Métodos Evolutivos

ENRIQUE ALBA[1], MANUEL LAGUNA[2] Y RAFAEL MARTÍ[3]

[1] Universidad de Málaga, Spain, eat@lcc.uma.es
[2] University of Colorado at Boulder, USA, laguna@colorado.edu
[3] Universitat de València, Spain, Rafael.marti@uv.es

Resumen

La existencia de una gran cantidad y variedad de problemas difíciles, que aparecen en la práctica y que necesitan ser resueltos de forma eficiente, ha impulsado el desarrollo de procedimientos eficientes para encontrar buenas soluciones. Estos métodos, en los que la rapidez del proceso es tan importante cómo la calidad de la solución obtenida, se denominan heurísticos o aproximados.

Los procedimientos metaheurísticos constituyen la nueva generación de método aproximados, y dan unas reglas o estrategias que guían la construcción o el diseño del algoritmo heurístico concreto que resolverá el problema dado. Aun así, estas reglas no son rígidas y tienen muchos grados de libertad, permitiendo, por un lado, el diseño de diferentes métodos basados en la misma metodología, pero, dejando por otro lado al investigador la libertad para tomar decisiones e iniciativas al diseñar el método.

Una de las familias de métodos que podemos encontrar dentro de los procedimientos metaheurísticos es la de los llamados algoritmos evolutivos. En este trabajo vamos a describir tres de estos métodos que han demostrado su efectividad en los últimos años: los algoritmos genéticos, la búsqueda dispersa y el re-encadenamiento de trayectorias.

1. Introducción

Los métodos evolutivos están basados en poblaciones de soluciones. A diferencia de los métodos clásicos de mejora basados en seguimiento de trayectorias, en cada iteración del algoritmo no se tiene una única solución sino un conjunto de éstas (véase Blue y Roli 2003). Estos métodos se basan en generar, seleccionar, combinar y reemplazar un conjunto de soluciones. Dado que mantienen y manipulan un conjunto en lugar de una única solución a lo largo de todo el proceso de búsqueda suelen presentar tiempos de computación sensiblemente más altos que los de otros metaheurísticos. Este hecho se puede ver agravado porque la "convergencia" de la población requiera de un gran número de iteraciones. Por ello se ha dedicado un gran esfuerzo a obtener métodos que sean más agresivos y logren obtener soluciones de calidad en un horizonte más cercano.

2. Algoritmos Genéticos

Los Algoritmos Genéticos (AG's) fueron introducidos por John Holland (1975) inspirándose en el proceso observado en la evolución natural de los seres vivos. Básicamente, los AG's imitan el proceso de evolución natural, el principal mecanismo que guía la aparición de estructuras orgánicas complejas y bien adaptadas. De forma

muy simplificada, la evolución es el resultado de las relaciones entre la *creación* de nueva información genética y los procesos de *evaluación+selección*.

Los biólogos han estudiado en profundidad los mecanismos de la evolución, y aunque quedan parcelas por entender, muchos aspectos están bastante explicados. De manera muy general podemos decir que en la evolución de los seres vivos el problema al que cada individuo se enfrenta cada día es la supervivencia.

Cada individuo en una población se ve afectado por el resto (compitiendo por recursos, emparejándose para procrear, huyendo de los depredadores, ...) y también por el entorno (disponibilidad de comida, clima, ...). Los individuos mejor adaptados son los que tienen mayores posibilidades de vivir más tiempo y reproducirse, generando así una progenie con su información genética (posiblemente modificada).

A nivel de los genes, el problema de la supervivencia es el de buscar aquellas adaptaciones beneficiosas en un medio hostil y cambiante. Debido en parte a la selección natural, cada especie gana una cierta cantidad de "conocimiento", el cual es incorporado a la información de sus cromosomas. En el transcurso de la evolución se generan poblaciones sucesivas con información genética de los individuos cuya adecuación es superior a la de la media. La naturaleza no determinista de la reproducción provoca una creación permanente de información genética nueva, y por tanto la aparición de distintos individuos.

Así pues, la evolución tiene lugar en los cromosomas, en donde está codificada la información del ser vivo. La información almacenada en el cromosoma varía de unas generaciones a otras. En el proceso de formación de un nuevo individuo, se combina la información cromosómica de los progenitores aunque la forma exacta en que se realiza es aún desconocida.

