Computación en Malla Aplicada a la Generación de Rutas de Trasporte Escolar

Grid Computing Applied to Scholar Routing Generation

Julián Orlando Díaz Rivera
Departamento de Ingeniería de Sistemas y Computación
Universidad de los Andes - Bogotá D.C. – Colombia
e-mail: julian.diaz@cav-ingenieros.com

Germán Enrique Bravo Córdoba
Departamento de Ingeniería de Sistemas y Computación
Universidad de los Andes - Bogotá D.C. – Colombia
e-mail: gbravo@uniandes.edu.co

RESUMEN

Hoy en día la computación en malla se impone en la ejecución de los procesos de optimización que necesitan de una gran capacidad de procesamiento, almacenamiento y colaboración entre recursos computacionales para llegar a la mejor solución. Esta resuelve más rápido problemas asociados a la optimización de recursos y las investigaciones que llevan consigo grandes costos computacionales, mediante la reducción de la inversión de componentes hardware. Dentro de éste marco, el problema de enrutamiento vehicular (VRP – Vehicle Routing Problem) se convierte en un excelente caso de estudio para probar las capacidades ofrecidas por SGE (Sun Grid Engine) la implementación grid de Sun Microsystems. La naturaleza NP-Hard de VRP, su complejidad computacional, el uso de heurísticas y metaheurísticas y la necesidad de llegar al mejor conjunto de rutas, son suficientes restricciones que indican la necesidad de una arquitectura computacional no convencional que acelere los procesos y magnifique los resultados. A lo largo de este artículo se intenta demostrar que la computación en malla se postula como una solución a los inconvenientes de procesamiento de información, envueltos en el proceso de solucionar los problemas de enrutamiento vehicular.

ABSTRACT

Today grid computing prevails in the execution of the optimization processes that need a great capacity of CPU processing, storage and collaboration among resources in order to get the best solution. Grid computing allows a more efficient solution to problems associated to the optimization of resources and costly research, by reducing the associated components hardware. Within this context, vehicular routing problems (VRP) become an excellent case of study to test the capacities offered by SGE (Sun Grid Engine) the grid middleware of Sun Microsystems. The NP-Hard nature of VRP, its computing complexity, the use of heuristics and metaheurístics and the obligation to get the best routes, are sufficient to reveal the necessity of non-conventional infrastructure in order to accelerate the processes and magnify the results. This article shows that grid computing is a valid solution to the problems of information processing dealing with vehicular routing problems.

1. INTRODUCCIÓN

El definir un nuevo paradigma a mediados de los años 90 llamada computación en malla, ha generado un revuelo en las entidades industriales, educativas y de investigación hacia el uso de esta nueva forma de hacer computación distribuida.

La capacidad de la computación en malla para sacar el mejor provecho de los recursos hardware y software disponibles en cualquier entidad, la hace prevalecer sobre otras soluciones anteriormente planteadas, ya que adquirir cualquiera de estas implica incurrir en costos monetarios muy altos que no todas las instituciones pueden sobrellevar. Por otro lado, los problemas *VRP* son uno de los casos de optimización combinatoria más conocidos en el mundo por la cantidad de recursos que son necesarios para llegar a una respuesta óptima.

Es por esto, que para demostrar las capacidades de *SGE* utilizamos este caso de estudio, el cual cumple con todas las características relevantes de los problemas que en los últimos años quieren ser solucionados a través de la computación en malla.

A lo largo de éste artículo se demostrará que *SGE* es una herramienta que proporciona todos elementos necesarios, para llegar a una solución del caso de estudio, reduciendo de forma significativa los tiempos de ejecución y aumentando el espacio de búsqueda.

2. MARCO TEORICO

De manera muy breve se explican los diferentes tópicos relativos a este trabajo, como son la computación en malla, los problemas VRP, entre los cuales se incluye el caso de estudio y la plataforma computacional sobre la cual se trabajó.

2.1. COMPUTACIÓN EN MALLA

La computación en malla se ha convertido en una solución tecnológica que permite no solo maximizar la capacidad de los recursos computacionales de las entidades, sino también una gran herramienta ahorradora de dinero y tiempo, dos de los factores más importantes en un mundo en el cual la relación tiempo-costo se ha convertido en el emblema de las compañías analistas, productoras e investigadoras.

La computación en malla reúne el conocimiento adquirido con tecnologías anteriores y lo mejora, teniendo como ideal el formar una red de computadores que se comporte como una red eléctrica, es decir que cualquier computador sea capaz de conectarse a una red en la cual los recursos (poder computacional y capacidad de almacenamiento) estén a su entera disposición, permitiendo así que el usuario de la malla la perciba como una unidad única de ejecución.

