ISSN 0717-9103
ISSN Online 0718-8307

ESTRATEGIAS DE PARALELIZACIÓN DE METAHEURÍSTICAS APLICADAS A PROBLEMAS DE LOCALIZACIÓN DE INSTALACIONES*

PARALLELIZATION STRATEGIES FOR METAHEURISTICS APPLIED TO FACILITIES LOCATION PROBLEMS

Armin Lüer Villagra[1] ,Bárbara Venegas Quintrileo[1],Jaime Bustos Gómez[1]
[1]Universidad de La Frontera, Facultad de Ingeniería, Ciencias y Administración, Departamento de Ingeniería de Sistemas. Avda. Francisco Salazar 01145. Temuco. Chile.

RESUMEN

El objetivo de este trabajo es poner a prueba distintos enfoques de paralelización (sincrónico, cooperativo, centralizado, etc.) para las metaheurísticas empleadas, en los algoritmos de resolución del problema de las p-medianas. Un análisis posterior arrojó: limitaciones en la cantidad de procesadores utilizados debido a la tecnología de red, ventajas en el uso de multiprocesadores y la penalización en el rendimiento al emplear estrategias intensivas en comunicación. Además, en la optimización mediante heurísticas siempre está implícito un balance entre la velocidad en la obtención de una solución y la calidad de ésta, por lo que se sugiere el uso de estrategias híbridas, que aprovechan lo mejor de cada una de ellas.

Palabras Clave: Paralelización, metaheurísticas, p-mediana.

ABSTRACT

The aim of this paper is to analyze the results of the implementation of several parallelization techniques for the metaheuristics applied to the p-median problem. Further analysis showed: constraints on the number of processors used because of the network technology, advantages in the use of multiprocessors and the penalty in performance when using intensive communications strategies. Moreover, within the optimization using heuristics there is always an implicit balance between speed in obtaining a solution and the quality of it, which suggests the use of hybrid strategies that exploit the best of each.

Keywords: Parallelization techniques, metaheuristics, p-median problem.

*Este trabajo fue presentado en el VIII Congreso Chileno de Investigación Operativa, realizado por la Universidad del Bio-Bio en la ciudad de Chillán en Octubre del año 2009

Autor para correspondencia: arminluer@gmail.com

Recibido: 13.04.2009 Aceptado: 19.11.2009

INTRODUCCIÓN

Las técnicas de paralelización de algoritmos son metodologías actualmente usadas para la resolución de problemas de gran escala y alta complejidad computacional. Entre estos problemas se cuentan los NP-Hard de optimización combinatorial, que debido a la imposibilidad de encontrar las soluciones óptimas (pues supone tiempos de proceso excesivamente largos) son resueltos con heurísticas en búsqueda de una buena solución aproximada.

El Problema de la P-Mediana

Consideraremos el problema de la p-mediana, común en el ámbito de la logística, en donde se asume que el costo de operación es proporcional a la distancia entre un proveedor y un cliente, además de la demanda de este último. De acuerdo con Kariv & Hakimi (1979) pertenece al conjunto NP-Hard. Genéricamente se trata de una red de nodos y arcos, en donde cada nodo representa un foco de demanda y una posible locación para una instalación. El objetivo es ubicar exactamente p instalaciones, minimizando la distancia total, a la vez que se satisfacen todas las demandas.
Suponiendo que no existen restricciones de capacidad, la formulación del problema como un modelo de programación lineal entera (binaria) mixta, detallada por Daskin (1995), es como sigue:

n: cantidad de clientes (nodos)

i,j : índices referenciales de un nodo.

d_{ij}: distancia más corta entre los nodos i y j

h_i: demanda del nodo i

$$y_{ij} = \begin{cases} 1\text{: si la demanda del nodo i se asigna a la instalación j} \\ 0\text{: en cualquier otro caso} \end{cases}$$

$$x_j = \begin{cases} 1\text{:si una instalación es ubicada en j y la demanda de j es asignada a ésta} \\ 0\text{:en cualquier otro caso} \end{cases}$$

p: cantidad de instalaciones que deben ser ubicadas

Utilizando esta notación podemos describir el modelo de la siguiente manera:

$$\textit{Minimizar} \quad Z = \sum_{i=1}^{n}\sum_{j=1}^{n} h_i d_{ij} y_{ij} \tag{1}$$

$$\textit{S.A.} \quad \sum_{j=1}^{n} y_{ij} = 1 \tag{2}$$

$$\sum_{j=1}^{n} x_j = p \tag{3}$$

$$y_{ij} - x_i \le 0, \quad i \ne j \tag{4}$$

$$y_{ij} \in \{0,1\} \quad (5)$$

$$x_i \in \{0,1\} \quad (6)$$

$$\forall\ i = 1,\ldots,n;\ j = 1,\ldots,n \quad (7)$$

La función objetivo a minimizar es la suma de las distancias asociadas a las demandas de cada nodo (1). La restricción (2) apunta a que sólo sea asignada una instalación a cada nodo. La restricción (3) asegura que se localicen p instalaciones. La restricción (4) permite que se asignen a los nodos sólo las instalaciones que efectivamente serán abiertas. Las expresiones (5) a (7) acotan el dominio de las variables del modelo.

