FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA

CENTRO DE INVESTIGACIÓN EN INGENIERÍA
Y CIENCIAS APLICADAS

ESTRUCTURA HÍBRIDA DE VECINDAD CON BÚSQUEDA TABÚ PARA CALENDARIZACIÓN DE TRABAJOS EN TALLERES DE MANUFACTURA

TESIS PROFESIONAL

PARA OBTENER EL GRADO DE:

MAESTRÍA EN INGENIERÍA Y CIENCIAS APLICADAS

P R E S E N T A:

IIF. WENDY TORRES MANJARREZ

ASESOR:

DR. MARCO ANTONIO CRUZ CHÁVEZ

CUERNAVACA, MOR. DICIEMBRE 2013

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

CIICAp

SECRETARIA DE INVESTIGACIÓN Y POSGRADO FCQeI-CIICAp

Av. Universidad 1001 Col. Chamilpa, Cuernavaca, Morelos, México, CP 62209
Tel. 01 (777) 329 70 84, ext. 6232.

OFICIO No 215/ 2013
ASUNTO: APROBACIÓN DE TESIS

C. WENDY TORRES MANJARREZ
P R E S E N T E

Por este conducto le notifico que su tesis de Maestría titulada,

"ESTRUCTURA HÍBRIDA DE VECINDAD CON BÚSQUEDA TABÚ PARA LA CALENDARIZACIÓN DE TRABAJOS EN TALLERES DE MANUFACTURA"

fue aprobada en su totalidad por el jurado revisor y examinador integrado por los ciudadanos

NOMBRE	FIRMA
DRA. MARGARITA TECPOYOTL TORRES	
DR. MARTIN HERIBERTO CRUZ ROSALES	
DR. MARTIN GERARDO MARTINEZ RANGEL	
DR. MARCO ANTONIO CRUZ CHAVEZ	
DR. FREDY JUAREZ PÉREZ	

Por consiguiente, se autoriza a editar la presentación definitiva de su trabajo de investigación para culminar en la defensa oral del mismo.

Sin otro particular aprovecho la ocasión para enviarle un cordial saludo.

A T E N T A M E N T E
"POR UNA HUMANIDAD CULTA"

DRA. ELSA CARMINA MENCHACA CAMPOS
SECRETARIA DE INVESTIGACIÓN Y POSGRADO
DE LA FCQeI-CIICAp

Resumen

En esta investigación se desarrolló una estructura de vecindad híbrida, la cual se implementó para el problema de calendarización de trabajos en un taller de manufactura. Se realizó un análisis de sensibilidad de los parámetros de control de las estructuras de vecindad, lo que permitió aplicar la metodología de sintonización, que hace que la estructura híbrida de vecindad se desempeñe mejor tanto en eficiencia como en eficacia.

El análisis de resultados se llevó a cabo en base a pruebas de eficiencia y eficacia de las estructuras de vecindad y de la estructura híbrida de vecindad propuesta, esto con el objetivo de conocer la calidad de la solución que dichas estructuras daban a las instancias de pruebas. Las pruebas experimentales se realizaron tomando en cuenta siete instancias para el problema de calendarización de trabajos en talleres de manufactura. De acuerdo con los resultados obtenidos, la estructura híbrida de vecindad demostró ser competitiva tanto en eficiencia como en eficacia.

Abstract

In this research a hybrid neighborhood structure was implemented to solve the Job shop Scheduling problem. The sensitivity analysis to the control parameters was performed and a tuning methodology was proposed and implemented on the Job shop Scheduling problem, allowing the hybrid neighborhood structure to work with the best possible performance in both efficiency and effectiveness.

The analysis of results was carried out based on evidence of effectiveness and efficiency of the structures of neighborhood and neighborhood hybrid structure proposed, this in order to know the quality of the solution structures such instances gave evidence. Experimental tests were performed taking into account seven instances for the Job Shop Scheduling problem. According to the hybrid structure neighborhood results proved competitive in both efficiency and effectiveness.

Agradecimientos

A CONACYT por brindarme el apoyo económico sin el cual no hubiera sido posible la realización de mis estudios de maestría.

Al Dr. Marco Antonio Cruz Chávez, asesor de este trabajo de investigación, por su orientación, apoyo, consejos y paciencia.

A los integrantes de mi comité tutoral y revisores de tesis: Dr. Marco Antonio Cruz Chávez, Dra. Margarita Tecpoyotl Torres, Dr. Martin Heriberto Cruz Rosales, Dr. Martin Gerardo Martínez Rangel, Dr. Fredy Juárez Pérez, por sus observaciones y comentarios para la realización de este trabajo de investigación.

Dedicatorias

A Dios por la ayuda y las oportunidades que me ha obsequiado.

A mi hija Julia Abigail Morales Torres por aguantar cuando tenía que estudiar e ir a las clases por la tarde, por ser el motor que impulsa mi vida y ser la razón de cada cosa que hago.

A mis padres Teresa Manjarrez Zavala y Salvador Torres Castañeda, por su cariño, apoyo, comprensión y consejos.

A mi asesor el Dr. Marco Antonio Cruz Chávez, por sus consejos y dirección para lograr la realización de este trabajo.

A mis hermanas y hermano: Lizzeth, Ana, Frida y Chava, por su cariño y palabras de aliento.

A mis compañeros y amigos de la maestría: René, Alan y Alfonso, por su apoyo y ayuda incondicionales, por todos los buenos momentos que pasamos juntos.

A todo el equipo de estudiantes de maestría y doctorado que conforman el grupo de optimización y software, gracias por sus comentarios en los seminarios.

Nomenclatura

Nomenclatura general

BL Siglas del algoritmo Búsqueda Local

TS Siglas en inglés para Búsqueda tabú (Tabú Search)

JSSP Siglas en inglés para el problema de calendarización de trabajos en talleres de manufactura (Job Shop Scheduling Problem).

Nomenclatura del problema JSSP y método de solución

N Número de Trabajos.

M Número de Máquinas.

j Trabajo que necesita ser calendarizado.

i Máquina en la que va a ser procesado un trabajo *j*.

σ Movimiento aplicado por una estructura de vecindad.

Parámetros de control

$\mathbf{T_L}$ Tamaño de la lista tabú.

K Tamaño de la vecindad.

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1 Introducción

El problema de calendarización de trabajos en un taller de manufactura, JSSP (Job Shop Scheduling Problem), ha sido considerado, como uno de los problemas con mayor relevancia en las áreas de manufactura y optimización combinatoria, ya que la administración, así como el manejo eficiente de los recursos, es de vital importancia en las empresas [Meeran, 1998]. Las empresas diseñadas para satisfacer pedidos de lotes de productos o servicios de gran tamaño poseen unos procesos productivos altamente repetitivos. En estos casos la calendarización de tareas se suele utilizar para determinar la secuencia productiva de lotes de un mismo producto o servicio que permita cumplir con fechas de entrega de pedidos. Un taller de manufactura consiste en un conjunto de máquinas diferentes como tornos, fresadoras, taladros, etc. que realizan trabajos, cada trabajo tiene un orden de procesamiento específico a través de las máquinas. La función de la calendarización es la asignación de recursos limitados a tareas a lo largo del tiempo y tiene como finalidad la optimización de una o más funciones objetivo.

El problema de programación del taller de manufactura se puede describir brevemente como sigue. Hay un conjunto de trabajos y un conjunto de máquinas. Cada trabajo consta de una secuencia de operaciones, cada una de las cuales utiliza una de las máquinas con una duración fija. Una vez iniciada, la operación no puede ser interrumpida. Cada máquina puede procesar una sola operación a la vez. El problema es encontrar la secuencia óptima de operaciones a ejecutarse en una máquina que minimice el máximo tiempo de terminación del último trabajo conocido como *Makespan* [Nowicki, 1996].

Este problema, que se ha estudiado durante mucho tiempo, se sabe que es NP-duro [Garey, 1976] y tiene la bien ganada reputación de ser uno de los más difíciles problemas combinatorios considerado hasta la fecha. Un indicio de su dificultad está dado por el hecho de que el famoso problema de 10 x 10 formulado por primera vez por Muth y Thompson en 1963, sólo se resolvió exactamente en 1989 por Carlier y Pinson con un algoritmo de Branch and Bound el cual requería de al menos 5 horas de tiempo de computo en una PRIME 2655 [Dell'Amico, 1993].

1.1 Estado del arte

Las técnicas de solución para este problema van desde simples reglas de prioridad de despacho, como FIFO (First In First Out, primero en entrar primero en salir), y SPT (Small Processing Time, Menor Tiempo de Procesamiento), a técnicas más elaboradas como Branch and Bound (ramificación y acotamiento) (Brucker *et al*. 1994), Busqueda Tabú (Nowicki and Smutnicki 1996), Shifting Bottleneck (cuello de botella) (Balas and Vazacopoulos 1998) y Colonia de Hormigas (Blum and Samples 2004). Metaheurísticas tales como la Búsqueda Tabú y el procedimiento de Shifting Bottleneck han tenido mucho éxito en este problema. Estos enfoques sobresalen en la calidad de la solución obtenida así como en el tiempo del cálculo [Chong *et al*, 2006]. Dentro de la clase de los métodos metaheurísticos, la Búsqueda Tabú, inicialmente propuesta por Glover, parece ser uno de los métodos más prometedores para el problema de calendarización de trabajos (*JSSP*) con el criterio de minimización del *Makespan*. La búsqueda Tabú fue aplicada al JSSP por primera vez por Taillard y cuya principal contribución fue el uso de la estructura de vecindad introducida por Van Laarhoven y propuso una estrategia de estimación rápida [Taillard, 1994]. Taillard observó que este algoritmo tiene una mayor eficiencia en las instancias rectangulares.

Desde entonces, los investigadores han introducido numerosas mejoras en el algoritmo original de Taillard y las contribuciones más importantes incluyen las de Nowicki and Smutnicki, Dell'Amico and Trubian, Barnes and Chambers, entre este método de Búsqueda Tabú se encuentra el algoritmo TSAB (Taboo Search Algorithm with Back Jump Tracking) Algoritmo de búsqueda tabú con regreso, diseñado por Nowicki and Smutnicki, el cual presenta un verdadero avance en la eficiencia y la eficacia para el JSSP (Job Shop Scheduling Problem).

Por ejemplo, encuentra la solución óptima para la notoria instancia FT10 en solo 30s en una computadora personal. La técnica i-TSAB de Nowicki and Smutnicki, es una extensión de su algoritmo anterior TSAB, y representa el actual estado del arte como algoritmo de aproximación para el JSSP y además mejora la mayoría de las cotas de las instancias sin resolver [Yong *et al* 2007]

Para grandes instancias, no existe un algoritmo determinístico que resuelva este tipo de problemas. Por esta razón las metaheurísticas son usadas para buscar el óptimo global del problema del cual se trate, porque pueden generar algoritmos que limitan este tipo de problemas en tiempo polinomial. Para poder evaluar la calidad de las soluciones, estas metaheurísticas requieren encontrar el makespan a cada paso del algoritmo heurístico. Para poder hacer esto se usa un algoritmo de calendarización. Cada vez que la metaheurística obtiene una nueva solución (calendarización), este algoritmo se aplica a la solución para poder asignar un tiempo de inicio a cada una de las operaciones que son parte del JSSP y obtener así el valor del makespan. El makespan se define como el tiempo de finalización de la última operación en el sistema [Cruz *et al*, 2007].

Para mejorar las soluciones encontradas por dicho algoritmo se han utilizado diferentes tipos de estructuras de vecindad, cuyas características influyen en la calidad de las soluciones obtenidas. La selección de una estructura de vecindad apropiada es un aspecto crítico para el diseño de esta clase de algoritmos que permite una mejor exploración y explotación del espacio de soluciones. En la literatura se han encontrado diversas estructuras híbridas de vecindad para problemas NP que a continuación se describen.

