GRISLAS: Un algoritmo genético paralelo que combina los modelos de grillas e islas para encontrar soluciones óptimas cercanas al problema del agente viajero

GRISLAS: A parallel genetic algorithm that combines the grid and island models for finding near optima solutions to the travel salesman problem

Roberto Poveda Ch., MSc., Jonatan Gómez P., PhD. y Elizabeth León G., PhD.
Departamento de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad Nacional de Colombia - Sede Bogotá
{rpovedac, jgomezpe, eleonguz}@unal.edu.co

Recibido para revisión: 20 de Septiembre de 2007, Aceptado: 28 de Noviembre de 2008, Versión final: 9 de Diciembre de 2008

Resumen—**Este trabajo implementa un modelo de Algoritmo Genético Paralelo llamado modelo de Gris-Las, mejorado con optimización local 2-opt. Este modelo está enfocado a encontrar soluciones óptimas cercanas al problema del Agente Viajero sobre un ambiente de cómputo distribuido. Gris-Las se basa esencialmente en dos modelos comunes de Algoritmos Genéticos Paralelos, el modelo de Grillas (modelo celular) y el modelo de Islas. El algoritmo heurístico de optimización local 2-opt también es utilizado para reducir el dominio de búsqueda en el problema.**

Palabras Clave—**Algoritmos genéticos paralelos, Agente viajero, Optimización distribuida**

Abstract—**In this paper a Genetic Parallel Algorithm model is implemented in order to find optimal solutions to the well known traveling salesman problem, in a distributed computational environment. This model is called GrisLas improved with local optimization 2-opt. GrisLas is essentially based on two well-known Genetic Parallel Algorithm models: the Grid model (cellular model) and the Island model. Our heuristic local optimization 2-opt model is also used to reduce the domain for the search.**

Keywords—**Parallel genetic algorithms, Travel salesman, Distributed optimization.**

I. INTRODUCCIÓN

El problema del Agente viajero es uno de los mas importantes problemas combinatoriales así como uno de los más destacados problemas de tipo NP-Completo. El problema trata de un viajero que visita cada una de n ciudades dadas exactamente una vez y retorna a la ciudad inicial. La solución del problema consiste en hallar la secuencia de ciudades visitadas (el ciclo) que minimice la distancia total del viajero, ver [9].

El problema del agente viajero ha sido enfrentado mediante diversas técnicas, entre las que sobresalen los Algoritmos Evolutivos. Gregory Gutin en su libro, ver [12], hace una descripción profunda de muchas de estas técnicas. Los Algoritmos Genéticos (AG's) es uno de los enfoques más sobresalientes en el campo de los Algoritmos Evolutivos y se definen como procedimientos iterativos de búsqueda adaptativa de propósito general con la virtud de describir de manera abstracta y rigurosa la adaptación colectiva de una población de individuos a un ambiente particular, basándose en un comportamiento similar a un sistema natural ver [13].

Los (AG's) fueron inventados por John Holland y un grupo de estudiantes de la Universidad de Michigan inspirados en los procesos que ocurren en la evolución biológica [17]. Ellos demostraron a la vez la facilidad de implementar estos procesos adaptativos en un sistema de cómputo. Goldberg en [13] y

Michalewicz en [14] proporcionan un estudio formal en este tema.

Los AG's resultan en una inmensa mayoria de casos más robustos y eficaces que las técnicas de optimización enumerativas y basadas en el cálculo.

Los AG's son propicios para ser implementados de manera paralela en un sistema de cómputo distribuido, primero por la razón de tratar de "economizar" tiempo distribuyendo cargas de trabajo y segundo por el comportamiento natural de paralelismo sobre poblaciones espacialmente distribuidas; Tomassini en [15] destaca esta bondad.

El propósito del presente trabajo es encontrar soluciones óptimas cercanas al problema del agente viajero usando Algoritmos Genéticos Paralelos (AGP). Para esto, se consideran dos de los modelos de AGP más importantes: Islas y Grillas, así como un procedimiento de optimización de búsqueda local como es el método heurístico 2-opt. Bocki en [5] y Sengoku en [8] explican la ventaja de este método heurístico en la solución del problema del Agente Viajero.

Este artículo está dividido de la siguiente manera: la sección 2 describe conceptos preliminares (modelo de Islas, modelo de Grillas y heurística 2-opt), la sección 3 hace un recuento de algunos trabajos relacionados, la sección 4 presenta el modelo del Algoritmo Genético Paralelo de Gris-Las, la parte 5 contiene la parte de experimentación y finalmente la sección 6 presenta algunas de las conclusiones obtenidas.

