Paralelización de Heurísticas en Procesadores Multi-núcleo: Aplicación en Problemas de Enrutado de Vehículos

Raúl Baños[1], Julio Ortega, [1] Consolación Gil,[2] Francisco de Toro [3]

Resumen— **El problema del enrutado de vehículos y sus múltiples variantes tienen importantes implicaciones económicas, organizativas y medioambientales en los ámbitos de logística y transporte. Diferentes técnicas deterministas y estocásticas han sido propuestas para su resolución. Sin embargo, incluso en el caso de utilizar métodos aproximados, como es el caso de las heurísticas, el tiempo requerido para obtener soluciones de calidad en instancias complejas suele ser elevado, siendo necesario en dicho caso plantear alternativas, entre las que se encuentra el uso de estrategias de procesamiento paralelo. En este trabajo se analiza cómo mejorar el rendimiento de algoritmos heurísticos secuenciales haciendo uso de procesadores multi-núcleo. Los resultados obtenidos en la resolución de un conjunto de benchmarks asociados al problema de enrutado de vehículos con ventanas temporales muestran que el modelo de paralelización basado en islas haciendo uso de varios núcleos mejora significativamente el rendimiento de la versión secuencial ejecutada en un solo núcleo del procesador.**

Palabras clave— **Procesamiento Paralelo, Optimización Computacional, Heurísticas Paralelas, Enrutado de Vehículos, Logística, Transporte.**

I. Introducción

El problema del enrutado de vehículos (Vehicle Routing Problem, VRP) es un problema de optimización que tiene aplicaciones en diferentes ámbitos relacionados con la logística y el transporte. Diferentes autores han aplicado técnicas de optimización deterministas y estocásticas para su resolución [1], aunque dado que es un problema NP-duro [2], no es posible garantizar la obtención de la solución óptima en un tiempo acotado. Aunque existe un elevado número de contribuciones relacionadas con problemas de enrutado de vehículos, la gran mayoría aplica algoritmos de optimización secuenciales. Sin embargo, dada la complejidad de dicho problema, el uso de procesamiento paralelo constituye una herramienta interesante, si no la mejor, para obtener soluciones de calidad en tiempos de ejecución aceptables, ya que permite explorar de forma más rápida y exhaustiva el espacio de búsqueda. Indicar que, tradicionalmente, el procesamiento paralelo ha venido relacionado con el uso de equipos de altas prestaciones, como es el caso de los clusters de computadores o arquitecturas de memoria compartida que, aunque permiten un elevado grado de escalabilidad, requieren de un elevado presupuesto para su adquisición y mantenimiento. Sin embargo, en la actualidad se ha generalizado el uso de procesadores multi-núcleo en la práctica totalidad de computadores personales, lo que hace posible beneficiarse del procesamiento paralelo sin necesidad de acceder a equipos especializados de altas prestaciones. Pese a que, durante años, los investigadores han diseñado algoritmos computacionales paralelos, aun hoy día su aplicación en plataformas multi-núcleo constituye un área de investigación con muchas posibilidades.

La presente investigación describe cómo mejorar el rendimiento de algoritmos heurísticos de optimización para resolver problemas de optmización combinatoria haciendo uso de procesamiento paralelo en computadores con procesadores multi-núcleo. La Sección II ofrece una descripción de una variante del problema de enrutado de vehículos que añade restricciones de capacidad en los vehículos y ventanas temporales en los servicios a clientes. En la Sección III se presenta el algoritmo paralelo propuesto, mientras que los resultados obtenidos por el mismo en la resolución de un conjunto de benchmarks se presentan en la Sección IV. Finalmente, la Sección V ofrece las conclusiones y el trabajo futuro.

II. El Problema de Enrutado de Vehículos con Ventanas Temporales

El problema de enrutado de vehículos con ventanas temporales (Vehicle Routing Problem with Time Windows, VRPTW) [3] consiste en determinar el conjunto de rutas óptimas que debe seguir un determinado número de vehículos con idéntica capacidad estacionados en un almacén (centro logístico) que operan dentro de una franja temporal y que son utilizados para visitar y suministrar las mercancías a los clientes. Los vehículos siguen una ruta que parte del centro logístico y vuelven a el tras realizar el servicio. La cantidad de mercancías suministradas por cada vehículo no puede exceder su capacidad. Los clientes, cuyas demandas pueden ser suministradas únicamente por un solo vehículo, están localizados en posiciones geográficamente diversas y tienen requisitos pre-establecidos en lo que respecta a las mercancías requeridas y al tiempo de servicio necesario para la descarga. La ventana temporal de cada cliente implica que el vehículo que le suministra puede llegar antes de que la ventana temporal abra (esperando a su apertura), pero no después de que esta haya cerrado. La distancia total viajada por todos los vehículos definen el tiempo total de viaje. Señalar que existen formulaciones en las