Para el problema del Agente Viajero se propuso una estructura de vecindad híbrida para ser aplicada en búsqueda local, la cual de acuerdo a las pruebas experimentales realizadas, obtiene mejores resultados que algunas estructuras de vecindad sencillas, [Cruz et al, 2010b].

Un método de optimización utilizado para resolver el problema de UPMP (problema de máquinas en paralelo no relacionadas), es el de recocido simulado, el cuál involucra el uso de una búsqueda local iterada con un procedimiento de movimiento e intercambio y un algoritmo de búsqueda tabú con vecindarios híbridos, el cual dio muy buenos resultados [Anagnostopoulos y Rabadi, 2002].

Para el problema del árbol de expansión mínima, se generó una estructura híbrida, con la finalidad de mejorar el desempeño del algoritmo, los resultados mostraron que la estructura híbrida fue la mejor en eficacia y se mostró competitiva en cuanto a la eficiencia [Martínez, 2011].

En el algoritmo de recocido simulado para maximizar la resistencia mecánica en aceros microaleados, se implementó un conjunto de estructuras de vecindad y una estructura híbrida, de acuerdo al análisis de desempeño realizado se encontró que la estructura híbrida de vecindad fue la que mejores resultados obtuvo [Juárez, 2011] .

En cuanto a lo referente a estructuras híbridas de vecindad aplicadas al problema de calendarización de trabajos en talleres de manufactura, no se encontraron hasta el momento referencias en la literatura de que se hayan implementado, descubriéndose así un nicho de oportunidad en este campo; es por este motivo que se propuso la implementación de una estructura híbrida de vecindad en este trabajo de investigación teniendo como base los resultados obtenidos por las diferentes estructuras híbridas de vecindad para los problemas de optimización anteriormente citados.

1.2 Objetivo de la investigación

Objetivo general

1. Diseñar y programar una estructura híbrida de vecindad para el problema de talleres de manufactura con el objetivo de poder usar esta estructura en la búsqueda de mejores soluciones.

Objetivo específico

2. Comparar la eficiencia y la eficacia de la estructura híbrida de vecindad propuesta contra las estructuras de vecindad ya utilizadas en la literatura.

1.3 Alcance de la investigación

1. Generación e implementación de una estructura de vecindad híbrida para evaluar su desempeño utilizando benchmarks del JSSP.

2. Los resultados obtenidos durante las pruebas con las estructuras de vecindad se compararán entre cada una de estas estructuras incluyendo la estructura híbrida.

1.4 Contribución de la Tesis

- En esta tesis se desarrolló una estructura de vecindad híbrida formada con las cuatro estructuras de vecindad que mejores resultados han reportado al ser aplicadas al JSSP. Estas estructuras de vecindad son: N1, N4, N5, N6.

- Se evaluó el desempeño de la estructura híbrida contra las otras estructuras N1, N4, N5, N6. A diferencia de lo que se encuentra en la literatura con respecto a otro tipo de problemas de optimización, la estructura híbrida fue competitiva pero no la mejor.

1.5 Organización de la Tesis

En el Capítulo 1 se da una introducción del problema a tratar, se da una explicación del estado del arte y se presentan los objetivos, alcances y la contribución de esta investigación y por último la organización general de la tesis.

En el Capítulo 2 se expone el problema del Job Shop Scheduling, la descripción del problema, la formulación matemática del mismo, y su representación por medio del grafo disyuntivo.

En el Capítulo 3 se describe la estructura de vecindad híbrida, se presenta la estrategia de sintonización, se explican las estructuras de vecindad a implementarse, se explica la estructura de vecindad híbrida propuesta, la implementación de la Lista Tabú, y se realiza un análisis de complejidad de las estructuras de vecindad.

En el Capítulo 4 se muestran los resultados experimentales de las estructuras de vecindad y de la estructura híbrida de vecindad, se muestra la descripción del equipo utilizado para realizar las pruebas. Se presentan las pruebas de las estructuras de vecindad propuestas para el problema de calendarización de trabajos en talleres de manufactura, el análisis de sensibilidad y por último se muestra el análisis de eficacia y eficiencia de las estructuras.

En el Capítulo 5 se dan las conclusiones finales del trabajo de investigación, así como los trabajos futuros.

Capítulo 2 El problema de los talleres de manufactura (JSSP)

En este capítulo se da una explicación del problema de calendarización de trabajos en un taller de manufactura o mejor conocido por sus siglas en inglés como JSSP (Job Shop Scheduling Problem), se muestra la formulación matemática del problema, la descripción del problema así como su representación mediante un grafo disyuntivo.

2.1 Descripción del Problema

El problema de calendarización de trabajos en un taller de manufactura o *Job Shop Scheduling Problem (JSSP)*, es considerado uno de los problemas más difíciles en las ciencias computacionales, ya que es clasificado dentro de la categoría de los *NP-Hard* [Papadimitriou and Steiglitz, 1998], [Lenstra and Rinnooy, 1979], y pertenece a los problemas considerados como intratables [Lawler *et al*, 1993]. El estudio de este problema es de suma importancia ya que es un problema práctico del área de la industria de la manufactura. El problema consiste de un conjunto de *m* máquinas y un conjunto de *n* trabajos, donde cada máquina lleva a cabo una operación de cada trabajo. Este problema intenta encontrar la secuencia óptima de operaciones en cada máquina que permita la minimización del tiempo máximo de terminación de los trabajos (*makespan*).

Cabe mencionar que el conjunto de soluciones para este problema está dado por $(n!)^m$, es decir que para un problema de 3 x 3 tendríamos 216 posibles soluciones, de las cuales, es importante comentar que no todas son soluciones factibles, debido a que algunas de ellas violan restricciones de precedencia [Cruz *et al*, 2009], el problema del Job Shop Scheduling con 3

máquinas y 3 trabajos (3x3) es considerado como *NP-Hard*, [Lenstra et al,1977], [Gonzalez and Sahni,1978], [Sotskov and Shakhlevich,1995].

Para llevar a cabo la calendarización de trabajos en un taller de manufactura, es necesario definir ciertas restricciones, mismas que aseguran la integridad de las operaciones y eficiencia del proceso. A continuación se enumeran las restricciones consideradas para este problema.

- Una máquina puede procesar solo un trabajo en un instante de tiempo.
- Un trabajo no debe procesarse en una misma máquina 2 veces.
- No existen restricciones entre operaciones de trabajos diferentes.
- Una vez que una operación es asignada, no puede ser interrumpida.
- Un trabajo consiste en *i* número de operaciones.
- Existe una restricción de precedencia entre las operaciones de un mismo trabajo, lo cual, al igual que los tiempos de procesamiento, son datos conocidos.
- Existe una restricción de capacidad de recursos, la cual es conocida como la secuencia de operaciones en una máquina, y son datos conocidos.
- Las operaciones de los trabajos tienen la misma prioridad.
- No se especifican fechas de liberación ni de vencimiento [Cruz *et al*, 2009], [Carlier *et al*, 1989].

2.2 Modelo Matemático

El JSSP intenta establecer una calendarización de máquinas en un taller de manufactura para llevar a cabo un conjunto de trabajos. Cada trabajo requiere un conjunto de operaciones a llevarse a cabo. La solución al problema es una secuencia óptima de los trabajos a ser ejecutados por cada máquina, respetando las restricciones de precedencia durante la ejecución de las operaciones en cada puesto de trabajo [Cruz *et al*, 2010].

Para el problema de calendarización de trabajos en un taller de manufactura (Job Shop Scheduling Problem) con el objetivo de minimizar el *Makespan* se usan los siguientes conjuntos y subconjuntos [Roy and Sussman, 1964]:

***** Un conjunto de trabajos $J = \{J_1, J_2, ..., J_n\}$, un conjunto de máquinas $=M \{M_1, M_2, ..., M_n\}$.

***** Un conjunto de operaciones $O = \{1, 2, 3, ...\}$.

***** Subconjuntos de operaciones para cada trabajo ($J_k \subset O$) y cada máquina ($M_k \subset O$).

$$Min\left[\frac{max}{j \in O}(s_j + p_j)\right] \qquad (2.1)$$

s. a.

$$\forall\, j \in O \qquad s_j \geq 0 \qquad (2.2)$$

$$\forall\, i, j \in O,\ (i \prec j) \in J_k \qquad s_i + p_i \leq s_j \qquad (2.3)$$

$$\forall\, i, j \in O,\ (i,j \in M_k) \qquad s_i + p_i \leq s_j \vee s_j + p_j \leq s_i \qquad (2.4)$$

$$\forall\, j \in O \qquad s_j + p_j \leq C_{max} \qquad (2.5)$$

En la Figura anterior se presenta el modelo matemático para el problema del *JSSP*, donde, la función objetivo es la de minimizar el *Makespan*, que definido en base a los tiempos de inicio queda como se especifica en la función *(2.1)*. En esta formulación, los índices *i* y *j* hacen referencia a las operaciones. El tiempo de procesamiento de cada operación se representa por *p*. El conjunto de restricciones *(2.2)* indican que el tiempo de inicio de cada operación debe ser positivo. El conjunto de restricciones *(2.3)* que afecta a todo par de operaciones adyacentes *i*, *j* de cada subconjunto J_k, define las restricciones de precedencia, el símbolo $\prec$ significa ”precede”. El conjunto de restricciones *(2.4)* define las restricciones de capacidad de recursos, estas restricciones afectan a cada par de operaciones *i*, *j* que pertenece a un subconjunto M_k, esto es, si la operación *i* se ejecuta antes que la operación *j*, entonces *i* debe terminar antes de que *j* inicie su ejecución o bien, si la operación *j* se ejecuta antes que la operación *i*, entonces *j* debe terminar antes de que *i* comience su ejecución.

El conjunto de restricciones *(2.5)* indica que el tiempo de término de cada operación *j* debe ser menor o igual que su *Makespan* (C_{max}) [Cruz, 2005].

El *Makespan* está dado por la longitud de la trayectoria ponderada más larga del inicio al final del grafo, se denomina ruta crítica y se compone de una secuencia de operaciones críticas, una secuencia de operaciones críticas consecutivas en la misma máquina se llama un bloque crítico [Zalzala *et al*, 1997].

2.3 Modelo del Grafo Disyuntivo

El modelo que ha sido más frecuentemente usado para aplicarlo al *JSSP* es el modelo del grafo disyuntivo debido a la facilidad de manipulación [Cruz *et al*, 2004].

El problema de calendarización de trabajos en talleres de manufactura (JSSP), se muestra en la Figura 2.1, es representado en la literatura mediante un grafo disyuntivo *G= (A, E, O)*, para una instancia de 3 x 3, este grafo disyuntivo está formado por tres conjuntos. El conjunto de operaciones *O*, está conformado por los nodos *G*, numerados del uno al nueve. El tiempo de procesamiento aparece junto a cada operación. Las operaciones de inicio y fin son ficticias, con tiempo de procesamiento igual a cero.

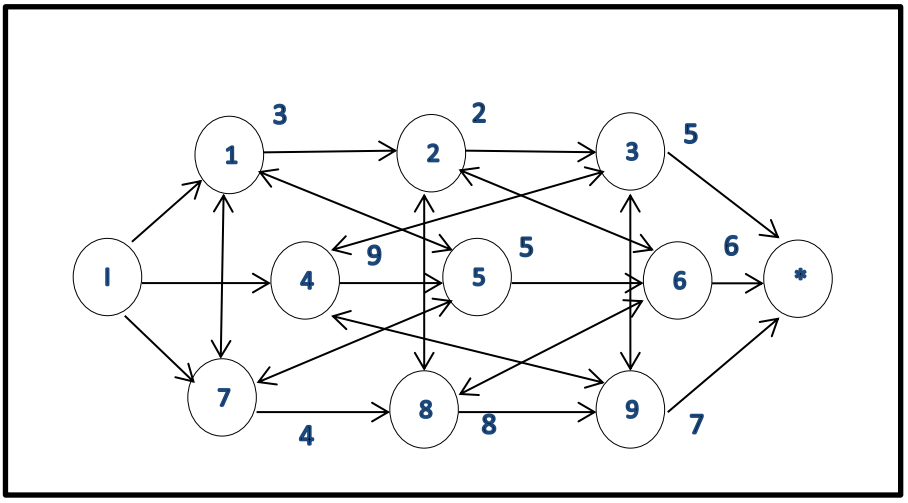

Figura 2.1. Grafo Disyuntivo del JSSP, 3 máquinas y 3 trabajos.

