La Búsqueda Local como Proveedora de Bloques Constructivos en Algoritmos Meméticos Auto-Generados

N.Krasnogor S.Gustafson

Resumen— **En este trabajo implementamos un algoritmo memético "auto-generado" para el problem del alineamiento máximo de mapas de contacto (MAX-CMO). Demostramos como la optimización de soluciones puede ser llevada a cabo al mismo tiempo en que se buscan estrategias de búsqueda local adecuadas a la instancia en cuestión. Las estrategias de búsqueda así producidas funcionan como proveedoras de bloques constructivos para el algoritmo evolutivo subyacente.**

Palabras clave— **Algoritmos meméticos, autoreconfiguración de programas, bioinformática.**

I. Introducción

Un vasto número de aplicaciones de algoritmos meméticos han sido reportadas en la literatura en los últimos años para una gran variedad de problemas. La mayoría de los trabajos en donde se reportan algoritmos meméticos exitosos muestran claramente que estos son obtenidos en base a una combinación adecuada de buscadores locales **pre-existentes** muy especializados y operadores genéticos adecuados al problema que se intenta resolver. Además, estos algoritmos requieren un considerable esfuerzo para el correcto ajuste de la búsqueda local y de las componentes evolutivas.

En trabajos previos hemos demostrado que, dada una colección de estrategias de búsqueda local alternativas que el algoritmo (multi)memético puede seleccionar para su uso, la selección optima de cual usar depende no solo de la instancia que se esta resolviendo sino también del momento particular del proceso de búsqueda en el cual se toma la decision. También pudimos mostrar que es posible producir algoritmos meméticos que se adaptan "al vuelo" a condiciones dinámicas en una variedad de dominios distintos[1],[2],[3],[4] and [5].

En [5] y [6] propusimos las llamadas "Metaheurísticas Auto-Generadas", siendo los algoritmos meméticos auto-generados(AMAGs) una instancia particular de las mismas. En los AMAGs, al contrario de los algoritmos meméticos tradicionales donde existe solo un operador de búsqueda local o de los algoritmos meméticos auto-adaptativos como en [1] o [4] donde existe una colección **fija** de búsquedas locales alternativas de donde elegir, las estrategias de búsqueda local son **creadas** de manera coevolutiva con las soluciones en donde esas estrategias serán aplicadas.

Automated Scheduling, Optimization and Planning Group, School of Computer Science and IT, University of Nottingham,United Kingdom. E-mail: {nxk,smg}@cs.nott.ac.uk

En los AMAGs ocurren dos procesos evolutivos simultáneos. Uno de ellos tiene lugar a nivel de cromosomas como en cualquier otro algoritmo evolutivo; los cromosomas y genes representan soluciones y características del problema que uno intenta solucionar. Por otro lado, el segundo proceso evolutivo ocurre al nivel de "memes", esto es, al nivel de los comportamientos que los individuos/agentes usaran para alterar sus chances de sobrevivir a la siguiente generación. Dado que los memes, que en nuestro caso representan estrategias de búsqueda local, se propagan, mutan y seleccionan en un sentido Darwinista, los algoritmos meméticos auto-generados que proponemos están mas cerca del concepto de memes introducido por R.Dawkins [7] que los algoritmos y trabajos previos sobre el tema, como por ejemplo, [8], [9], [10], [11], [12].

En [5],[6],[13],[14] se propuso y demostró que el concepto de AMAGs puede ser implementado y que, al menos para los dominios considerados en esos trabajos, era posible obtener beneficios de performance.

En el contexto de AMAGs los memes especifican conjuntos de reglas, programas, heurísticas, estrategias, comportamientos u operadores de movimiento que los individuos/agentes de la población pueden usar con el objetivo de mejorar su calidad (función objetivo). La interacción de genes y memes es indirecta y esta mediada por el vehículo de ambos: individuos (en otros contextos también son llamados agentes).

L.M.Gabora en [15] menciona tres fenómenos que son únicos a la evolución cultural (esto es, memetica). Estos son *bases de conocimiento*, *imitación* y *simulación mental*. Gabora dice:

The first is Knowledge-based operators: brains detect regularity and build schemas that

they use to adapt the mental equivalents of mutation and recombination to their meme substrate. The second is imitation: ideas spread when members of a society observe and copy one another. This is seen in both animal and human societies. Imitation enables individuals to share complete or partial solutions to the problems they face. The third is mental simulation: individuals can imagine what would happen if a meme were implemented before resources are spent on it. This provides them with a rudimentary form of selection before the phenotypic expression of a meme.

Estos tres fenómenos son capturados e implementados dentro de un AMAG y en virtud de ellos es que es posible crear y descubrir estrategias de búsqueda local útiles.

La representación de los operadores de bajo nivel (en este trabajo los buscadores locales) incluye características como la estrategia de aceptación (por ejemplo aceptación aleatoria, aceptación fuzzy, greedy, etc), el número máximo de vecinos a explorar, el número de iteraciones que la estrategia será utilizada, una función de valuación que le dirá a la estrategia si es o no razonable ejecutarse en una determinada solución o en un determinado momento de la búsqueda, y aún más importante, el operador de vecindario en sí mismo (que será la base de la heurística)[5].