Aunque muchos aspectos están todavía por discernir, existen unos principios generales de la evolución biológica ampliamente aceptados por la comunidad científica. Algunos de estos son:

- La evolución opera en los cromosomas en lugar de en los individuos a los que representan.
- La selección natural es el proceso por el que los cromosomas con "buenas estructuras" se reproducen más a menudo que los demás.
- En el proceso de reproducción tiene lugar la evolución mediante la combinación de los cromosomas de los progenitores. Llamamos *recombinación* a este proceso en el que se forma el cromosoma del descendiente. También son de tener en cuenta las mutaciones que pueden alterar dichos códigos.
- La evolución biológica tiene una memoria implícita (no explícita) en el sentido de que en la formación de los cromosomas únicamente se considera la información del período anterior, pero la representación *n*-ploide (diploide en los humanos) permite almacenar información que resultó de interés en el pasado en un entorno ambiental distinto

Este modelo *neo-Darwiniano* de la evolución orgánica se refleja en la estructura de un algoritmo genético. Así, los algoritmos genéticos establecen una analogía entre el conjunto de soluciones de un problema y el conjunto de individuos de una población natural, codificando la información de cada solución en una cadena (vector binario) a modo de cromosoma. Este cromosoma (único o múltiple) forma junto con su aptitud un "individuo" sobre el que el algoritmo aplica sus operaciones. En palabras del propio Holland:

> *"Se pueden encontrar soluciones aproximadas a problemas de gran complejidad computacional mediante un proceso de evolución simulada"*

Los AG's están basados en integrar e implementar eficientemente dos ideas fundamentales: Las representaciones simples (genotipos, tales como los vectores binarios) de las soluciones del problema y la realización de transformaciones simples para modificar y mejorar estas representaciones.

Para llevar a la práctica el esquema anterior y concretarlo en un algoritmo, hay que especificar los siguientes elementos:

- Una **representación cromosómica (genotipo)**
- Una población inicial
- Una medida de **evaluación (*fitness* o adecuación)**
- Un **criterio de selección** / reemplazo de individuos
- Una o varias operaciones de **recombinación**
- Una o varias operaciones de **mutación**

La figura 1 muestra un esquema básico de un algoritmo genético. Podemos observar que se trata de un algoritmo estocástico en el que es posible distinguir las tres etapas clásicas (Wah y Chang, 1997): *generación de la muestra inicial*, *paso de optimización* y *comprobación de la condición de parada*.

```
t := 0;
inicializar [P(t)];
evaluar [P(t)];
mientras no terminar hacer
      P'(t) := selección_pareja [P(t)];
      P'(t) := recombinación [P'(t)];
      P'(t) := mutación [P'(t)];
      evaluar [P'(t)];
      P(t+1) : =selecc_entorno [P'(t) ∪ P(t)];
      t := t +1;
fin mientras
```

Figura 1. Algoritmo genético

En el algoritmo de la figura 1, *P(t)* denota una población de μ individuos en la generación *t*. El algoritmo genera una nueva población *P'(t)* de λ individuos aplicando a *P(t)* un conjunto de operadores de variación. Típicamente tenemos el caso mostrado en el que dichos operadores son la selección y recombinación de parejas junto con la mutación de los nuevos individuos generados. Los operadores de variación son en general no deterministas, viniendo por tanto su comportamiento regulado por reglas de transición probabilísticas.

Los AG's inician la búsqueda a partir de una población de soluciones posibles. La población inicial suele ser generada de forma aleatoria (restringiéndose a soluciones que sean factibles), aunque recientemente se están considerando diseños en los que se utiliza información sobre el problema para generar soluciones de cierta calidad.

La evaluación sirve para asociar un valor de calidad, calculado por la función objetivo $f(\vec{x}_k)$, para cada solución $\vec{x}_k$ representada por el individuo *k*-ésimo de *P'(t)*, *k*:1, ..., λ. Aunque se suele utilizar la *calidad* como medida de la bondad según el valor de la función objetivo, se puede añadir un factor de penalización para controlar la infactibilidad en algunos problemas. Este factor puede ser estático o ajustarse dinámicamente, lo cual produciría un efecto similar al de la *Oscilación Estratégica* en Tabu Search (Glover y Laguna, 1997):

*calidad = ValorObjetivoNormalizado - Penalización * MedidaInfactibilidad*

El proceso de selección en un GA consta de dos etapas: la primera decide quiénes compiten por la reproducción (emparejamiento) y la segunda decide cuáles de entre todos los individuos (nuevos y viejos) van a sobrevivir, simulando así el proceso de selección del entorno (o ambiental). Ambas etapas de la selección utilizan de alguna manera los valores de adecuación (*fitness*) calculados en la evaluación y asociados a cada individuo para guiar el proceso hacia soluciones mejores.

Los diferentes tipos de selección utilizan el valor de adecuación directamente (*fitness proportionate*) o bien la relación entre dichos valores (torneo estocástico, *ranking* lineal, etc.). El reemplazo es en la mayor parte de aplicaciones prácticas elitista, de forma que se asegura la conservación de las k mejores soluciones presentes en la población actual cuando se va a generar la siguiente población. En el reemplazo se distingue entre técnicas que usan la población actual de μ individuos más la nueva de λ individuos para generar la próxima población –algoritmos $(\mu+\lambda)$– y los que únicamente usan los λ nuevos individuos para reemplazar a los μ individuos antiguos –algoritmos (μ,λ)–.