El conjunto *A* está compuesto por arcos conjuntivos, cada de estos arcos, une un par de operaciones que pertenecen al mismo trabajo. Las operaciones *1*, *2* y *3* están conectadas por uno de estos arcos y forman el trabajo *1*. Los trabajos *2* y *3* están conformados por las operaciones *4*, *5*, *6* y *7*, 8, *9* respectivamente. Cada arco de *A* representa una restricción de precedencia.

El conjunto *E* está compuesto por arcos disyuntivos, cada arco de *E* une un par de operaciones que pertenecen a la misma máquina. Se puede apreciar que las operaciones *1*, *5* y *7* son ejecutadas por la máquina uno. De la misma manera, las máquinas dos y tres ejecutan las operaciones *2*, *6*, *8* y *3*, *4*, *9* respectivamente.

Cada máquina forma un cliqué (un subconjunto de *E* completamente conectado). Cada arco de *E* representa una restricción de la capacidad de recursos entre un par de operaciones que pertenecen a una misma máquina. Este tipo de restricciones indican que una máquina no puede ejecutar más de una operación en el mismo intervalo de tiempo [Cruz *et al*, 2004].

Capítulo 3 Estructura de vecindad híbrida

3.1 Estructuras de vecindad

La vecindad de una solución se define como el conjunto de todas aquellas soluciones que pueden ser alcanzables a partir de una solución *s* por medio de una perturbación *σ* [Morales A., 2006], [Papadimitriou y Steiglitz, 1998], una perturbación puede ser un intercambio entre elementos que conforman la solución *s*.

$$N(s) = \{s' \in S : s \xrightarrow{\sigma} s'\} \qquad (3.1)$$

La relación (*3.1*) presenta la vecindad generada a partir de la solución *s*, donde N(*s*) representa la vecindad con respecto a *s*, *s* representa una solución tomada del espacio total de soluciones *S* y *s'* representa el vecino de *s* generado a partir de *σ* movimiento. Un movimiento puede ser una inserción, eliminación o intercambio de componentes en una solución [Cruz *et al*, 2008]. La parte esencial de una vecindad es su estructura y su tamaño [Ravindra *et al*, 2000]. Mientras el tamaño de la vecindad es grande, la calidad de las soluciones óptimas locales será mejor, así como la precisión de la solución final encontrada, por lo tanto, una heurística con una vecindad grande será más eficaz, de lo contrario, el tiempo requerido para realizar una iteración dentro de la vecindad será cada vez mayor de acuerdo con el tamaño de la vecindad [Cruz *et al*, 2010b]. Actualmente se han usado una diversidad de estructuras de vecindad aplicadas a búsquedas locales para atacar este problema.

En este proyecto se busca hacer un análisis de diversas estructuras de vecindad que han sido aplicadas al problema de calendarización de trabajos en un taller de manufactura (JSSP, Job Shop Scheduling Problem) para generar una estructura que combine las mejores características de las estructuras tomadas. Las estructuras de vecindad usadas en el JSSP se describen a continuación.

N1, introducida por *Van Laarhoven et al* (1992), en esta estructura se propone tomar aleatoriamente un par adyacente de operaciones miembros de la ruta crítica e intercambiarlas, con la restricción de que usen la misma máquina. Esta estructura ha sido probada en algoritmos como el de recocido simulado debido a su fácil implementación y a la calidad de resultados que se obtienen [Blazewicz *et al*, 1996].

N2, introducida por Matsuo *et al* (1988), intercambia dos operaciones de un bloque de la ruta crítica excepto aquellas que involucran dos operaciones internas. Ha sido probada en algoritmos como búsqueda local y algoritmos genéticos [Blazewicz *et al*, 1996].

N3, introducida por Dell'Amico *et al* (1993), intercambia las dos últimas operaciones de un bloque. Ha sido probada en Búsqueda Tabú [Blazewicz *et al*, 1996].

N4, propuesta por *Dell'Amico and Trubian*, cuyo movimiento indica tomar una operación miembro de la ruta crítica de un bloque y moverla ya sea al principio o al final del bloque. Ha sido probada en búsqueda tabú [Dell'Amico and Trubian, 1993].

N5, creada por *Nowicki and Smutnicki*, propone el uso de bloques como agrupamientos para operaciones que se efectúen en la misma máquina, ahí los movimientos son como sigue, en el primer bloque de operaciones, intercambia el último par de operaciones, en el bloque siguiente, intercambia el primer par de operaciones del bloque. Aplicada en algoritmos de búsqueda Tabú, es la que mejores resultados ha obtenido [Blazewicz *et al*, 1996].

N6, propuesta por *Balas and Vazacopoulos*, se toma una operación que no sea la primera ni la última del camino crítico y se inserta hacia adelante o hacia atrás, Ha sido aplicada en el algoritmo de Búsqueda local guiada, [Blazewicz *et al*, 1996].

Se muestra en la Tabla 3.1, el compendio de las diferentes estructuras de vecindad que se han aplicado al problema de calendarización de trabajos en un taller de manufactura, se describe el tipo de algoritmo en donde se han aplicado y el valor de la solución encontrada para una instancia de 10 x 10 [Carlier *et al*, 1989].

Tabla 3-1 Compendio de estructuras de vecindad para FT10

ESTRUCTURA/AUTORES	**TIPO DE INTERCAMBIO**	**TIPO DE ALGORITMO**	**COTA**
N1 Van Laarhoven *et al*, (1992)	Intercambio de un par de operaciones adyacentes	Recocido Simulado/ Búsqueda Tabú	937
N2 Matsuo et al, (1988)	Intercambio de dos operaciones de un bloque excepto las internas.	Búsqueda local/Algoritmos Genéticos	969
N3 Dell'Amico and Trubian, (1993)	Intercambio de las dos últimas operaciones de un bloque	Búsqueda Tabú	977
N4 Dell'Amico and Trubian (1993)	Intercambio de una operación interna por la operación que inicia el bloque o que termina el bloque.	Búsqueda Tabú	935
N5 Nowicki and Smutnicki, (1993)	En el 1er bloque se intercambian las dos últimas operaciones, en el 2do bloque se intercambian las dos primeras operaciones.	Búsqueda Tabú	930
N6 Balas and Vazacopoulos, (1995)	Inserción de una operación interna al principio o al final de su bloque	Búsqueda Local Guiada	930

3.2 Estructura de vecindad híbrida

Todo problema de optimización tiene un conjunto, ya sea finito o infinito de posibles soluciones, por lo que se requiere la utilización de técnicas que permitan la mejor exploración del espacio de soluciones, y como resultado obtener soluciones de buena calidad.

El funcionamiento de las estructuras de vecindad es iterativo, debido a que permiten la mejor explotación del espacio de soluciones, de modo que define la vecindad así como la forma de acceder a soluciones vecinas de una solución *s* mediante una operación denominada movimiento, donde el tipo de movimiento aplicado define la estructura y tamaño de una vecindad.

Un aspecto crítico del diseño de algunos algoritmos de optimización es la elección de una estructura de vecindad adecuada, es decir, elegir aquella estructura de vecindad que permita la mejor explotación del espacio de soluciones de acuerdo al algoritmo utilizado y al problema tratado.

Para desarrollar una estructura de vecindad híbrida, se realizó una revisión en la literatura, para buscar las estructuras de vecindad que contaran con baja complejidad computacional y que hayan obtenido buenos resultados en su aplicación. A continuación se mencionan las características (movimientos realizados) de cada una de las estructuras de vecindad utilizadas para desarrollar la estructura híbrida.

- **N1**, en la Figura 3.1 se muestran los movimientos que en esta estructura se proponen, toma aleatoriamente un par adyacente de operaciones miembros de la ruta crítica y las intercambia, con la restricción de que usen la misma máquina.

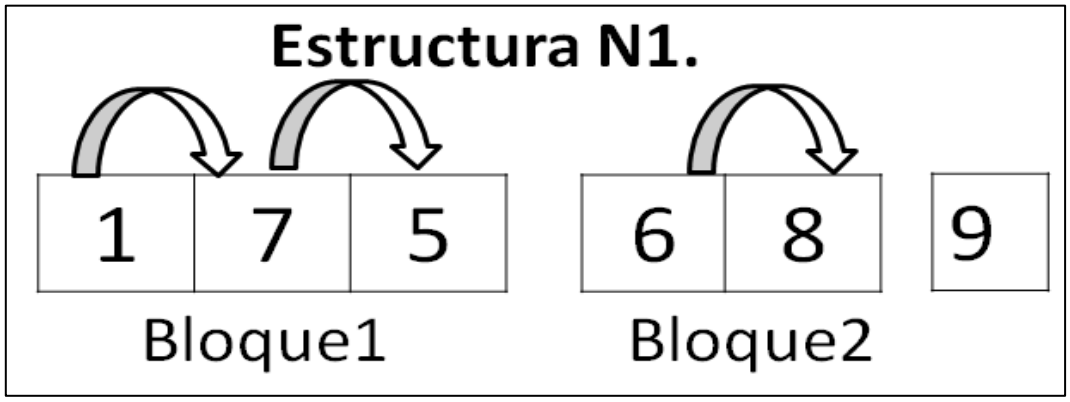

Figura 3.1. Movimientos realizados por la estructura de vecindad N1 [Blazewicz et al, 1996].

- **N4**, cuyo movimiento, mostrado en la Figura 3.2, indica tomar una operación miembro de la ruta crítica de un bloque y moverla ya sea al principio o al final del bloque.

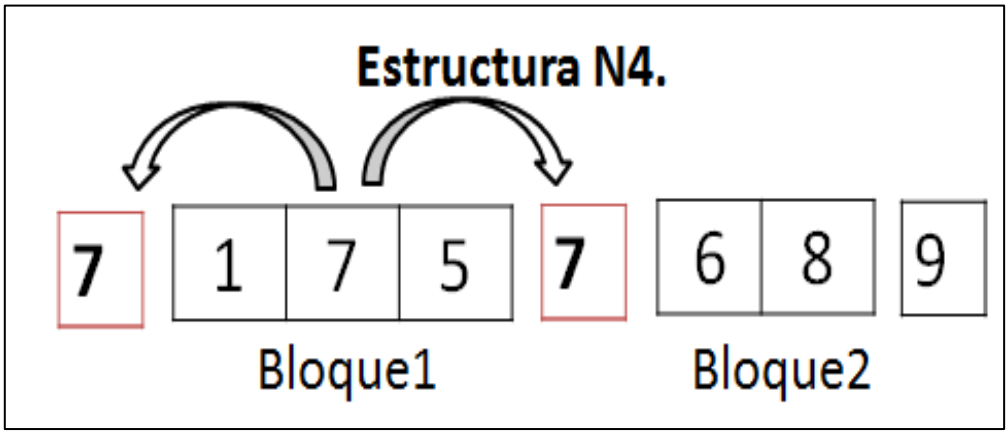

Figura 3.2. Movimientos realizados por la estructura N4 [Dell'Amico and Trubian, 1993]

- **N5**, los movimientos, mostrados en la Figura 3.3, son como sigue, en el primer bloque de operaciones, intercambia el último par de operaciones, en el bloque siguiente, intercambia el primer par de operaciones del bloque.