2.1.1. COMPONENTES PRINCIPALES

La literatura internacional que se puede encontrar y que profundiza el tema de la computación en malla es de grandes proporciones; así mismo se hallan grandes cantidades de discusiones y definiciones que tratan de realizar una definición estándar de los principales componentes del grid. Una de las lecturas que mejor describen estos componentes es la de Bertis [2]:

- **Recursos**: La capacidad de almacenamiento y de procesamiento. Se busca aprovechar al máximo los recursos que se tienen.
- **Software**: Licencias de una aplicación específica que se desea ejecutar y de la cual se debe realizar un licenciamiento. Esto eleva los costos económicos de la investigación.
- **Tareas**: Son las unidades de trabajo que se ejecutan en los recursos que conforman la grid. En su mayoría son tratadas por algún recurso en específico de acuerdo con sus características.
- **Aplicaciones**: Están compuestas por un grupo de tareas que son divididas y enviadas a un recurso en específico según las políticas de configuración del grid.
- **Scheduling**: Es un mecanismo del grid que define cómo repartir las tareas en los recursos disponibles.

2.2. PROBLEMA VRP

Los problemas VRP intentan determinar un conjunto de rutas de costo mínimo para realizar un conjunto de tareas del estilo entregar o recoger mercancía, recoger y repartir pasajeros o simplemente transportar algún objeto o persona.

Los problemas de enrutamiento vehicular se caracterizan por aumentar su complejidad a medida que se involucran más clientes en el problema, pues aumenta de manera combinatoria el número de soluciones posibles. Es por esto que estos problemas de optimización combinatoria son considerados de tipo NP-Hard cuando se habla de su complejidad computacional.

El enrutamiento escolar, el caso de estudio escogido para probar la capacidad de la computación en malla, involucra varios de los aspectos para los cuales aplica la computación en malla. Entre ellos encontramos como principal restricción el tiempo de ejecución de este tipo de aplicaciones, ya que de este es directamente proporcional al número de iteraciones que se realicen, a la búsqueda de la mejor solución y a la asignación de grandes cantidades de clientes y de vehículos de transporte, en donde es de suma importancia tener en cuenta la capacidad de éstos últimos.

Las mejores soluciones encontradas a estos problemas de enrutamiento vehicular se han encontrado utilizando metaheurísticas conocidas como simulated annealing y la búsqueda tabú. A continuación se explica el Simulated Annealing, que fue el utilizado en este estudio.

2.3. SIMULATED ANNEALING

Simulated Annealing es una de las Metaheurísticas más conocidas y usadas para resolver los problemas de optimización combinatoria. Este algoritmo fue creado a mediados de los ochenta por Kirkpatrick y se basa en el proceso físico de enfriamiento de los metales. Es no determinístico, pues tiene bases aleatorias, e intenta evaluar el entorno de soluciones posibles con el fin de llegar a la mejor de las respuestas, a través de la reducción de la probabilidad de aceptación de las soluciones exploradas.

El algoritmo consiste en realizar una búsqueda aleatoria de soluciones, restringidas por la probabilidad de ser aceptadas a medida que las iteraciones aumentan. Es decir a medida que nos acercamos al límite de iteraciones, es menos probable aceptar una solución lejana al óptimo. Este algoritmo tiene algunos parámetros que deben ser definidos antes de ejecutalo, como son la temperatura inicial (restricción del espacio de soluciones), la curva de enfriamiento (intervalos cortos o largos de búsqueda) y el criterio de aceptación (probabilidad de ser aceptado); estos parámetros no son estándar y son diferentes para cada tipo de problema.

2.4. SGE (SUN GRID ENGINE)

Sun Grid Engine, la propuesta *grid* de *SUN Microsystems* se caracteriza por ser un *grid* de campus, y por lo tanto no tiene la capacidad de involucrarse con las mallas de cobertura mundial existentes hoy en día. Al contrario de los demás, pretende ofrecer capacidad computacional a las entidades que necesitan de un gran poder computacional, de almacenamiento y de colaboración para llevar a cabo sus procesos internos. Aunque este tipo de malla no cumple con la expectativa general de la malla, se postula como una solución diseñada para maximizar la capacidad de recursos hardware y software de cualquier institución sin incurrir en grandes gastos.

La arquitectura de SGE propone cuatro tipos de máquinas huéspedes (hosts):

- **Maestro**: Es la central de toda la actividad del cluster, por lo tanto se encarga de controlar todos los componentes del motor grid, como las colas y las tareas.
- **Ejecución**: Son sistemas que tienen permiso de ejecutar tareas, por lo tanto tienen instancias de colas anexas a ellos.
- **Administración**: Estos son huéspedes que tienen permisos suficientes para consultar cualquier tipo de actividad administrativa.
- **Clientes**: Estos permiten a los usuarios del la malla, el enviar, monitorear y controlar las tareas.