[1]Dpto. de Arquitectura y Tecnología de Computadores, Univ. de Granada, e-mail:{rbanos,julio}@atc.ugr.es
[2]Dpto. de Arquitectura de Computadores y Electrónica, Univ. de Almería, e-mail:{cgilm}@ual.es
[3]Dpto. de Teoría de la Señal, Telemática y Comunicaciones, Univ. de Granada, e-mail:ftoro@ugr.es

que el proceso de optimización consta de dos fases: una primera fase en la que se minimiza el número de vehículos utilizados y una segunda fase en la que, a igualdad de vehículos, se reduce la distancia total recorrida [4]. Sin embargo, en el presente trabajo consideramos directamente la distancia total recorrida como el objetivo a minimizar ya que, al ser el combustible el coste variable más importante del proceso de transporte, la reducción de la distancia total recorrida implica reducir su consumo. Además, en la gran mayoría de aplicaciones prácticas, el uso de más o menos vehículos y conductores no afecta al coste operativo real, ya que los vehículos son propiedad de la empresa y los conductores son empleados de la misma. Se han propuesto diferentes métodos exactos para resolver el problema del enrutado de vehículos, incluyendo métodos basados en relajación, programación dinámica, etc. [3]. Sin embargo, dada la complejidad del problema, los métodos aproximados, como algoritmos evolutivos, búsqueda tabu, etc. resultan más convenientes ya que obtienen resultados aceptables en tiempos reducidos [5].

A. VRPTW: Formulación del Problema

La definición matemática de los términos y constantes del VRPTW, incluyendo las notaciones frecuentemente utilizadas para representar las rutas, almacenes, clientes, y vehículos puede ser modelada utilizando un grafo completo no dirigido $G=(V,E)$, donde los vértices $V=\{0,..,n\}$ corresponden al centro logístico donde se encuentran los vehículos y mercancías, y las aristas $e \in E\{(i,j) : i,j \in V, i \leq j\}$ a las rutas existentes entre ellos.

Variables de Decisión
X_{ij}^k es igual a 1 si el k-ésimo vehículo viaja desde el nodo i al nodo j, y 0 en caso contrario.

Parámetros
a_i es el tiempo más temprano en el que el cliente i permite inicializar el servicio;
b_i es el tiempo más tardío en el que el cliente i permite inicializar el servicio;
C_{ij} es la distancia (coste) necesario para viajar desde el nodo i al nodo j;
d_i es la demanda del cliente i;
K es el máximo número de vehículos que pueden ser utilizados;
N es el número de clientes más el almacén;
Q es la capacidad de carga de los vehículos;
q_{ij} es el flujo de bienes desde el nodo i al nodo j;
s_i es el tiempo de servicio del cliente i.

El VRPTW se define de la siguiente forma:

$$Minimizar$$

$$TD = \sum_{i=0}^{N} \sum_{j=0,j\neq i}^{N} \sum_{k=1}^{K} X_{ij}^k C_{ij} \quad (1)$$

$$\text{t.q.:} \begin{cases} X_{ij}^k \in \{0,1\}, \quad 1 \leq k \leq K, \quad 0 \leq i,j \leq N & (2) \\ \sum_{i=0}^{N} \sum_{j=0,j\neq i}^{N} X_{ij}^k d_i \leq Q, \quad 1 \leq k \leq K & (3) \\ \sum_{j=1}^{N} \sum_{k=1}^{K} X_{0j}^k \leq K & (4) \\ \sum_{j=1}^{N} \sum_{k=1}^{K} X_{0j}^k - \sum_{j=1}^{N} \sum_{k=1}^{K} X_{j0}^k = 0 & (5) \\ \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=1}^{K} X_{ij}^k = 1 & (6) \\ \sum_{i=0}^{N} \sum_{j=1}^{N} q_{ij} - \sum_{i=0}^{N} \sum_{j=1}^{N} q_{ji} = d_j & (7) \\ a_{ij} \leq s_{ik} \leq b_i, \forall i \in N, \forall k \in K & (8) \end{cases}$$