En general, un algoritmo genético es un metaheurístico poblacional en donde se pueden mezclar las ideas de combinación de bloques de construcción básicos con la modificación de bloques parcialmente útiles, para avanzar hacia la región más prometedora de la búsqueda. Los operadores de recombinación se aplican con alta probabilidad y usualmente han utilizado uno o varios puntos de cruce en los individuos para intercambiar las porciones resultantes entre los dos padres. También existen variantes que consideran más de dos padres en el proceso de recombinación. Algunos de los operadores de recombinación mas utilizados son:

De un punto:	Se elige aleatoriamente un punto de ruptura en los padres y se intercambian sus bits.
De 2 puntos:	Se eligen dos puntos de ruptura al azar para intercambiar.
Uniforme:	En cada bit se elige al azar un padre para que contribuya con su bit al del hijo, mientras que el segundo hijo recibe el bit del otro padre.
PMX, SEX:	Son operadores más sofisticados fruto de mezclar y aleatorizar los anteriores.

Las mutaciones más comunes son las que modifican un bit aleatoriamente (*bit flip*) a baja probabilidad, o las que utilizan un ruido con distribución normal, en el caso de genotipos flotantes.

A este respecto, debemos decir que tradicionalmente se han manejado representaciones binarias en los individuos evolucionados, aunque las representaciones en punto flotante y otras más complejas rápidamente han configurado un panorama muy amplio, donde únicamente se reconoce la plantilla de búsqueda básica y se observan variados operadores dependientes de la representación. De hecho, muchos resultados actuales trabajan sobre permutaciones, genotipos de longitud variable, etc. Puede consultarse (Bäck, Fogel y Michalewicz, 1997) para un compendio de variantes, representaciones y aplicaciones de los algoritmos genéticos y evolutivos en general. De forma más resumida e incluso unificada, el trabajo de Bäck (1996) permite comprender los fundamentos matemáticos y similitudes entre los algoritmos genéticos y las estrategias de evolución y otras técnicas evolutivas.

Dado que el algoritmo genético opera con una población en cada iteración, se espera que el método converja de modo que al final del proceso la población sea muy similar, y en el infinito se reduzca a un sólo individuo.

Se ha desarrollado una extensa teoría para estudiar la convergencia de estos algoritmos en el caso de vectores binarios. Esta teoría se basa principalmente en

considerar que un individuo binario es realmente un representante de una clase de equivalencia o ***esquema***, reinterpretando la búsqueda en lugar de entre vectores, entre esquemas. De este modo se concluye lo que se conoce como **paralelismo intrínseco**: *"En una población de m vectores binarios se están procesando implícitamente O(m^3) esquemas".* A partir de este resultado el *teorema de esquemas* prueba que la población converge a unos esquemas que cada vez son más parecidos, y en el límite a una única solución. En el caso de vectores no binarios se introducen los conceptos de *forma* y *conjunto de similitud* que generalizan al de esquema. Este dominio (teoría) está en fase de extensión, ya que las explicaciones actuales no suelen incorporar los detalles propios del espacio de búsqueda y por tanto no es posible dar teoremas de convergencia para problemas arbitrarios aún.

Como ocurre con los otros metaheurísticos, aún cuando existan explicaciones teóricas de funcionamiento, el problema es que en la práctica no se suelen respetar las condiciones necesarias para garantizar la convergencia del método, ya que son difíciles de seguir y probar. Por ello, nos encontramos con que, en ocasiones los algoritmos genéticos resuelven satisfactoriamente un problema de optimización dado y otras se quedan bastante alejados del óptimo.

Por último, debemos mencionar que entre las técnicas más prometedoras para el avance en estos metaheurísticos y su aplicación a problemas del mundo real, podemos destacar la inclusión de técnicas paralelas y descentralizadas, en las que la población se divide con algún criterio de entorno y se hace evolucionar de forma separada a zonas distintas dentro de la tradicional población única (Alba y Tomassini, 2002). Adicionalmente, la inclusión de técnicas híbridas, entendido esto como colaboración o incorporación de operadores inspirados en otros metaheurísticos es un campo que goza de amplios éxitos (Alba y Luque, 2003), de manera que cada vez es más difusa la separación entre los distintos algoritmos poblaciones, y mayor las interacciones con los algoritmos basados en trayectorias, que consideran un punto cada vez durante la búsqueda del óptimo.

