Optimización en Entornos Geográficamente Distribuidos. Proyecto MALLBA

Enrique Alba y Carlos Cotta

Resumen— **En este artículo presentamos un resumen de los resultados más interesantes logrados en el seno del proyecto MALLBA. Este proyecto está encaminado a permitir la especificación y evaluación eficiente de motores de optimización exactos, heurísticos e híbridos para la resolución de problemas de optimización combinatoria. Presentamos la arquitectura de un sistema que cumple con estos requisitos y que además permite ejecución de manera secuencial o paralela, tanto en redes de área local (LAN) como en sistemas de área extensa (WAN). Además de la arquitectura, discutimos el diseño según el paradigma orientado a objetos y presentamos algunos resultados que demuestran el nivel de aplicabilidad y eficiencia del software resultante, tanto desde la perspectiva del paralelismo como desde la perspectiva del uso de algoritmos híbridos.**

Palabras clave— **Algoritmos Híbridos, Esqueletos Algorítmicos, Paralelismo LAN/WAN.**

I. Introducción

PUEDE afirmarse con gran certeza que uno de los motivos fundamentales por los que las técnicas modernas de optimización [1] (y en particular, las técnicas de computación evolutiva) no han alcanzado aún el nivel de popularidad y uso de otros enfoques más clásicos es la gran brecha existente entre el ámbito académico desde el que se promueven, y el ámbito práctico en el que deben ser empleadas. Más precisamente, nos encontramos ante el hecho de que muchos usuarios potenciales de este tipo de técnicas desisten de utilizarlas (y en muchos casos incluso de considerarlas) debido a la dificultad práctica que para ellos supone plasmar la solución a su problema mediante las mismas. Es en este escenario en el que se plantea la necesidad del Proyecto MALLBA.

El Proyecto MALLBA es una propuesta coordinada entre tres grupos de investigación radicados en Málaga, La Laguna (Tenerife) y Barcelona que –con el apoyo de la Comisión Interministerial de Ciencia y Tecnología (CICYT)– surge con el objetivo de proporcionar una biblioteca de esqueletos algorítmicos para optimización combinatoria. En esencia, podemos considerar que un esqueleto algorítmico es una plantilla que incorpora aspectos predeterminados (en nuestro caso, el núcleo del algoritmo de optimización), y que presenta ciertos *huecos* que el usuario final deberá rellenar de manera acorde con el problema que desee resolver. La filosofía de este enfoque es por lo tanto simplificar la labor del mencionado usuario final, proporcionándole la base del algoritmo y dejando que centre sus esfuerzos únicamente en detalles puntuales propios de su problema de interés. Esta labor de abstracción resulta especialmente interesante si tenemos en cuenta que a través de la misma el usuario podrá obtener algoritmos de optimización distribuidos (tanto en redes locales como en sistemas geográficamente distribuidos), sin necesidad de involucrarse en la problemática subyacente (comunicaciones, sincronización, etc.).

Departamento de Lenguajes y Ciencias de la Computación, E.T.S.I. Informática, Campus Teatinos, 29071 Málaga (España) {eat, ccottap}@lcc.uma.es

Este artículo presenta una panorámica general de la biblioteca de optimización MALLBA, tanto desde el punto de vista de su arquitectura interna, como del de los resultados computacionales que se están obteniendo. El artículo está organizado como sigue: en la Sección II se detallan los diferentes niveles de interfaz de la biblioteca, que dan lugar a otras tantas visiones de la misma. En la Sección III se procederá a describir la infraestructura física de la que se dispone en este proyecto, la cual está siendo utilizada para la extracción de resultados numéricos. Precisamente en este sentido, la Sección IV proporciona una descripción de los algoritmos que han sido implementados en el marco de este proyecto (Subsección IV-A), así como resultados numéricos, tanto desde el punto de vista de la gestión básica de la infraestructura de comunicaciones (Subsección IV-B), como del de la aplicación de los algoritmos implementados (Subsección IV-C). El articulo finaliza con un resumen de las conclusiones más importantes, junto con indicaciones de hacia dónde apunta el futuro desarrollo de la biblioteca (Sección V).

II. La Biblioteca MALLBA

La biblioteca MALLBA ofrece un conjunto de técnicas de resolución para problemas de optimización. Sus cualidades son eficiencia, reusabilidad, genericidad, portabilidad y facilidad de uso. Para lograrlas, MALLBA utiliza una novedosa aproximación a la programación genérica con objetos.

Una visión general de la arquitectura de la biblioteca se muestra en la Figura 1. Como puede apreciarse en la misma, existen tres niveles de interfaz que dan lugar a otras tantas visiones de la misma. A continuación se describirán estas diferentes visiones de la biblioteca.

A. Interfaz de Uso

Desde el punto de vista del uso de la biblioteca, resulta fundamental la separación entre dos conceptos: el problema que se pretende resolver y la técnica de resolución que se utilizará. La descripción del primero es una labor que claramente compete al usuario. Por su parte, la descripción de la técnica de resolución es proporcionada por la biblioteca. Tal como se adelantó anteriormente, la integración de estos dos elementos se realiza como si se rellenase un

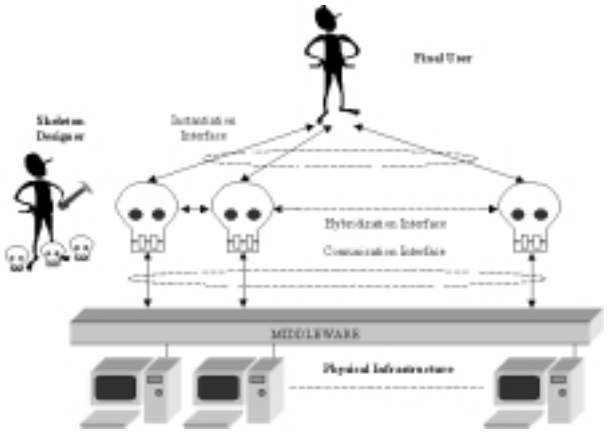

Fig. 1. Arquitectura de la biblioteca MALLBA.