Algoritmos de Resolución del Problema de las P-Medianas

De acuerdo con Alba & Domínguez (2006) los enfoques clásicos que existen para solucionar este problema son la relajación langrangiana usando branch and bound, búsqueda local y tabú. En este sentido, existen numerosos algoritmos de solución que van desde la búsqueda del óptimo haciendo simplificaciones del modelo como es el caso de COBRA, propuesto por Church (2003), hasta la aplicación de (meta-)heurísticas como algoritmos genéticos, recocido simulado, herramientas de clustering, etc.

Alba & Domínguez (2006) muestran, como herramientas actuales, los algoritmos evolutivos, redes neuronales y búsqueda dispersa, en donde la principal característica se refiere a la mutación y mejora de poblaciones de soluciones, como se describe más adelante. Así también lo proponen Hansen & Mladenovic (2008), agregando además, como una buena estrategia, la búsqueda en vecindarios variables (VNS).

Berkhin (2002) resume un conjunto de herramientas de clustering donde se divide el conjunto de datos, en este caso con partición por locación como clustering probabilístico, k-Medoids y k-Means. Este último es ampliamente usado en el ámbito científico y en aplicaciones industriales por su simplicidad y fácil implementación.

Algoritmos Evolutivos

Estos algoritmos actúan como guía en búsquedas aleatorias, emulando la evolución biológica; de ahí que también sean denominados como algoritmos genéticos, y que la nomenclatura que se utilice referencie a términos usados en la genética.

Se trabaja con una población donde cada individuo (cromosoma) representa una posible solución. El algoritmo avanza conforme el conjunto de soluciones cambia, encontrando nuevos individuos que resultan de la interacción de otros dentro de la misma población, a través de los mecanismos de selección, donde se escogen los cromosomas padres para luego cruzarlos, mutación donde se alteran las soluciones para crear la diversidad genética y recombinación, que le permite a la población evolucionar hacia nuevas regiones.

La forma de aplicar este concepto a la solución del problema de la p-mediana, según Alba & Domínguez (2004), es considerar a cada cromosoma de la población como un conjunto de exactamente p componentes y donde la función objetivo a evaluar es la suma de las distancias ponderadas.

Búsqueda Dispersa

Es un algoritmo basado también en poblaciones empleando una componente, como conjunto de referencia fundamental, donde se almacenan soluciones buenas y dispersas. La forma de construir nuevas soluciones es combinando elementos de este conjunto de referencia obteniendo mejoras y actualizando el conjunto de acuerdo con el avance que se experimente. García *et al*. (2003) propone un método de esta naturaleza, en donde el conjunto de referencia antes mencionado se genera a partir de una población de soluciones, para luego seleccionar un subconjunto que se combinará para conseguir soluciones iniciales. Posteriormente se hace una búsqueda local, donde la mejora resultante puede terminar con una actualización del conjunto de referencia e, incluso, de la población de soluciones.

Búsqueda en Vecindarios Variables

Trabaja haciendo una búsqueda local dentro de los vecindarios de un punto dado, cuya definición se va actualizando a medida que la solución mejora. Se funda en tres principios básicos: el óptimo para un vecindario no es necesariamente un óptimo para otro; un óptimo global será un óptimo para cualquier vecindario; y, basándose en resultados obtenidos, se prueba que todos los óptimos locales están cerca el uno del otro.

Según Hansen & Mladenovic (2008), inicialmente se definen el número máximo de vecindarios ($k_{máx}$) que se incluirán en la búsqueda, la estructura de dichos vecindarios y una solución inicial. Funciona de manera iterativa, explorando vecindarios cada vez mayores, mediante búsqueda local. Si se encuentra una mejor solución, se reinicia la búsqueda en ésta. En caso contrario, se continúa hasta que se cumpla una condición de parada.

K-Medoids o K-Vecinos

Berkhin (2002) la describe como una estrategia de clustering, en donde se seleccionan elementos representativos (medoids) dentro del conjunto de datos. Posteriormente se divide el conjunto en clusters, estando cada uno conformado por el subconjunto de puntos más cercanos a cada uno de los elementos. Luego, estos últimos se van cambiando por otro del conjunto en la medida en que mejoran la solución. La función objetivo queda definida como la distancia promedio u otra medida de diferenciación entre un punto y su correspondiente medoid.

K-Means

El algoritmo parte definiendo centros para los futuros cluster a formar, asignando a cada punto el centro más cercano. Una vez que se tiene un estado inicial o asignación para los datos, el algoritmo avanza conforme se van calculando los nuevos centros dentro de un cluster, como el centro de masa de un mismo conjunto. Actúa de manera iterativa, calculando y asignando centros cada vez, en donde el criterio de parada más común es la no actualización de las asignaciones de cada uno de los puntos. Una variante de este algoritmo mejora la semilla con la que parte, eliminando el factor aleatorio; un ejemplo de ello es la idea propuesta por Arthur & Vassilvitskii (2007), en donde se elige un primer centro al azar desde el conjunto de datos, y luego el resto de los centros es seleccionado, uno a uno, en torno a la proporción de la distancia hacia el punto que tiene asignado con la suma de todas las distancias de cada punto al centro que le corresponde, denominado potencial.