Para la configuración de tareas, SGE permite definir el orden de ejecución de las mismas teniendo en cuenta el nivel de prioridad asignado. Este orden es conocido como las políticas de uso, ya que definen cómo deben ser usados los componentes grid para realizar la ejecución de las tareas. Estas políticas son de **Urgencia** en la cual la prioridad de las tareas se basa en el valor de urgencia de las mismas, **Funcional** donde se da tratamiento especial a los usuarios o tareas que pertenecen a un grupo o proyecto en particular, **Basado en acceso** la cual depende de los privilegios de acceso asignados, la libertad de acceso y los grupos de usuarios, y por último de **Sustitución** donde el administrador modifica manualmente las prioridades.

3. SOLUCIÓN PROPUESTA

A través de la historia se ha intentado reducir los tiempos de ejecución de los algoritmos por medio de su optimización. Cuando esto no es posible por las características mismas del problema (NP, NP Duro o NP Completo), es necesario explorar alternativas que intenten mejorar la capacidad del hardware utilizado, sin generar altos costos y maximizando los recursos computacionales ya existentes. Esto se puede realizar dividiendo el problema en tareas más pequeñas que son repartidas a través de una red de computadores sincronizados, siempre enfocados a llevar al máximo su capacidad, su rendimiento y su robustez, logrando con esto incrementar las iteraciones necesarias para llegar a una mejor respuesta.

Al incluir la computación en malla como una de las posibilidades de solucionar los problemas de transporte escolar VRP, es necesario hablar de la programación en paralelo, por tal motivo uno de los objetivos de ésta investigación fue llegar a una paralelización adecuada para realizar el estudio de SGE como medio de solución a nuestro caso de estudio.

3.1. Arquitectura

A continuación se hace una descripción detallada de las arquitecturas hardware y software que fueron diseñadas para la implementación de esta investigación.

3.1.1. **Hardware.** Teniendo en cuenta las diferentes configuraciones que se pueden realizar gracias a la versatilidad que ofrece en este aspecto SGE, se decidió implementar la siguiente arquitectura hardware:

- **Maestro:** El maestro es único, se encarga de dirigir la malla y por lo tanto es quien envía y encola cada una de las tareas que se deseen ejecutar en cada uno de los ejecutores. Éste fue configurado para que en un principio no controlara el scheduler de la aplicación.
- **Clientes:** Dentro de la aplicación existe un solo cliente, encargado de dividir la ejecución de la una instancia del problema en tareas simples o en grupos, de enviar las peticiones de ejecución al maestro y de recolectar y unificar los resultados.
- **Ejecutores**: Son los obreros de la malla, pues son los encargados de ejecutar las tareas que se le son asignadas por el maestro.

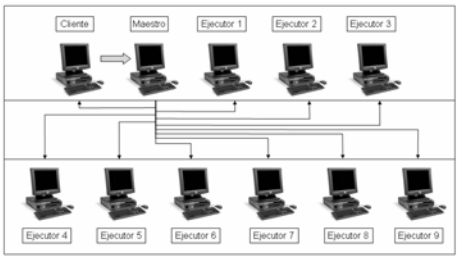

Figura 3-1 Arquitectura hardware.

3.1.2. **Software.** La arquitectura software diseñada para esta aplicación desacopla la ejecución del algoritmo de solución planteado dentro de la malla, como se muestra en la Figura 3-2. En nuestro caso de estudio utilizamos la metaheurística simulated annealing, sin embargo es posible utilizar otro método y ejecutarlo a través de este diseño.

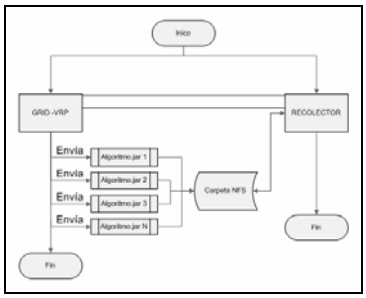

Figura 3-2 Arquitectura software.

Grid–VRP: Aplicación encargada de recibir los parámetros de ejecución, dividir las tareas y de enviarlas a la malla para que sean ejecutadas.

Algoritmo.jar: Es un archivo .jar en el que se encuentra comprimido el algoritmo de la metaheurística que se va a ejecutar.

Carpeta NFS *(Network File System)***:** Carpeta de acceso compartido que le permite a cada ejecutor de la malla acceder a los diferentes archivos necesarios para la ejecución de cada una de las tareas.

Recolector: Como su nombre lo indica es el encargado de recoger los resultados generados por cada uno de los ejecutores, compararlos y generar la solución final.

3.2. Simulated Annealing en paralelo

La implementación realizada por Cesar Niño [1] para la solución de este tipo de problemas de optimización combinatoria, implica una generación de ruta a través de la heurística Metrópoli Montecarlo y una evaluación de la misma a través del *simulated annealing.* Es por esto que de ahora en adelante se hablará de las iteraciones de simulated annealing (ISA) y de las iteraciones Montecarlo (IMC).