Tecnologías anteriores de algoritmos evolutivos, GRASP, recocido simulado, búsqueda tabu, etc, se concentraron en, por ejemplo, la auto-adaptación de las probabilidades con la cual diferentes operadores deben ser aplicados[16], el tamaño de la lista tabu[17], el tamaño de las poblaciones [18], la adaptación de los criterios de aspiracion[19], los puntos de sobrecruzamiento[20], las frecuencias e intensidades de las mutaciones[21], el operador de búsqueda local que debe ser usado [4], la intensidad de la busqueda[22], el balance de explotación y exploración [23], entre otros. Los trabajos recién mencionados pertenecen a lo que podría llamarse "auto-adaptación", que presupone en esencia la adaptación en base a presión selectiva de parámetros de los algoritmos existentes. La auto-generación sin embargo permitiría la "aumentación" de algoritmos con nuevas componentes que no necesariamente pueden ser conseguidas en base a una variación de parámetros.

El papel que juega la búsqueda local en los algoritmos meméticos y multimeméticos ha sido tradicionalmente asociada a un refinamiento de la búsqueda. Se asume que el aspecto evolutivo del algoritmo produce una búsqueda global (exploración) del espacio de búsqueda mientras que el buscador local explota una solución y trata de afinar la búsqueda en los alrededores de la misma.

El objetivo de este trabajo es demostrar que las estrategias de búsqueda pueden ser creadas por un proceso evolutivo para un problema combinatorio de teoría de grafos y, mas importante aún, sugerir un nuevo rol para la búsqueda local en algoritmos evolutivos en general y meméticos en particular: *la estrategia de búsqueda no como un refinador(explotador) sino como proveedora de bloques constructivos.* El lector interesado en un estudio detallado de algoritmos evolutivos selecto-recombinativos puede referirse a [24].

II. El Problema de la Superposición Máxima de Mapas de Contacto

En este trabajo exploramos la búsqueda local como proveedora de bloques constructivos en el contexto de un problema extraído de la biología computacional. Un *mapa de contacto* es un grafo no dirigido que da un representación concisa de la estructura 3-dimensional de una proteína. En estos grafos, cada residuo[1] es representado por un nodo y existe una arista entre dos nodos si los correspondientes residuos son vecinos en la estructura 3-dimensional de la proteína. Dos residuos se consideran vecino cuando su distancia es menor que cierto umbral pre-especificado. Un *alineamiento* entre dos mapas de contacto es una asignación de residuos en el primer mapa de contacto a residuos en el segundo mapa. Los residuos así alineados se consideran equivalentes. El valor de un alineamiento entre dos mapas de contacto es el número de contactos en el primer mapa cuyos residuos están alineados a residuos en el segundo mapa de tal forma que estos, a su vez, también están en contacto. Estos es, el número de ciclos no dirigidos de tamaño 4 que se forman mediante dos contactos (el uno proveniente del primer mapa y el otro del segundo) y dos aristas de alineación. Este número es llamado la *superposición* de los mapas de contacto y el objetivo es su maximización. La complejidad de MAX-CMO fue demostrada NP-Hard en [25] y mas tarde por una reducción de NK-Landscapes en [5]. En la figura 1 se muestran los mapas de contacto de dos proteínas y su respectivo alineamiento (los residuo equivalentes se identifican con las líneas verticales).

III. Algoritmos Meméticos Auto-Generados para el MAX-CMO

En un algoritmo genético para MAX-CMO[26] un cromosoma es representado por un vector

[1] El residuo es el elemento constitutivo básico de las proteínas.

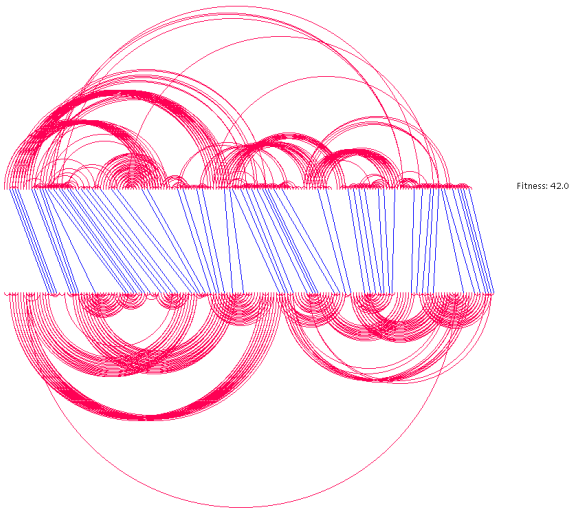

Fig. 1. Un alineamiento, también llamado superposición, candidato para las proteins 1aa9 y 1hnf. El valor de este alineamiento es 42.

$c \in [0, \ldots, m]^n$ donde m es el tamaño de la proteína mas larga y n el de la más pequeña. Una posición j en c, $c[j]$, especifica que el residuo j^{th} en la proteína mas larga esta alineado al residuo $c[j]^{th}$ de las más chica. Un valor de -1 en esa posición codifica que el residuo j no esta alineado a ningún otro residuo en la segunda proteína (llamado un "gap" en el alineamiento estructural). Configuraciones infactibles no son permitidas, esto es, si $i < j$ y $v[i] > v[j]$ o $i > j$ y $v[i] < v[j]$ (dos o mas alineamientos que se cruzan) entonces el cromosoma es desechado. Es simple definir operadores genéticos para este problema que preservan la factibilidad. El operador de sobrecruzamiento de dos puntos con chequeo de limites fue introducido en [26]. Aun cuando los dos padres representan soluciones factibles, los hijos podrían resultar en cromosomas inválidos. Una vez construido un hijo la factibilidad es restaurada al borrar cualquier alineamiento que se cruza con otro alineamiento.