Paralelización

La computación paralela se basa en el uso compartido y simultáneo de recursos, en donde la arquitectura de un sistema de este tipo está conformada por uno o varios computadores. Cuando

nos referimos a un único centro de cómputo, se trata de un mismo computador con múltiples procesadores, y cuando hablamos de dos o más, tratamos sobre una red de computadores conectada intercambiando información. Estas dos alternativas son perfectamente combinables en los enfoques mixtos que serán descritos más adelante en este artículo.

Al diseñar un algoritmo paralelo, lo que se busca es ejecutar tareas pequeñas que son el resultado de la descomposición de los problemas. Esta división puede realizarse seccionando la memoria, donde a cada tarea le corresponde una porción de los datos, o bien se divide el trabajo a realizar en tareas; así, cada una es un conjunto de instrucciones y funciones.

Para asegurar una ejecución eficiente del algoritmo paralelo es necesario distribuir las tareas de la manera óptima. Si esto no se logra, podrían ocurrir problemas como, por ejemplo, la no sincronización de las tareas o el mal uso del tiempo. Un buen balance de carga puede llegar a incrementar de forma importante el rendimiento de estos algoritmos, especialmente en entornos heterogéneos.

En el momento de diseñar programas o algoritmos paralelos se deben tomar decisiones con respecto a la arquitectura de la memoria que utilizará el sistema, la metodología de obtención de soluciones, el balance de carga, etc.

Arquitectura de Memoria

Cuando nos referimos a la arquitectura de memoria hablamos sobre cómo es el acceso y almacenamiento de datos dentro del conjunto. En este sentido podemos clasificarla en:

- Memoria Compartida: existe un espacio global de direccionamiento, donde cada uno de los procesadores puede acceder a la memoria utilizando este espacio. En este sentido, los procesadores, aun cuando actúan independientemente, utilizan los mismos recursos y una alteración realizada por cualquiera de ellos es completamente transparente para todo el resto. Una de las ventajas de utilizar esta metodología es el rápido acceso a los datos; pero si, consideramos un aumento en la cantidad de procesadores, el tráfico generado puede ralentizar el intercambio de información.

- Memoria Distribuida: los procesadores involucrados actúan de forma independiente y con una memoria local. En este caso, si un procesador necesita una cantidad de datos que se encuentran contenidos en la memoria de otro de los procesadores, precisa recibirla a través de mensajes. El usar esta arquitectura de memoria, elimina las interferencias que se producen en la memoria compartida, aumentando la rapidez de acceso a los datos.

- Memoria Compartida - Distribuida: se trata de un clúster o de una división de procesadores, donde se cuenta con el acceso a una memoria compartida, pero sin hacer uso de un canal compartido. Cada procesador se conecta a un dispositivo de alta velocidad, obteniéndose como resultado que el cada procesador vea la memoria de cada uno si se tratase de un espacio de direccionamiento global.

Algoritmos Paralelos

Cada uno de los algoritmos anteriormente descritos para la p-mediana, supone la utilización de un alto poder de cómputo en la medida en que la escala de los problemas va *in crescendo*, y en este sentido Guerriero & Mancini (2005) sostienen que la paralelización de estas estrategias puede facilitar ampliamente la tarea.

Las técnicas de paralelización pueden ser aplicadas a algoritmos exactos, metaheurísticas,

estrategias híbridas y también a la optimización multiobjetivo. Centraremos nuestra atención en metaheurísticas, como los algoritmos evolutivos, la búsqueda dispersa o la búsqueda en vecindarios variables. Algunas de ellas tienen una naturaleza paralela intrínseca y otras precisan de un análisis que permita dividir la resolución en instrucciones paralelas. Pardalos *et al.* (1995) muestra que, en el primer caso, los algoritmos genéticos poseen esta naturaleza, y es por esto que la implementación de ellos utilizando computación paralela ha resultado exitosa.
En la sección 3 retomaremos este tema para referirnos a la paralelización de las metaheurísticas implementadas para el análisis y comparación de resultados.

METODOLOGÍA

Este trabajo se funda en dos partes fundamentales: la elección de los algoritmos a utilizar, con su consecuente implementación, y la forma en que estos se ejecutan haciendo uso de computación paralela. La combinación de ambos elementos se explicita a continuación.

Implementación de Algoritmos

Para este análisis se utilizarán estrategias de búsqueda en vecindarios variables y la implementación de un algoritmo genético que será alimentado con soluciones generadas a través de las heurísticas mencionadas.

Búsqueda en Vecindarios Variables (VNS)

En este caso se implementó la Búsqueda en Vecindarios Variables (VNS) con mejoras en la búsqueda local propuestas por Resende & Werneck (2004), además de una forma reducida de esta metaheurística (RVNS).

Considerando la notación $N_k(x)$ para referirse al vecindario k de una solución x dada, el algoritmo funciona como sigue:

1) Inicialización:
   - Se encuentra una solución inicial x.
   - Sea $k \leftarrow 1$.
   - Se establece una condición de parada.
2) Repetir los siguientes pasos hasta que la condición de parada se cumpla, o bien se alcance el vecindario mayor para buscar (k_{max}):
   - Proceso de agitación: Sea $x' \leftarrow x$. Se genera una solución de partida x'' desde $N_k(x')$.
   - Búsqueda Local: Se aplica una búsqueda local desde la solución de partida x'' usando el vecindario $N_k(x'')$ hasta que se encuentra un óptimo local x^*.
   - Si existe mejora: Si x^* es mejor que x, entonces $x \leftarrow x^*$ y $k \leftarrow 1$. En cualquier otro caso $k \leftarrow k+1$.