"formulario": al seleccionar la técnica de resolución que se pretende emplear, la biblioteca proporciona una plantilla sobre la cual el usuario describe los elementos que componen el problema objetivo. El propio sistema MALLBA se encarga de unir ambas partes, proporcionando como resultado un programa adaptado al problema del usuario y que puede ser ejecutado en una sola máquina, en un entorno LAN o en una WAN.

Con la excepción de algunos ejemplos ilustrativos, MALLBA no aporta ningún código específico para solucionar problemas concretos. Por el contrario, provee código genérico que el usuario debe particularizar. De esta forma, una misma implementación –abstracta pero eficiente– se puede reutilizar en diversos contextos. El usuario no necesita tener conocimientos de paralelismo, puesto que éstos ya están incluidos dentro de la biblioteca. Más aún, MALLBA ofrece herramientas para ejecutar y monitorizar la ejecución en paralelo de los trabajos.

Todo el código MALLBA ha sido desarrollado en C++; para cada algoritmo de optimización se proporciona un conjunto de clases que –en función de su dependencia del problema objetivo– pueden agruparse en dos categorías: clases provistas y clases requeridas.

- **Clases Provistas**: Las clases englobadas dentro de esta categoría son las responsables de implementar toda la funcionalidad básica del algoritmo correspondiente. En primer lugar, la clase `Solver` encapsula el motor de optimización del algoritmo que se trate. Este motor de optimización es plenamente genérico, interactuando con el problema a través de las clases que el usuario debe proporcionar, y que serán descritas más adelante. En segundo lugar, existe la clase `SetupParams`, encargada de contener los parámetros propios de la ejecución del algoritmo, e.g., el número de iteraciones, el tamaño de la población en un algoritmo genético, el mecanismo de gestión de la cola de subproblemas en un algoritmo de ramificación y poda, etc. Otra clase provista es `Statistics`, cuya finalidad es la recolección de estadísticas propias del algoritmo empleado. Finalmente, existen las clases `StateVariable` y `StateCenter` cuya necesidad y finalidad será descrita posteriormente.

- **Clases Requeridas**. Estas clases son las responsables de proporcionar al esqueleto detalles sobre todos los aspectos dependientes del problema objetivo. No obstante, debe reseñarse que a pesar de esta dependencia del problema, la interfaz de las mismas es única y prefijada, permitiendo de esta manera el que las clases provistas puedan usarlas sin necesidad de relacionarse directamente con los detalles del problema. Entre las clases requeridas están la clase `Problem` (que debe proporcionar los métodos necesarios para manipular los datos propios del problema), la clase `Solution` (que encapsulará el manejo de soluciones al problema tratado), la clase `UserStatistics` (que permitirá al usuario recoger aquella información estadística de su interés que no estuviera siendo monitorizada por la clase provista `Statistics`) así como otras clases que dependen del esqueleto algorítmico elegido (por ejemplo, en el caso de un algoritmo de *Simulated Annealing* será necesario especificar el operador de movimiento empleado mediante una clase `Move`).

Así pues, el usuario de un esqueleto MALLBA se verá abocado a implementar las estructuras de datos dependientes del problema, así como de dotar de un comportamiento específico a todos los métodos incluidos en las interfaces de las clases requeridas.

B. Interfaz de Comunicaciones

El empleo de la biblioteca MALLBA en ambientes paralelos es uno de los aspectos fundamentales para la consecución de los objetivos del proyecto. Téngase en cuenta que las redes locales de ordenadores son un recurso económico y muy extendido hoy en día. Más aún, la expansión de Internet está facilitando la interconexión de estas redes locales, ofreciendo amplísimas posibilidades para la explotación de unos recursos que en la práctica se encuentran muy frecuentemente infrautilizados.

Con vistas a este objetivo es necesario disponer de un mecanismo de comunicación que permita ejecutar esqueletos algorítmicos tanto en red local como de área extensa. Puesto que dichos esqueletos se implementarán en un lenguaje de alto nivel de abstracción, es deseable que se desarrolle un sistema también de alto nivel que permita interaccionar a los componentes algorítmicos entre sí y al mismo tiempo gestionar un sistema de procesos paralelos.

El sistema resultante se denominará genéricamente *middleware*, y estará encargado de realizar las tareas del módulo básico de comunicaciones para este proyecto de investigación. Para su elaboración se han seguido varias etapas; inicialmente se han evaluado los sistemas existentes afines, para después continuar con una propuesta

de servicios y terminar con una implementación en C++ de este sistema *middleware*. A continuación describimos brevemente los resultados obtenidos en esta línea.