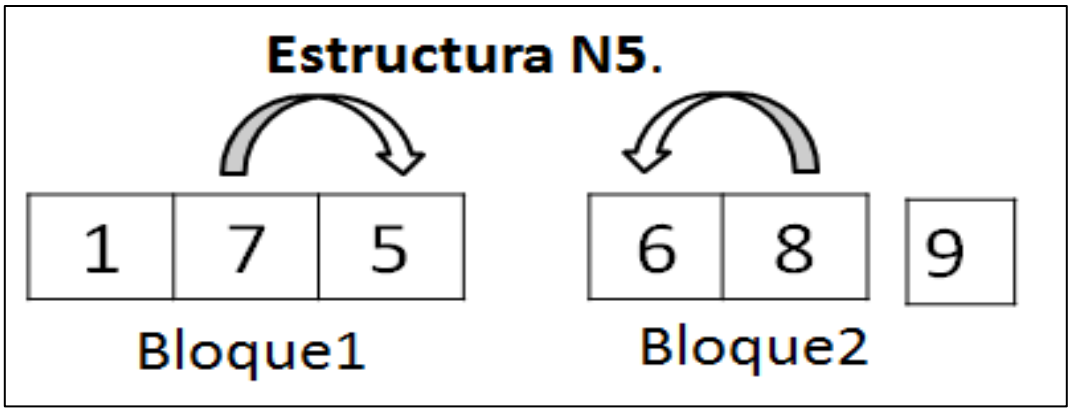

Figura 3.3. Movimientos realizados por la estructura N5 [Blazewicz et al, 1996]

- **N6**, los movimientos se muestran en la Figura 3.4, se toma una operación que no sea la primera ni la última del camino crítico y se intercambia hacia adelante, después se toma otra operación y se intercambia hacia atrás del camino crítico.

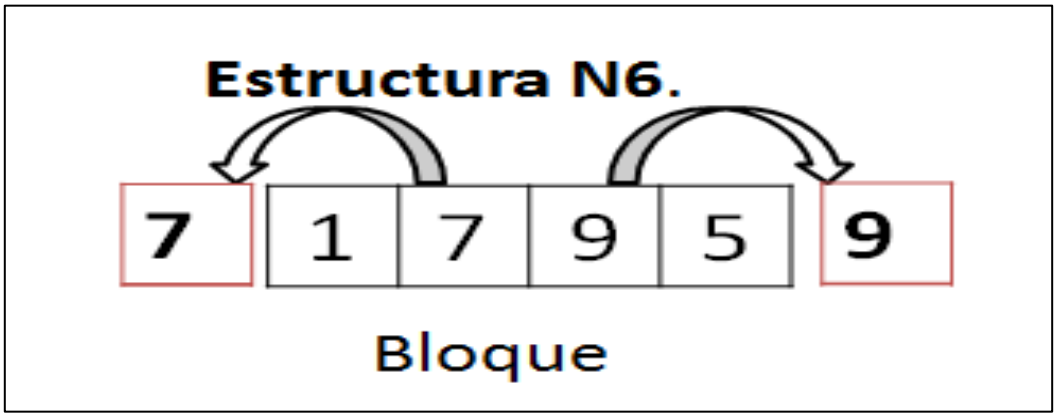

Figura 3.4. Movimientos realizados por la estructura N6 [Blazewicz et al, 1996]

Tomando en cuenta las ventajas de las estructuras de vecindades presentadas, así como su desempeño reportado en la literatura, se propuso el desarrollo de una estructura de vecindad híbrida que permitiera la incorporación de las ventajas de cada una de las cuatro estructuras, de modo que las perturbaciones realizadas por la estructura de vecindad híbrida interactúen de forma aleatoria.

El diagrama de flujo de la estructura de vecindad híbrida se muestra en la Figura 3.5, esta estructura se compone de las cuatro estructuras de vecindad (N1, N4, N5, N6). Donde, RC se refiere al método de la ruta crítica , BloquesRc, a los bloques de operaciones miembros de la ruta crítica, EVH, es la estructura de vecindad híbrida; el procedimiento de este algoritmo es partir de una solución inicial factible, se elige de forma aleatoria que tipo de movimiento se aplica, si la decisión es 0 se aplica la estructura que intercambia un par adyacente de operaciones (estructura N1), explicada en la figura 3.1, si la decisión es 1 entonces se aplica la estructura que indica tomar una operación miembro de la ruta crítica de un bloque que sea interna y moverla ya sea al principio o al final del bloque (estructura N4) explicada en la figura 3.2, si la decisión es 2 se aplica la estructura que intercambia las dos últimas operaciones de un bloque e intercambia las dos primeras operaciones de otro bloque (estructura N5) explicada en la figura 3.3, si es 3 la decisión, se aplica la estructura que toma una operación interna de un bloque y la inserta ya sea al inicio o al final del bloque (estructura N6) explicada en la figura 3.4.

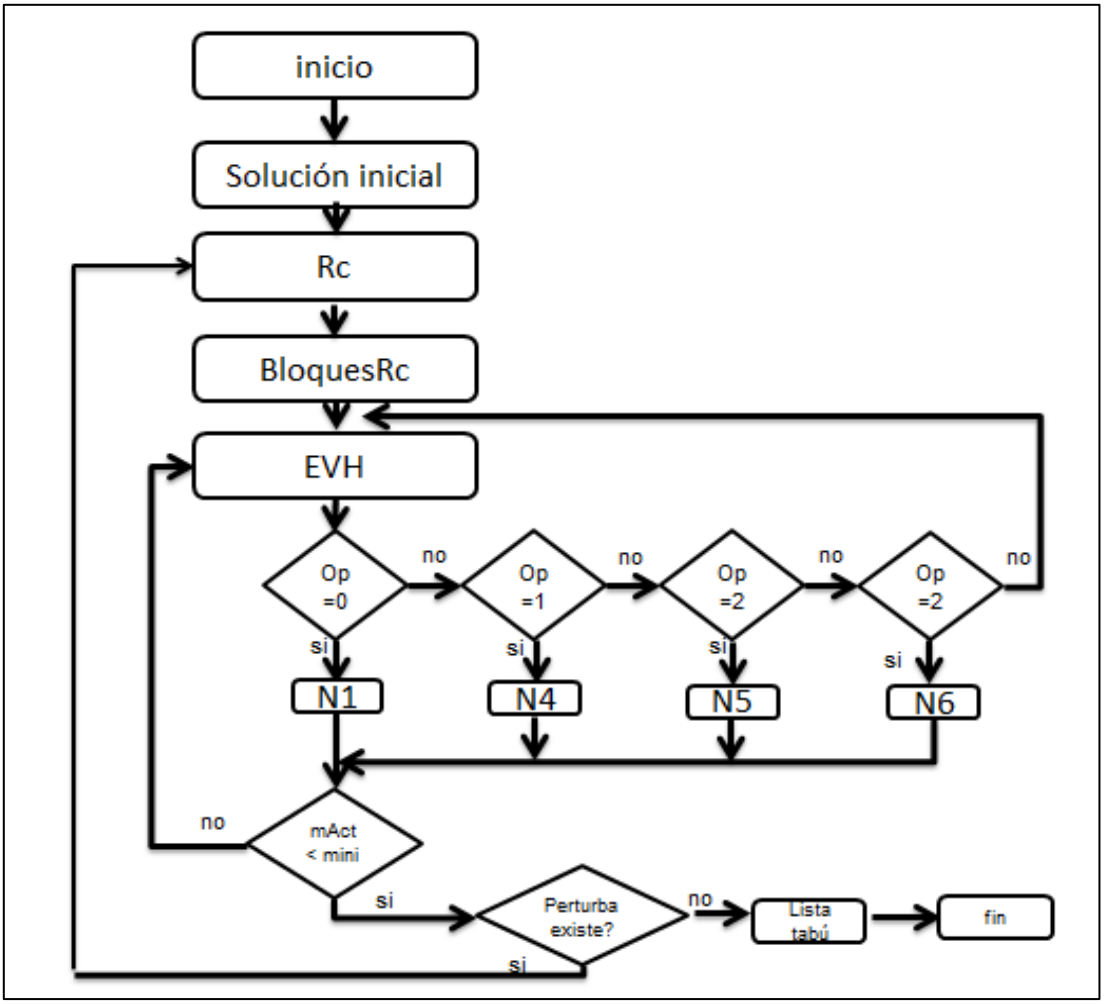

Figura 3.5. Diagrama de flujo de la estructura de vecindad híbrida

Para mejorar aún más la eficiencia y eficacia del algoritmo, aparte de la implementación de una estructura híbrida también se aplicó una lista tabú que a continuación se explica a detalle.

3.3 Lista tabú para estructuras de vecindad

La abundancia de problemas de optimización difíciles que se encuentran en situaciones prácticas, como las telecomunicaciones, la planificación logística, financieros, de transporte y la producción ha motivado el desarrollo de técnicas de optimización de gran alcance. La filosofía de la búsqueda tabú es obtener y explotar una colección de principales respuestas de problemas. Se puede decir que la búsqueda tabú se basa en los conceptos de selección que unen los campos de la inteligencia artificial y la optimización. La forma básica de búsqueda tabú se basa en las ideas propuestas por Fred Glover [Glover, 1990]. El método se basa en los procedimientos diseñados para cruzar los límites de factibilidad u optimalidad local, las cuales fueron tratadas generalmente como barreras. La búsqueda Tabú es una metaheurística que guía a un procedimiento heurístico de búsqueda local para explorar el espacio de soluciones más allá de la optimalidad.

El procedimiento local es una búsqueda que utiliza una operación llamada perturbar para definir un vecindario de cualquier solución dada. Uno de los componentes principales de Búsqueda Tabú es el uso de memoria adaptativa, que crea un comportamiento de búsqueda más flexible. Estrategias basadas en memoria, son el sello distintivo de los enfoques de búsqueda tabú [Glover *et al*, 1997].

3.4 Implementación de la lista Tabú

La búsqueda tabú, TS (por sus siglas en inglés, Tabú Search) es una de las técnicas más utilizadas en algoritmos no determinísticos, proviene de la inteligencia artificial y está basada en una memoria adaptativa que le permite explorar un espacio de soluciones de manera eficiente a través de la estructura de vecindad [Glover, 1989]. TS considera dos tipos de memoria a largo y corto plazo.

En este trabajo de investigación se emplea una lista tabú para mejorar la búsqueda de soluciones. La lista tabú debe ser dinámica después de un cierto número de iteraciones, debido a que la búsqueda se encuentra en una región distinta y las soluciones antiguas pueden liberarse del statús tabú [Glover and Laguna, 1997].

Esta lista tabú ayuda a que las soluciones no queden atrapadas en óptimos locales, lo que permite mejorar aún más la eficacia y la eficiencia del algoritmo. La lista tabú utilizada contiene los movimientos que se realizan entre operaciones para encontrar una solución vecina, esta lista tiene la función de evitar que una solución sea visitada nuevamente en base a una repetición del movimiento.

A continuación se presenta el procedimiento del uso de la lista tabú para el problema de calendarización de trabajos en talleres de manufactura.

Paso 1:

Se parte de una solución inicial y una lista tabú vacía, cabe mencionar que los tamaños de la lista tabú son de 5 a 9 registros, tomados de [Pinedo, 2008], [Martínez, 2011]. En la Figura 3.6, se presenta un ejemplo de una lista con un tamaño de 9 x 2, donde 9 representa el número máximo de operaciones perturbadas.