3. Búsqueda Dispersa

La Búsqueda Dispersa (BD) es un método evolutivo que ha sido aplicado en la resolución de un gran número de problemas de optimización (Glover, Laguna y Martí, 2003). Los conceptos y principios fundamentales del método, fueron propuestos a comienzo de la década de los setenta, basados en las estrategias para combinar reglas de decisión, especialmente en problemas de secuenciación, así como en la combinación de restricciones (como el conocido método de las restricciones subrogadas). La BD se basa en el principio de que la información sobre la calidad o el atractivo de un conjunto de reglas, restricciones o soluciones puede ser utilizado mediante la combinación de éstas. En concreto, dadas dos soluciones, se puede obtener una nueva mediante su combinación de modo que mejore a las que la originaron.

Al igual que los algoritmos genéticos, la BD se basa en mantener un conjunto de soluciones y realizar combinaciones con éstas; pero a diferencia de éstos, no está fundamentado en la aleatorización sobre un conjunto relativamente grande de soluciones sino en las elecciones sistemáticas y estratégicas sobre un conjunto pequeño. Como ilustración basta decir que los algoritmos genéticos suelen considerar una población de 100 soluciones mientras que en la búsqueda dispersa es habitual trabajar con un conjunto de tan sólo 10 soluciones.

La primera descripción del método fue publicada en 1977 por Fred Glover donde establece los principios de la BD. En este primer artículo se determina que la BD realiza una exploración sistemática sobre una serie de "buenas" soluciones llamadas conjunto de referencia. Los siguientes comentarios resumen los principales aspectos de este trabajo:

- El método se centra en combinar dos o más soluciones del conjunto de referencia. La combinación de más de dos soluciones tiene como objetivo el generar centroides.
- Generar soluciones en la línea que unen dos dadas se considera una forma reducida del método.
- Al combinar se deben de seleccionar pesos apropiados y no tomar valores al azar.
- Se deben de realizar combinaciones "convexas" y "no convexas" de las soluciones.
- La distribución de los puntos se considera importante y deben de tomarse dispersos.

En Glover (1977) se introduce la combinación ponderada (*weighted combination*) como el mecanismo principal para generar nuevas soluciones. En esta versión se enfatizan las búsquedas lineales entre dos soluciones y el uso de pesos para muestrear en dicha línea. Asimismo, se introduce el concepto de combinar soluciones de calidad con soluciones diversas. Además, el método incluye una componente de intensificación que consiste en tomar una muestra mayor de la línea que ha producido mejores soluciones.

En este artículo el autor especifica que para trabajar con problemas con variables enteras, binarias o que forman una permutación, hay que diseñar métodos específicos de combinación (notar que no tiene sentido hablar de combinación lineal de dos permutaciones). Para ello se introducen los mecanismos de combinación basados en votos. En estos se definen reglas mediante las que cada solución "vota" para que sus características aparezcan en la solución que se está construyendo. Estos métodos de votos han sido muy utilizados en las rutinas de combinación de los algoritmos de BD y parece que constituyen uno de las claves del éxito de estos métodos.

En 1998, Glover publica una versión más específica del método en donde se recogen y simplifican muchas de las ideas expuestas en trabajos anteriores. Esta publicación tuvo un gran impacto en lo que a la difusión del método se refiere. Numerosos investigadores comenzaron a aplicar a partir de ese momento la BD a la resolución de problemas de optimización obteniendo resultados de gran calidad.

El método de BD se basa en combinar las soluciones que aparecen en el llamado conjunto de referencia (el cual es equivalente a la población en algoritmos genéticos). En este conjunto se tienen las soluciones buenas que se han ido encontrando. Es importante destacar que el significado de buena no se restringe a la calidad de la solución, sino que también se considera la diversidad que esta aporta al conjunto. La Búsqueda Dispersa consta básicamente de los siguientes elementos:

1. **Un generador de soluciones diversas**. El método se basa en generar un conjunto P de soluciones diversas (alrededor de 100), del que extraeremos un subconjunto pequeño (alrededor de b=10) con el que realizar las combinaciones y que denominamos:
2. **Un conjunto de referencia**. Extraído del conjunto de soluciones diversas según el criterio de contener soluciones de calidad y diferentes entre sí (Calidad y Diversidad). Si el método no logra mejorar a la solución, se considera que el output es la propia solución considerada. Las soluciones en este conjunto están ordenadas de mejor a peor respecto de su calidad.

   2.1. **Creación**. Iniciamos el conjunto de referencia con las $b/2$ mejores soluciones de P. Las $b/2$ restantes se extraen de P por el criterio de máxima distancia con las ya incluidas en el conjunto de referencia. Para ello debemos de definir previamente una función de distancia que generalmente depende del contexto del problema que se está resolviendo.

2.2. **Actualización**. Las soluciones fruto de las combinaciones pueden entrar en el conjunto de referencia y reemplazar a alguna de las ya incluidas si las mejoran. Así pues, el conjunto de referencia mantiene un tamaño *b* constante pero va mejorando a lo largo de la búsqueda. En implementaciones sencillas, la actualización de este conjunto se realiza únicamente por calidad, aunque podemos hacerlo también por diversidad. También en implementaciones más avanzadas es posible variar el tamaño del conjunto de referencia durante la búsqueda.