Inicialmente, hemos hecho en MALLBA una revisión detallada de las ventajas e inconvenientes de utilizar sistemas ya existentes, tanto basados en el paradigma de paso de mensajes como en sistemas para ejecución y gestión de objetos y programas distribuidos. En este sentido, hemos evaluado PVM, MPI, Java RMI, CORBA y Globus, así como algunas bibliotecas de comunicaciones más específicas [2]. Nuestra conclusión después de este análisis ha sido que necesitamos un sistema propio diseñado para satisfacer los requisitos de nuestro sistema de algoritmos, pero basado en un estándar eficiente y con visos de mantenerse vigente en el futuro. El resultado ha sido utilizar MPI como base para desarrollar una biblioteca de comunicaciones denominada `NetStream`. La necesidad de eficiencia en este proyecto es muy importante, y ha sido un factor determinante al tomar esta decisión; además, MPI (en sus dos implementaciones más conocidas, MPICH y LAM/MPI) tiene características de estándar de comunicaciones y se ha integrado en sistemas muy nuevos y prometedores como Globus.

Así pues, la biblioteca `NetStream` permite a los esqueletos intercambiar sus estructuras de datos de manera eficiente y manteniendo alto el nivel de abstracción y facilidad de uso. Para ello, se ha intentado minimizar el número de parámetros en los métodos resultantes y proveer un grupo nutrido de servicios. Los servicios disponibles pueden clasificarse en dos grupos: básicos y avanzados. Entre los servicios básicos incluimos:

- **Envío-recepción de tipos de datos primitivos:** `int`, `double`, `char`, cadenas de caracteres, etc. tanto en formato básico (*raw*) como de manera empaquetada, pensando en su eficiencia para la implementación de la fase WAN. El formato de uso es a través de flujos (*streams*) de entrada/salida por la red.
- **Servicios de sincronización:** barreras, difusión, chequeo de mensajes pendientes, etc. Todo ello se utiliza tal como en el manejo de flujos del estándar C++, es decir, usando *manipuladores*, i.e., métodos que varían el comportamiento de un flujo y que se alimentan al flujo como si datos se tratase.
- **Gestión básica de procesos paralelos:** consulta del identificador de proceso, número de procesos, establecimiento y consulta del proceso origen/destino de un flujo por defecto, etc.
- **Misceláneos:** arranque y parada del sistema de comunicación (usando métodos estáticos pertenecientes a la clase, no a una instancia concreta), etc.

Entre los servicios avanzados hemos implementado por el momento un sistema de *gestión de grupos de procesos* para que los esqueletos puedan formar "nichos" de optimización paralela. Los métodos creados permiten manipular comunicadores e intercomunicadores entre grupos distintos con el sentido que MPI les asigna a este concepto. Esta organización es de especial relevancia para algunos tipos de algoritmos distribuidos y también en el caso de diseño de algoritmos híbridos.

En el futuro se dotará a `NetStream` de nuevos métodos para trabajar con un modelo de los enlaces de comunicación y del estado de las máquinas que intervienen en la ejecución, para facilitar así la toma de decisiones de los algoritmos que se desarrollen en la fase WAN.

C. Interfaz de Hibridación

Empleado en diferentes contextos con diversos significados, el término *hibridación* ha de entenderse aquí como combinación de diferentes algoritmos de búsqueda (la denominada hibridación *débil* [3]). Tal como se ha demostrado tanto en teoría [4] como en la práctica [5], la hibridación es en un sentido amplio un mecanismo esencial para la consecución de algoritmos efectivos en la resolución de problemas específicos. Por ello, la disponibilidad de herramientas dentro de la biblioteca MALLBA para la construcción de tales algoritmos híbridos se ha considerado en todo momento una prioridad.

Debido a que los esqueletos algorítmicos serán reutilizados y combinados tanto por los usuarios de MALLBA como por los componentes de la red de investigación, es necesario que se especifique como parte integral de su dieño los puntos de interacción con ellos de manera estándar y uniforme. Para ello, hemos propuesto la incorporación de la noción de *estado* a cualquier esqueleto. El estado de un esqueleto es el punto de su interacción con el entorno; esto supone que a través del estado es posible inspeccionar la situación actual de la búsqueda tanto para trazar su evolución como para tomar decisiones sobre los siguientes pasos que debe llevar a cabo dicho esqueleto. En consecuencia, además de poder inspeccionar el esqueleto, es imprescindible que el estado pueda modificarse desde el entorno. Esto permitirá que un usuario u otro esqueleto controlen la dirección futura de búsqueda.

El estado, por tanto, permite analizar y dirigir el trabajo de los esqueletos, facilitando su estudio y también su hibridación con nuevos esqueletos. Esto se hace además independientemente de la implementación actual del esqueleto, ventaja importante en cualquier proyecto de envergadura. Básicamente, el estado debe permitir consultar variables críticas y estadísticas de funcionamiento; al mismo tiempo, debe permitir modificar en tiempo de ejecución los parámetros que dirigen la búsqueda, así como la memoria interna asociada a dicha búsqueda (e.g., insertar o extraer soluciones).

La utilización del concepto de estado garantiza la utilización futura de cualquier algoritmo por personas distintas a los que lo desarrollaron. El coste es mínimo, aunque la definición de un estado de manera

uniforme no es simple, y es un punto de investigación de interés actual. En este sentido, la propuesta realizada dentro de la biblioteca MALLBA se articula en torno a dos clases básicas ya mencionadas anteriormente: `StateVariable` y `StateCenter`. La primera permite definir y manipular cualesquiera elementos de información que se desee formen parte del estado del algoritmo. Así, `StateVariable` proporciona los mecanismos para asignar un nombre identificativo a tales elementos de información, así como para consultar y actualizar el valor de los mismos.

