducido número de generaciones para encontrar una solución. Una población con pocos individuos explorará una menor porción del espacio de búsqueda en cada generación y, aunque será computacionalmente menos costoso este algoritmo, probablemente requerirá un mayor número de generaciones para converger a una solución.

2.2.3. Función de aptitud

De acuerdo al modelo darwiniano de la evolución, los individuos que poseen las mejores características tienen mayores chances de sobrevivir y reproducirse. Con el objeto de evaluar esta habilidad, los AE utilizan una función Ψ que mapea el *fenotipo* de un cromosoma a un valor escalar, el cual refleja la calidad de la solución representada por ese cromosoma. Esta función objetivo Ψ es una componente de otra función más general denominada *función de aptitud* $\mathcal{A}$, y que es empleada para guiar el proceso de búsqueda [Bäck et al., 2000]. Una descripción más detallada de la función de aptitud es

$$\mathcal{A} : \mathbf{g} \xrightarrow{\Phi} \mathbf{f} \xrightarrow{\Psi} \mathbb{R} \xrightarrow{\Upsilon} \mathbb{R}^{+}, \tag{2.1}$$

donde $\mathbf{g}$ representa el *genotipo* de un individuo y $\mathbf{f}$ su *fenotipo* asociado. La función Φ es responsable de la decodificación del cromosoma, generando el *fenotipo* a partir del *genotipo* del individuo; Ψ es la función objetivo del problema y asigna a cada fenotipo un valor real; Υ es una función que escala los valores de aptitud para que tomen valores positivos [Engelbrecht, 2007].

En muchas situaciones es posible trabajar directamente con el *fenotipo*, por lo que se puede prescindir de Φ. Además, a menos que se requieran

valores estrictamente positivos para el valor de aptitud (por ejemplo, para aplicar selección proporcional), también es posible prescindir de Υ. Cuando estas situaciones se combinan, la función de aptitud pasa a ser simplemente la función objetivo del problema.

Una característica importante de la función objetivo es que admite formulaciones con pocos requisitos formales. Esto permite emplear funciones discontinuas, multimodales o no derivables como función objetivo en estos algoritmos. Finalmente, se debe enfatizar el papel principal que juega la función de aptitud en los AE. Si bien su principal tarea consiste en evaluar la calidad de las soluciones, estos resultados son luego empleados por diversos operadores para realizar su trabajo.

2.2.4. Selección

La selección es uno de los principales operadores empleados por los AE, y está fuertemente ligada al concepto de *supervivencia del más apto*. El objetivo principal detrás de los operadores que componen esta familia consiste en favorecer las mejores soluciones (mayor aptitud). Existen muchos operadores de selección, algunos de los cuales son diseñados para problemas específicos. Estos operadores son empleados principalmente para seleccionar los padres de la nueva población. En esta etapa el operador debe favorecer la supervivencia de los mejores individuos para la siguiente generación. En algunos casos particulares estos operadores también son empleados como parte de la estrategia de reemplazo, eligiendo los individuos que conformarán la generación siguiente [Engelbrecht, 2007].

Los operadores de selección pueden ser caracterizados a través de su *presión selectiva*, que es la cantidad de generaciones que se requieren para producir una población uniforme si sólo se aplicara el operador de selección. Si los operadores de selección poseen una elevada presión selectiva, se produce una disminución en la diversidad de la población más acelerada. Por lo tanto, una elevada presión selectiva puede conducir a soluciones sub-óptimas como consecuencia de una limitada capacidad para explorar el espacio de soluciones [Bäck et al., 2000].

Conceptualmente, el operador de selección aleatoria es el más simple ya que asigna a cada individuo la misma probabilidad para ser elegido. Suponiendo una población con N individuos, la probabilidad de seleccionar uno

cualquiera será $\frac{1}{N}$. Este operador no emplea información de la aptitud de los individuos, lo que permite que tanto los mejores como los peores individuos de la población tengan la misma probabilidad de ser elegidos para formar parte de la generación siguiente. Dado que la elección de los individuos no privilegia a ninguno, este operador posee la menor *presión selectiva*.

Los operadores de selección proporcional crean una distribución de probabilidad proporcional a la aptitud de los individuos de la población, y luego llevan a cabo un muestreo de ésta para seleccionar los individuos. Estos operadores están sesgados hacia la elección de los mejores individuos y poseen una elevada *presión selectiva*. La distribución de probabilidad utilizada es creada de acuerdo a

$$p(\mathbf{c}_i) = \frac{\mathcal{A}_\Upsilon(\mathbf{c}_i)}{\sum_{j=1}^{N} \mathcal{A}_\Upsilon(\mathbf{c}_j)} \tag{2.2}$$

donde $p(\mathbf{c}_i)$ es la probabilidad de seleccionar el individuo $\mathbf{c}_i$, y $\mathcal{A}_\Upsilon(\cdot)$ es la función objetivo escalada para tomar valores positivos, como en (2.1). La elección de la función de escalado depende del problema específico y de la información disponible para su aplicación. El operador de selección por ruleta es uno de los métodos más populares de selección por muestreo. La distribución de probabilidad es utilizada como una ruleta, donde a cada individuo $\mathbf{c}_i$ se le asigna un sector de la ruleta proporcional a su probabilidad $p(\mathbf{c}_i)$ de ser seleccionado. Un individuo es elegido haciendo girar la ruleta y observando la posición donde se detiene. Naturalmente, los individuos más aptos abarcarán una mayor área de la ruleta y será más probable que sean elegidos, pero esto no está garantizado. Debido a esto, la selección por ruleta posee la mayor presión de selección, ya que los individuos con mayor área en la ruleta (individuos más aptos) son los más favorecidos durante el muestreo. La implementación de este operador puede apreciarse en el Algoritmo 2.4. Claramente, el giro de la ruleta está representado por la probabilidad acumulada al recorrer los distintos cromosomas (línea 6) y el selector del individuo se representa mediante un valor aleatorio seleccionado a partir de una distribución uniforme (línea 3).

Otro operador ampliamente utilizado es el operador de selección por tor-

Algoritmo 2.4: Operador de selección por ruleta.

```
1 i ← 1 inicializar índice
2 α ← p(c_i) asignar la probabilidad del cromosoma c_i
3 u ← U(0,1) seleccionar un valor aleatorio con distribución uniforme
4 while α < u do
5 |   i ← i + 1 avanzar al siguiente cromosoma
6 |   α ← α + p(c_i)
7 return individuo c_i seleccionado
```

neo, que se basa en la competencia de un subgrupo de individuos de la población para elegir al más apto. Dada una población de N individuos, el operador selecciona $k < N$ de ellos y compara sus aptitudes para seleccionar el más apto. Aunque este operador emplea información acerca de la aptitud de los individuos, su presión selectiva es menor que la del operador de selección proporcional debido a que los individuos seleccionados para el torneo son elegidos en forma aleatoria. La elección del número de competidores juega un papel importante en la *presión selectiva* del operador. Si consideramos el caso extremo donde $k = N$, la presión selectiva será muy elevada debido a que la competencia se llevará a cabo sobre toda la población, seleccionando siempre el mejor individuo de la generación. El otro caso extremo, que corresponde a $k = 1$ individuo, conduce a la aplicación del operador de selección aleatoria (con baja presión selectiva) ya que este se elige en forma aleatoria y no compite con otro miembro de la población. Situaciones con valores para k entre estos extremos permiten obtener presiones selectivas crecientes que favorecen a los individuos más aptos sin descartar la posibilidad de elegir otros con menor aptitud.

Estos operadores favorecen la selección de los individuos más aptos para que formen parte de las nuevas generaciones. Sin embargo, dado que los mecanismos poseen componentes aleatorias, no hay garantías de que el mejor individuo de cada generación se preserve en la siguiente. El uso de elitismo soluciona este inconveniente mediante la copia sin alteración de los mejores k individuos de cada generación en la siguiente. En general, este operador se emplea seleccionando y preservando sin alteraciones al mejor individuo en cada generación ($k = 1$). Existen variantes donde se preserva un número mayor de individuos entre generaciones a costa de disminuir la diversidad de la población. En la práctica, este operador se emplea en conjunto con los operadores presentados previamente ya que su funcionamiento no altera el

desempeño de los demás.

2.2.5. Reproducción y reemplazo

La explotación y exploración del espacio de búsqueda son factores claves que deben considerarse para el buen desempeño de cualquier técnica de búsqueda y optimización. Mientras la explotación guía la búsqueda en base a la información de las mejores soluciones encontradas hasta el momento, la exploración promueve la inspección de nuevas regiones para evitar una convergencia prematura. El proceso de reproducción debe proveer ambos mecanismos en forma balanceada para garantizar un correcto funcionamiento [Gen and Chen, 2000]. En el caso de los AE, la explotación del espacio de búsqueda está influenciada por la *presión selectiva* de los operadores de selección empleados. Además, el operador de cruza explota las soluciones encontradas realizando una búsqueda local mediante el intercambio de información entre los individuos de la población. La exploración es mediada por el operador de mutación, introduciendo pequeñas variaciones en la composición de los cromosomas para favorecer la diversidad genética. El funcionamiento de ambas familias de operadores está estrechamente relacionado con la representación elegida para el problema.

La cruza consiste en la generación de uno o más individuos mediante la combinación del material genético de los padres. Debido a que los individuos que actuarán como padres durante la cruza son elegidos usando operadores de selección, la diversidad que presente este conjunto afectará fuertemente a la diversidad de la descendencia. Si los padres son elegidos con un operador de alta presión selectiva, la descendencia generada por la cruza presentará poca diversidad y una tendencia hacia una convergencia prematura. A modo de ejemplo, los operadores de cruza en uno y dos puntos son operadores típicos de los algoritmos genéticos. Estos operadores seleccionan una o dos posiciones en los cromosomas padres, respectivamente, y combinan la información genética contenida entre esas posiciones para generar la descendencia. La Figura 2.5 presenta un esquema del funcionamiento del operador de cruza en un punto (izquierda) y dos puntos (derecha). Las posiciones donde se producirá la cruza deben ser las mismas en ambos padres, en caso de que se requiera preservar el tamaño del cromosoma. Como se aprecia en la imagen de la izquierda (Figura 2.5a), la porción de $\mathbf{c}_1$ comprendida entre el

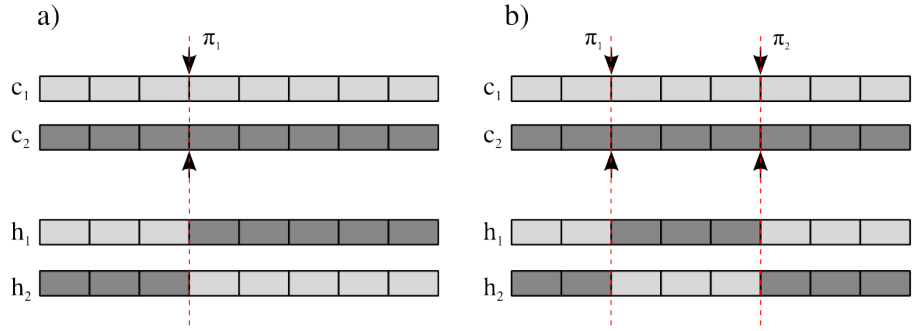


Figura 2.5: Ejemplos de operadores de cruza. a) Operador de cruza en un punto. b) Operador de cruza en dos puntos.

inicio del cromosoma y el punto de cruza π_1 es combinada con la porción de $\mathbf{c}_2$ comprendida entre π_1 y el final del cromosoma para generar el hijo $\mathbf{h}_1$. Repitiendo la operación pero intercambiando los roles de $\mathbf{c}_1$ y $\mathbf{c}_2$ se obtiene el segundo hijo $\mathbf{h}_2$. El operador de cruza en dos puntos trabaja de manera similar, pero intercambiando el material genético comprendido entre los puntos de cruza π_1 y π_2. La imagen de la derecha (Figura 2.5b) indica el resultado de este tipo de cruza.

El proceso de mutación implica la modificación de uno o más genes del cromosoma, reemplazando el material genético por *alelos* seleccionados en forma aleatoria. En general, este operador es aplicado con una baja probabilidad a cada gen en todos los individuos en la población, procurando no introducir disrupciones severas sobre las mejores soluciones. Sin embargo, algunos AE basan su funcionamiento en este operador y la probabilidades de mutación suelen ser más elevadas. Una forma de reducir la disrupción de las buenas soluciones es aplicar el operador con una probabilidad inversamente proporcional a la aptitud del individuo, mutando con mayor probabilidad aquellos individuos con baja aptitud. Otra aproximación que suele emplearse consiste en aplicar altas tasas de mutación al inicio del algoritmo para promover la exploración, y reducir la probabilidad gradualmente favoreciendo la explotación de las soluciones encontradas. Nuevamente y a modo de ejemplo se presentan dos operadores empleados por los algoritmos genéticos. Cuando se emplea una representación binaria de las soluciones, el operador de mutación actúa negando el bit sobre el cual es aplicado. Por otro lado, cuando la representación está basada en la permutación de números naturales, el intercambio de genes entre dos posiciones seleccionadas al azar suele

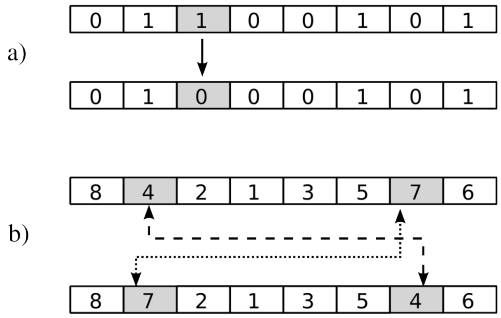


Figura 2.6: Ejemplos de operadores de mutación. a) Negación de bit. b) Intercambio de posiciones.

emplearse para introducir mutaciones en el cromosoma. Las Figuras 2.6a y 2.6b esquematizan ambos ejemplos.

Una vez generada la descendencia, la etapa siguiente es la construcción de la nueva generación. Este proceso puede llevarse a cabo empleando distintas estrategias que preservan en mayor o menor grado los individuos de la población actual. Dependiendo de la proporción de individuos conservados, o del grado de *solapamiento generacional*, las estrategias pueden dividirse en dos grandes grupos [Eiben and Smith, 2003]:

- *Reemplazo generacional*: todos los individuos de la población son reemplazados por su descendencia. Este método suele usarse en combinación con una estrategia elitista para preservar al individuo más apto, debido a que podría no reproducirse y perderse a lo largo de las generaciones.
- *Reproducción de estado estacionario*: sólo un número reducido de individuos (uno o dos por generación) son reemplazados por la descendencia. Existen diversas estrategias para elegir los individuos a reemplazar, pero generalmente se reemplazan los individuos menos aptos en cada generación.

También suelen ser habituales algunos casos donde se produce el reemplazo de una cantidad de individuos que es intermedia a estas dos situaciones límite. El término *brecha generacional* es empleado para describir estas situaciones, por ejemplo, indicando que una brecha generacional reducida está asociada a la conservación de una proporción elevada de individuos entre generaciones.

2.2.6. Criterios de finalización

La evolución de la población se realiza aplicando en forma iterativa los diferentes operadores que los componen hasta que se alcanza una condición de finalización. La condición más simple consiste en establecer un número máximo de generaciones. Este valor debe ser seleccionado con cuidado ya que valores demasiado bajos pueden limitar el espacio explorado. Adicionalmente suele emplearse algún criterio de convergencia para determinar si la población ha alcanzado alguna condición especificada. Algunos criterios utilizados comúnmente son:

- No se observan mejoras luego de una dada cantidad de generaciones. La aptitud del mejor individuo no sufre modificaciones luego de un número especificado de generaciones.
- No hay cambios en la población. La variación de la aptitud media de la población es menor que algún valor especificado durante un número especificado de generaciones.
- Cuando se encuentra una solución aceptable. Suponiendo un valor $\mathcal{A}^\epsilon$ de aptitud tolerada, la evolución se detiene cuando se encuentra un individuo $\mathbf{c}_i$ para el que $\mathcal{A}(\mathbf{c}_i) \geq \mathcal{A}^\epsilon$ (maximización) ó $\mathcal{A}(\mathbf{c}_i) \leq \mathcal{A}^\epsilon$ (minimización).
- Criterios específicos del problema. El *fenotipo* del mejor individuo cumple con alguna condición que satisface la búsqueda/optimización.

2.3. Comentarios finales

En el presente capítulo se introdujeron los elementos que describen a un problema de búsqueda y se presentaron dos familias de algoritmos que se utilizan para su resolución. En primer lugar se describieron los métodos clásicos de búsqueda, que exploran el espacio de soluciones construyendo un árbol de búsqueda empleando diferentes estrategias que realizan la tarea con y sin el uso de información adicional del problema. Las estrategias más representativas fueron explicadas de forma detallada y se analizaron en términos de complejidad espacial y temporal. En segundo lugar se presentaron los métodos pertenecientes a la computación evolutiva, discutiendo sus principales

ventajas y describiendo los elementos básicos que constituyen a los miembros de esta familia.

Como se discutió en las primeras secciones, los métodos clásicos de búsqueda se vuelven poco prácticos cuando el espacio de búsqueda es muy extenso. Si bien las estrategias como A* mejoran sustancialmente el proceso de búsqueda, la información adicional requerida sobre el problema no siempre se encuentra disponible. Los algoritmos de computación evolutiva presentan características que los hacen adecuados para explorar espacios de búsqueda extensos, sin requerir demasiada información para guiar la búsqueda.

En los capítulos siguientes se utilizarán los conceptos que aquí fueron presentados para abordar el problema de búsqueda de vías metabólicas. En el Capítulo 3 se presentará el desarrollo de un algoritmo evolutivo para buscar vías metabólicas entre dos compuestos, y su desempeño se comparará con el de BFS y DFS para la misma tarea. Tomando como referencia este nuevo algoritmo propuesto, en el Capítulo 4 se extenderá la propuesta para desarrollar un algoritmo basado en CE que busque vías metabólicas ramificadas entre un mayor número de compuestos.

Capítulo 3

Búsqueda de vías metabólicas lineales

El uso de métodos de búsqueda es habitual para resolver problemas en diversas áreas del conocimiento. En muchos casos, el uso de estrategias clásicas de búsqueda para explorar en forma secuencial el espacio de estados permite encontrar soluciones rápidamente. Sin embargo, existen problemas en los que debe evaluarse un elevado número de soluciones y para los que las técnicas de exploración secuencial se vuelven prácticamente inaplicables. Por ejemplo, la base de datos KEGG contiene aproximadamente 17000 compuestos con 14000 conexiones entre ellos, además de una elevada ramificación. Combinando esta información con la de otras bases de datos, Küffner et al. [2000] realizaron la búsqueda de vías metabólicas con hasta 9 reacciones entre los compuestos glucosa y piruvato. Sus resultados revelaron la existencia de aproximadamente $5 \cdot 10^5$ vías metabólicas, aunque muchas de ellas carecían de validez en términos biológicos.

En los últimos años se han propuesto diferentes estrategias para la búsqueda de vías metabólicas entre dos compuestos, pero los enfoques requieren información estructural de los compuestos [McShan et al., 2003] o cantidades significativas de memoria para almacenar los árboles de búsqueda [Vempaty et al., 1991]. En este contexto, los algoritmos evolutivos son una alternativa prometedora para sortear estas dificultades. Como se discutió en el Capítulo

2, la principal diferencia que presentan estos algoritmos frente a los métodos clásicos de búsqueda es que evitan la exploración exhaustiva del espacio de soluciones empleando diferentes heurísticas para seleccionar las regiones más prometedoras que serán exploradas. Este aspecto ha hecho de los algoritmos evolutivos una opción atractiva para abordar diferentes problemas en biología [Morbiducci et al., 2005; Zou et al., 2010; Yu and Lee, 2012].

La principal contribución realizada en este capítulo consiste en la propuesta de un algoritmo evolutivo para buscar vías metabólicas lineales, donde cada reacción sólo depende de la reacción previa, para relacionar dos compuestos. Dados un compuesto inicial y un compuesto que se desea producir, el algoritmo busca una secuencia de reacciones concatenadas que transforman el compuesto de partida en el producto deseado. La organización del capítulo es la siguiente: la Sección 3.1 describe el algoritmo propuesto para la búsqueda de caminos entre dos compuestos. La Sección 3.2 detalla el conjunto de datos empleados en los experimentos y las medidas de desempeño utilizadas en los distintos análisis. La Sección 3.3 presenta los resultados de la evaluación de algunos parámetros sobre el desempeño del algoritmo, junto con la comparación con dos métodos clásicos de búsqueda.

3.1. Desarrollo del algoritmo evolutivo

En esta sección se describe el algoritmo propuesto denominado *algoritmo evolutivo para la búsqueda de vías metabólicas lineales* (AEBVML). En primer lugar se define el espacio de estados y los operadores genéticos empleados. Luego se presenta la estructura de cromosoma y la forma en que se codifica la información, junto con los operadores genéticos empleados en el algoritmo. Finalmente se describe la función de aptitud utilizada, analizando los términos que la componen y la contribución de cada uno en el proceso de búsqueda.

Existen diversas formas de explorar el espacio que contiene todas las posibles vías metabólicas que vinculan dos compuestos. Nuestro enfoque emplea un conjunto de relaciones entre pares "sustrato–producto" para representar reacciones químicas [Kotera et al., 2004b]. Cada uno de estos pares se encuentra etiquetado de acuerdo a la función que desempeña dentro de la reacción. Por ejemplo, los pares reactantes principales identifican un sustrato y un producto que comparten una fracción importante de sus estructuras. Em-

pleando la información de estos pares principales se pretende modelar vías metabólicas que relacionen los compuestos que se especifiquen. El espacio de estados para este problema queda definido por el conjunto C de todos los pares reactantes principales o transformaciones r, y los compuestos que se relacionan a través de estas transformaciones. Para unificar la notación, el compuesto sobre el cual se aplicará la transformación se denominará *sustrato* s, mientras que el resultado o nuevo estado será el *producto* p. Las transformaciones se indicarán como pares ordenados $r_i = (s_i, p_i)$, donde $r_i \in C$ y $s_i \neq p_i$. El sustrato y el producto de la transformación r_i se identificarán como s_i y p_i, respectivamente. Además, el compuesto inicial de la búsqueda se indicará como $\widehat{s}$, mientras que el compuesto final que se desea producir será $\widehat{p}$. De esta manera, una vía metabólica está formada por una secuencia ordenada de transformaciones que produce $\widehat{p}$ a partir de $\widehat{s}$.

3.1.1. Estructura del cromosoma

La representación seleccionada para el cromosoma codifica la secuencia de transformaciones r_i que produce $\widehat{p}$ a partir de $\widehat{s}$ como

$$\mathbf{c} = [r_1, r_2, \ldots, r_i, \ldots, r_N],$$

donde N indica el número de genes en el cromosoma. Este valor puede variar en el intervalo $[1, N_M]$, siendo N_M el número máximo de transformaciones que puede contener la vía metabólica. Cuando la cantidad de transformaciones excede este nivel, el cromosoma se trunca para contener sólo los primeros N_M genes. La secuencia de transformaciones que describen la vía metabólica debe ser leída de izquierda a derecha para que la secuencia produzca $\widehat{p}$ a partir de $\widehat{s}$.

3.1.2. Función de aptitud

La evaluación de las características deseadas de la solución se realiza mediante una función de aptitud diseñada a partir de distintos aspectos de las reacciones bioquímicas. Dado el cromosoma $\mathbf{c}$, esta función se define como

$$\mathcal{A}(\mathbf{c}) = \alpha \left[\mathcal{V}(\mathbf{c}) + \beta \mathcal{E}(\mathbf{c}) + \mathcal{Q}(\mathbf{c}) + \mathcal{I}(\mathbf{c}) \right], \tag{3.1}$$

donde $\mathcal{V}$ es la validez de la secuencia, $\mathcal{E}$ es la medida de extremos válidos, $\mathcal{Q}$ es la tasa de transformaciones únicas, $\mathcal{I}$ es la tasa de compuestos únicos,

$\alpha = 1/(3+\beta)$ es una constante de normalización que garantiza que los valores de aptitud estén en el intervalo [0,1], y β determina el peso relativo de la contribución del término E. La función $\mathcal{A}(\mathbf{c})$ alcanza su máximo valor cuando se encuentra una vía metabólica válida y sin bucles que produce $\widehat{p}$ a partir de $\widehat{s}$. A continuación se describen las cuatro medidas que componen la función de aptitud.

3.1.2.1. Validez

El término $\mathcal{V}(\cdot)$ en (3.1) cuantifica el número de conexiones válidas en el cromosoma. En este contexto, se define *conexión válida* como la concatenación de dos transformaciones en donde el producto p_i de la transformación r_i actúa como el sustrato s_{i+1} de la transformación r_{i+1}. En base a esto, la validez se calcula como

$$\mathcal{V}(\mathbf{c}) = \frac{\delta(\widehat{s}, s_1) + \sum_{i=1}^{N-1} \delta(s_{i+1}, p_i) + \delta(p_N, \widehat{p})}{N+1}, \tag{3.2}$$

donde $\delta(\cdot,\cdot)$ es la función delta de Kronecker, que toma valor 1 cuando sus argumentos son iguales. Esta medida varía en el intervalo $[0,1]$, siendo máxima cuando todas las transformaciones están concatenadas mediante conexiones válidas y los compuestos s_1 y p_N son los especificados. Este último requisito se impone para reforzar la importancia de conservar los compuestos inicial y final en el cromosoma.

3.1.2.2. Extremos válidos

El término $\mathcal{E}(\cdot)$ de (3.1) evalúa específicamente las reacciones r_1 y r_N del cromosoma, y determina si los compuestos $\widehat{s}$ y $\widehat{p}$ son empleados, respectivamente, por ellas. Su cálculo se realiza de acuerdo a

$$\mathcal{E}(\mathbf{c}) = \frac{1}{2}\left[\delta(\widehat{s}, s_1) + \delta(p_N, \widehat{p})\right], \tag{3.3}$$

Este término puede tomar los valores $\{0, 0.5, 1\}$, dependiendo de si s_1 y p_N corresponden a los compuestos especificados. La contribución de esta medida se aprecia, principalmente, cuando el cromosoma debe ser truncado.

3.1.2.3. Tasa de transformaciones únicas

El término $\mathcal{Q}(\cdot)$ penaliza la repetición de transformaciones en el cromosoma. Para calcular esta medida, se define la función auxiliar $\varphi(\cdot)$ que determina el número de elementos únicos presentes en una secuencia. En base a esto, la tasa de transformaciones únicas se calcula como

$$\mathcal{Q}(\mathbf{c}) = \frac{\varphi(\mathbf{c}) - 1}{N - 1}. \tag{3.4}$$

Esta medida puede tomar valores en el intervalo [0,1], y alcanza el valor mínimo cuando el cromosoma contiene un único elemento que se repite N veces ($\varphi(\mathbf{c}) = 1$). También debe notarse que $Q(\mathbf{c}) = 0$ cuando el cromosoma contienen una única transformación.

3.1.2.4. Tasa de compuestos únicos

El término $\mathcal{I}(\cdot)$ penaliza la repetición de compuestos en la vía. Esta medida se calcula como

$$\mathcal{I}(\mathbf{c}) = \frac{\varphi(\mathbf{q}) - 2}{N - 1}, \tag{3.5}$$

donde $\mathbf{q} = [\widehat{s}, p_1, p_2, \ldots, \widehat{p}]$ es la secuencia de estados/compuestos de la vía metabólica codificada en $\mathbf{c}$. La tasa de compuestos únicos puede variar en el intervalo [0,1], alcanzando el valor mínimo para $N = 1$ o cuando el cromosoma contiene transformaciones que sólo producen los compuestos s_i o p_i (cromosoma que contiene repeticiones de las transformaciones (s_i, p_i) y/o (p_i, s_i). Por ejemplo, la vía metabólica codificada en el cromosoma $\mathbf{c} = [(a, b), (b, a), (a, b), (b, d)]$ contiene $N = 4$ reacciones, y la secuencia de estados asociada es $\mathbf{q} = [\underline{a}, \underline{b}, a, b, \underline{d}]$ (ver definición en página 41). Claramente sólo los compuestos a, b, d son únicos, por lo que $\varphi(\mathbf{q}) = 3$ e $I(\mathbf{c}) = \frac{1}{3}$.

