Heurística GSOM paralela para la solución aproximada del TSP

Trabajo de Grado presentado a la ilustre Universidad Central de Venezuela para optar por el Título de Magíster Scientiarium en Computación Emergente

José Luis Bastardo

Caracas, Abril 17 de 2007

Certifico que he leído este Trabajo de
Grado y que lo encuentro apropiado
tanto en su contenido como en su
formato y apariencia externa.

Dr. José Alí Moreno

Agradecimientos

Agradezco al Prof. José Alí Moreno por haberme mostrado el potencial de la Computación Emergente.

Agradezco a los compañeros del Laboratorio de Computación Emergente: Cristina García, Luigina Damato y Arturo Fernandez, por toda su colaboración.

Muchas gracias a Alexis, José y Pedro Arreaza así como también a Mirian Zanoja y a Elias Guerra por sus sabios consejos.

Gracias también a Cezar Arreaza y a Jesús y Miguel Guerra por sus interesantes preguntas.

Dedicatoria

Para Alejandra, Arcadia, Aura y Elena.

Resumen

Este trabajo toma como punto de partida una heurística, para la solución aproximada del problema del agente viajero, que se basa en el uso de un mapa autoorganizativo creciente unidimensional con topología de anillo. Esta heurística inicial se denomina GSOM.

Se diseña, se implanta y prueba una versión paralela, de la heurística GSOM, denominada PGSOM. Antes de abordar el diseño en paralelo, la heurística original es modificada para mejorar el tiempo de ejecución y conservar o mejorar la calidad de las soluciones. PGSOM se construye a partir de la heurística GSOM modificada.

Finalmente se hacen recomendaciones para continuar mejorando el desempeño de las heurísticas desarrolladas.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

El estudio de lesiones en el cerebro ha sido la primera metodología utilizada en la determinación de la distribución espacial de las funcionalidades cerebrales. El diagnóstico de una lesión en una región particular del cerebro y la observación de una deficiencia en la realización de una actividad, ofrecen evidencia de un vínculo entre localidad y funcionalidad cerebral. Los logros de esta metodología para la realización de un mapa del cerebro, han sido ampliamente superados por metodologías más modernas [12]. La Imagen de Resonancia Magnética(MRI) ofrece representaciones gráficas que parecen radiografías grisáceas de tejidos claramente diferenciados. La Imagen de Resonancia Magnética funcional (fMRI) opera sobre la anterior imagen anatómica, agregándole la demarcación de áreas con intensa actividad cerebral. fMRI es una técnica muy rápida que puede mostrar claramente, en tiempo real, la oleada de actividad de distintas partes del cerebro cuando reacciona ante diferentes estímulos o se ocupa de distintas tareas. fMRI trabaja con una resolución espacial de milímetros y una resolución temporal de milisegundos sobre la totalidad del cerebro.

Los estudios sobre algunos mapas han sugerido la presencia de características muy interesantes, tales como la representación abstracta de cualidades en la distribución espacial del tejido encargado del procesamiento. Por ejemplo, la región de Broca, asociada al procesamiento sintáctico del

lenguaje, muestra evidencia de una organización espacial de sus operaciones, que podría resultar linguísticamente significativa [5]. El estudio de lesiones tempranas en el cerebro infantil ha permitido el seguimiento del proceso de redistribución de las funciones afectadas [15]. Se observa un proceso de adaptación en el cual la tarea que debía realizar la región lesionada es asumida, en mayor o menor grado, por regiones saludables. Se observa un proceso de reacomodo seguido por un cerebro de estructura restringida por la lesión.

En un tipo particular de red neuronal artificial, usado para modelar el sistema nervioso, las neuronas forman vecindades que implican interacciones laterales mutuas y mediante una competencia por el procesamiento de las señales, se realiza un proceso de adaptación. En este tipo de red neuronal, el aprendizaje es denominado competitivo, no-supervisado o auto-organizativo [8]. Los mapas auto-organizativos son redes neuronales artificiales cuyas células se ajustan a varios patrones de señales de entrada mediante un proceso de aprendizaje no supervisado. En la versión básica, sólo una célula o grupo local de células reacciona ante la presentación de un estímulo. Las células se ordenan y arman una representación abstracta de ciertas características del conjunto de señales. Los resultados de los mapas auto-organizativos presentan un aspecto muy natural, y sugieren que el proceso adaptativo del mapa puede ser similar a aquellos que se desarrollan en el cerebro. Actualmente se tiene un conocimiento claro del proceso por el cual se pueden formar estos mapas artificiales de manera adaptativa y automática. Se ha observado que existen dos efectos esenciales que conducen a la organización espacial de los mapas:

1. La concentración espacial de la actividad de la red alrededor de la célula que mejor se ajusta al estímulo actual.

2. La actualización de la célula (y su vecindad) de mejor ajuste al estímulo actual.

Resulta crucial, para la formación de mapas ordenados, que las células que realizan el aprendizaje no sean afectadas independientemente unas de

las otras, por el contrario, se debe realizar un tipo similar de actualización sobre un subconjunto de células topológicamente relacionadas.

Teuvo Kohonen estableció al inicio de los 80 [7], los principios de funcionamiento de un mapa autoorganizativo. Desde entonces se ha dedicado un gran esfuerzo al estudio de sus propiedades y aplicaciones [9].

Los mapas autoorganizativos inducen de manera no supervisada agrupamientos de datos n-dimensionales bajo una topología restringida que habita un espacio m-dimensional donde, en general, $m < n$. La estructura de topología restringida se diseña con el fin de representar una de las posibles semejanzas m-dimensionales, existente entre los patrones originales. Por ejemplo, si se crea una red con una vecindad unidimensional y se le presentan entradas bidimensionales, se observa que la red organiza sus neuronas de forma tal que emerge una ruta, generalmente corta, que permite el recorrido de los patrones bidimensionales de entrada, tal como se ilustra en la figura 1.1.

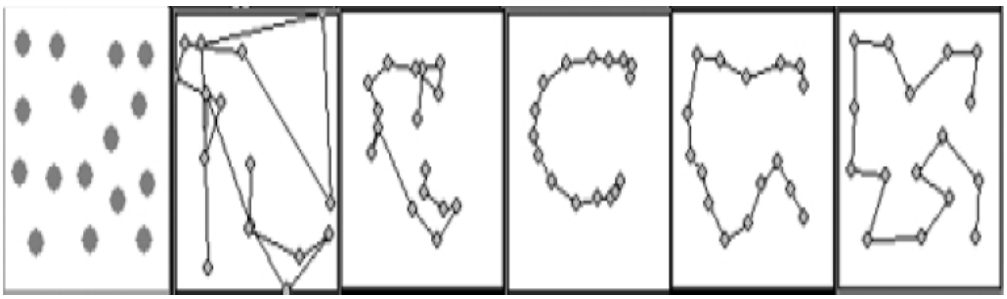

Figura 1.1: Patrones bidimensionales y Red unidimensional

Este comportamiento hace plausible el uso de mapas autoorganizativos en problemas de determinación de rutas. Particularmente se han obtenido resultados muy interesantes en el problema del agente viajero [6][16][13][4], mejor conocido como TSP por las siglas de su nombre en inglés *Traveling Salesman Problem*.

En el TSP, dado un conjunto de ciudades, se desea encontrar un camino cerrado que permita realizar un recorrido que pasa por cada una de las ciu-

dades exactamente una vez y regresa a la ciudad de partida. Tales caminos cerrados son denominados rutas TSP y tienen un costo asociado que corresponde a la suma de las distancias recorridas al pasar de una ciudad a otra. La solución del TSP es una ruta TSP de costo mínimo. Por razones prácticas y teóricas el TSP ha sido objeto de numerosos estudios, ha sido abordado de diversas formas y constituye una referencia en optimización combinatoria [1].

El TSP es un problema que tiene una gran comunidad de investigadores quienes cuentan con una librería, de instancias del problema, que sirve como referencia a la hora de comparar el exito de los distintos métodos desarrollados para su solución. Esta librería se conoce como TSPLIB[11] y cuenta con instancias de hasta 859000 ciudades. Los nombres de las instancias en la TSPLIB siempre incluyen el número de ciudades consideradas luego de un nemónico que hace referencia al origen de la instancia.

El TSP es un problema NP-Completo. Todo algoritmo, conocido, que pretende encontrar una solución exacta, presenta un tiempo de ejecución que crece más rápido que cualquier polinomio, en función del número de ciudades N. El más exitoso de estos algoritmos es del tipo *branch and cut*[3], y permite encontrar una solución óptima o muy cercana al óptimo mediante la realización de una eficiente exploración sobre el espacio de soluciones con un árbol de búsqueda que es convenientemente podado en aquellas ramas que conducen a soluciones no óptimas. En el cuadro 1.1 se presenta el tiempo de ejecución de una implantación de este algoritmo[3] al resolver diversas instancias en la TSPLIB.

El costo excesivo de la búsqueda de una ruta TSP optima, ha desviado la atención hacia la construcción de heurísticas que sean capaces de encontrar una ruta TSP sub-optima en un tiempo razonablemente corto.

Las heurísticas para el TSP basadas en mapas autoorganizativos, generalmete definen una red de M neuronas ($M \geq N$) con una topología restringida unidimensional, en forma de anillo [16]. La red realiza el proceso de autoorganización guiada por la presentación de los vectores de posición de las

Cuadro 1.1: Tiempo de ejecución de *branch and cut*

Instancia	Ciudades	Tiempo[horas]
pcb3038	3038	8.6
fl3795	3795	8.2
fnl4461	4461	3.4
rl5915	5915	46.1
rl5934	5934	48.3
pla7396	7396	106.9
usa13509	13509	72.0
brd14051	14051	72.0
d15112	15112	72.0
d18512	18512	72.0

ciudades. Al concluir el proceso, la red se ha adaptado a la distribución de las ciudades. La asignación de cada ciudad a la neurona más cercana disponible, permite la construcción de una ruta TSP. En general la ruta TSP obtenida es sometida a un proceso de mejoramiento k-optimo[10].

Los resultados obtenidos por heurísticas basadas en mapas autoorganizativos, para la solución aproximada del TSP, son muy interesantes y prometedores, sin embargo , hasta la fecha, sólo han podido abordar instancias pequeñas.

En [4] se presenta la heurística GSOM para el TSP con un mapa autoorganizativo creciente que ha permitido el tratamiento de problemas de hasta 4461 ciudades, lo que resulta una cantidad alta para esta metodología. En GSOM, el anillo inicialmente posee un número pequeño de neuronas. Además de adaptarse a los estímulos, el anillo también crece al incorporar nuevas neuronas en las áreas de mayor densidad en la distribución de ciudades. El proceso continúa hasta completar un anillo de tamaño $M = 2N$. Luego de la asignación de cada ciudad a la neurona más cercana disponible, se construye una ruta TSP que es sometida a un proceso de mejoramiento

tipo 2-optimo.

Este esquema creciente mejora el tiempo de ejecución, dado que la búsqueda de la unidad de mejor ajuste se realiza, en general, sobre un anillo de menor tamaño. Los resultados también indican una mayor calidad de las soluciones. La incorporación de nuevas neuronas en las áreas de mayor densidad, mejora la capacidad de la dinámica para encontrar rutas cortas.

El presente trabajo utiliza la heuristica GSOM para el TSP como punto de partida. Se pretende aumentar su velocidad de ejecución mientras se conserva o incrementa la calidad de las soluciones. Estas mejoras deben permitir el tratamiento de las instancias de mayor tamaño en la TSPLIB.

Las mejoras en la heurística GSOM, se realizan en dos etapas:

1. Se entona la heurística GSOM mediante ligeras modificaciones a la implementación de su dinámica. Aquí se hace especial énfasis en acortar el tiempo de búsqueda de la neurona que mejor se ajusta al estímulo presentado.

2. Se desarrolla una versión paralela de la heurística GSOM. Esta versión es denominada PGSOM.

Capítulo 2

Redes Autoorganizativas

2.1. Mapas del cerebro

Hace más de cien años, se inició el proceso de cartografía del cerebro gracias a deducciones hechas a partir de deficiencias funcionales e impedimentos inducidos por diversos tipos de lesiones, hemorragias, tumores o malformaciones. Diferentes regiones del cerebro parecían dedicadas a tareas específicas. La cartografía del cerebro se ha revelado con mayor detalle, gracias al empleo de metodologías más modernas, tales como las imágenes de resonancia magnética [12].

La Imagen de Resonancia Magnética(MRI) se basa en la alineación magnética de partículas atómicas en los tejidos del cuerpo, bombardeadas con ondas de radio. Esto hace que las partículas emitan distintas señales de radio según el tipo de tejido del que se trate. MRI ofrece imágenes que parecen radiografías grisáceas de tejidos claramente diferenciados. La Imagen de Resonancia Magnética funcional (fMRI) opera sobre la anterior imagen anatómica, agregándole la demarcación de áreas con intensa actividad cerebral. Se induce el disparo neuronal con la adición de glucosa y oxígeno, normalmente transportado por la sangre. Cuando un área del cerebro se activa, estas sustancias fluyen hacia allí y la fMRI hace visibles las áreas donde hay más oxígeno. fMRI es una técnica muy rápida que puede mostrar claramente, en tiempo

real, la oleada de actividad de distintas partes del cerebro cuando reacciona ante diferentes estímulos o se ocupa de distintas tareas. fMRI trabaja con una resolución espacial de milímetros y una resolución temporal de milisegundos sobre la totalidad del cerebro humano.