Todas las `StateVariable` se hallan inmersas en un `StateCenter`. Este último es el punto de conexión que permite el acceso a la información del estado de un algoritmo. Dicho acceso no se realiza mediante métodos específicos para cada variable de estado, sino que se articula a través de métodos genéricos de acceso por identificador, lo cual dota la interfaz de hibridación de una gran genericidad (no existe dependencia en este punto del problema tratado, o del algoritmo de optimización empleado) y flexibilidad (la gestión de las variables de estado es sumamente simple, permitiendo la fácil definición de las mismas, incluso de manera dinámica).

Sobre la base de estas clases para la manipulación de los estados internos, la construcción de esqueletos algorítmicos híbridos se facilita sobremanera, ya que simplemente se debe especificar el patrón de comportamiento a través de la manipulación adecuada de los estados de los algoritmos combinados. Como ejemplo de la flexibilidad de este modelo de hibridación puede reseñarse el hecho de que se han implementado metaalgoritmos híbridos débiles en los que simplemente es necesario indicar cuáles son los algoritmos básicos que intervendrán en el mismo; el patrón de comportamiento indicado sigue siendo válido en presencia de cualesquiera de dichos algoritmos básicos.

III. La Infraestructura MALLBA

La infraestructura del proyecto MALLBA está compuesta por computadores y redes de comunicación de las universidades de Málaga (UMA), La Laguna (ULL) y Barcelona (UPC). Las tres universidades están conectadas a través de RedIRIS, la red informática de ámbito académico y científico financiada por el Plan Nacional de la Ciencia y gestionada por el CSIC (Consejo Superior de Investigaciones Científicas) que une los principales centros universitarios y de investigación del estado español. RedIRIS es una red WAN con tecnología de conmutación ATM y accesos ATM de 34/155 Mbps. Hay un nodo en cada comunidad autónoma.

A. Barcelona

El grupo situado en la Universidad Politécnica de Cataluña se conecta directamente a la red troncal de la UPC (denominada XarxaUPC, con un ancho de banda máximo de 155 Mbps) a través de un enlace ATM de 155 Mbps desde un *router*. A este *router* se conectan el *switch* del edificio C5 con un enlace de fibra óptica y el *switch* del edificio C6 con un enlace a 10 Mbps. Al *switch* del C5 se conectan directamente las estaciones de trabajo de los usuarios, con enlaces a 10 Mbps. Al *switch* del C6 se conectan también estaciones de trabajo de usuarios, pero a 100 Mbps. También se conecta a este *switch* el *switch* MALLBA, donde están los computadores dedicados exclusivamente al proyecto. Los enlaces entre las máquinas y el *switch* son también a 100 Mbps.

B. La Laguna

El *cluster* de la Universidad de La Laguna esta formado por 14 máquinas iguales con nombres `beouwulfXX.csi.ull.es`, `XX` $\in \{\epsilon, 2, 3, \cdots, 13\}$. La máquina `beowulf.csi.ull.es` hace las veces de *front-end*, aunque la configuración no es un *cluster* protegido. Esta máquina es un Pentium II con 256 MB de memoria y 8 GB de disco duro. Las máquinas `beowulf2..5` disponen del mismo procesador pero con 48 MB de memoria y 1.2 GB de disco duro. No disponen de pantalla, ratón o tarjeta gráfica. Los `beowulf6..13` llevan procesadores AMD-K6 con 256 MB de memoria y 8 GB de disco duro y son estaciones de trabajo completas.

El sistema operativo es Linux con el *kernel* 2.2.12-20; `beowulf.csi.ull.es` es un servidor NFS del que el resto de máquinas son clientes. Entre los paquetes software instalados, se encuentra la versión 1.2.0 de MPICH.

C. Málaga

El grupo situado en la Universidad de Málaga dispone de un entorno heterogéneo de recursos computacionales formado por máquinas multiprocesador Digital AlphaServer, un cluster de estaciones de trabajo Sun UltraSparc I, un *cluster* de PCs con Pentium III y algunas estaciones de trabajo aisladas. Las redes utilizadas para interconectar estas máquinas son variadas: ATM a 155 Mbps, Fast Ethernet a 100 Mbps, tecnología *Memory Channel* para conectar con memoria compartida los 16 procesadores del multiprocesador Digital AlphaServer, y una red Myrinet que estará operativa en breve para transmisiones a muy alta velocidad (2+2 Gbps en enlaces full-dúplex). El software disponible permite satisfacer los requisitos de trabajo de MALLBA y consiste en varios tipos de compiladores de C, C++, Java, varios tipos de UNIX/Linux y entornos de desarrollo de aplicaciones y de producción científica.

IV. Resultados Computacionales

En esta sección presentamos resultados desde diferentes frentes de trabajo. En primer lugar describiremos la batería de algoritmos desarrollados, tanto exactos, como heurísticos y con especial atención a los híbridos. Seguidamente, presentaremos algunos resultados relativos a las comunicaciones, para discutir finalmente resultados numéricos concretos con los algoritmos desarrollados. Nuestra intención es proporcionar una visión tanto abstracta como también concreta y experimental de todo el trabajo realizado hasta el momento.

A. Algoritmos Desarrollados en MALLBA

El objetivo del trabajo es desarrollar algoritmos híbridos basados en otros algoritmos heurísticos y exactos que se hayan desarrollado previamente para dar lugar a técnicas más sofisticadas y eficientes. Esto supone que los algoritmos que van a colaborar de alguna manera atienden a una interfaz común de gestión de su estado.