II. PRELIMINARES

A. Modelo de Islas

El modelo de Islas se basa en la estructura espacialmente distribuida de las poblaciones naturales. La población se divide en subpoblaciones disjuntas donde en cada una de ellas se ejecuta en forma paralela un algoritmo genético secuencial simple con intercambios regulares de individuos entre estas subpoblaciones según una cantidad de generaciones preestablecida y algunos parámetros determinados [19]. El objetivo principal de este enfoque es reinyectar diversidad de manera periódica a subpoblaciones estancadas que tienden a converger a óptimos locales. El algoritmo del Modelo de Islas se presenta en Algoritmo 1.

Algoritmo 1. Modelo de Islas

Producir P subpoblaciones de tamaño N cada una
generación := 1
CICLO mientras no se consiga la condición de terminación
PARA cada subpoblación **HACER** en paralelo
Evaluar y seleccionar individuos por aptitud
SI [*generacion* mod *frecuencia* = 0] entonces
Enviar K ($K < N$) mejores individuos a una subpoblación vecina
Recibir K individuos de una subpoblación vecina
Reemplazar K individuos en la subpoblación
Finalizar **SI**
Producir nuevos individuos
Mutar individuos
Finalizar **HACER** en paralelo
generación := generación + 1
Final del **CICLO** mientras.

La variable frecuencia en el Algoritmo 1 representa el número de generaciones antes que se efectúe un intercambio de individuos entre subpoblaciones. Generalmente los K mejores individuos de cada subpoblación migran y reemplazan a los K peores individuos de la subpoblación vecina [15].

B. Modelo de Grillas

El modelo de Grillas (modelo celular) basa su trabajo sobre individuos distribuidos sobre una malla bidimensional, un individuo por cada celda. La interacción genética está restringida a los vecinos "cercanos" de cada individuo [19].

La forma como se realiza la selección de un individuo en la vecindad para aparearse con el individuo particular destacado en la celda, es seleccionar K individuos de la vecindad con probabilidad uniforme (pero sin reinserción) y entre ellos realizar una selección por torneo (es recomendable hacer torneo solo entre dos individuos). La selección por torneo se hace de manera determinista o probabilista. Este método de selección es ideal en este modelo de Grillas pues brinda la ventaja que no necesita ser implementado globalmente sobre toda la población, si no más bien puede ser implementado sobre subpoblaciones organizadas espacialmente. El cruce entre individuos se hace generalmente cambiando el individuo de la celda por el mejor de los hijos si este presenta mejor nivel de aptitud. El algoritmo 2 resume este modelo.

Algoritmo 2. Modelo de Grillas

Para cada celda i en la grilla **HACER** en paralelo
Generar un individuo aleatorio i
Finalizar **HACER** en paralelo
CICLO mientras no se consiga la condición de terminación
PARA cada celda i **HACER** en paralelo
Evaluar individuo i
Seleccionar un individuo k en la vecindad
Producir hijos de i y k
Asignar uno de los hijos a i
Mutar i con probabilidad *pmut*
Finalizar **PARA** en paralelo
Final del **CICLO** mientras

C. Heurística 2-opt

El algoritmo heurístico de optimización 2-opt es un algoritmo de búsqueda local para la solución del problema del agente

viajero. Bocki en [5] y Sengoku en [8] describen la bondad del método 2-opt en la solución del problema del Agente viajero. Este método mejora la trayectoria arista por arista y reversa el orden de subtrayectorias. Cada dos aristas (a,b) y (c,d) de cada trayectoria son chequeadas comprobando la posibilidad de conectar estos cuatro nodos de una manera diferente con la intención de obtener una trayectoria más corta. Se chequea si $ab + cd > ac + bd$ (xy representa la distancia del nodo x al nodo y). Si este es el caso, reemplazamos la arista (a,b) y (c,d) por las aristas (a,c) y (b,d), así como se reversa el orden de la subtrayectoria (desde b hasta c), es decir, la subtrayectoria (a,b,t,c,d,s) se optimiza, con la subtrayectoria (a,c,t,b,d,s). Asumimos que a, b, c y d aparecen en un orden específico en la trayectoria aún si b y c no están conectados. Las Figura 1 y 2 detallan tal procedimiento.