Para ilustrar cómo se realiza el cálculo de la función de aptitud, se empleará como ejemplo el cromosoma $\mathbf{c} = [(a, b), (b, q), (q, y), (y, r), (r, e)]$. Suponiendo que los compuestos inicial y final son $\widehat{s} = a$ y $\widehat{p} = e$, se obtiene $\mathcal{E}(\mathbf{c}) = 1$ debido a que los compuestos de inicio y fin son los especificados. El término $\mathcal{V}(\mathbf{c}) = 1$ debido a que todas las conexiones son válidas (el sustrato de cada transformación es generado por la transformación previa). Además, dado que cada transformación es empleada sólo una vez en el cromosoma, $\mathcal{Q}(\mathbf{c}) = 1$. Finalmente, cuando se analiza la secuencia de com-

puestos $\mathbf{q}$ asociada al cromosoma se obtiene $\mathbf{q} = [a, b, q, y, r, e]$. Como puede verse, $\varphi(\mathbf{q}) = 6$ dado que cada compuesto aparece una sola vez en $\mathbf{q}$ y lleva a que $\mathcal{I}(\mathbf{c}) = 1$. En base a estos resultados parciales, la función de aptitud vale $\mathcal{A}(\mathbf{c}) = 1$.

Si suponemos que el tercer gen del cromosoma es reemplazado por la transformación (b, q) para obtener el cromosoma $\mathbf{c}'$, entonces ahora $V(\mathbf{c}') = \frac{2}{3}$ debido a que las conexiones entre los genes 2–3 y 3–4 son inválidas. Dado que el compuesto inicial y el final no se modifican, la medida de extremos válidos mantiene su valor. La tasa de transformaciones únicas toma el valor $\mathcal{Q}(\mathbf{c}') = \frac{3}{4}$ debido a la presencia de un gen repetido. La tasa de compuestos únicos también disminuye ya que para la nueva secuencia $\mathbf{q} = [\underline{a}, \underline{b}, \underline{q}, q, \underline{r}, \underline{e}]$, el número de compuestos únicos $\varphi(\mathbf{q}) = 5$ y la medida toma valor $\mathcal{I}(\mathbf{c}') = \frac{3}{4}$. La combinación de los términos calculados conduce a una aptitud $\mathcal{A}(\mathbf{c}') \sim 0.792$.

3.1.3. Inicialización, reemplazo y operadores genéticos

3.1.3.1. Estrategias de inicialización y reemplazo

La inicialización del algoritmo requiere la definición del número M de individuos que compondrán la población, y un número inicial de genes $N_I \leq N_M$ que contendrá cada individuo. Una vez establecidos estos parámetros, los cromosomas pueden ser inicializados de dos maneras diferentes. Cuando se emplea la estrategia de *inicialización aleatoria* (3.6), el cromosoma es construido de acuerdo a

$$\mathbf{c} = \begin{cases} r_i \in R^+ & \text{si } N = 1 \\ [r_i, r_j], r_i \in R^1, r_j \in R^N & \text{si } N = 2 \\ [r_i, \ldots, r_k, \ldots, r_j], r_i \in R^1, r_k \in R^*, r_j \in R^N & \text{si } N > 2 \end{cases} \tag{3.6}$$

donde cada gen r corresponde a un transformación seleccionada en forma aleatoria a partir del conjunto correspondiente. El conjunto R^* está conformado por todas las transformaciones permitidas. $R^1 = \{r_i / r_i = (\widehat{s}, p_i)\} \ \wedge R^1 \subset R^*$ contiene sólo aquellas transformaciones que tienen a $\widehat{s}$ como sustrato, mientras que $R^N = \{r_i / r_i = (s_i, \widehat{p})\} \ \wedge R^N \subset R^*$ contiene todas las transformaciones que producen $\widehat{p}$. Finalmente, $R^+ = R^1 \cup R^N$ está

formado por la unión de los dos conjuntos anteriores. Esta metodología asegura que todos los genes tengan la misma probabilidad de ser elegidos. Sin embargo, no se contempla el carácter secuencial que posee una vía metabólica al incorporar las transformaciones en el cromosoma.

La estrategia de *inicialización parcialmente válida* (3.7) incorpora los genes en el cromosoma manteniendo la validez de la conexiones. Cada nueva transformación es elegida de forma aleatoria de entre todas las posibles transformaciones que emplean como sustrato el producto de la transformación previa, de acuerdo con

$$\mathbf{c} = \begin{cases} r_i \in R^+ & \text{si } N = 1 \\ [r_i, r_j], r_i \in R^1, r_j \in R^N & \text{si } N = 2 \\ [r_i, \ldots, r_k, \ldots, r_j], r_i \in R^1, r_k \in R^* / s_k = p_{k-1}, r_j \in R^N & \text{si } N > 2 \end{cases} \quad (3.7)$$

Las transformaciones que forman parte del cromosoma no son consideradas nuevamente para ser insertadas en la secuencia. Un caso particular de esta estrategia, denominada estrategia de *inicialización parcialmente válida con longitud fija*, consiste en inicializar los cromosomas con el mismo número inicial de genes N_x.

La estrategia de reemplazo empleada preserva la mejor solución en cada generación mediante el uso de elitismo ($\varepsilon = 1$ individuo), y genera los $M - \varepsilon$ individuos restantes a partir de los padres seleccionados para la cruza y su descendencia. Esta estrategia emplea brecha generacional, ya que los padres seleccionados para la cruza son incluidos sin modificaciones en la nueva población. El número de individuos seleccionados para la cruza corresponde a una fracción de la población y se determina como $(1 - p_x)M$, siendo p_x la probabilidad de cruza empleada en el algoritmo. Así, probabilidades de cruza elevada conducen a la generación de un mayor número de hijos a partir de unos pocos individuos seleccionados y a una elevada explotación de las soluciones codificadas por esos padres.

3.1.3.2. Operador de cruza

Este operador presenta una modificación respecto del operador clásico de cruza en un punto ya que favorece la formación de conexiones válidas en la posición donde se empalma el material genético. El primer paso consiste en

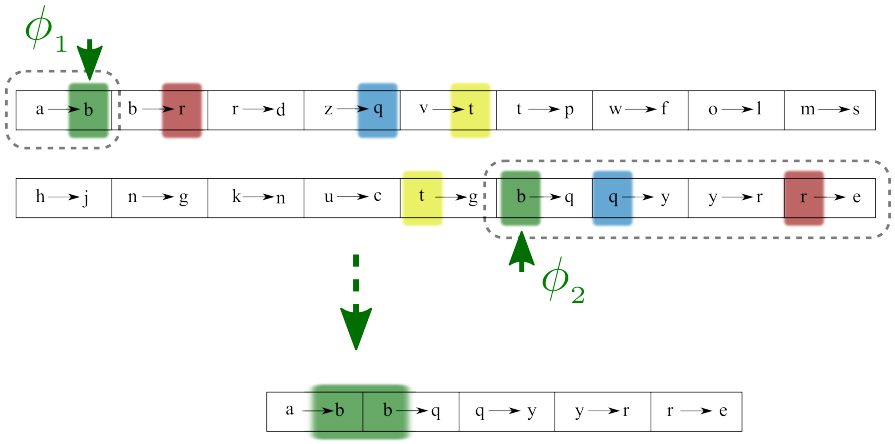


Figura 3.1: Esquema de funcionamiento del operador de cruza. Cada bloque indica una transformación codificada por un gen. Para cada transformación, las letras representan el sustrato y el producto de la transformación, a izquierda y derecha respectivamente. Las regiones sombreadas indican pares de posiciones (ϕ_1, ϕ_2) donde puede efectuarse una cruza válida.

seleccionar el punto de cruza en cada padre. Suponiendo que los individuos $\mathbf{c}_1$ y $\mathbf{c}_2$ son seleccionados para la cruza, el operador busca pares de puntos de cruza (ϕ_1, ϕ_2) en cada padre que satisfagan la condición de validez en el punto de empalme. Para ello analiza ambos padres buscando pares de transformaciones en las que se cumple $\delta(s_i, p_j) = 1$, donde $s_i \in \mathbf{c}_1$ y $p_j \in \mathbf{c}_2$. La Figura 3.1 presenta un ejemplo del funcionamiento del operador para el caso de dos padres que no son completamente válidos. Cada bloque en el cromosoma corresponde a un gen y codifica una transformación química; las letras indican el sustrato y producto de cada transformación. Claramente, si se aplica un operador de cruza que no contempla la estructura de la secuencia de transformaciones, es altamente probable que se produzca una descendencia cuya validez sea menor que la de sus padres. En cambio, si la cruza se realiza en alguno de los pares de puntos (ϕ_1, ϕ_2) seleccionados con el operador de cruza modificado (resaltados en la Figura 3.1), la validez de la descendencia se incrementará debido a que será mayor el número de conexiones válidas presentes en el hijo. Cuando no es posible identificar al menos un par de puntos (ϕ_1, ϕ_2) en los padres considerados, se seleccionan dos padres nuevos para realizar la cruza.

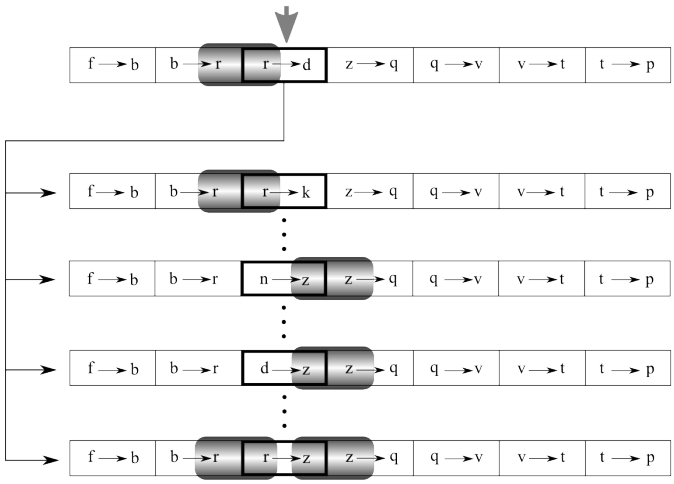


Figura 3.2: Diagrama del funcionamiento del operador de mutación. El gen a mutar se indica con un rectángulo negro. La aplicación del operador de mutación puede conservar o aumentar la validez del cromosoma.

3.1.3.3. Operador de mutación

Este operador reemplaza un gen r_i por otro seleccionado aleatoriamente del conjunto formado por todas las transformaciones que producen s_{i+1} o que emplean como sustrato p_{i-1}. Para cada cromosoma, la probabilidad de mutar un gen está determinada por el parámetro p_m, y el conjunto a partir del cual el gen es seleccionado está definido por

$$mut(r_i) = \begin{cases} r_i \in R^+ & \text{si } N = 1, \\ r_i \in R^1 & \text{si } N > 1 \ \wedge \ i = 1, \\ r_i \in R^N & \text{si } N > 1 \ \wedge \ i = N, \\ r_i \in R^* / p = s_{i+1} & \text{si } N > 1 \ \wedge \ 1 < i < N \ \wedge \ u \leq 0.5, \\ r_i \in R^* / s = p_{i-1} & \text{si } N > 1 \ \wedge \ 1 < i < N \ \wedge \ u > 0.5, \end{cases} \tag{3.8}$$

donde s_{i+1} es el sustrato del gen r_{i+1} y p_{i-1} el producto del gen r_{i-1}. El valor de u es seleccionado aleatoriamente a partir de una distribución uniforme en el intervalo [0,1]. En la Figura 3.2 se presenta un ejemplo del funcionamiento del operador, en donde el gen a mutar posee una conexión válida con el gen previo en el cromosoma. Si se aplica un operador de mutación clásico es altamente probable que la validez del cromosoma disminuya debido a que

el nuevo gen podría no establecer una conexión válida con ninguno de sus vecinos. Por el contrario, si se aplica la estrategia de mutación presentada en (3.8), es más probable que el cromosoma conserve o incluso incremente su validez, como ocurre en el cromosoma ubicado en la parte inferior de la Figura 3.2. Si bien en algunos casos es posible que la validez del nuevo cromosoma disminuya, este es un pequeño costo que se debe pagar para ser capaces de explorar regiones distantes del espacio de búsqueda y escapar de mínimos locales.

3.2. Conjunto de datos y medidas de evaluación

3.2.1. Datos de compuestos y reacciones

Los compuestos considerados durante las búsquedas y las relaciones entre ellos fueron extraídos de la base de datos KEGG[1]. Esta base de datos fue seleccionada debido a que es ampliamente citada en la literatura y contiene información acerca de una gran cantidad de organismos. Cada compuesto y reacción tiene asociado un código de identificación. Las reacciones están almacenadas como conjuntos de relaciones entre pares de compuestos. Cada par "sustrato–producto" (s, p) entre los que existe una relación dentro de la reacción se denomina *par reactante* y tiene asociada una etiqueta que especifica el tipo de relación entre esos compuestos [Faust et al., 2009].

El conjunto de datos empleado se construyó extrayendo los pares reactantes principales de todas las reacciones almacenadas en KEGG, y filtrando los resultados para conservar una única copia de cada uno. Con esos datos se construyó el conjunto de relaciones posibles entre compuestos. Aquellos compuestos que no participan de algún par reactante no pueden ser considerados durante las búsquedas. Cada par reactante es dividido en dos transformaciones con sentidos opuestos (s, p) y (p, s). Luego de procesar las reacciones almacenadas en la base de datos KEGG[2] se extrajeron 5936 compuestos relacionados mediante 14346 pares reactantes principales.

Una vez que se dispone de la información de las conexiones entre compuestos, existen diferentes formas de elegir el par de compuestos entre los que se realiza la búsqueda, ya que la forma en que éstos son elegidos no influye en

[1] `http://www.genome.jp/kegg/`

[2] La última versión libre de la base de datos fue descargada en mayo de 2011.

Tabla 3.1: Agrupamientos seleccionados para buscar vías metabólicas.

Agrupamiento A			Agrupamiento B		
Compuestos	Isómeros		Compuestos	Isómeros	
	I	II		I	II
Arginina	C00062	C00792	Asparagina	C00152	C01905
Glicerato	C00258		Glicina	C00037	
Lisina	C00047		Histidina	C00135	
Ornitina	C00077	C00515	Isoleucina	C00407	
			Serina	C00065	C00740
			Tirosina	C00082	
			Treonina	C00188	C00820
			Valina	C00183	C06417

el funcionamiento de AEBVML. Por ejemplo, los compuestos podrían seleccionarse de forma aleatoria o siguiendo algún criterio específico, como puede ser buscar una vía entre compuestos que presentan algún interés especial para el usuario. Otra alternativa podría consistir en generar agrupamientos a partir de datos metabólicos empleando alguna herramienta de minería de datos y luego buscar vías metabólicas que relacionen dos compuestos dentro de un mismo agrupamiento. Para esto sólo sería necesario emplear un algoritmo de búsqueda y un conjunto de relaciones permitidas.

Los experimentos descriptos en las secciones siguientes se realizaron seleccionando compuestos empleando esta última metodología. A partir de datos de perfiles metabólicos y transcripcionales de frutos de tomate cultivados bajo condiciones controladas y cosechados durante la etapa de maduración [Carrari et al., 2006], se generaron agrupamientos utilizando una herramienta de minería de datos basada en una red neuronal [Milone et al., 2010]. Esta herramienta utiliza mapas auto–organizativos y el principio de *culpable por asociación* [Saito et al., 2008], para generar los agrupamientos. Este principio establece que dos elementos que participan del mismo proceso varían de forma coordinada su comportamiento. Por lo tanto, es altamente probable que compuestos que se encuentran agrupados participen en algún mismo proceso.

La Tabla 3.1 presenta los compuestos que conforman dos de los agrupamientos generados. Los experimentos realizados en la sección siguiente se llevaron a cabo empleando los compuestos de esta tabla. Para ambos agrupamientos se emplearon los compuestos pertenecientes a la columna I de

los isómeros[3] debido a que son los que poseen mayor importancia biológica [Nelson and Cox, 2004]. En todos los casos las búsquedas se realizaron entre compuestos de un mismo agrupamiento. Debido a que el número de posibles combinaciones entre pares de compuestos de un agrupamiento entre los que se realizan las búsquedas es muy elevado, se seleccionaron en forma aleatoria tres pares para cada agrupamiento. Los pares seleccionados para el agrupamiento A fueron C00062–C00047, C00258–C00077 y C00047–C00258, mientras que para el agrupamiento B se eligieron C00135–C00065, C00135–C00082 y C00037–C00082.

3.2.2. Medidas para evaluación de características de las soluciones y desempeño del algoritmo

Los resultados obtenidos empleando diferentes algoritmos se compararon a través de la medición del tiempo requerido para encontrar una vía metabólica (t), el número de transformaciones (L) que componen la solución y el número de generaciones (G) empleado para encontrar una solución.

Los experimentos consistieron de 20 corridas para cada par de compuestos empleado. En cada corrida se determinaron los valores máximo (indicado con el subíndice M), mínimo (indicado con el subíndice m) y la mediana (indicado con el símbolo "$\widehat{\ }$") para distintas medidas. En la mayoría de los casos se emplea el valor de la mediana en lugar del valor medio debido a que el primero es un estimador más robusto frente a distribuciones asimétricas. Para determinar la normalidad de la distribución se empleó la prueba de Jarque-Bera[4] [Ruppert, 2011]. En todos los casos los resultados presentaron distribuciones no-normales. Las diferentes corridas se llevan a cabo para evaluar la diversidad en las vías metabólicas encontradas empleando distintos algoritmos, ya que las soluciones podrían estar sesgadas a vías con una composición particular para algún algoritmo. Para evaluar la proporción de compuestos de un agrupamiento que son incorporados en la vía metabólica, definimos la tasa de explicación del agrupamiento como

[3]Los isómeros son compuestos químicos que poseen igual fórmula molecular (iguales proporciones relativas de los átomos que conforman sus moléculas) pero estructuras moleculares distintas y diferentes propiedades.

[4]Esta prueba realiza un test de hipótesis donde se compara una métrica que considera la simetría y kurtosis de la distribución evaluada frente a la correspondiente a la distribución normal.

$$\Lambda = \frac{\max_k \{\psi_k\}}{|\Psi|}, \tag{3.9}$$

donde k indica el número de la corrida, ψ_k es el número de compuestos incluidos en el agrupamiento al que pertenecen los compuestos inicial y final, y que forman parte de la vía metabólica encontrada en la corrida k, y $|\Psi|$ es el número de compuestos que componen el agrupamiento. Esta medida varía en el intervalo [0,1] e indica la fracción de compuestos del agrupamiento que son incorporados en la solución. Valores de Λ cercanos a 1 indican que una gran proporción de los compuestos del agrupamiento son incorporados en la vía.

El análisis estadístico de los resultados se realiza empleando la prueba de Wilcoxon de rangos con signo y la prueba de comparación múltiple de Friedman, debido a que no es necesario asumir ninguna distribución para los datos [Derrac et al., 2011]. La primera es similar a la prueba paramétrica t de Student, y permite evaluar si dos distribuciones de datos son iguales. La segunda cumple una función similar al análisis de la varianza (ANOVA), ya que evalúa si en un conjunto de poblaciones hay al menos dos que presentan diferencias para un dado valor de significancia.

3.3. Resultados y discusión

En esta Sección se presentan los resultados obtenidos para las diferentes evaluaciones realizadas sobre AEBVML. Primero se analiza la sensibilidad del algoritmo frente a la variación de distintos parámetros y al uso de diferentes operadores. Luego se compara el desempeño del AEBVML con el de las estrategias de búsqueda en amplitud y en profundidad. Finalmente se presentan dos casos de estudio donde se analizan y discuten algunos aspectos biológicos de las soluciones encontradas.

3.3.1. Probabilidad de cruza

Como se presentó en la Sección 3.3.2, la estrategia de reemplazo implementada en AEBVML depende fuertemente de la probabilidad de cruza usada durante la evolución, ya que valores elevados para p_x están asociados a un recambio generacional importante.

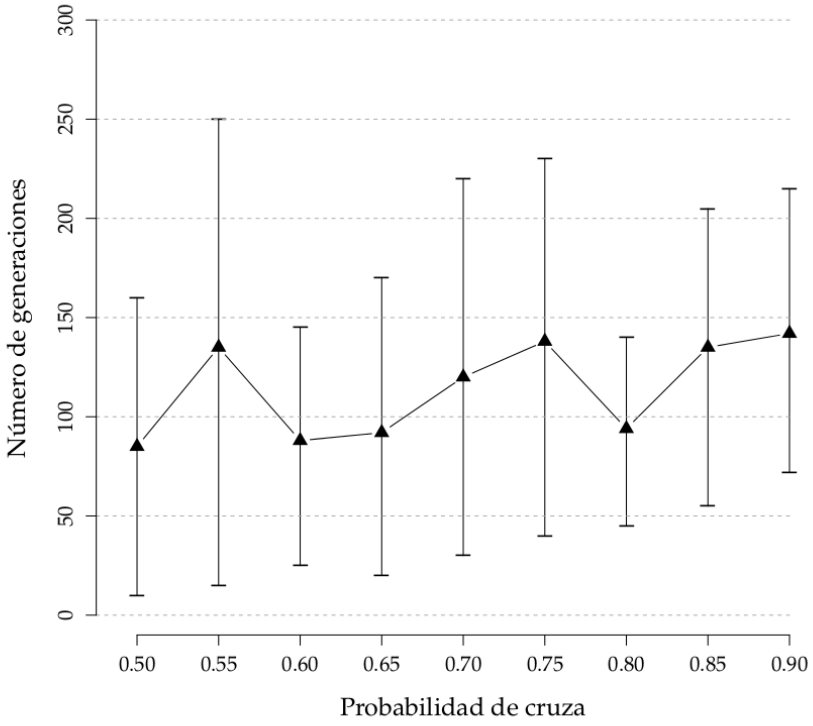


Figura 3.3: Efecto de la probabilidad de cruza sobre el número de generaciones empleadas por AEBVML. Los triángulos indican la mediana del número de generaciones para todas las corridas realizadas. Las líneas verticales indican el intervalo de confianza para cada mediana.

Para conocer el efecto que produce la probabilidad de cruza sobre el desempeño del algoritmo empleando inicialización aleatoria, se evaluaron valores de p_x en el intervalo [0.5,0.9] utilizando un paso de 0.05. Durante los experimentos se empleó una población de $M = 1000$ individuos para lograr un muestreo adecuado del espacio de soluciones, una probabilidad de mutación $p_m = 0.05$, y un número inicial de genes $N_I = 50$, fijando un tamaño máximo de cromosoma de $N_M = 100$ genes para permitir vías con mayor tamaño que el inicial. El criterio de finalización de la búsqueda consistió en fijar el número máximo de generaciones $G_M = 1000$, o encontrar un individuo cuya aptitud sea $\mathcal{A}(\mathbf{c}_i) = 1.0$.

La Figura 3.3 presenta la mediana del número de generaciones empleado por AEBVML para encontrar una solución con cada probabilidad de cruza evaluada. Para cada resultado se dibuja superpuesto el intervalo de confianza calculado como la desviación absoluta respecto a la mediana.

Los valores presentados en la Figura 3.3 muestran que la mediana del número de generaciones varía en el intervalo [80,140] sin una tendencia definida. Los intervalos de confianza son amplios y se superponen, indicando que no hay diferencias significativas entre las probabilidades estudiadas. Es interesante destacar que el intervalo de confianza de menor tamaño (50 generaciones) se obtiene con $p_x = 0.8$, que a su vez está asociado a una de las medianas mas bajas (92 generaciones) registradas. Es probable que el uso de esta probabilidad de cruza produzca el mejor balance entre exploración y explotación de las soluciones. En base a estos resultados se eligió el valor $p_x = 0.8$ en los experimentos posteriores.

3.3.2. Estrategias de inicialización y probabilidad de mutación

El siguiente aspecto analizado consistió en la evaluación del efecto combinado de la estrategia de inicialización y la probabilidad de mutación. Esta última se analizó particularmente debido a que afecta en forma directa el grado de exploración del algoritmo. Las tasas de mutación elevadas introducen un mayor número de disrupciones pero aumentan la probabilidad de explorar regiones distantes en el espacio de búsqueda.

La probabilidad de mutación se evaluó en el intervalo [0.01,0.1] empleando las tres estrategias de inicialización presentadas en la Sección 3.3.2. En todos los casos se utilizó elitismo ($\varepsilon = 1$) para conservar la mejor solución en cada generación. Todas las corridas se realizaron empleando $M = 1000$ individuos, un tamaño máximo de cromosoma con $N_M = 100$ genes, un número inicial máximo $N_I = 50$ de genes por cromosoma, y una probabilidad de cruza $p_x = 0.8$. Para el caso de la inicialización parcialmente válida con longitud fija se forzó a que los cromosomas tuvieran el máximo número de genes durante la inicialización ($N_I = N_M$). La Figura 3.4 presenta los resultados obtenidos para el número de generaciones (Figura 3.4a) y el tiempo de búsqueda[5] (Figura 3.4b). Los círculos, triángulos y cuadrados indican el valor de la mediana asociada a cada estrategia de inicialización, mientras que las líneas verticales representan el intervalo de confianza del 95 % calculado mediante bootstrap [Efron and Tibshirani, 1993]. Los valores para ambas medidas se presentan en escala logarítmica.

[5]Los experimentos se realizaron en un cluster de 40 nodos, cada uno compuesto por un PC INTEL Pentium IV 3 GHz. Cada corrida se realizó en un nodo separado.

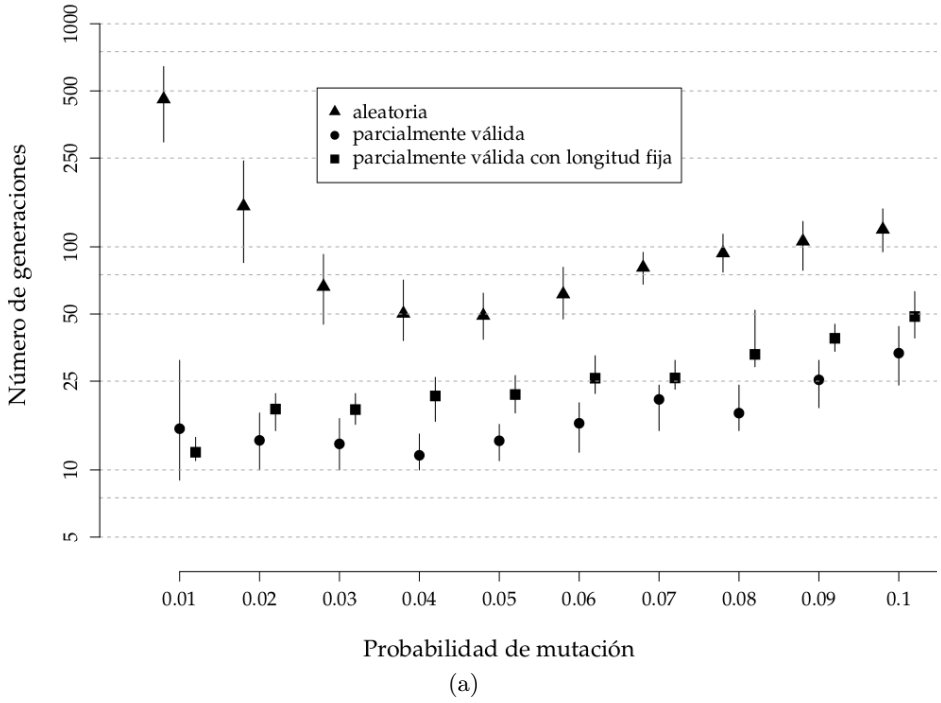
El siguiente aspecto analizado consistió en la evaluación del efecto combinado de la estrategia de inicialización y la probabilidad de mutación. Esta última se analizó particularmente debido a que afecta en forma directa el grado de exploración del algoritmo. Las tasas de mutación elevadas introducen un mayor número de disrupciones pero aumentan la probabilidad de explorar regiones distantes en el espacio de búsqueda.

Claramente, el comportamiento de cada estrategia presenta la misma tendencia en las dos gráficas incluidas en la Figura 3.4. Sin embargo, las diferencias encontradas entre las tres estrategias se aprecian de manera más acentuada cuando se analizan los tiempos. Por lo tanto, el análisis realizado a continuación se hará considerando el tiempo, aunque las conclusiones son igualmente válidas si se analizan las generaciones.