Del lado de los algoritmos exactos, se han desarrollado en La Laguna algoritmos de ramificación y poda, divide y vencerás, y programación dinámica. Asimismo, el nodo de Barcelona ha dispuesto varios esqueletos algorítmicos heurísticos del tipo Búsqueda Tabú, Metrópolis y otros heurísticos en desarrollo actual. En nuestro caso, se han desarrollado los siguientes heurísticos híbridos:

- Un algoritmo de búsqueda local EP-$(1^+_,\lambda)$, dinámicamente reconfigurable y adaptado para su uso como algoritmo de búsqueda independiente/subordinado.
- Varios algoritmos concurrentes que permitieron estudiar las necesidades de paralelismo en el caso híbrido. Estos algoritmos combinan motores de optimización SA (recocido simulado) entre sí y varios EP entre sí, siguiendo un esquema denominado CLS -*Cooperative Local Search* (englobable dentro de la hibridación débil). La innovación adicional es la posibilidad de utilizar un conjunto de operadores diferente en cada uno de los algoritmos combinados, aumentando de esta manera la flexibilidad del híbrido resultante. Debe reseñarse que dada la homogeneidad de las interfaces algorítmicas alcanzada mediante el uso de los `StateCenter`, la reconfiguración del tipo de `solver` básico empleado en el modelo puede realizarse mediante la modificación de una declaración en el código fuente.
- Un algoritmo genético que utiliza al recocido simulado como operador para ser aplicado tras las tradicionales operaciones de cruce y mutación, atendiendo a una cierta probabilidad de uso. Esto supone un esquema de hibridación coercitiva, con implementaciones secuencial y en red local.
- Un algoritmo basado en el concepto de hibridación débil que ejecuta de manera contigua un GA hasta terminar, selecciona individuos de la población final y ejecuta sobre ellos un algoritmo SA. Se ha implementado tanto una versión secuencial que trabaja como se ha mencionado así como una versión distribuida en LAN que aplica estos dos pasos en cada una de las islas dispuestas en un anillo unidireccional con migración. Se han implementado varios tipos de algoritmos con este esquema cuya diferencia reside en el criterio de selección para elegir los individuos sobre los que aplicar el SA.

A continuación presentamos como ejemplo el diseño UML del algoritmo de recocido simulado (SA) que hemos desarrollado para ilustrar la importancia de investigar en el apartado software de los algoritmos que se diseñen, y no únicamente en su comportamiento algorítmico. Puede observarse en la Figura 2 el uso de clases para los objetos solución, problema y cómo todos ellos se reutilizan en el motor de optimización (`solver`). Este esqueleto es especialmente simple en su construcción y permite apreciar las ventajas de la reutilización de código que ya se ha hecho efectiva desde la clase `Solver_Seq` a `Solver_Lan` y que se está aprovechando para el algoritmo geográficamente distribuido con motor `Solver_Wan`. La filosofía es la de proporcionar servicios sin pensar en el uso que vaya a tener el esqueleto, es decir, pensar en un motor de optimización como un objeto de alto nivel de abstracción y complejidad, pero con los mismos requisitos que un objeto cualquiera.

Fig. 2. Diseño UML del algoritmo SA.

B. Modelado del Entorno de Trabajo Distribuido

Además de estudiar el diseño y verificar sobre problemas reales los esqueletos resultantes, existe una serie de resultados intermedios que dirigen el sentido de la investigación en el diseño de algoritmos en la WAN. Entre estos resultados se cuentan las medidas de conectividad entre los nodos MALLBA, la eficiencia de la bilioteca de comunicaciones desarrollada, la implementación de monitores e interfaces gráficos y estudios internos sobre *profiles* de trabajo para comprender mejor los esqueletos y la eficiencia de los métodos usados para su implementación orientada a objetos. En esta sección nos centraremos en los aspectos relativos a las comunicaciones.

Comencemos por los estudios de conectividad. Para optimizar la ejecución de la fase WAN durante el último año hemos estado monitorizando el rendimiento medio de los enlaces entre los tres grupos participantes en MALLBA. Para dar una idea de los valores manejados en la fase WAN presentamos en las tablas I y II los resultados de este estudio durante una semana típica respecto a tiempos de envío (en segundos).

Puede observarse el considerable tiempo de envío de paquetes dentro de la península, así como su incre-

TABLA I
TIEMPOS DE CONEXIÓN ENTRE MÁLAGA Y LA LAGUNA

Test Date	Min / Avg / Max	Packet Loss
09mar	61/203/429	53%
12mar	56/288/726	56%
14mar	51/182/568	26%
15mar	68/315/652	34%
16mar	54/245/595	44%
Avg. $\pm$ *Dev.*	246.6 ± 49.94	42.6 ± 11.31

TABLA II
TIEMPOS DE CONEXIÓN ENTRE MÁLAGA Y BARCELONA

Test Date	Min / Avg / Max	Packet Loss
09mar	27/45/625	31%
12mar	29/88/839	51%
14mar	77/121/732	40%
15mar	28/113/803	27%
16mar	79/124/767	28%
Avg. $\pm$ *Dev.*	98.2 ± 29.46	35.4 ± 9.05

mento cuando se trata de las comunicaciones con La Laguna. La considerable tasa de errores supone que se necesita usar un servicio orientado a la conexión confiable para las comunicaciones entre algoritmos que simplifique su implementación en la medida de lo posible.

Existe otro nivel de trabajo en relación a la WAN, consistente en los problemas derivados de conectar máquinas gestionadas por terceras personas. Esto supone que aspectos como el encaminamiento, los protocolos de ejecución remota y la existencia de mecanismos de seguridad deben considerarse seriamente como paso previo a cualquier uso conjunto de nodos distantes que vaya a hacerse.