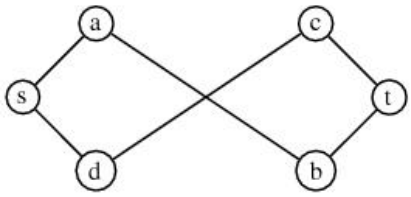

Figura 1. Procedimiento heurístico 2-opt. Trayectoria original

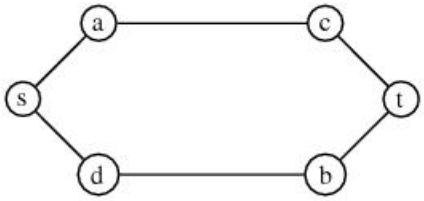

Figura 2. Procedimiento heurístico 2-opt. Trayectoria optimizada

Este procedimiento se repite hasta que ninguna mejora adicional se pueda efectuar.

III. TRABAJOS RELACIONADOS

Diversas investigaciones utilizan AG's para encontrar soluciones óptimas cercanas al problema del agente viajero de maneras diferentes, sobre todo implementando variantes en los operadores de cruce y mutación. Sin embargo, todos tratan este problema desde la misma perspectiva, es decir, considerando cada elemento de la población como una trayectoria (ciclo) de las ciudades dadas y los operadores genéticos combinan y modifican dichas trayectorias. Algunos artículos que proponen "soluciones" al problema del agente viajero mediante Algoritmos Genéticos y Algoritmos Genéticos Paralelos (AGP) son los siguientes: Lee Wang y otros en [1] comparan cinco diferentes AG's Paralelos. El estudio utiliza inicialmente un enfoque de un AG serial, estimado por P. Jog en [2] eficiente en el tratamiento del problema del agente viajero. En cada iteración del AG se consideran esencialmente tres operadores genéticos: un cruce heurístico modificado, una mutación 2-opt y una mutación O-opt modificada. El primer AG Paralelo (AGP independiente) se "paraleliza" ejecutando múltiples copias del mismo AG serial sobre cada procesador pero evolucionando las soluciones de forma independiente (sin ningún tipo de intercomunicación entre procesadores). Para este AGP se toma la mejor solución obtenida, después de las múltiples ejecuciones. Otro AGP (AGP con migración) descrito en el mismo trabajo considera migración entre procesadores. Migraciones ocurren cada 5 iteraciones y las mejores soluciones reemplazan a las peores. El tercer AGP (AGP particionado) examina sobre subespacios totalmente independientes en lugar de un examen sobre el espacio completo, este AGP proporciona las soluciones más malas en términos de calidad y tiempo, debido a que todos los procesadores examinan subespacios que no necesariamente contienen buenas soluciones. El cuarto AGP (AGP con segmentación) utiliza sobre cada procesador un AG serial donde cada trayectoria es segmentada en pequeñas sub-trayectorias, cada sub-trayectoria es mejorada y recombinada de la forma referenciada en [2] hasta obtener una trayectoria completa de longitud más corta que la inicial. El quinto AGP (AGP con segmentación - migración) considera el mismo esquema del cuarto AGP con la variante de migraciones periódicas de cromosomas. Los experimentos se hacen sobre el mismo conjunto de 50 diferentes poblaciones iniciales divididas en 16 subpoblaciones independientes cada una evolucionada por un procesador diferente. Todos los algoritmos son tratados sobre un mismo sistema de cómputo con un máximo de 150 iteraciones. El AGP con mejor desempeño es el AGP con segmentación - migración.

Chao-Xue Wang en [3] utiliza un AG basado en la terapia del gen. La investigación se basa en la idea que los seres humanos pueden utilizar medicamentos que pueden corregir defectos fisiológicos, virtud que no pueden aprovechar otros seres vivos. La evolución del AG se compara con la mejora en salud de un ser humano. Esencialmente considera un "pool" de genes donde se destacan genes prominentes, genes normales y genes malos que son almacenados en un caché que interactúa con la población del AG. El operador de terapia considerado es un operador de inserción y de remoción, donde se reemplazan genes generales por genes eminentes y a la vez genes malos por genes generales, una mutación 2-opt es utilizada para mejorar el desempeño del algoritmo. Los resultados que aquí se proporcionan son comparados con el estudio presentado en [4].

Lucas Brocki en [5] proporciona una solución cercana al problema del agente viajero aplicando un algoritmo Kohonen Self-Organizing Map y mejorándolo por medio de heurística 2-opt. El Kohonen Self-Organizing Map por si solo es un buen algoritmo con el cual se obtiene soluciones rápidas pero no localmente óptimas, 2-opt mejora en gran manera dichas soluciones.

Bonachea D. y otros en [6] utilizan un enfoque adaptativo mejorado para hallar soluciones óptimas cercanas al problema del agente viajero. Este enfoque optimiza la población de individuos de manera local aplicando Lin-Kernighan a muchas trayectorias iniciales generadas de manera aleatoria.