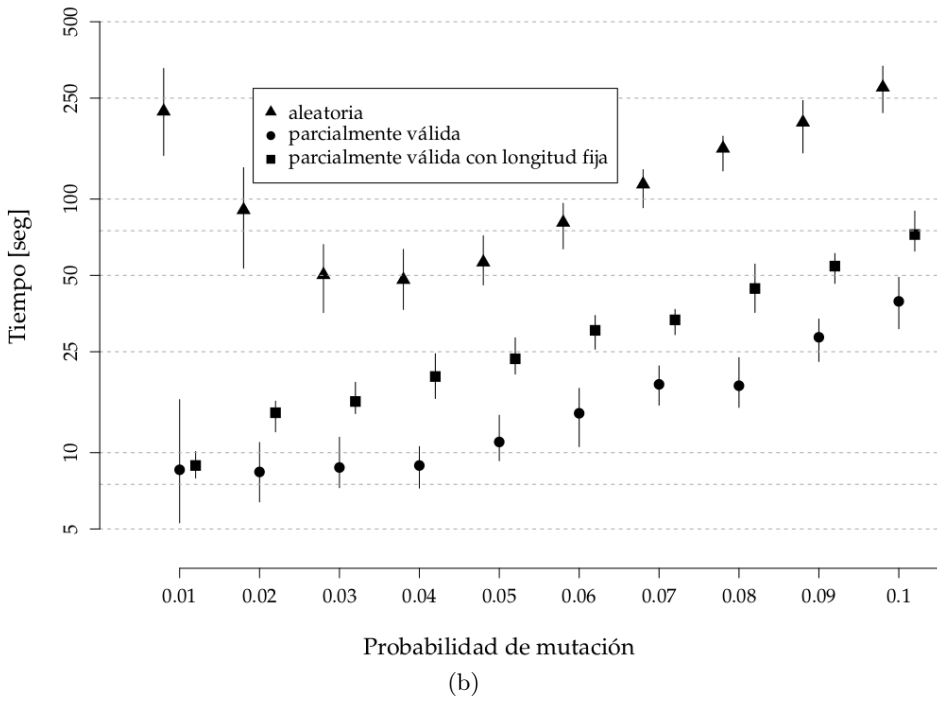
La Figura 3.4b muestra que ambas estrategias de inicialización parcialmente válida presentan diferencias mínimas en los tiempos de búsqueda, aunque sólo para $p_m = 0.01$ son similares. Igualmente, ambas logran una mejora significativa en comparación con la inicialización aleatoria. El menor tiempo de búsqueda para la inicialización *aleatoria* se obtiene con $p_m = 0.04$, aunque no presenta diferencias significativas en el intervalo [0.03,0.05]. La estrategia de inicialización parcialmente válida presenta el mínimo en $p_m = 0.02$, aunque los valores para las probabilidades de mutación en el intervalo [0.01,0.05] no presentan diferencias significativas.

Estos resultados pueden deberse a la forma en que el operador de mutación actúa. En el contexto de la inicialización aleatoria pueden identificarse dos fases de trabajo para este operador. Durante la primera fase el operador ayuda a la construcción de bloques de genes para incrementar la validez promedio de la población. En la segunda fase, una vez que cada individuo está formado por unos pocos bloques, éstos bloques son modificados por el operador de mutación para facilitar el empalme por parte del operador de cruza modificado. En consecuencia, la primera fase puede extenderse si se aplican probabilidades de mutación bajas, mientras que la segunda fase puede volverse más lenta al aplicar tasas de mutación elevadas debido a que el nivel de disrupciones introducidas genera excesiva variabilidad. Por lo tanto resulta esperable que valores intermedios para la tasa de mutación produzcan mejores resultados.

Al analizar ambas estrategias de inicialización parcialmente válida se observa que el efecto que ejerce la tasa de mutación sobre ellas es similar, prin-



(a)



(b)

Figura 3.4: Efecto de la tasa de mutación sobre el funcionamiento de tres estrategias de inicialización analizadas. (a) Número de generaciones, (b) Tiempo de búsqueda. Los símbolos indican el valor de la mediana y las barras verticales el intervalo de confianza de 95 %. La variable independiente se muestra en escala logarítmica.

cipalmente para probabilidades elevadas de mutación donde se incrementa el tiempo de búsqueda. Dado que los cromosomas son parcialmente válidos al inicio y debido a que el operador de cruza también produce individuos parcialmente válidos, la incorporación de un elevado número de mutaciones hace que la búsqueda de soluciones sea más lenta. Por lo tanto, aunque las mutaciones previenen que la búsqueda se detenga en un mínimo local, ellas alejan la búsqueda de la región inicial de la soluciones posibles. Las diferencias encontradas entre las estrategias de inicialización parcialmente válidas podrían ser explicadas por el número inicial de genes en la población. Dado que el número de genes en la población inicial para la inicialización parcialmente válida con longitud fija es mayor que para la otra estrategia, el aumento en la tasa de mutación da como resultado una mayor cantidad de perturbaciones en los cromosomas. Esto es una consecuencia de que el número de mutaciones es proporcional al número de genes en la población. Además, dado que al inicio todos los genes son parcialmente válidos (sólo la última reacción no está conectada), el número de puntos de cruza aumenta y la velocidad a la que los genes innecesarios son removidos se reduce.

Otro hecho llamativo es que el rango donde se produce el incremento exponencial en el tiempo de búsqueda es más amplio para la inicialización parcialmente válida con longitud fija, probablemente debido a que el número de genes que pueden ser mutados al inicio es mayor que para las otras estrategias. Esto lleva a que incluso pequeños valores para la probabilidad de mutación tengan una gran influencia en el número de modificaciones introducidas. Con respecto a las medidas de evaluación de las vías metabólicas (L, ψ y Λ), los resultados alcanzados fueron similares para las tres estrategias. Esto muestra que las estrategias de inicialización parcialmente válidas disminuyen el tiempo de búsqueda sin producir efectos significativos en las características de las soluciones encontradas.

En resumen, la estrategia de inicialización parcialmente válida reduce en forma significativa el tiempo de búsqueda, priorizando regiones del espacio de búsqueda donde es más probable encontrar una solución. Debido a que los menores tiempos de búsqueda fueron obtenidos con la inicialización parcialmente válida, se seleccionó esta estrategia para los experimentos siguientes. Aunque la inicialización parcialmente válida con longitud fija evaluada con $p_m = 0.01$ produce resultados similares en el intervalo [0.01,0.05] a los alcanzados con la inicialización parcialmente válida, se descartó su uso debido

Tabla 3.2: Generaciones requeridas por AEBVML empleando dos operadores de selección. Los valores presentados corresponden a la mediana y el desvío del número de generaciones requeridos para encontrar una vía metabólica. Los valores de k indican los números de individuos utilizados en el torneo.

	Ruleta	Torneo		
		$k = 3$	$k = 5$	$k = 7$
G	11.5 ± 7.5	121 ± 118	118 ± 114	155 ± 153

a que presenta mayor sensibilidad a este parámetro. Finalmente, la tasa de mutación elegida para futuros experimentos es $p_m = 0.04$ debido a que corresponde al mínimo de la inicialización parcialmente válida y produce la mayor diferencia entre las tres estrategias.

3.3.3. Operador de selección

Como se discutió en el Marco Teórico, el operador de selección juega un papel importante en los algoritmos evolutivos ya que es el responsable de aplicar la presión selectiva y determinar qué individuos sobreviven en cada generación. Por tal motivo, se evaluaron los operadores de selección por ruleta y torneo, y se evaluó el efecto que cada uno produce sobre la búsqueda.

Los experimentos empleando torneo se realizaron utilizando los mejores parámetros encontrados en la sección previa y la estrategia de inicialización parcialmente válida. Este operador se evaluó considerando $k = \{3, 5, 7\}$ individuos durante el torneo. La mediana del número de generaciones empleando ambos operadores se presenta en la Tabla 3.2. Claramente se observa que las soluciones encontradas empleando ruleta requieren significativamente menos generaciones ($p < 0.0001$) que con el otro operador estudiado. También es importante destacar que el intervalo de confianza para ruleta es notablemente más reducido que para las tres condiciones de torneo estudiadas. Estos resultados sugieren que el uso de una estrategia de selección con elevada presión selectiva favorece el proceso de búsqueda.

3.3.4. AEBVML vs estrategias de búsqueda clásica

Una vez evaluado el comportamiento de AEBVML frente a diferentes modificaciones de sus parámetros y operadores, se procedió a comparar su

desempeño con el de los algoritmos clásicos BFS y DFS. En el contexto del problema en que se está trabajando, los nodos del árbol de búsqueda generado por BFS y DFS representan compuestos y las ramas corresponden a las transformaciones posibles entre ellos. Además, cada nodo padre es el sustrato requerido para generar un producto o nodo hijo, y una solución corresponde a la primera secuencia de compuestos y transformaciones encontrada por uno de estos algoritmos, que permite producir $\widehat{p}$ a partir de $\widehat{s}$ (nodo raíz). Dado que las soluciones que pueden encontrarse con BFS y DFS dependen del orden en que los operadores son aplicados, una única búsqueda podría producir soluciones sesgadas por el orden elegido. Para evitar este efecto, se aleatorizó el orden en que se aplican los operadores para expandir los nodos durante la búsqueda. Además, se incorporó en ambos algoritmos un control de estados repetidos en el camino evaluado para descartar acciones que generan compuestos ya incluidos en la secuencia. La profundidad del árbol de búsqueda se limitó a 100 transformaciones para aplicar la misma restricción empleada con AEBVML.

La comparación entre los tres algoritmos se realizó considerando el tiempo de búsqueda y el número de transformaciones que contienen las soluciones encontradas por cada uno. Aunque el tiempo es una medida que depende fuertemente de la eficiencia de la implementación, permite conocer a grandes rasgos el desempeño de cada algoritmo. Por su parte, el tamaño de las soluciones brinda información acerca de las características que favorece cada algoritmo, tales como vías metabólicas largas o cortas, con alta o baja variabilidad. La Figura 3.5 presenta los tiempos de búsqueda (cajas en blanco) y la diversidad de tamaños (cajas en gris) de las vías encontradas con los tres algoritmos. Las cajas contienen el 50 % de los datos, y el diagrama completo refleja la variabilidad en la medida analizada. Los límites inferior y superior de cada caja corresponden a los cuartiles Q_1 (25 % de los datos) y Q_3 (75 % de los datos) respectivamente, y el segmento que la divide corresponde a la mediana de la distribución.

Como se observa en la Figura 3.5, DFS emplea tiempos cercanos al segundo para encontrar soluciones, pero el número de transformaciones que contienen las soluciones está sesgado hacia la cantidad máxima permitida (comportamiento característico de este algoritmo). En consecuencia, los resultados obtenidos con DFS son excluidos de análisis posteriores ya que considerar vías metabólicas que sólo relacionan dos compuestos empleando 100

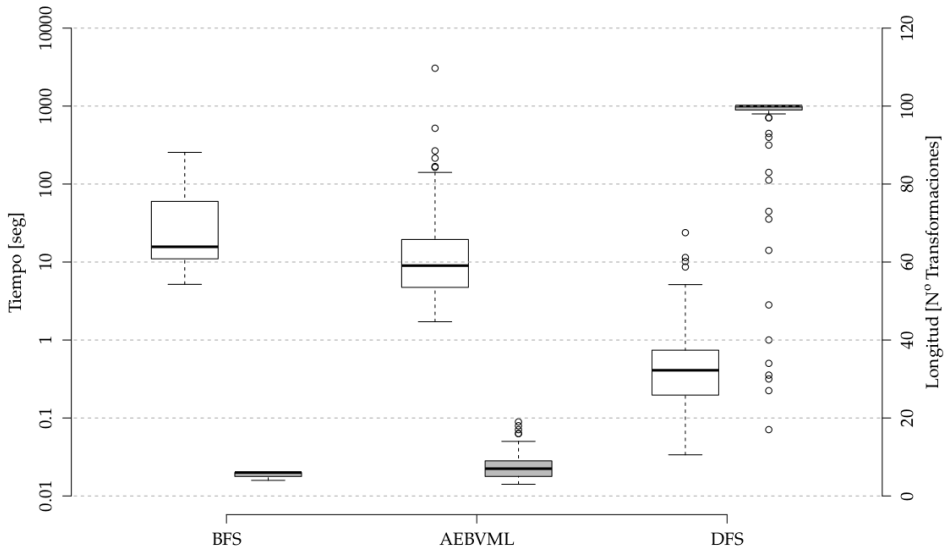


Figura 3.5: Tiempos de búsqueda y longitud de las vías metabólicas encontradas con BFS, AEBVML y DFS. El tiempo de búsqueda se expresa en segundos y se muestra con cajas blancas; la longitud de las vías se expresa en número de transformaciones y se representa mediante cajas grises. El tiempo se presenta en escala logarítmica. Los segmentos que se extienden fuera de las cajas indican los límites máximo y mínimo para los que un valor es considerado atípico (círculos).

transformaciones no son de interés biológico debido a su excesivamente baja eficiencia energética. El tiempo de búsqueda medio para BFS (45.2 s) fue ligeramente menor que para AEBVML (51.2). Sin embargo, estos tiempos deben considerarse similares ya que las diferencias no son significativas. Claramente, AEBVML encuentra soluciones con tamaños intermedios a DFS y BFS, además de encontrar los caminos más cortos. Esta característica es interesante ya que soluciones con una mayor diversidad en el número de reacciones ofrecen alternativas que pueden resultar de interés para el análisis biológico. Por ejemplo, suponiendo que la vía más corta entre dos compuestos contiene cinco transformaciones, una solución de tamaño algo mayor podría incorporar compuestos adicionales de interés que no son considerados en la vía más corta, generando rutas alternativas para la producción del compuesto de interés.

Tabla 3.3: Comparación entre BFS y AEBVML. El tiempo $\widehat{t}$ se expresa en segundos y L en número de transformaciones. $|\Psi|$ indica el número de compuestos que contiene cada agrupamiento. Sólo se indica el número del código KEGG de los compuestos a relacionar.

	Búsqueda ⟶	1	2	3	4	5	6
	$\|\Psi\|$ ⟶		6			12	
Compuestos a Relacionar ⟶		62 - 47	258 - 77	47 - 258	37 - 82	135 - 65	135 - 82
$\widehat{t}$	AEBVML	17.6	8.6	8.3	3.4	15.2	8.4
	BFS	16.1	77.5	142.0	10.2	12.6	12.9
L_M	AEBVML	13	11	17	9	19	18
	BFS	4	5	5	3	5	5
$\widehat{L}$	AEBVML	6.5	7	8	5	9	6
	BFS	4	5	5	3	5	5
L_m	AEBVML	4	5	5	3	5	5
	BFS	4	5	5	3	5	5
ψ_M	AEBVML	3	3	2	4	3	4
	BFS	2	2	2	3	2	2
$\overline{\psi}$	AEBVML	2.4	2.1	2.0	2.3	2.1	2.1
	BFS	2.0	2.0	2.0	2.4	2.0	2.0
Λ	AEBVML	0.50	0.50	0.33	0.33	0.33	0.33
	BFS	0.33	0.33	0.33	0.25	0.17	0.17

Para profundizar en la comparación entre AEBVML y BFS, en la Tabla 3.3 se detallan las medidas evaluadas para las soluciones encontradas con ambos algoritmos. Las filas corresponden a la mediana del tiempo de búsqueda ($\widehat{t}$), el valor máximo (L_M), mínimo (L_m) y la mediana ($\widehat{L}$) del número de transformaciones, el número máximo (ψ_M) y medio ($\overline{\psi}$) de compuestos del agrupamiento que son incorporados en la vía metabólica, y a la tasa de explicación del agrupamiento (Λ). Las columnas indican las búsquedas numeradas de 1 a 6, el número de elementos que posee el agrupamiento del cual fueron seleccionados los compuestos para cada búsqueda y los pares de compuestos entre los que se realizaron las corridas. Lo primero que se aprecia es que BFS emplea tiempos de búsqueda similares a AEBVML, excepto para las búsquedas 2, 3 y 4 donde estos son significativamente mayores ($p < 0.05$). Si bien ambos algoritmos son capaces de encontrar la vía más corta (valores idénticos para L_m), AEBVML es capaz de encontrar soluciones con un mayor número de transformaciones que BFS, lo que se refleja en los valores de mediana significativamente mayores ($p < 0.001$) para el primero. Esta característica es deseable ya que la posibilidad de obtener vías con reacciones adicionales o no incluidas en la vía más corta proveen información acerca de mecanismos alternativos para la síntesis de compuestos. Las medidas relativas al número de compuestos del agrupamiento incluidos en la vía fueron

similares para ambos algoritmos, siendo significativamente mayor ($p < 0.01$) el número de compuestos del agrupamiento relacionados por AEBVML que BFS sólo en la búsqueda 1. En términos generales, la tasa de explicación del agrupamiento (Λ) refleja que ningún algoritmo presenta una tendencia marcada a incorporar compuestos del agrupamiento. Las pequeñas diferencias observadas para AEBVML son solamente debidas a la variabilidad en el tamaño de las vías.

En resumen, la comparación de AEBVML con DFS y BFS revela que DFS encuentra soluciones con el máximo número de transformaciones permitido, como es lo esperado para este algoritmo. Esta característica dificulta su aplicación ya que requiere especificar un tamaño que no es conocido *a priori*. AEBVML tiene un desempeño similar a BFS en lo referente al tiempo de búsqueda. Sin embargo, el algoritmo evolutivo es capaz de encontrar los caminos más cortos y otros con mayor número de transformaciones, los cuales pueden proveer mecanismos alternativos para la síntesis del producto de interés.

3.3.5. Casos de estudio

En esta sección se analiza el desempeño del algoritmo evolutivo propuesto frente a dos problemas biológicos concretos. El primero consiste en la búsqueda de una vía metabólica alternativa a la vía estándar de la glicólisis[6], que permita relacionar dos compuestos que forman parte de ella. El segundo problema consiste en la búsqueda de una vía metabólica que relacione dos compuestos para los que no existe una vía de referencia conocida en *Arabodopsis thaliana*. En ambos casos se emplea el conjunto de transformaciones construido en la Sección 3.2.1, y el conjunto de parámetros utilizado en la Sección 3.3.4.

3.3.5.1. Caso 1

El problema considerado a continuación consiste en la búsqueda de una vía metabólica entre los compuestos C00103 (α-D-glucosa-1P) y C00631 (glicerato-2P), que sea alternativa a la vía de referencia de la glicólisis. En esta vía se requiere un mínimo de cinco reacciones para producir C00631 a partir de C00103. A los fines de realizar una comparación de las característi-

[6] `http://www.genome.jp/kegg-bin/show_pathway?map00010`

cas de las vías que se encuentren, la búsqueda también se realizó empleando BFS.

La Figura 3.6 presenta la vía de la glicólisis junto a las vías encontradas por AEBVML (líneas de trazos cortos) y BFS (líneas de trazo largo). Debe notarse que la figura está compuesta por tres vías metabólicas de referencia indicadas con rectángulos grises, cada una agrupando compuestos (círculos) y reacciones (flechas) característicos. Esto es sólo un marco de referencia para el caso de estudio, pero debe tenerse presente que la finalidad del método es ser aplicado a los casos donde no se conozcan vías previamente. Los compuestos inicial y final se destacan como hexágonos. En todos los casos los compuestos y las reacciones se indican de acuerdo con la codificación de KEGG.

Lo primero que se observa es que BFS encuentra una vía metabólica de cinco reacciones, alternativa a la vía estándar de la glicólisis. Esta alternativa sustituye las reacciones R04779/R09084 y R01070 de la vía estándar por la reacción R01827 de la vía de las pentosas. La vía encontrada por AEBVML contiene seis reacciones, al igual que la glicólisis, y sólo la primera de ellas forma parte de la vía estándar. Las cinco reacciones restantes pertenecen a la vía de la manosa y la fructosa, y a la vía de las pentosas. Comparando ambas soluciones se aprecia que el algoritmo evolutivo provee una solución interesante basada en un nuevo conjunto de reacciones diferente a las empleadas en la glicólisis, indicando que la producción de C00631 puede realizarse a través de mecanismos alternativos de síntesis.

3.3.5.2. Caso 2

En este experimento se plantea la búsqueda de una vía metabólica que relacione los compuestos C01019 (Fucosa) y C00037 (Glicina). No se conoce una vía que relacione ambos compuestos en *A. thaliana*, por lo que se realiza la búsqueda empleando todas las reacciones almacenadas en KEGG. Se emplearon los mismos parámetros utilizados previamente para el algoritmo.

La Figura 3.7 presenta la vía metabólica encontrada por AEBVML. Claramente puede apreciarse que la vía descubierta está compuesta por reacciones asociadas a tres vías metabólicas de referencia, y constituye un mecanismo alternativo para producir Glicina. La importancia de esto radica en que este aminoácido actúa como precursor de un gran número de compuestos [Li et al., 2003], y una vía alternativa para su síntesis podría explicar la redundancia que caracteriza a todo proceso esencial.

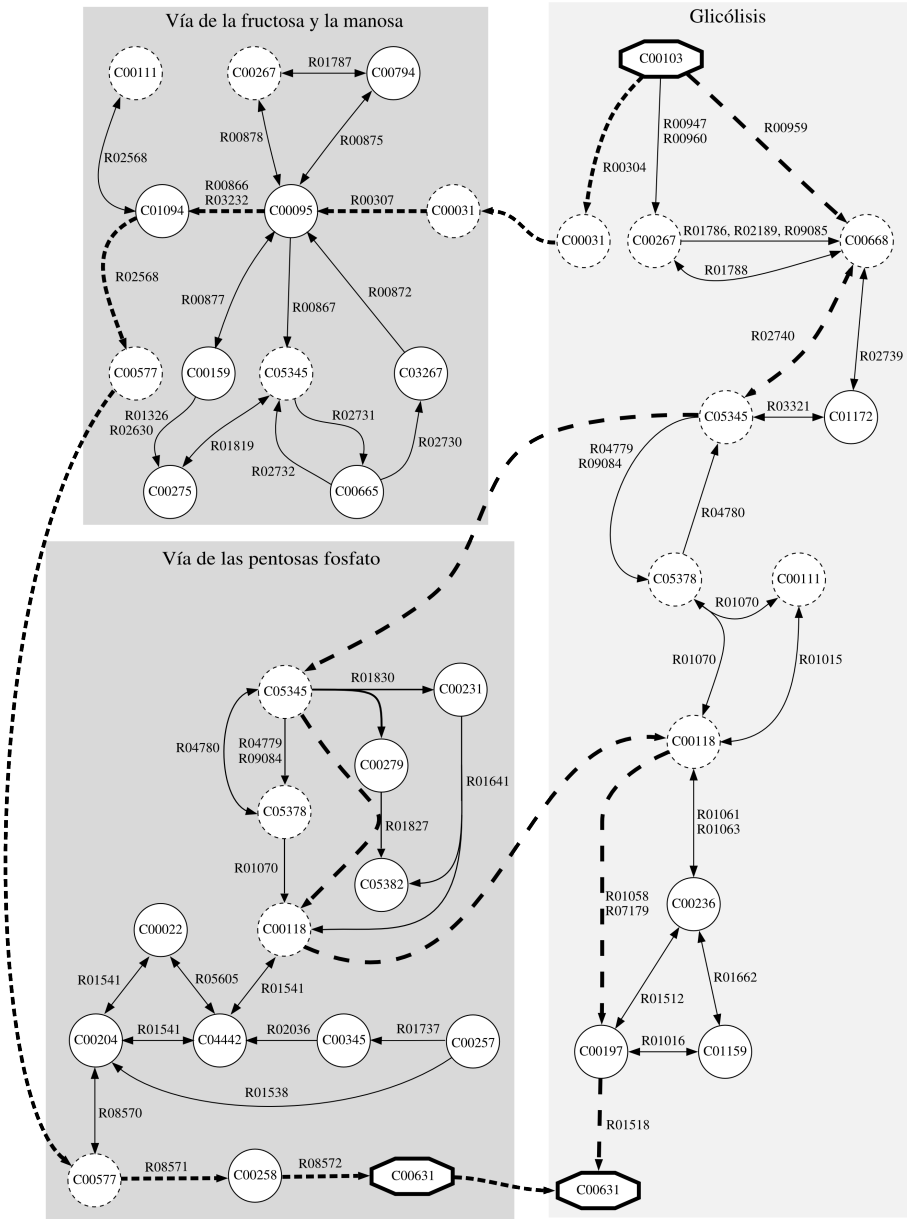


Figura 3.6: Vías metabólicas alternativas a la glicólisis encontradas por BFS y AEBVML. La solución producida por AEBVML se indica en líneas de trazo corto; la solución encontrada por BFS se muestra en líneas de trazo largo. Los compuestos inicial (C00103) y final (C00631) se indican con hexágonos de bordes gruesos. Los rectángulos grises identifican agrupación de compuestos (círculos) y reacciones (flechas) pertenecientes a una misma vía metabólica de referencia. Todos los nombres se indican con la codificación empleada en KEGG.

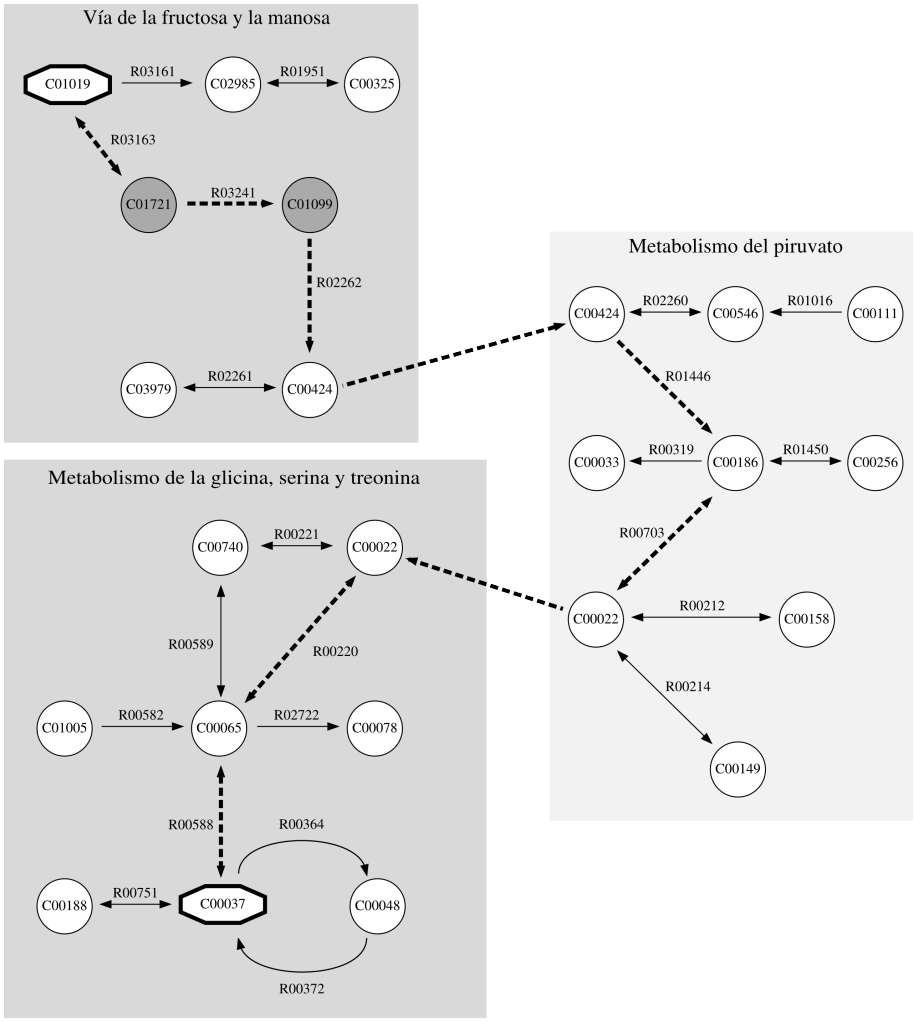


Figura 3.7: Vía metabólica que relaciona los compuestos C01019 y C00037. La solución encontrada se indica en líneas de trazo corto. Los círculos resaltados indican compuestos que aún no han sido reportados en *A. thaliana*. Los compuestos inicial (C01019) y final (C00037) se indican con hexágonos de bordes gruesos. Los rectángulos grises agrupan compuestos (círculos) y reacciones (flechas) pertenecientes a una misma vía metabólica de referencia. Todos los nombres se indican con la codificación empleada en KEGG.

Una posible manera de estimar la factibilidad de esta vía consiste en determinar la proporción de compuestos y reacciones que no se encuentran presentes en el organismo de estudio. Todos los compuestos que componen esta vía metabólica se encuentran reportados para *A. thaliana* [Mueller et al., 2003], con excepción de los compuestos C01721 (L-Fuculosa) and C01099 (L-Fuculosa-1P), los cuales se indican en la figura mediante círculos grises. Una posibilidad es que estos compuestos no hayan sido medidos experimentalmente todavía. Con respecto a las tres reacciones en donde intervienen estos compuestos, dos de ellas han sido identificadas en diferentes especies. Un gen que codifica la enzima para la reacción R03163 se encuentra reportado para la planta *Oryza sativa*, mientras que otro gen asociado a la reacción R02262 fue reportado para la planta leguminosa *Medicago truncatula* [Liang et al., 2008]. Al igual que como ocurre con estas dos reacciones, no ha sido reportado aún en la literatura un gen asociado a la reacción R03241 para *A. thaliana* (ni a otra especie vegetal). Sin embargo, la función para una gran cantidad de genes en esta especie todavía es desconocida [Lan et al., 2007]. Es posible entonces que las tres reacciones puedan ser efectuadas por esta especie, siendo necesario corroborar esta nueva vía propuesta realizando experimentos biológicos apropiados.