3.2.1. Individual. Esta paralelización envía una sola ruta a un ejecutor en específico, es decir si queremos realizar 100 ISA y 10 IMC, lo cual implica la evaluación de 1.000 rutas donde cada 10 de ellas son evaluadas con la misma temperatura, se genera el mismo número de rutas que de tareas, es decir de ejecutan 1.000 tareas donde cada una de ellas es enviada a un único ejecutor, que genera una respuesta.

Cuando el número de tareas excede el número de ejecutores disponibles, a cada ejecutor se le envían varias tareas, que son atendidas en orden de llegada por un manejador de colas en cada ejecutor. El tamaño máximo de estas colas es definido por SGE, en 100.

3.2.2. Grupos. Esta paralelización envía a cada uno de los ejecutores un grupo de rutas (no sólo una) que deben ser evaluadas. Por lo tanto cada ordenador ejecuta un proceso simulated annealing con el fin de explorar más alternativas dentro de todo el espacio de soluciones. Al igual que en la individual cada computador genera una respuesta.

Esta forma de paralelizar se implementó con dos estrategias diferentes:

Dividiendo: Se dividen las ISA entre el número de computadores que se utilicen. Es decir se quieren ejecutar 500 ISA y 20 IMC y se utilizan 10 computadores, cada computador ejecutará 50 ISA y 20 IMC.

No Dividiendo: En este caso no se realiza ninguna división y por lo tanto si se va a ejecutar un proceso con la misma configuración del ejemplo anterior, cada ejecutor realiza 500 ISA y 20 IMC.

4. PLAN DE PRUEBAS Y ANÁLISIS DE RESULTADOS

Teniendo en cuenta las diferentes configuraciones que SGE ofrece en cuanto a su arquitectura y número de máquinas, y la cantidad de iteraciones que se ejecutan de la metahaurísica, se implementaron las siguientes pruebas para evaluar el incremento en la eficiencia a medida que se aumenta la cantidad de equipos involucrados en la malla (Ver Tabla 4-1). Para cada una de las pruebas realizadas se realizó el calculo del tiempo de envío, tiempo de ejecución, tiempo de espera y de recolección de las tareas.

Tabla 4-1 Configuración de pruebas.

Descripción	Valor		
Plataforma	Linux		
Número máquinas	1, 2, 4,8 y 10 ordenadores.		
Archivos de entrada	12 nodos. BPC-234.txt 34 nodos. ftv36.atsp(Vector).csv 64 nodos. ftv64.atsp(Vector).csv		
Paralelización	Individual		
	Grupos.	Dividiendo	
		No Dividiendo	
Simmulated Annealing	Iteraciones SA		
	Iteraciones MC	110 % del valor de las iteraciones SA.	

4.1. Resultados y análisis

Después de la ejecución del plan de pruebas realizado se obtuvieron los siguientes resultados:

4.1.1. Paralelización individual. En estas pruebas solo se utilizó la ruta de 12 nodos ya que a medida que se fue avanzando en la ejecución del plan de pruebas, los resultados mostraban que los tiempos se incrementaban de forma significativa y no se encontraba una mejor respuesta.

De la Figura 4-1 a la Figura 4-4 se muestran algunos de los resultados obtenidos a lo largo de estas pruebas. En ellas se muestra que la forma como se intentó explorar el campo de soluciones no fue la mejor, pues no existe ningún orden a seguir, sencillamente se envía una única ruta y se evalúa.

Sin embargo estas pruebas nos muestran que SGE presenta estabilidad a la hora de realizar los envíos de las tareas como se observa en la Figura 4-2.

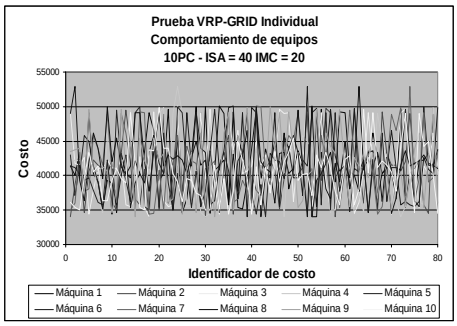

Figura 4-1. Comportamiento de equipos - Paralelización individual.

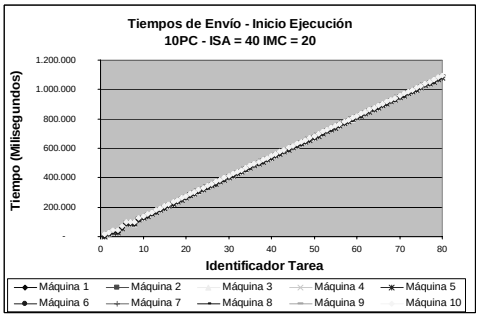

Figura 4-2 Tiempos de envío - Paralelización individual.