3. **Un método de combinación**. BD se basa en combinar todas las soluciones del conjunto de referencia. Para ello, se consideran subconjuntos de 2 o más elementos del conjunto de referencia y se combinan mediante una rutina diseñada a tal efecto. La solución o soluciones que se obtienen de esta combinación pueden ser inmediatamente introducidas en el conjunto de referencia (actualización dinámica) o almacenadas temporalmente en una lista hasta terminar de realizar todas las combinaciones y después ver qué soluciones entran en éste (actualización estática).
4. **Un método de mejora**. Típicamente se trata de un método de búsqueda local para mejorar las soluciones, tanto del conjunto de referencia como las combinadas antes de estudiar su inclusión en el conjunto de referencia.

El algoritmo de la figura 2 muestra cómo actúan los elementos descritos en un esquema básico del algoritmo. El algoritmo hace referencia a los subconjuntos de *R* ya que podemos combinar parejas, tríos o cualquier número de soluciones. Es usual limitar las combinaciones a parejas, por lo que el punto 6 equivaldría a decir: "Generar todas las parejas de soluciones de *R* en las que al menos una de las dos sea nueva"; donde por nueva entenderemos que haya entrado al conjunto después de realizar la última combinación de todo *R*.

1. *Comenzar con P = Ø. Utilizar el **método de generación** para construir una solución y el **método de mejora** para tratar de mejorarla; sea x la solución obtenida. Si $x \notin P$ entonces añadir x a P. (i.e., $P = P \cup x$), en otro caso, rechazar x. Repetir esta etapa hasta que P tenga un tamaño prefijado.*
2. *Construir el **conjunto de referencia** $R = \{ x^1, \ldots, x^b \}$ con las b/2 mejores soluciones de P y las b/2 soluciones de P más diversas a las ya incluidas.*
3. *Evaluar las soluciones en R y ordenarlas de mejor a peor respecto a la función objetivo.*
4. *Hacer NuevaSolución = TRUE*

***Mientras** (NuevaSolución)*

5. *NuevaSolucion = FALSE*
6. *Generar los subconjuntos de R en los que haya al menos una nueva solución.*
   Mientras (Queden subconjuntos sin examinar)
   7. *Seleccionar un subconjunto y etiquetarlo como examinado.*
   8. *Aplicar el **método de combinación** a las soluciones del subconjunto.*
   9. *Aplicar el **método de mejora** a cada solución obtenida por combinación. Sea x la solución mejorada:*
      Si($f(x) < f(x^b)$ y x no está en R)
   10. *Hacer $x^b = x$ y reordenar R*
   11. *Hacer NuevaSolucion = TRUE*

Figura 2. Algoritmo básico de la búsqueda dispersa

Notar que el algoritmo se detiene cuando al tratar de combinar vemos que no hay nuevos elementos en el conjunto de referencia (la variable *NuevaSolución* está a FALSE). Este algoritmo puede ser anidado en un esquema global que permita reconstruir el conjunto de referencia cuando éste ya ha sido utilizado. Así, si el límite de tiempo (o evaluaciones) no se ha excedido, una estrategia habitual es regenerar el conjunto de referencia dejando la mitad superior ($b/2$ mejores) y eliminando la mitad

inferior. Después, se genera un conjunto P como al comienzo del algoritmo, del que se extraen únicamente las $b/2$ soluciones más diversas con las ya existentes en R. De esta forma obtenemos un nuevo conjunto de referencia en el que mantenemos las soluciones de calidad y renovamos las debidas a diversidad. Después se volvería a combinar como anteriormente sobre este conjunto de referencia (pasos 5 a 11). De este modo se obtiene un esquema cíclico indefinido al que hay que añadirle una variable de control para detenerlo. Típicamente esta variable está en función del tiempo o del número de iteraciones (evaluaciones de la función objetivo).

Aunque los orígenes de la búsqueda dispersa fueron publicados hace casi 30 años, la metodología se puede considerar como relativamente reciente y en constante desarrollo. Durante los últimos años se han realizado nuevas contribuciones aplicando la búsqueda dispersa a la resolución tanto de conocidos problemas de optimización como de otros encontrados en la práctica. Algunas de estas aplicaciones han abierto nuevos campos de estudio, ofreciendo alternativas a los diseños conocidos. Laguna y Martí (2003) realizan una revisión exhaustiva del método, tanto de los elementos más utilizados y conocidos, como de los aspectos más novedosos y las últimas propuestas. Entre éstas podemos destacar:

I. **Aplicar la rutina de mejora de forma selectiva**. Las pruebas indican que el aplicar el método de mejora a todas las soluciones generadas y combinadas no garantiza obtener mejores resultados finales. Establecer umbrales de calidad para no aplicar la mejora a soluciones que difícilmente van a proporcionar la mejor solución ahorra un gasto innecesario de tiempo de computación. Por otro lado, al aplicar el método de mejora a todas las soluciones se acelera la convergencia de éste, lo cual pude ser deseable si disponemos de poco tiempo de computación, pero debemos de evitarlo si queremos ejecutar el método en un horizonte largo para obtener soluciones de gran calidad.
II. Es necesario estudiar el porcentaje de tiempo que el método está generando soluciones y el tiempo que está combinando. En esencia esta es la cuestión que se plantea en todos los métodos heurísticos: el **equilibrio entre la intensificación y la diversificación**.
III. Comparar las **actualizaciones** del conjunto de referencia **estática y dinámica**. Notar que al combinar las soluciones podemos aplicar dos estrategias, introducirlas en el conjunto nada más generarlas, si procede, o anotarlas en una "pila" y cuando terminemos de realizar todas las combinaciones, realizar la actualización. La primera estrategia es dinámica y más agresiva, en tanto que las soluciones buenas entran rápidamente en el conjunto de referencia, pero dado que éste es de tamaño constante, esto implica que hay soluciones que pueden salir sin llegar a haber sido utilizadas para realizar combinaciones.

4. Re-encadenamiento de Trayectorias

Como hemos mencionado, uno de los principales objetivos de cualquier método de búsqueda es crear un buen equilibrio o interacción entre lo que se denominan estrategias de intensificación y de diversificación. El método denominado re-encadenamiento de trayectorias (*Path-Relinking*) surgió como un proceso de integración de estas dos estrategias de búsqueda (Glover y Laguna, 1997). El proceso genera nuevos caminos entre soluciones que ya han sido previamente conectadas por una serie de movimientos durante un proceso de búsqueda.

El re-encadenamiento de trayectorias (RT) se basa en el hecho de que entre dos soluciones se puede trazar un camino que las una, de modo que las soluciones en dicho camino contengan atributos de ellas. Las soluciones originales pueden haber sido generadas mediante un método basado en una búsqueda local y estar unidas por un camino, o haber sido generadas por otro método y no estar unidas de ningún modo; en cualquier caso, se genera un nuevo camino que las una. Las características

de dicho camino vendrán especificadas respecto de los atributos que son añadidos o eliminados, o por los movimientos realizados para alcanzar una solución desde la otra. Esto constituye una extensión del concepto de combinación visto en las secciones anteriores en tanto que se obtienen varias soluciones a partir de dos o más originales.

La figura 3 ilustra el mecanismo de RT en el caso de dos soluciones: la solución A que denominamos solución inicial (*initiating solution*) y la solución B que denominamos solución guía (*guiding solution*). El diagrama muestra ambas soluciones unidas por dos caminos, uno en trazo continuo representando el camino original que describió el algoritmo de búsqueda local para obtener A y B, y otro en trazo discontinuo que representa el camino que ahora vamos a obtener aplicando RT. En el eje de abcisas representamos los movimientos que se van realizando, mientras que en el de ordenadas el valor de la función objetivo (suponemos que el problema es de minimizar).

Tal y como muestra la figura 3, la idea en la que se basa el método es que al realizar movimientos desde A hacia B, guiándonos únicamente por incorporar a A propiedades de B, obtengamos alguna solución intermedia que mejore en valor a A y a B.

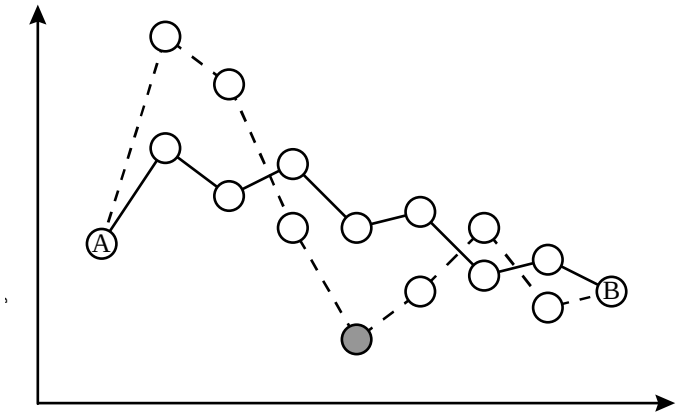

Figura 3. Esquema de Path relinking

Se puede considerar el método RT como una extensión del método de combinación de la búsqueda dispersa. En lugar de construir una nueva solución, combinando una o más soluciones iniciales directamente, RT genera un conjunto de soluciones en los caminos entre y fuera de las soluciones iniciales y guías seleccionadas.