Para terminar esta sección incluimos una evaluación del módulo de comunicaciones. Sólo presentamos el caso de envío de valores en punto flotante (*doubles*) por cuestiones de espacio y porque son representativos, ya que el resto de valores ofrece curvas similares o más eficientes. Hemos analizado tanto el envío de valores separados como empaquetados, una función que nos será de utilidad en los envíos WAN donde enviar valores simples puede producir una sobrecarga inaceptable en las comunicaciones.

Puede observarse que en el caso de valores individuales (Figura 3) la eficiencia es la misma que ofrece MPI, pero con la ventaja de la facilidad y abstracción disponible en `NetStream`. En el caso de empaquetar valores (Figura 4), apreciamos un aumento de tiempo usando `NetStream`, lógido debido al manejo de paquetes. Este descenso de eficiencia no es preocupante porque debemos considerar que la mayoría de paquetes tendrá un tamaño de 1 a 2 KB (mínima sobrecarga), tiempos además despreciables en términos WAN.

C. Algunos Resultados Numéricos

En esta sección presentamos los resultados de ejecutar versiones paralelas de algoritmos puros e híbridos siguiendo el esquema de esqueletos de MALLBA. Los algoritmos que presentamos son un algoritmo genético (GA), otro de recocido simulado

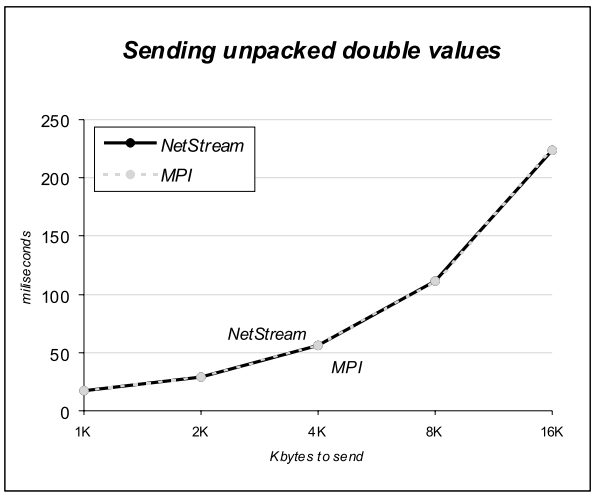

Fig. 3. Envío de *doubles* entre dos máquinas con MPI y con `NetStream`.

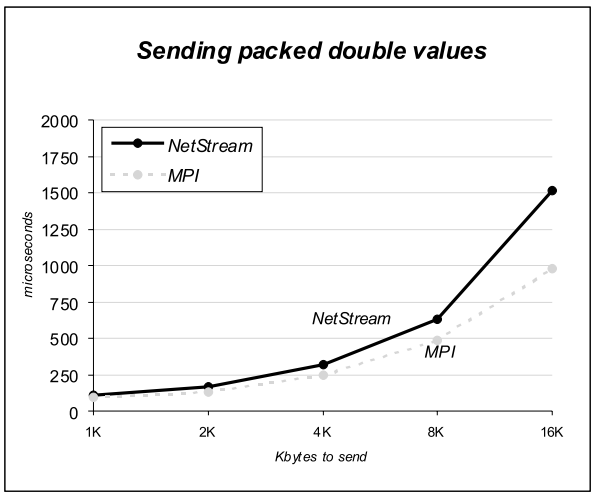

Fig. 4. Envío de *doubles* empaquetados entre dos máquinas con MPI y con `NetStream`.

(SA) y varias combinaciones paralelas de ellos. Los algoritmos han sido utilizados para resolver dos instancias del problema de corte máximo en grafos [6] y otras dos instancias del problema de la asignación de frecuencias a enlaces de radio [7].

En todos los casos los experimentos son una media de 50 ejecuciones independientes. Se ha analizado el comportamiento de los siguientes esqueletos:

- un GA paralelo,
- un SA paralelo sin cooperación entre subalgoritmos (SA1),
- un SA paralelo con cooperación (SA2),
- un algoritmo híbrido GASA1 donde se utiliza SA como operador de mejora en el ciclo reproductor del GA,
- un segundo algoritmo híbrido GASA2 donde se aplica el GA hasta terminar y después se ejecuta en cada sub-población un SA sobre cadenas según su adecuación usando torneo y
- un último híbrido similar al anterior pero usando una selección aleatoria del número de cadenas a las que se quiere aplicar SA (GASA3).

Para el problema del corte máximo en grafos se han resuelto dos instancias, una con 20 vértices y alta densidad de arcos (`cut20-0.9`) y otra instancia con 100 vértices (`cut100`). Pueden consultarse resultados de otros algoritmos sobre estas mismas instancias tanto en [6] como en [8]. Se trata de un problema cuyo objetivo es particionar un grafo en dos sub-conjuntos de manera que se maximice la suma de los pesos de los arcos que conectan vértices situados en sub-conjuntos distintos.