La diferencia entre este algoritmo y la búsqueda en vecindarios variables reducida (RVNS) es que no se realiza una búsqueda local a partir del punto seleccionado. Considerando esto, el algoritmo se ejecuta de la siguiente forma:

1) Inicialización:

   - Se selecciona un conjunto finito de vecindarios que serán considerados para la búsqueda: N_k con $k = 1,2,...,k_{\max}$.
   - Se encuentra una solución inicial x.
   - Se establece una condición de parada.

2) Repetir los siguientes pasos hasta que la condición de parada se cumpla:

   - Repetir los siguientes pasos hasta que $k = k_{\max}$
     - Proceso de agitación: Sea $x' \leftarrow x$. Se genera una solución de partida x'' desde $N_k(x')$.
     - Cambio o no: Si $x*$ es mejor que x, entonces $x \leftarrow x*$ y $k \leftarrow 1$. En cualquier otro caso $k \leftarrow k+1$.

Algoritmo Genético

Se implementó el algoritmo genético propuesto por Alp et al. (2003), donde se consideran los siguientes conceptos:

- En una posible solución (cromosoma) los genes corresponden a los nodos escogidos para ser medianas del problema.
- La calidad de la solución (función de fitness) corresponde a la función objetivo.
- El tamaño de la población se basa en dos características:
  - Debe contener todos los genes posibles para que la búsqueda en el espacio factible no sea parcial.
  - Hay una relación entre el tamaño de la población y el número de soluciones. Así, entre mayor sea el conjunto factible del problema, más difícil se torna el encontrar una solución mejor.

Ocupando estos principios y con alteraciones al tamaño de población propuesto por Alp *et al*. (2003), el algoritmo es el siguiente:

1) Se genera una población inicial cuyo tamaño está en función de la cantidad de nodos y medianas del problema a considerar, como lo muestra la expresión (8):

$$\frac{10 \cdot \log(n \cdot p)}{\sqrt{d}}, \qquad d = \frac{n}{p} \tag{8}$$

2) Inicializar MaxIter, que servirá para establecer un criterio de parada que corresponde a la cantidad de iteraciones sin éxito a considerar.

3) Repetir los siguientes pasos hasta que MaxIter < $\left\lceil n\sqrt{p} \right\rceil$

   - Seleccionar al azar dos miembros de la población actual.
   - Ejecutar el operador de generación utilizando los dos miembros escogidos en el punto anterior para el cruzamiento, que los une en una solución infactible que a través de un algoritmo voraz se transformará en un miembro candidato.
   - Ejecutar el operador de reemplazo utilizando el miembro candidato, determinando si debe entrar a la población en función de su fitness (si es mejor que el peor actual).

- Si la mejor solución encontrada no ha mejorado hacer MaxIter ← MaxIter + 1

2) Seleccionar el mejor miembro encontrado como la solución final.

Estrategias de Paralelización Empleadas

Consideraremos la paralelización del algoritmo genético con poblaciones generadas por búsqueda en vecindarios variables reducida. También se implementó una forma paralela para la búsqueda en Vecindarios Variables, utilizando algunas de las mejoras en la búsqueda local propuestas por Resende & Werneck (2007).
Para el primer caso emplearemos dos estrategias, cuya diferencia radica en la variabilidad de poblaciones que se ocuparán:

- Cada uno de los procesadores crea una población a partir de las soluciones generadas por RVNS. Una vez que calculan la mejor solución, la retornan al procesador maestro. El tamaño de cada población viene dado por la ecuación (8).

- En la segunda estrategia es el procesador maestro el único que crea una población, también a partir de soluciones generadas por RVNS. Cada procesador trabaja con esa población, retornando al maestro la mejor solución hallada. El tamaño de la población está dado por (8).

Para el segundo caso, para hacer la búsqueda en vecindarios variables, el procesador maestro genera la solución inicial a partir de la cual cada procesador inicia la búsqueda, pero en direcciones distintas. Se utilizó un criterio de parada correspondiente a 100 iteraciones sin éxito.

RESULTADOS COMPUTACIONALES

Para la comparación se utilizó la OR-Library de Beasley (1990), que consta de 40 problemas de la p-Mediana, con entre 100 y 900 nodos de demanda, y entre 5 y 200 medianas a localizar.

Resultados Algoritmos Secuenciales

Se empleó una estación de trabajo, con procesador Intel E2180 de 2 GHz, 1GB de RAM y Ubuntu Server 8.04 LTS. Se corrieron 5 réplicas de cada algoritmo, para cada instancia.

Para el caso del algoritmo genético en forma secuencial se obtuvo un total de 26 óptimos con un error promedio para todas las instancias de 0,058% como se muestra en la Tabla N°1. En el caso de VNS (ver Tabla N°2), se lograron 24 óptimos con un error promedio de 0,063%.

Resultados Algoritmos Paralelos

Para la ejecución se utilizó un cluster de 15 nodos, cada uno con un procesador Intel Pentium 4 de 3,06 GHz HT, 1 GB de RAM y Rocks 5.1 como Sistema Operativo, usando MPI.