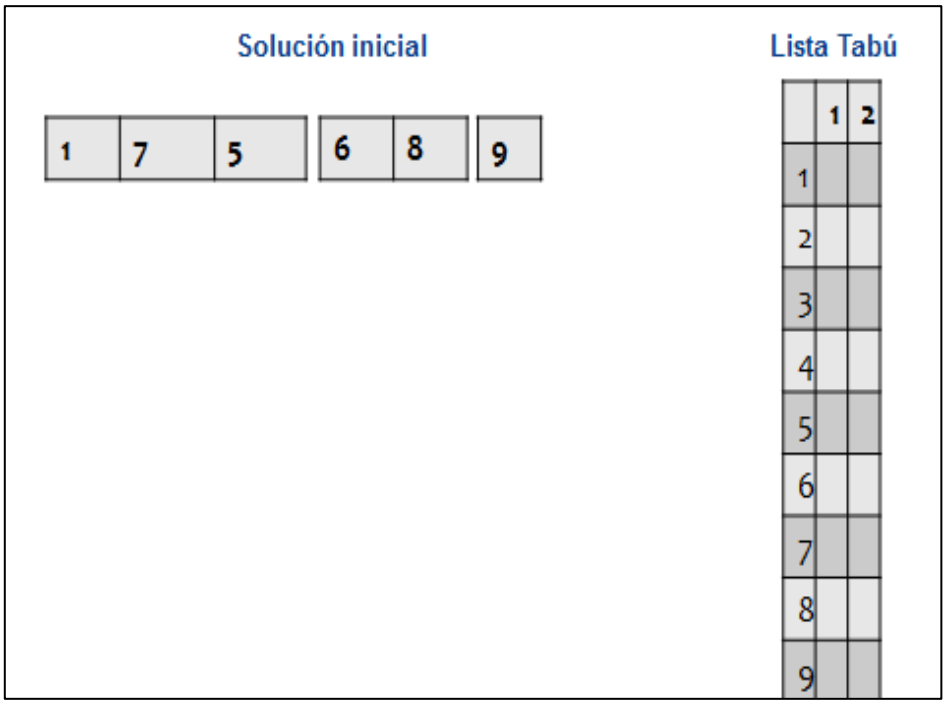

Figura 3.6. Solución inicial y lista tabú vacía

Paso 2:

Se elige de forma aleatoria un tipo de estructura de vecindad (N1, N4, N5, N6), por ejemplo si se elige la estructura N1, que conlleva el intercambio de dos operaciones adyacentes, se guarda el inverso de este par de operaciones si y solo si mejora la solución inicial del problema de calendarización de tareas para talleres de manufactura. Su funcionamiento se muestra en la figura 3.7.

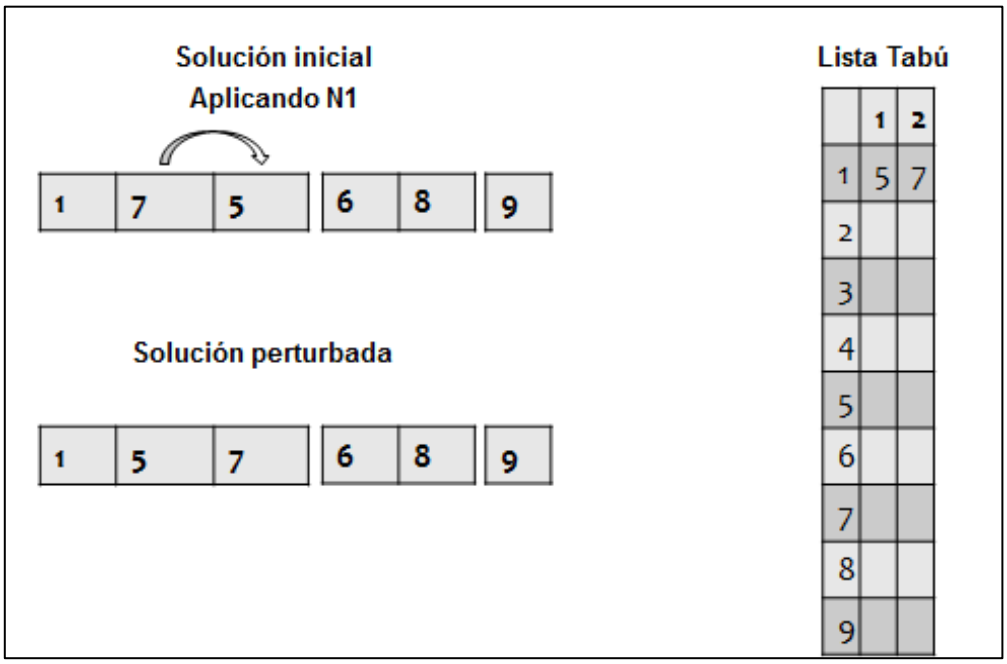

Figura 3.7. Solución inicial, estructura N1 y lista tabú

Paso 3:

Se vuelve a elegir aleatoriamente una estructura de vecindad, en este caso la N5, involucra el uso de dos bloques. En el primer bloque se intercambian las dos últimas operaciones, en el siguiente bloque se intercambian las dos primeras operaciones. Se guarda el inverso de este doble intercambio de operaciones si y solo si mejora la solución inicial del problema de calendarización de tareas para talleres de manufactura. En este caso en específico el intercambio del primer bloque ya estaba almacenado en la lista tabú (marcado con la flecha verde), por lo que este intercambio ya no se lleva a cabo y solo se realiza el que no está en la lista tabú. Su funcionamiento se muestra en la figura 3.8.

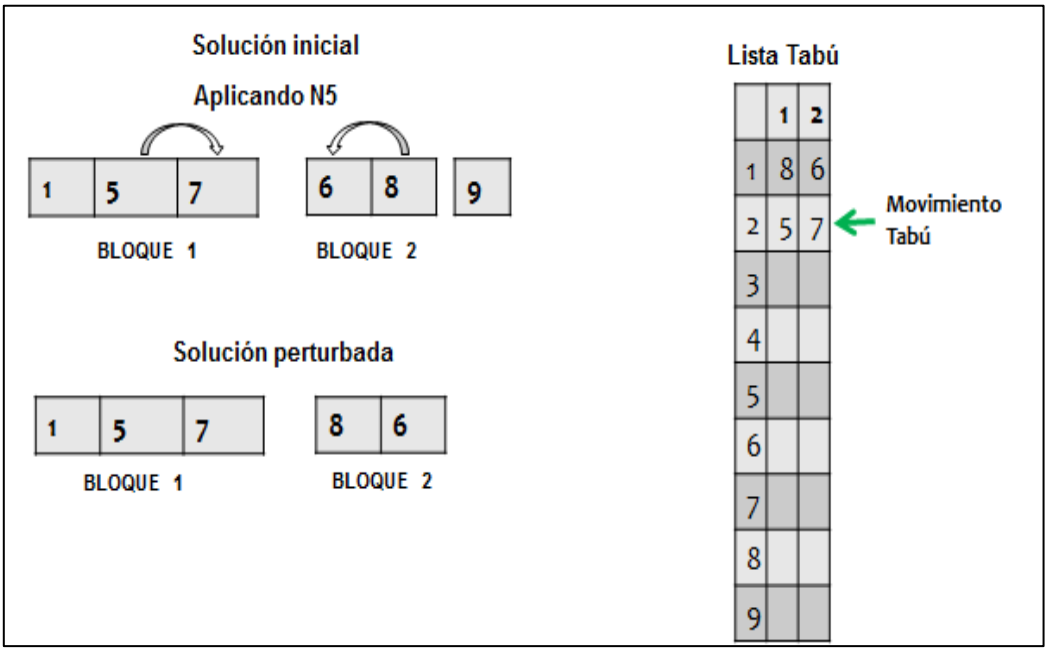

Figura 3.8. Solución inicial, estructura N5 y lista tabú

Paso 4:

Se continúa eligiendo movimientos de forma aleatoria, se verifican en toda la lista tabú que no existan dichos movimientos y se agregan, si la lista ya está llena el último movimiento se libera de la lista y así sucesivamente.

Cuando demasiados pasos transcurren sin mejorar el óptimo actual, se cambia la estructura de vecindad [Brandimarte, 1993].

3.5 Metodología de sintonización

Para la sintonización de los parámetros de control de un algoritmo, es necesario llevar a cabo un análisis de sensibilidad. El análisis de sensibilidad es un componente importante en la construcción de modelos matemáticos, computacionales y de simulación [Pannell, 2009]. El análisis de sensibilidad puede ser utilizado con simulación numérica para estudiar el comportamiento de un algoritmo. La finalidad del análisis de sensibilidad, es encontrar la mejor sintonización de los parámetros de control del algoritmo, de manera que este tenga el mejor funcionamiento posible en cuanto a la eficiencia y eficacia.

A continuación se explica la metodología de sintonización tomada del trabajo de investigación de [Cruz, 2005] y [Martínez, 2010], quienes proponen esta metodología para encontrar la proporción adecuada en los valores correspondientes a los parámetros de control, tomando en cuenta el problema y el método implementado para obtener la solución.

1. *Selección de los Parámetros de Control.*

El primer paso es seleccionar los parámetros de control del algoritmo, es necesario llevar a cabo una revisión en la literatura en donde se haya implementado dicho algoritmo, de esta forma es posible hacer un análisis de los parámetros de control que utiliza el algoritmo tomado en investigaciones anteriores.

2. *Establecer rangos de evaluación*

Para establecer los rangos que van a ser utilizados para realizar el análisis de sensibilidad, es necesario ya haber identificado los parámetros de control, para establecer los rangos a cada uno de los parámetros, se hizo una revisión en la literatura para identificar y analizar los valores utilizados por varios autores en el algoritmo propuesto, de esta forma es más fácil definir un rango para cada uno de los parámetros de entrada al algoritmo. Es muy importante mencionar que si se lleva a cabo una adecuada sintonización de los parámetros de control, esto influye dentro del algoritmo para obtener buenas soluciones.

3. *Pruebas para rangos de evaluación*

Una vez establecido los rangos para cada parámetro de control mencionados anteriormente, se calcula una cantidad de muestra que permitan evaluar el comportamiento del algoritmo. Se realizan las pruebas experimentales, ejecutando 30 pruebas por cada una de las muestras calculadas, para una adecuada sintonización de los parámetros de control, es necesario realizar las pruebas de los valores correspondientes a una de las variables, pero manteniendo fijos los demás, hasta encontrar el valor que mejore la calidad

de las soluciones. En cuanto se obtenga el mejor valor de dicha variable, se fija el valor y se comienza con la variación de otra variable, llevando a cabo el mismo proceso, hasta obtener el conjunto de valores que permita a las estructuras de vecindad mejorar en eficacia y eficiencia.

4. Sintonización de parámetros

Una vez terminadas las pruebas experimentales, para obtener los valores para cada uno de los parámetros de control definidos para todas y cada una de las muestras, estos valores son definidos como los *valores de sintonización.*

La sintonización de parámetros se lleva a cabo con el objetivo de identificar los valores correspondientes a los parámetros de control, los cuales permitan obtener una mejora en el desempeño de las estructuras de vecindad tanto en eficacia como en eficiencia.

De acuerdo al análisis realizado tanto al método aplicado como al problema tratado, las variables utilizadas para llevar a cabo el análisis de sensibilidad de las estructuras de vecindad aplicadas al problema de calendarización de trabajos en un taller de manufactura (JSSP), son las siguientes:

- Tamaño de la lista Tabú ($\boldsymbol{T_L}$), que representa el tamaño de la lista tabú.
- Tamaño de la vecindad ($\boldsymbol{K}$), representa al tamaño de la vecindad.

En el capítulo 4 se detallará la explicación de los parámetros arriba mencionados.

3.6 Análisis de la Complejidad de las estructuras de vecindad

Cuando se hallan definido los parámetros de control que hacen que las estructuras de vecindad funcionen correctamente, es necesario realizar un estudio que permita conocer su comportamiento y así poder medir su rendimiento, centrándose principalmente en su simplicidad y el uso eficiente de los recursos.