El proceso se puede considerar como un caso extremo de una estrategia que trata de incorporar atributos de una solución de calidad en otra solución considerada. Sin embargo, en lugar de usar un criterio que simplemente motive la inclusión de tales atributos, RT subordina otras consideraciones con el fin de seleccionar los movimientos que introduzcan atributos como el criterio prioritario. Así, en cada paso se determina el mejor movimiento como aquel que incorpore la mayor cantidad, o el más destacado de los atributos de la solución guía.

Para generar los caminos es necesario seleccionar movimientos que cumplen los siguientes objetivos: empezando por una solución inicial, los movimientos deben introducir progresivamente los atributos de la solución guía (o reduciendo la distancia entre los atributos de la solución inicial y los de la solución guía). Los papeles de ambas soluciones son intercambiables; además, cada solución puede moverse hacia la otra como una manera de generar combinaciones. En el primer paso consideramos la creación de un camino que une dos soluciones seleccionadas x' y x'', produciendo una secuencia de soluciones $x'=x(1), x(2), \ldots, x(r)=x''$.

Puede que en el nuevo camino encontremos soluciones que no mejoren la calidad de las dos soluciones extremas consideradas, pero algunas de ellas pueden ser puntos de

partida para alcanzar otras soluciones que sí sean mejores que las originales. Por está razón es interesante y valioso examinar los entornos de las soluciones a lo largo del camino construido, ya que pueden proporcionar nuevos puntos de partida para lanzar una búsqueda adicional.

Si abordamos ahora el algoritmo RT en su conjunto hemos de considerar que tenemos una colección de soluciones previamente obtenidas a las que vamos a aplicar la estrategia descrita de generación de nuevos caminos. Para unificar la notación con la búsqueda dispersa, Martí y otros (2003) proponen el mismo término de ***conjunto de referencia*** (*RefSet*) para dicho conjunto con *b* soluciones. Desde este punto de vista, RT y BD se consideran métodos basados en poblaciones que operan en un conjunto de soluciones y básicamente se difieren en la manera de construir, manipular, actualizar y re-generar el *RefSet*.

En un diseño básico de la búsqueda dispersa, a todos los pares de soluciones en *RefSet* se les aplica el método de combinación. Análogamente, en una versión básica del RT todos los pares de soluciones son llamados a participar en la fase del re-encadenamiento. Para cada par de soluciones (x', x'') se pueden establecer dos camino; uno desde x' hacia x'' y el otro desde x'' hacia x'. La actualización del *RefSet* se basa en la mejora de la peor solución del conjunto, es decir, una nueva solución generada en un camino, se admite en el *RefSet* si mejora la peor solución en dicho conjunto. Si es así, re-emplaza a dicha solución (el tamaño del *RefSet*, al igual que ocurre en BD se considera fijo).

Muchos trabajos han demostrado la eficacia de aplicar algún método de búsqueda local desde algunas soluciones intermedias, generadas en los caminos, para explorar su entorno (Laguna y Martí, 1999; y Piñana y otros, 2001) Por otro lado, dos soluciones consecutivas obtenidas durante el proceso de re-encadenamiento son bastante similares y difieren solamente en los atributos que han sido introducidos por un solo movimiento. Por ello, generalmente no es eficiente aplicar el método de mejora en cada iteración del proceso, sobre todo si su aplicación es de coste considerable en términos del tiempo de ejecución. Una posibilidad es aplicar el método de mejora cada cierto número de iteraciones, o aplicarlo sólo a aquellas que cumplan un determinado criterio basado en su calidad (o calidad y diversidad respecto a las que ya se les aplicó dicha mejora).

El proceso de **re-encadenamiento simultáneo** empieza con dos puntos x' y x'' simultáneamente produciendo dos secuencias $x'=x'(1), x'(2), \ldots, x'(r)$ y $x''= x''(1), x''(2), \ldots, x''(s)$. Las elecciones en este caso son dirigidas con el fin de obtener $x'(r)=x''(s)$, para los valores finales de r y s. Éste se ve como un proceso en donde dos soluciones guías se intercambian dinámicamente hasta su convergencia a un solo punto.

El diseño básico se puede generalizar de diferentes formas. La primera consiste en generar caminos que vayan más allá de los extremos, lo que equivaldría a usar combinaciones no convexas. El problema consistiría en definir los atributos que guían el camino más allá de dichos extremos.

Otra posible generalización consiste en considerar **varias soluciones guías** de la manera siguiente. En lugar de realizar movimientos desde un punto x' hacia otro punto x'', se reemplaza este último con una colección de soluciones X''. Así, una vez alcanzada una solución $x(i)$, las opciones para determinar la siguiente solución $x(i+1)$ vienen dadas por la unión de las soluciones en X'', o más precisamente por la unión A'' de los atributos $A(x)$, para todos los x en X''. Esta variante ha sido utilizada con éxito en distintos trabajos.