3.4. Comentarios finales

En este capítulo se abordó el problema de buscar vías metabólicas lineales que relacionaran dos compuestos. La propuesta presentada consiste en un algoritmo evolutivo denominado AEBVML, que modela en cada cromosoma una secuencia de transformaciones entre compuestos, para producir un compuesto deseado partiendo de otro que se especifique.

El algoritmo propuesto fue evaluado empleando distintas estrategias de inicialización para un rango amplio de probabilidades de mutación. Los resultados indicaron que el uso de una estrategia de inicialización que construye cromosomas con transformaciones conectadas secuencialmente disminuye el número de generaciones empleados en la búsqueda. También se observó que probabilidades de mutación baja y un operador con alta presión selectiva contribuyen a reducir el número de generaciones. El desempeño de AEBVML se comparó con DFS y BFS. DFS realizó las búsquedas en los menores tiempos, pero las soluciones encontradas carecen de interés biológico ya que poseen un

elevado número de reacciones. La comparación con BFS muestra que ambos algoritmos emplean tiempos de búsqueda similares. Sin embargo, además de encontrar los caminos más cortos, AEBVML encuentra vías con una mayor variedad de tamaños. Este resultado es interesante porque muestra que el algoritmo evolutivo explora mecanismos alternativos para la síntesis de los compuestos, en los que pueden participar reacciones pertenecientes a distintas vías metabólicas conocidas. Los casos de estudio propuestos condujeron a resultados que validan la utilidad del algoritmo en problemas de aplicación práctica, produciendo vías alternativas a las conocidas e incluso planteando la posibilidad de una nueva vía metabólica.

El problema abordado en este capítulo es un caso particular de otro más complejo que consiste en relacionar un conjunto de compuestos mediante una red de relaciones. Aunque algunos elementos del algoritmo propuesto podrían modificarse para que la vía relacionara más compuestos, las soluciones producidas aún conservarían la estructura lineal. En el siguiente capítulo se aborda el problema más general de búsqueda de vías entre varios compuestos, y se propone un nuevo algoritmo evolutivo que contempla redes de interacciones.

Capítulo 4

Búsqueda de vías metabólicas ramificadas

El algoritmo evolutivo presentado en el capítulo anterior realiza la búsqueda de vías metabólicas que relacionan dos compuestos. Aunque su aplicación se encuentra limitada a vías metabólicas lineales, la experiencia adquirida en el diseño de los operadores y la función de aptitud resultan de utilidad para el desarrollo de una nueva propuesta que permita relacionar un número mayor de compuestos mediante una red de relaciones.

La propuesta desarrollada en este capítulo consiste en un nuevo algoritmo evolutivo que emplea conjuntos de compuestos para buscar vías metabólicas que los relacionan, mediante redes con diferentes topologías. El presente capítulo está organizado de la siguiente manera: la Sección 4.1 presenta el algoritmo evolutivo propuesto. Se describe el modelo de conjuntos anidados y su codificación en el cromosoma, además de los operadores genéticos utilizados. También se analiza la función de aptitud y las medidas que la componen. La Sección 4.2 presenta los datos empleados, su procesamiento, y las medidas utilizadas para evaluar propiedades de las soluciones y desempeño del algoritmo. La Sección 4.3 expone los resultados alcanzados. Finalmente, la Sección 4.4 presenta los comentarios finales.

4.1. Algoritmo evolutivo basado en conjuntos anidados de compuestos

4.1.1. Modelo de conjuntos anidados y codificación en el cromosoma

Las redes biológicas están conformadas por grupos de reacciones bioquímicas r_k que comparten compuestos. Cada reacción representa una relación entre dos conjuntos de compuestos conocidos como sustratos y productos, donde los elementos del primer conjunto interactúan para generar los elementos del segundo. Así, una reacción puede ser representada como $S(r_k) \overset{r_k}{\longleftrightarrow} P(r_k)$, donde $S(r_k)$ y $P(r_k)$ corresponden a los sustratos y productos de la reacción. Una reacción sólo puede efectuarse si dispone de todos los sustratos que utiliza. Algunos de esos sustratos pueden estar disponibles en el medio donde ocurre la reacción, pero otros deben ser producidos por alguna reacción específica. Sin embargo, cada reacción puede incrementar la diversidad de los compuestos disponibles debido a que adiciona sus productos al conjunto de compuestos disponibles y permite que nuevas reacciones puedan efectuarse. Siguiendo esta idea, una vía metabólica puede verse como un conjunto de reacciones que se llevan a cabo de forma secuencial, y un cromosoma que utilice esta estructura podría ser capaz de modelar diferentes tipos de vías metabólicas.

La Figura 4.1 ejemplifica una vía metabólica codificada en un cromosoma junto con los conjuntos anidados asociados a las reacciones. Los sustratos necesarios para la reacción r_k deben estar disponibles en el conjunto anidado C_{k-1}, de otra forma la reacción no será válida y el conjunto anidado permanecerá sin modificaciones ($C_k = C_{k-1}$). Por ejemplo, los sustratos para la reacción r_k se encuentran disponibles en el conjunto $C_{k-1} = C_{k-2} \cup P(r_{k-1})$. Así, la reacción r_k produce el nuevo conjunto $C_k = C_{k-1} \cup P(r_k)$. De esta forma, los conjuntos anidados continúan generándose hasta que se alcanza el conjunto C_N.

4.1.2. Descripción del algoritmo

En esta sección se describe el *algoritmo evolutivo basado en conjuntos anidados de compuestos* (AEBCAC), que permite encontrar vías metabólicas que relacionan un conjunto de compuestos D especificado. Dado un

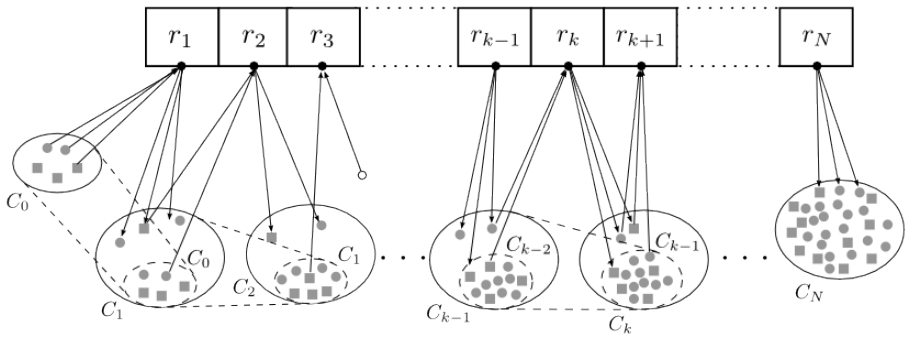


Figura 4.1: Representación del modelo de conjuntos anidados en un cromosoma. *Arriba:* cromosoma que codifica las reacciones de una ruta metabólica. *Abajo:* Los conjuntos de compuestos disponibles para cada reacción (líneas continuas) y conjuntos anidados (líneas discontinuas). Los cuadrados y los círculos corresponden a compuestos abundantes y no abundante, respectivamente. El círculo vacío usado por r_3 corresponde a un sustrato requerido por la reacción que no está disponible en C_2.

conjunto A, conteniendo compuestos abundantes de libre disponibilidad, este algoritmo busca una red que relaciona los compuestos en D mediante la evolución de secuencia de reacciones. El pseudocódigo del AEBCAC se presenta en el Algoritmo 4.1.

La estructura general del algoritmo no presenta diferencias con la empleada por AEBVML. Sin embargo, se han introducido modificaciones en la función de aptitud y en algunos operadores debido al nuevo modelo empleado. El algoritmo comienza con la definición de una aptitud mínima tolerada $\mathcal{A}^\epsilon$ que debe satisfacer una solución del problema, y luego procede a inicializar y evaluar la aptitud de cada individuo $\mathbf{c}_i$ que compone la población $\mathbb{P}^0$ (líneas 2–3). La población es inicializada como un conjunto de subpoblaciones de igual tamaño, cada una de las cuales emplea uno de los compuestos $d_I \in D$ como compuesto inicial. Esto lleva a que la vía metabólica modelada por cada subpoblación relacione los compuestos en D partiendo de uno de ellos. Cada individuo $\mathbf{c}_i$ en una subpoblación tiene asociado un compuesto d_I^i, y es inicializado usando el mismo conjunto C_0^i de compuestos disponibles a través de un proceso iterativo que sólo incorpora reacciones que pueden efectuarse a partir de los compuestos disponibles y de los productos de reacciones previamente seleccionadas (esto se describirá en detalle en la Sección 4.1.4). La evaluación de la aptitud se realiza empleando medidas que ca-

Algoritmo 4.1: Algoritmo evolutivo basado en conjuntos anidados de compuestos.

1 $\mathcal{A}^{\epsilon} \leftarrow$ mínima aptitud tolerada
2 $\mathbb{P}^0 \leftarrow$ inicializar individuos
3 $\mathcal{A}(\mathbf{c}_i)_{\forall \mathbf{c}_i \in \mathbb{P}^0} \leftarrow$ evaluar la aptitud de la población
4 $G \leftarrow 0$
5 **while** $(\mathcal{A}(\mathbf{c}_i)_{\forall \mathbf{c}_i \in \mathbb{P}^G} < \mathcal{A}^{\epsilon})$ ***and*** $(G < G_m)$ **do**
6 $\quad \mathbf{c}^* \leftarrow$ extraer el individuo más apto de $\mathbb{P}^G$
7 $\quad \mathbb{X}^G \leftarrow$ seleccionar de $\mathbb{P}^G$ los padres a cruzar
8 $\quad \mathbb{C}^G \leftarrow$ cruzar $\mathbb{X}^G$ y producir descendencia
9 $\quad \mathbb{C}^G \leftarrow$ mutar $\mathbb{C}^G$
10 $\quad \mathbb{P}^{G+1} \leftarrow \{\mathbf{c}^*\} \cup \mathbb{X}^G \cup \mathbb{C}^G$
11 $\quad \mathbf{A} \leftarrow \left[\mathcal{A}(\mathbf{c}_i) = \frac{1}{4}\left(\mathcal{V}(\mathbf{c}_i) + \mathcal{R}(\mathbf{c}_i) + \mathcal{N}(\mathbf{c}_i) + \mathcal{C}(\mathbf{c}_i)\right) \ \forall\, \mathbf{c}_i \in \mathbb{P}^{G+1}\right]$
12 $\quad G \leftarrow G + 1$
13 **return** $\arg\max_{\forall \mathbf{c}_i \in \mathbb{P}^G}\{\mathcal{A}(\mathbf{c}_i)\}$

racterizan a las vías metabólicas (esto se describirá en detalle en la Sección 4.1.3).

El paso siguiente en el algoritmo consiste en evaluar el criterio de finalización, que determina si el proceso evolutivo debe continuar (línea 5). En particular, el criterio empleado está compuesto por dos condiciones. La primera consiste en limitar el número máximo de generaciones que pueden transcurrir antes de detener la ejecución del algoritmo si no se encuentra una solución. La segunda condición consiste en encontrar un individuo cuya aptitud sea mayor a $\mathcal{A}^{\epsilon}$. Para el caso de $\mathcal{A}^{\epsilon} = 1$, esto implica que todas las reacciones en la vía metabólica son válidas y producen al menos un compuesto no producido por otra reacción, y que cada compuesto perteneciente al conjunto $D - \{d_I^i\}$ está conectado a d_I^i mediante una secuencia de reacciones válidas.

El proceso evolutivo consiste en seis operaciones que se repiten cíclicamente hasta que logra satisfacerse el criterio de finalización. Primero se extrae el individuo $\mathbf{c}^*$ con mayor aptitud de la población (línea 6) y se seleccionan los individuos $\mathbb{X}^G$ que serán los padres de la siguiente generación empleando el método de torneos (línea 7). Luego se realiza la cruza de los individuos $\mathbb{X}^G$ previamente seleccionados (línea 8), empleando un operador de cruza modificado para seleccionar dos puntos de cruza en uno de los padres (esto se describirá en detalle en la Sección 4.1.5.2). Como resultado se produce la

descendencia $\mathbb{X}^G$ de la generación actual, la cual es sometida a mutaciones en el paso siguiente (línea 9). El operador empleado tiene la capacidad de eliminar una reacción o insertar una nueva. En tal caso, utiliza información de los conjuntos anidados para seleccionar la reacción que será incorporada (esto se describirá en detalle en la Sección 4.1.5.1). Finalmente, se aplica un esquema de reemplazo que conserva el tamaño de la población construyendo la nueva generación $\mathbb{P}^{G+1}$ con el individuo $\mathbf{c}^*$ de mayor aptitud, los padres seleccionados para la cruza $\mathbb{X}^G$ y su descendencia $\mathbb{C}^G$ (línea 10). Esta población es evaluada nuevamente para determinar la aptitud de cada individuo en $\mathbb{P}^{G+1}$ (línea 11), y se verifica si se cumple el criterio de finalización (línea 5).

Las vías metabólicas encontradas por el algoritmo contienen reacciones válidas que, en su mayoría, generan secuencias que relacionan los compuestos especificados. Sin embargo, algunas no lo hacen y son descartadas en un paso posterior para lograr mejores soluciones. Por lo tanto, una vez obtenida la red se realiza un proceso de poda para conservar sólo las reacciones indispensables para establecer las conexiones. Este proceso consiste en retirar de la red todas las reacciones que no producen los compuestos deseados o que no participan de su síntesis. De esta forma, la vía resultante sólo contiene las reacciones que son indispensables para establecer las conexiones entre los compuestos que se desea relacionar.

4.1.3. Función de aptitud

La función de aptitud propuesta en el Capítulo 3 demostró ser adecuada para guiar el proceso de búsqueda, por lo que se tomaron las medidas empleadas en ella y se adaptaron a este nuevo modelo. Como resultado, la función de aptitud $\mathcal{A}$ aplicada sobre el cromosoma $\mathbf{c}_i$ se define como

$$\mathcal{A}(\mathbf{c}_i) = \frac{1}{4}\left[\mathcal{V}(\mathbf{c}_i) + \mathcal{R}(\mathbf{c}_i) + \mathcal{P}(\mathbf{c}_i) + \mathcal{C}(\mathbf{c}_i)\right]. \tag{4.1}$$

donde $\mathcal{V}$ evalúa la validez de la vía metabólica, $\mathcal{R}$ mide la fracción de los compuestos que se encuentran relacionados, $\mathcal{P}$ evalúa la cooperación de las reacciones y $\mathcal{C}$ mide la conectividad de la red. La función de aptitud, al igual que sus términos, varía en el intervalo $[0, 1]$ y alcanza el máximo cuando se encuentra una solución. Una vía metabólica, para ser considerada solución, debe cumplir dos condiciones: i) los sustratos de cada reacción deben estar

disponibles, ii) debe existir una secuencia válida de reacciones que relacionen d_I^i con los compuestos en $D - \{d_I^i\}$.

4.1.3.1. Validez

El término $\mathcal{V}(\cdot)$ de (4.1) cumple la misma función que su par en AEBVML (página 40). Sin embargo, la validez está ahora definida en términos de la disponibilidad de todos los sustratos, por lo que esta medida indica la proporción de las reacciones de la vía metabólica que dispone de los sustratos necesarios para que puedan efectuarse. Su cálculo está basado en el concepto de conjuntos anidados: una reacción r_k es válida si $S(r_k) \subseteq C_{k-1}$. Esta medida se calcula como

$$\mathcal{V}(\mathbf{c}_i) = \frac{1}{|\mathbf{c}_i|} \sum_{k=1}^{|\mathbf{c}_i|} \mathbf{1}_{S(r_k) \subseteq C_{k-1}}, \tag{4.2}$$

donde $|\mathbf{c}_i|$ es el número de genes en el cromosoma, y $\mathbf{1}_{A \subseteq B}$ es la *función indicadora*, que toma valor 1 si $A \subseteq B$ y 0 si $A \nsubseteq B$. La validez de una vía metabólica es máxima cuando todas las reacciones incluidas en $\mathbf{c}_i$ disponen de los sustratos que requieren. Para explicar el cálculo de esta medida, tomemos como ejemplo la Figura 4.1. La reacción r_1 es válida ya que sus sustratos $S(r_1)$ están disponibles en el conjunto C_0. De manera similar, los sustratos $S(r_2)$ de la reacción r_2 están disponibles en el conjunto $C_1 = C_0 \cup P(r_1)$, por lo que ésta también es una reacción válida. Cuando se analiza r_3 se observa que no todos los sustratos $S(r_3)$ están disponibles en $C_2 = C_1 \cup P(r_2)$, por lo que r_3 no será una reacción válida y $C_3 = C_2$ debido a que los productos $P(r_3)$ no pueden ser producidos por la reacción.

4.1.3.2. Compuestos relacionados

El término $\mathcal{R}(\cdot)$ de (4.1) evalúa dos aspectos de la vía metabólica: i) el compuesto d_I^i es sustrato de al menos una reacción, ii) la proporción de compuestos de los compuestos $D - \{d_I^i\}$ que son producidos. Esta medida es una modificación de la medida de extremos válidos (página 40) propuesta para AEBVML, ya que debe contemplar un mayor número de compuestos que no necesariamente son el inicio y fin de la secuencia. El cálculo de esta

medida se realiza según

$$\mathcal{R}(\mathbf{c}_i) = \frac{1}{2}\left(|S^*(\mathbf{c}_i) \cap \{d_I^i\}| + \frac{|P^*(\mathbf{c}_i) \cap \left(D - \{d_I^i\}\right)|}{|D - \{d_I^i\}|}\right), \tag{4.3}$$

donde $S^*(\mathbf{c}_i) = \cup_{k=1}^{|\mathbf{c}_i|} S(r_k)$ y $P^*(\mathbf{c}_i) = \cup_{k=1}^{|\mathbf{c}_i|} P(r_k)$ son los conjuntos de todos los sustratos y productos de la red, respectivamente. Esta medida alcanza su valor máximo cuando una reacción emplea d_I^i como sustrato, y los compuestos en $D - \{d_I^i\}$ son producidos por alguna reacción.

4.1.3.3. Tasa de compuestos únicos

El término $\mathcal{P}(\cdot)$ de (4.1) evalúa la proporción de las reacciones en la vía metabólica que expanden sus conjuntos anidados asociados. Visto de otro modo, determina la fracción de las reacciones que generan al menos un compuesto no producido previamente. Esta medida cumple la función conjunta de la tasa de transformaciones únicas y de la tasa de compuestos únicos (página 41) empleadas por AEBVML. En el primer caso, incorporar la misma reacción dos o más veces no expande el conjunto anidado asociado y es considerada una reacción repetida. En el segundo caso, la incorporación de reacciones alternativas para producir un dado compuesto generan redundancia en la producción de dicho compuesto, haciendo que las nuevas instancias conduzcan a compuestos repetidos. Esta medida se define como

$$\mathcal{P}(\mathbf{c}_i) = \frac{1}{|\mathbf{c}_i|}\sum_{k=1}^{|\mathbf{c}_i|} \mathbf{1}_{P(r_k) \not\subseteq C_{k-1}}. \tag{4.4}$$

El valor máximo es alcanzado cuando cada reacción produce al menos un nuevo compuesto en la vía.

4.1.3.4. Conectividad

El término $\mathcal{C}(\cdot)$ de (4.1) evalúa la proporción de los compuestos incluidos en D para los que existe una secuencia válida de reacciones que permite producirlos a partir de d_I^i. En AEBVML la conectividad se evalúa en forma implícita a través de la medición de la validez, ya que debido al modelado empleado sólo era necesario generar una vía completamente válida para unir dos compuestos. Al emplear el modelo de los conjuntos anidados, la situación es diferente. Supongamos que se dispone de los compuestos a y b. El com-

Algoritmo 4.2: Búsqueda de compuestos vinculados a d_I^i.

```
1 Z ← {d_I^i}
2 for k ← 1 to |c_i| do
3     if |S(r_k) ∩ C_{k-1}^i| = |S(r_k)| then
4         C_k^i ← P(r_k) ∪ C_{k-1}^i
5         if |S(r_k) ∩ Z| > 0 then
6             Z ← Z ∪ (P(r_k) − C_0^i)
7     else
8         C_k^i ← C_{k-1}^i
9 return Z
```

puesto a podría emplearse como sustrato de una secuencia que no produce el compuesto b, aunque sí una secuencia válida, y el compuesto b podría ser producido a partir de una secuencia válida que no emplea el compuesto a como sustrato inicial. Esto último es posible ya que la reacción inicial sólo requiere disponer de los sustratos para realizarse. Cuando se busca relacionar un mayor número de compuestos la situación se vuelve más compleja, por lo que se requiere de una medida que evalúe en forma explícita las relaciones encontradas entre los compuestos. El cálculo de esta medida se realiza en dos pasos. El primero consiste en generar un conjunto Z (que incluye a d_I^i) que contiene los productos de todas las reacciones que tienen alguna relación con d_I^i. Esto se realiza a través del Algoritmo 4.2.

El algoritmo se inicia con un conjunto de compuestos Z que sólo contiene al compuesto inicial d_I^i para el cromosoma $\mathbf{c}_i$ que se está evaluando. Luego cada reacción del cromosoma es sometida a dos evaluaciones. La primera consiste en verificar que los sustratos de la reacción r_k están presentes en el conjunto anidado C_{k-1} (línea 3). En caso de tratarse de una reacción válida, se actualiza el conjunto anidado con los productos de la reacción y se realiza la segunda verificación (línea 5). Esta consiste en verificar si alguno de los sustratos de la reacción está contenido en el conjunto Z. En caso de haber al menos un compuesto, el conjunto se actualiza con los productos de la reacción que no forman parte del conjunto inicial de compuestos C_0^i. Cuando la reacción es inválida (línea 7), el conjunto C_k es una copia del conjunto C_{k-1}. El algoritmo devuelve un conjunto de compuestos que son empleados para relacionar d_I^i con cada compuesto en $D - \{d_I^i\}$. A partir de este conjunto

se calcula la conectividad como

$$\mathcal{C}(\mathbf{c}_i) = \frac{|Z \cap D| - 1}{|D| - 1}. \tag{4.5}$$

El valor máximo de la medida se alcanza cuando se encuentra un camino que vincula d_I^i con todos los otros compuestos en D.

4.1.4. Estrategia de inicialización basada en conjuntos anidados

Los resultados logrados en el capítulo anterior indican que el uso de una estrategia de inicialización que contemple la validez de las reacciones mejora significativamente los resultados frente a una inicialización aleatoria, ya que esta última podría conducir a soluciones muy pobres. La estrategia de inicialización empleada debe ser modificada para adaptarse al nuevo modelo y para contemplar diferentes compuestos iniciales. Este requisito es de especial interés cuando no se dispone de información acerca de cuál compuesto emplear para iniciar la búsqueda. Para esto se plantea una estrategia basada en la creación de subpoblaciones de individuos, cada una inicializada con uno de los compuestos a relacionar, que serán puestas a competir durante la evolución para seleccionar el compuesto inicial. El Algoritmo 4.3 describe los pasos del proceso de inicialización, teniendo en cuenta estos requisitos.

El algoritmo requiere la definición de un conjunto D de metabolitos a relacionar, y un conjunto A de compuestos abundantes tales como agua o dióxido de carbono. A continuación se define en forma automática un nuevo conjunto $A' = A \cup E$, donde E corresponde al conjunto de todos los compuestos que no son sintetizados por ninguna reacción en el espacio de búsqueda (línea 1). La inicialización de la población se divide en dos etapas, en la que se definen las subpoblaciones y otra en donde se inicializan los individuos propiamente.

La primera etapa abarca las líneas 3–7 y consiste en la determinación del número de subpoblaciones que serán creadas durante la inicialización. El proceso consiste en crear un conjunto R_j para cada compuesto en $d_j \in D$, que contiene los sustratos de todas las reacciones que emplean d_j para producirse (línea 5). Los compuestos pertenecientes a D que no actúen como sustratos de ninguna reacción tendrán asociado un conjunto R_j vacío y no serán empleados para inicializar ninguna población (línea 6).

Algoritmo 4.3: Estrategia de inicialización basada en conjuntos anidados.

```
1  A' ← A ∪ E
2  n ← 0
3  for j ← 1 to |D| do
4  |   d_j ← extraer un compuesto de D
5  |   R_j ← ⋃_{∀r_m/d_j∈S(r_m)} S(r_m)
6  |   if R_j ≠ ∅ then
7  |   |   n ← n + 1
8  for j ← 1 to n do
9  |   for i ← 1 to M/n do
10 |   |   inicializar c_i como una lista vacía
11 |   |   C_0^i ← (A' ∪ R_j) − (D − {d_j})
12 |   |   Q ← {r_m/|S(r_m) ∩ C_0^i| = |S(r_m)| ∧ d_j ∈ S(r_m)}
13 |   |   k ← 1
14 |   |   while k ≤ N_I and Q ≠ ∅ do
15 |   |   |   r_k ← seleccionar una reacción r_k ∈ Q no incluida en c_i
16 |   |   |   incorporar r_k a c_i
17 |   |   |   C_k^i ← C_{k−1}^i ∪ P(r_k)
18 |   |   |   k ← k + 1
19 |   |   |   Q ← {r_m/|S(r_m) ∩ C_{k−1}^i| = |S(r_m)|}
20 |   |   ℙ ← insertar c_i
21 return ℙ
```

La segunda etapa engloba las líneas 8–19 y describe el proceso de inicialización de los individuos propiamente dicha. Primero se determina el número de subpoblaciones y la proporción de los M individuos de la población que formarán parte de cada subpoblación. En caso de que el número de integrantes de cada subpoblación no sea un valor entero, este número se redondea al entero superior y al finalizar la etapa de inicialización, se remueven individuos de forma aleatoria hasta conservar M individuos en la población. Cada individuo $\mathbf{c}_i$ es inicializado como una lista vacía en la que se incorporan secuencialmente las reacciones. Luego, dados un compuesto d_j y su conjunto R_j no nulo asociado, se construye el conjunto C_0^i de compuestos disponibles para inicializar el cromosoma $\mathbf{c}_i$ (línea 11). Este conjunto contiene los compuestos abundantes (A), los externos (E) y todos los compuestos necesarios (R_j) para que cada reacción que emplea d_j pueda efectuarse. Como se presentó en la Sección 4.1.2, cada cromosoma $\mathbf{c}_i$ tiene asociado un com-

puesto inicial d_I^i que corresponde al compuesto d_j empleado para construir R_j. Empleando el conjunto de compuestos iniciales creado en el paso anterior se construye el conjunto Q, que contiene todas las reacciones posibles a partir de los compuestos abundantes y que emplean el compuestos d_I^i como sustrato (línea 12). A continuación se construye el cromosoma siguiendo un proceso iterativo de adición de genes (líneas 15–19), que se detiene cuando se alcanza el tamaño N_I máximo de cromosoma permitido durante el inicio, o no se dispone de nuevos genes para incorporar en el cromosoma. En cada ciclo se realiza la extracción de una reacción r_k de Q que no se encuentre previamente en $\mathbf{c}_i$ (línea 15) y se inserta en el cromosoma. Luego se actualiza C_{k-1}^i con los productos de r_k (línea 17), y se lo emplea para construir un nuevo conjunto Q, que contendrá todas las reacciones que son posibles empleando los compuestos del conjunto anidado actualizado (línea 19). Una vez que el cromosoma completa el proceso de inicialización es incorporado en la población $\mathbb{P}$ (línea 20).

4.1.5. Operadores genéticos

4.1.5.1. Operador de mutación basado en conjunto de compuestos

El operador de mutación que se propone utiliza información del contexto para aplicar la modificación, al igual que el operador propuesto para AEBVML. Sin embargo, la información empleada por este nuevo operador proviene del conjunto anidado previo al gen a mutar. La aplicación del operador de mutación es controlada por la probabilidad de mutación p_m. Este operador borra o inserta un nuevo gen en el cromosoma con probabilidades p_e y $1 - p_e$, respectivamente. Cuando la mutación implica el borrado de un gen, éste simplemente es retirado de la secuencia. Si la mutación implica la inserción de un nuevo gen, la posición se elige en forma aleatoria en el intervalo $[1, N + 1]$. Cuando la posición seleccionada pertenece al intervalo $[1, N]$, el gen correspondiente es desplazado a la derecha junto con los restantes genes en la secuencia para dar lugar a la inserción. Una vez definida la posición se inserta una reacción que puede ser válida o inválida, con probabilidades p_v y $1 - p_v$, respectivamente, seleccionada teniendo en cuenta el conjunto anidado correspondiente a la posición que se seleccionó. Cuando la inserción es válida, los sustratos de la reacción deben estar disponibles en el conjunto anidado.