La ecuación (1) representa la distancia total de viaje recorrida por los vehículos para suministrar a los clientes, esto es, la función objetivo a optimizar. La restricción (2) indica que una ruta entre dos clientes puede o no ser cubierta por un vehículo; la restricción (3) indica que el límite de la capacidad que los vehículos no pueden acceder; la restricción (4) especifica que, como mucho, hay K rutas (vehículos) saliendo del centro logístico; la restricción (5) garantiza que los vehículos parten desde y vuelven hacia el centro logístico; la restricción (6) especifica que un cliente es visitado una sola vez por un solo vehículo; la restricción (7) indica las restricciones de flujo relacionadas con el envío de mercancías correspondiente al cliente j; la restricción (8) asegura que los clientes son visitados dentro de su ventana temporal.

B. Algoritmos Paralelos para Problemas de Enrutado de Vehículos

Entre las ventajas del procesamiento paralelo encontramos la reducción del tiempo de ejecución de las versiones secuenciales o la mejora de las mismas sin provocar un incremento del esfuerzo computacional. Los programas paralelos han venido siendo aplicados habitualmente en equipos de altas prestaciones, como clusters de computadores o máquinas de memoria compartida. Sin embargo, la introducción de procesadores multi-núcleo, que constan de varios núcleos de procesamiento y una caché de memoria compartida en un solo microchip, permite aplicar procesamiento paralelo incluso en computadores personales utilizando componentes software estándar.

El uso de procesamiento paralelo requiere compartir recursos y distribuir la carga de trabajo dinámicamente para utilizar los recursos globales de forma eficiente. Las librerías de paso de mensajes de alto nivel, como es el caso del Message Passing Interface (MPI), ofrecen rutinas que permiten escribir programas paralelos con los que configurar el entorno de forma eficiente, así como enviar y recibir paquetes de datos. Al usar este tipo de librerías, el programador tiene que desarrollar una cantidad de código importante para manejar algunas de las tareas de paralelización, como son descomponer y distribuir los puntos de búsqueda, determinar el número de comunicaciones y la estrategia de sincronización. El problema de descomposición puede hacer referen-

cia al algoritmo, de forma que haya diferentes procesos trabajando sobre los mismos datos localizados en diferentes procesadores, o puede hacer referencia a datos de la instancia/problema a resolver, de forma que el dominio de dicho problema se particiona y cada uno de los componentes de la descomposición se resuelve aplicando un determinado método. Las comunicaciones, que pueden ser síncronas o asíncronas, permiten intercambiar información entre procesos, siendo importante determinar de forma eficiente la frecuencia de dichas comunicaciones y el volumen de información intercambiada. Algorítmicamente, el paralelismo puede ser explotado utilizando diferentes paradigmas: maestro-esclavo, islas, y difusión [6].

Existe un interés creciente en el diseño de métodos más rápidos para resolver problemas de enrutado de vehículos mediante la incorporación de técnicas de paralelización de diferentes heurísticas [5], con el objetivo de generar soluciones de calidad en un intervalo de tiempo corto, además de poder abordar problemas con un mayor número de clientes a visitar. Uno de los primeros algoritmos paralelos para resolver problemas de enrutado de vehículos fue la propuesta de Taillard [7], quién presentó un método paralelo que descompone dicho problema en regiones polares incluyendo un elevado número de vehículos en sub-problemas que son resueltos independientemente. Sin duda, las heurísticas, en especial la búsqueda tabu, han sido habitualmente utilizadas para resolver este problema, aunque la mayoría de las aproximaciones utilizan el paradigma maestro-esclavo [9]. Algunos autores han resuelto el problema VRPTW utilizando el paradigma de maestro-esclavo en el que el componente maestro coordina los operadores genéticos y gestionan la selección de los padres, mientras que los esclavos ejecutan lo operadores de mutación y reproducción [8]. Otros autores han propuesto algoritmos paralelos para resolver el VRPTW que hacen uso de diferentes estrategias para inicializar las soluciones, búsqueda tabu y algoritmos genéticos como operadores de búsqueda, y un procedimientos de post-optimización basados en búsqueda local [10].

Indicar que la paralelización de algoritmos heurísticos basados en enfriamiento simulado utilizando el modelo de islas en procesadores multinúcleo no ha sido analizado hasta el momento para resolver el problema VRPTW, aspecto que planteamos en la presente investigación.

III. Procedimiento

A continuación describimos la heurística secuencial basada en enfriamiento simulado y su paralelización utilizando el paradigma basado en islas.