El operador de mutación empleado en nuestros experimentos es llamado mutación de desplazamiento. Este operador selecciona una región consecutiva de un cromosoma y suma, desplazando a la derecha, o resta, desplazando a la izquierda, un número pequeño al contenido del vector que representa al alineamiento. El efecto fenotipico que esto produce es el "tilting" a derecha o izquierda del alineamiento. En [26] también se mencionan algunas variaciones sobre este operador.

En nuestro trabajo previo[2] empleamos un algoritmo multimemético que, además de usar la misma mutación y sobrecruzamiento ya mencionados, disponía de un conjunto de 6 buscadores locales diseñados a mano. Cuatro de los buscadores locales implementaban versiones parametrizadas del operador de desplazamiento. La dirección del desplazamiento, el factor de "tilting", etc se decidían al azar en cada iteración del proceso de búsqueda local. El tamaño de la ventana a desplazar tomaba valores en el conjunto $\{2, 4, 8, 16\}$. Además de estos, se definieron dos operadores nuevos, un operador de "wiper" y uno de "split" Los detalles de estos operadores pueden ser encontrados en [26],[5] y [2].

A. Descripción de los Memes

Como mencionamos en las secciones anteriores, intentamos producir una metaheurística que crea de cero la estrategia de búsqueda local adecuada a las circunstancias del momento. La reificación de las estrategias de búsqueda local se lleva a cabo como *memeplexes*[27].

Se define un memeplex como un complejo de memes co-adaptados. Un meme representa una manera particular de llevar a cabo la búsqueda local. Los memes pueden adaptarse mediante cambios en sus parámetros (auto-adaptación) o mediante cambios en las acciones que ellos ejecutan (auto-generación). La búsqueda local involucrada puede ser muy compleja y estar compuesta de varias fases y procesos. En el caso mas general queremos ser capaces de explorar el espacio de todos los memes (estrategias de búsqueda local) posibles. Uno puedo lograr esto usando una gramática que describe memeplexes y permitiendo que un sistema basado en programación genética (u otro methodo de síntesis automática de programas) descubra sentencias en el lenguaje generado por esta gramática. Las sentencias en el lenguaje así inducido representan estrategias de búsqueda complejas sintacticamente validas. Estas sentencias son las instrucciones usadas para implementar comportamientos de búsqueda local específicos. Por limitaciones de espacio no describiremos aquí la gramática utilizada. El lector interesado puedo referirse a [5] y [14].

En este trabajo, nos concentraremos solamente en la representación usada para evolucionar el operador de movimiento propiamente dicho en base a algunos ejemplos. En la figura III-A se pueden ver dos mapas de contacto listos para ser alineados por nuestro algoritmo. Para simplificar la exposición, los dos mapas son idénticos, esto es, estamos alineando un map de contacto consigo mismo. Este mapa tiene un patrón de contactos muy específico. En el ejemplo, con cierta probabilidad, un residuo esta conectado con su vecino mas cercano, con otro residuo que está a cuatro residuos de distancia o a ambos. En la Fig.

III-A(a) el mapa es de 10 residuos de longitud, mientras que en (b) tiene 50 residuos de longitud (pero manteniendo el patrón de conectividad).

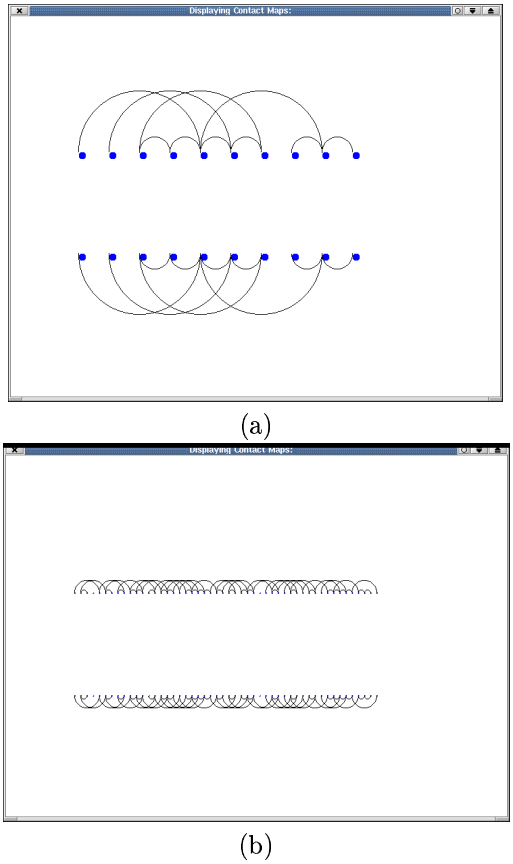

Fig. 2. Dos mapas de contacto. En (a) las dos proteínas generadas al azar tienen 10 residuos, mientras que en (b) el patrón de contactos se mantiene pero la proteína tiene 50 residuos.