Una investigación liderada por Yosef Grodzinsky [5], sobre la región de Broca, asociada al procesamiento sintáctico del lenguaje, muestra evidencia (no concluyente) para sustentar la siguiente conjetura:

1. Las principales operaciones sintácticas están neurológicamente individualizadas.

2. La organización de estas operaciones en el espacio del cerebro es linguísticamente significativa.

Los estudios sobre otros mapas también arrojan evidencia sobre la representación abstracta de cualidades en la distribución espacial del tejido encargado del procesamiento.

En un tipo particular de red neuronal artificial, usado para modelar el sistema nervioso, las neuronas forman vecindades que implican interacciones laterales mutuas y mediante una competencia por el procesamiento de las señales, se realiza un proceso de adaptación. La red tiende a organizarse, distribuyendo sus neuronas, de forma tal que revela características subyacentes en la distribución de las señales. Este proceso adaptativo parece similar a los que ocurren en el cerebro. En este tipo de red neuronal, el aprendizaje es denominado competitivo, no-supervisado o auto-organizativo [8].

2.2. Mapas autoorganizativos

Con el nombre de Mapas Autoorganizativos se conoce a un conjunto de métodos capaces de inducir, de forma no supervisada, agrupamientos de datos n-dimensionales. Estos agrupamientos están restringidos por una topología de vecindad m-dimensional, donde en general, $m < n$. La estructura de topología restringida es diseñada con el fin de hacer emerger una de las

posibles semejanzas m-dimensionales, existente entre los patrones originales. Los siguientes elementos intervienen en la autoorganización:

- Un conjunto arbitrario de estímulos o patrones n-dimensionales.
- Un conjunto de unidades de procesamiento (UP). Cada una posee un vector de pesos n-dimensional.
- Una red, o estructura restringida de interconexión de las UPs, que representa de alguna forma la relación de semejanza m-dimensional deseada.
- Una medida de distancia definida en el espacio n-dimensional.
- Una medida de distancia definida en la red de estructura restringida m-dimensional.

Cada UP tiene un vector de pesos w, que es un punto del espacio R^n, lo que permite calcular la distancia, en R^n, entre un patrón y una UP. Además cada UP se haya inserta en la red de topología restringida y allí tiene un conjunto de UPs cercanas, aquellas con las cuales está conectada; estas UPs conforman su vecindad inmediata. La estructura de la vecindad depende de la topología de la red y del tamaño del entorno considerado.

La semejanza m-dimensional deseada queda establecida al definir un mapeo inyectivo del conjunto de estímulos en el conjunto de UPs.

El proceso autoorganizativo para encontrar el mapeo se denomina entrenamiento de la red. El entrenamiento utilizado es del tipo competitivo, donde sólo una unidad(la ganadora), la más cercana al estímulo, y posiblemente algunos elementos de su vecindad son corregidos de acuerdo al patrón presentado. Tras estructurar la red de topología restringida con M UPs y definir las funciones de distancia, se puede sintetizar el entrenamiento en los siguientes pasos:

1. Inicializar los vectores de peso w_i de las UPs de forma aleatoria $\forall i \in 1..M$.

2. Seleccionar aleatoria y uniformemente uno de los patrones de entrada.

3. Buscar la UP que se encuentra más cerca del patrón seleccionado. Esta UP es la unidad ganadora(UG).

4. Realizar una corrección del w de la UG. La corrección hace que el w de la UG se parezca más al patrón presentado.

5. Si es el caso, realizar una corrección más débil sobre los w_i de las UPs de la vecindad(en la red) de UG.

6. Repetir (2) a (5) un número de iteraciones fijo o hasta que las correcciones sean menores que un umbral pre-establecido.

En el entrenamiento, el grado de cercanía entre las UPs se mide sobre la estructura de la red, mientras que el grado de cercanía entre una UP y un patrón presentado, se mide sobre el espacio R^n. A medida que el proceso de entrenamiento avanza los vectores de peso w de las UPs son modificados de forma tal que el conjunto de los w_i $; i : 1..M$, forma una nube de puntos distribuidos sobre el espacio R^n donde habitan los patrones presentados. El factor de corrección se decrementa con el tiempo de tal forma que eventualmente la red alcanza una configuración estable. Al final del entrenamiento, cada UP, además de tener una ubicación en la estructura de la red, tiene un vector de pesos w en R^n que la ubica en el espacio donde habitan los patrones de entrada. Esta doble ubicación de las UPs revela rasgos estadísticos intrínsecos contenidos en los patrones de entrada. Este cambio por el cual las UPs y su vecindad pasan de ser cercanas sólo en la estructura de la red, a ser también cercanas en el espacio n-dimensional constituye el proceso de autoorganización.

Finalmente se construye el mapeo inyectivo.Cada patrón de entrada es asociado a la UP con vector de pesos w más cercano, en R^n, al patrón.

La principal característica de la proyección dada por un mapa autoorganizativo es la preservación de la topología [2]. En la medida de lo posible,

patrones cercanos en R^n son mapeados en UPs cercanas en la red. Esta característica hace de los mapas autoorganizativos una herramienta muy poderosa en el análisis de datos, donde frecuentemente se desea construir una representación de data de un espacio de alta dimensionalidad en un espacio de baja dimensionalidad mientras se preserva la estructura interna de la data de entrada.

2.3. Red de Rasgos de Kohonen

La Red de Rasgos de Kohonen es el método autoorganizativo más utilizado. Este consiste, típicamente, en una capa de unidades de procesamiento conectadas, conformando una red de topología restringida. Comúnmente las UPs se estructuran en una línea (1D) de M unidades. Esta línea está abierta en sus extremos (figura2.1(a)) o se cierra para formar un anillo (figura 2.1(b)). También se pueden configurar como una rejilla rectangular (2D), tal como se muestra en la figura 2.2. La topología de la red depende de la característica que se desea hacer emerger.

(a) Rejilla Lineal (b) Anillo

Figura 2.1: Red de Rasgos de Kohonen 1D

Tanto la UG como las unidades que conforman su vecindad son ajustadas de tal forma que el vector de pesos de cada una de estas unidades se hace más parecido al patrón de entrada. Las unidades de la vecindad de la UG se ajustan con menor intensidad y de acuerdo a la distancia topológica que las

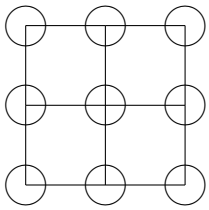

Figura 2.2: Red de Rasgos de Kohonen 2D

separa de la UG. Sea j la UG, los vectores de pesos w_i se ajustan por medio de la regla de aprendizaje de Kohonen:

$$w_i(t+1) = w_i(t) + \varepsilon(t)\Phi_{ij}(t)(x(t) - w_i(t)) \tag{2.1}$$

donde $w_i(t)$ es el vector de peso de la unidad i en el instante $t, \varepsilon(t)$ es la tasa de aprendizaje, $x(t)$ es el vector de entrada, o patrón, presentado en t y $\Phi_{ij}(t)$ es una función de vencidad entre las unidades i y j, dada por:

$$\Phi_{ij}(t) = exp(\frac{-\tilde{d}(i,j)^2}{\sigma^2(t)}) \tag{2.2}$$

donde $\sigma(t)$ es el parámetro de ancho y la función $\tilde{d}$ mide la distancia de las unidades i y j en la red. En una rejilla cuadrada, sea (f_i, c_i) la posición, fila y columna respectivamente en la cual se encuentra la unidad i y (f_j, c_j) la posición de la unidad j. La distancia $\tilde{d}(i,j) = ((f_i - f_j)^2 + (c_i - c_j)^2)^{\frac{1}{2}}$ es la distancia euclídea de las posiciones de las unidades en la red.

La función de distancia en el espacio n-dimensional, aquella que mide la cercanía entre el patrón k y la UP i, es la distancia euclídea:

$$d(x_k, w_i) = (\sum_{l=1}^{n}(x_k^{(l)} - w_i^{(l)})^2)^{\frac{1}{2}} \tag{2.3}$$

La tasa de aprendizaje $\varepsilon(t)$ y el parámetro de ancho $\sigma^2(t)$ se suponen funciones lineales decrecientes en el tiempo. Su forma funcional es la siguiente:

$$\varepsilon(t) = m_\varepsilon(t - t_0) + \varepsilon_0 \tag{2.4}$$

$$m_\varepsilon = \frac{\varepsilon_f - \varepsilon_0}{t_f - t_0} \tag{2.5}$$

$$\sigma(t) = m_0(t - t_0) + \sigma_0 \tag{2.6}$$

$$m_0 = \frac{\sigma_f - \sigma_0}{t_f - t_0} \tag{2.7}$$

Los sub-índices 0 y f indican los valores iniciales y finales, respectivamente.

Debido a que la tasa de aprendizaje y el parámetro de ancho son funciones decrecientes en el tiempo, los ajustes realizados sobre los pesos de las unidades resultan cada vez más pequeños a medida que avanza el proceso de entrenamiento. Esto conduce a la estabilización de la red.

La figura 2.3 ilustra el uso de mapas de vecindad bidimensional para representar funciones de densidad uniforme tridimensional [8].

2.4. Red con Crecimiento

La Red con Crecimiento es una versión no-estática de la Red de Rasgos de Kohonen. En este modelo el número de unidades de procesamiento, de la red de topología restringida, aumenta con el tiempo. En otras palabras, el número de unidades M es ahora una función creciente del tiempo $M(t)$. Inicialmente la red está formada por un número pequeño de unidades de procesamiento M_0 agrupadas según la topología restringida seleccionada. Al finalizar el entrenamiento, la red estará conformada por un total de $M_f (> M_0)$ unidades de procesamiento.

El proceso de entranamiento sigue los mismos pasos de adaptación listados para la regla de aprendizaje de Kohonen, pero debe incluir la inserción de las nuevas unidades. Esto implica la determinación de cuando y donde

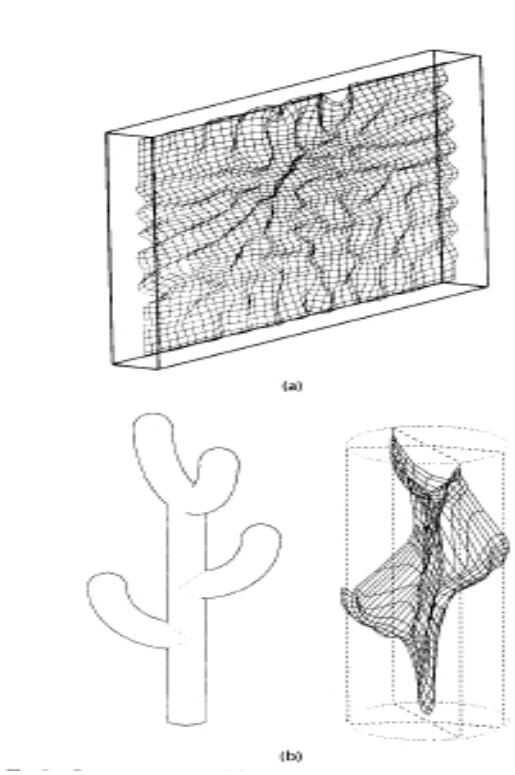

Figura 2.3: Representación de funciones de densidad 3D mediante mapas de vecindad 2D.

insertar la nueva unidad. Una inserción se realizará luego de un número dado de presentaciones de patrones *mpre*. La ubicación de la nueva unidad se determina mediante el registro del número de veces que cada unidad resulta ganadora durante las presentaciones. La nueva unidad se conectará a la unidad más ganadora en las últimas *mpre* presentaciones.

Si una UP gana muchas veces en una secuencia de presentaciones de patrones, entonces se está haciendo cargo de la representación de una gran porción de los patrones de entrada o el conjunto de patrones resulta más denso en esa región. Hacer que la red crezca justamente en la vecindad de la UP más ganadora, contribuye a mejorar la representación de la data de

entrada mediante la red.

Con el crecimiento de la red ya no resulta necesario disminuir el parámetro de ancho de la función de vecindad Φ_{ij} en la ecuación 2.2, dado que, a medida que la red crece, la fracción de unidades que son ajustadas, junto con la UG, disminuye. Se mantiene de esta forma la garantía de obtener, al pasar el tiempo, una configuración estable.

Capítulo 3

Heurística GSOM para el TSP

El TSP es el más conocido y estudiado de los problemas NP-completos. Este problema es usado frecuentemente como una referencia en la comparación de algoritmos de optimización combinatoria. El TSP puede ser planteado en los siguientes términos: dado un conjunto de N ciudades, para las cuales se conoce la distancia que separa cualquier par de ellas, se desea encontrar un camino cerrado de costo mínimo. Es decir, un circuito que recorre el conjunto de N ciudades, pasa por cada una de ellas exactamente una vez, retorna a la ciudad de partida y la suma de las distancias recorridas es menor o igual a la de cualquier otro circuito.

La búsqueda de soluciones exactas para el TSP está asociada a un consumo de tiempo que presenta un fuerte crecimiento en función de N. Por esta razón ante instancias de gran tamaño, como las que se presentan en situaciones prácticas, se prefiere el uso de heurísticas capaces de ofrecer una solución aproximada en un tiempo razonable. Muchas de estas heurísticas tienen inspiración en procesos naturales y presentan un buen rendimiento para instancias de tamaño moderado. Aún cuando se ha argumentado que, entre las heurísticas inspiradas en procesos naturales, las redes neuronales resultan el método de peor rendimiento [14], recientes investigaciones han presentado a las redes neuronales como una alternativa competitiva y prometedora para la construcción de heurísticas eficientes para el TSP. Particular interés se ha

dedicado a la solución aproximada del TSP usando redes autoorganizativas tipo Kohonen, conocidas como SOM por las siglas del nombre en inglés *self organizing map*.