Lin-Kernighan (LK) (referenciado en [7]) es considerado un estándar en métodos de búsqueda local, e incluye los métodos de heurística local de 2-opt y 3-opt. Luego de aplicar (LK) el trabajo de Bonachea, continua, creando nuevas trayectorias a partir de las trayectorias "padres", esto es, mediante la adición reiterada de aristas, donde cada arista se agrega con una probabilidad proporcional a la aptitud de cada arista. La aptitud de una arista se determina por la longitud y el número de trayectorias padres que contienen la arista. Las aristas creadas se recombinan de forma aleatoria y 2-opt actúa de nuevo. El proceso antes descrito se aplica hasta cuando la población alcance un punto donde no se puede obtener ningún progreso. Otro artículo que basa su trabajo en técnicas 2-opt es el trabajo de Hiroaki Sengoku y Ikou Yoshihara en [8]. Ellos recurren a un algoritmo híbrido (AG y heurística 2-opt). Se destaca también el operador de selección que consiste en eliminar de la población actual individuos idénticos con el fin de evitar convergencias prematuras. El operador de cruce utilizado en este trabajo es un operador de cruce codicioso (GSX).

Wayne Pullan en [9] usa de igual manera un híbrido entre AG y algoritmos heurísticos, estos últimos con la intención de siempre, como es, el de reducir el espacio de búsqueda del problema. El trabajo comienza generando un "pool" de individuos muy superior al tamaño de la población del AG. El "pool" de individuos es construido aplicando operadores de cruce (3 tipos de cruce diferentes) a todos los posibles pares de individuos de la población inicial, generados de manera aleatoria, y optimizados mediante una heurística de vecinos cercanos. Luego un operador de mutación es aplicado a cada uno de estos individuos que conforman el "pool". Después de volver a aplicar una optimización local (de tipo 2-opt y de optimización codiciosa), todos los individuos son insertados a la población y finalmente la población se reduce a su tamaño original eliminando todos aquellos individuos cercanos a otros y seleccionando individuos con una longitud de trayectoria "pequeña".

IV. MODELO DE ALGORITMO GENETICO PARALELO DE GRIS –LAS

El modelo de Gris-las es un modelo de AGP que combina los modelos de Grillas e Islas descritos antes. El modelo de Gris-Las evoluciona así: Un proceso maestro (proceso cliente) se encarga de distribuir las tareas (ejecutar un AG) a varios procesos esclavos (procesos servidores). Cada proceso esclavo (asociado a un único procesador) trabaja de manera simultánea con eventuales intercambios de información. Inicialmente el modelo de Gris-Las basa su desempeño en el método de Grillas. Se utilizaron 7 topologías de Grillas en la implementación del modelo de Gris-Las en ocho posibles direcciones con respecto a un individuo particular, estas fueron:

4-n: Vecinos horizontales y verticales.
5-n: Vecinos horizontales y verticales más el centro.
8"-: Todos los vecinos de distancia 1.
9-n: Todos los vecinos de distancia 1 más el centro.
16-n: En todas las 8 direcciones, vecinos con distancia 2.
17-n: En todas las 8 direcciones, vecinos con distancia 2 más el centro.
16-n: Todos los vecinos a una distancia de 1, más los vecinos verticales y horizontales a una distancia 2.

Las Figuras 3 y 4 describen algunas de estas topologías.

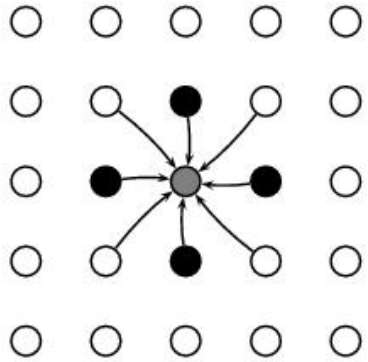

Figura 3. Topología 4 - n

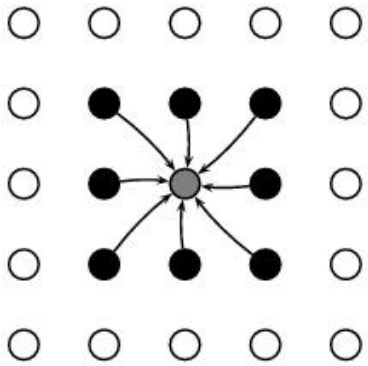

Figura 4. Topología 8 - n

La rata de cruce *pc*, ($0 < pc < 1$) es la misma en cada procesador y es manejada de manera constante durante toda la ejecución del AG. El operador de cruce se maneja en dos fragmentos diferentes en la ejecución del problema, la primera utilizada propiamente en el aporte del modelo de Grillas y la segunda en la aplicación concerniente al aporte del modelo de Islas.