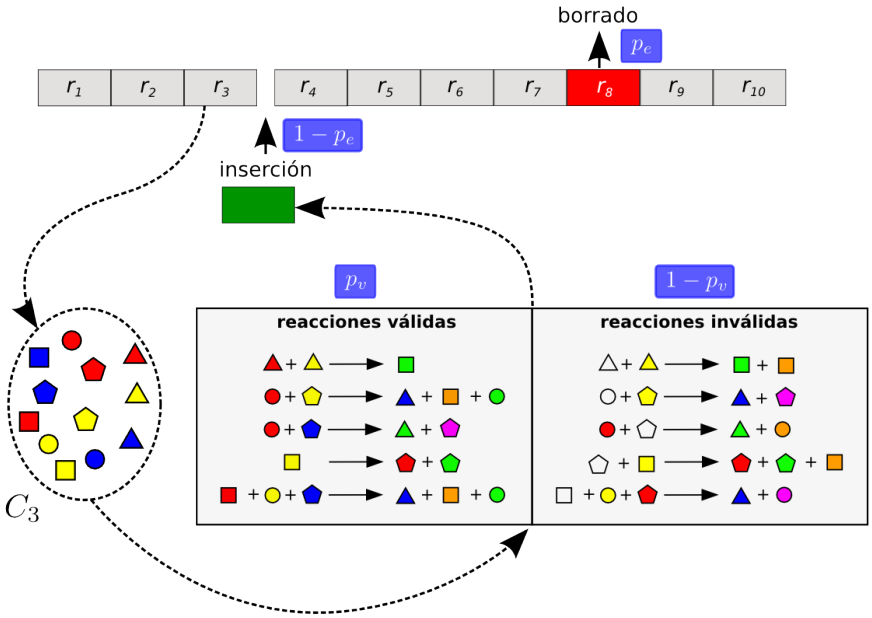


Figura 4.2: Funcionamiento del operador de mutación. *arriba:* cromosoma codificando una vía metabólica. El gen seleccionado para ser borrado de la secuencia se indica en rojo. Esta acción retira el gen y concatena r_7 con r_9. El gen que será insertado en la posición 4 del cromosoma (color verde) es seleccionado en forma aleatoria de uno de dos posibles conjuntos de reacciones (válidas e inválidas) construido considerando C_3.

El borrado de reacciones permite descartar conexiones redundantes en las vías o elementos que no contribuyen en la conexión. La inserción válida permite incorporar potenciales conexiones entre reacciones ya presentes en la vía, pudiendo en algunos casos completar la secuencia que conecta dos de los compuestos incluidos en D. La inserción de reacciones inválidas juega un papel importante al introducir diversidad en la composición de genes de la población. Por ejemplo, la reacción puede ser inválida en la posición donde se inserta, pero puede ubicarse correctamente luego de aplicar el operador de cruza. Además, cuando se recombina el material genético de dos individuos se modifica la composición de los conjuntos anidados de compuestos que éstos representan.

La Figura 4.2 esquematiza el funcionamiento del operador de mutación propuesto. En la parte superior se presenta un cromosoma conteniendo $N = 10$ reacciones. Los compuestos se representan como polígonos. Los po-

lígonos sin relleno indican compuestos que no se encuentran disponibles en el conjunto anidado C_3. Si suponemos que la mutación implica el borrado de una reacción, el proceso consiste en seleccionar uno de los diez posibles genes y retirarlo de la secuencia. En este ejemplo, el octavo gen (indicado en color rojo) es seleccionado para ser retirado del cromosoma. Si en cambio la mutación consiste en una inserción, primero se selecciona la posición donde se incorporará la reacción. En particular para este cromosoma se dispone de once posibles sitios para incorporar la reacción, que corresponden a las posiciones ocupadas por los genes y a la inserción de un gen al final del cromosoma. Suponiendo que se selecciona la cuarta posición del cromosoma (el nuevo gen será insertado entre las reacciones r_3 y r_4), la inserción se realizará teniendo en cuenta el conjunto anidado C_3. Como se muestra en la Figura, sólo las reacciones válidas tienen disponibles todos los sustratos en C_3. Los sustratos faltantes se indican con polígonos sin relleno. Dependiendo del tipo de inserción seleccionada se extrae en forma aleatoria una reacción del conjunto correspondiente.

4.1.5.2. Operador de cruza en dos puntos

Este operador combina el material genético de dos cromosomas $\mathbf{c}_i$ y $\mathbf{c}_j$ favoreciendo el descarte de reacciones que no contribuyen en las conexiones y que, generalmente, se ubican al final del cromosoma. El proceso de cruza se realiza en dos etapas. Primero se seleccionan en forma aleatoria los puntos de cruza que determinan las porciones de material genético que serán combinadas. Para esto se elige un punto de cruza $\pi_{i,1}$ en el cromosoma $\mathbf{c}_i$ y dos puntos $\pi_{j,1}, \pi_{j,2}$ en el cromosoma $\mathbf{c}_j$. Luego se combina el material genético de $\mathbf{c}_i$ contenido entre el inicio del cromosoma y el punto $\pi_{i,1}$ con la porción de $\mathbf{c}_i$ contenida entre los puntos de cruza $\pi_{j,1}, \pi_{j,2}$ seleccionados. El operador de cruza genera dos hijos, intercambiando los padres y realizando nuevamente el proceso una vez producido el primer hijo. La Figura 4.3 esquematiza el proceso de cruza.

La elección de dos puntos de cruza permite extraer porciones centrales del material genético, que es donde se concentra mayormente el conjunto de reacciones que definen las conexiones entre los compuestos incluidos en D. En general, ninguna de las reacciones que establecen la relación entre los compuestos en D se encuentra en la porción final del cromosoma. Esto hace que su eliminación reduzca el tamaño del cromosoma y mejore al mismo tiempo

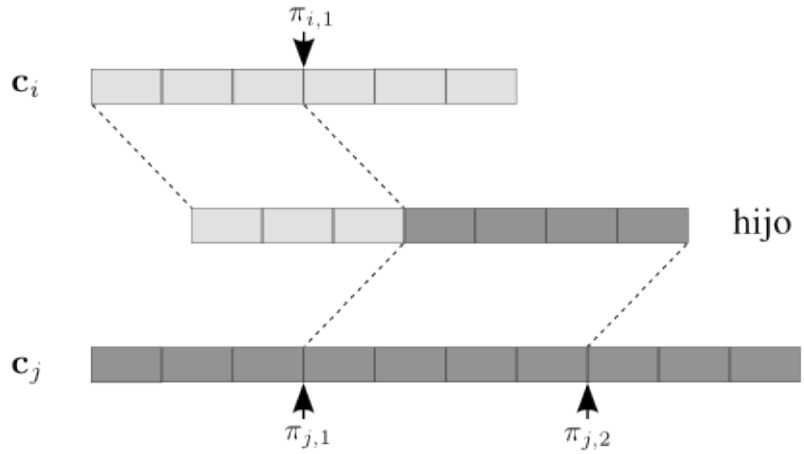


Figura 4.3: Esquema de funcionamiento del operador de cruza modificado. Los padres seleccionados para la cruza se ubican en la parte superior ($\mathbf{c}_i$) e inferior ($\mathbf{c}_j$) de la figura. El hijo se construye combinando el material genético de $\mathbf{c}_i$ hasta el punto de cruza $\pi_{i,1}$ y el material genético de $\mathbf{c}_j$ contenido entre los puntos de cruza $\pi_{j,1}$ y $\pi_{j,2}$.

las medidas relativas. Suponiendo que se desea relacionar tres compuestos, uno es utilizado al inicio del cromosoma y los dos restantes serán producidos en algún punto de la secuencia. Las reacciones posteriores a la que produce el último compuesto no contribuyen a establecer las relaciones deseadas, ya que no son empleadas para sintetizar algún compuesto de las secuencias que relacionan los tres compuestos. Como consecuencia, estas reacciones pueden ser descartadas ya que sólo incrementan el tamaño de cromosoma. La aplicación de este operador es controlada mediante la probabilidad de cruza p_x.

4.2. Datos, medidas de evaluación y visualización

4.2.1. Datos de reacciones

Las reacciones empleadas en los experimentos fueron extraídas de la base de datos LIGAND [Goto et al., 2002], que forma parte de KEGG y contiene información acerca de todos los compuestos metabólicos conocidos y de las reacciones que los vinculan. La información asociada a la reversibilidad de las reacciones se almacena en *mapas de referencia*, que agrupan conjuntos de reacciones asociadas a un mismo proceso y las representan como grafos dirigidos.

Tabla 4.1: Compuestos abundantes empleados en la búsqueda de vías metabólicas ramificadas. En la tabla se indica el nombre del compuesto y el código KEGG correspondiente.

Código KEGG	Nombre	Código KEGG	Nombre
C00001	H_2O	C00009	Fosfato
C00002	ATP	C00010	CoA
C00003	NAD^+	C00011	CO_2
C00004	NADH	C00014	Amoníaco
C00005	NADPH	C00020	AMP
C00006	$NADP^+$	C00028	Aceptor de Hidrógeno
C00007	O_2	C00030	Donador de Hidrógeno
C00008	ADP	C00080	H^+

Durante los experimentos se emplearon conjuntos de reacciones con diferente número de elementos. En todos los casos, la reversibilidad de las reacciones se definió de acuerdo a los mapas de referencia almacenados en KEGG. Cada una de las reacciones reversibles fue modelada como una pareja de reacciones independientes y de sentido opuesto. Por ejemplo, la reacción $S(r_k) \longleftrightarrow P(r_k)$ fue separada en las semireacciones $S(r_k) \rightarrow P(r_k)$ y $P(r_k) \rightarrow S(r_k)$. La primera cataliza la transformación directa de sustratos en productos, mientras que la segunda permite la regeneración de los sustratos a partir de los productos.

El conjunto de compuestos abundantes A empleado en los distintos experimentos se presenta en la Tabla 4.1. La identificación de los compuestos externos E se realizó en forma independiente para cada conjunto de reacciones.

4.2.2. Medidas para evaluación de características de las redes y desempeño del algoritmo

El estudio del desempeño del algoritmo se realizó empleando 30 corridas para cada combinación de parámetros. En los casos en los que la distribución de los resultados para las diferentes corridas poseían distribuciones asimétricas se empleó el valor de la mediana y su desvío para caracterizarlas. El análisis estadístico de los resultados se realizó empleando la prueba de Wilcoxon de rangos con signo [Derrac et al., 2011].

Las medidas de evaluación consideradas se dividen en dos grupos. El primero corresponde a medidas que evalúan el desempeño del algoritmo, e incluye el número de generaciones N_G empleado para encontrar una solución, el número de generaciones N_g necesario para que sólo una subpoblación sobreviva y la proporción de las corridas F_S que encuentran una vía metabólica que relaciona a todos los compuestos en D.

El segundo grupo corresponde a medidas que evalúan características de las vías metabólicas encontradas. Las medidas empleadas para caracterizar las vías metabólicas son:

- Reacciones (N_R): cuenta el número de reacciones en la vía metabólica.
- Ramificación (ρ): evalúa el nivel medio de dependencia entre reacciones. Esta medida se calcula como el número promedio de reacciones en las que un compuesto es empleado. Los compuestos incluidos en el conjunto A' (compuestos abundantes) no son considerados para el cálculo de la medida.
- Hojas (λ): esta medida evalúa el número de compuestos producidos por la vía metabólica que no son empleados como sustrato por ninguna reacción. Sean $S^*(\mathbf{c}_i)$ y $P^*(\mathbf{c}_i)$ los conjuntos de los sustratos y productos de todas las reacciones codificadas en $\mathbf{c}_i$, respectivamente. El número de hojas λ se calcula como

$$\lambda(\mathbf{c}_i) = |P^*(\mathbf{c}_i) - \left(S^*(\mathbf{c}_i) \cup A'\right)|. \tag{4.6}$$

- Diferencia entre vías metabólicas (σ): esta medida evalúa la proporción de los compuestos compartidos entre dos vías. Sean $Q_i = (P^*(\mathbf{c}_i) \cap S^*(\mathbf{c}_i) \cup D) - A'$ y $Q_j = (P^*(\mathbf{c}_j) \cap S^*(\mathbf{c}_j) \cup D) - A'$ los subconjuntos de compuestos pertenecientes a las vías codificadas en $\mathbf{c}_i$ y $\mathbf{c}_j$ respectivamente. La diferencia entre las vías metabólicas se calcula como

$$\sigma(\mathbf{c}_i, \mathbf{c}_j) = 1 - \left[\frac{|Q_i \cap Q_j|}{\text{mín}(|Q_i|, |Q_j|)}\right]. \tag{4.7}$$

  Dos vías presentarán diferencia cero cuando empleen la misma secuencia de compuestos para relacionar los compuestos de interés.

4.2.3. Visualización de redes metabólicas

Un aspecto muy importante en el estudio de las redes metabólicas es la visualización de las conexiones entre reacciones y compuestos. Una adecuada representación gráfica de la vía facilita la interpretación de las conexiones, agilizando la extracción de conclusiones. Las representaciones basadas en grafos de compuestos o de reacciones producen representaciones simplificadas de la red, en las que se pierde información de las relaciones entre reacciones y compuestos. Las representaciones basadas en grafos bipartitos (emplean dos tipos de nodos y las relaciones se establecen siempre entre nodos de distinto tipo) [Arita, 2012] proveen una alternativa interesante que permite detectar los sustratos requeridos para cada reacción. Sin embargo, cuando las redes contienen un elevado número de compuestos y reacciones se vuelven muy difíciles de interpretar.

Una forma de facilitar la visualización consiste en representar sólo los elementos esenciales que contribuyen a establecer las relaciones entre reacciones. Además, la codificación de la información empleando diferentes símbolos y colores permite identificar rápidamente aspectos relevantes de la red. Para ilustrar el modo en que se realiza la visualización, consideremos por ejemplo las reacciones

R01512: C00002 + C00197 $\longrightarrow$ C00008 + <u>C00236</u>
R01061: C00004 + C00080 + <u>C00236</u> $\longrightarrow$ C00003 + C00009 + C00118

Ambas reacciones se encuentran acopladas, ya que uno de los productos de la primera reacción (subrayado) es empleado como sustrato de la segunda. Supongamos que la secuencia de reacciones se emplea para producir C00118 a partir de C00197. La Figura 4.4 representa la secuencia resultante empleando símbolos y colores para codificar la información. El gráfico se construye recorriendo el cromosoma de izquierda a derecha, incorporando las reacciones en el gráfico de forma descendiente y partiendo del compuesto inicial especificado. Para este ejemplo, la primera reacción almacenada en el cromosoma correspondería a R01512 y la segunda a R01061.

En esta representación, las conexiones sustrato–reacción sólo se establecen si el compuesto ha sido sintetizado previamente. Cada reacción es representada mediante un cuadrado azul o rojo, dependiendo de que se trate de reacciones válidas o inválidas, respectivamente. Los compuestos pertene-

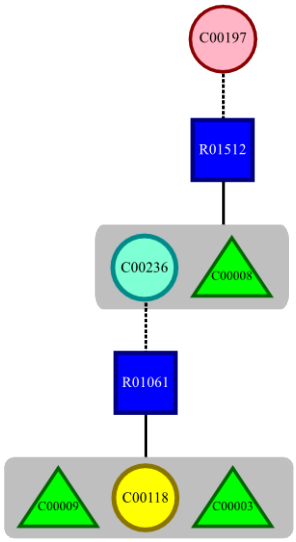


Figura 4.4: Ejemplo de visualización de una vía metabólica. En rojo se indica el compuesto inicial; en amarillo el compuesto a relacionar; en verde los compuestos abundantes y en celeste los productos intermedios.

cientes al conjunto de los compuestos abundantes se indican con triángulos verdes, mientras que los compuestos restantes se emplean círculos celestes. El compuesto inicial d_I (C00197) y los compuestos a relacionar $D - \{d_I\}$ (sólo C00118 en este ejemplo) son una excepción, ya que se indican en rojo y amarillo, respectivamente. Los productos de cada reacción se grafican agrupados en un rectángulo de color gris. Debe notarse que los sustratos de cada reacción que forman parte de los compuestos abundantes que se estén empleando no son graficados ya que al ser de libre disponibilidad, no revisten un carácter esencial en la reacción. Esto permite obtener una vista más limpia de la vía metabólica. Sólo los compuestos abundantes producidos en la red son incluidos en el gráfico. Los sustratos de cada reacción (que no pertenecen a los compuestos abundantes) se indican con líneas de trazo que unen el compuesto con la reacción correspondiente. Los productos se indican mediante una línea continua que une la reacción con el rectángulo gris que agrupa los compuestos. En caso de que más de una reacción produzca el sustrato requerido por otra, cada instancia del compuesto es vinculada a la reacción correspondiente mediante una línea de puntos.

4.3. Resultados y discusión

El estudio del desempeño del algoritmo propuesto en este capítulo se realiza en tres etapas. La primera, presentada en la Sección 4.3.1, estudia el comportamiento del algoritmo frente a la variación de distintos parámetros y al uso de diferentes operadores. En la Sección 4.3.2 se analiza el desempeño del algoritmo al extender el conjunto de reacciones empleado previamente. Finalmente, en la Sección 4.3.3 se presentan dos casos de estudio donde se analizan y discuten algunos aspectos biológicos de las soluciones encontradas.

En todos los experimentos se empleó una mínima aptitud tolerada de $\mathcal{A}^{\epsilon} = 1$. Además se utilizó una estrategia de inicialización de los cromosomas con tamaño variable en el intervalo $[\frac{1}{2}N_M, N_M]$. Estos límites fueron elegidos para lograr un muestreo adecuado del espacio de soluciones.

4.3.1. Parámetros característicos

En esta sección se presentan los resultados obtenidos para diferentes evaluaciones realizadas sobre AEBCAC. Primero se analiza el efecto que ejerce el operador de selección sobre el proceso de búsqueda. Después se estudia la influencia de las probabilidades que controlan el comportamiento del operador de mutación sobre las características de las soluciones encontradas. Finalmente se analiza la sensibilidad del algoritmo al número inicial de genes en el cromosoma.

El espacio de búsqueda empleado está definido por el conjunto de reacciones pertenecientes a la vía metabólica de la arginina y prolina (*apdata*)[1], la cual corresponde a una vía de referencia de KEGG. Habiendo considerado la información de reversibilidad de la vía, se extrajeron 139 reacciones, de las cuales 24 son reversibles (descompuestas en 48 reacciones) y 91 son irreversibles. Los experimentos consistieron en la búsqueda de vías metabólicas que relacionen los compuestos L-Glutamato (C00025), Fumarato (C00122) y L-Prolina (C00763).

4.3.1.1. Influencia del operador de selección

Experimentos preliminares realizados con AEBCAC requirieron más de 400 generaciones para encontrar una solución al emplear el operador clásico

[1] `http://www.genome.jp/kegg-bin/show_pathway?map00330`

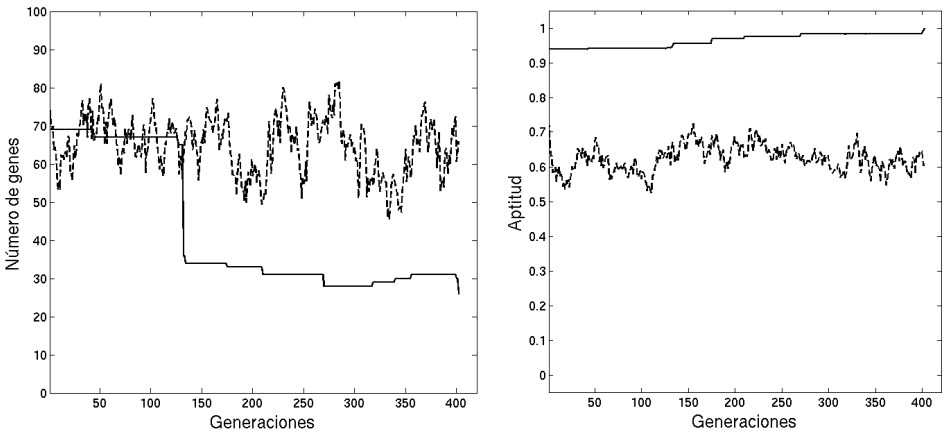


Figura 4.5: Variación del número de genes y la aptitud a lo largo de las generaciones empleando selección por ruleta. *Izquierda:* número de genes. *Derecha:* aptitud. Los valores correspondientes al mejor individuo y al valor medio de la población se indican en líneas continua y de trazo, respectivamente.

de cruza en un punto y la selección por ruleta. En estas pruebas se utilizó una población con $M = 200$ individuos. Las probabilidades de mutación y cruza se fijaron $p_m = 0.05$ y $p_x = 0.80$ (esta probabilidad de cruza produjo los mejores resultados con AEBVML). Por su parte, las probabilidades de borrado e inserción válida empleadas se fijaron en $p_e = 0.50$ y $p_v = 0.50$, respectivamente. En todas las corridas se observaron oscilaciones importantes en el número medio de genes de la población, y una aptitud media sin mejora sustancial durante la evolución. La Figura 4.5 presenta un ejemplo de este comportamiento. Es probable que las oscilaciones estén causadas por la alta presión de selección que ejerce la ruleta, ya que el tamaño de los mejores individuos entre generaciones sucesivas varía sustancialmente, produciendo cambios importantes en el número medio de genes en la población. Otro aspecto interesante es que las oscilaciones se producen en torno a 70 genes, sin una tendencia a disminuir. Esto resulta llamativo ya que lo esperable sería que las reacciones innecesarias tiendan a ser eliminadas durante la evolución y este efecto no se observa en la figura.

En base a los resultados obtenidos con la ruleta, se evaluó el desempeño del algoritmo comparando los operadores de selección por ruleta y por el método de torneo, realizando 30 corridas con cada operador de selección, y considerando $k = 5$ individuos en cada torneo. La Figura 4.6 presenta un ejemplo para la evolución del número de genes y de la aptitud en una corrida

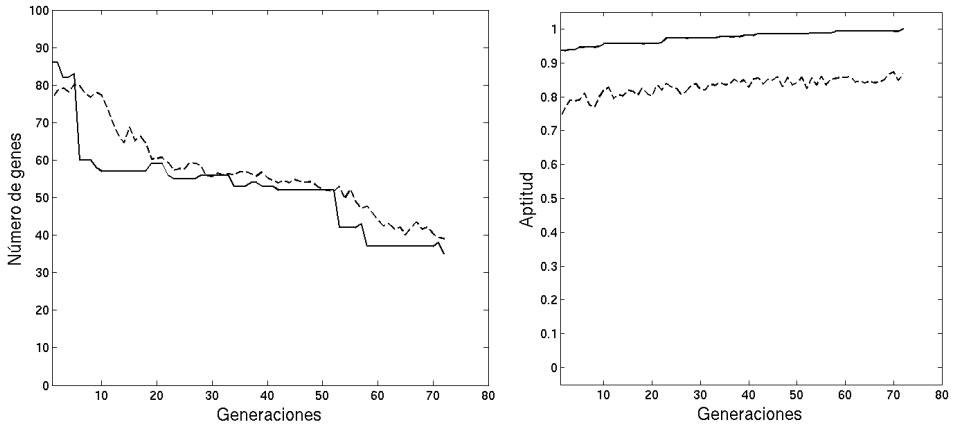


Figura 4.6: Variación del número de genes y la aptitud a los largo de las generaciones empleando selección por torneo. *Izquierda:* número de genes. *Derecha:* aptitud. Los valores correspondientes al mejor individuo y al valor medio de la población se indican en líneas continua y de trazo, respectivamente.

Tabla 4.2: Efecto del operador de selección sobre el desempeño del algoritmo evolutivo.

	Ruleta	Torneo
N_S	13	25
N_G	775±99	59±27
N_g	16±5	3±0
N_R	9±2	9±2
ρ	1.2±0.1	1.2±0.1
λ	6±1	6±1

empleando selección por torneo. Puede apreciarse que las oscilaciones en el número medio de genes de la población desaparecen (línea de trazos en la figura de la izquierda), y que ahora la población evoluciona para producir individuos con tamaños que se aproximan al del mejor individuo. También se observa un incremento en la aptitud media de la población durante la evolución (no observado en la Figura 4.5), indicando que la población completa evoluciona hacia una solución del problema.

La Tabla 4.2 presenta los resultados de la comparación de los operadores de selección. Notablemente, el número de soluciones encontradas utilizando selección por torneo es prácticamente el doble que las encontradas con el otro método de selección. Además, la reducción en la presión de selección reduce

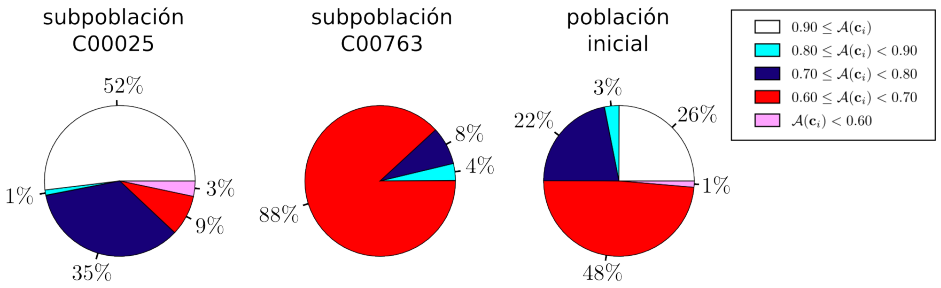


Figura 4.7: Composición de cada subpoblación, en términos de aptitud de los individuos, en la primera generación. *Izquierda:* subpoblación inicializada con el compuesto C00025. *Centro:* subpoblación inicializada con el compuesto C00763. *Derecha:* composición de la población conjunta.

en un orden de magnitud el número de generaciones requeridas para encontrar una solución. Es interesante destacar que el número de generaciones necesarios para que sólo una población sobreviva es marcadamente menor empleando torneo. Este resultado puede deberse a la presencia de una muy elevada proporción de individuos con baja aptitud en la población inicial. Finalmente, los valores de N_R, ρ y λ son iguales para ambos operadores debido a que ninguna de las estrategias de selección incorporan mecanismos que influyan directamente sobre las características de las vías metabólicas buscadas. Puede verse entonces que el operador de selección por torneo favorece la reducción en el número de generaciones N_g requeridos para encontrar una vía metabólica que relacione los tres compuestos.

Para analizar en mayor profundidad la causa de la marcada diferencia en los valores de N_g, se analizó la composición una población inicial en términos de aptitud. Los resultados se presentan en la Figura 4.7, la cual está compuesta por tres gráficos que corresponden a la composición, en términos de aptitud, de las subpoblaciones inicializadas con d_I = C00025 y d_I = C00763, y a la población total. Claramente la subpoblación que emplea C00025 está compuesta en su mayoría por individuos que poseen valores de aptitud superiores a 0.7. En cambio, la subpoblación que emplea C00763 presenta mayormente individuos con aptitudes en el intervalo [0.6,0.7]. Como resultado, la mitad de la población inicial contiene individuos con aptitud superior a 0.7 (inicializados con C00025) y la otra mitad contiene individuos con aptitud inferior a 0.7 (inicializados con C00763). Cuando se emplea ruleta, una

Tabla 4.3: Efecto del operador de cruza modificado sobre el desempeño del algoritmo evolutivo.

	Cruza clásica	Cruza modificada
N_S	25	29
N_G	59±27	57±18
N_g	3±0	3±0
N_R	9±2	8±1
ρ	1.2±0.1	1.2±0.1
λ	6±1	6±1

gran proporción de la misma está ocupada por individuos inicializados con C00763, a pesar de que su aptitud sea menor a 0.7. Cuando se inicia la evolución, esta fracción de individuos comienza a reducirse hasta desaparecer, aunque el proceso requiere de varias generaciones. En cambio, al emplear torneo la reducción en el número de individuos que emplean C00763 es mucho más acelerada ya que en una mayor proporción de casos ganan los individuos inicializados con C00025.

4.3.1.2. Influencia del operador de cruza

Las vías metabólicas son modeladas de izquierda a derecha en los cromosomas, por lo que las reacciones que se ubican hacia el final del cromosoma son más sensibles a las modificaciones que se realicen sobre la vía codificada. Si una reacción al inicio de la vía es modificada, probablemente las últimas reacciones de la secuencia dejen de ser válidas. Teniendo esto presente puede entenderse que el operador de cruza genera descendencia con una gran proporción de reacciones en la parte final del cromosoma que no contribuyen a la vía metabólica. Por lo tanto se propuso modificar el operador de cruza para seleccionar sólo una porción del segundo padre, buscando limitar el número de reacciones que se incorporan al cromosoma.