La complejidad computacional de un algoritmo se puede clasificar de acuerdo a la dificultad de resolverlo en función de los siguientes parámetros:

***Espacio**: c*antidad de memoria requerida para el almacenamiento de los datos durante la ejecución del algoritmo.

***Tiempo**:* duración de la ejecución del algoritmo.

Los parámetros mencionados representan el costo requerido por las estructuras de vecindad según el tipo de problema que se está tratando, para encontrar una solución.

El tiempo de ejecución de un algoritmo o complejidad temporal T(n), donde n es el tamaño de la entrada, está en función de los parámetros: Datos de entrada, velocidad del procesador y complejidad del algoritmo. La complejidad temporal representa el número de instrucciones simples (asignaciones, comparaciones, operaciones aritméticas, etcétera) que serán ejecutadas por el algoritmo. Por lo regular se considera la complejidad del algoritmo en el peor de los casos, aunque también es importante conocer la complejidad en el mejor y el caso promedio.

Este análisis representa los requerimientos de memoria y tiempo respectivamente, requeridos por un algoritmo para encontrar una solución a un problema dado.

Este análisis de complejidad permite determinar la eficiencia de un algoritmo y por lo tanto, compararlo con otros algoritmos que resuelvan un mismo problema.

De acuerdo a esto, se tiene que *T(n)* representa el número de instrucciones simples (asignaciones, comparaciones, operaciones aritméticas, etc.) que son ejecutadas en el algoritmo.

A reserva de que se indique lo contrario, siempre se considera la complejidad del algoritmo en el peor de los casos, aunque también pueden ser analizados el mejor caso y el caso promedio.

Para calcular la complejidad de las estructuras de vecindad es necesario realizar un análisis del número de instrucciones requeridas por las estructuras para encontrar una solución al problema tratado. De acuerdo a esto, se muestra la ecuación correspondiente a la función temporal de las estructuras (Ecuación 1).

$$\mathrm{n}^3+n^2(nxm)^3\ln n \qquad (1)$$

Donde *n* representa el número de trabajos de la instancia, *m* al número de máquinas.

De acuerdo a esto, se puede concluir que la complejidad de las estructuras de vecindad y de la estructura híbrida de vecindad para el problema de calendarización de trabajos en un taller de manufactura, tiene una complejidad en el peor de los casos de $O(n^3)$.

El análisis de complejidad completo se presenta en el apéndice B.

Capítulo 4 Resultados experimentales

En este capítulo se presentan los resultados obtenidos en las pruebas experimentales realizadas a las estructuras de vecindad y a la estructura de vecindad híbrida. Se aplica también un análisis de sensibilidad con el fin de mejorar el desempeño de las estructuras. Estos resultados son comparados con los resultados de la estructura híbrida de vecindad para mostrar la eficiencia y eficacia de la estructura propuesta.

4.1 Descripción del equipo utilizado

Para llevar a cabo las pruebas experimentales correspondientes a las estructuras de vecindad y a la estructura híbrida, se utilizó una computadora portátil con las siguientes características:

- **Procesador:** AMD A6-3420M APU Radeon (tm) HD Graphics 1.50 GHz, Quad Core
- **Memoria (RAM):** 4 GB
- **Sistema operativo:** Windows 7 Home Premium
- **Compilador:** Dev-C + + 4.9.9.2

4.2 Análisis de sensibilidad

Para llevar a cabo el análisis de sensibilidad de los parámetros de control de las estructuras de vecindad, aplicadas al problema de calendarización de trabajos en un taller de manufactura (JSSP), se propuso una metodología, la cual fue explicada en el Capítulo 3. A continuación se aplica la metodología a las estructuras de vecindad.

1. *Selección de los Parámetros de Control*

De acuerdo al análisis realizado tanto al método aplicado como al problema tratado, las variables utilizadas para llevar a cabo el análisis de sensibilidad de las estructuras de vecindad al problema de calendarización de trabajos en un taller de manufactura (JSSP), son las siguientes:

- Tamaño de la lista Tabú (T_L)
- Tamaño de la vecindad (K), donde m es la cantidad de máquinas disponibles y n el número de trabajos a realizar.
  Para la estructura de vecindad N1 el tamaño de la vecindad se calcula con la siguiente formula: $m\ (n-1)$, para la estructura de vecindad N4 es $2m\ (n-1)$, para N5 es $m(n-1)$ y finalmente para N6 es $2m\ (n-1)$.

2. *Establecer rangos de evaluación*

En cuanto se hayan definido los parámetros de control, es necesario determinar los rangos que serán utilizados para el análisis de sensibilidad a cada uno de los parámetros mencionados anteriormente.

Se hizo una revisión en la literatura donde se la lista tabú [Pinedo, 2008]; [Taillard,1994]; [Geyik, 2004]; [Nakano, 1991] y de acuerdo a los valores utilizados en estos trabajos se determinó un rango para los parámetros de $\boldsymbol{T_L}$ y $\boldsymbol{K}$.

Para el caso del parámetro $\boldsymbol{T_L}$, que se refiere al tamaño de la lista tabú, se contempla un rango de evaluación (Límite inferior=8, Límite superior=14), que se refiere al número de pares de operaciones que podrá almacenar; para el parámetro $\boldsymbol{K}$, que se refiere al número de veces que se repite el tamaño de la vecindad en la búsqueda local, de 1 a 5 veces, la información se muestra en la Tabla 4-1.

Tabla 4-1 Rangos utilizados para realizar el análisis de sensibilidad

Parámetro de control	Límite inferior	Límite superior
T_L	8	14
K	1	5

3. *Pruebas sobre rangos de evaluación*

Para llevar a cabo el análisis de sensibilidad a los parámetros de control mencionados anteriormente, se realizan los pasos siguientes:

En principio, para cada uno de los rangos mostrados en la Tabla 4-1, se calcularon 30 pruebas, utilizando la instancia de prueba más grande que es de 20 trabajos y 20 máquinas (YN1), del conjunto de pruebas de Yamada y Nakano [Yamada and Nakano, 1992]; este número de pruebas se tomó en base al análisis estadístico donde se dice que una muestra debe tener mínimo 30 pruebas para cada tamaño de instancia.

Se probaron las repeticiones del tamaño de vecindad, parámetro ***K***, de acuerdo a los rangos establecidos (de uno a cinco veces el tamaño de la vecindad).

Se muestra el análisis de sensibilidad en la Tabla 4-2, que se llevó a cabo realizando 30 ejecuciones correspondientes a cada estructura de vecindad para el parámetro ***K***, que es el tamaño de la vecindad, de acuerdo al incremento y a la formula respectiva para cada estructura de vecindad, manteniendo constante el valor del parámetro restante de T_L por cada serie de pruebas.

Tabla 4-2 Tabla para sintonizar el valor del parámetro *K*

Estructura de vecindad	Mejor Solución	Peor Solución
N1	**1506**	2551
N4	1651	2814
N5	1589	2632
N6	1708	**2818**

En la Tabla 4-3 se observan los valores sintonizados para el parámetro *K*. El tamaño de **K** que mejores resultados presento en cuanto el makespan fue el de **1506**, correspondiente a la estructura de vecindad N1, probado para la instancia **YN1** de 20 máquinas, 20 trabajos y cuyo incremento de vecindad corresponde a **3** veces el tamaño de la vecindad.

Tabla 4-3 Valores sintonizados para *K*

Estructura de vecindad	Tamaño de vecindad
N1	3
N4	3
N5	3
N6	3

En cuanto se termina de realizar la evaluación de los resultados obtenidos para cada una de las pruebas, se fija el valor de ***K***, que haya obtenido los mejores resultados de acuerdo a la función objetivo del problema.

El mismo proceso se lleva a cabo para el fijar el valor del parámetro $\boldsymbol{T_L}$, correspondiente a la lista tabú, que permita obtener los mejores resultados para el algoritmo. Se puede observar, en la Tabla 4-4, las mejores soluciones obtenidas para el parámetro $\boldsymbol{T_L}$ con los rangos de valor de 8-14, la mejor solución obtenida fue de **1378** con un tamaño de lista tabú de **14** registros, obtenida por la estructura de vecindad N1.

Tabla 4-4 Valores para sintonizar parámetro de T_L

Tamaño lista tabú	Mejor solución N1	Mejor solución N4	Mejor solución N5	Mejor solución N6
8	1506	**1651**	**1589**	1708
9	1569	1959	1733	1812
10	1501	1841	1921	1725
11	1470	1989	1590	**1602**
12	1571	1791	1643	1721
13	1671	1876	1922	1970
14	**1378**	1861	1960	1882

De acuerdo a esto, se obtuvieron los siguientes valores Sintonizados para los parámetros ***K*** (tamaño de la vecindad) y $\boldsymbol{T_L}$ (tamaño de la lista tabú). Los parámetros sintonizados se describen en la Tabla 4-5, de acuerdo al análisis de sensibilidad, donde quedan con los siguientes valores, para ***K***=3 (3 veces el tamaño de la vecindad), y para el tamaño de la lista tabú $\boldsymbol{T_L}$ de **14** registros.

Tabla 4-5 Valores sintonizados de los parámetros de control

Parámetro de control	Valor sintonizado
K	3
T_L	14

4. Sintonización de Parámetros

Los valores obtenidos para cada uno de los parámetros de control, al término de la evaluación de todas y cada uno de los incrementos (Tabla 4-5), dan como resultado la *sintonización de los parámetros de control*.

La sintonización de los parámetros de control se lleva a cabo con el objeto de identificar los valores correspondientes a los variables de control, lo cual permite obtener una mejora en el desempeño de las estructuras.

4.3 Pruebas de estructuras de vecindad

Las pruebas experimentales fueron realizadas para cada estructura de vecindad. Y de acuerdo a los resultados obtenidos se compara la eficiencia y la eficacia de la estructura híbrida. Las instancias de prueba utilizadas para el problema de calendarización de tareas para un taller de manufactura fueron 7, los benchmarks son: FT10 [Fisher and Thompson, 1963], LA39 Y LA40 [Lawrence,1984], YN1, YN2, YN3, YN4 [Yamada and Nakano,1992], cada estructura de vecindad fue ejecutada 30 veces.

Se presentan los resultados obtenidos para diversos benchmarks para el problema de calendarización de trabajos en talleres de manufactura, con las cinco estructuras de vecindad presentadas en el capítulo 3, de las cuales se hizo una evaluación de la mejor solución, peor solución, la función de makespan promedio, así como su desviación estándar σ para las 30 pruebas realizadas.

Para el problema de calendarización de tareas para talleres de manufactura (JSSP) se observa, en la tabla 4-6, que la estructura con peor eficacia es la N5, y en cuanto a la mejor solución encontrada de las 30 pruebas es la N5 también. En segundo lugar en cuanto a la mejor solución encontrada esta la estructura de vecindad híbrida, en tercer lugar la estructura de vecindad N1.

La estructura híbrida es la mejor en cuanto a la peor solución de mejor calidad y al promedio, mientras que la estructura N6 es la mejor en cuanto a la desviación estándar.

Tabla 4-6 Resultados para el benchmark FT10

Tipo de estructura	Mejor Solución	Peor Solución	Promedio	σ
N1	1376	2772	2404.3	284.94
N4	2046	2841	2578.7	248.05
N5	**1184**	3100	2393.7	390.26
N6	1908	2789	2529.8	**230.41**
HÍBRIDA	1279	**2770**	**2310.8**	354.85

Los resultados obtenidos para el benchmark LA39 para el problema de calendarización de tareas para talleres de manufactura (JSSP) se observan en la tabla 4-7, la estructura con peor eficacia es la híbrida, y en cuanto a la mejor solución encontrada de las 30 pruebas es la N5, mientras que la estructura N1 es la mejor en cuanto a la peor solución de mejor calidad, la estructura N6 es la mejor en cuanto a la desviación estándar.