De cualquier manera el operador de cruce se implementa así: La cadena que representa al primer padre es copiada desde el inicio hasta un punto de cruce seleccionado de manera aleatoria a la primera cadena hijo, luego se "escanean" los valores que componen la cadena del segundo padre y si estos no hacen parte todavía en la primera cadena hijo, son entonces adicionados a este. La segunda cadena hijo se obtiene de manera semejante, comenzando el proceso a partir de la segunda cadena padre. Por ejemplo, si se tienen los padres 321|45 y 235|14, y el punto de cruce sucede en la tercera posición, los hijos obtenidos son 32154 y 23514 respectivamente.

La rata de mutación *pm*, ($0 < pm < 1$) es implementada para ser proporcionada de manera diferente en cada procesador, pero es manejada, al igual que el cruce, de forma constante durante

toda la ejecución del AG. La rata de mutación es fundamental en la eficiencia del método de Grislas, se sugiere proporcionar una rata de mutación de *0.01* (valor usual en la ejecución de un AG Simple) en una gran mayoría de procesadores, un valor más alto en algunos otros procesadores y un valor $pm = 0.6$ en por lo menos un procesador con el fin de conseguir mayor diversidad en el material genético de la población y por lo tanto menor probabilidad de estancamiento del algoritmo (convergencia a un óptimo local).

El operador de mutación se implementa de una manera simple. Dos valores son seleccionados e intercambiados. Por ejemplo si se tiene el individuo *32154* y se seleccionan para intercambio los valores *2* y *4* el individuo resultante es *34152*.

Un valor de frecuencia sincrónica de intercambio de información entre procesadores también es manejado desde el proceso cliente hacia los procesos servidores, este valor se utiliza para determinar en que generación sucede un intercambio de soluciones parciales encontradas por cada procesador o bien para determinar en que generación ocurre un intercambio parcial de material genético (cruce) entre individuos manejados por diferentes procesadores. La figura 6 muestra como ocurre este último procedimiento de cruce; La parte superior de la figura, es una grilla de un AG manejada por un procesador i , y parte inferior es una grilla de un AG manejada por un procesador $(i+1) \pmod n$ (n: número de procesadores disponibles). Regularmente el individuo central en cada grilla sufre cruce con alguno de sus individuos vecinos, pero según el valor de frecuencia, el cruce puede ocurrir, entre el individuo central de cada grilla con alguno de los vecinos del individuo central de la otra grilla.

Los nuevos individuos así generados entran en pugna con los individuos de la población actual para determinar cuales 1Se puede recurrir a una mutación multi-punto donde dos ó más valores son seleccionados e intercambiados en la misma cadena en lugar de uno solo.

Individuos componen la población en la próxima generación; de nuevo, los individuos con mayor aptitud conformarán esa nueva población. El modelo de Grislas recurre ahora al modelo de Islas para continuar su curso, cruce y mutación usual son ahora efectuados localmente en el correspondiente AG ejecutado por cada procesador. Se evalúa la aptitud de la población resultante después de aplicar los operadores anteriores y de nuevo cada cierto número de iteraciones (frecuencia), se efectúa un intercambio de información entre procesadores. Se determinan los mejores individuos así como los peores en cada AG ejecutado por cada procesador y los mejores k individuos en el procesador i reemplazan a los peores k individuos de la población manejada por el procesador $(i+1) \pmod n$ (8i 2 Z) (n: número de procesadores disponibles). La figura 6 exhibe como se hace tal intercambio de individuos. Por último cada AG calcula la aptitud de los individuos resultantes de cada población y el algoritmo continua su ejecución en una próxima iteración.

En nuestro Algoritmo Genético Paralelo de Gris-Las el método heurístico 2-opt provee otra implementación del operador de mutación.

El algoritmo heurístico de optimización local 2-opt es ejecutado por cada Algoritmo Genético simple según la misma rata de cruce utilizada antes. Por lo tanto, ciertos individuos son mejorados de manera significativa por este método.