Una red SOM aborda la búsqueda de un circuito de costo mínimo sobre un conjunto de N ciudades mediante la inspección, no supervisada, de los valores de las coordenadas de las ciudades y el ajuste de los vectores de peso w de las UPs para parecerse a la data de entrada. La dinámica autoorganizativa provoca un movimiento gradual de de los vectores de peso w de las UPs hacia las ciudades. Una vez completado el entrenamiento y realizado el mapeo inyectivo de ciudades en UPs, cada ciudad tiene una ubicación en la red. Debido a la propiedad de preservación de la topología [2], la distancia entre ciudades vecinas, en la red, será tan corto como sea posible.

Una heurística exitosa que usa una red SOM creciente es presentada en [4], donde se abordan problemas de hasta 4461 ciudades. Se usa una topología restringida de anillo. Inicialmente el anillo esta formado por unas pocas UPs; con el avance del tiempo las UPs se adaptan como respuesta a la presentación de los estímulos de las ciudades, además la red crece mediante la incorporación de nuevas UPs. Al finalizar el proceso de entrenamiento se tiene una red a partir de la cual se construye una ruta TSP factible, mediante la construcción de un mapeo inyectivo de ciudades en UPs. La ruta TSP obtenida es mejorada mediante un proceso de transformaciones, tipo 2-cambios, hasta obtener una ruta TSP 2-Optima. Esta heurística es denominada *growing SOM* o GSOM.

La heurística GSOM se realiza mediante la ejecución de los siguientes pasos:

1. El vector de posición $r(k)$ de una ciudad seleccionada aleatoriamente es presentado a la red.

2. Se selecciona la neurona ganadora: la UP más cercana a $r(k)$.

3. El vector de pesos de la neurona ganadora y los de sus dos vecinos

directos se actualizan usando la siguiente regla:

$$w_j = w_j + \eta(r(k) - w_j) \tag{3.1}$$

donde η es la tasa de aprendizaje, que permanece constante durante todo el proceso. La tasa de aprendizaje de las unidades vecinas es menor que la de la neurona ganadora.

4. El contador win_j, encargado de contar el número de veces en que gana la neurona ganadora actual, es incrementado.

La fase de actualización se completa al realizar un número fijo de presentaciones. Este número es el parámetro $mpre$. Luego de cada fase de actualización viene la fase de inserción; la cual se realiza mediante los siguientes pasos:

1. Seleccionar una UP k, de forma tal que esta sea la unidad más ganadora; $win_k \geq win_i$ para toda unidad de procesamiento i.

2. Entre las dos unidades vecinas directas de la unidad k, seleccionar aquella más ganadora. Sea esta la unidad $k+1$ con win_{k+1}.

3. Entre las unidades k y $k+1$ se inserta una nueva unidad, designada new, con vector de pesos dado por

$$w_{new} = p_1 w_k + p_2 w_{k+1} \tag{3.2}$$

con $p_1 = win_k/(win_k + win_{k+1})$ y $p_2 = win_{k+1}/(win_k + win_{k+1})$.

4. Restaurar el valor de los contadores a cero; $win_i = 0$ para toda UP i.

Las fases de actualización e inserción se alternan repetidamente hasta que el número de UPs en el anillo, duplique al número de ciudades. La organización final del anillo de UPs, es tal que el vector de pesos de cada UP la ubica en la cercanía del vector de posición de una ciudad. De esta forma cada ciudad puede ser asignada a la UP más cercana y disponible. Luego de la

asignación de todas las ciudades, la ruta TSP está representada por el orden de las UPs, asociadas a ciudades, en el anillo.

Una vez establecida la ruta TSP, se pasa a una fase de mejoramiento. Aquí la ruta TSP es somentida a una secuencia de transformaciones para convertirla en una ruta TSP 2-optima. Esta fase de mejoramiento se realiza mediante la siguiente heurística:

1. Dividir la ruta TSP en diez secciones. Cada sección contiene una décima parte de la ruta TSP. Realizar las siguientes acciones para cada ciudad i en cada sección.

2. Evaluar la distancia $d(i, fc)$ entre la ciudad i y la siguiente ciudad fc en la sección.

3. Para toda ciudad k en la sección de la ruta TSP, comparar las distancias $d(i, fc)$ y $d(i, k)$.

4. Si $d(i, fc) > d(i, k)$ entonces se propone una inversión $2-optimal$ entre las ciudades fc y k.

5. La inversión propuesta es aceptada si la longitud de la sección de la ruta TSP modificada es menor que la de la sección original. En tal caso la ciudad k pasa a ser la ciudad siguiente fc.

El GSOM usa los siguientes valores de parámetros:

- $mpre$: 2000
- Tasa de aprendizaje de la neurona ganadora η :0.02.
- Tasa de aprendizaje de las UPs vecinas de la neurona ganadora: 0.6η
- El máximo número de unidades de procesamiento:$2N$; con N el número de ciudades.

Con el GSOM se han obtenido buenos resultados experimentales. La calidad de los resultados y la rapidez en la ejecución son comparables a la de otros métodos exitosos de inspiración natural[4]. El cuadro 3.1 presenta el mejor resultado obtenido en diez corridas para diversas instancias del TSP, contenidas en la TSPLIB [11].

Cuadro 3.1: Mejores resultados del GSOM

Instancia	Ciudades	Optimo	GSOM	Error Porcentual
ATT48	48	10628	10648	0.188
EIL51	51	426	427	0.235
EIL101	101	629	642	2.066
KROA150	150	26524	27229	2.657
KROA200	200	29368	30249	2.999
LK318	318	42029	43350	3.143
PCB442	442	50778	54351	7.03
ATT532	532	27686	28701	3.66
PR1002	1002	259045	269371	3.986
NRW1379	1379	56638	60431	6.96
PR2392	2392	378032	404055	6.884
PCB3038	3038	137694	149759	8.762
FNL4461	4461	182566	195582	7.13

En el cuadro 3.2 se presentan los tiempos promedio de ejecución para algunas de las instancias anteriores.

La curva que describe el crecimiento del tiempo de ejecución respecto al tamaño del problema tiene una estructura aproximadamente lineal tal como se aprecia en la representación gráfica de la data del cuadro 3.2, que se hace en la figura 3.1.

Cuadro 3.2: Tiempos de ejecución promedio para algunas instancias

Nro. Ciudades	Tiempo[seg]
101	4
200	12
532	88
1002	282
2392	1860
3038	3046
4461	5365

3.1. Modificaciones al GSOM

El objetivo de este trabajo es aumentar la velocidad de ejecución del GSOM mediante la paralelización de la heurística, sin embargo, la dinámica es suceptible a ser mejorada antes de recurrir a la ejecución en paralelo.

En GSOM, el mayor esfuerzo computacional es dedicado a la fase de actualización, aquí podemos hacer varias observaciones que revelan aspectos a mejorar.

El número de presentaciones a realizar antes de una inserción, está dado por el valor de *mpre* que se ha fijado en 2000. La realización de un número menor de presentaciones tendría un impacto positivo en la velocidad de la heurística. Para realizar el mismo trabajo, ejecutando menos presentaciones, resulta natural considerar la necesidad de aumentar la tasa de aprendizaje η, que se ha fijado, originalmente, en 0.02. Nos enfrentamos de esta forma a un proceso de entonación de la dinámica original con la orientación de usar un número de presentaciones bajo y una tasa de aprendizaje alta.

El *mpre* en la heurística original se mantiene constante y no depende del tamaño del problema. Resulta razonable suponer que los problemas pequeños requieren menos presentaciones que los problemas grandes. De esta suposición surge la idea de que el valor del *mpre* sea una fracción del número

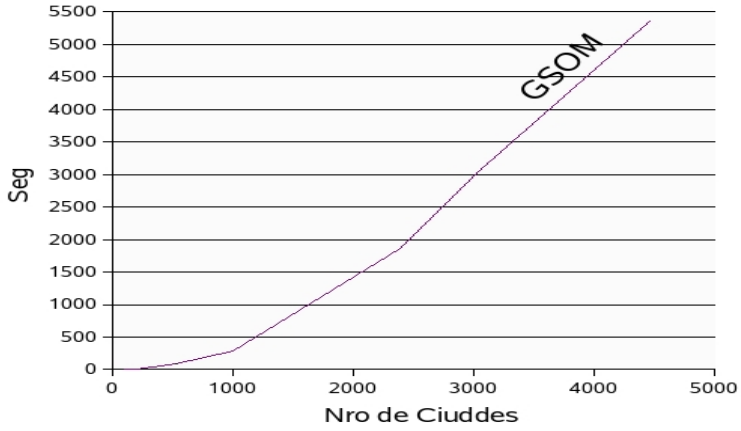

Figura 3.1: Promedio de tiempo de ejecución de GSOM

de ciuadades. Además debe ser menor que 2000 cuando se consideran problemas de hasta 4461 ciudades (máximo tamaño reportado para el GSOM). En la fase de actualización, a cada presentación de un estímulo le sigue la búsqueda de la UP ganadora, la realización de un recorrido exhaustivo sobre el anillo resulta costoso y posiblemente innecesario. Si la UP k resultó ganadora en la última presentación del vector de posición de la ciudad i, entonces en la siguiente ocurrencia de una presentación de la ciudad i, es altamente probable que resulte ganadora nuevamente la UP k. De no ser así la nueva UP ganadora, con gran probabilidad, debe ser una cercana a la UP k en la estructura de anillo.

Al incorporar las consideraciones anteriores en el diseño de una nueva heurística se logra la eliminación de las búsquedas exhaustivas, una vez que el anillo ha alcanzado un cierto tamaño. En esta nueva situación cada ciudad i debe recordar a una UP k como la UP ganadora en su última presentación. Ante una nueva presentación de la ciudad i se debe seleccionar alguna de las siguientes estrategias de búsqueda local:

- No se realiza búsqueda alguna y se asume, nuevamente, como UP ganadora a la UP k.

- Se realiza una búsqueda sobre una sección del anillo centrada en la UP k. La UP k', ganadora en esta búsqueda local, sustituye a la UP k en el recuerdo de la ciudad i.

Para la implantación de estas estrategias es necesario la incorporación de los siguientes parámetros:

- *Rbusqueda*. Indica cada cuantas presentaciones debe realizarse una búsqueda local. Cuando la presentación de la ciudad i coincide con la completación de *Rbusqueda* presentaciones, se realiza una búsqueda local centrada en la UP k recordada por i. La UP resultante de la búsqueda además de ser usada como UP ganadora, también se toma como nuevo recuerdo de i. Cuando no se realiza una búsqueda local, ante la prentación de la ciudad i, se toma la UP k asociada como ganadora.

- *RadioSeccion*. Indica el número de UPs, a ambos lados de la UP k, que deben ser considerados en la búsqueda local. El número total de UPs consideradas es $2RadioSeccion + 1$.

Al iniciar la ejecución de la heurística, todas las búsquedas son exhaustivas. En esta etápa temprana, cada ciudad i registra, por primera vez, a su UP k más cercana . La aplicación de búsquedas locales comienza una vez que el anillo tiene más de $2RadioSeccion + 1$ UPs y cada ciudad mantiene el recuerdo de una UP más cercana.

Luego de un proceso de entonación, los valores de los parámetros de la heurística GSOM modificada fueron fijados de la siguiente manera:

- $$mpre = \begin{cases} \frac{N}{4} & \text{si } \frac{N}{4} < 3000 \\ 3000 & \text{si } \frac{N}{4} \geq 3000 \end{cases}$$

- Tasa de aprendizaje de la neurona ganadora η :0.1.

- Tasa de aprendizaje de las UPs vecinas de la neurona ganadora: 0.9η

- El máximo número de unidades de procesamiento:$2N$.
- $$Rbusqueda = \begin{cases} 2 & \text{si } N < 3000 \\ \frac{N}{1000} & \text{si } N \geq 3000 \end{cases}$$
- $RadioSeccion = 1000$

Además de aumentar la velocidad, se desea conservar o mejorar la calidad de las soluciones aportadas por GSOM. Siguiendo esta dirección se ha incorporado al proceso de mejoramiento de la ruta TSP la realización de transformaciones $2\frac{1}{2}$-cambios luego de obtener una ruta TSP inicial 2-optima. Las transformaciones $2\frac{1}{2}$-cambios son similares a las transformaciones 2-cambios y se realizan para lograr mejoras que no son alcanzables con 2-cambios. Los detalles de la ejecución de un $2\frac{1}{2}$-cambio se presentan en el capítulo 4. La ruta TSP resultante no es necesariamente $2\frac{1}{2}$-optima. El mejoramiento $2\frac{1}{2}$-optimo se detiene al completar un ciclo de verificación y transformación en el que se han realizado menos de 10 cambios. Se ha utilizado este criterio de parada dado que el mejoramiento $2\frac{1}{2}$-optimo es bastante pesado y en sus últimas etapas no ofrece mayores mejoras.