La Tabla 4.3 presenta los resultados alcanzados empleando el operador de cruza en un punto y el nuevo operador de cruza propuesto. Lo primero que se observa es que el número de corridas que conducen a una solución se incrementa al emplear el operador modificado. Este resultado indica que la elección de dos puntos de cruza en el segundo padre disminuye el número de reacciones que deben ser removidas, permitiendo que la vía metabólica sea encontrada en el número máximo de generaciones establecido. No se ob-

Tabla 4.4: Generaciones requeridas por AEBCAC para encontrar una vía metabólica empleando la inicialización con tamaño de cromosoma variable. Los resultados corresponden a los valores de mediana y sus desvíos. † indica experimentos donde se encuentra alguna solución en más del 90 % de las corridas. En negrita se resaltan los mejores resultados obtenidos con cada probabilidad de mutación.

N_G	$p_m = 0.02$			$p_m = 0.05$			$p_m = 0.08$		
	$p_e = 0.20$	$p_e = 0.50$	$p_e = 0.80$	$p_e = 0.20$	$p_e = 0.50$	$p_e = 0.80$	$p_e = 0.20$	$p_e = 0.50$	$p_e = 0.80$
$p_v = 0.20$	$87 \pm 37^{\dagger}$	72±25	41±10	164±65	$60 \pm 19^{\dagger}$	$36 \pm 9^{\dagger}$	256±130	69±18	$39 \pm 13^{\dagger}$
$p_v = 0.50$	57±18	$56 \pm 13^{\dagger}$	**40±11**	102±42	$49 \pm 10^{\dagger}$	**35±11**	159±80	$70 \pm 29^{\dagger}$	**33±8**†
$p_v = 0.80$	$72 \pm 35^{\dagger}$	$45 \pm 6^{\dagger}$	41±13	$97 \pm 48^{\dagger}$	47±17	$37 \pm 12^{\dagger}$	$162 \pm 95^{\dagger}$	$62 \pm 28^{\dagger}$	$36 \pm 8^{\dagger}$

servan diferencias significativas en el número de generaciones para encontrar una solución y para que sobreviva una subpoblación. Tampoco se aprecian diferencias en las medidas relacionadas con las características de las vías metabólicas, ya que ambos operadores no contemplan mecanismos que favorezcan alguna diferencia. Sólo se aprecia una tendencia a reducir el tamaño de las soluciones cuando se utiliza el operador modificado, posiblemente debido a la capacidad de limitar el número de reacciones adicionales incorporadas en la red.

4.3.1.3. Efecto de la variación de las probabilidades que controlan la mutación

El operador de mutación propuesto juega un papel muy importante en la modificación de las vías metabólicas mediante la inserción o borrado de reacciones. La incorporación de nuevas reacciones en el cromosoma puede conducir a la producción de nuevos compuestos para que reacciones posteriores puedan realizarse. El borrado de reacciones permite la eliminación de reacciones que pueden ser redundantes o inválidas. Un apropiado balance entre estas operaciones debería conducir a búsquedas que requieran pocas generaciones para encontrar la solución. El estudio del efecto de las diferentes probabilidades sobre el comportamiento del operador de mutación se realizó utilizando las condiciones experimentales empleadas previamente y los valores $p_m = \{0.02, 0.05, 0.08\}$, $p_e = \{0.20, 0.50, 0.80\}$ y $p_v = \{0.20, 0.50, 0.80\}$ para cada una de las probabilidades.

La Tabla 4.4 presenta el número de generaciones empleadas para cada combinación de las tres probabilidades estudiadas. Sobre la tabla se indican

con una marca las combinaciones de probabilidades para las que se obtuvieron soluciones en más del 90 % de las corridas. La probabilidad de borrado muestra un marcado efecto sobre el número de generaciones. Independientemente de la probabilidad de mutación empleada, probabilidades de borrado más elevadas conducen a una disminución en el número de generaciones. Este resultado es consistente con el proceso de búsqueda ya que durante la inicialización se incorpora un elevado número de reacciones que luego deben ser descartadas durante la evolución. Por consiguiente, resulta esperable que la aplicación de mutaciones que favorecen la eliminación de reacciones mejore el desempeño del algoritmo.

También puede observarse un efecto de interacción entre las probabilidades de mutación y borrado, evidenciado principalmente cuando se emplea una baja probabilidad de borrado. Cuando la probabilidad de mutación es $p_m = 0.08$, el uso de una probabilidad de borrado $p_e = 0.20$ favorece la inserción de reacciones y el crecimiento de la red metabólica, haciendo que se requieran más generaciones para encontrar una solución. A medida que se reduce la probabilidad de mutación, el efecto del operador se diluye. Por lo tanto, aunque el uso de $p_m = 0.02$ y $p_e = 0.20$ hace que se requieran más generaciones que para probabilidades de borrado más elevadas, estos valores son menores que los obtenidos con probabilidades de mutación más elevadas. Los resultados presentados en la tabla no evidencian alguna interacción entre la probabilidad de inserción válida y las demás probabilidades, probablemente debido a que la mayor contribución del operador de mutación es el borrado.

Resulta claro que el uso de $p_e = 0.80$ y $p_v = 0.50$ conduce a búsquedas que requieren el menor número de generaciones para todas las probabilidades de mutación. Sin embargo, cuando se emplea $p_m = 0.08$ se requiere el menor número de generaciones, por lo que esta combinación de parámetros se empleará en los experimentos posteriores. No se observaron diferencias en el valor de N_g para todas las combinaciones estudiadas, probablemente debido a que la elección de la subpoblación ganadora se realiza en la tercera generación, prácticamente sin que el operador de mutación hubiera podido ejercer alguna influencia.

Tabla 4.5: Generaciones requeridas por AEBCAC para encontrar una vía metabólica empleando diferentes tamaños iniciales de cromosoma. Los valores resaltados corresponden a experimentos donde una solución es encontrada en más del 90 % de las corridas. Los resultados corresponden a los valores de mediana y sus desvíos.

	$N_I = 20$	$N_I = 30$	$N_I = 40$
N_G	230±201	**23±5**†	**24±5**†

4.3.1.4. Sensibilidad al tamaño inicial de los cromosomas

Los experimentos realizados previamente fueron llevados a cabo empleando una estrategia de selección aleatoria del tamaño inicial para los cromosomas. Aunque esta estrategia provee una solución al problema de la elección del tamaño inicial adecuado, es importante conocer si los límites entre los que se inicializan estos tamaños son apropiados. Para evaluar el efecto que produce el tamaño inicial del cromosoma, se realizaron una serie de experiencias variando el número inicial de genes. Los cromosomas se inicializaron con $N_I = \{20, 30, 40\}$ genes dado que son suficientes aproximadamente 8 reacciones para establecer las conexiones entre los tres compuestos. Las probabilidades de mutación, borrado e inserción válida se fijaron en $p_m = 0.08$, $p_e = 0.80$ y $p_v = 0.50$, respectivamente.

Los resultados correspondientes al número de generaciones se presentan en la Tabla 4.5. Sobre la tabla se indican con una marca las combinaciones de probabilidades para las que se obtuvieron soluciones en más del 90 % de las corridas. Es evidente que inicializar los cromosomas empleando N_I =30 genes o más conduce a búsquedas que requieren un reducido número de generaciones para encontrar una solución. Estos resultados además concuerdan con los valores de N_I necesarios para encontrar alguna solución en más del 90 % de las corridas. Puede verse claramente la importancia de la elección del número adecuado de genes en la inicialización, ya que determinan la diversidad inicial de reacciones con la que se realizará la búsqueda. Si el número de genes es demasiado pequeño como para realizar un correcto muestreo inicial del conjunto de reacciones (N_I=20 para este problema), se requerirá un elevado número de generaciones para encontrar la solución. La inicialización con tamaño variable reduce estos problemas proporcionando diferente número de reacciones y asegurando que algunos individuos poseerán más genes

Tabla 4.6: Comparación del desempeño del algoritmo empleando *apdata* y *sdata*.

	apdata	*sdata*
F_S	1.00	0.97
N_G	33±8	29±7
N_g	3±0	4±1
N_R	8±1	6±1
ρ	1.2±0.1	1.3±0.1
λ	5±1	4±1

que el número mínimo requerido.

4.3.2. Comportamiento en un espacio de búsqueda extendido

En la subsección anterior se estudió el desempeño del algoritmo empleando el conjunto de reacciones *apdata*. Si bien en la mayoría de las ejecuciones se encuentra una vía metabólica que relaciona los tres compuestos especificados, el estudio se realiza sobre un espacio de búsqueda con un reducido número de reacciones. Aunque este conjunto de reacciones permite entender el efecto de algunos parámetros sobre el comportamiento del algoritmo, no se puede garantizar que este comportamiento sea el mismo al utilizar un conjunto de reacciones más amplio. Por tal motivo se procedió a estudiar la escalabilidad del AEBCAC, expandiendo el conjunto de reacciones *apdata* y evaluando el desempeño del algoritmo durante la capacidad de búsqueda en este nuevo espacio de soluciones.

El nuevo conjunto de datos, denominado *sdata*, se construyó adicionando a *apdata* las reacciones pertenecientes a cuatro vías metabólicas de referencia. El nuevo conjunto de reacciones quedó conformado por 443 reacciones con sentido único, donde 132 reacciones son reversibles (descompuestas en 264 reacciones) y 179 reacciones son irreversibles.

Las medidas de evaluación para los resultados encontrados empleando *apdata* y *sdata* se presentan, de forma comparativa, en la Tabla 4.6. Puede apreciarse que el desempeño del algoritmo no varía sustancialmente al incrementar el espacio de búsqueda, ya que la fracción de las corridas que producen alguna solución es similar para ambos algoritmos. Aunque el nú-

mero de generaciones es menor al emplear *sdata*, la reducción de este valor no es significativa respecto de los resultados obtenidos con *apdata*. Si bien el valor de N_g se encuentra incrementado en una generación, esta diferencia no impacta en el resultado desde un punto de vista práctico debido a que la elección de la subpoblación ganadora continúa definiéndose prácticamente al inicio de la evolución.

Cuando se analizan las medidas asociadas a la estructura de las vías metabólicas debe tenerse en cuenta la característica que cada medida evalúa. Mientras que la ramificación aporta información acerca del grado de conexión que existe entre las reacciones midiendo el número promedio de éstas que emplean alguno de los compuestos no abundantes sintetizados por la red, el número de hojas brinda una idea del grado de especificidad que tiene la red encontrada. Una vía con elevado número de hojas indica que participa como intermediario de una gran variedad de procesos; una vía con un reducido número de hojas indica una alta especificidad para la síntesis de los compuestos indicados. Lo primero que se observa es una disminución significativa ($p < 0.0001$) en el tamaño de las vías encontradas con *sdata* respecto de las que fueron encontradas empleando *apdata*. Esto resulta esperable debido a que el número de posibles conexiones entre compuestos es mayor y posibilita la existencia de caminos más cortos para conectar los compuestos. El valor de ramificación (ρ) obtenido para *sdata* confirma esto, ya que experimenta un incremento significativo ($p < 0.005$), pasando de 1.2 a 1.3 al emplear el conjunto de reacciones extendido. El número de hojas también experimenta una reducción significativa ($p < 0.0001$), indicando que las vías encontradas al emplear *sdata* incluyen reacciones que generan un menor número de productos innecesarios. Esto posiblemente pueda deberse a que se emplean reacciones más específicas en el proceso. Es importante destacar que independientemente de las diferencias encontradas para la ramificación, ambos conjuntos de reacciones conducen a soluciones cuya ramificación es superior a la unidad. Esto indica que algunos compuestos de la red actúan como sustrato en más de una reacción. Este resultado demuestra que el algoritmo es capaz de establecer conexiones más complejas que en una vía metabólica lineal (cuyo valor de ramificación sería 1).

Dado que *sdata* es una ampliación de *apdata* resulta esperable que el algoritmo sea capaz de encontrar las soluciones halladas previamente. Este análisis se llevó a cabo calculando el grado de diferencia entre las soluciones

Tabla 4.7: Valores de la diferencia entre las familias de soluciones encontradas con *apdata* y *sdata*. En negrita se destacan los valores de diferencia menores a 0.15.

		sdata						
		Ib	IIb	IIIb	IVb	Vb	VIb	VIIb
apdata	Ia	0.50	0.43	0.38	0.29	0.20	**0.13**	**0.00**
	IIa	0.50	0.43	0.25	0.29	0.20	0.22	**0.00**
	IIIa	0.38	0.43	0.38	0.29	0.20	0.22	**0.00**
	IVa	0.25	0.29	0.38	0.29	**0.00**	0.22	**0.00**
	Va	0.25	0.29	0.38	0.29	**0.00**	**0.11**	**0.00**

encontradas con ambos conjuntos de reacciones y utilizando un conjunto unificado de compuestos abundantes, construido por la unión de los compuestos abundantes empleados en las búsquedas realizadas con *apdata* y *sdata*. Esto es necesario debido a que el conjunto de los compuestos externos E en ambos casos es diferente, llevando a que los compuestos abundantes sean también distintos.

La medida de diferencia (σ) empleada compara la secuencia de compuestos utilizada para vincular los compuestos especificados. Esto permite que diferentes secuencias de reacciones puedan generar secuencias de compuestos idénticas (mediante reacciones alternativas) y no presenten diferencias para esta medida. En este contexto se define *familia de soluciones* como el conjunto de vías metabólicas que han sido encontradas empleando el mismo conjunto de reacciones y que no presentan diferencias entre ellas a nivel de la secuencia de compuestos que vinculan a los miembros del conjunto D. Los valores de diferencia indicaron la presencia de cinco *familias de soluciones* para *apdata* y siete para *sdata*. La Tabla 4.7 presenta los valores de diferencia entre las familias de ambos conjuntos de reacciones. Los compuestos C00049 y C00091 no son producidos por *apdata*, por lo que son considerados dentro del conjunto unificado de compuestos abundantes para el cálculo de las diferencias.

El análisis de la tabla debe realizarse considerando que la intersección entre una fila y una columna indica el valor de diferencia entre las familias consideradas. Resulta esperable que el algoritmo sea capaz de encontrar las

mismas soluciones halladas previamente, además de otras adicionales debido al mayor número de conexiones entre compuestos. Se puede observar que la familia `VIIb` no presenta diferencias con ninguna de las soluciones encontradas con *apdata*. Esto se debe a que la secuencia de compuestos empleada por `VIIb` para establecer los vínculos es también empleada por las cinco familias de *apdata*, junto a otros compuestos adicionales. La familia `IVb` también muestra un comportamiento similar, aunque en este caso sólo comparte una porción de la secuencia de compuestos que establecen los vínculos con todas las soluciones de *apdata*, ya que presenta una diferencia de 0.29. Es interesante notar que la familia `Vb` comparte su secuencia de compuestos en forma completa con las familias `IVa` y `Va`, y sólo en forma parcial con las restantes familias. Esta marcada diferencia que se establece entre entre `Vb` y las familias `IVa`, `Va` y las tres familias restantes también se observa para la familia `IIb`. Sin embargo, en ningún caso la diferencia entre esas familias es nula, lo que está indicando que al menos uno de los compuestos de la secuencia que relaciona a los miembros de D no es compartido por las familias.

Para profundizar el análisis de las diferencias encontradas, se seleccionaron dos familias de cada conjunto de datos y se visualizaron las vías metabólicas representadas. Las Figuras 4.8, 4.9, 4.10 y 4.11 presentan las vías metabólicas que corresponden a las familias `IVa`, `Vb`, `Ia` y `Ib`, respectivamente. En los cuatro casos las vías deben ser analizadas en forma descendiente, partiendo del compuesto C00025. Para facilitar la explicación del cálculo de las diferencias se dibuja junto a cada vía metabólica el grafo metabólico correspondiente, conteniendo sólo los compuestos no abundantes. Las flechas indican la dependencia entre compuestos, mientras que los colores indican los mismos tipos de compuestos que en la vía metabólica. Sobre cada grafo se destacan las secuencias de compuestos que establece la conexión entre el compuesto inicial d_I y cada uno de los compuestos a relacionar incluidos en $D - \{d_I\}$ dibujando conjuntos que los engloban.

Lo primero que debe destacarse en estas cuatro figuras es la ramificación de las vías metabólicas, ya que algunos compuestos son empleados por más de una reacción para que la vía metabólica sea posible. El compuesto C00025 cumple esta función en tres de las cuatro vías, actuando como precursor de dos secuencias de reacciones que sintetizan los compuestos deseados. Los compuestos C00077 (`Ia`) y C00148 (`IVa`) también desempeñan un papel similar.

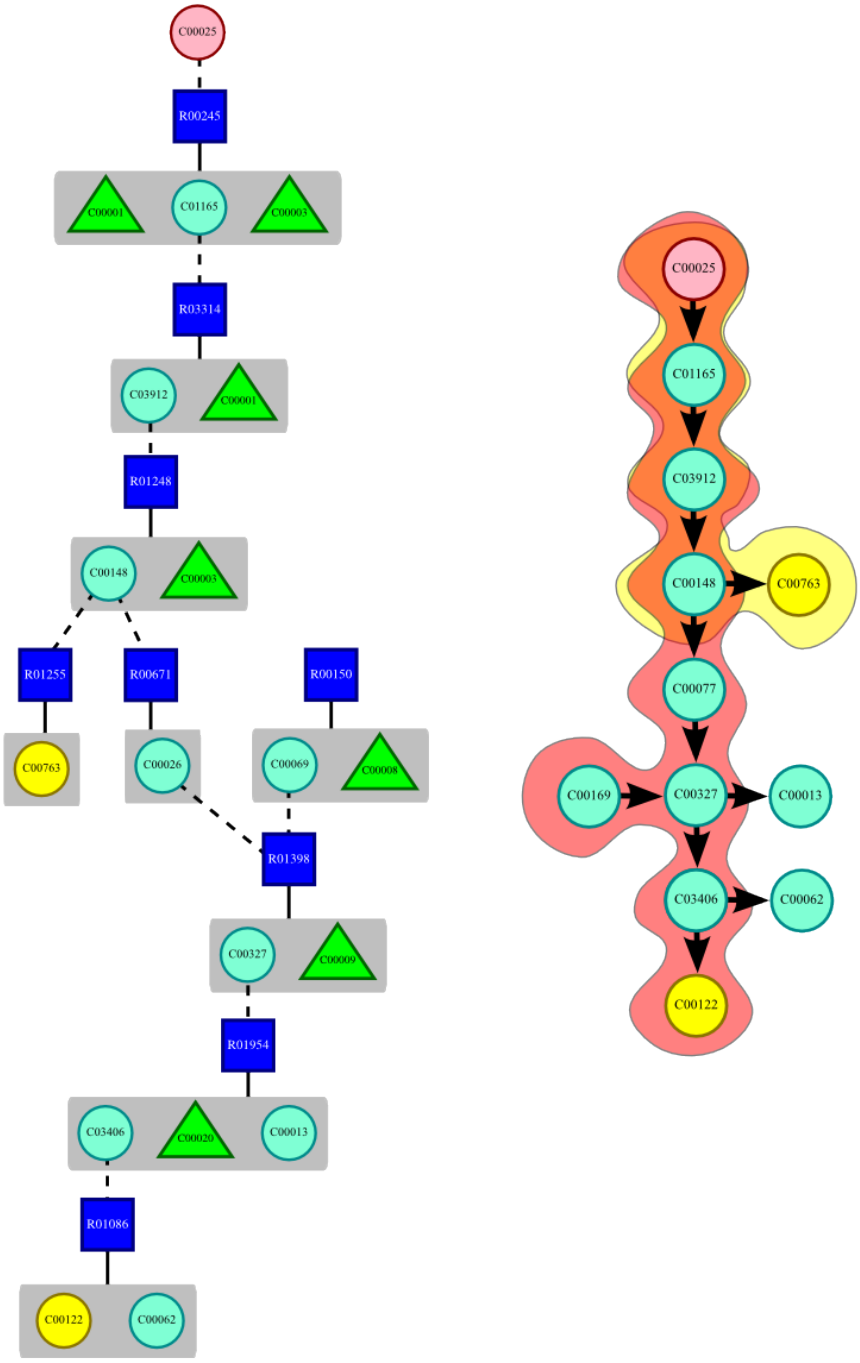


Figura 4.8: Familias de soluciones `IVa`. Junto a cada vía metabólica se muestra el grafo metabólico asociado. Las regiones sombreadas sobre los grafos metabólicos indican los compuestos que participan en la secuencia que relaciona: C00025–C00122 (anaranjado) y C00025–C00763 (amarillo).

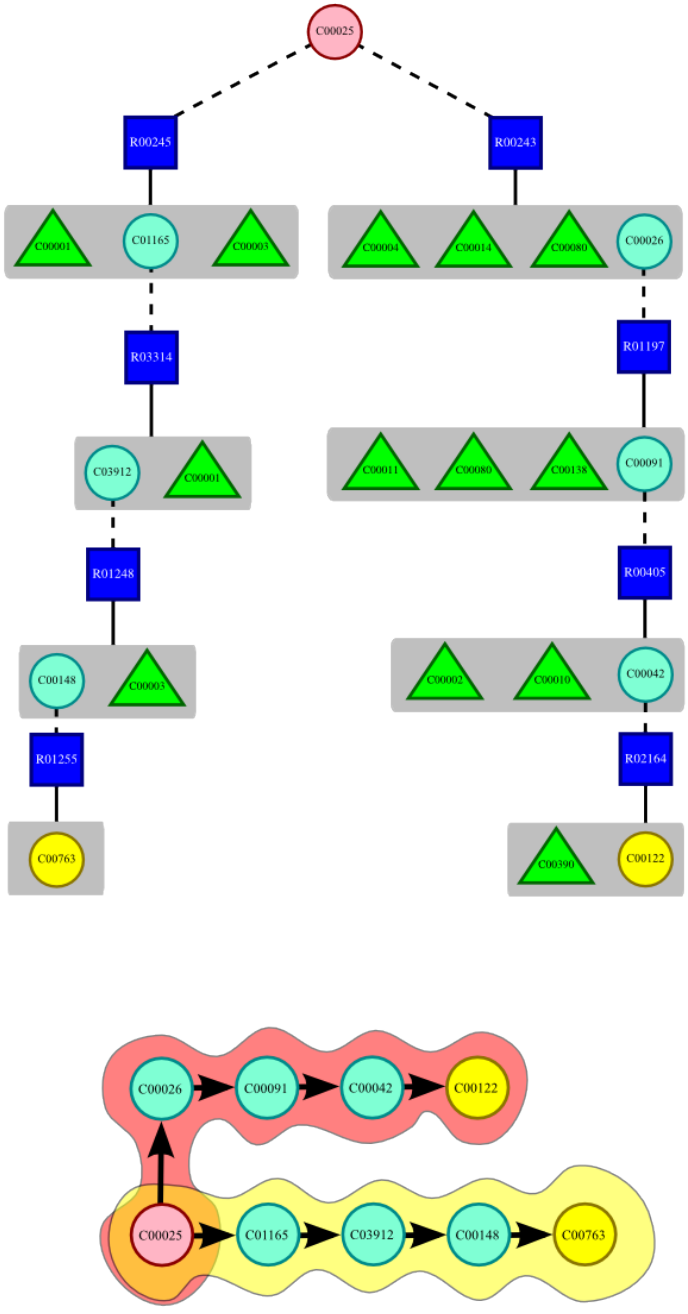


Figura 4.9: Familias de soluciones `Ib`. Junto a cada vía metabólica se muestra el grafo metabólico asociado. Las regiones sombreadas sobre los grafos metabólicos indican los compuestos que participan en la secuencia que relaciona: C00025–C00122 (anaranjado) y C00025–C00763 (amarillo).

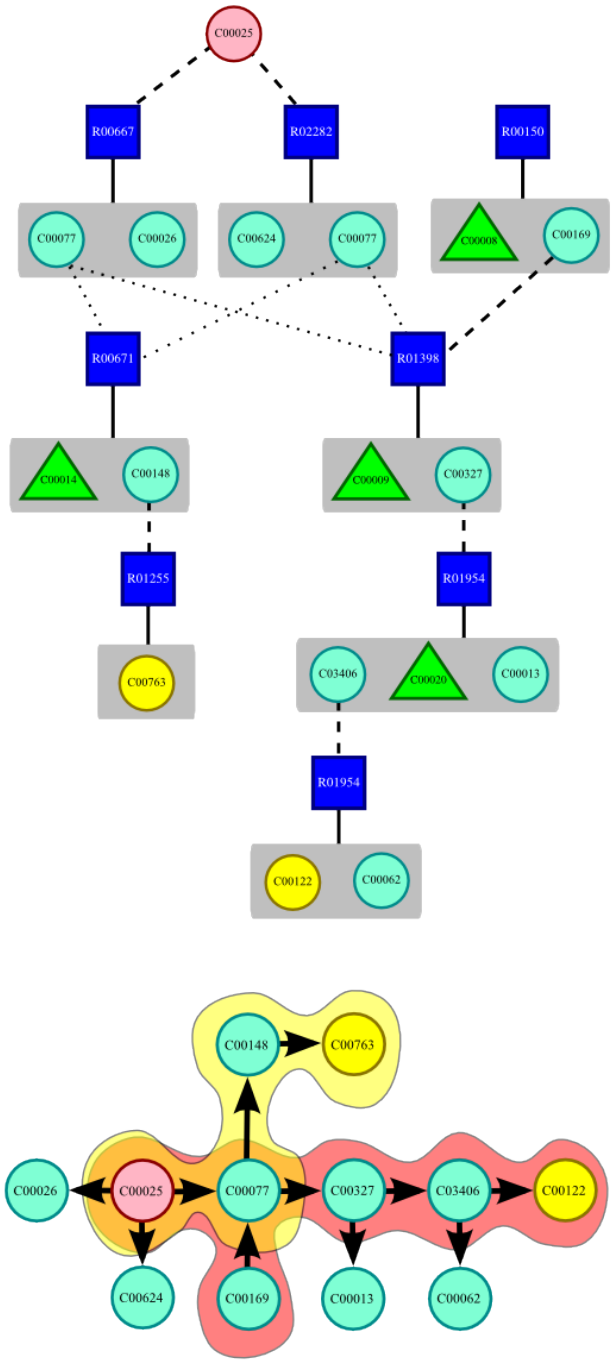


Figura 4.10: Familias de soluciones `Ia`. Junto a cada vía metabólica se muestra el grafo metabólico asociado. Las regiones sombreadas sobre los grafos metabólicos indican los compuestos que participan en la secuencia que relaciona: C00025–C00122 (anaranjado) y C00025–C00763 (amarillo).

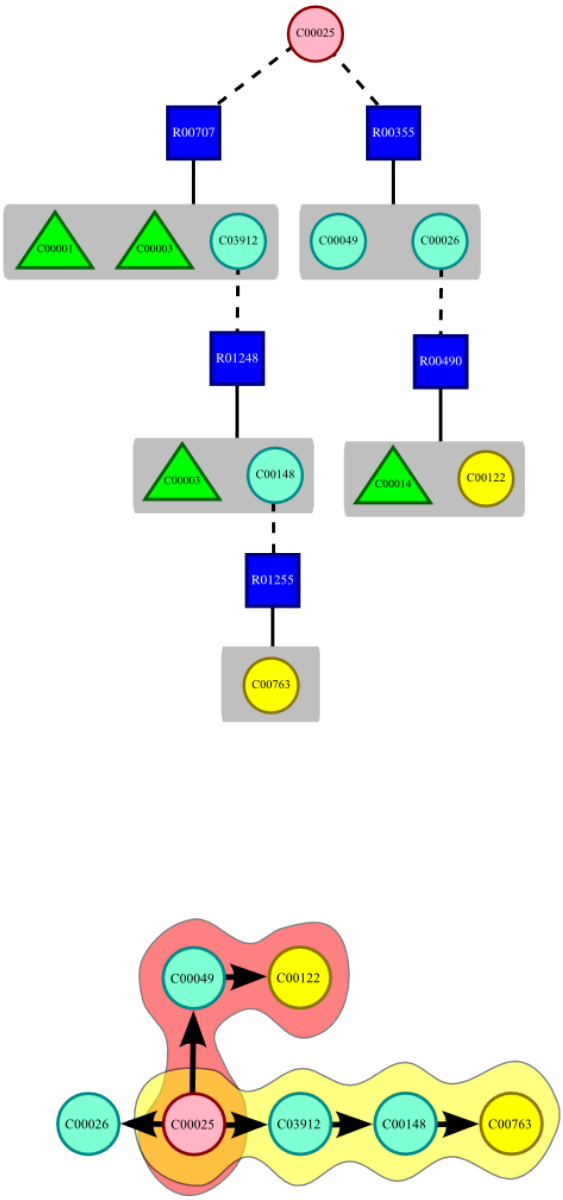


Figura 4.11: Familias de soluciones `Vb`. Junto a cada vía metabólica se muestra el grafo metabólico asociado. Las regiones sombreadas sobre los grafos metabólicos indican los compuestos que participan en la secuencia que relaciona: C00025–C00122 (anaranjado) y C00025–C00763 (amarillo).