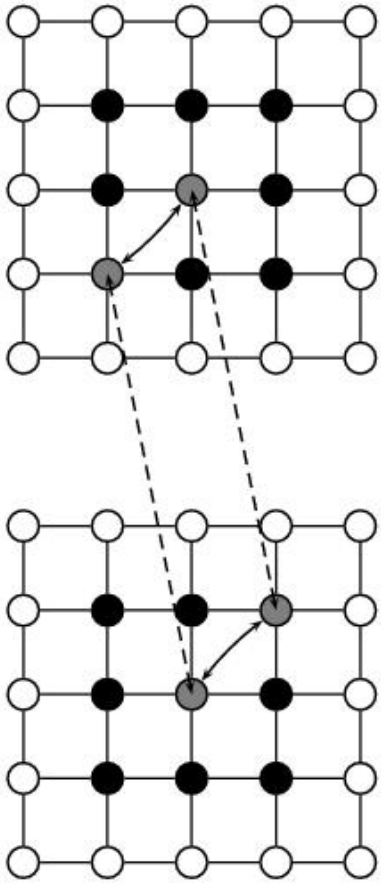

Figura 5. Cruce de individuos de una misma grilla.

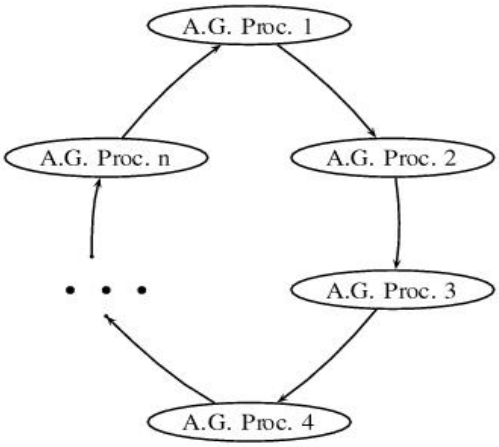

Figura 6. Intercambio entre procesadores

El modelo paralelo de Gris-Las, mejorado con optimización local 2-opt se resume en el Algoritmo 3:

Algoritmo 3. Modelo de Gris-Las

```
Producir P subpoblaciones de tamaño N cada una
generación := 1
CICLO mientras no se consiga la condición de terminación
  PARA cada subpoblación HACER en paralelo
    Evaluar y seleccionar individuos por aptitud
    PARA cada celda i en la grilla HACER
      Generar un individuo aleatorio i
    Finalizar PARA
    PARA cada celda i HACER
      Evaluar individuo i
      SI [generacion mod frecuencia = 0] entonces
        PARA cada individuo i HACER en paralelo
          Seleccionar un individuo k en la vecindad
          en la subpoblación siguiente
          Producir hijos de i y k
        Finalizar PARA en paralelo
      SI NO
        PARA cada individuo i HACER
          Seleccionar un individuo k en la vecindad
          Producir hijos de i y k
        Finalizar PARA
      Finalizar SI
      Asignar uno de los hijos a i
    Finalizar PARA
    Producir nuevos individuos
    Mutar individuos
    Mutar individuos con algoritmo heurístico 2-opt
    SI [generación mod frecuencia = 0] entonces
      Enviar K (K < N) mejores individuos a una
      subpoblación siguiente
      Recibir K individuos de la subpoblación siguiente
      Reemplazar K individuos en la subpoblación
    Finalizar SI
    Evaluar individuos por aptitud
  Finalizar PARA en paralelo
  generación := generación + 1
Final del CICLO mientras.
```

V. EXPERIMENTACION

El modelo de Gris-Las se implementó sobre un sistema distribuido de 5 procesadores bajo el esquema de pasada de mensajes utilizando la herramienta de software PVM (Parallel Virtual Machine). El código fue escrito en lenguaje C (computadores Pentium IV) bajo Linux Fedora Core 5. Cada uno de los procesadores ejecuta un AG con una población inicial de 64 individuos. Los problemas examinados son problemas contenidos en TSPLIB [16] y se compararon con algunos resultados referenciados en la bibliografía. La Tabla I compara los mejores resultados de cada uno de los problemas analizados, encontrados por Gris-Las y las contrapartes, al igual que los propios resultados presentados por TSPLIB. Gris-Las efectuó una cantidad máxima de 300 iteraciones para cada problema y la Tabla I reporta la mejor solución de 14 ejecuciones efectuadas (2 por cada una de las 7 topologías enunciadas en la sección anterior).