Este patrón de contacto puede ser representado por la cadena $1-4$, cuyo significado es que el residuo que ocupa la posición i en la secuencia de la proteína esta en contacto en el estado nativo con los residuos $(i+1)$ y/o $(i+4)$. Esto es, el patrón $1-4$ es una representación sucinta de un bloque constructivo potencial que, si el buscador local es capaz de alinear, podría ser propagado luego por medio del operador de sobrecruzamiento a otras soluciones. Un operador de movimiento adecuado para la búsqueda local actuando en cualquiera de los mapas de contacto de las figuras III-A(a) y (b) sería uno que iterase sobre cada residuo en uno de los mapas de contacto verificando cuales de ellos en un mapa satisface los patrones de conectividad, y armase una lista de ellos. El mismo procedimiento de aplicaría al segundo mapa de contacto produciendo una segunda lista. La búsqueda local entonces emparejaría los residuos de la primera lista con los de la segunda produciendo así un nuevo y correcto alineamiento que incluiría el bloque constructivo capturado por la cadena mencionada mas arriba. En general los dos mapas a alinear tendran patrones de contactos distintos, es por ello que los memes se representan como $P_{m_1} \rightarrow P_{m_2}$. De esta forma P_{m_1} y P_{m_2} codifican una lista de posibles contactos en el primer y segundo mapa respectivamente.

El número de residuos que verifican el patrón en cada lista pone un limite superior a cuan costoso el operador de búsqueda local puede ser. Si el tamaño de la primera lista es L_1 y el de la segunda L_2 y sin perdida de generalidad asumimos que $L_1 \leq L_2$ entonces existen a lo sumo $\sum_{i=1}^{i=L_1} (L_2!)/((L_2-i)!)$. Claramente este número es demasiado grande para realizar una búsqueda exhaustiva, es por eso que la gramática que define los memeplexes permite la co-adaptación del tamaño del muestreo del vecindario. Por otro lado, aun cuando es conocido que proteínas de la vida real presentan estos tipos de patrones[28],[29], es imposible saber a priori cual de estos patrones proveerá de la mayor mejora en el valor de superposición para un par de proteínas en particular. Es por esto que AMAGs deben descubrir esto por ellos mismos.

Si los grafos a ser alineados fuesen distintos, entonces un operador de movimiento capaz de contemplar esta variación en los patrones de los dos mapas debe ser evolucionado. El operador de movimiento así definido induce un vecindario para cada alineamiento factible. Si un alineamiento s es representado como se explicó mas arriba y L_1, L_2 son las listas de vertices que verifican los patrones en el operador de movimiento, entonces cada solución factible que pueda ser obtenida al agregar a s uno o mas alineamientos entre vertices de L_1 y vertices de L_2 es un vecino de s. Los otros componentes del meme decidirán luego como muestrear este vecindario y que soluciones aceptar como la siguiente solución. Al ser este trabajo un raconto de las primeras investigaciones que realizamos con AMAGs, dejamos fijos varios aspectos de los memeplexes que bien podrían haberse dejado libre para su co-adaptación. Los memes empleados en este trabajo utilizan una estrategia golosa donde la primera mejora que se genera es aceptada y la búsqueda local acontece después del sobrecruzamiento. El muestreo que los memes realizan es 50 o 500 soluciones vecinas y esto se repite dos veces (dos iteraciones). Como describimos en la introducción, existen 3 procesos meméticos, imitación, innovación y simulación mental. Luego de la reproducción un nuevo individuo hereda los memes de uno de sus padres de acuerdo a un mecanismo simple de herencia[4]. Adicionalmente y con cierta probabilidad (llama-

da de imitación) un agente puede elegir ignorar el meme heredado de sus padres y copiar el meme de algún otro agente exitoso en la población. Para seleccionar de quien copiar los memes, se lleva a cabo un "torneo" entre individuos 4 individuos. El ganador del torneo actúa como modelo del nuevo agente y este último imita el comportamiento del modelo. La innovación es un proceso aleatorio de mutación de la especificación de un meme antes o después de $\rightarrow$.

Si durante 10 generaciones sucesivas no se produce ninguna mejora ni por el proceso evolutivo ni por la búsqueda local, entonces se lleva a cabo una etapa de simulación mental.

Durante la simulación mental cada agente (con cierta probabilidad) mutara intensivamente su meme, lo aplicara a su solución actual y si el meme mutado produce una mejora, ambos, el meme y la nueva solución serán mantenidos por el agente. Esto es, la simulación mental puede ser considerada como una optimización guiada en el espacio de memes. Si 10 ciclos consecutivos de simulaciones mentales terminan sin producir mejoras se finaliza el proceso de simulación y continua el algoritmo memético estándar.