Se ha modificado también el esquema de asignación de ciudades a UPs, durante la construción de la ruta TSP inicial. En GSOM simplemente se asigna una ciudad a la UP más cercana disponible. En la heurística GSOM modificada, una ciudad intenta asociarse a su UP más cercana, si no está disponible entonces se inserta una nueva UP entre la UP más cercana y su vecina más cercana a la ciudad. La nueva UP es asignada a la ciudad.

3.2. Resultados del nuevo algoritmo GSOM

Con el proposito de verificar el desempeño del algoritmo GSOM modificado, se ha realizado, con esta nueva heurística, el mismo experimento que permitió la construcción del cuadro 3.1, para el GSOM original. Para 13 ins-

tancias del TSP, de la TSPLIB, se realizaron 10 corridas y de estas se reporta la mejor solución.

El cuadro 3.3 compara los menores errores porcentuales alcanzados por la heurística GSOM y la heurística GSOM modificada. La heurística GSOM modificada es denotada por GSOM*.

Cuadro 3.3: Errores porcentuales de GSOM y GSOM*

Instancia	Error GSOM	Error GSOM*
ATT48	0.188	0.188
EIL51	0.235	0.47
EIL101	2.066	1.59
KROA150	2.657	2.26
KROA200	2.999	1.22
LK318	3.143	3.21
PCB442	7.03	5.12
ATT532	3.66	2.98
PR1002	3.986	4.59
NRW1379	6.96	4.83
PR2392	6.884	6.24
PCB3038	8.762	7.83
FNL4461	7.13	5.38

De las 13 instancias consideradas, GSOM* obtuvo un resultado menor o igual en 10 ocasiones. Se ha logrado mantener la calidad de los resultados.

El mayor cambio en el desempeño del GSOM modificado se aprecia en el tiempo de ejecución. El cuadro 3.4 compara los tiempos de GSOM y GSOM* para algunas de las 13 instancias anteriores.

Las figuras 3.2 y 3.3 muestran una representación gráfica de la data contenida en los cuadros 3.3 y 3.4, respectivamente.

El aumento de la velocidad resulta claro. Con GSOM* se ha logrado una reducción en tiempo de ejecución superior al 89 por ciento. Es de esperar

Cuadro 3.4: Tiempos de ejecución promedio de GSOM y GSOM* para algunas instancias

Nro. Ciudades	Tiempo GSOM[seg]	Tiempo GSOM*[seg]
101	4	0.2
200	12	1.2
532	88	7.1
1002	282	20.4
2392	1860	204.1
3038	3046	247.8
4461	5365	414.6

que este incremento en la velocidad, aunado a la paralelización, permita el tratamiento de las instancias de mayor tamaño en la TSPLIB.

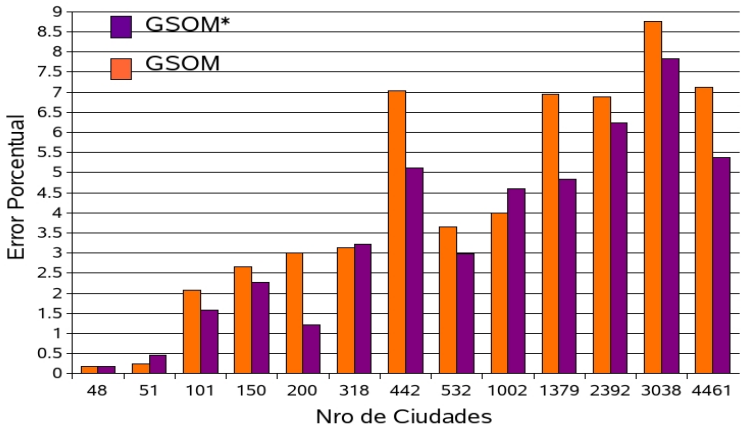

Figura 3.2: Menor error porcentual en 10 corridas de GSOM y GSOM*

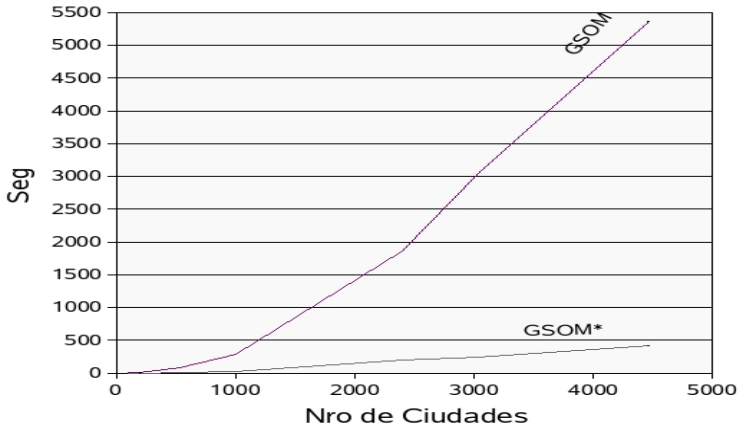

Figura 3.3: Promedio de tiempo de ejecución de GSOM y GSOM*

Capítulo 4

GSOM paralelo

Un anillo de UPs, que sigue la heurística GSOM, es sometido a una sucesión de transformaciones de actualización e inserción. En ambas transformaciones el cambio queda confinado a una región muy pequeña del anillo. La velocidad de la heurística GSOM se incrementaría si se permitiera que el anillo fuera modificado simultáneamente en varias regiones, tanto para realizar actualizaciones como para realizar inserciones. Una condición necesaria para el éxito de este esquema de modificación simultánea, es que las regiones por ser modificadas se encuentren suficientemente separadas, de tal forma que una actualización (o inserción) no interfiera con otra. En la figura 4.1 se ilustra el proceso de actualización simultánea e independiente de 2 regiones. Las regiones actualizadas aparecen sombreadas; dentro de estas, se encuentran las 3 UPs que sufren un cambio en su vector de pesos, tales UPs se dibujan más oscuras que aquellas que no son modificadas.

Tales actualizaciones deben estar dirigidas por la presentación de estímulos diferentes. Es decir el anillo debe recibir y procesar varios estímulos a la vez, tal como ocurre en las redes neuronales biológicas.

A cada fase de actualización le sigue una fase de inserción. Según la heurística GSOM, la incorporación de una nueva UP debe realizarse justo al lado de la UP más ganadora en la fase de actualización. Si al completar la fase de actualización 2 o más UPs resultan las más ganadoras, la heurística

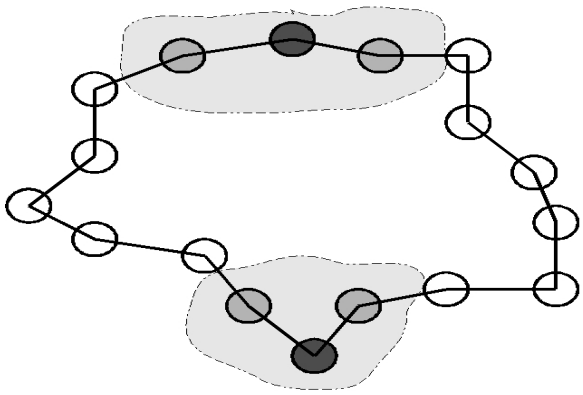

Figura 4.1: Actualización simultanea del anillo de UPs

original GSOM, simplemente toma sólo una de las UPs más ganadoras y realiza la inserción de sólo una nueva UP. Una mejora en la velocidad del GSOM sería considerar a todas las UPs más ganadoras e insertar una nueva UP por cada una de ellas. Para un esquema paralelo de GSOM resulta natural realizar simultaneamente una inserción por cada UP más ganadora.

A continuación se desarrolla un esquema de procesamiento paralelo de la heurística GSOM. Esta heurística paralela es denominada PGSOM y soporta actualizaciones e inserciones simultáneas.

4.1. División del anillo de UPs

Anteriormente se ha comprobado que una búsqueda local resulta suficiente para determinar la UP ganadora ante la presentación del vector de posición de una ciudad, durante la fase de actualización. Es decir, no es necesario manejar la totalidad del anillo para procesar un estímulo. Este simple hecho, es un indicio de la viabilidad de un esquema de procesamiento en el cual el anillo es dividido en secciones que se procesan en paralelo.

En un algoritmo paralelo para el GSOM resulta muy conveniente la división del anillo de UPs en secciones que son asignadas a diferentes procesadores

para su procesamiento "independiente". La figura 4.2 ilustra una etapa temprana del desarrollo de un anillo de UPs dividido en 4 secciones procesadas en paralelo.

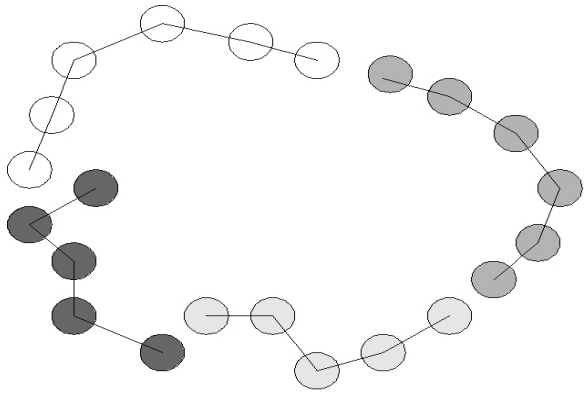

Figura 4.2: Anillo de UPs divido en 4 secciones

En este esquema debe seguir existiendo una única secuencia de estímulos. En general, ante la presentación de una ciudad solo debe reaccionar una de las secciones, en otras palabras, uno de los procesadores. Esto implica una partición del conjunto de ciudades en s clases. Aquí s es el número de procesadores dedicados al tratamiento de secciones del anillo. Una sección por procesador.

Esta partición se construye siguiendo el esquema de aprendizaje no supervisado que caracteriza a los mapas autoorganizativos.

La heurística GSOM paralela, se inicia en un solo procesador. Aquí el anillo de UPs se desarrolla normalmente hasta alcanzar un cierto tamaño. En este momento el anillo se divide en s secciones de igual tamaño y cada sección es asignada a un procesador. Luego de esto, la heurística PGSOM entra en una etapa cuyo objetivo pricipal es establecer la partición del conjunto de ciudades.

Durante la etapa de establecimiento de la partición, se siguen alternando las fases de actualización e inserción. Con la presentación de una ciudad,

cada procesador realiza una búsqueda exhaustiva sobre su sección a fin de determinar su neurona ganadora. Esto produce s neuronas ganadoras locales. La neurona ganadora global surge luego de la comparación de las s neuronas ganadoras locales.

Si un procesador posee la neurona ganadora global k, entonces debe asociar la ciudad i presentada a esta neurona. De esta forma el procesador está, temporalmente, incorporando la ciudad i a su clase y estableciendo a la UP k como recuerdo de la ciudad i. Estas asociaciones no son definitivas, durante esta etapa, una ciudad puede pasar de la clase de un procesador a otra. Esta etapa finaliza una vez que el anillo ha alcanzado cierto tamaño y cada ciudad i ha sido asociada temporalmente a una UP k y definitivamente a un procesador.

En la figura 4.3 se representan las ciudades como pequeños cuadros sombreados. Cada ciudad tiene el sombreado correspondiente a su sección a asociada por encontrarse en ella la UP más cercana.

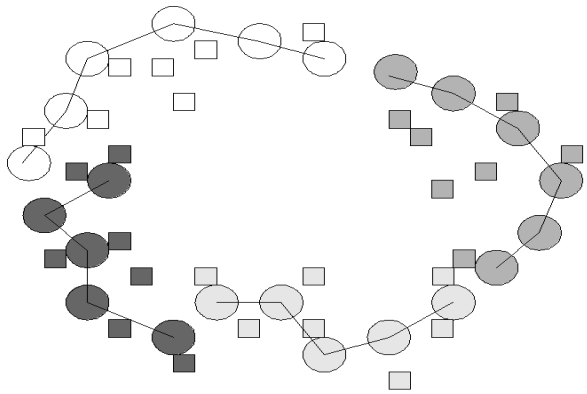

Figura 4.3: Partición de las ciudades

Una vez establecida la partición, cada procesador responderá a la presentación de las ciudades que le fueron asignadas. Cuando una ciudad es presentada a los procesadores, sólo uno de ellos debe reaccionar y procesar el estímulo. Cada uno de los restantes procesadores puede seguir avanzan-

do sobre la secuencia de estímulos hasta encontrar uno que le corresponda. La figura 4.4 muestra un arreglo que contiene estímulos repartidos entre 4 procesadores. Cada procesador es representado como un ovalo sombreado y está ubicado sobre la primera entrada que debe procesar, según su sombreado.

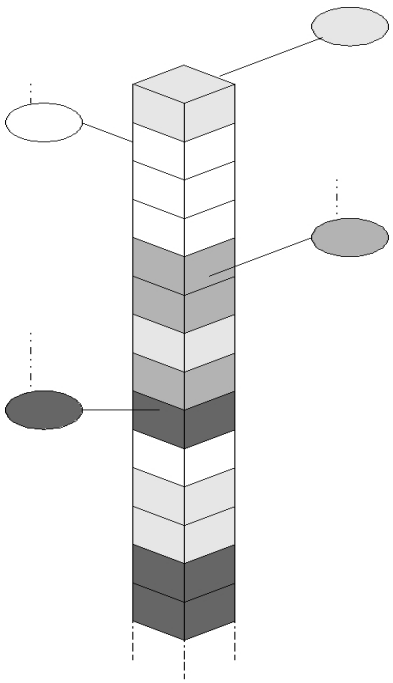

Figura 4.4: Recorrido del arreglo de estímulos

El procesamiento de cada sección del anillo no es completamente independiente.En la fase de actualización, si en una sección resulta ganadora global (ante la presentación de la ciudad i) una UP ubicada en un extremo, la actualización involucra 2 UPs de la sección ganadora global y una UP extrema de una sección vecina. Este hecho se representa en la figura 4.5, donde la flecha señala la UP ganadora global. Las 3 neuronas por ser actualizadas están agrupadas.