Gris-Las utilizó ratas de mutación *pm* para cada procesador de *0.01*, excepto en uno o dos procesadores donde se utilizó una rata $pm = 0.6$. Los procesadores que utilizaron el valor de $pm = 0.01$ trabajaron con mutación usual de 1-punto, mientras los procesadores que utilizaron $pm = 0.6$ trabajaron con una mutación 2-puntos. La rata de cruce pc fué la misma en cada procesador con un valor constante durante toda la ejecución del algoritmo de $pc = 0.6$. La mutación 2-opt ocurre también según una rata de probabilidad de *0.6* y la frecuencia de migración entre procesadores ocurre cada *3* iteraciones. Las diferentes ratas de cada uno de los operadores genéticos así como la de 2-opt se estableció de tal manera, dado, que ejecuciones previas de Gris-Las mostraron que ellas arrojaban los mejores resultados, por ejemplo, el considerar una rata de mutación alta (de *0.6* en este caso) en uno o dos procesadores, se logró salir más rápidamente de estancamientos prematuros, pues se obtiene mayor diversidad de población en este procesador que se comparte con las "buenas" soluciones encontradas por los otros procesadores.

Gris-Las alcanzó los mejores resultados cuando se trabajó con una frecuencia de migración de 3 iteraciones. Una frecuencia menor no aprovecha la propiedad de diversidad de resultados que puede alcanzar cada población independiente propuesta por el modelo de Islas, mientras una frecuencia mayor hace que buenos resultados se demoren en ser obtenidos. En la mayoría de problemas tratados Gris-Las consiguió aceptables soluciones en un número bajo de iteraciones. El problema de mayor cantidad de ciudades (Ch130) fue muy bien resuelto por Gris-Las pero la cantidad de iteraciones fue alta. La Tabla II proporciona una comparación en cantidad de iteraciones de Gris-Las y sus contrapartes.

De todas formas debemos considerar que los resultados conseguidos por Gris-Las fueron buenos y en algunos casos mejores que en algunas o en todas las contrapartes. La tabla III exhibe tal característica; se observa que en todos los casos (excepto en uno) el porcentaje de error cometido está siempre abajo del 1%, el peor caso (Eil101) muestra un error inferior al 1.9% y en un caso (KroB100) se consigue un mejor resultado que el presentado en TSPLIB y en todas las otras comparaciones bibliográficas.

No se consideraron comparaciones en tiempo, debido a que los resultados en Gris-Las y los resultados de sus contrapartes se efectuaron sobre diferentes máquinas; algunas referencias por ejemplo utilizaron máquinas con un muy alto poder computacional.

Tabla 1. Comparación de Mejores Resultados de Grislas con Algunos otros Resultados reportados en la Literatura

Nombre	Eil51	St70	KroA100	Ch130	Eil101	KroB100
Gris-Las	428.9817	677.1097	21285.4512	6110.7217	640.9332	22139.0762
TSPLIB [16]	426	675	21282	6110	629	22141
Xue [3]	428.87	677.11	21285.44	6110.72	-	-
AG Xue [3]*	429.53	690.66	21311.32	6251.81	-	-
Pullan [9]	426	675	21282	6110	629	22141
Brocki [5]	426	-	-	-	646	-
Denzinger[17]	-	-	-	-	-	22141

* Este es un AG simple referenciado también por Xue en [3]

Tabla 2. Comparación en Número de Iteraciones de Mejores Resultados de Grislas con Algunos otros Resultados Reportados en la Literatura

Nombre	Eil51	St70	KroA100	Ch130	Eil101	KroB100
Gris-Las	18	13	17	283	53	39
Xue [3]	22	10	35	32	-	-
AG Xue [3]	936	1209	1809	2959	-	-
Pullan [9]	9	4	4	7	6	4
Brocki [5]	-	-	-	-	-	-
Denzinger[17]	-	-	-	-	-	-

Tabla 3. Porcentaje de Error de la Solución Encontrada en Grislas con Respecto a la Solución TSPLIB

Nombre	Eil51	St70	KroA100	Ch130	Eil101	KroB100
Gris-Las	0.6999	0.3125	0.01621	0.0118	1.8971	-

VI. CONCLUSIONES

El artículo describe el modelo de Gris Las que es un algoritmo que soluciona el problema del Agente Viajero combinando dos de los modelos de Algoritmos Genéticos Paralelos más importantes como son el modelo de Islas y el modelo de Grillas junto con una técnica de optimización local como es la heurística 2-opt. El modelo de Gris-Las alcanza mejores resultados si se utiliza cada vez un número mayor de procesadores pues se contempla que se consigue mucha más diversidad en la población y por lo tanto se evitan estancamientos prematuros, esta mayor diversidad también se obtiene considerando valores adecuados en los operadores genéticos, como fueron indicados en la fase de experimentación. El modelo de Gris-Las es competitivo con otros métodos que aparecen en la literatura sobretodo en la calidad de la solución. Trabajos futuros podrán mejorar los resultados conseguidos por Gris-Las recurriendo a técnicas más sofisticadas de búsqueda de optimización local como heurísticas 3-opt ó Lin-Kernighan en lugar de la heurística 2-opt utilizada en este trabajo.