En la ejecución del algoritmo genético con población distinta para cada procesador y generada a través de RVNS se obtuvo un total de 28 óptimos con un error promedio del 0,046% (Tabla N°3) en contraste con los resultados al utilizar una población común, donde se llegó a 24 óptimos con un error promedio para todas las instancias de 0,060% (Tabla N°4).

Para el último caso, Búsqueda en Vecindarios Variables, se obtuvieron 29 óptimos con un error promedio para todas las instancias de 0,043% (Tabla N°5), con un tiempo de ejecución promedio de 3700 segundos.

ISSN 0717-9103
ISSN Online 0718-8307

Tabla N°1. Resultados ejecución Algoritmo Genético secuencial, población inicial generada con RVNS.

Instancia				Algoritmo Genético con RVNS Secuencial		
Nombre	**n**	**p**	**Óptimo**	**Solución Final**	**Error**	**Tiempo (s)**
pmed1	100	5	5819	5819	0,00%	0,38
pmed2	100	10	4093	4093	0,00%	0,69
pmed3	100	10	4250	4250	0,00%	0,74
pmed4	100	20	3034	3034	0,00%	2,28
pmed5	100	33	1355	1355	0,00%	5,33
pmed6	200	5	7824	7824	0,00%	0,51
pmed7	200	10	5631	5631	0,00%	2,04
pmed8	200	20	4445	4445	0,00%	5,06
pmed9	200	40	2734	2734	0,00%	23,29
pmed10	200	67	1255	1255	0,00%	65,06
pmed11	300	5	7696	7696	0,00%	0,84
pmed12	300	10	6634	6634	0,00%	3,02
pmed13	300	30	4374	4374	0,00%	27,50
pmed14	300	60	2968	2968	0,00%	118,65
pmed15	300	100	1729	1730	0,06%	377,76
pmed16	400	5	8162	8162	0,00%	1,16
pmed17	400	10	6999	7009	0,14%	6,71
pmed18	400	40	4809	4809	0,00%	129,79
pmed19	400	80	2845	2848	0,11%	807,25
pmed20	400	133	1789	1789	0,00%	1318,05
pmed21	500	5	9138	9138	0,00%	1,97
pmed22	500	10	8579	8579	0,00%	12,25
pmed23	500	50	4619	4623	0,09%	479,35
pmed24	500	100	2961	2969	0,27%	2183,80
pmed25	500	167	1828	1835	0,38%	8054,98
pmed26	600	5	9917	9917	0,00%	3,40
pmed27	600	10	8307	8307	0,00%	21,06
pmed28	600	60	4498	4499	0,02%	1033,84
pmed29	600	120	3033	3039	0,20%	4224,18
pmed30	600	200	1989	2000	0,55%	7667,00
pmed31	700	5	10086	10086	0,00%	4,82
pmed32	700	10	9297	9297	0,00%	33,91
pmed33	700	70	4700	4703	0,06%	1879,31
pmed34	700	140	3013	3018	0,17%	5701,00
pmed35	800	5	10400	10400	0,00%	7,09
pmed36	800	10	9934	9944	0,10%	44,96
pmed37	800	80	5057	5060	0,06%	2759,26
pmed38	900	5	11060	11060	0,00%	9,17
pmed39	900	10	9423	9423	0,00%	45,58
pmed40	900	90	5128	5134	0,12%	4393,51
Promedios					0,058%	1036,41

Tabla N°2. Resultados ejecución VNS secuencial, con soluciones iniciales aleatorias.

Instancia				VNS Secuencial		
Nombre	**n**	**p**	**Óptimo**	**Solución Final**	**Error**	**Tiempo (s)**
pmed1	100	5	5819	5819	0,00%	0,78
pmed2	100	10	4093	4093	0,00%	1,00
pmed3	100	10	4250	4250	0,00%	1,15
pmed4	100	20	3034	3038	0,13%	2,28
pmed5	100	33	1355	1355	0,00%	4,94
pmed6	200	5	7824	7824	0,00%	241,59
pmed7	200	10	5631	5631	0,00%	4,15
pmed8	200	20	4445	4445	0,00%	10,94
pmed9	200	40	2734	2734	0,00%	32,76
pmed10	200	67	1255	1256	0,08%	61,96
pmed11	300	5	7696	7696	0,00%	8,23
pmed12	300	10	6634	6634	0,00%	12,58
pmed13	300	30	4374	4374	0,00%	53,34
pmed14	300	60	2968	2974	0,20%	116,27
pmed15	300	100	1729	1734	0,29%	631,75
pmed16	400	5	8162	8162	0,00%	19,09
pmed17	400	10	6999	6999	0,00%	22,62
pmed18	400	40	4809	4811	0,04%	154,76
pmed19	400	80	2845	2848	0,11%	909,04
pmed20	400	133	1789	1792	0,17%	623,78
pmed21	500	5	9138	9138	0,00%	13,31
pmed22	500	10	8579	8579	0,00%	38,61
pmed23	500	50	4619	4619	0,00%	461,74
pmed24	500	100	2961	2966	0,17%	2446,38
pmed25	500	167	1828	1836	0,44%	2084,84
pmed26	600	5	9917	9917	0,00%	50,69
pmed27	600	10	8307	8307	0,00%	64,02
pmed28	600	60	4498	4503	0,11%	763,19
pmed29	600	120	3033	3036	0,10%	1427,31
pmed30	600	200	1989	1990	0,05%	4847,68
pmed31	700	5	10086	10086	0,00%	16602,37
pmed32	700	10	9297	9297	0,00%	45,33
pmed33	700	70	4700	4708	0,17%	1971,48
pmed34	700	140	3013	3019	0,20%	3273,65
pmed35	800	5	10400	10400	0,00%	7114,71
pmed36	800	10	9934	9934	0,00%	191,79
pmed37	800	80	5057	5060	0,06%	4034,27
pmed38	900	5	11060	11060	0,00%	33081,41
pmed39	900	10	9423	9423	0,00%	133,44
pmed40	900	90	5128	5139	0,21%	10464,45
Promedios					0,063%	2300,59