Las dos primeras familias que se analizan son `IVa` y `Vb` (Figuras 4.8 y 4.9, respectivamente), para verificar que no presentan diferencias en la secuencia de compuestos. Claramente, el número de reacciones de `IVa` es mayor que el que emplea `Vb`. Sin embargo, cuando se analizan las secuencias de compuestos que se indican en amarillo en los grafos metabólicos se observa que los compuestos que emplea `Vb` están incluidos en la secuencia de `IVa`. Una situación similar ocurre con las secuencias resaltadas en rojo, haciendo una salvedad para el compuesto C00049. Este se encuentra incluido en el conjunto de compuestos abundantes unificado, por lo que no debe ser considerado para el cálculo de las diferencias. Por lo tanto, los compuestos usados por `Vb` también son empleados por `IVa` para establecer la conexión, haciendo que la diferencia entre estas vías sea nula.

Al analizar las dos familias restantes (Figuras 4.10 y 4.11) se aprecia que ambas contienen el mismo número de reacciones. Las secuencias de compuestos que se emplean para sintetizar C00763 a partir de C00025 sólo comparten el compuesto intermediario C00148, el cual es producido a través de diferentes reacciones. La síntesis de C00122 se realiza empleando secuencias de compuestos totalmente diferentes en ambas vías metabólicas, por lo que sólo comparten los extremos de la secuencia. Esto lleva a que, de forma general, sólo compartan 4 compuestos (C00148 y los tres compuestos a relacionar) y presenten una diferencia de 0.5 en la secuencia.

4.3.3. Casos de estudio

En esta subsección se evalúa la capacidad del algoritmo para encontrar dos vías metabólicas conocidas empleando el conjunto completo de reacciones que componen el metabolismo de un organismo. Para llevar a cabo los experimentos se emplearon las reacciones del metabolismo de la bacteria *E. coli* almacenadas en KEGG, y la reversibilidad de las reacciones se estableció de acuerdo a los *mapas de referencia* del organismo. Se eligió este organismo en particular para comparar las vías encontradas por AEBCAC con aquellas que encuentra el algoritmo de búsqueda de vías ramificadas propuesto por Faust et al. [2010] para el mismo problema. Este se basa en la búsqueda y combinación de los $k-$caminos de menor costo entre todos los pares de compuestos que se quiere relacionar. Las vías metabólicas de referencia seleccionadas para reproducir corresponden a dos de las vías empleadas por *Faust et al.*, las cuales pertenecen a la base de datos MetaCyc [Altman et al.,

2013]. Las reacciones que componen estas vías también están presentes en KEGG. Cabe mencionar que la reproducción de vías metabólicas conocidas no es el uso habitual de AEBCAC. Esto se realiza simplemente con la finalidad de validar los resultados encontrados y compararlos con los que pueden obtenerse empleando otras técnicas del estado del arte.

Durante los experimentos no se consideraron los compuestos externos. Aunque su uso sólo incrementa el número de reacciones que pueden llevarse a cabo, las mismas no participan de las vías estudiadas. Por lo tanto, no fueron incorporados durante las búsquedas.

4.3.3.1. Caso 1: conexión entre dos compuestos

El primer caso de estudio consiste en la búsqueda de una vía metabólica que sintetice el compuesto C00112 (CDP) partiendo de C00064 (L-Glutamina). En particular, se busca reproducir la vía metabólica que realiza la síntesis *de novo* de los ribonucleótidos pirimidínicos[2]. Esta vía metabólica está formada por una secuencia de 10 reacciones, donde el noveno paso de la secuencia puede ser llevado a cabo por dos reacciones diferentes. La Figura 4.12 presenta un esquema de la misma.

Los experimentos se realizaron limitando los compuestos necesarios sólo a aquellos requeridos por la primera reacción de la vía (R00575). Además, se empleó un conjunto de compuestos abundantes conteniendo sólo los compuestos necesarios junto con C00008, C00014, C00049, C00119, C00122. La búsqueda se realizó empleando la mejor configuración obtenida en la Sección 4.3.1. Las Figuras 4.13 y 4.14 presentan dos de las vías encontradas.

Claramente puede apreciarse que ambas vías emplean una secuencia válida de reacciones para producir C00112 (indicado en amarillo) a partir de C00064 (indicado en rojo). En particular, la Figura 4.13 presenta una vía metabólica que contiene 10 reacciones, y que reproduce el 90 % de las reacciones de la vía de referencia de la Figura 4.12. Esto se debe a que reemplaza la reacción R00156 por una reacción válida alternativa que genera el mismo producto para continuar con la síntesis de C00112. Por su parte, la vía presentada en la Figura 4.14 reproduce completamente la vía de referencia, adicionando una reacción al inicio que permite sintetizar el compuesto inicial a partir de C00025. En cambio, Faust *et al.* logran recuperar el 28 %

[2] `http://www.metacyc.org/META/NEW-IMAGE?type=NIL&object=PWY0-162`

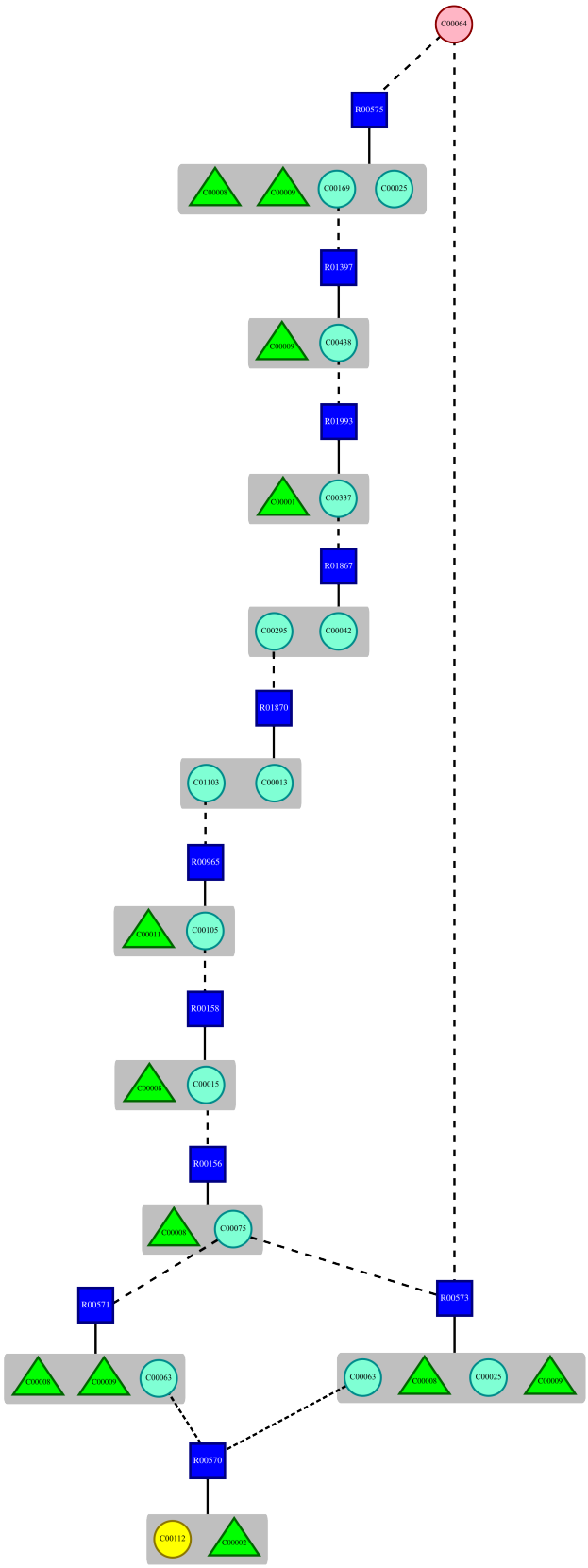


Figura 4.12: Vía metabólica de referencia para la síntesis *de novo* de los ribonucleótidos pirimidínicos.

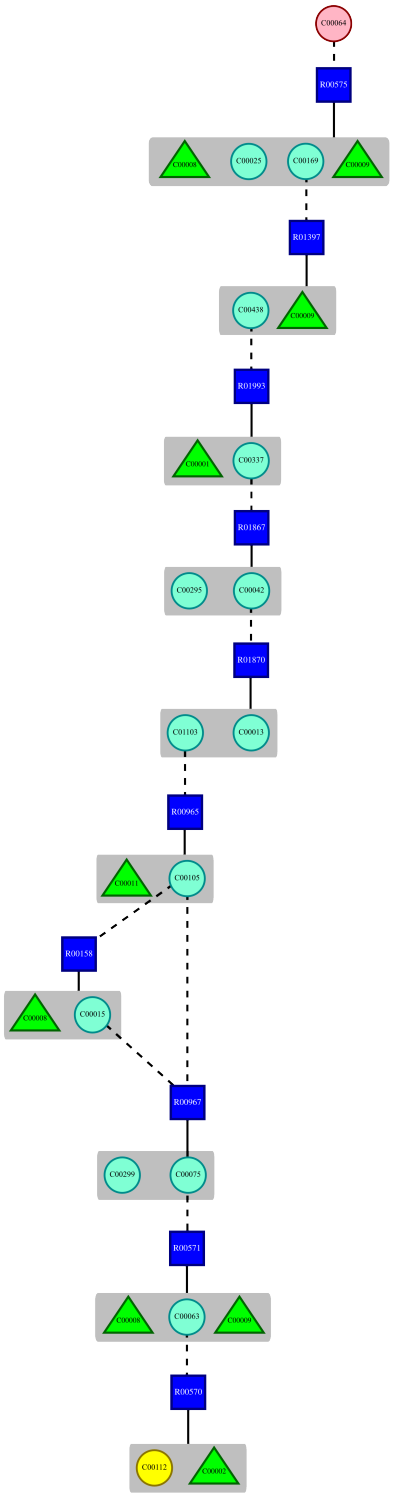


Figura 4.13: Vía metabólica que relaciona los compuestos C00064 y C00112, encontrada por AEBCAC empleando $N_M = 100$ genes. Los compuestos a relacionar se indican en rojo y amarillo.

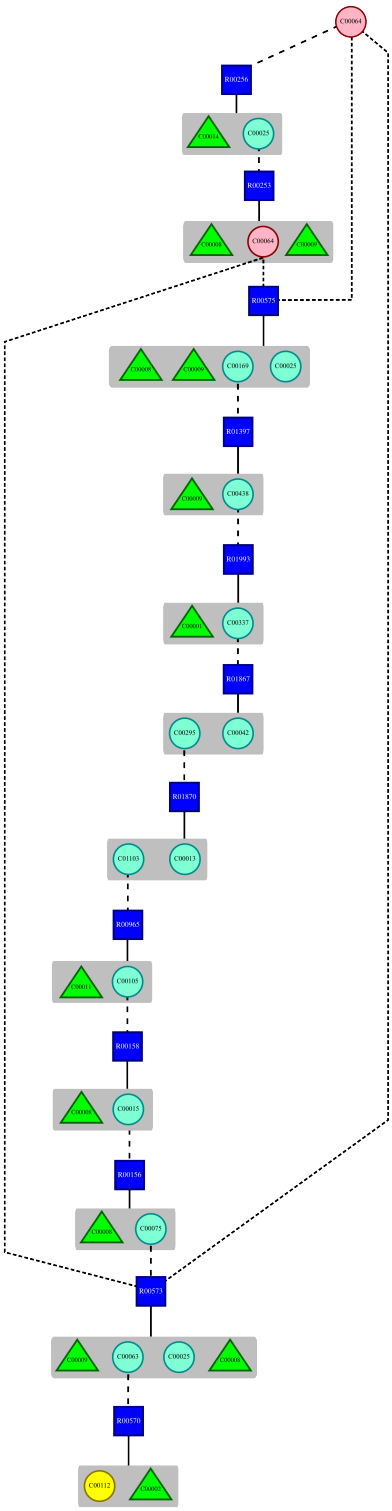


Figura 4.14: Vía metabólica que relaciona los compuestos C00064 y C00112, encontrada por AEBCAC empleando $N_M = 100$ genes. Los compuestos a relacionar se indican en rojo y amarillo.

y 59 % de la vía de referencia empleando 2 y 4 nodos intermedios para la búsqueda, respectivamente. Sin embargo, la primera vía que presentan en su trabajo es biológicamente irrelevante ya que sugiere la síntesis en tres pasos de C00112 a partir de C00064 mediante las reacciones R00575, R00573 y R00570 (a través de los compuestos C00025 y C00063), sin contemplar los compuestos adicionales que se requieren para esas reacciones (ver Figura 4.12). A diferencia de estas soluciones, AEBCAC proporciona algunas vías alternativas que no reproducen completamente a la vía de referencia, pero que contienen reacciones que son válidas a partir del conjunto de compuestos disponibles especificado. Además, estas vías alternativas pueden ser de utilidad para identificar reacciones que se conservan entre vías diferentes, y que pueden ser claves para la síntesis de los compuestos buscados.

Habiendo reproducido la vía de referencia a partir de un conjunto restringido de compuestos disponibles, se amplió este conjunto mediante la incorporación de los compuestos abundantes empleados en las secciones previas. En experiencias preliminares se observó que la ampliación del conjunto de compuestos disponibles conduce a un importante incremento en el número de reacciones posibles. Por tal motivo fue necesario incrementar el número máximo de genes en el cromosoma para lograr un muestreo adecuado del conjunto de reacciones ($N_M = 200$). La Figura 4.15 presenta una vía metabólica obtenida bajo estas condiciones experimentales. Esta vía contiene 10 reacciones y reproduce el 60 % de las reacciones de la vía de referencia. Sin embargo, esta vía metabólica requiere sólo de 9 reacciones para realizar la síntesis ya que la reacción adicional (R01071) es redundante y aporta el mismo compuesto que es sintetizado por R01870. Además, resulta interesante observar que la reacción R00662 requiere del producto de dos reacciones previas que deben ocurrir de forma secuencial para que los sustratos de la misma estén disponibles.

Estos experimentos se realizaron en un PC INTEL CORE i7 3.4GHz con 6Gb de memoria RAM. El algoritmo fue implementado en Matlab, paralelizando en cada corrida la evaluación de la función de aptitud empleando 8 núcleos. La ejecución de 100 generaciones insumió aproximadamente 1.5 hs y 2.7 hs para los experimentos realizados con $N_M = 100$ y $N_M = 200$ genes, respectivamente.

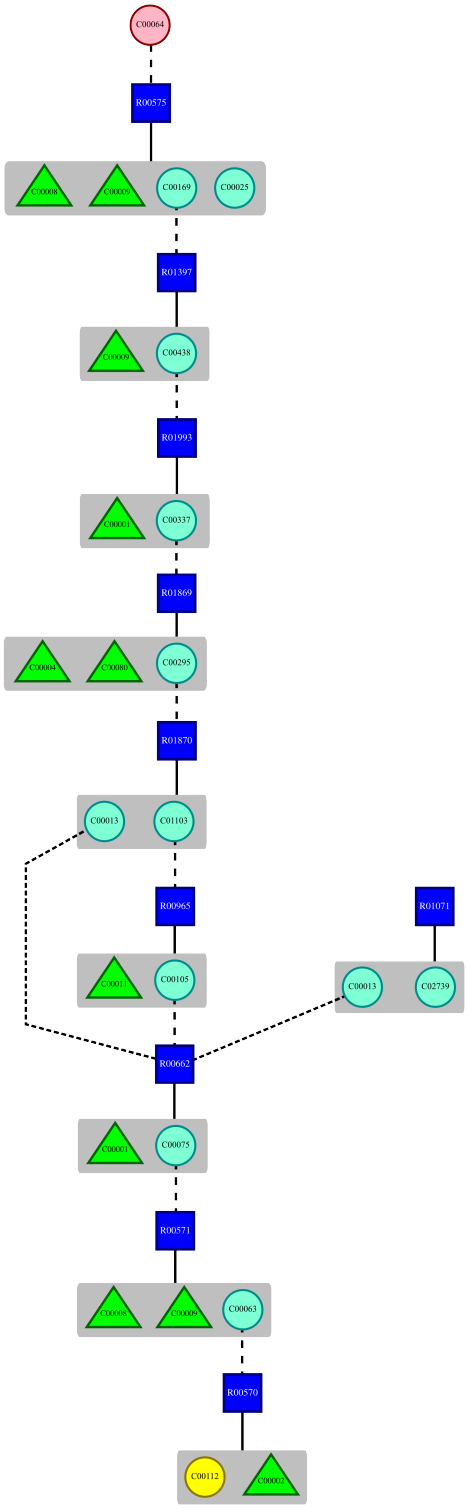


Figura 4.15: Vía metabólica que relaciona los compuestos C00064 y C00112, encontrada por AEBCAC empleando $N_M = 200$ genes. Los compuestos a relacionar se indican en rojo y amarillo.

4.3.3.2. Caso 2: conexión entre cuatro compuestos

El segundo caso de estudio consiste en reproducir la vía que sintetiza los compuestos C00047 (L-Lisina), C00073 (L-Metionina) y C00188 (L-Treonina) a partir de C00036 (Oxaloacetato). Ésta corresponde a la vía de referencia perteneciente a la base de datos MetaCyc denominada *Superpathway of lysine, threonine and methionine biosynthesis I*[3], aunque la reacciones que sintetizan estos compuestos se encuentran distribuidas en distintas vías de referencia en KEGG. La vía metabólica de referencia se presenta en la Figura 4.16. Esta vía está formada por una secuencia de 18 reacciones, y el compuesto C00441 es empleado por tres subsecuencias de reacciones que conducen a la síntesis de los tres compuestos de interés.

La búsqueda se llevó a cabo limitando los compuestos necesarios sólo a aquellos requeridos por la primera reacción de la vía (R00355). Los experimentos se realizaron empleando la misma configuración que en las búsquedas previas, limitando el tamaño máximo del cromosoma a $N_M = 200$ genes. Se utilizó el conjunto de compuestos abundantes presentado en la Tabla 4.1, adicionando los compuestos C00022, C00025, C00091, C00097, C00440 y C04489 ya que son necesarios para algunas de las reacciones de la vía de referencia. Las Figuras 4.17 y 4.18 presentan dos vías encontradas con esta configuración experimental.

La vía presentada en la Figura 4.17 reproduce el 28 % de las reacciones de la vía de referencia. Aunque la vía encontrada no presenta una secuencia que genere el compuesto C00047, el cromosoma que la representa contiene una reacción que lo produce pero que no es incorporada a la vía ya que carece de los sustratos para ser válida. Es probable que el uso de un número de generaciones mayor que el actual ($G_M = 1000$) sea necesario para que la vía relacione los cuatro compuestos. Por su parte, la vía de la Figura 4.18 recupera el 44 % de la vía de referencia. Las reacciones no recuperadas corresponden principalmente a la rama de la vía que sintetiza C00047.

De igual manera que Faust *et al.*, AEBCAC reproduce parcialmente la vía de referencia. En su trabajo, el algoritmo de los k-caminos más cortos recupera el 65 % y 85 % de las reacciones de la red empleando 5 y 7 reacciones como nodos por los que debe atravesar la red, respectivamente. Si bien estos porcentajes son mayores, deben tenerse en cuenta dos aspectos. En primer

[3] `http://www.metacyc.org/META/NEW-IMAGE?type=NIL&object=P4-PWY`

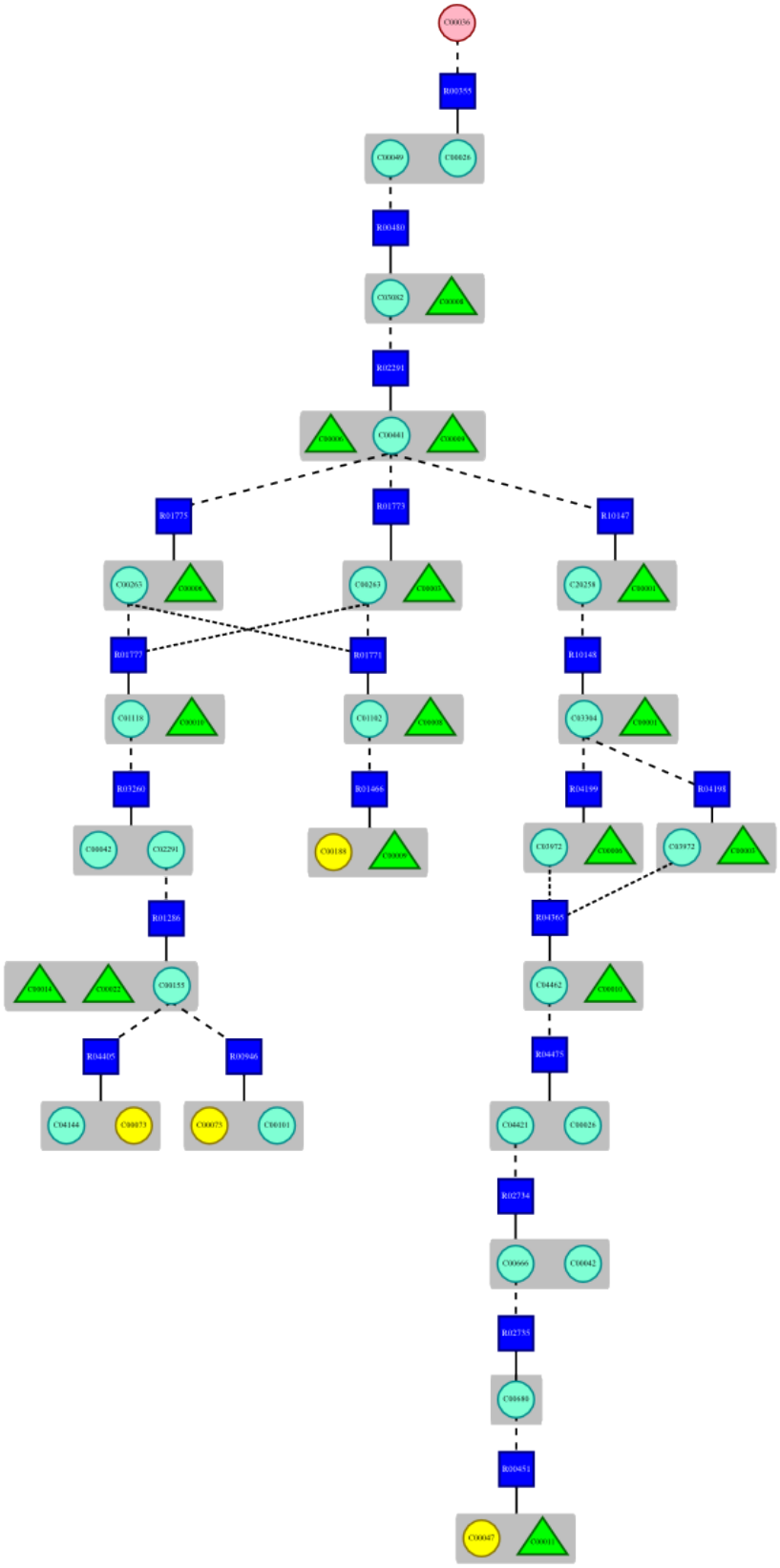


Figura 4.16: Vía metabólica de referencia denominada *Superpathway of lysine, threonine and methionine biosynthesis I.*

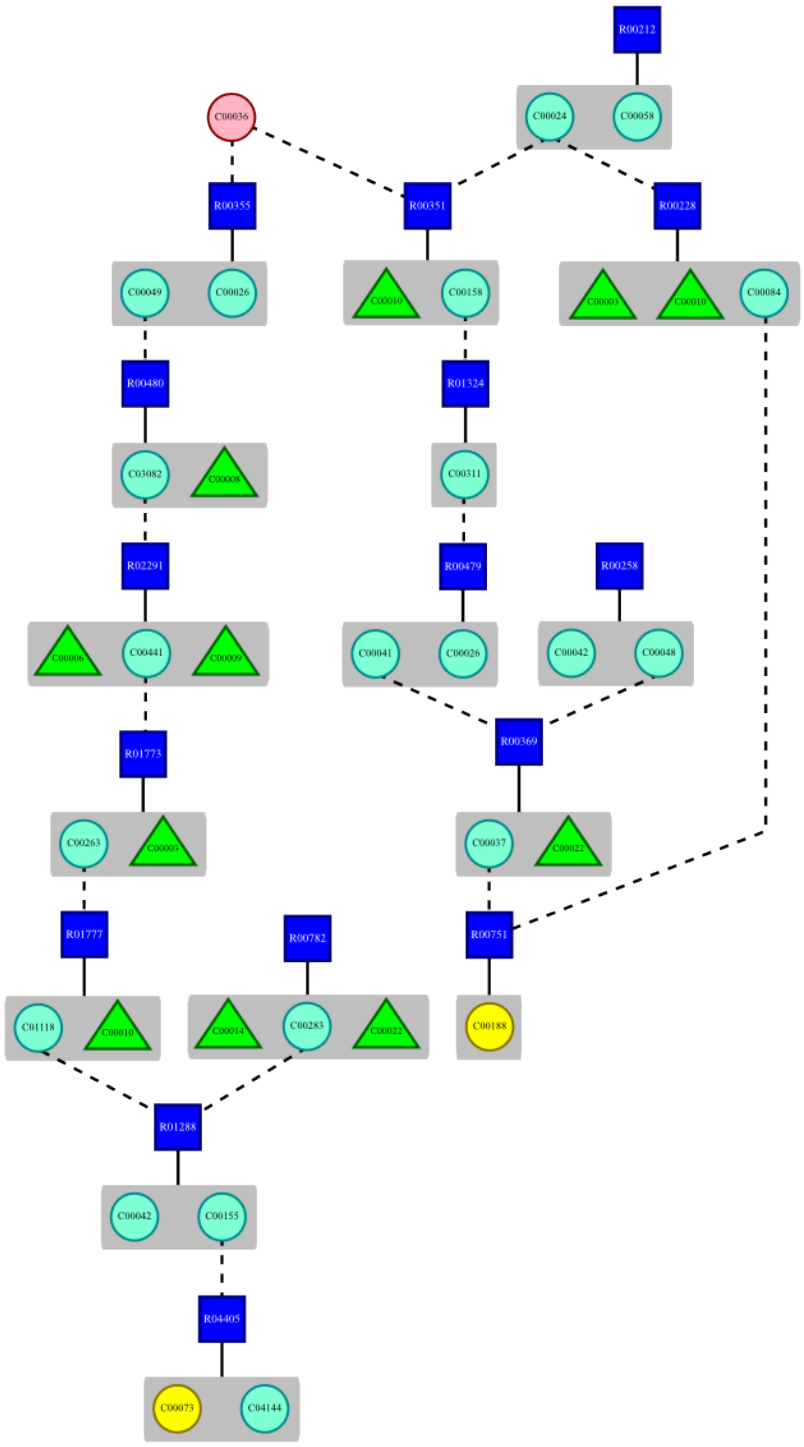


Figura 4.17: Vía metabólica que relaciona los compuestos C00036, C00047, C00073 y C00188, encontrada por AEBCAC empleando $N_M = 200$ genes. El compuesto inicial se indica en rojo y los restantes compuestos a relacionar se resaltan en amarillo.