IV. Descripción de los Experimentos

Diseñamos un generador de instancias aleatorias con el propósito de parametrizar la complejidad de los mapas de contactos a alinear. Los datos de entrada del generador random tienen la siguiente forma:

$r\ d\ n\ p_1\ pr_1\ p_2\ pr_2\ \ldots\ p_n\ pr_n$

donde r es el número de residuos que constituirán el mapa de contactos aleatorio, d es la densidad de aristas aleatorias (i.e. ruido) y n es el número de patrones existentes en el mapa de contacto. Para cada uno de los n patrones existen dos números, p_i y pr_i, donde p_i especifica que el residuo j esta conectado al residuo $j + p_i$ con probabilidad pr_i para todo $i \in [1, n]$. Esto es, cada patrón ocurre con cierta probabilidad en cada residuo, entonces el número tope esperado de contactos en el mapa esta dado por $r * d + r * \sum_{i=1}^{i=n} pr_i \leq r * (n + d)$.

Para nuestros experimentos consideramos mapas de contactos entre 10 y 250 residuos de longitud $r \in \{10, 50, 100, 150, 200, 250\}$. Además, se utilizaron $d = 0.01$ y $n \in \{1, 2, 3, 4\}$.

Para cada longitud, se utilizó cada número de patrones posible, dando lugar a 24 pares de valores (r, n) . Para cada par, 5 instancias aleatorias fueron generadas cubriendo un rango de baja y alta densidad[2]. Un total de 120 instancias fueron generadas. De todos los apareamientos posibles de estos 120 mapas de contacto elegimos al azar 96 pares para ser alineados. Cada uno de los 96 pares fue alineado 10 veces.

Presentamos a continuación comparaciones de la performance de un algoritmo genético y un AMAG. En este experimento queremos elucidar si es posible amortizar la sobrecarga del aprendizaje de estrategias de búsqueda local durante la operación del algoritmo memético o si la sobrecarga es detrimental y hace inservible todo el enfoque sugerido en este trabajo. Para ejecutar los experimentos implementamos un algoritmo genético (AG) tal cual fue descrito mas arriba. Fuimos capaces de reproducir los resultados de [26] y [2] y por lo tanto consideramos nuestra implementación equivalente a aquellas otras. La diferencia en comportamiento entre el AG y el AMAG se describe a continuación.

En las gráficas IV,IV,IV y IV comparamos el valor de los alineamientos obtenidos[3] en función del *first hitting times*. First hitting time (FHT) es el momento en número de evaluaciones de energía que el mejor valor producido por una ejecución es obtenido. Cada gráfico muestra resultados para 1,2,3, y 4 patrones respectivamente y para un rango de tamaños de mapas de contacto. Los parámetros del AG fueron 0.15 y 0.75 como probabilidades de mutación y sobrecruzamiento, $(50, 75)$ la estrategia de reemplazo. El AMAG usa 0.15,0.75,1.0,1.0,1.0,1.0 para las probabilidades de mutación, sobrecruzamiento, búsqueda local, imitación, simulación mental e innovación respectivamente. El algoritmo usa la misma estrategia de reemplazo y para ambas, búsqueda local y simulación mental, se utilizo la misma cantidad de tiempo de cpu, esto es, 50 muestreos.

Los gráficos en las Figuras IV,IV,IV y IV son buenos representantes de los resultados obtenidos con ambos algoritmos. Esto es, bajo una variedad de cambios en los parámetros mencionados, los resultados relativos se mantienen.

De las figuras IV,IV,IV y IV podemos ver que el AMAG produce resultados amortizados que son mucho mejores que los del AG. Esto es, si se le facilita al AMAG de suficiente tiempo, tarde o temprano descubrirá una búsqueda local apropiada que proveerá nuevos bloques constructivos que, a su vez ,permitirán obtener valores de alineamiento de un orden de magnitud mayor que el AG.

También se puede observar que la performance del AG no depende del tamaño de los mapas de

[2]El programa para generar los mapas de contactos aleatorios fue escrito en Java 1.1.8 y esta puede obtenerse de los autores por e-mail.

[3]Un valor de alineamiento mas grande significa un mejor alineamiento estructural

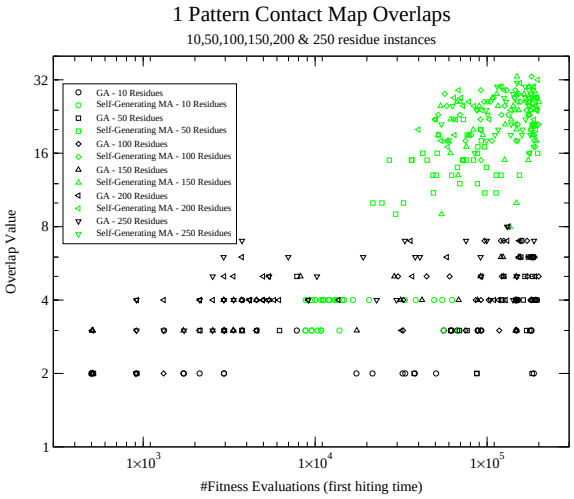

Fig. 3. Comparación de los "first hitting times" y la calidad de los overlaps obtenidos para el AG y el AMAG para instancias generadas al azar de complejidad creciente. La complejidad crece en función del número de residuos. Estos mapas de contacto tienen solo un patron.