Para el procesamiento de un estímulo,es necesario encontrar la UP con

vector de pesos más cercano al vector de posición de la ciudad presentada. Aquí sólo se realizarán búsquedas exhautivas sobre la sección, cuando el índice del estímulo sea múltiplo del valor del parámetro *Rbusqueda*. Cuando no se reliza una búsqueda sobre la sección, se toma como ganadora la UP k, actualmente asociada a la ciudad i presentada.

La fase de actualización concluye una vez que cada procesador ha recorrido el arreglo que contiene los *mpre* estímulos y ha realizado las actualizaciones que le corresponden. En la fase de inserción cada procesador presenta el contador de su UP más ganadora. De la comparación de los contadores de estas UPs, surge(n) la(s) UP(s) más ganadora(s) global(es). El anillo incorpora una nueva UP por cada una de las UPs más ganadoras globales.

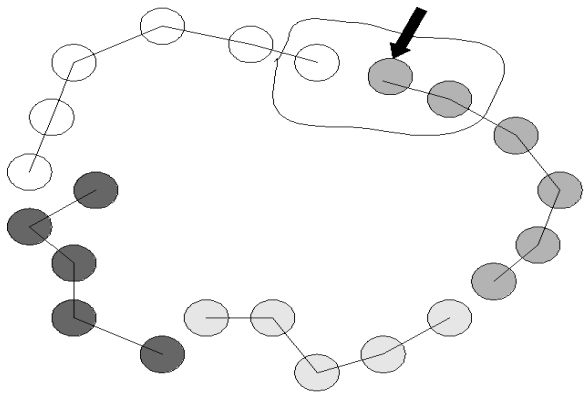

Figura 4.5: Actualizaciones fronterizas

4.2. Descripción de la heurística PGSOM

Las ideas anteriores son incorporadas en el diseño del pseudocódigo para la heurística PGSOM que se implementa sobre 8 procesadores. En un primer nivel, la heurística PGSOM está dividida en las siguientes etapas:

1. Etapa inicial

2. Etapa de particionamiento

3. Etapa de secciones independientes

4. Etapa de mejoramiento $2 - Optimo$

5. Etapa de mejoramiento $2\frac{1}{2} - Optimo$

1. Etapa inicial:Se ejecuta la heurística GSOM en un solo procesador hasta que el anillo alcanza un tamaño de 1000 UPs. Esta etapa finaliza con la asignación de una seccción de $\frac{1000}{8}$ UPs a cada procesador. A partir de aquí cada procesador se encarga de la actualización y crecimiento de su sección asignada.

2. Etapa de particionamiento:Esta etapa tiene como objetivo pricipal particionar el conjuto de ciudades, estableciendo una clase para cada procesador. Durante esta etapa se siguen alternando las fases de actualización e inserción. Esta etapa finaliza con la incorporación de 250 nuevas UPs al anillo global. Al finalizar esta etapa, dada una ciudad i, esta pertenecerá a C_p, la clase asociada al procesador p si y sólo si, la UP k más ceracana a i está en la sección administrada por el procesador p. Ante la presentación de una ciudad i, todos los procesadores realizan una búsqueda exhaustiva sobre las UPs de su sección con el fin de determinar la más cercana. Luego los procesadores comparan las UPs ganadoras para seleccionar una única UP k ganadora absoluta. El procesador p que posee la sección a la que pertenece la UP k, incorpora la ciudad i a su clase C_p, mediante la asociación de la ciudad i a la UP k. Los procesadores no ganadores se aseguran de que la ciudad i no pertenezca a su clase. La fase de inserción se realiza, como se indicó anteriormente, mediante la comparación de los contadores de las UPs más ganadoras en cada sección, con el fin de determinar la(s) UP(s) más ganadora(s) global(es), se realiza la inserción de una nueva UP por cada una UP más ganadora global.

3. Etapa de secciones independientes:En esta etapa y dentro de la fase de actualización, cada procesador recorre una copia de un único arreglo

de *mpre* estímulos y procesa sólo aquellas entradas que contienen ciudades pertenecientes a su clase. En todo momento, una ciudad i tiene asociada una UP k como unidad más cercana. Cuando se acepta una entrada ,de índice j,que contiene a la ciudad i,se toma la UP k como ganadora si y sólo si j no es múltiplo de *rbusqueda*. Cuando j es múltiplo de *rbusqueda* se realiza una búsqueda sobre la sección para determinar la UP k' más cercana. Luego de la búsqueda, la UP k' es asociada a la ciudad i. La fase de inserción se realiza igual que en la etapa anterior. Cuando el anillo de UPs alcanza un tamaño mayor o igual a $2N$,se detienen las fases de actualización e inserción y se procede a construir una ruta TSP mediante la asignación de cada ciudad a una UP tal como se hace en GSOM*. Cada procesador se encarga de la asignación de las ciudades en su partición.

4. Etapa de mejoramiento 2-Optimo:En esta etapa, cada procesador realiza una secuencia de transformaciones, del tipo dos cambios, para convertir su sección de la ruta TSP, en una sección 2-Optima.

5. Etapa de mejoramiento $2\frac{1}{2}$-Optimo:En esta etapa, todos los procesdores contribuyen para obtener una ruta TSP mejor mediante la realización de transformaciones del tipo $2\frac{1}{2} - Cambios$. Se usa el mismo criterio de parada de GSOM*, por lo que la ruta TSP resultante no es necesariamente $2\frac{1}{2} - Optima$. El esquema de procesamiento paralelo de esta etapa es descrito en la siguiente sección.

4.3. Procesamiento paralelo del mejoramiento $2\frac{1}{2} - Optimo$

En una tranformación $2\frac{1}{2}$-cambios, se reemplazan 3 arcos de la ruta TSP original por 3 arcos nuevos, sin embargo 2 de los arcos que entran y 2 de los arcos que salen deben estar conectados. Este cambio es esquematizado en la figura 4.6.

Esta tranformación puede involucrar ciudades que se encuentran muy distantes en la ruta TSP. Una revisión exhaustiva de los cambios, requiere

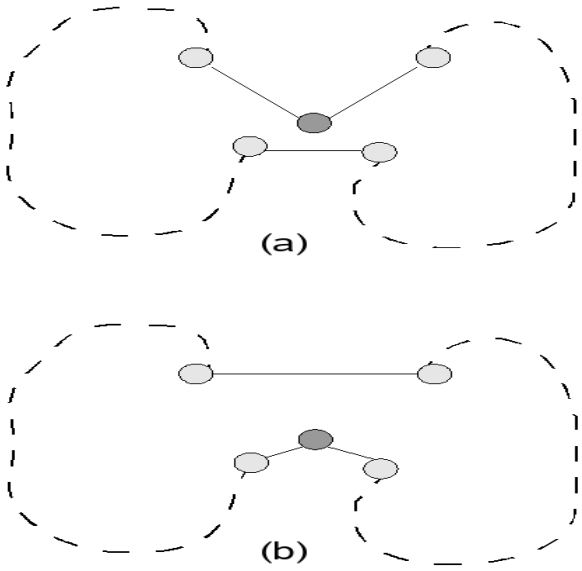

Figura 4.6: Transformación $2\frac{1}{2}$-cambios

que toda la estructura de la ruta TSP este disponible para un procesador. Sin embargo, es posible considerar un esquema de procesamiento paralelo en cascada, donde cada procesador ubicado en un nivel se encarga de la revisión y ejecución de los cambios en una sección de la ruta TSP. La labor de un procesador en un nivel inferior se detiene cuendo la sección a su cargo no acepta nuevas transformaciones $2\frac{1}{2} - cambios$.

Supongamos que, en un cierto nivel de la cascada, los procesadores P1 y P2 se encargan del mejoramiento $2\frac{1}{2}$-optimo de 2 secciones vecinas de la ruta TSP. Cada procesador verifica y ejecuta los cambios en su sección, tal como se muestra en la figura 4.7(a). Aquí los arcos, candidatos a salir, están remarcados. Una vez alcanzada la $2\frac{1}{2}$-optimalidad en cada sección, se pasa a un nivel superior, donde un solo procesador (P1), se encarga de procesar una nueva sección obtenida al concatenar las 2 secciones, $2\frac{1}{2}$-optimas, del nivel inferior. En el procesamiento de la nueva sección, figura 4.7(b), sólo deben verificarse aquellos cambios que no han sido verificados en el nivel anterior.

Es decir, parte del trabajo de verificación para el procesamiento de la nueva sección, por parte de un procesador en el nivel superior, ya ha sido realizado por 2 procesadores en el nivel anterior. El procesador que tranfiere su cadena (P2) permanece en espera.

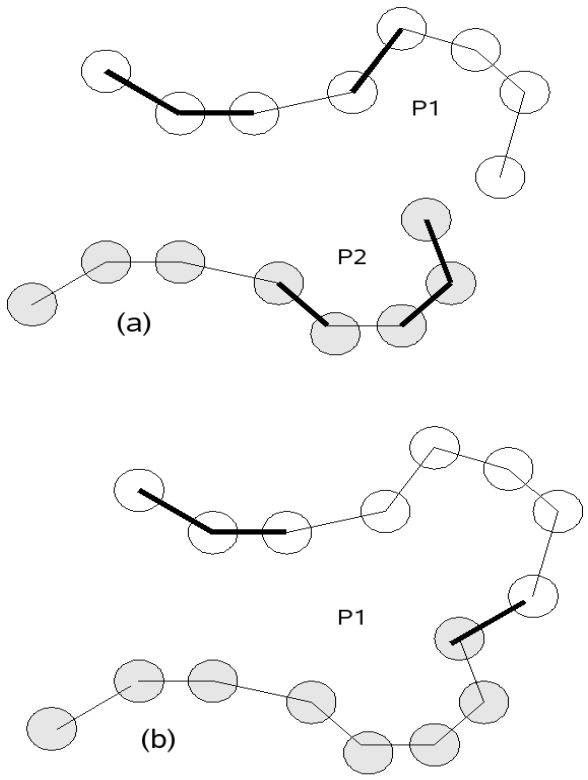

Figura 4.7: $2\frac{1}{2}$-cambios en secciones

El esquema de procesamiento en cascada se ilustra en la figura 4.8. Aquí se muestran 4 niveles de procesamiento. En el nivel más bajo (nivel 0) intervienen 8 procesadores(círculos numerados), realizando la transformación $2\frac{1}{2}$-optima de 8 secciones de la ruta TSP. En este nivel cada procesador detiene el procesamiento al obtener una sección $2\frac{1}{2}$-optima. Cuando el procesador par $2r$ detiene el procesamiento, envía su sección al procesador impar $2r-1$ para su concatenación y posterior procesamiento en el nivel superior (nivel

1). El envío de una sección $2\frac{1}{2}$-optima, dentro de un mismo nivel se representa mediante una flecha horizontal punteada. Un procesador que realiza uno de estos envíos, permanece en espera.

Si en los niveles intermedios (1 y 2) un procesador realiza cambios durante una vuelta de verificación sobre su sección, entonces debe dividir la sección en 2 partes de igual longitud, conservar una de ellas y enviar la otra al correspondiente procesador inactivo que espera en un nivel inferior. El procesador activo desciende de nivel y los 2 procesadores inician el procesamiento de las secciones asignadas. El envío de una seccion de un nivel superior a un nivel inferior se representa con una flecha descendente en línea sólida.

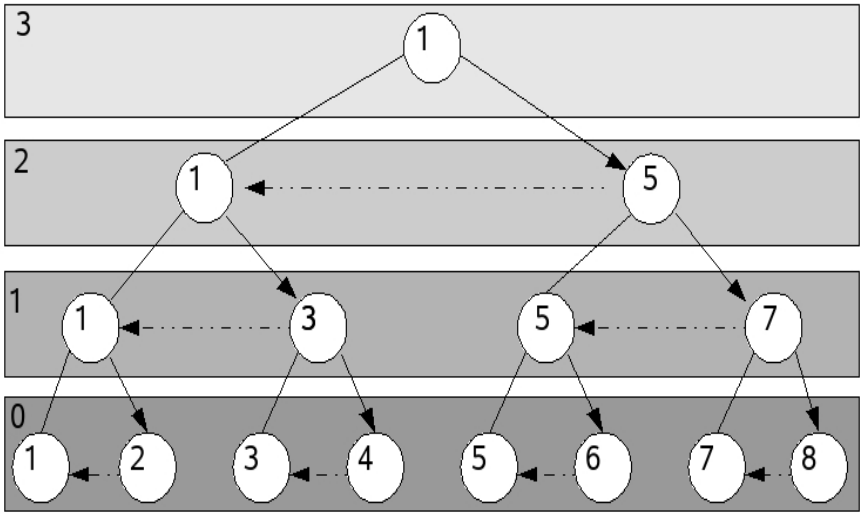

Figura 4.8: Cascada de procesadores

La idea, en esta estrategia, es ocupar el mayor número de procesadores y así realizar el mayor número de verificaciones y cambios simultáneamente. Esto se logra privilegiando el procesamiento en los niveles inferiores. La realización de cambios en los niveles 1,2 y 3, probablemente, generará la necesidad de nuevos cambios. Se quiere que los nuevos cambios sean controlados por

más procesadores.