Los tiempos de espera tienen un rango máximo aproximado de 20 segundos, pues se ve en las pruebas realizadas que la mayoría de las tareas debían esperar en promedio entre 13 y 20 segundos para iniciar su ejecución.

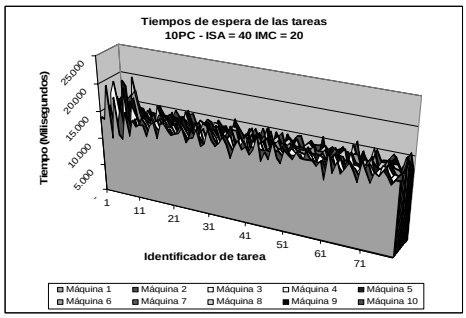

Figura 4-3 Tiempo de espera de las tareas - Paralelización individual.

A medida que se incrementan las tareas, los ordenadores son saturados de procesos y por lo tanto los tiempos de ejecución se incrementan como se muestra en la siguiente figura.

El comportamiento de los tiempos de recolección está directamente relacionado con el de ejecución, por tal motivo, al ser tan variante el tiempo en que se ejecutan las tareas, los valores de recolección varían. Esto se muestra en la Figura 4-5.

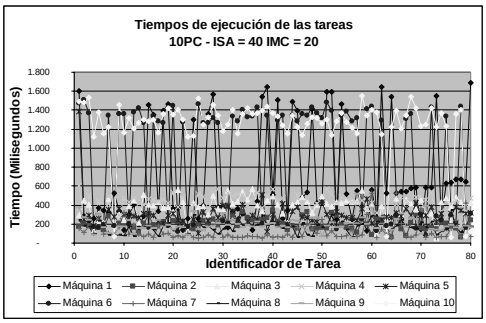

Figura 4-4 Tiempo de ejecución de las tareas - Paralelización individual.

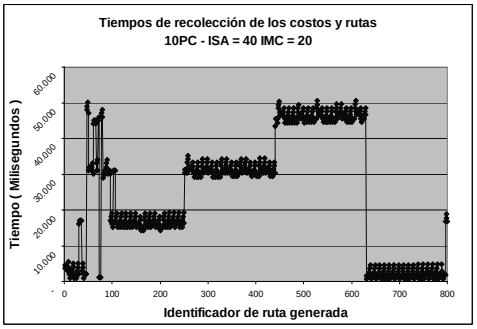

Figura 4-5 Tiempo de recolección de tareas - Paralelización individual.

4.1.2. Tiempos y solución VRP. Como resultado de las pruebas se puede concluir que a medida que se aumentan las iteraciones y el número de ordenadores es pequeño, los tiempos de ejecución se incrementan, en cambio si se aumenta la cantidad de computadores, los tiempos se reducen considerablemente.

La Figura 4-6 se muestra que aunque se incrementa el número de máquinas el costo se mantiene igual. Esto mismo sucede con el número de iteraciones (Ver Figura 4-7).

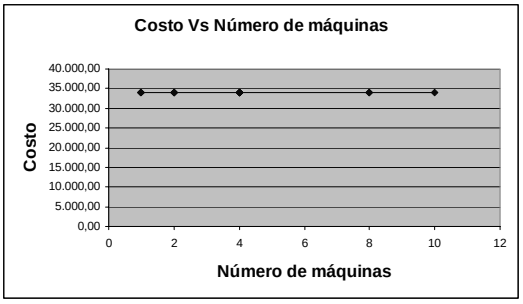

Figura 4-6 Costo vs. Número de máquinas.

A medida que se aumentan los computadores y se mantienen fijas el número de iteraciones, el tiempo de ejecución se reduce considerablemente como se muestra en la Figura 4-8.

Después de observar la información obtenida por estas pruebas se puede ver que este tipo de paralelización no es buena para resolver problemas de optimización combinatoria, ya que el tiempo necesario para la ejecución del algoritmo es muy alto y el número de iteraciones muy bajo (Tabla 4-2Tabla 4-2)

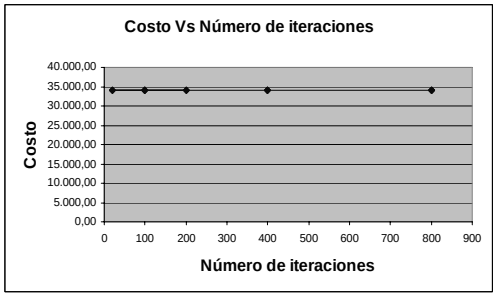

Figura 4-7 Costo vs. Número de iteraciones.