REFERENCIAS

[1] Wang, L., Maciejewski, A. A., Siegel, H. J. y Roychowdhury, V. P., 1998. A comparative Study of five Parallel Genetic Algorithms using the traveling Salesman Problem, Parallel Processing Symposium. IPPS/SPDP. Proceedings of the First Merged International and Symposium on Parallel and Distributed Processing, March 1998, pp.345-349.

[2] Jog, P., Suh, J. Y. y Van Gucht., 1989. The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem, 3rd Int'l Conf. Genetic Algorithms, pp. 110-115.

[3] Wang, C.-X., Cui, D.-W., Wan, Di.-S. y Wang, L., 1989.A novel genetic algorithms based on gene therapy theory. Transactions of the Institute of Measurement and Control.

[4] Jiao, L.C. y Wang L., 2000. A novel genetic algorithm based on immunity. IEEE Transactions on System, Man, and Cybernetics, Part A, pp. 552-561.

[5] Brocki, L., 2002. Kohonen Self-Organizing Map for the Traveling Salesperson Problem. Springer-Berlin.

[6] Bonachea, D., Ingerman, E., Levy, J. y McPeak, S., 2000. An Improved Adaptive Multi-Start Approach to Finding Near-Optimal Solutions to the Euclidean TSP. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)

[7] Lin, S. and Kernighan, B.,1973. An effective heuristic algorithm for the traveling salesman problem, Operations Research 21. pp. 498516.

[8] Sengoku, H. y Yoshihara, I., 1998. A Fast TSP Solver Using GA on JAVA. Third International Symposium on Artificial Life, and Robotics(AROB III'98).

[9] Pullan, W., 2003. Adapting the Genetic Algorithm to the Travelling Salesman Problem. Evolutionary Computation, 2003. CEC '03. The 2003 Congress on, Vol.2., pp. 1029-1035

[10] http://www.tsp.gatech.edu/

[11] http://en.wikipedia.org/wiki/Travelling salesman problem

[12] Gutin G. and Punnen, A. P. (Eds.)., s.a. Combinatorial Optimization. The traveling Salesman Problem and Its Variations.

[13] Goldberg, D. E., 1989. Genetic Algorithms in Search, Optimizations and Machine Learning. Addison-Wesley.

[14] Michalewicz, Z., 1994. Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, 2nd edition.

[15] Tomassini, M., s.a. A Survey of Genetic Algorithms. Volume III of Annual Reviews of Computational Physics, World Scientific.

[16] Reinelt G., 1995. TSPLIB 95 documentation, University of Heidelberg.

[17] Denzinger, J. y Scholz, S., 1997. Using Teamwork for the Distribution of Approximately Solving the Traveling Salesman Problem with Genetic Algorithms.

[18] Larrañaga, P., s.a. C.M.H. Kuijpers. Genetic Algorithms for the Travelling Salesman Problem A Review of Representations and Operator, Springer Volume 13 Number 2 199 pp. 129-170

[19] Cantú Paz, E., 1997. A Survey of Parallel Genetic Algorithms, Technical Report 97003, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana Champaign, Urbana, IL.

Universidad Nacional de Colombia Sede Medellín
Facultad de Minas

Escuela de Ingeniería de Sistemas

Misión

La misión de la Escuela de Ingeniería de Sistemas es fomentar y apoyar la generación o la apropiación de conocimiento, la innovación y el desarrollo tecnológico en el área de ingeniería de sistemas e informática sobre una base científica, tecnológica, ética y humanística.

Visión

La formación integral de profesionales desde el punto de vista científico, tecnológico y social que les permita adoptar, aplicar e innovar conocimiento en el campo de los sistemas e informática en sus diferentes aspectos, aportando con su organización, estructuración, gestión, planeación, modelamiento, desarrollo, procesamiento, validación, transferencia y comunicación; para lograr un desempeño profesional, investigativo y académico que contribuya al desarrollo social, económico, científico y tecnológico del país.

Escuela de Ingeniería de Sistemas
Dirección Postal:
Carrera 80 No. 65 - 223 Bloque M8A
Facultad de Minas. Medellín - Colombia
Tel: (574) 4255350 Fax: (574) 4255365
Email: esistema@unalmed.edu.co
http://pisis.unalmed.edu.co/