En este esquema las secciones suben concatenandose y bajan dividiendose. En el nivel 0 se realizan todas las vueltas de verificación y cambios hasta generar secciones $2\frac{1}{2}$-optimas. En los niveles intermedios 1 y 2 se realiza una sola vuelta de verificación y cambios. Si no hay cambios se sube de nivel, si hay cambios se baja de nivel. En el nivel superior, nivel 3, donde un solo procesador verifica la $2\frac{1}{2}$-optimalidad de todo la ruta TSP, se finaliza el procesamiento cuando se completa una vuelta de verificación con la realización de menos de 10 cambios.

Para el procesamiento en cascada es necesario que el número de procesadores sea potencia de 2. Es por esta razón que PGSOM ha sido implementado sobre 8 procesadores.

4.4. Aspectos de implementación del PGSOM

Message Passing Interface(MPI) es un estandar de librería de pase de mensajes. Constituye una especificación para desarrolladores y usuarios. En si misma no es una librería, más bien es una especificación de como debe ser una librería de pase de mensajes. Siguiendo esta especificación se han desarrollado librerías para C/C++ y Fortran.

MPI no es un estandar IEEE o ISO, pero se ha convertido en el estandar de la industria para la escritura de programas con pase de mensajes.

La heurística paralela para el GSOM, PGSOM, ha sido codificada en lenguaje C y MPI.

MPI ofrece una abrumadora variedad de instrucciones que permiten la composición de programas que exhiben comportamientos muy complejos. Una vez que se tiene una idea clara de la lógica del programa paralelo, el siguiente paso es seleccionar las instrucciones MPI que permiten realizar una implantación. La siguiente lista presenta un conjuto de instrucciones MPI (bastante sencillas), claves en esta implantación del PGSOM.

- `MPI_Bcast()`:permite la difusión de un mensaje a todos los procesado-

res.

- `MPI_Send()`:permite el envío de un mensaje a un procesador.
- `MPI_Recv()`:permite la recepción de un mensaje.
- `MPI_Allreduce()`:cada procesador aporta data para la realización de una operación. Cada procesador recibe el resultado de la operación.
- `MPI_Iprobe()`:permite determinar si se tiene un mensaje por recibir.

Estas instrucciones son utilizadas de la siguiente manera:

- `MPI_Bcast()`:un solo procesador es el encargado de generar el conjunto de estímulos que se utiliza durante la fase de actualización. Esta instrucción es usada para que cada procesador tenga su propia copia de este conjunto.
- `MPI_Send()`:se usa en las actualizaciones fronterizas, es decir cuando resulta ganadora una UP que está en el extremo de la sección controlada por un procesador; en tal circunstancia, el procesador sólo puede actualizar la UP ganadora y una de sus vecinas, la otra vecina se encuentra en una sección administrada por otro procesador. El procesador asociado a la UP ganadora debe enviar un mensaje de actualización a un procesador vecino, para que actualice el vector de pesos de la UP correspondiente. Esta instrucción también es usada repetidamente durante la etapa de mejoramiento que realiza $2\frac{1}{2}$-cambios, aquí es frecuente el intercambio de secciones de ruta TSP, entre los procesadores. Es una instrucción bloqueante.
- `MPI_Recv()`:esta instrucción es complementaria a `MPI_Send()` y se utiliza en los mismos casos, para realizar la recepción del mensaje. Es una instrucción bloqueante.

- `MPI_Allreduce()`:se utiliza para econtrar la(s) UP(s) más ganadora(s) global(es), mediante la determinación del máximo de los contadores de las UPs más ganadoras de cada sección.

- `MPI_Iprobe()`:las actualizaciones fronterizas se presentan de manera asíncrona; un procesador no sabe cuando va a recibir un mensaje por actualización fronteriza. Esta instrucción permite preguntar, sin bloquearse, si se tiene un mensaje de actualización fronteriza. Si se tiene, entonces se ejecuta la instrucción `MPI_Recv()`. Durante el procesamiento de un estímulo cada procesador ejecutará la instrucción `MPI_Iprobe()` con una cierta probabilidad. Esta probabilidad es una función decreciente del número de UPs en el anillo y es una estimación de la frecuencia con la que un estímulo provoca que la UP ganadora sea una UP extrema y vecina en una sección vecina.

Capítulo 5

Experimentación y Resultados

En este capítulo se presentan los resultados obtenidos por la heurística PGSOM reportando la calidad de la solución y el tiempo de ejecución al resolver las siguientes instancias de la TSPLIB:

1. pr1002
2. nrw1379
3. pr2392
4. pcb3038
5. fnl4461
6. pla7397
7. rl11849
8. d15112
9. d18512
10. pla33810

El número dentro del nombre de cada instancia indica el número de ciudades N.

Primero se realiza una comparación en cuanto a tiempo total de ejecución necesario para dar respuesta a algunas de las instancias anteriores, con *branch and cut* y con PGSOM. El cuadro 5.1 permite hacer tal comparación. Los tiempos de ejecución de *branch and cut* han sido tomados de [3] y no incluyen un tiempo adicional dedicado a preprocesamiento. Si se adiciona el tiempo de preprocesamiento las instancias mayores, d15112 y d18512, deberían reportar un tiempo de ejecución total de 144 horas.

Cuadro 5.1: Tiempo de ejecución de *branch and cut* y PGSOM

Instancia	Tiempo*branch and cut* [horas]	Tiempo PGSOM [horas]
pcb3038	8.6	0.020
fnl4461	3.4	0.040
pla7396	106.9	0.085
d15112	72.0	0.150
d18512	72.0	0.213

Es muy claro que PGSOM presenta una gran ventaja sobre *branch and cut* en cuanto al tiempo de ejecución. Esta es una característica que debe poseer una buena heurística para la solución aproximada del TSP.

Seguidamente los resultados de PGSOM son comparados con el desempeño de la heurística GSOM*. Se ofrece inicialmente una comparación donde se considera el tiempo total de ejecución, luego se presenta una comparación donde las heurísticas GSOM* y PGSOM se dividen en dos partes: la primera, va desde el inicio hasta la completación del mejoramiento 2-optimo, la segunda parte es el mejoramiento $2\frac{1}{2} - optimo$.

Finalmente se presentan los resultados de la heurística PGSOM ejecutada sobre la instancia más grande en la TSPLIB: pla85900. De esta instancia no se conoce, a la fecha, una ruta optima. Para el cálculo de los errores porcentuales, se usa el costo de la mejor ruta TSP encontrada hasta ahora.

Se ha repetido el anterior esquema de experimentación, donde para cada instancia se realizan 10 corridas y se registra el error porcentual y el tiempo de ejecución. A partir de este registro se calculan los valores mayor, menor y promedio del error porcentual y del tiempo de ejecución para cada instancia.

5.1. Resultados considerando el tiempo total de ejecución

Se considera aquí la ejecución total de las heurísticas GSOM* y PGSOM. En el cuadro 5.2 se comparan los menores errores porcentuales alcanzados por GSOM* y PGSOM.

Cuadro 5.2: Errores porcentuales de GSOM* y PGSOM

Instancia	Error GSOM*	Error PGSOM
pr1002	4.590	3.863
nrw1379	4.834	4.549
pr2392	6.235	6.207
pcb3038	7.825	7.698
fnl4461	5.377	5.936
pla7397	7.330	8.152
rl11849	11.186	12.128
d15112	6.795	7.209
d18512	7.243	7.850
pla33810	9.922	10.622

Se destaca, en general, la similitud del porcentaje de error alcanzado por ambas heurísticas. Sin embargo, se aprecia que más alla de las 3038 ciuadades, PGSOM, consistentemente, presenta un error porcentual ligeramente superior al error correspondiente a GSOM*. La diferencia en ningún caso es mayor que 1 y alcanza su máximo valor con la instancia rl11849. Esta

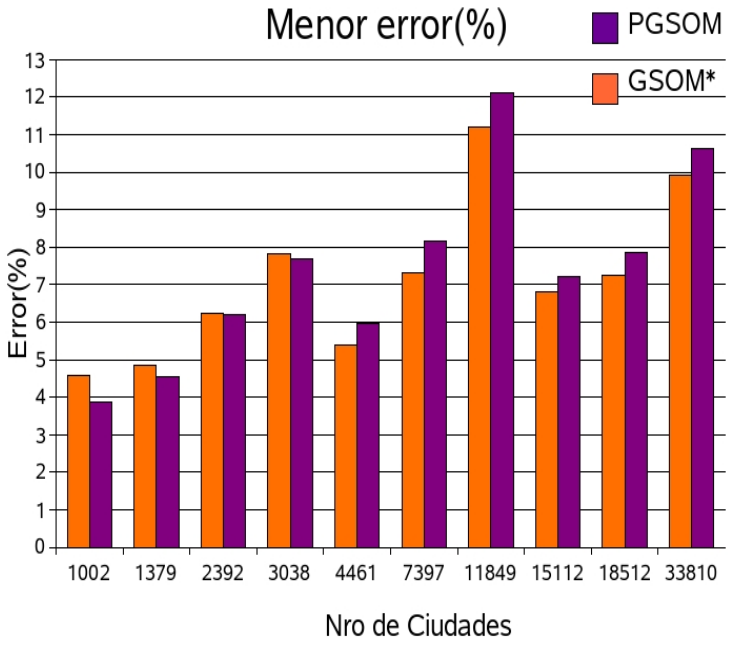

Figura 5.1: Menor error porcentual en 10 corridas de GSOM* y PGSOM

instancia resulta particularmente difícil para ambas heurísticas. Los errores alcanzados con rl11849 superan los errores registados para todas las instancias, incluso las 2 de mayor tamaño:pla33810, como se aprecia en el cuadro 5.2 y pla85900, como se verá más adelante.

En general las instancias que corresponden a circuitos impresos resultan difíciles de tratar.

En la figura 5.1 se muestra una representación gráfica de la data contenida en el cuadro anterior. La figura permite apreciar la similitud de los errores porcentuales de las 2 heurísticas y destaca el pico alcanzado en la instancia de 11849 ciudades.

La ligera disminución en la calidad de los resultados que presenta PGSOM respecto a GSOM* se ve compensada por el mejor desempeño en cuanto a tiempo de ejecución. La figura 5.2 muestra el tiempo de ejecución prome-

dio de ambas heurísticas para cada una de las instancias consideradas. Se observa un crecimiento moderado en el tiempo de ejecución de PGSOM que contrasta con un crecimiento mucho más fuerte de GSOM*. Esta moderación en el crecimiento del tiempo de ejecución le permite a PGSOM ofrecer una buena solución para la instancia pla85900, en un tiempo muy razonable.

Los tiempos de ejecución representados por ambas curvas revelan crecimientos aproximadamente cuadráticos, con una pendiente muy suave. PGSOM aventaja a GSOM* por presentar una pendiente considerablemente menos pronunciada. Este crecimeinto cuadrático se debe al mejoramiento con transformaciones $2\frac{1}{2}$-cambios.

Se debe destacar que en la menor instancia considerada, pr1002, GSOM* corre más rápido que PGSOM. Esto se debe a que en la etapa de establecimiento de la partición del conjunto de ciudades, PGSOM realiza muchas comunicaciones. Esta etapa finaliza cuando el anillo global ha incorporado 250 nuevas UPs y alcanza un tamaño de 1250 UPs. Las etapas siguientes al particionamiento, hasta el mejoramiento $2\frac{1}{2}$-optimo, resultan mucho más rápidas debido a que las comunicaciones se reducen drásticamente. Al fijar 1250 como el tamaño que debe alcanzar el anillo en la etapa de particionamiento de las ciudades, para una instancia de tamaño 1002, gran parte de la evolución total del anillo ocurre en la etapa inicial en un solo procesador y en la etapa de particionamiento.

5.2. Resultados considerando el tiempo de ejecución repartido en dos partes

PGSOM incluye dos esquemas de paralelización bien definidos: el primero permite la construcción del anillo y la generación de una ruta TSP $2-optima$, el segundo realiza transformaciones del tipo $2\frac{1}{2} - cambios$. La marcada diferencia entre estos dos esquemas de paralelización justifica la realización de análisis separados con el fin de determinar la eficiencia de cada esquema.