A. Algoritmo Secuencial

Los componentes principales del algoritmo secuencial aquí paralelizado son: la representación de las soluciones, el método de construcción de soluciones y la condición de parada, los operadores de búsqueda según las estructuras de vecindario, la estrategia para mantener la diversidad de las soluciones, la selección de soluciones vecinas a la actual.

- *La representación de las soluciones* se realiza mediante una codificación entera. Dado que el método utilizado es poblacional, la población P consta de un total de p individuos (soluciones), P=$\{I_1, I_2, \ldots, I_p\}$, cada uno de los cuales representa las rutas recorridas por los K vehículos para suministrar a los N clientes. Así, cada individuo, I_i, se representa mediante cromosomas, C_{ik}, que consiste en un conjunto variable de genes, $C_{ik} = \{0, G^1_{ik}, G^2_{ik}, ..., G^l_{ik}, 0\}$ que representan la ruta del k-ésimo vehículo en el i-ésimo individuo, con $1 \leq G^j_{ik} \leq N$. Así, el cromosoma $C_{5,3} = \{0, 7, 29, 3, 8, 0\}$ indica que el tercer vehículo del quinto individuo parte del centro logístico, y visita a los clientes 7, 29, 3, y 8, antes de volver a dicho centro logístico.
- *Las soluciones iniciales* se crean mediante una heurística basada en las ventanas temporales (Time Window based Insertion Heuristic, TWIH), que consiste en visitar a los clientes según su orden de apertura de ventana temporal, que son insertados en los vehículos siempre y cuando se cumplan las restricciones del problema. Este procedimiento es muy rápido (inferior a un segundo en todos los benchmarks utilizados en nuestros experimentos) e intrínsecamente reduce el número de vehículos utilizados inicialmente.
- *Los operadores de búsqueda* tienen como objetivo mejorar la calidad de las soluciones creadas mediante TWIH. En la literatura del VRPTW encontramos un elevado elevado número de operadores de mutación [3], [11], de entre los cuales hemos implementado diez con el objetivo de diversificar la búsqueda. Algunos de ellos se basan en elegir un cliente y relocalizarlo en un orden diferente en el mismo vehículo, otras se basan en modificar el vehículo asignado a los clientes, mientras que otros operadores dividen, crean, o eliminan una determinada ruta.
- *El Enfriamiento Simulado (Simulated Annealing, SA)* [12] es una meta-heurística de búsqueda local que simula el proceso seguido por los átomos de un metal cuando se funde y posteriormente se enfría lentamente. En términos computacionales, este proceso implica que las soluciones iniciales tienen asociadas una temperatura inicial elevada, Ti, la cual se va reduciendo conforme avanza el proceso se búsqueda según un factor de enfriamiento, T_{factor}, hasta que cae debajo de un determinado umbral, T_{umbral}, de forma que en un instante dado, aquellas soluciones del vecindario, generadas a través de un operador de variación (mutación), que mejoren a la solución de partida serán directamente aceptadas, mientras que si la empeoran se aceptará con una probabilidad que decrece conforme se reduce la temperatura actual, t. La variable t se incluye dentro de la

denominada función de Metrópolis [13] y actúa como factor de probabilidad para aceptar o rechazar una determinada solución. La estrategia aquí presentada, denominada Multiple Temperature Simulated Annealing (MT-SA), optimiza una población de p individuos utilizando los diez operadores de mutación comentados anteriormente. Estudios previos en otros problemas han demostrado que, en lugar de utilizar un esquema de enfriamiento fijo, el uso de diferentes parámetros de enfriamiento en los individuos de la población permiten diversificar la búsqueda [14]. Por ello, establecemos un intervalo de temperaturas iniciales $[Ti_{min}, Ti_{max}]$, de forma que la temperatura inicial del individuo I_1 es Ti_{min}, la del individuo I_p es Ti_{max}, mientras que el resto se distribuyen regularmente a lo largo de este intervalo.

B. Paralelización basada en el Modelo de Islas

El objetivo de nuestra paralelización (parallel MT-SA, pMT-SA) es obtener soluciones de mayor calidad que los algoritmos secuenciales sin incrementar el tiempo de ejecución. En particular, utilizamos el paradigma basado en islas, en el que la población secuencial se divide en NP islas de forma que cada proceso se encarga de gestionar una de esas islas.