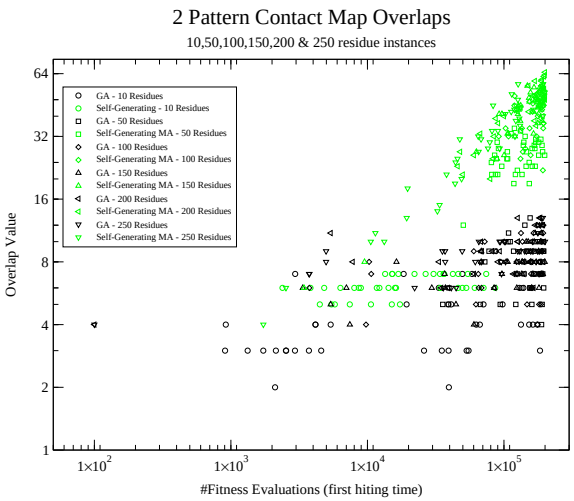

Fig. 4. Comparación de los "first hitting times" y la calidad de los overlaps obtenidos para el AG y el AMAG para instancias generadas al azar de complejidad creciente. La complejidad crece en función del número de residuos. Estos mapas de contacto tienen dos patrones.

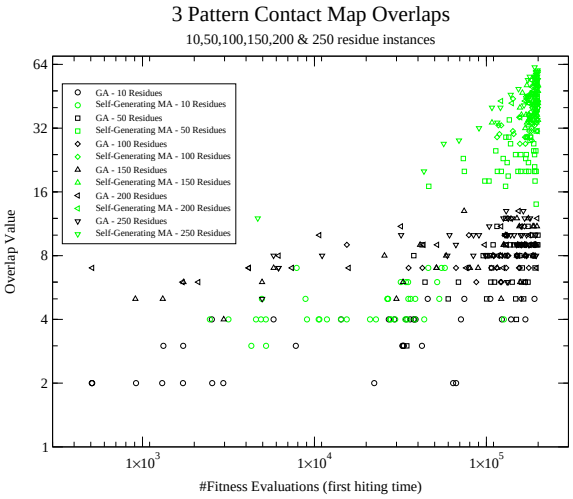

Fig. 5. Comparación de los "first hitting times" y la calidad de los overlaps obtenidos para el AG y el AMAG para instancias generadas al azar de complejidad creciente. La complejidad crece en función del número de residuos. Estos mapas de contacto tienen tres patrones.

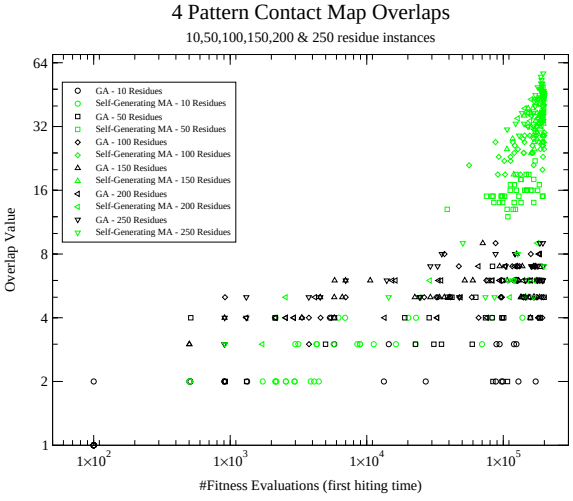

Fig. 6. Comparación de los "first hitting times" y la calidad de los overlaps obtenidos para el AG y el AMAG para instancias generadas al azar de complejidad creciente. La complejidad crece en función del número de residuos. Estos mapas de contacto tienen cuatro patrones.

contacto (número de residuos) ya que produce soluciones mediocres, aun cuando utilice el máximo número de evaluaciones de energía disponible (en estos experimentos $2 * 10^5$) para todo el rango de 10 a 250 residuos. El AG converge muy pronto en óptimos locales, esto se ve en los gráficos en las líneas paralelas al eje X para el rango de evaluaciones de la función objetivo para valores bajo de alineamientos. Por el contrario, ya que el AMAG continuamente mejora sus soluciones, no es hasta muy tarde en la ejecución del algoritmo (hacia la derecha del eje X), que la mejor solución es encontrada. en contraste al AG, el AMAG (como se espera) es sensible al número de residuos en los mapas de contacto, esto es, mapas de contacto mas grandes requieren un mayor número de evaluaciones para recibir una mejor solución. Esto se manifiesta en los gráficos en los patrones de aglomeraciones para los diferentes números de residuos.

Otro importante aspecto a notar es que ambos ejes están representados en escalas logarítmicas. Teniendo esto en consideración es evidente que la calidad de los alineamientos producidos por

el AMAG es mucho mejor que aquellos producidos por el AG. Para instancias suficientemente pequeñas (las de 10 residuos y algunas de 50) no es realmente favorable usar el AMAG ya que requiere mas esfuerzo del procesador para generar la misma calidad de alineamientos que el AG. Sin embargo, a medida que el número de residuos aumenta entonces las instancias se hacen lo suficientemente complejas como para favorecer la emergencia de estrategias de búsqueda locales favorables a tiempo para sobrepasar y mejorar los resultados del AG.