En la sección anterior observamos que PGSOM ofrece soluciones con

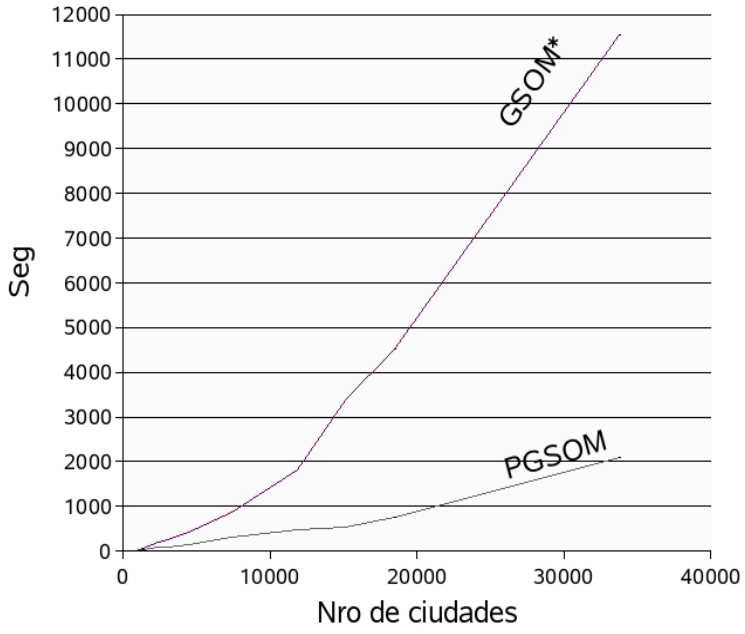

Figura 5.2: Tiempo promedio de ejcución de GSOM* y PGSOM

una calidad ligeramente inferior a la calidad de las soluciones arrojadas por GSOM*. A simple vista resulta fácil pensar que esta disminución en la calidad debe originarse en la etapa de construcción del anillo, dado que la dinámica paralela difiere de la dinámica original por el particionamiento del conjunto de ciudades que trae con sigo búsquedas de menor radio centradas en neuronas próximas a los extremos de una sección del anillo. Esta particularidad alcanza su mayor expresión cuando resulta ganadora una neurona que constituye un extremo de una sección, en este caso se realiza la búsqueda completa , sobre la sección, a un lado de la neurona pero no se realiza busquda alguna al otro lado de la neurona por que la sección donde debería realizarse la búsqueda está bajo la administración de otro procesador.

En la figura 5.3 se presenta el menor error porcentual alcanzado por GSOM* y PGSOM al completar la primera parte con la generación de una

solución $2-optima$. Contrario a la suposición anterior PGSOM, en su primera parte, alcanza soluciones que compiten favorablemente con las soluciones alcanzadas por GSOM*, en esta misma parte. Cabe suponer entonces que el

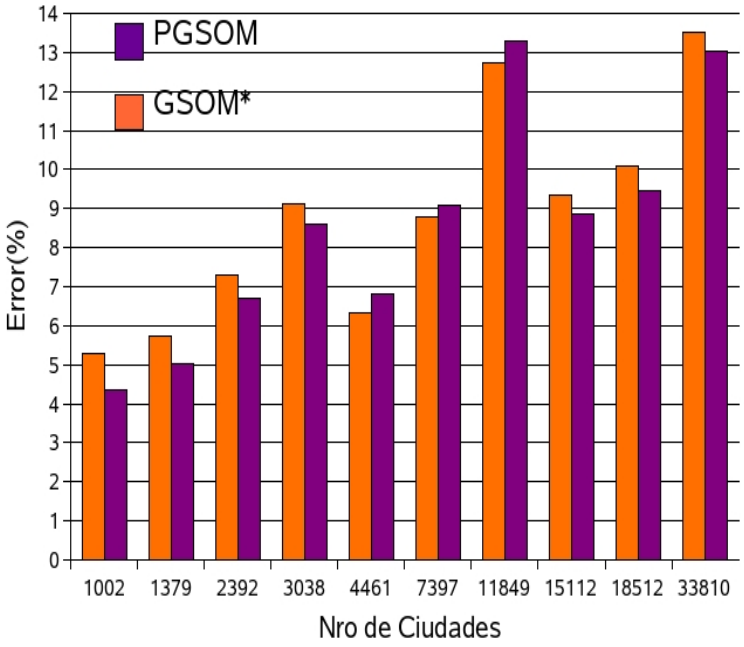

Figura 5.3: Menor error porcentual de la solución inicial 2-optimal de GSOM* y PGSOM

proceso de mejoramiento mediante transformaciones $2\frac{1}{2}-cambios$ en paralelo resulta menos efectivo que su similar en monoprocesamiento. Sin embargo, esta suposición luce poco probable, pues en ambos casos se usa el mismo criterio de parada que resulta directamente relacionado con la calidad de la solución alcanzada. Este criterio es la realización de una vuelta de verificación y transformación, de la ruta TSP, donde se realicen menos de 10 cambios. La búsqueda de la causa de la ligera disminución en la calidad de las soluciones tiene 2 alternativas inmediatas por confirmar o descartar: la primera es que el proceso de mejoramiento paralelo en cascada tiene menor poder de mejo-

ramiento que su contraparte en un solo procesador, la otra alternativa es que aún cuando las soluciones iniciales 2-optimales de PGSOM son tan buenas como las de GSOM*, también resultan menos suceptibles a ser mejoradas mediante transformaciones $2\frac{1}{2}$-cambios.

La figura 5.4 presenta el tiempo promedio de ejecución de GSOM* y PGSOM desde su inicio hasta la completación de una solución inicial 2-optima. Se observa el mayor distanciamiento entre las dos curvas a medida que aumenta el número de ciudades. Tanto GSOM* como PGSOM presentan curvas muy cercana a la linealidad. PGSOM se destaca por su mayor linealidad y su moderada pendiente.

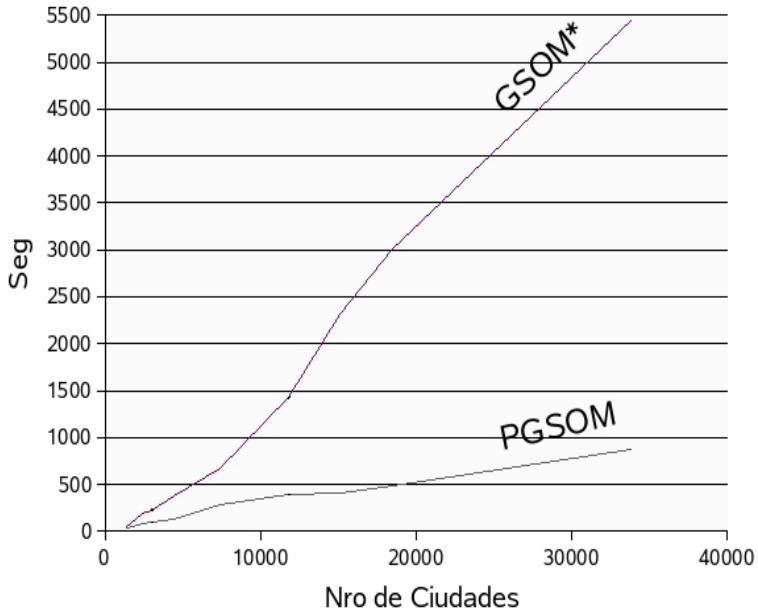

Figura 5.4: Tiempo promedio de generación de solución 2-optima de GSOM* y PGSOM

Como cabe esperar, en la instancia pr1002 GSOM* aventaja a PGSOM y logra obtener la solución inicial 2-optima en menor tiempo. Específicamente

en esta parte inicial GSOM* promedia un tiempo de 21.2seg mientras que PGSOM realiza el mismo trabajo en 29.7seg.

La parte de mejoramiento mediante transformaciones $2\frac{1}{2}$-cambios debe presentar un crecimiento lejano a la linealidad pues su computo involucra la realización de 2 ciclos anidados. En el primero de ellos una subsección de 2 arcos unidos rota sobre la ruta TSP, en el segundo se consideran todos los arcos restantes que estén separados de la subsección de dos arcos unidos (solo 2 arcos no satisfacen esta restricción). Este anidamiento produce un efecto multiplicativo que genera un crecimiento de orden cuadrático en el tiempo de procesamiento al incrementar el número de ciudades. Con más claridad se dice que la heurística es de orden N^2, con N: el número de ciudades. El esquema de paralelización que incluye PGSOM para realizar esta tarea pretende amortiguar este crecimiento. Estas observaciones son corroboradas en la figura 5.5 donde se presenta el tiempo promedio de ejecución de GSOM* y PGSOM durante la parte de mejoramiento mediante transformaciones $2\frac{1}{2}$-cambios.

La figura 5.6 presenta una ruta TSP para pla33810 encontrada por PGSOM.

5.3. Resultados de PGSOM sobre pla85900

Sobre pla85900 se han realizado 10 corridas de PGSOM. Se han obtenido soluciones de buena calidad en un tiempo muy razonable, en promedio 5 horas y media. En el cuadro 5.3 se muestran los errores porcentuales mínimo, promedio y máximo que han sido alcanzados en las dos partes en que se ha dividido la ejecución de PGSOM. Se destaca que el error final no ha presentado mayor crecimiento.

En el cuadro 5.4 se presentan los valores mínimo, promedio y máximo del tiempo de ejecución total de PGSOM sobre pla85900, así como de la ejecución de su primera y segunda parte. Se destaca la gran proporción de tiempo de ejecución que se dedica a la parte de mejoramiento $2\frac{1}{2}$-optimo, en

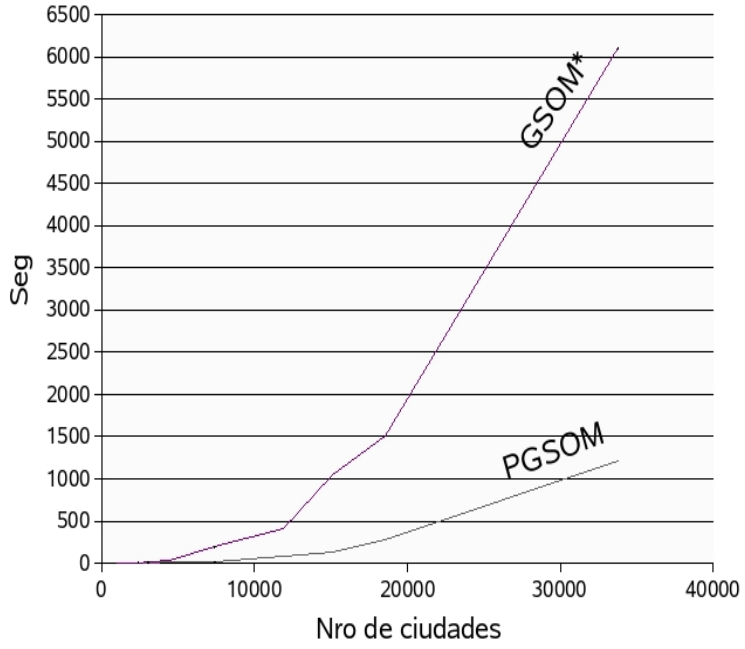

Figura 5.5: Tiempo promedio de transformaciones $2\frac{1}{2}$-optimas de GSOM* y PGSOM

promedio corresponde al 90 por ciento del tiempo total de ejecución. Esta es una consecuencia del crecimiento lineal de la primera parte y cuadrático de la segunda.

En la figura 5.7 se superponen las curvas de tiempo promedio de ejecución de las 2 partes, incluyendo los resultados alcanzados con pla859000. Esta figura confirma el crecimiento cuadrático de la parte 2 (mejoramiento $2\frac{1}{2}$-optimal) y el lento crecimiento lineal de la parte 1.

El mayor trabajo de mejoramiento sobre la ruta TSP inicial es realizada por el mejoramiento que usa transformaciones 2-cambios, este mejoramiento inicial es muy rápido y para instancias grandes es capaz de mejorar la ruta TSP inicial al pasar de un error porcentual inicial que puede superar el 50 por ciento a un error por debajo del 15 por ciento. El segundo mejoramien-

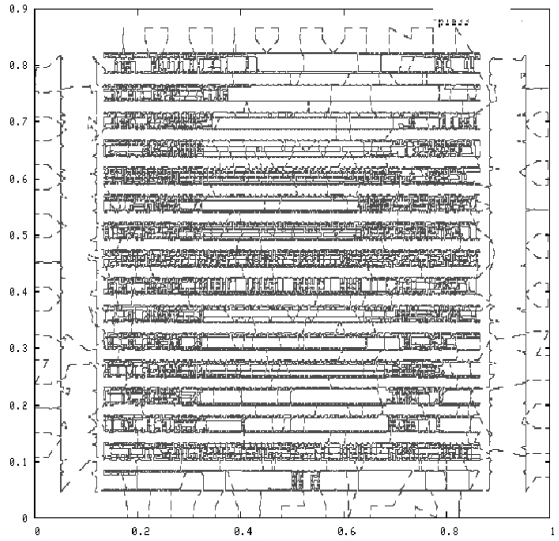

Figura 5.6: Una ruta TSP para pla33810 propuesta por PGSOM

to mediante transformaciones $2\frac{1}{2}$-cambios, para instancias grandes, consume una gran cantidad de tiempo y realiza una mejora que no supera los 4 puntos porcentuales. Sin embargo, esta reducción aportada por el mojoramiento $2\frac{1}{2}$-optimo es muy importante para ofrecer soluciones de mayor calidad.

El cuadro 5.3 revela la gran ayuda que brinda el mejoramiento $2\frac{1}{2}$-optimo para la obtención de soluciones de calidad. La figura 5.7 revela la gran diferencia en el comportamiento de las curvas de crecimiento de tiempo de ejecución de las 2 partes de PGSOM.

El PGSOM puede ser considerado como una heurística capaz de ofrecer soluciones de dos niveles de calidad. En el primer nivel se ofrecen soluciones de menor calidad en un tiempo muy corto realizando solo el mejoramiento 2-optimo. En el segundo nivel se ofrecen soluciones de mayor calidad en un tiempo más largo realizando el mejoramiento 2-optimo y el mejoramiento $2\frac{1}{2}$-optimo. El nivel de calidad debe ser fijado en cada situación práctica, teniendo en cuenta las exigencias de calidad de la solución y las restricciones

Cuadro 5.3: Errores porcentuales de PGSOM sobre pla85900

	Error 2-opt	Error final
Mínimo	14.491	10.442
Promedio	15.417	11.135
Máximo	16.012	11.863

Cuadro 5.4: Tiempos de ejecución, en segs, de PGSOM sobre pla85900

	t 2-optimal[seg]	t $2\frac{1}{2}$-optimal[seg]	t total[seg]
Mínimo	1603	13848	15451
Promedio	1809	18138	19947
Máximo	2198	23362	25560

de tiempo.