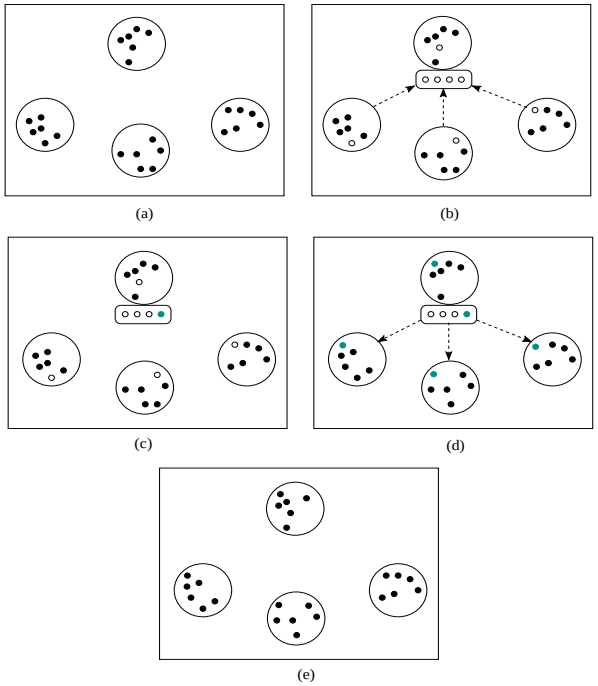

Fig. 1. Funcionamiento del modelo de islas con 4 procesos.

Cada proceso comienza con un conjunto de soluciones iniciales que son mejoradas aplicando MT-SA (pMT-SA) como muestra la Figura 1(a). Además, con el objetivo de equilibrar la diversificación y la intensificación en la búsqueda, esta aproximación paralela permite migrar (copiar) periódicamente el mejor individuo encontrado hasta el momento. Con este objetivo, cada sub-población envía periódicamente la mejor solución encontrada a un proceso central (Figura 1(b)). Dicho proceso central evalúa todas las soluciones recibidas, además de las propias (Figura 1(c)), y distribuye la mejor de todas las soluciones al resto de islas (Figura 1(d)). Tras ello, el proceso de búsqueda continua utilizando su población actualizada con sus propios parámetros (Figura 1(e)). Señalar que cuando se ejecuta el proceso de migración utilizando pMT-SA cada proceso continúa optimizando la solución recibida utilizando el mismo esquema de enfriamiento que el de la solución previamente reemplazada.

Algoritmo 1: MT-SA paralelo (pMT-SA)

1. Parámetros de entrada: *NP, MR, p, TC,* Pr_{mut}, T_{min}, T_{max}, T_{factor}, T_{umbral};
2. Inicializar MPI en cada núcleo $C_i, i \in [1...NP]$
3. Para cada individuo $P_j \in C_i[P]$
4. $C_i[P_j] \leftarrow$TWIH();
5. $C_i[P_j].t \leftarrow T_{min}$+j*(($T_{max}$-$T_{min}$)/$p$);
6. Mientras no se cumpla la condición de parada TC
7. Para cada $C_i[P_j] \in C_i[P]$
8. C_i'$[P_j] \leftarrow$Mutación($C_i[P_j]$,Pr_{mut},Rand(1,10));
9. $C_i[P_j] \leftarrow$Metropolis($C_i[P_j]$,C_i'$[P_j]$),$C_i[P_j].t$);
10. If ($C_i[P_j] < C_i$.Mejor_C then
11. C_i.Mejor_C$\leftarrow C_i[P_j]$;
12. $C_i[P_j].t \leftarrow C_i[P_j].t * T_{factor}$;
13. If($C_i[P_j].t < T_{umbral}$) then
14. $C_i[P_j].t \leftarrow T_{min}$ + j*((T_{max}-T_{min})/p);
15. Si (Migrar(*iteracón_actual*,*instante_actual*,MR))
16. Si (i=1)
17. Para cada proceso C_r, $\forall r \in$ [2....NP]
18. Recibir(C_r.Mejor_C,C_r);
19. Mejor_global$\leftarrow$Mejor(C_j.Mejor_C),$\forall$j$\in$[1..NP];
20. Para cada proceso C_r, $\forall r \in$ [2....NP]
21. Enviar(Mejor_global,C_r);
22. Si (i>1) then
23. Enviar(C_i.Mejor_C,C_1);
24. Recibir(Mejor_global,C_1);
25. Para cada $C_i[P_j] \in C_i[P]$
26. Si (Rand(1,100)<25)
27. $C_i[P_j] \leftarrow$Mejor_global;
28. Retornar(Mejor_global);