ISSN 0717-9103
ISSN Online 0718-8307

Tabla N°3. Resultados Ejecución del Algoritmo Genético con poblaciones distintas generadas en cada nodo mediante RVNS.

Instancia				Algoritmo Genético con RVNS, Paralelo		
Nombre	**n**	**p**	**Óptimo**	**Solución Final**	**Error**	**Tiempo (s)**
pmed1	100	5	5819	5819	0,00%	0,28
pmed2	100	10	4093	4093	0,00%	0,64
pmed3	100	10	4250	4250	0,00%	0,54
pmed4	100	20	3034	3034	0,00%	2,10
pmed5	100	33	1355	1355	0,00%	5,34
pmed6	200	5	7824	7824	0,00%	0,50
pmed7	200	10	5631	5631	0,00%	1,48
pmed8	200	20	4445	4445	0,00%	4,91
pmed9	200	40	2734	2734	0,00%	20,32
pmed10	200	67	1255	1255	0,00%	73,51
pmed11	300	5	7696	7696	0,00%	0,92
pmed12	300	10	6634	6634	0,00%	2,41
pmed13	300	30	4374	4374	0,00%	16,66
pmed14	300	60	2968	2968	0,00%	76,63
pmed15	300	100	1729	1730	0,06%	259,23
pmed16	400	5	8162	8162	0,00%	1,47
pmed17	400	10	6999	6999	0,00%	4,36
pmed18	400	40	4809	4809	0,00%	151,52
pmed19	400	80	2845	2847	0,07%	608,58
pmed20	400	133	1789	1789	0,00%	1580,86
pmed21	500	5	9138	9138	0,00%	2,00
pmed22	500	10	8579	8579	0,00%	14,11
pmed23	500	50	4619	4619	0,00%	451,35
pmed24	500	100	2961	2966	0,17%	2310,14
pmed25	500	167	1828	1837	0,49%	6600,06
pmed26	600	5	9917	9917	0,00%	3,86
pmed27	600	10	8307	8307	0,00%	20,54
pmed28	600	60	4498	4499	0,02%	1292,33
pmed29	600	120	3033	3039	0,20%	6290,81
pmed30	600	200	1989	1998	0,45%	11040,29
pmed31	700	5	10086	10086	0,00%	5,18
pmed32	700	10	9297	9297	0,00%	47,56
pmed33	700	70	4700	4702	0,04%	3037,62
pmed34	700	140	3013	3017	0,13%	9354,74
pmed35	800	5	10400	10400	0,00%	7,22
pmed36	800	10	9934	9941	0,07%	59,96
pmed37	800	80	5057	5060	0,06%	4945,94
pmed38	900	5	11060	11060	0,00%	14,77
pmed39	900	10	9423	9423	0,00%	85,73
pmed40	900	90	5128	5131	0,06%	8526,14
Promedios					0,046%	1423,07

Tabla N°4. Resultados Ejecución Algoritmo Genético con población centralizada generada mediante RVNS.

Instancia				Algoritmo Genético con RVNS, Paralelo		
Nombre	**n**	**p**	**Óptimo**	**Solución Final**	**Error**	**Tiempo (s)**
pmed1	100	5	5819	5819	0,00%	5,08
pmed2	100	10	4093	4093	0,00%	5,44
pmed3	100	10	4250	4250	0,00%	5,48
pmed4	100	20	3034	3034	0,00%	6,72
pmed5	100	33	1355	1355	0,00%	8,71
pmed6	200	5	7824	7824	0,00%	5,42
pmed7	200	10	5631	5631	0,00%	5,86
pmed8	200	20	4445	4445	0,00%	7,38
pmed9	200	40	2734	2734	0,00%	18,39
pmed10	200	67	1255	1255	0,00%	63,71
pmed11	300	5	7696	7696	0,00%	5,69
pmed12	300	10	6634	6648	0,21%	6,56
pmed13	300	30	4374	4374	0,00%	14,71
pmed14	300	60	2968	2968	0,00%	77,43
pmed15	300	100	1729	1730	0,06%	189,31
pmed16	400	5	8162	8162	0,00%	6,21
pmed17	400	10	6999	7009	0,14%	7,62
pmed18	400	40	4809	4809	0,00%	83,25
pmed19	400	80	2845	2847	0,07%	667,98
pmed20	400	133	1789	1789	0,00%	1400,00
pmed21	500	5	9138	9138	0,00%	6,32
pmed22	500	10	8579	8579	0,00%	11,92
pmed23	500	50	4619	4623	0,09%	377,30
pmed24	500	100	2961	2964	0,10%	2043,61
pmed25	500	167	1828	1835	0,38%	5150,27
pmed26	600	5	9917	9917	0,00%	6,97
pmed27	600	10	8307	8307	0,00%	16,10
pmed28	600	60	4498	4499	0,02%	977,59
pmed29	600	120	3033	3039	0,20%	6093,68
pmed30	600	200	1989	2000	0,55%	8222,75
pmed31	700	5	10086	10087	0,01%	8,03
pmed32	700	10	9297	9297	0,00%	41,69
pmed33	700	70	4700	4701	0,02%	3108,97
pmed34	700	140	3013	3017	0,13%	10026,81
pmed35	800	5	10400	10400	0,00%	10,14
pmed36	800	10	9934	9963	0,29%	41,76
pmed37	800	80	5057	5060	0,06%	4851,36
pmed38	900	5	11060	11060	0,00%	13,50
pmed39	900	10	9423	9423	0,00%	68,66
pmed40	900	90	5128	5131	0,06%	6853,90
Promedios					0,060%	1263,06