5. Discusión Final

En este trabajo hemos presentado tres modelos metaheurísticos de optimización de gran impacto actual: algoritmos genéticos, búsqueda dispersa y re-encaenamiento de trayectorias. La diversidad de operaciones y problemas resueltos en este dominio es muy elevada, por lo que se han revisado únicamente los elementos canónicos de estos algoritmos y se ha diferido al lector interesado hacia referencias donde podrá encontrar un mayor nivel de detalle.

Las similitudes entre los métodos discutidos, tanto como sus diferencias, permitirán establecer mecanismos refinados de solución de problemas complejos en base a colaboraciones o extensiones de las ideas delineadas en este trabajo.

Referencias

Alba E., Tomassini M., *Parallelism and Evolutionary Algorithms*, IEEE Transactions on Evolutionary Computation, IEEE Press, 6(5):443-462, October 2002.

Alba E., Luque G., *Algoritmos Híbridos y Paralelos para la Resolución de Problemas Combinatorios*, Dpto. Informática-Univ. Oviedo (ed.), Actas del Segundo Congreso Español de Metaheurísticas y Algoritmos Evolutivos y Bioinspirados (MAEB'03) , Gijón, pp. 353-362, 2003

Bäck T.. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press. 1996.

Bäck T., Fogel D., Michalewicz Z. (eds.), *Handbook of Evolutionary Computation* (Oxford University Press). 1997.

Blue C., Roli A., *Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison*, ACM Computing Surveys 35(3), 268-308, 2003.

Campos V., Laguna M., Martí R. *Context-Independent Scatter and Tabu Search for Permutation Problems*, Technical report TR03-2001, Departamento de Estadística e I.O., Universidad de Valencia. 2001.

Campos V., Laguna M., Martí R., *Scatter Search for the Linear Ordering Problem*, New Ideas in Optimisation, D. Corne, M. Dorigo and F. Glover (Eds.), McGraw-Hill. 331-341. 1999.

Campos V., Glover F., Laguna M. , Martí R., *An Experimental Evaluation of a Scatter Search for the Linear Ordering Problem*, to appear in Journal of Global Optimization. 1999.

Corberán A., Fernández E., Laguna M., Martí R., *Heuristic Solutions to the Problem of Routing School Buses with Multiple Objectives*, Technical report TR08-2000, Departamento de Estadística e I.O., Universidad de Valencia. 2000.

Glover, F., *A Template for Scatter Search and Path Relinking*, in Artificial Evolution, Lecture Notes in Computer Science 1363, J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer and D. Snyers (Eds.), Springer-Verlag, pp. 13-54. 1998.

Glover F. , Laguna M. , *Tabu Search*, Kluwer Academic Publishers, Boston. 1997.

Glover F., Laguna M., Martí R., *Fundamentals of Scatter Search and Path Relinking*, Control and Cybernetics, 29 (3), 653-684. 2000.

Glover, F., Laguna M., Martí R., *Scatter Search*, to appear in Theory and Applications of Evolutionary Computation: Recent Trends, A. Ghosh and S. Tsutsui (Eds.), Springer-Verlag. 1999.

Glover, F., Laguna M., Martí R., *Scatter Search and Path Relinking: Advances and Applications*, Handbook of MetaHeuristics**,** 1-36, F. Glover and G. Kochenberger (Eds.), Kluwer, 2003.

Glover F., Tabu search for non-linear and parametric optimisation (with links to genetic algorithms), Discrete Applied Mathematics 49, 231-255. 1994.

Holland J., Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor. 1975.

Laguna M., Armentano V., *Lessons from Applying and Experimenting with Scatter Search*, to appear in *Adaptive Memory and Evolution: Tabu Search and Scatter Search*, C. Rego and B. Alidaee (eds.), Kluwer Academic Publishers. 2004.

Laguna M., Martí R., *Experimental Testing of Advanced Scatter Search Designs for Global Optimization of Multimodal Functions*. Technical report TR11-2000, Departamento de Estadística e I.O., Universidad de Valencia. 2000.

Laguna M., Martí R., *The OptQuest Callable Library* to appear in Optimization Software Class Libraries, Voss and Woodruff (Eds.), Kluwer., 2000.

Laguna M. Martí R., Scatter Search: Methodology and Implementations in C, Kluwer Academic Publishers. 2003.

Piñana E., Plana I., Campos V., Martí, R., *GRASP and Path relinking for the matrix bandwidth minimization*, European Journal of Operational Research 153, pp. 200-210, 2004.

Ugray Z., Lasdon L., Plummer J., Glover F., Kelly J., Martí R., *A Multistart Scatter Search Heuristic for Smooth NLP and MINLP Problems*, Technical report, University of Texas at Austin. 2001

Wah B. W., Chang Y-J. *Trace-Based Methods for Solving Nonlinear Global Optimization and Satisfiability Problems*. Journal of Global Optimization, Kluwer Academic Press, 10(2), 107-141. 1997.