El GA usado codifica la solución en forma de cadena binaria $(x_1, x_2, \ldots, x_n)$ de longitud n, donde cada dígito corresponde a un vértice. Cada cadena codifica una partición del grafo completo. Un valor 1 indica que el vértice correspondiente estará en el conjunto V_1, y si está a 0 entonces dicho vértice pertenecerá al conjunto V_0. Por tanto, cada cadena en $\{0,1\}^n$ representa una partición de los vértices. La función que se desea maximizar es:

$$f(\vec{x}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} \cdot [x_i(1-x_j) + x_j(1-x_i)] \quad (1)$$

En el caso de la asignación de frecuencias podemos encontrar numerosos trabajos de referencia como [7]. Se trata de asignar a un elevado número de enlaces de radio el mínimo número de frecuencias posibles de entre un conjunto reducido de frecuencias disponibles. Este problema se encuentra en numerosas aplicaciones realeas debido a que tanto en telefonía móvil como en aplicaciones militares se divide la zona geográfica en celdas, cada una utilizando una frecuencia distinta de conexión respecto a las celdas vecinas para evitar las interferencias. Precisamente, debido a posibles interferencias, la asignación de frecuencias a enlaces radio no es libre, sino que deben respetarse restricciones de *lejanía* de las frecuencias en enlaces cercanos. Esto produce problemas de elevada complejidad, donde primero hemos debido atacar un sub-problema de obtención de soluciones factibles (primer orden) para pasar después a minimizar el número de frecuencias usadas (problema de segundo orden).

Existen numerosos bancos de pruebas internacionales; entre ellos, CELAR es una base de datos excepcionalmente popular en este dominio. Aquí hemos elegido las dos primeras instancias. CELAR01 debe asignar un número mínimo de frecuencias a 916 enlaces, y el resultado debe cumplir 5548 restricciones de no interferencia. La instancia CELAR02 debe asignar un número mínimo de frecuencias a 200 enlaces y cualquier solución debe cumplir cada una de las 1235 restricciones de no interferencia.

Para resolver el problema hemos codificado una posición por cada enlace, de manera que se almacene (indirectamente) la frecuencia asignada a este enlace. La función objetivo penaliza el número de restricciones violadas e incluye un término para minimizar el número de frecuencias utilizadas. Un estudio en detalle está fuera del ámbito de este trabajo, ya que nuestro objetivo es demostrar la validez de los esqueletos desarrollados, y en este problema debemos hacer numerosas consideraciones adicionales respecto a cómo implementar la función, qué información almacenar sobre cada solución para buscar en su vecindad soluciones alternativas, cómo manipular la violación de restricciones, etc.

Pasamos ahora a describir brevemente los parámetros utilizados para cada problema. En el caso del corte máximo, el esqueleto GA-(50,100) utiliza selección por torneo, cruce de dos puntos (probabilidad 0.8) y mutación por inversión (probabilidad 0.01). El esqueleto SA usa un factor de caída de temperatura de 0.99 y una longitud de cadena de Markov de 250. En las variantes GASA se aplica SA con probabilidad 0.1 y la cadena de Markov usada es de tamaño 10 durante 100 iteraciones. En el caso de cooperación paralela se utiliza un anillo con migración de una solución de manera asíncrona cada 200 iteraciones del algoritmo paralelo.

Para resolver el problema de la asignación de frecuencias se ha usado un EA-(20+40) con selección por torneo y mutación especializada para el problema (probabilidad 0.8). El resto de parámetros (incluyendo los que afectan a SA y GASA) son idénticos a los arriba mencionados, con la diferencia de que en GASA la probabilidad de aplicación de SA es de 0.01.

Presentamos resultados en todos los casos de versiones paralelas, usando 6 máquinas Pentium III a 700 MHz y 128 Mb de RAM con Linux, conectados por una red Fast Ethernet. Las Tablas III y IV contienen un resumen del funcionamiento de los algoritmos. En la Tabla III, I es el número de iteraciones/generaciones para encontrar el óptimo, t es el tiempo en segundos y F es el número de ejecuciones en las que se encuentra el óptimo de entre las 50 realizadas. En la Tabla IV, V representa el valor de adecuación y t el tiempo en segundos.

TABLA III
RESULTADOS DEL PROBLEMA MAXCUT.

Algoritmo	Cut20-0.9			Cut100		
	I	t	F	I	t	F
SA1	1742	0.117	46	4048	3.961	39
SA2	856	0.080	50	2660	2.669	10
GA	36	0.364	43	399	78.667	3
GASA1	5	0.505	50	54	104.548	18
GASA2	56	0.594	50	178	43.840	2
GASA3	71	0.739	47	171	33.650	4

TABLA IV
RESULTADOS DEL PROBLEMA DE ASIGNACIÓN DE FRECUENCIAS.

Algoritmo	CELAR01 (ópt.=16)		CELAR02 (ópt.=14)	
	V	t	V	t
SA1	18	23.00	14	1.39
SA2	20	16.48	14	1.32
GA	18	313.05	14	16.79
GASA1	18	100.74	14	2.94

Puede observarse en relación al problema del corte máximo que los resultados para el problema con 20 vértices son extremadamente eficientes (menos

de un segundo). Debido a su facilidad para estos algoritmos, de las 50 ejecuciones independientes, prácticamente todos ellos han localizado el óptimo. En relación al grafo con 100 vértices, podemos ya distinguir a los algoritmos por su tiempo de búsqueda y su porcentaje de localización del óptimo (Tabla III). El mejor resultado lo consigue SA1, un SA paralelo sin cooperación con apenas 4 segundos de cómputo y 39 de 50 localizaciones de la solución, seguido de la versión paralela con cooperación (SA2), más rápida pero a costa de reducir su eficacia hasta 10 de 50 ejecuciones con éxito. La versión paralela híbrida GASA1 obtiene un valor intermedio de 18 sobre 50, aunque a un muy elevado tiempo de búsqueda (104 segundos). Debemos además añadir que derivar todos estos algoritmos ha sido simple, permitiendo una rápida solución a un problema siguiendo la arquitectura MALLBA. El enfoque no ha sido únicamente resolver el problema, sino más bien demostrar la simplicidad de abordar tanto hibridación como paralelización usando MALLBA.