Sin embargo, es evidente la necesidad de actuar sobre el mejoramiento $2\frac{1}{2}$-optimo. Es posible atenuar el crecimiento usando ciertos principios de localidad para racionalizar la verificación de los cambios. Sin embargo, la esencia cuadrática del mejoramiento $2\frac{1}{2}$-optimo siempre estará presente. Una idea más interesante es la sustitución del mejoramiento $2\frac{1}{2}$-optimo con un nuevo método que funcione en tiempo lineal o logarítmico.

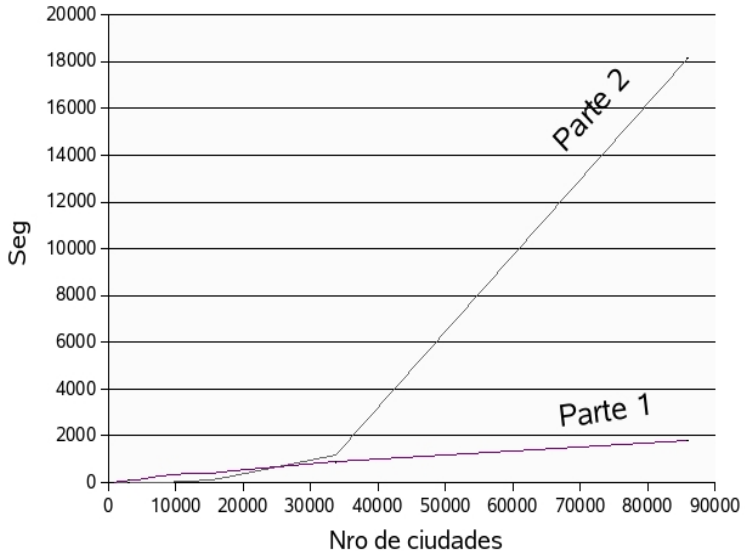

Figura 5.7: Tiempo de ejecución promedio de las 2 partes de PGSOM

La figura 5.8 muestra una ruta TSP propuesta por PGSOM para pla85900.

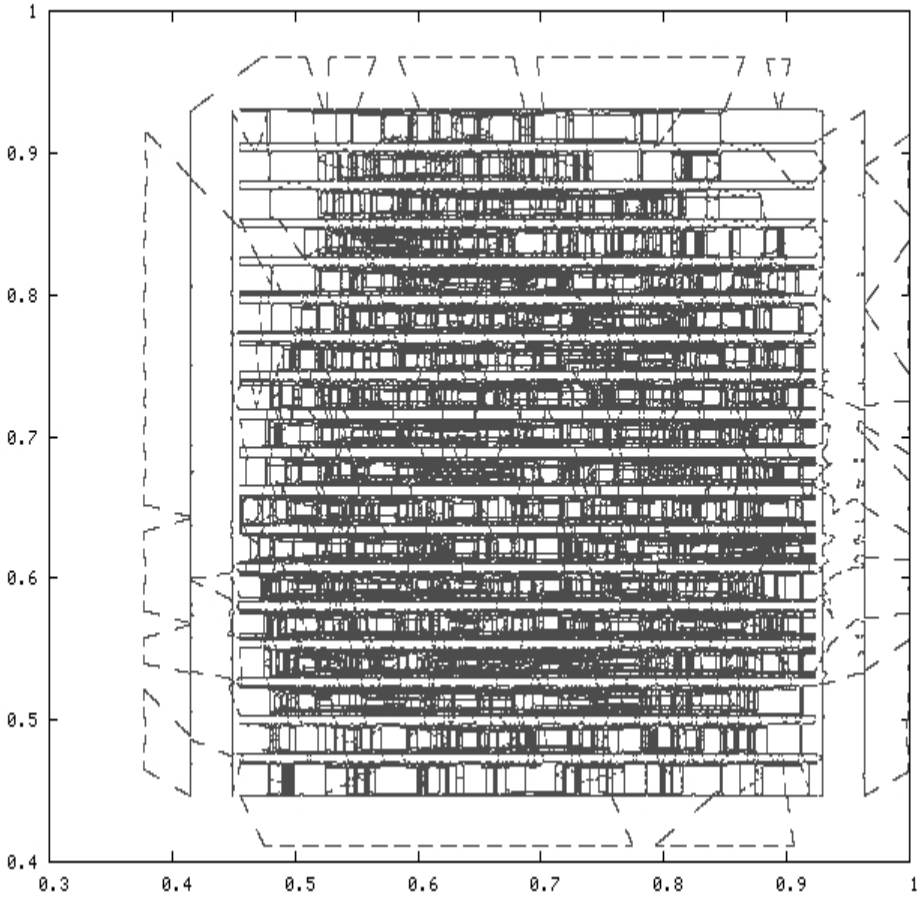

Figura 5.8: Una ruta TSP para pla85900 propuesta por PGSOM

Capítulo 6

Discusión y Conclusiones

Las heurísticas que usan mapas autoorganizativos para encontrar una solució aproximada al TSP se han caracterizado hasta ahora por tratar problemas de tamaño moderado. En este trabajo se ha desarrollado una versión paralela de la heurística GSOM sobre el TSP, que aumenta considerablemente el tamaño de las instancias tratables con una heurística basada en mapas autoorganizativos.

GSOM usa un mapa autoorganizativo creciente de una dimensión y topología de anillo para construir soluciones iniciales del TSP, que luego son mejoradas mediante la aplicación de transformaciones del tipo 2-optimo.

En la heurística paralela del GSOM, denominada PGSOM, se distribuye entre 8 procesadores el trabajo computacional asociado a la evolución del anillo, la contrucción de la ruta TSP inicial y el mejoramiento.

Antes de construir la heurística PGSOM, se ha propuesto una heurística GSOM modificada donde se incrementa la tasa de aprendizaje, se reduce el número de presentaciones previas a la inserción de una nueva UP, se incluye una caché de UPs ganadoras y se sustituyen las búsquedas globales por búsquedas locales para encontrar la neurona ganadora. Además, el mejoramiento 2-optimo es acompañado de un mejoramiento con transformaciones de tipo $2\frac{1}{2}$-cambio. Estas modificaciones a la heurística GSOM permiten obtener, en general, soluciones de mayor calidad y sobre todo producen una gran

reducción en tiempo de procesamiento que hace posible abordar problemas de mayor tamaño, como la instancia pla33810 de la TSPLIB.

PGSOM se construye a partir de la hurística GSOM modificada. En PGSOM, un anillo evolucionado en un solo procesador, hasta alcanzar un cierto tamaño, fijado en 1000 UPs, se divide en 8 secciones. Cada sección tiene un tamaño de 1000/8 UPs y es asignada a un procesador distinto, donde es evolucionada de manera casi independiente.

Además de tener una sección del anillo asignada, cada procesador maneja su propio subconjunto de ciudades. De esta forma cada procesador se ocupa del desarrollo de su sección del anillo, la construcción de una sección de la ruta TSP inicial y de su mejoramiento 2-optimo. La actividad casi independiente de los 8 procesadores hace que el tour inicial 2-optimal sea obtenido en un tiempo muy corto.

El mejoramiento mediante transformaciones del tipo $2\frac{1}{2}$-cambios, se realiza siguiendo un esquema de paralelización en cascada donde se promueve la participación del mayor número posible de procesadores en la realización de los ciclos de búsqueda y transformación. Durante este mejoramiento se presenta una interacción mucho mayor entre los procesadores, sin embargo el esquema de paralelización en cascada ha logrado una reducción considerable del tiempo de ejecución.

La reducción en tiempo de ejecución del PGSOM comienza a percibirse para problemas de tamaño superior a 1002 ciudades. La velocidad de PGSOM ha permitido el tratamiento, en tiempo promedio de 5 horas y media, de la instancia de mayor tamaño en la TSPLIB: pla85900. En general PGSOM tiene un tiempo de ejecución de varios ordenes de magnitud menor que el tiempo de ejecución de los algoritmos exactos más rápidos. Este hecho y la calidad de las soluciones convierten a PGSOM en una heurística competitiva.

El mayor trabajo de mejoramiento sobre la ruta TSP inicial es realizada por el mejoramiento que usa transformaciones 2-cambios, este mejoramiento inicial es muy rápido y para instancias grandes es capaz de mejorar la ruta TSP inicial al pasar de un error porcentual inicial que puede superar el 50 por

ciento a un error cercano 15 por ciento. El segundo mejoramiento mediante transformaciones $2\frac{1}{2}$-cambios, para instancias grandes, consume una gran cantidad de tiempo y reliza una mejora que no supera los 4 puntos porcentuales. Sin embargo, esta reducción aportada por el mojoramiento $2\frac{1}{2}$-optimo es muy importante para ofrecer soluciones de mayor calidad.

El PGSOM puede ser considerado como una heurística capaz de ofrecer soluciones de dos niveles de calidad. En el primer nivel se ofrecen soluciones de menor calidad en un tiempo muy corto realizando solo el mejoramiento 2-optimo. En el segundo nivel se ofrecen soluciones de mayor calidad en un tiempo mayor, realizando el mejoramiento 2-optimo y el mejoramiento $2\frac{1}{2}$-optimo. El nivel de calidad debe ser fijado en cada situación práctica, teniendo en cuenta las exigencias de calidad de la solución y las restricciones de tiempo.

La gran reducción en tiempo de PGSOM vino acompañada de una ligera disminución en la calidad de la solución para instancias mayores de 3038 ciudades, al comparar con la heurística GSOM modificada. La detección y corrección de la(s) causa(s) de esta ligera disminución en la calidad de los resultados marca una vía a seguir en la realización de un futuro trabajo.

La primera parte de PGSOM muestra un crecimiento lineal del tiempo de ejecución con una pendiente muy moderada mientras que la segunda parte (el mejoramiento $2\frac{1}{2}$-optimo) tiene un crecimiento cuadrático. Esto hace que el mejoramiento $2\frac{1}{2}$-optimo consuma una gran proporción del tiempo total de ejecución para instancias grandes del TSP.

Es necesario tratar de equilibrar las proporciones de consumo de tiempo, por lo que futuros trabajos pueden abordar la reducción del tiempo de procesamiento del mejoramiento $2\frac{1}{2}$-optimo o su sustitución por un nuevo método de mejoramiento que presente un crecimiento lineal o logarítmico de su tiempo de ejecución.

En definitiva (y sin considerar las posibles mejoras) PGSOM se presenta como una muy buena heurística para la solución aproximada de instancias de gran tamaño del TSP.

Bibliografía

[1] E. Aarts and J.K. Lenstra. *Local Search in Combinatorial Optimization*. John Wiley and Sons, London, 1997.

[2] E. Arsugua. Topology preservation in som. *Transactions on Engineering,Computing and Technology*, 15:187 – 190, 2006.

[3] W. Cook, D. Espinoza, and M. Goycoolea. A study of domino-parity and k-parity constraints for the tsp. *Lecture Notes in Computer Science*, 3509/2005:452 – 467, 2005.

[4] C. Garcia and J. Moreno. An efficient heuristic for the travelling salesman problem based on a growing som-like algorithm. *Adaptive and natural computer algorithms*, pages 177 – 180, 2005.

[5] Y. Grodzinsky and K. Amunts. *Broca's Region*. Oxford Universty Press, New York, 2006.

[6] H. Jin, K. Leung, M. Wong, and Z. Xu. A efficient sel-organizing map designed by genetic algorithms for the traveling salesman problem. *IEEE Transactions on System,Man, and Cybernetics*, 33:877 – 888, 2003.

[7] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59 – 69, 1982.

[8] T. Kohonen. The self-organizing map. *Procedings of the IEEE*, 78:1464 – 1480, 1990.

[9] T. Kohonen, O. Simula, A. Visa, and J. Kangas. Engineering aplications of the self-organizing map. *Procedings of the IEEE*, 84:1358 – 1384, 1996.

[10] S. Lin and B.W. Kernighan. An effective heuristic for the travelling salesman problem. *Operations Research*, 21:498 – 516, 1973.

[11] G. Reinelt. Tsplib- a travelling salesman problem library. *ORSA Journal on Computing*, 3:376 – 384, 1991.

[12] B. Rosen, R. Buckner, and A. Dale. Event-related functional mri: Past, present, and future. *Nat. Acad. Sci*, 95:773 – 780, 1998.

[13] M. Schwardt and J. Deethloff. Solving a continuous location-routing problem by use of a self-organizing map. *International Journal of Physical Distribution Logistics Management*, 35:390 – 408, 2005.

[14] K. Smith. An argument for abandoning the travelling salesman problem as a neural-network benchmark. *IEEE Transactions on Neural Networks*, 7:1542 – 1544, 1996.

[15] J. Stiles, J. Reilly, B. Paul, and P. Moses. Cognitive development following early brain injury:evidence for neural adaptation. *ELSEVIER TRENDS in Cognitive Science*, 9:136 – 143, 2005.

[16] F. Carvalho Vieira, A. Duarte Doria Neto, and J. Costa. An efficient approach to the travelling salesman problem using self-organizing maps. *Int. J. Neural Syst*, 13:59 – 66, 2003.