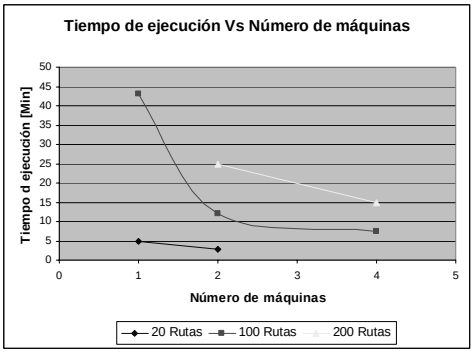

Figura 4-8 Tiempo de ejecución vs. Número de máquinas.

Tabla 4-2 Rutas evaluadas vs. Tiempos de ejecución.

Número PC's	Rutas evaluadas	Tiempo ejecución [Min]
1	100	43,03
2	200	24,85
4	400	43,97
8	800	29,9
10	800	20,3

4.2. Paralelización por grupos

Para este tipo de paralelización se lograron utilizar tres tipos de rutas diferentes ya que los resultados obtenidos con el primer archivo de pruebas fueron muy satisfactorios.

Como se muestra de la Figura 4-9 el comportamiento de este tipo de pruebas en grupo es más organizado y muestra una congruencia hacia una respuesta final.

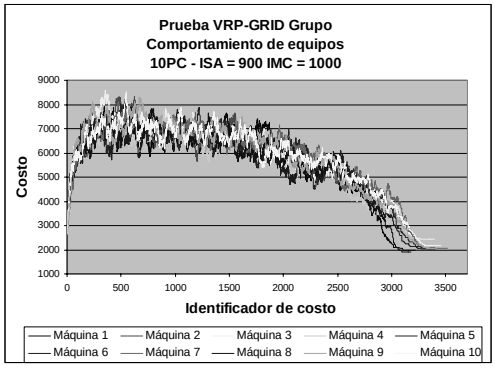

Figura 4-9 Comportamiento de equipos - Paralelización en grupos.

Por otro lado los tiempos de envió de tareas se mantienen bajos frente a los de ejecución y recolección que son los que sobresalen a la hora de ejecutar la implementación. Esto se debe a que el tiempo de ejecución es afectado directamente por la cantidad de iteraciones que se ejecuten, así si estas incrementan, el tiempo aumentará. De la misma manera el tiempo de recolección es directamente proporcional al tiempo de ejecución de las tareas y por lo tanto a la cantidad de rutas aceptadas, de allí que cuando se aumenta el número de aceptaciones, el recolector tome más tiempo en obtener la mejor respuesta de la ejecución.

A continuación se muestran los resultados obtenidos:

4.2.1. Dividiendo iteraciones. A continuación se muestra el comportamiento de las pruebas realizadas para los tres conjuntos de datos de prueba, con 12, 36 y 64 nodos respectivamente**.**

4.2.1.1. Para 12 nodos. Como se muestra en la Figura 4-10 los tiempos de ejecución de las tareas se reducen respecto a la paralelización individual donde los tiempos oscilan entre 2,8 y 43,7 minutos.

En cuanto a los costos como se muestra en la Tabla 4-3 se obtienen mejores resultados ya que estos se reducen hasta casi un valor de 32.000.

Tabla 4-3 Comparación individual vs. grupo dividiendo - 12 nodos.

	Costo Final	**Tiempo [Min]**
Mejor Costo Individual	33.919,13	2,79
Mejor Costo Grupo Dividiendo	32.110,57	0,25

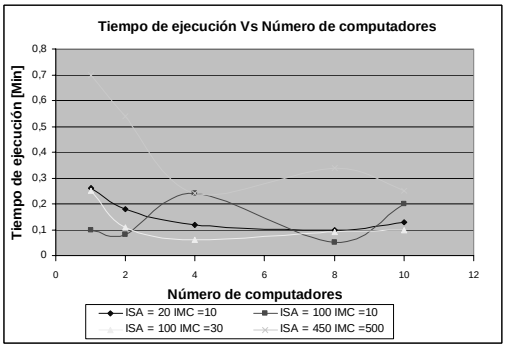

Figura 4-10 Tiempo de ejecución vs. Número de computadores para 12 nodos.

4.2.1.2. Para 36 nodos. En la Tabla 4-4 se muestran los mejores resultados obtenidos a lo largo de las pruebas realizadas para la ruta de 36 nodos. En ellas se muestra que se obtienen buenos resultados reduciendo a la vez el tiempo de ejecución de las tareas.

Tabla 4-4 Mejores costos obtenidos.

	Costo Final	**Tiempo [Min]**
1	1.348,00	1,7
2	1.403,00	2,04
3	1.454,00	1,21

Tabla 4-5 Comparación de costos y tiempo obtenidos.