El Algoritmo 1 describe el funcionamiento de pMT-SA. Los parámetros de entrada son el número de procesos utilizados NP, el ratio de migración MR (número de iteraciones o tiempo de ejecución entre migraciones), el tamaño de la población p, la condición de parada TC, la probabilidad de aplicar el operador de búsqueda (mutación), Pr_{mut}, y los parámetros de enfriamiento simulado: T_{min}, T_{max}, T_{factor}, T_{umbral}. Primeramente, se inicializan los parámetros de la librería de paso de mensajes MPI (línea 2), tras lo cual se ejecuta el mismo código en cada núcleo (líneas 3-27). Cada uno de esos núcleos inicializa su población de soluciones de acuerdo a la heurística TWIH (línea 4), y se asigna una temperatura inicial a cada solución (línea 5). El bucle principal (líneas 6-27) se repite hasta que se cumple la condición de parada. El bucle interno (líneas 7-14) consiste en evolucionar la población de soluciones utilizando los operadores de mutación (línea

8), aceptando dichas variaciones de acuerdo al criterio de Metropolis (línea 9). En caso de que la nueva solución obtenida mejore a la mejor encontrada durante el proceso de búsqueda, esta es almacenada en una estructura externa (líneas 10-11). La temperatura se actualiza de acuerdo con la temperatura actual y el factor de enfriamiento (línea 12), y si esta temperatura cae por debajo de T_{umbral} se reinicializa (líneas 13-14). Antes de continuar el proceso de búsqueda, se analiza si los procesos se han de comunicar entre sí (líneas 15-27), en cuyo caso, un proceso central recibe la mejor solución encontrada por los restantes procesos, determina cuál es la mejor solución global, y lo envía al resto de procesos, que reciben esta solución copiándola en el 25% de las soluciones de sus respectivas sub-poblaciones.

Este esquema de paralelización es muy flexible ya que puede ser aplicado a un elevado número de problemas de optimización utilizando una gran variedad de algoritmos con una metodología unificada.

IV. Análisis Experimental

A. Benchmarks

La evaluación de pMT-SA en la resolución del VRPTW se lleva a cabo utilizando el conjunto de benchmarks propuesto por Solomon [16], que se compone de 56 instancias divididas en seis categorías, todas ellas con 25 vehículos y 100 clientes. Cada benchmark incluye las capacidades máximas de los vehículos, así como las características de los clientes (coordenadas geográficas, demandas, ventanas temporales y tiempo de servicio). La distancia entre dos clientes es calculada a partir de sus coordenadas cartesianas utilizando distancias euclídeas. Las seis categorías son C1, C2, R1, R2, RC1, y RC2, donde los benchmarks de categoría R tienen los clientes distribuidos aleatoriamente, los de la categoría C están concentrados, siendo la categoría RC una mezcla de ambas. Por otro lado, los problemas de la categoría 1 (C1, R1, y RC1) tienen ventanas temporales más estrechas que los de la categoría 2 (C2, R2, y RC2).

B. Configuración Paramétrica

La versión secuencial (MT-SA) utiliza una población de $|P|$=200 individuos, por lo que estos son distribuidos equitativamente entre las islas, cada una de las cuales es gestionada por un proceso ejecutado en un núcleo del procesador. Los individuos son inicializados utilizando la heurística TWIH antes descrita, y la probabilidad de aplicar un operador de mutación en cada iteración para un determinado individuo es Pr_{mut}=50%. Si se aplica dicha mutación, se elige uno de los diez operadores con una probabilidad aleatoria de entre el 5% y el 15%. El rango de temperaturas iniciales para los individuos de la población oscila entre $Ti=[T_{min}, T_{max}]$=[1,100], el factor de enfriamiento utilizado es T_{factor}=0.995, mientras que la temperatura de reinicialización es T_{umbral}=0.001, de forma que si la temperatura de cualquier individuo cae por debajo de T_{umbral}, dicha temperatura es reinicializada ($t=Ti$) y el proceso de búsqueda continúa con la solución actual.

Dado que el objetivo de la paralelización (usando NP={2,3,4} procesos/núcleos) es mejorar la calidad de las soluciones obtenidas por la versión secuencial (NP=1) sin incrementar el esfuerzo computacional, hemos establecido como condición de parada que el tiempo de ejecución de las ejecuciones sea de 60 segundos. Investigaciones recientes en el ámbito del enrutado de vehículos han utilizado el tiempo de ejecución como medida para estimar el coste computacional [17]. En cuanto a las comunicaciones, se ha establecido el criterio de realizar una comunicación cada 6 segundos, es decir, el 10% del tiempo de ejecución total, de forma que en esos instantes concretos todos los procesos envían la mejor solución encontrada en su isla a un proceso central, el cual determina cual de todas, incluida la mejor propia, es la mejor global. Tras ello, el proceso central envía dicha solución al resto, y se copia en el 25% de los individuos de las islas, que continúan el proceso de búsqueda con la misma configuración paramétrica.