Tabla 4-7 Resultados para el benchmark LA39

Tipo de estructura	Mejor Solución	Peor Solución	Promedio	σ
N1	2084	**3864**	3351.4	349.17
N4	2682	4058	3463.43	346.42
N5	**1742**	3933	**3299.93**	437.72
N6	3052	4011	3464.46	**309.14**
HÍBRIDA	2026	4495	3403.13	450.58

A continuación, en la Tabla 4-8, se muestran los resultados obtenidos para el benchmark LA40, para el problema de calendarización de tareas para talleres de manufactura (JSSP), se tiene que la estructura con peor eficacia es la N5, y en cuanto a la mejor solución encontrada de las 30 pruebas es la N1, mientras que la estructura N1 es la mejor en cuanto a la peor solución de mejor calidad, la estructura N4 es la mejor en cuanto a la desviación estándar y la estructura híbrida es la mejor en cuanto al promedio.

Tabla 4-8 Resultados para el benchmark LA40

Tipo de estructura	**Mejor Solución**	**Peor Solución**	**Promedio**	**σ**
N1	**1656**	**3641**	3294.5	374.52
N4	2647	3729	3249.1	**258.34**
N5	1841	4245	3269	397.62
N6	2559	3890	3450.4	316.82
HÍBRIDA	1823	3782	**3245.5**	422.19

Los resultados obtenidos para el problema de calendarización de tareas en talleres de manufactura, utilizando el benchmark YN1, con las cinco estructuras de vecindad presentadas en el capítulo 3, de las cuales se hizo una evaluación de la mejor solución, peor solución, la función de makespan promedio, así como su desviación estándar σ para las 30 pruebas realizadas, se presentan en la Tabla 4-9. Se observa que la estructura con peor eficacia es la N5, en cuanto a la mejor solución encontrada de las 30 pruebas es la N5, mientras la estructura N1 es la mejor en cuanto a la peor solución de mejor calidad, la estructura N4 es la mejor en cuanto a la desviación estándar y la estructura híbrida es la mejor en cuanto al promedio.

Tabla 4-9 Resultados para el benchmark YN1

Tipo de estructura	Mejor Solución	Peor Solución	Promedio	σ
N1	1735	**3003**	2555.1	292.14
N4	2123	3054	2583.8	**214.84**
N5	**1348**	3174	2561.1	334.91
N6	1983	3071	2604.6	254.50
HÍBRIDA	1504	3159	**2519.1**	320.69

En la Tabla 4-10, se presentan los resultados obtenidos para el problema de calendarización de tareas en talleres de manufactura utilizando el benchmark YN2, con las cinco estructuras de vecindad presentadas en el capítulo 3, de las cuales se hizo una evaluación de la mejor solución, peor solución, la función de makespan promedio, así como su desviación estándar σ para las 30 pruebas realizadas; se observa que la estructura con peor eficacia y en cuanto a la mejor solución encontrada de las 30 pruebas es la híbrida, por otro lado, la estructura N5 es la mejor en cuanto a la peor solución de mejor calidad y al promedio, mientras la estructura N4 es la mejor en cuanto a la desviación estándar.

Tabla 4-10 Resultados para el benchmark YN2

Tipo de estructura	Mejor Solución	Peor Solución	Promedio	σ
N1	1666	3023	2636.2	280.50
N4	2301	2929	2656	**179.69**
N5	1484	**2843**	**2492.3**	271.57
N6	2290	3085	2693.2	213.65
HÍBRIDA	**1458**	3129	2549.2	301.16

Los resultados obtenidos que se muestran en la Tabla 4-11, para el problema de calendarización de tareas en talleres de manufactura, utilizando el benchmark YN3 con las cinco estructuras de vecindad presentadas en el capítulo 3, de las cuales se hizo una evaluación de la mejor solución, peor solución, la función de makespan promedio, así como su desviación estándar σ para las 30 pruebas realizadas, arrojan la siguiente información: que la estructura con peor eficacia y en cuanto a la mejor solución encontrada de las 30 pruebas es la híbrida así como también es la mejor en cuanto a la peor solución de mejor calidad, que la estructura N1 es la mejor en cuanto al promedio y la N6 es la mejor en cuanto a la desviación estándar.

Tabla 4-11 Resultados para el benchmark YN3

Tipo de estructura	**Mejor Solución**	**Peor Solución**	**Promedio**	**σ**
N1	1579	2984	**2547.2**	265.78
N4	2349	3024	2641.9	187.17
N5	1546	2927	2586.9	240.29
N6	2308	2940	2612.4	**175.41**
HÍBRIDA	**1425**	**2911**	2565.2	283.74

A continuación en la Tabla 4-12, se presentan los resultados obtenidos utilizando el benchmark YN4 para el problema de calendarización de tareas en talleres de manufactura, con las cinco estructuras de vecindad presentadas en el capítulo 3, de las cuales se hizo una evaluación de la mejor solución, peor solución, la función de makespan promedio, así como su desviación estándar σ para las 30 pruebas realizadas. Se observa que la estructura con peor eficacia es la N4, en cuanto a la mejor solución obtenida y al promedio es la N5, la estructura N6 es la mejor en cuanto a la peor solución de mejor calidad y en la desviación estándar.

Tabla 4-12 Resultados para el benchmark YN4

Tipo de estructura	**Mejor Solución**	**Peor Solución**	**Promedio**	**σ**
N1	1800	3304	2697.5	290.81
N4	2528	3322	2863.7	224.58
N5	**1537**	3202	**2679.1**	322.34
N6	2478	**3184**	2768.7	**170.48**
HÍBRIDA	1632	3192	2699.8	314.85

Los resultados de las 30 ejecuciones obtenidas para el benchmark FT10 del problema de calendarización de trabajos en los talleres de manufactura para cada estructura de vecindad se muestran en la Figura 4.1. Tomando una cota para el benchmark FT10 en la función objetivo, que es de 930, se observa que las 30 soluciones están por encima de la cota superior, se puede ver claramente que la estructura que mejor desempeño tiene es la estructura de vecindad N5, en segundo lugar la estructura híbrida de vecindad, en tercer lugar la estructura N1, por mencionar a las 3 mejores.

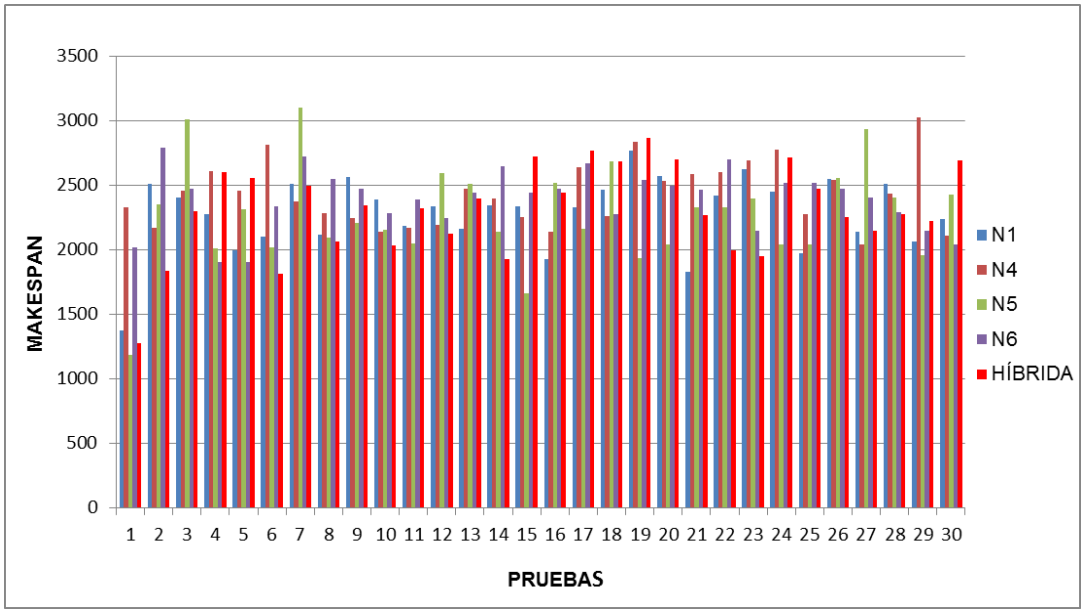

Figura 4.1. Resultados de las estructuras de vecindad para FT10

A continuación, en la Figura 4.2, se muestran los resultados obtenidos de las 30 ejecuciones obtenidas para el benchmark LA39 del problema de calendarización de trabajos en los talleres de manufactura para cada estructura de vecindad. En la gráfica de los resultados de las 30 pruebas para el benchmark LA39, que consta de 15 trabajos x 15 máquinas, con una cota en la función objetivo de 1676; en la ejecución 1, se puede observar que la estructura de vecindad N5, aunque está por encima de la cota superior, tiene mejor eficacia que la estructura híbrida, la cual se sitúa en segundo lugar, seguida por la estructura de vecindad N1 en tercer lugar, en cuarto lugar la estructura N4 y en último lugar la estructura N6.

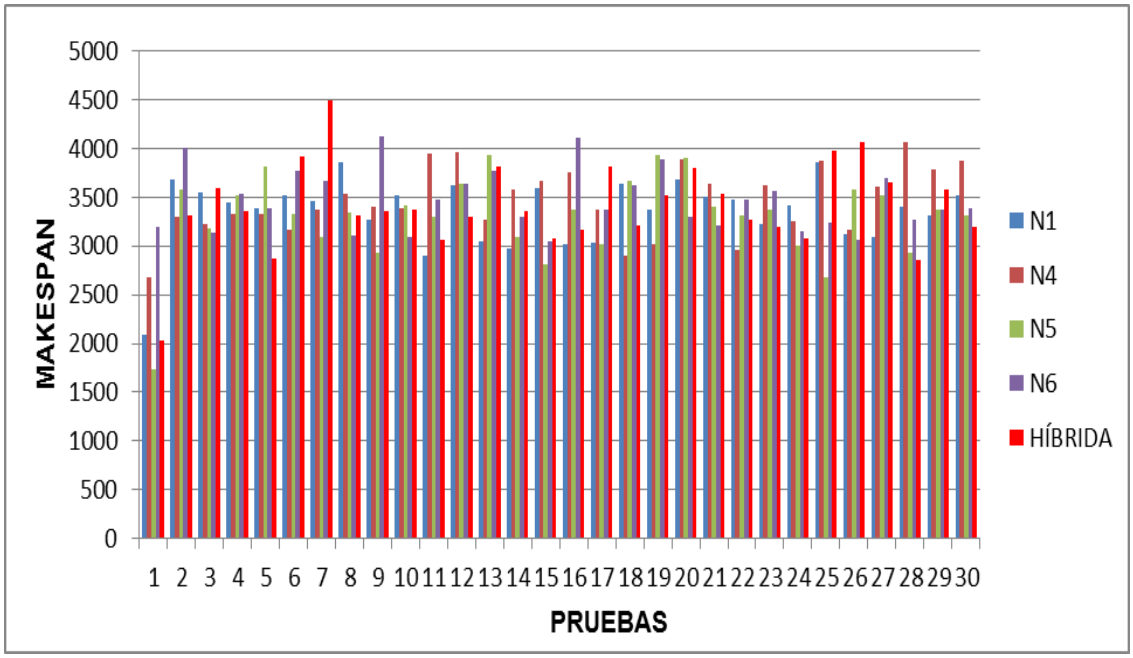

Figura 4.2. Resultados LA39

Los resultados de las 30 ejecuciones obtenidas para el benchmark LA40 del problema de calendarización de trabajos en los talleres de manufactura para cada estructura de vecindad se muestran en la Figura 4.3. Con una cota superior en la función objetivo de 1222, en la ejecución 1 se observa que la estructura con mejor desempeño es la N1, aunque el valor se encuentra por encima de la cota establecida, en segundo lugar la N5, en tercer lugar la estructura híbrida, en cuarto lugar la estructura N6 y en último lugar la N6.