	Costo	**Tiempo [Min]**
Mejor Respuesta obtenida	1.348,00	1,70
Mejor Respuesta Tesis César Niño	1.476,33	5,92
% de mejora	8,7%	71,3%

Teniendo en cuenta la mejor respuesta obtenida en la tesis de Cesar Niño [1] y los resultados obtenidos, se puede observar que la ruta obtenida mejora en un 8,7% el costo y en un 71,3% el tiempo de ejecución, como se muestra en la **¡Error! La autoreferencia al marcador no es válida.**.

Como era de esperar los tiempos son más pequeños a medida que se incrementa el número de computadores involucrados en la ejecución. Además al igual que en las pruebas realizadas para la ruta de 12 nodos, cuando se utilizan 4 y 8 máquinas se reducen los costos, mientras que cuando se usan 2 o 10 los costos se incrementan. Esto puede llevar a pensar que el aumento de máquinas no implica obtener la mejor de las respuestas, por lo menos para problemas relativamente pequeños.

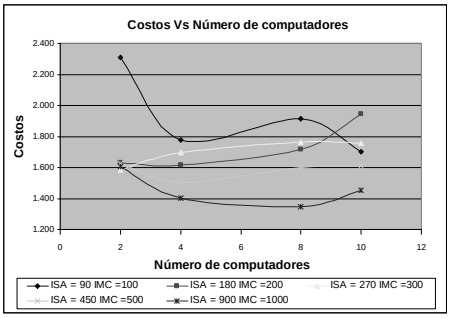

Figura 4-11 Costos vs. Número de computadores para 36 nodos dividiendo.

4.2.1.3. Para 64 nodos. La Tabla 4-6 muestra los mejores resultados obtenidos a lo largo de las pruebas realizadas para la ruta de 64 nodos.

Tabla 4-6 Mejores costos obtenidos.

	Costo Final	Tiempo [Min]
1	1.894	5,49
2	1.911	12,22

Los tiempos son más pequeños a medida que se incrementa el número de computadores involucrados en la ejecución de la aplicación (Ver Figura 4-12). Sin embargo la reducción máxima de se encuentra cuando se utilizan entre 8 y 10 máquinas.

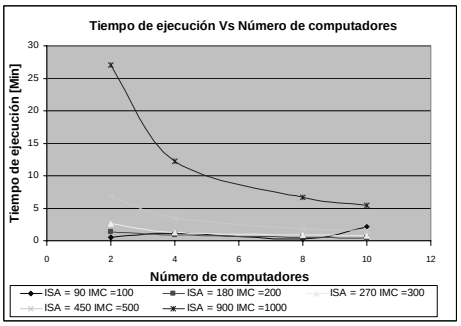

Figura 4-12 Tiempo de ejecución vs. Número de computadores para 64 nodos dividiendo.

Tabla 4-7 Comparación de costos y tiempo obtenidos.

	Costo Final	Tiempo [Min]
Mejor Respuesta obtenida	1.894,00	5,49
Mejor Respuesta Tesis César Niño	1.932,30	17,80
% de mejora	2,0%	69,2%

Cuando se utilizan 4 máquinas se reducen los costos y se obtienen buenos resultados, sin embargo cuando se usan 10 ordenadores con 900 ISA y 1000 IMC se obtiene el menor de los costos. En la Tabla 4-7 se observa que las rutas obtenidas mejoran en un 2,0% el costo y en un 69,2% el tiempo de ejecución.

4.2.2. Sin Dividir iteraciones

4.2.2.1. Para 36 nodos. En la Tabla 4-8 se muestran los resultados que se tuvieron a lo largo de las pruebas.

Tabla 4-8 Mejores costos obtenidos.

	Costo Final	Tiempo [Min]
1	1.348	1,26
2	1.382	12,82
3	1.394	3,05
4	1.408	11,58
5	1.412	3,32
6	1.444	12,67
7	1,409	12,11

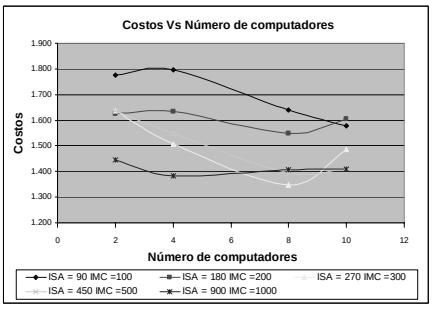

Figura 4-13 Costos vs. Número de computadores para 36 nodos sin dividir.

Tabla 4-9 Comparación de costos y tiempo obtenidos.

	Costo Final	Tiempo [Min]
Mejor Respuesta obtenida	1.348,00	1,26
Mejor Respuesta Tesis César Niño	1.476,33	5,92
% de mejora	8,7%	78,7%

El comportamiento de todas las configuraciones es similar ya que cuando se utilizan 8 computadores se obtienen los tiempos de ejecución más pequeños. Sin embargo la ejecución de 900 ISA y 1.000 IMC sobresale por su incremento en el tiempo.