Se han lanzado 15 ejecuciones independientes de pMT-SA usando NP={1,2,3,4} procesos/núcleos para cada benchmark, retornando del mejor resultado y la mediana de dicha tanda de ejecuciones. El código secuencial ha sido implementado utilizando C++, y ha sido paralelizado utilizando MPI (MPICH2 versión 1.2.1p1) utilizando una estación de trabajo con un procesador Intel Quad Core (4 núcleos), 2.27 GHz, y 4 GB de memoria RAM.

C. Resultados y Discusión

La Tabla I muestra los resultados obtenidos por las versiones secuencial (NP=1 procesos/núcleos) y paralelas (NP=2, NP=3, y NP=4), en términos del mejor resultado y la mediana de las 15 ejecuciones independientes lanzadas para resolver cada uno de los 56 benchmarks que componen las seis categorías de problemas de Solomón. Dichos resultados muestran claramente como las versiones paralelas mejoran a la versión secuencial, y que el incremento del número de procesos/núcleos utilizados permiten mejorar la calidad de las soluciones. En concreto, el coste medio de la ejecución mediana correspondiente a la versión secuencial (NP=1 proceso) es 1489.8, mientras que en el caso de las ejecuciones secuenciales se obtiene un resultado de 1170.6 para NP=2, de 1160.3 para NP=3, y de 1140.5 para NP=4. Por tanto, esos resultados muestran claramente como es posible mejorar la calidad de las soluciones obtenidas haciendo uso de procesamiento paralelo utilizando el modelo de islas. Además, los resultados desglosados denotan que haciendo uso de cuatro núcleos es posible obtener soluciones muy próximas a las mejores soluciones conocidas para estos problemas [4].

La Figura 2 muestra la tendencia de la mejor solución encontrada por MT-SA/pMT-SA en el benchmark C102. Como se observa, el uso de núcleos adicionales ayuda a escapar de óptimos locales con mayor facilidad, permitiendo mejorar la calidad de las soluciones.

TABLA I

Resumen de los resultados obtenidos por MT-SA y pMT-SA en los seis tipos de problemas de Solomon.

		NP=1	NP=2	NP=3	NP=4
R1	mejor	1485.0	1269.7	1264.5	**1256.6**
	mediana	1537.5	1321.0	1316.1	**1303.7**
R2	mejor	1338.0	1011.0	1005.5	**999.8**
	mediana	1400.3	1091.6	1087.2	**1069.2**
C1	mejor	1336.7	900.7	882.6	**869.0**
	mediana	1427.0	1006.1	985.5	**953.2**
C2	mejor	1039.6	683.6	674.2	**673.9**
	mediana	1119.2	795.7	783.0	**765.8**
RC1	mejor	1703.2	1458.3	1451.9	**1432.4**
	mediana	1776.8	1518.5	1512.5	**1493.4**
RC2	mejor	1584.4	1173.6	1171.8	**1154.7**
	mediana	1685.5	1290.6	1277.4	**1258.0**
MEDIA	mejor	1414.5	1082.8	1075.1	**1064.4**
	mediana	1489.8	1170.6	1160.3	**1140.5**

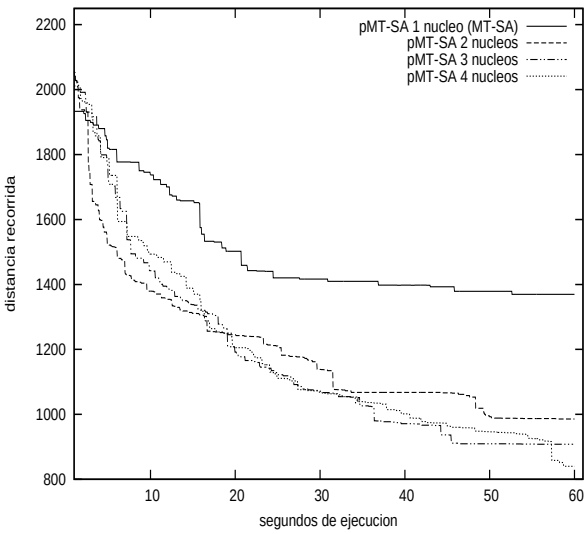

Fig. 2. Mejor solución encontrada en tiempo de ejecución con 1, 2, 3 y 4 núcleos (benchmark C102).