ISSN 0717-9103
ISSN Online 0718-8307

Tabla N°5. Resultados de la ejecución de Búsqueda en Vecindarios Variables Paralela, con soluciones iniciales aleatorias.

Instancia				VNS Paralelo		
Nombre	**n**	**p**	**Óptimo**	**Solución Final**	**Error**	**Tiempo (s)**
pmed1	100	5	5819	5819	0,00%	0,77
pmed2	100	10	4093	4093	0,00%	1,30
pmed3	100	10	4250	4250	0,00%	1,00
pmed4	100	20	3034	3034	0,00%	4,59
pmed5	100	33	1355	1355	0,00%	4,26
pmed6	200	5	7824	7824	0,00%	4,59
pmed7	200	10	5631	5631	0,00%	3,88
pmed8	200	20	4445	4445	0,00%	22,45
pmed9	200	40	2734	2734	0,00%	24,62
pmed10	200	67	1255	1255	0,00%	63,98
pmed11	300	5	7696	7696	0,00%	63,98
pmed12	300	10	6634	6634	0,00%	11,78
pmed13	300	30	4374	4374	0,00%	63,18
pmed14	300	60	2968	2968	0,00%	211,60
pmed15	300	100	1729	1729	0,00%	663,23
pmed16	400	5	8162	8162	0,00%	17,92
pmed17	400	10	6999	6999	0,00%	25,40
pmed18	400	40	4809	4809	0,00%	151,85
pmed19	400	80	2845	2851	0,21%	572,18
pmed20	400	133	1789	1795	0,34%	570,51
pmed21	500	5	9138	9138	0,00%	13,45
pmed22	500	10	8579	8579	0,00%	42,75
pmed23	500	50	4619	4619	0,00%	527,22
pmed24	500	100	2961	2962	0,03%	2017,21
pmed25	500	167	1828	1833	0,27%	2000,06
pmed26	600	5	9917	9917	0,00%	61,63
pmed27	600	10	8307	8307	0,00%	97,57
pmed28	600	60	4498	4500	0,04%	858,17
pmed29	600	120	3033	3036	0,10%	1386,53
pmed30	600	200	1989	1993	0,20%	4188,34
pmed31	700	5	10086	10086	0,00%	20715,65
pmed32	700	10	9297	9297	0,00%	73,31
pmed33	700	70	4700	4707	0,15%	1694,17
pmed34	700	140	3013	3019	0,20%	2840,90
pmed35	800	5	10400	10400	0,00%	7525,93
pmed36	800	10	9934	9934	0,00%	8558,29
pmed37	800	80	5057	5058	0,02%	5026,05
pmed38	900	5	11060	11060	0,00%	83198,91
pmed39	900	10	9423	9423	0,00%	119,90
pmed40	900	90	5128	5136	0,16%	4586,20
Promedios					0,043%	3700,38

CONCLUSIONES

Cada una de las estrategias utilizadas supone disponer de un poder de cómputo alto, además de ofrecer un rendimiento y comportamiento similares. Ahora, las diferencias radican en los tiempos de obtención de las soluciones y la mejora, cuando los parámetros que las delimitan son variados.

Para el caso del algoritmo genético propuesto por Alp et al. (2003), cambiando la forma en que se genera la población inicial, pasando desde la aleatoriedad a RVNS, se logró disminuir el tamaño de la población, y sin afectar al algoritmo original, en cuanto a calidad general de las soluciones.

Para los algoritmos secuenciales se encontró que tienen una desviación similar entre el Algoritmo Genético y VNS. Se observa que el menor error promedio que se obtiene utilizando estos algoritmos es de un 0,058%, siendo de un 0,043% para estrategias paralelas.

En el caso de las estrategias paralelas aplicadas al algoritmo genético, la segunda alternativa implementada que ocupa una población única generada en el procesador maestro, se demora un tiempo menor que la primera alternativa en un 11%. Pero la desviación del óptimo es un 34% superior, razón por la cual una mayor variabilidad de la población inicial se traduce en mejores soluciones. Una posible mejora para la segunda estrategia es implementar una población mayor que se encuentre centralizada; pero, por otra parte, esto aumentaría el tráfico en la red de comunicación de los procesadores con el maestro, que sólo coordinaría las acciones de los nodos.