En lo que se refiere al problema de la asignación de frecuencias, véase en la Tabla IV los resultados. Compruebe que la instancia CELAR02 ha sido siempre resuelta con éxito (el número mínimo de frecuencias de una solución factible es 14) para todos los algoritmos. Los tiempos de búsqueda de las variantes paralelas SA1 y SA2 junto con el híbrido GASA han sido muy pequeños; el tiempo del GA puro ha sido un poco mayor. Estos tiempos de apenas 2 segundos son muy prometedores de cara al uso de algoritmos paralelos en este dominio. En el caso de la instancia CELAR01 podemos comprobar que su mayor dificultad requiere tiempos más elevados. El algoritmo más rápido es SA2, aunque no consigue alcanzar el óptimo de 16 frecuencias usadas. De hecho, el resto de algoritmos se quedan en 18, ligeramente mejor que SA2; el algoritmo más eficiente y efectivo es SA1, ya que llega a este valor en una media de 23 segundos, frente a los 313 ó 100 segundos que necesita GA o el híbrido GASA1. Para esta instancia, hemos obtenido el óptimo usando una búsqueda tabú, cuyos detalles no se incluyen por cuestiones de espacio.

V. Conclusiones y Trabajo Futuro

En este artículo hemos presentado los requisitos de un sistema encaminado a proporcionar un servicio de optimización útil en dominios de ingeniería. Dos importantes objetivos han sido permitir usar paralelismo internamente y proveer al usuario con técnicas modernas de solución de problemas. Para satisfacer los requisitos de facilidad de uso, independencia de la implementación y eficiencia se ha creado en el proyecto MALLBA un sistema de motores de optimización usando esqueletos de código implementados en C++.

Los requisitos de paralelismo se solventan mediante un servicio denominado `NetStream` basado actualmente en MPI. La interacción con el usuario se soluciona por la facilidad de uso de los esqueletos C++ y a través de interfaces gráficas para usuarios no expertos en las técnicas o en el paralelismo. Finalmente, el usuario avanzado con habilidades para el diseño de esqueletos híbridos dispone de una potente y flexible interfaz para realizar combinaciones algorítmicas.

El sistema pueda ejecutarse en secuencial, en un cluster de máquinas en LAN y su extensión a una red WAN para optimizar problemas que necesiten un elevado coste computacional. Entre las aportaciones de este trabajo podemos mencionar la implementación de varios motores de optimización modernos bajo una interfaz unificada, fácilmente instanciables con el problema del usuario, así como un nivel de eficiencia considerable, siendo estos dos requisitos un objetivo de compromiso difícil de conseguir con otras herramientas.

Hemos presentado resultados empíricos que demuestran que estos objetivos (facilidad, generalidad y eficiencia) pueden obtenerse a la vez en la misma herramienta, tanto para problemas tradicionales de optimización combinatoria como para problemas de aplicación directa en el mundo real. El trabajo futuro pasa por conseguir ejecuciones eficientes en la WAN para problemas de un elevado coste, así como por definir un modelo eficiente de ejecución en el nuevo medio WAN.

El software de MALLBA puede conseguirse a través de una petición de licencia de uso gratuita a cualquiera de los tres nodos involucrados.

Reconocimientos

Este trabajo está siendo realizado con el apoyo de la Comisión Interministerial de Ciencia y Tecnología (CICYT), a través del contrato TIC99-0754-C03-03.

Referencias

[1] C.R. Reeves, *Modern Heuristic Search Techniques for Combinatorial Problems*, Blackwell Scientific Publications, Oxford, 1993.

[2] E. Alba, C. Cotta, M. Daz, E. Soler, and J.M. Troya, "Mallba: Middleware for a geographically distributed optimization system," Tech. Rep., Dept. Lenguajes y Ciencias de la Computación, Universidad de Málaga (documento interno), 2000.

[3] C. Cotta and J.M. Troya, "On decision-making in strong hybrid evolutionary algorithms," in *Tasks and Methods in Applied Artificial Intelligence*, A.P. Del Pobil, J. Mira, and M. Ali, Eds., vol. 1416 of *Lecture Notes in Computer Science*, pp. 418–427. Springer-Verlag, Berlin Heidelberg, 1998.

[4] D.H. Wolpert and W.G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1(1), pp. 67–82, 1997.

[5] L. Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold Computer Library, New York, 1991.

[6] E. Alba and S. Khuri, "Applying evolutionary algorithms to combinatorial optimization problems," in *Proceedings of the International Conference on Computational Science (ICCS'01)*, Berlin Heidelberg, 2001, vol. 2074 (Part II) of *Lecture Notes in Computer Science*, pp. 689–700, Springer-Verlag.

[7] A. Kapsalis, V.J. Rayward-Smith, and G.D. Smith, "Using genetic algorithms to solve the radio link frequency assigment problem," in *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms*, D.W. Pearson, N.C. Steele, and R.F. Albretch, Eds. 1995, pp. 37–40, Springer-Verlag.

[8] S. Khuri, T. Bäck, and J. Heitkötter, "An evolutionary approach to combinatorial optimization problems," in *Proceedings of the 22nd ACM Computer Science Conference*, Phoenix, Arizona, 1994, pp. 66–73, ACM Press.