V. Conclusiones

La optmización computacional de problemas de enrutado de vehículos se ha convertido en un importante área de investigación debido a su influencia en el campo de la logística y del transporte. Dado que dicho problema es NP-duro, no es posible obtener la solución óptima aplicando métodos deterministas, siendo necesario aplicar métodos aproximados, como es el caso de heurísticas. Sin embargo, en determinadas aplicaciones es necesario abordar instancias de gran tamaño u obtener soluciones de alta calidad en un tiempo reducido, resultando conveniente incorporar técnicas que mejoren la eficiencia de dichas heurísticas. En el presente trabajo se ha abordado cómo mejorar el rendimiento de una técnica heurística basada en enfriamiento simulado mediante la paralelización basada en el modelo de islas utilizando MPI. Los resultados obtenidos en la resolución de un conjunto de benchmarks asociados al problema VRPTW muestran que dicha paralelización es capaz de mejorar la calidad de las versiones secuenciales sin incrementar el esfuerzo computacional. Como trabajo futuro se plantea la paralelización haciendo uso de esquemas asíncronos, el uso de otros esquemas de paralelización haciendo uso de librerías de memoria compartida, en especial OpenMP, así como la extensión al caso multiobjetivo haciendo uso de técnicas basadas en optimización de Pareto.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Ciencia e Innovación a través de los proyectos TIN2008-01117 y SAF2010-20558. R. Baños agradece la finaciación del contrato postdoctoral Juan de la Cierva.

Referencias

[1] B. Eksioglu, A.V. Vural, A. Reisman, *The vehicle routing problem: a taxonomic review*, Computers & Industrial Engineering 57(4), 1472-1483 (2009).

[2] J.K. Lenstra, A.H.G.R. Rinnooy Kan, *Complexity of vehicle routing and scheduling problems*, Networks 11, 221-227 (1981).

[3] N.A. El-Sherbeny, *Vehicle routing with time windows: an overview of exact, heuristic and metaheuristic methods*, Journal of King Saud University (Science) 22(3), 123-131 (2010).

[4] Y. Nagata, O. Bräysy, W. Dullaert, *A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows*, Computers & Operations Research 37, 724-737 (2010).

[5] O. Bräysy, M. Gendreau, *Vehicle routing problem with time windows, part II: metaheuristics*, Transportation Science 39, 119-139 (2005).

[6] C.A., Coello, G.B. Lamont, D.A. Van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*, Genetic and Evolutionary Computation Series, Springer (2007).

[7] É. Taillard, *Parallel iterative search methods for vehicle routing problems*, Networks 23(8), 661-673 (1993).

[8] J. Berger, M. Barkaoui, *A parallel hybrid genetic algorithm for the vehicle routing problem with time windows*, Computers & Operations Research 31(12), 2037-2053 (2004).

[9] H. Gehring, J. Homberger, *Parallelization of a two-phase metaheuristic for routing problems with time windows*, Journal of Heuristics 8(3), 251-276 (2002).

[10] A. Le Bouthillier, T.G. Crainic, *A cooperative parallel meta-heuristic for the vehicle routing problem with time windows*, Computers & Operations Research 32(7), 1685-1708 (2005).

[11] G.B. Alvarenga, G.R. Mateus, G. de Tomic, *A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows*, Computers & Operations Research 34(6), 1561-1584 (2007).

[12] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, *Optimization by simulated annealing*, Science 220(4598), 671-680 (1983).

[13] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A. Teller, E. Teller, *Equation of state calculations by fast computing machines*, The Journal of Chemical Physics 21(6), 1087-1092 (1953).

[14] R. Baños, C. Gil, J. Ortega, F.G. Montoya, *A parallel multilevel metaheuristic for graph partitioning*, Journal of Heuristics 10(3), 315-336 (2004).

[15] C. Rego, *Node-ejection chains for the vehicle routing problem: Sequential and parallel algorithms*, Parallel Computing 27(3), 201-222 (2001).

[16] M.M. Solomon, *Algorithms for the vehicle routing and scheduling problems with time window constraints*, Operations Research 35(2),254-65 (1987).

[17] H. Derbel, B. Jarboui, S. Hanafi, H. Chabchoub, *An iterated local search for solving a location-routing problem*, Electronic Notes in Discrete Mathematics 36, 875-882 (2010).