Con respecto a los parámetros utilizados por el algoritmo genético, se podrían considerar mejoras en el criterio de parada, pues se utilizó el propuesto por Alp *et al.* (2003). Un criterio más adecuado al grado de complejidad de las instancias consideradas podría traer consigo mejoras en los tiempos de ejecución.

Otro de los parámetros a considerar es el cálculo del tamaño de población a ocupar. Para este artículo se presentaron resultados, tanto secuenciales como paralelos, en donde la población era calculada con la ecuación (8); pero también se intentó cambiando el factor 10 por 15, incrementando la cantidad de cromosomas del conjunto con mejoras en la desviación promedio del óptimo, aunque con incrementos en los tiempos de obtención de creación de la población.

El que las estrategias se comporten de un modo similar, se refiere a que las demoras en los cómputos de las soluciones crecen con respecto a la densidad de las instancias consideradas. Cuando esta densidad disminuye, el tiempo de ejecución de cualquiera de los algoritmos aumenta considerablemente. Por ejemplo, para 500 nodos y 5 medianas, tanto el Algoritmo Genético y VNS paralelos demoran un tiempo que va entre los 2 a los 14 segundos, mientras que cuando se trata también de 500 nodos, pero 167 medianas, los tiempos de ejecución son del orden de los miles de segundos.

Si se observan los resultados computacionales para todos los algoritmos expuestos en el apartado 4, la desviación del óptimo promedio más baja corresponde a la búsqueda en vecindarios variables paralela con un 0,043%, pero para lograr este porcentaje el tiempo de ejecución necesario supera en más de un 100% a las otras estrategias. Una buena alternativa para disminuir estos tiempos para el caso de VNS podría ser partir de soluciones de buena calidad generadas por otras heurísticas como el clustering de los datos del problema.

Dentro de las perspectivas futuras de este trabajo se encuentran implementar y probar combinaciones de estos algoritmos para formar nuevas estructuras híbridas, además de la formulación de criterios y parámetros más flexibles que permitan obtener soluciones con rangos de desviación similares, pero con tiempos de ejecución menores.

También se pretende seguir otros enfoques de paralelización, considerando que las estrategias actuales apuntan a la naturaleza misma de los algoritmos mediante una interrelación básica entre los procesadores, en vez de emplear técnicas más complejas que impliquen una comunicación mayor entre estos. Existe una relación que debe ser estudiada entre el nivel de comunicación entre los procesos, el tiempo de ejecución de los algoritmos y la calidad de las soluciones obtenidas.

AGRADECIMIENTOS

Al Laboratorio de Ingeniería Aplicada del Departamento de Ingeniería de Sistemas de la Universidad de La Frontera, por facilitar los equipos para probar los algoritmos secuenciales y realizar este trabajo, y a la Unidad de Bioinformática del Centro de Genómica Nutricional Agroacuícola, ubicada en la Universidad de La Frontera, por permitir utilizar su cluster para probar los algoritmos paralelos que aparecen en este trabajo.

REFERENCIAS

Alba, E. & Domínguez, E. (2006). Comparative analysis of modern optimization tools for the p-median problem. *Statistics and Computing*, 16(3), 251-260.

Alp, O., Erkut, E. & Drezner, Z. (2003). An efficient genetic algorithm for the p-Median Problem. *Anals of Operation Research,* 122(1-4), 21-42.

Arthur, D. & Vassilvitskii. S. (2007). K-Means++: The Adventages of Careful Seeding. In: SODA 2007 (pp. 1027-1035) New Orleans.

Beasley, J.E. (1990). OR-Library – Distributing Test Problems by Electronic Mail. *Journal of the Operational Research Society*, 41(11), 1069–1072.

Berkhin, P. (2002). Survey of Clustering Data Mining Techniques. Technical report, Accrue Software.

Church, R. (2003). COBRA: A new formulation of the classic *p*-Median location problem. *Anals of Operation Research*, 122(1-4), 101-120.

Daskin, M. (1995). Network and Discrete Location: Models, Algorithms and Applications. New York: John Wiley and Sons, Inc.

García, F., Melián, B., Moreno, J.A. & Moreno-Vega, J.M. (2003). Búsqueda Dispersa para problemas de localización de p-servicios con objetivos múltiples. In: II Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirado (pp. 146-152).

Guerriero F. & Mancini M. (2005). Parallelization Stratagies for Rollout Algorithms. *Computacional Optimization and Applications*, 31(2), 221-244.

Hansen, P. & Mladenovic, N. (2008). Complement to a comparative analysis of heuristics for the p-median problem. *Statistics and Computing*, 18(1), 41-46.

Kariv O., & Hakimi, S.L. (1979). An algorithmic approach to network location problems. II: The p-medians. *SIAM Journal on Applied Mathematics*, 37(3), 539-560.

Pardalos, M., Pitsoulis L., Mavridou T. & Resende, M. (1995). Parallel Search for Combinatorial Optimization: Genetic Algorithms Simulated Annealing, Tabu Search and GRASP. In: IRREGULAR 1995 (pp:317-331).

Resende, M. & Werneck, R. (2007). A fast swap-based local search procedure for location problems. *Annals of Operations Research,* 150(1), 205-230.