Los mejores resultados en la mayoría de los casos se obtienen con la configuración de 8 computadores, ya que con esta configuración se reducen los costos.

El mejor costo y menor tiempo de ejecución en estas pruebas, es generado por la configuración de 8 computadores, 270 ISA y 300 IMC. Hay que resaltar que al igual que en la configuración en la que se dividen iteraciones, la configuración que proporciona el mejor costo está conformada por 8 ordenadores.

La ruta obtenida por las diferentes configuraciones de la malla mejoran en un 8,7% el costo y en un 78,7% el tiempo de ejecución para la configuración sin división de iteraciones. Estos resultados implican que el tiempo de ejecución respecto a la mejor respuesta obtenida en las pruebas en las que se dividen las iteraciones, es mejorado aproximadamente en un 7 % y que para los costos el porcentaje de mejora se mantiene.

4.2.2.2. **Para 64 nodos.** Con este tipo de paralelización, se obtuvieron 6 resultados que se muestran en la Tabla 4-10.

Tabla 4-10 Mejores costos obtenidos.

	Costo Final	Tiempo [Min]
1	1.839	25,49
2	1.845	26,53
3	1.857	34,39
4	1.874	2,47
5	1.881	4,97
6	1.890	6,58

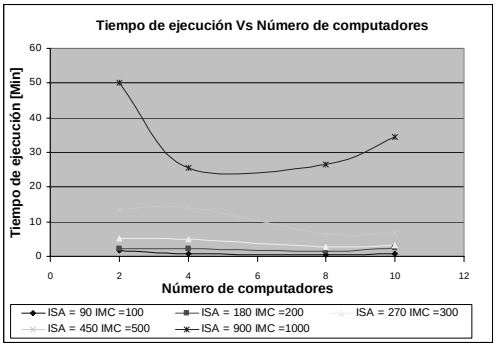

Figura 4-14 Tiempo de ejecución vs. Número de computadores para 64 nodos sin dividir.

En esta configuración de malla los tiempos son más pequeños a medida que se incrementa el número de computadores. Sin embargo la reducción máxima de tiempo, se encuentra cuando se utilizan entre 4 y 8 máquinas (Ver Figura 4-14).

En la Tabla 4-11 se observa que la ruta obtenida por las diferentes configuraciones de la malla mejoran en un 3,0% el costo, con una mejora del 86.1% en el tiempo de ejecución.

Tabla 4-11 Comparación de costos y tiempo obtenidos.

	Costo	Tiempo [Min]
Mejor Respuesta obtenida	1.874,00	2,47
Mejor Respuesta Tesis César Niño	1.932,30	17,80
% de mejora	3,0%	86,1%

5. CONCLUSIONES

La paralelización de los algoritmos utilizados para implementar las metaheurísticas utilizadas para resolver los problemas de optimización combinatoria NP-Hard, es un gran avance encaminado a la búsqueda de la solución óptima mientras sea acompañada por las herramientas computacionales indicadas, ya que, al realizar de forma secuencial estos procedimientos, el incremento de las iteraciones realizadas dentro del algoritmo implica más tiempo de ejecución de las tareas.

La computación en malla se perfila como la solución tecnológica más adecuada para la ejecución de los algoritmos utilizados para encontrar las mejores soluciones a los problemas de enrutamiento vehicular *(VRP)*, por su esencia distribuida, su capacidad de ejecución paralela y su velocidad de sincronización.

El objetivo de las pruebas es verificar si el efecto malla, utilizando SGE, realmente se cumple en este tipo de problemas de optimización combinatoria. Además se quiere comprobar que SGE es un grid lo suficientemente robusto para ser exigido por grandes aplicaciones que requieran una gran capacidad de procesamiento y almacenamiento y por último lograr estabilizar SGE dentro de los laboratorios de Sistemas para realizar montajes futuros.

6. REFERENCIAS

[1] Niño Méndez, C (2006): HARITI proyecto para el enrutamiento de transporte escolar.
[2] Bertis, V. (2002): Fundamentals of Grid Computing.
[3] Aarts, E.H.L, Korst, JH.M.(1989): Simulated annealing and boltzman machines, Wiley, Chichester.
[4] Sun Microsystems, Inc.: N1 Grid Engine 6 User's Guide.
[5] Kathry A. Dowsland, Belarmino Adenso Díaz: Huristic desing and Fundamentals of the Simulated Annealing.
[6] Zbigniew J. Czech: Parallel simulated annealing for the vehicle routing problem with time windows.
[7] Sun Microsystems, Inc.: N1 Grid Engine 6 Administration Guide