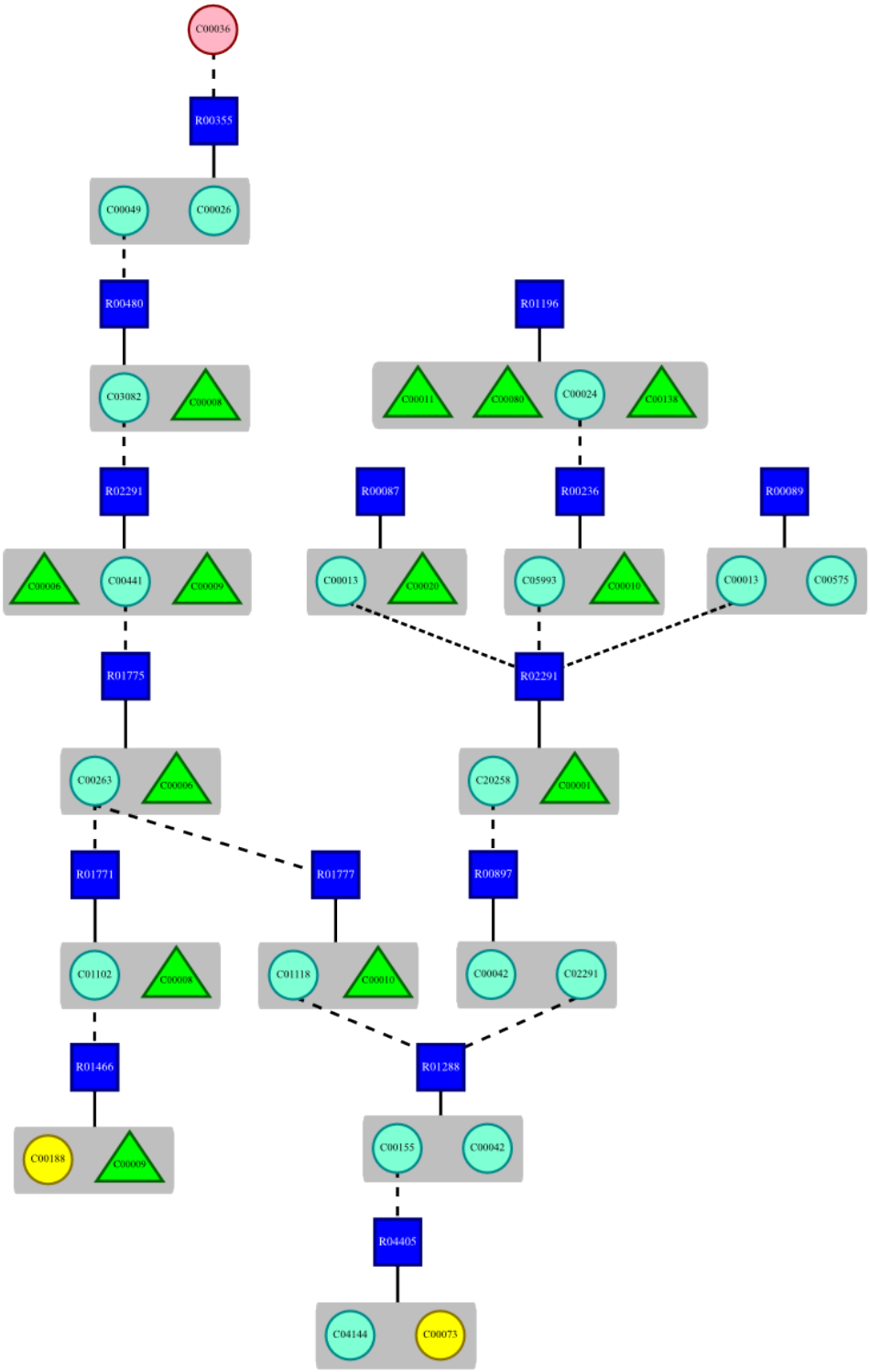


Figura 4.18: Vía metabólica que relaciona los compuestos C00036, C00047, C00073 y C00188, encontrada por AEBCAC empleando $N_M = 200$ genes. El compuesto inicial se indica en rojo y los restantes compuestos a relacionar se resaltan en amarillo.

lugar, la información empleada por AEBCAC es menos restrictiva que la que emplea el método de Faust *et al.*, ya que se define un conjunto de compuestos en lugar de reacciones para orientar la búsqueda. Esto marca una diferencia importante si sólo se conocen algunos compuestos intermediarios de la vía, en lugar de conocer algunas reacciones que participan. En segundo lugar, las vías encontradas con el algoritmo evolutivo son completamente posibles a partir de los compuestos disponibles especificados.

4.4. Comentarios finales

En el presente capítulo se abordó el problema de buscar vías metabólicas que relacionen un conjunto dado de compuestos. Para ello se propuso un algoritmo evolutivo denominado AEBCAC, basado en principios similares a los empleados en el capítulo anterior. Este nuevo algoritmo modela una secuencia de reacciones entre compuestos dentro de cada cromosoma, la cual relaciona un conjunto de compuestos mediante una red que se inicia en uno de ellos. Esta codificación se basa en la propuesta de un modelo denominado *conjuntos anidados de compuestos*, que permite generar redes de reacciones en las que los sustratos de cada reacción se encuentran disponibles.

El algoritmo se evaluó en términos de medidas de desempeño y de características de las vías metabólicas que encuentra. Los resultados muestran que el uso de una estrategia de inicialización de los cromosomas con tamaños variables y el uso de probabilidades de mutación y borrado de genes relativamente elevadas producen el mejor desempeño del algoritmo. Las búsquedas realizadas sobre dos conjuntos de reacciones, donde uno es una expansión del otro, encuentran vías similares, además de otras adicionales debido al mayor número de conexiones entre compuestos. Cuando se abordó el problema de buscar vías de referencia considerando el conjunto completo de reacciones de un organismo dado, se observó que el algoritmo es capaz de reproducir dichas vías en forma total o parcial. En este último caso, los resultados obtenidos proveyeron información interesante sobre mecanismos alternativos de síntesis de los compuestos.

Las vías metabólicas encontradas a lo largo de los experimentos que se llevaron a cabo en el presente capítulo reflejan claramente la capacidad de encontrar vías ramificadas. En todos los casos las vías encontradas a partir de un conjunto de compuestos especificado fueron factibles, proveyendo

resultados con validez biológica. Además, en los casos donde las reacciones requerían de un sustrato no abundante, el algoritmo fue capaz de incorporar previamente las reacciones que los generan.

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
M. Gerard; "Desarrollo de algoritmos evolutivos para el descubrimiento de relaciones en datos metabólicos"
Universidad Nacional del Litoral, 2014.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

En esta tesis se abordó el problema de búsqueda de vías metabólicas que relacionen un conjunto de compuestos, mediante el desarrollo de nuevos algoritmos de búsqueda basados en la computación evolutiva. Las principales conclusiones de esta tesis se resumen en los siguientes puntos:

- Se propusieron dos algoritmos evolutivos originales que realizan la búsqueda de vías metabólicas con diferentes topologías. Ambas propuestas toman elementos de los algoritmos genéticos e incorporan modificaciones en la representación del problema y los operadores de variación. Además, emplean estrategias de inicialización que generan individuos que contienen secuencias de transformaciones/reacciones válidas, donde cada gen cumple con los requisitos para que una transformación/-reacción bioquímica pueda producirse.

- Se desarrollaron dos representaciones para modelar los distintos tipos de relaciones. Ambas emplean relaciones con sentido único, de manera que cada relación reversible fue descompuesta en dos relaciones independientes de sentido opuesto. La primera representación emplea relaciones sustrato–producto para caracterizar a las reacciones. Cada gen codifica una relación, la cual depende de la relación previa para poder

producirse. La secuencia en que éstas se realizan se almacena en forma ordenada en el cromosoma, permitiendo modelar vías metabólicas lineales que transforman un compuesto en otro. La segunda representación emplea reacciones que pueden producirse a partir de un conjunto de compuestos. Las reacciones son almacenadas en el cromosoma en el orden en que se producen, siendo factibles sólo si los sustratos están disponibles en un conjunto de compuestos, el cual es actualizado con los productos de cada nueva reacción factible. Esto permitió modelar secuencias de reacciones con distinto tipo de interconexiones.

- Se diseñaron nuevos operadores de mutación y cruza que emplean información del cromosoma. Para el caso de la búsqueda de vías metabólicas lineales, el operador de cruza modificado selecciona puntos de cruza en ambos padres, de manera que el empalme del material genético produzca una conexión válida. El operador de mutación modificado fue diseñado para reemplazar un gen por otro que estableciera una conexión válida con alguno de los genes vecinos. Para la búsqueda de vías ramificadas se modificó el operador de cruza en un punto para seleccionar dos puntos de cruza en el segundo padre. Esto permitió reducir el número de reacciones inválidas que son incorporadas por la cruza. El operador de mutación fue diseñado para eliminar una reacción o insertar una reacción nueva que pueda ser producida a partir de los compuestos disponibles en el punto donde se realiza la inserción. El uso de estos operadores mejoró el desempeño respecto a los operadores tradicionales, reduciendo el número de generaciones requerido por ambos algoritmos para encontrar una vía metabólica que relacione los compuestos especificados.

- Se exploraron diferentes medidas para evaluar propiedades deseadas de las soluciones buscadas. Se analizaron aspectos tales como la validez de la secuencia de reacciones, en términos de disponibilidad de sustratos, para establecer relaciones que tengan sentido biológico. También se consideraron medidas para penalizar la inclusión de bucles en la solución. El uso combinado de estas medidas proporcionó una amplia variedad de soluciones en tiempos de búsqueda razonables.

- Se propusieron diferentes medidas para analizar y comparar las soluciones encontradas, entre las que se consideraron el número de reacciones,

la ramificación de la vía metabólica y el número de hojas (productos de reacciones no usados). Además, se propuso una medida para evaluar el nivel de diferencia entre dos vías metabólicas en términos de los compuestos usados para relacionar los compuestos de interés.

- Los algoritmos propuestos fueron evaluados empleando bases de datos biológicos y realizando diferentes tipos de búsqueda. El algoritmo para la búsqueda de vías metabólicas lineales fue comparado con la estrategia de búsqueda en amplitud, obteniendo un desempeño similar, pero siendo capaz de encontrar los caminos más cortos y otros con mayor número de reacciones. Esta variabilidad proporcionó mecanismos alternativos para la síntesis de los compuestos. Realizando búsquedas para resolver problemas de interés biológico, el algoritmo encontró una vía metabólica alternativa a la vía metabólica de referencia de la glicólisis para sintetizar el compuesto glicerato-2P a partir de α-D-glucosa-1P. También fue capaz de encontrar una vía que relaciona los compuestos L-fucosa y glicina en la planta *Arabidopsis thaliana*, proponiendo el uso de dos reacciones no reportadas aún en la literatura para este organismo. El algoritmo para la búsqueda de vías ramificadas fue capaz de encontrar diferentes vías que relacionaron tres compuestos específicos, en pruebas realizadas sobre espacios de soluciones de diferente tamaño. Todas las soluciones encontradas por este algoritmo relacionaron los tres compuestos, y las ramificaciones evidenciaron las relaciones establecidas entre las reacciones. El funcionamiento de este algoritmo fue validado mediante la búsqueda de dos vías metabólicas de referencia. La primera, que relacionaba dos compuestos, fue reproducida completamente. El algoritmo también encontró nuevas secuencias de reacciones para relacionar estos mismos compuestos. La segunda vía, que relacionaba cuatro compuestos, fue reproducida parcialmente generando soluciones con nuevos mecanismos para sintetizar algunos de estos compuestos. En todos los casos, las soluciones encontradas fueron viables a partir de los compuestos iniciales especificados.

- Se implementó una estrategia para representar gráficamente las vías metabólicas encontradas, codificando la información mediante código de colores y símbolos que facilitan la identificación de elementos relevantes de las soluciones encontradas. Las representaciones logradas

reflejan el orden en que las reacciones se llevan a cabo, permitiendo seguir el proceso de síntesis de los compuestos de forma gráfica y estructurada.

5.2. Publicaciones resultantes del desarrollo de esta tesis

[1] Gerard M., Stegmayer G., Milone D.H. (2013), An evolutionary approach for searching metabolic pathways, *Computers in Biology and Medicine* (IF2012: 1.162), 43(11), 1704–1712.
DOI: 10.1016/j.compbiomed.2013.08.017

[2] Stegmayer G., Gerard M., Milone D.H. (2012), Data mining over biological datasets: an integrated approach based on computational intelligence, *IEEE Computational Intelligence Magazine - Special Issue on Computational Intelligence in Bioinformatics* (IF2012: 4.629), 7(4), 22–34.
DOI: 10.1109/MCI.2012.2215122

[3] Gerard M., Stegmayer G., Milone D.H., Un algoritmo evolutivo para la búsqueda de vías metabólicas (2012), *Revista Iberoamericana de Inteligencia Artificial*, 15(49), 1–12.
http://polar.lsi.uned.es/revista/index.php/ia/article/view/680/794

[4] Gerard M., Stegmayer G., Milone D.H. (2012), Metabolic pathfinding based on genetic algorithms, *3er Congreso Argentino de Bioinformática y Biología Computacional (3CAB2C)*, Oro Verde - Entre Ríos, Argentina, Setiembre (2012).

[5] Gerard M., Stegmayer G., Milone D.H., Búsqueda evolutiva de vías metabólicas, *Argentine Symposium On Artificial Intelligence* (ASAI 2010), *40 Jornadas Argentinas de Informática* (40 JAIIO), Buenos Aires, Argentina, Agosto (2010).

[6] Gerard M., Integración y minería de datos en bioinformática, Seminarios PAE-CELTIC 2009, Febrero (2010), INGAR, Santa Fe, Argentina.
http://www.celtic.santafe-conicet.gov.ar/seminarios2009

[7] Milone D.H., Stegmayer G., Gerard M., Kamenetzky L., López M., Carrari F.: Analysis and integration of biological data: a data mining approach using neural networks. In: Knowledge Discovery Practices and Emerging Applications of Data Mining: Trends and New Domains, pp.287–314. IGI-Global (2010).
DOI: 10.4018/978-1-60960-067-9.ch014

[8] Milone D.H., Stegmayer G., Gerard M., Kamenetzky L., López M., Carrari F., Métodos de agrupamiento no supervisado para la integración de datos genómicos y metabólicos de múltiples líneas de introgresión, *Argentine Symposium On Artificial Intelligence* (ASAI 2009), *41 Jornadas Argentinas de Informática* (39 JAIIO), Mar del Plata, Argentina, Setiembre (2009).

[9] Milone D.H., Stegmayer G., Gerard M., Kamenetzky L., López M., Carrari F. (2009), Métodos de agrupamiento no supervisado para la integración de datos genómicos y metabólicos de múltiples líneas de introgresión, *Revista Iberoamericana de Inteligencia Artificial*, 13(44), 56–66.
http://polar.lsi.uned.es/revista/index.php/ia/article/view/624/607

5.3. Trabajo futuro

Los trabajos futuros estarán orientados principalmente a mejorar y extender los resultados alcanzados con la segunda propuesta. En lo referente al algoritmo, se pretende:

- Desarrollar una estrategia para buscar redes metabólicas que utilicen múltiples compuestos iniciales para relacionar un conjunto de compuestos especificado;
- Evaluar el uso de estrategias de *niching* para favorecer la diversidad poblacional y la coexistencia de subpoblaciones que interactúen;
- Desarrollar un operador que seleccione el punto de cruza en base a la información de los conjuntos anidados;
- Evaluar otros mecanismos de mutación, tales como el reemplazo de reacciones, operadores que eliminen reacciones no usadas y que favorezcan el reordenamiento del cromosoma;

- Profundizar el estudio del operador de selección, considerando torneos entre un mayor número de individuos y evaluando otras estrategias de selección;
- Estudiar el efecto de relajar el criterio de finalización sobre el desempeño de los algoritmos y de las características de las soluciones obtenidas.

Con respecto al modelado del problema, los esfuerzos estarán orientados hacia:

- Incorporar el concepto de *utilidad* de las reacciones en el modelado, considerando *útiles* a todas las reacciones que proporcionan alguno de los sustratos requeridos para la síntesis de los compuestos a relacionar.
- Incorporar información acerca de la estequiometría de las reacciones, la conectividad de los compuestos, la disponibilidad de las enzimas y la energía de activación en el modelado del problema. Además, considerar el uso de niveles de acumulación de compuestos y de expresión de genes medidos experimentalmente.
- Evaluar el uso de herramientas provenientes del Análisis de Modos Elementales [Trinh et al., 2009] y del Análisis de Balance de Flujos [Lee et al., 2006] para favorecer redes metabólicas que sean más viables.

En forma complementaria a las propuestas anteriores, también se trabajará en

- Desarrollar medidas para evaluar la calidad y factibilidad biológica de las vías metabólicas encontradas, por ejemplo, tomando en cuenta el número de compartimientos intracelulares que participan o la cantidad de veces que son atravesados. La evaluación de la validez de las vías se podría realizar empleando bases de datos biológicas tales como AraCyc [Mueller et al., 2003], EcoCyc [Keseler et al., 2005], BioCyc y MetaCyc [Caspi et al., 2011].
- Profundizar en el estudio de los algoritmos Coevolutivos [Potter and Dong, 2000], los cuales evolucionan guiados por una función de aptitud que depende del tipo de interacción definido entre las distintas subpoblaciones.

- Evaluar otras técnicas de búsqueda metaheurísticas de inteligencia colectiva, tales como Colonia de Hormigas [Dorigo et al., 1996] y Optimización por Enjambre de Partículas [Kennedy and Eberhart, 1995], empleando el modelo de conjuntos anidados.

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
M. Gerard; "Desarrollo de algoritmos evolutivos para el descubrimiento de relaciones en datos metabólicos"
Universidad Nacional del Litoral, 2014.

Bibliografía

Aittokallio, T. and Schwikowski, B. (2006). Graph-based methods for analysing networks in cell biology. *Briefings in Bioinformatics*, 7:243–255.

Albert, R. and Barabasi, A. (2002). Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97.

Altman, T., Travers, M., Kothari, A., Caspi, R., and Karp, P. (2013). A systematic comparison of the MetaCyc and KEGG pathway databases. *BMC Bioinformatics*, 14:112.

Arita, M. (2000). Metabolic reconstruction using shortest paths. *Simulation Practice and Theory*, 8:109–125.

Arita, M. (2012). *Bacterial Molecular Networks: Methods and Protocols*, volume 804 of *Methods in Molecular Biology*, chapter From Metabolic Reactions to Networks and Pathways, pages 93–106. Springer.

Bäck, T., Fogel, D., and Michalewicz, Z. (2000). *Evolutionary Computation I: Basic Algorithms and Operators*. Institute of Physics Publishing.

Blum, T. and Kohlbacher, O. (2008). MetaRoute: fast search for relevant metabolic routes for interactive network navigation and visualization. *Bioinformatics*, 24:2108–2109.

Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., and Hwang, D.-U. (2006). Complex networks: Structure and dynamics. *Physics Reports*, 424:175–308.

Carrari, F., Baxter, C., Usadel, B., Urbanczyk-Wochniak, E., Zanor, M., A. Nunes-Nesi, V. Nikiforova, Centero, D., Ratzka, A., Pauly, M., Sweetlove, L., and Fernie, A. (2006). Integrated Analysis of Metabolite and Transcript Levels Reveals the Metabolic Shifts That Underlie Tomato Fruit Development and Highlight Regulatory Aspects of Metabolic Network Behavior. *Plant Physiology*, 142:1380–1396.

Caspi, R., Altman, T., Dreher, K., Fulcher, C., Subhraveti, P., Keseler, I., Kothari, A., Krummenacker, M., Latendresse, M., Mueller, L., Ong, Q., Paley, S., Pujar, A., Shearer, A., Travers, M., Weerasinghe, D., Zhang, P., and Karp, P. (2011). The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of pathway/genome databases. *Nucleic Acids Research*, 40:D742–53.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. MIT Press.

Costa, L., Oliveira, O. N., Travieso, G., Rodrigues, F. A., Boas, P. R. V., Antiqueira, L., Viana, M. P., and Rocha, L. E. C. (2011). Analyzing and modeling real-world phenomena with complex networks: a survey of applications. *Advances in Physics*, 60:329–412.

Croes, D., Couche, F., Wodak, S., and van Helden, J. (2006). Inferring meaningful pathways in weighted metabolic networks. *Journal of Molecular Biology*, 356:222–236.

Derrac, J., García, S., Molina, D., and Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1:3–18.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, 26:1–13.

Easton, J., Harris, L., Viant, M., Peet, A., and Arvanitis, T. (2010). Linked Metabolites: A tool for the construction of directed metabolic graphs. *Computers in Biology and Medicine*, 40:340–349.

Efron, B. and Tibshirani, R. (1993). *An introduction to the bootstrap*. Chapman & Hall.

Eiben, A. E. and Smith, J. E. (2003). *Introduction to Evolutionary Computing*. Springer.

Engelbrecht, A. (2007). *Computational Intelligence: An Introduction*. Jhon Wiley.

Faust, K., Croes, D., and van Helden, J. (2009). Metabolic Pathfinding Using RPAIR Annotation. *Journal of Molecular Biology*, 388(2):390–414.

Faust, K., Dupont, P., Callut, J., and van Helden, J. (2010). Pathway discovery in metabolic networks by subgraph extraction. *Bioinformatics*, 26:1211–1218.

Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). *Artificial Intelligence through Simulated Evolution*. John Wiley.

Gen, M. and Chen, R. (2000). *Genetic Algorithms and Engineering Optimization*. John Wiley.

Gerlee, P., Lizana, L., and Sneppen, K. (2009). Pathway identification by network pruning in the metabolic network of *Escherichia coli*. *Bioinformatics*, 25:3282–3288.

Goldberg, D. E. (1997). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

Goto, S., Okuno, Y., Hattori, M., Nishioka, T., and Kanehisa, M. (2002). LIGAND: database of chemical compounds and reactions in biological pathways. *Nucleic Acids Research*, 30:402–404.

Gould, R. . (2012). Graph Theory. In *Dover Books on Mathematics*. Dover Publications.

Hammel, T. B. U. and Schwefel, H. (1997). Evolutionary Computation: Comments on the History and Current State. *IEEE Transactions on Evolutionary Computation*, 1:3–17.

Heath, A., Bennett, G., and Kavraki, L. (2010). Finding metabolic pathways using atom tracking. *Systems Biology*, 26:1548–1555.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press.

Huss, M. and Holme, P. (2007). Currency and commodity metabolites: their identification and relation to the modularity of metabolic networks. *IET Systems Biology*, 1:280–285.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948.

Keseler, I. M., Collado-Vides, J., Gama-Castro, S., Ingraham, J., Paley, S., Paulsen, I. T., Peralta-Gil, M., and Karp, P. D. (2005). EcoCyc: a comprehensive database resource for *Escherichia coli*. *Nucleic Acids Research*, 33:D334–D337.

Kotera, M., Hattori, M., Oh, M., Yamamoto, R., Komeno, T., Yabuzaki, J., Tonomura, K., Goto, S., and Kanehisa, M. (2004a). RPAIR: a reactant-pair database representing chemical changes in enzymatic reactions. *Genome Inform.*, 15:P062.

Kotera, M., Okuno, Y., Hattori, M., Goto, S., and Kanehisa, M. (2004b). Computational assignment of the EC Numbers for Genomic-Scale Analysis of Enzymatic Reactions. *Journal of the American Chemical Society*, 126:16487–16498.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

Küffner, R., Zimmer, R., and Lengauer, T. (2000). Pathway analysis in metabolic databases via differential metabolic display (DMD). *Bioinformatics*, 16(9):825–836.

Kumar, E. (2008). *Artificial Intelligence*. I.K. International Publishing House.

Lan, H., Carson, R., Provart, N., and Bonner, A. J. (2007). Combining classifiers to predict gene function in Arabidopsis Thaliana using large–scale gene expression measurements. *BMC Bioinformatics*, 8:358.

Lee, J. M., Gianchandani, E. P., and Papin, J. A. (2006). Flux balance analysis in the era of metabolmics. *Briefings in Bioinformatics*, 7:140–150.

Li, R., Moore, M., and King, J. (2003). Investigating the Regulation of One-carbon Metabolism in Arabidopsis thaliana. *Plant Cell Physiology*, 44:233–241.

Liang, C., Jaiswal, P., Hebbard, C., Avraham, S., Buckler, E., Casstevens, T., Hurwitz, B., McCouch, S., Ni, J., Pujar, A., Ravenscroft, D., Ren, L., Spooner, W., Tecle, I., Thomason, J., Tung, C., Wei, X., Yap, I., Youens-Clark, K., Ware, D., and Stein, L. (2008). Gramene: a growing plant comparative genomics resource. *Nucleic Acids Research*, 36:947–953.

McShan, D., Rao, S., and Shah, I. (2003). PathMiner: predicting metabolic pathways by heuristic search. *Bioinformatics*, 19:1692–1698.

Milone, D. H., Stegmayer, G., Kamenetzky, L., López, M., Lee, J., Giovannoni, J., and Carrari, F. (2010). *omeSOM: a software for clustering and visualization of transcriptional and metabolite data mined from interspecific crosses of crop plants. *BMC Bioinformatics*, 11:438.

Morbiducci, U., Tura, A., and Grigioni, M. (2005). Genetic algorithms for parameter estimation in mathematical modeling of glucose metabolism. *Computers in Biology and Medicine*, 35:862–874.

Mueller, L., Zhang, P., and Rhee, S. (2003). Aracyc: A biochemical pathway database for Arabidopsis. *Plant Physiology*, 132:453–460.

Nelson, D. L. and Cox, M. M. (2004). *Lehninger. Principles of Biochemistry.* W. H. Freeman.

Ogata, H., Goto, S., Fujibuchi, W., and Kanehisa, M. (1998). Computation with the KEGG pathway database. *BioSystems*, 47:119–128.

Palsson, B. Ø. (2006). *Systems Biology: Properties of Reconstructed Networks.* Cambridge University Press.

Palsson, B. Ø. (2011). *Systems Biology: Simulation of Dynamic Network States.* Cambridge University Press.

Pitkänen, E., Jouhten, P., and Rousu, J. (2009). Inferring branching pathways in genome-scale metabolic networks. *BMC Systems Biology*, 3:103–124.

Planes, F. and Beasley, J. (2008). A critical examination of stoichiometric and path-finding approaches to metabolic pathways. *Briefings in Bioinformatics*, 9:422–436.

Poole, D. L. and Mackworth, A. K. (2010). *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press.

Potter, M. A. and Dong, K. A. D. (2000). Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents. *Evolutionary Computation*, 8:1–29.

Rahman, S., Advani, P., Schunk, R., Schrader, R., and Schomburg, D. (2005). Metabolic pathway analysis web service (Pathway Hunter Tool at CUBIC). *Bioinformatics*, 21:1189–1193.

Ravasz, E., Somera, A. L., Mongru, D. A., Oltvai, Z. N., and Barabási, A.-L. (2002). Hierarchical Organization of Modularity in Metabolic Networks. *Science*, 297:1551–1555.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme und Prinzipien der biologischen Evolution*. Frommann-Holzboog.

Rothlauf, F. (2006). *Representations for Genetic and Evolutionary Algorithms*. Springer.

Ruppert, D. (2011). *Statistics and Data Analysis for Financial Engineering*. Springer.

Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach (3 Ed.)*. Prentice Hall.

Rutkowski, L. (2008). *Computational Intelligence: Methods and Techniques*. Springer.

Saito, K., Hirai, M. Y., and Yonekura-Sakakibara, K. (2008). Decoding genes with coexpression networks and metabolomics - 'majority report by precogs'. *Trends in Plant Science*, 13:36–43.

Sayama, H., Pestov, I., Schmidt, J., Bush, B. J., Wong, C., Yamanoi, J., and Gross, T. (2013). Modeling complex systems with adaptive networks. *Computers and Mathematics with Applications*, in press.

Schuster, S., Fell, D., and Dandekar, T. (2000). A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic networks. *Nature Biotechnology*, 18:326–332.

Sivanandam, S. and Deepa, S. (2008). *Introduction to Genetic Algorithms*. Springer.

Terzer, M. and Stelling, J. (2008). Large-scale computation of elementary flux modes with bit pattern trees. *Bioinformatics*, 24:2229–2235.

Trinh, C. T., Wlaschin, A., and Sriene, F. (2009). Elementary mode analysis: a useful metabolic pathway analysis tool for characterizing cellular matabolism. *Applied Microbiology and Biotechnology*, 81:813–826.

Vempaty, N. R., Kumar, V., and Korf, R. E. (1991). Depth-First Search vs Best-First Search. In *Proceedings of the Ninth National Conference on Artificial Intelligence*.

Yu, S. and Lee, M. (2012). Bispectral analysis and genetic algorithm for congestive heart failure recognition based on heart rate variability. *Computers in Biology and Medicine*, 42:816–825.

Zou, L., Wang, Z., Wang, Y., and Hu, F. (2010). Combined prediction of transmembrane topology and signal peptide of β-barrel proteins: Using a Hidden Markov model and genetic algorithms. *Computers in Biology and Medicine*, 40:621–628.

Doctorado en Ingeniería
Mención Inteligencia Computacional, Señales y Sistemas

Título de la obra:

Desarrollo de algoritmos evolutivos para el descubrimiento de relaciones en datos metabólicos

Autor: Matias Fernando Gerard

Director: Diego Humberto Milone

Codirector: Georgina Silvia Stegmayer

Lugar: Santa Fe, Argentina

Palabras Claves:

Bioinformática,
Algoritmos evolutivos,
Estrategias de búsqueda,
Vías metabólicas.