Es preciso señalar que a medida que el número de patrones presentes en los mapas de contacto a alinear aumenta, ambos algoritmos requieren mas esfuerzo para producir la mejor solución de cada corrida. Sin embargo, aun es posible ver que el AG es insensible al número de residuos mientras que el AMAG se aglomera en el costado superior derecho (en la figura IV). Esto indica que durante su ejecución el algoritmo progresa continuamente mejorando sus soluciones, la mejor de las cuales es encontrada cerca del final de la ejecución. Mas aun, este comportamiento indica que el AMAG no queda atrapado prematuramente en un óptimo local mediocre como le ocurre al AG.

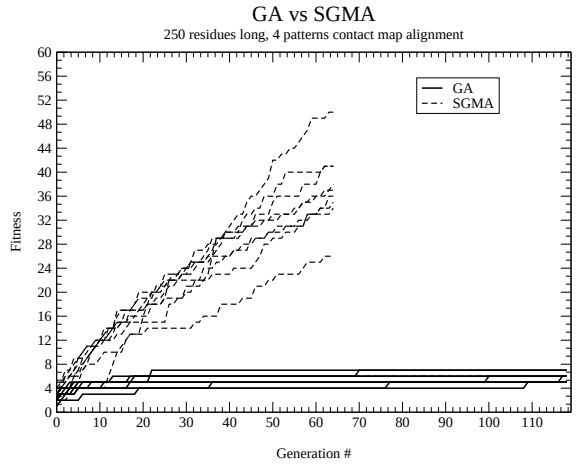

Fig. 7. Ejemplo representativo de 10 ejecucione del Ag y del AMAG para una instancia de 250 residuos con 4 patrones.

La habilidad del AMAG de evitar los óptimos locales proviene del hecho de que las estrategias de búsquedas locales evolucionadas introducen buenos bloques constructivos que se corresponden bien a cada instancia. La provisión de bloques constructivos es esencial para la operación sinergetica de los buscadores locales y los operadores genéticos. Esto es, usando la notación de Goldberg [24], tenemos que el *take over time* t^* es mayor que el *innovation time* t_i, que permite al algoritmo mejorar continuamente.

En la figura IV se comparan 10 corridas del AG contra 10 corridas del AMAG. Se puede observar que las corridas del AG quedan atrapadas muy temprano (alrededor de la generación 20) en un óptimo local pobre, mientras que el AMAG se mantiene mejorando durante toda la duración de la corrida. Todas las corridas en IV usaron el mismo número total de evaluaciones de la función objetivo.

V. Conclusiones y Trabajos Futuros

En este trabajo introdujimos el concepto de "Metaheurísticas Auto-Generadas" y ejemplificamos su uso en un problema combinatorio difícil de la teoría de grafos. La implementación de Metaheurísticas auto-generadas que empleamos esta basada en algoritmos meméticos.

A diferencia de la mayoría de los algoritmos meméticos, e híbridos en general, no recurrimos a diseñar a mano buscadores locales sino que le permitimos a las AMAG que descubra y ensamble "al vuelo" los buscadores locales que le resulten mas beneficiosos en cada circunstancia. Una estrategia similar puede ser usada en otras metaheurísticas como recocido simulada, búsqueda tabu, colonias de hormigas, GRASP, etc, donde implementaciones mas sofisticadas usando tal vez programación genética podría ser necesaria para co-evolucionar los operadores de búsqueda.

Una de las razones del éxito de estos algoritmos es que las estrategias de búsqueda local descubiertas al vuelo actúan como proveedoras de bloques constructivos de orden pequeño y medio. La provision continua de bloques constructivos asiste al proceso evolutivo a mejorar las soluciones continuamente al producir una operación mas sinergistica de los operadores genéticos y de búsqueda local como atestigua la figura 7. En [14] el lector interesado puede ver un estudio mas detallado de los temas tratados en este trabajo.

Agradecimientos

N.Krasnogor agrece a W.E. Hart y R.D. Carr por introducirlo al problema del alineamiento máximo de mapas de contactos.

Referencias

[1] N. Krasnogor and D.A. Pelta, "Fuzzy memes in multimeme algorithms: a fuzzy-evolutionary hybrid," in *Book chapter in "Fuzzy Sets based Heuristics for Optimization"*, J.L. Verdegay, Ed. 2002, Physica Verlag.

[2] B. Carr, W.E. Hart, N. Krasnogor, E.K. Burke, J.D. Hirst, and J.E. Smith, "Alignment of protein structures with a memetic evolutionary algorithm," in *GECCO-2002: Proceedings of the Genetic and Evolutionary Computation Conference*. 2002, Morgan Kaufman.

[3] N. Krasnogor, B.P. Blackburne, E.K. Burke, and J.D. Hirst, "Multimeme algorithms for protein structure prediction," in *Proceedings of the Parallel Problem Solving from Nature VII. Lecture notes in computer science*, 2002.

[4] N. Krasnogor and J.E. Smith, "Emergence of profitable search strategies based on a simple inheritance mechanism," in *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*. 2001, Morgan Kaufmann.

[5] N. Krasnogor, "http://www.cs.nott.ac.uk/~nxk/papers.html," in *Studies on the Theory and Design Space of Memetic Algorithms*. 2002, Ph.D. Thesis, University of the West of England, Bristol, United Kingdom.

[6] N. Krasnogor and S. Gustafson, "Toward truly "memetic" memetic algorithms: discussion and proof of concepts," in *Advances in Nature-Inspired Computation: The PPSN VII Workshops*, W. Hart J. Knowles N. Krasnogor R. Roy J.E. Smith D. Corne, G. Fogel and A. Tiwari, Eds. 2002, PEDAL (Parallel, Emergent and Distributed Architectures Lab). University of Reading. ISBN 0-9543481-0-9.

[7] R. Dawkins, *The Selfish Gene*, Oxford University Press, New York, 1976.

[8] P.M. França, A.S. Mendes, and P. Moscato, "Memetic algorithms to minimize tardiness on a single machine with sequence-dependent setup times," in *Proceedings of the 5th International Conference of the Decision Sciences Institute, Athens, Greece*, July 1999.

[9] P. Moscato, "Memetic algorithms: A short introduction," in *New Ideas in Optimization*, D. Corne, F. Glover, and M. Dorigo, Eds. McGraw-Hill, 1999.

[10] P.A. Moscato, "Problemas de otimizacão np, aproximabilidade e computacão evolutiva:da prática à teoria," *Ph.D Thesis, Universidade Estadual de Campinas,Brasil*, 2001.

[11] E.K. Burke, J.P. Newall, and R.F. Weare, "A memetic algorithm for university exam timetabling," in *The Practice and Theory of Automated Timetabling*, E.K. Burke and P. Ross, Eds., vol. 1153 of *Lecture Notes in Computer Science*, pp. 241–250. Springer Verlag, 1996.

[12] E.K. Burke and A.J. Smith, "A memetic algorithm for the maintenance scheduling problem," in *Proceedings of the ICONIP/ANZIIS/ANNES '97 Conference, Dunedin, New Zealand*. 24-28 November 1997, pp. 469–472, Springer.

[13] J.E. Smith, "The co-evolution of memetic algorithms for protein structure prediction," in *Advances in Nature-Inspired Computation: The PPSN VII Workshops*, W. Hart J. Knowles N. Krasnogor R. Roy J.E. Smith D. Corne, G. Fogel and A. Tiwari, Eds. 2002, PEDAL (Parallel, Emergent and Distributed Architectures Lab). University of Reading. ISBN 0-9543481-0-9.

[14] N. Krasnogor, "Self-generating metaheuristics in bioinformatics: The proteins structure comparison case," *To appear in the Journal of Genetic Programming and Evolvable Machines*, 2003.

[15] L.M. Gabora, "Meme and variations: A computational model of cultural evolution," in *1993 Lectures in Complex Systems*, Ed by L.Nadel and D.L. Stein, Eds., pp. 471–494. Addison Wesley, 1993.

[16] J.E. Smith and T.C. Fogarty, "Operator and parameter adaptation in genetic algorithms," *Soft Computing*, pp. 81–87, 1997.

[17] E. G. Talbi, Z. Hafidi, and J-M. Geib, "A parallel adaptive tabu search approach," *Parallel Computing*, vol. 24, no. 14, pp. 2003–2019, 1998.

[18] R.E. Smith and E. Smuda, "Adaptively resizing populations: Algorithms, analysis and first results," *Complex Systems*, vol. 1, no. 9, pp. 47–72, 1995.

[19] R. Battiti and G. Tecchiolli, "The reactive tabu search," *ORSA Journal on Computing*, vol. 6, no. 2, pp. 126–140, 1994.

[20] J. Schaffer and A. Morishima, "An adaptive crossover distribution mechanism for genetic algorithms," in *Proceedigns of Second International Conference on Genetic Algorithms*, J.J. Grefenstette, Ed. 1987, Lawrence Erlbaum.

[21] Jim Smith and T.C. Fogarty, "Self adaptation of mutation rates in a steady state genetic algorithm," in *Proceedings of the Third IEEE International Conference on Evolutionary Computing*. 1996, pp. 318–323, IEEE Press.

[22] M.W.S. Land, "Evolutionary algorithms with local search for combinatorial optimization," *Ph.D. Thesis, University of California, San Diego*, 1998.

[23] N. Krasnogor and J.E. Smith, "A memetic algorithm with self-adaptive local search: Tsp as a case study," in *Proceedings of the 2000 Genetic and Evolutionary Computation Conference*. 2000, Morgan Kaufmann.

[24] David E. Goldberg, *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, Kluwer Academic Publishers, 2002.

[25] D. Goldman, S. Istrail, and C. Papadimitriou, "Algorithmic aspects of protein structure similarity," *Proceedings of the 40th Annual Symposium on Foundations of Computer Sciences*, pp. 512–522, 1999.

[26] G. Lancia, R. Carr, B. Walenz, and S. Istrail, "101 optimal pdb structure alignments: a branch-and-cut algorithm for the maximum contact map overlap problem," *Proceedings of The Fifth Annual International Conference on Computational Molecular Biology, RECOMB 2001*, 2001.

[27] S. Blackmore, *The Meme Machine*, Oxford University Press, 1999.

[28] T. E. Creighton, Ed., *Protein Folding*, W. H. Freeman and Company, 1993.

[29] P.L. Privalov, "Physical basis of the stability of the folded conformations of proteins," in *Protein Folding*, T.E. Creighton, Ed. 1999, W.H. Freedman and Company.