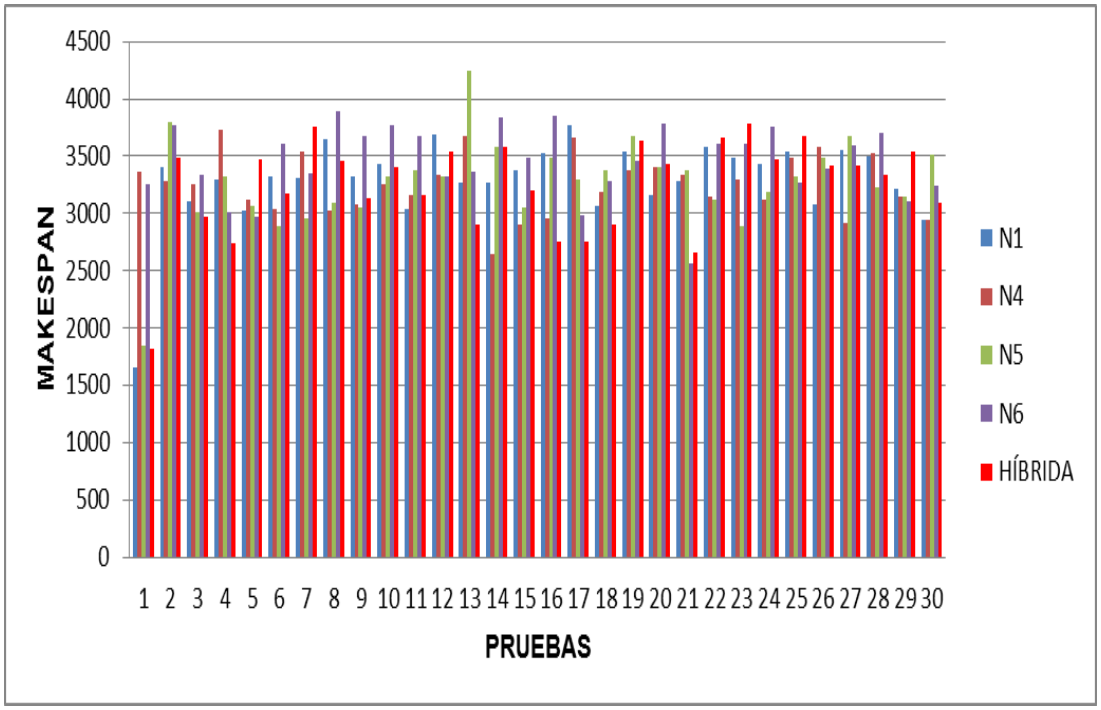

Figura 4.3. Resultados para LA40

Se muestran en la Figura 4.4, los resultados de las 30 ejecuciones obtenidas para el benchmark YN1 del problema de calendarización de trabajos en los talleres de manufactura para cada estructura de vecindad. Con una cota superior de 885 en la función objetivo, la mejor estructura de vecindad fue la estructura N5, seguida en segundo lugar por la estructura híbrida, en tercer lugar la estructura de vecindad N1, en cuarto lugar.

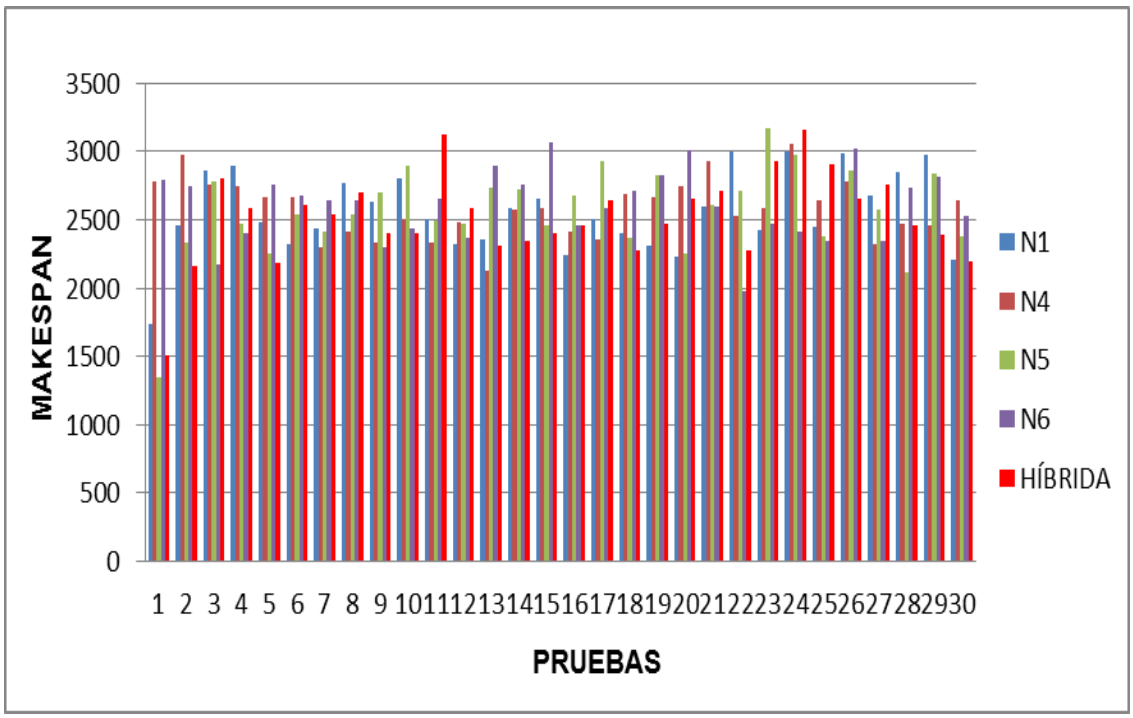

Figura 4.4. Resultados YN1

Los resultados de las 30 ejecuciones obtenidas para el benchmark YN2 del problema de calendarización de trabajos en los talleres de manufactura para cada estructura de vecindad se muestran en la Figura 4.5. Con una cota superior de 909 en la función objetivo, la mejor estructura de vecindad fue la estructura híbrida, seguida en segundo lugar por la estructura N5, en tercer lugar la estructura de vecindad N1, en cuarto lugar N6.

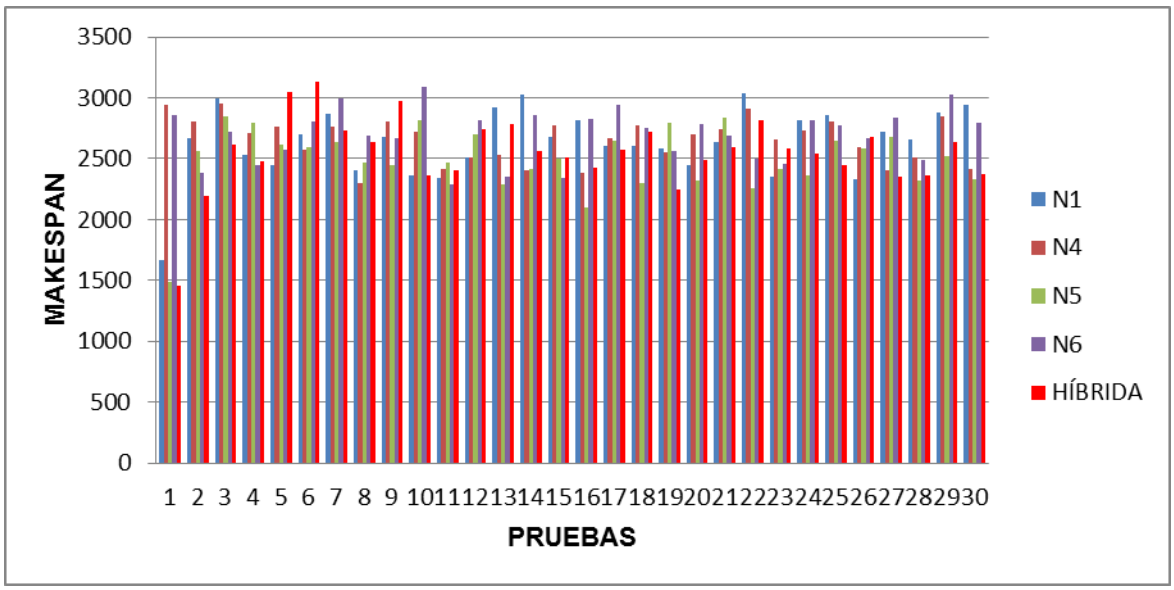

Figura 4.5. Resultados YN2

En la gráfica de la Figura 4.6, se observan los resultados de las 30 ejecuciones obtenidas para el benchmark YN3 del problema de calendarización de trabajos en los talleres de manufactura para cada estructura de vecindad. Con una cota superior de 892 en la función objetivo, la mejor estructura de vecindad fue la estructura híbrida, seguida en segundo lugar por la estructura N5, en tercer lugar la estructura de vecindad N1, en cuarto lugar N6.

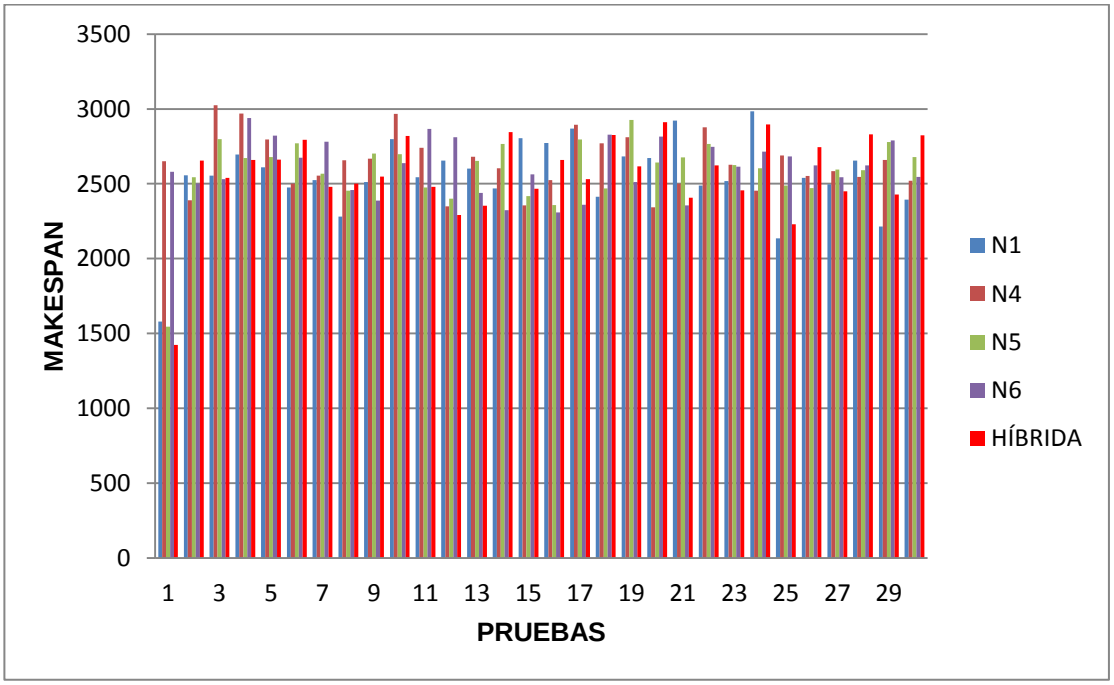

Figura 4.6. Resultados YN3

A continuación los resultados de las 30 ejecuciones obtenidas para el benchmark YN4 del problema de calendarización de trabajos en los talleres de manufactura para cada estructura de vecindad, se muestran en la Figura 4.7. Con una cota superior de 968 en la función objetivo, la mejor estructura de vecindad fue la estructura N5, seguida en segundo lugar por la estructura híbrida, en tercer lugar la estructura de vecindad N1, en cuarto lugar N4.

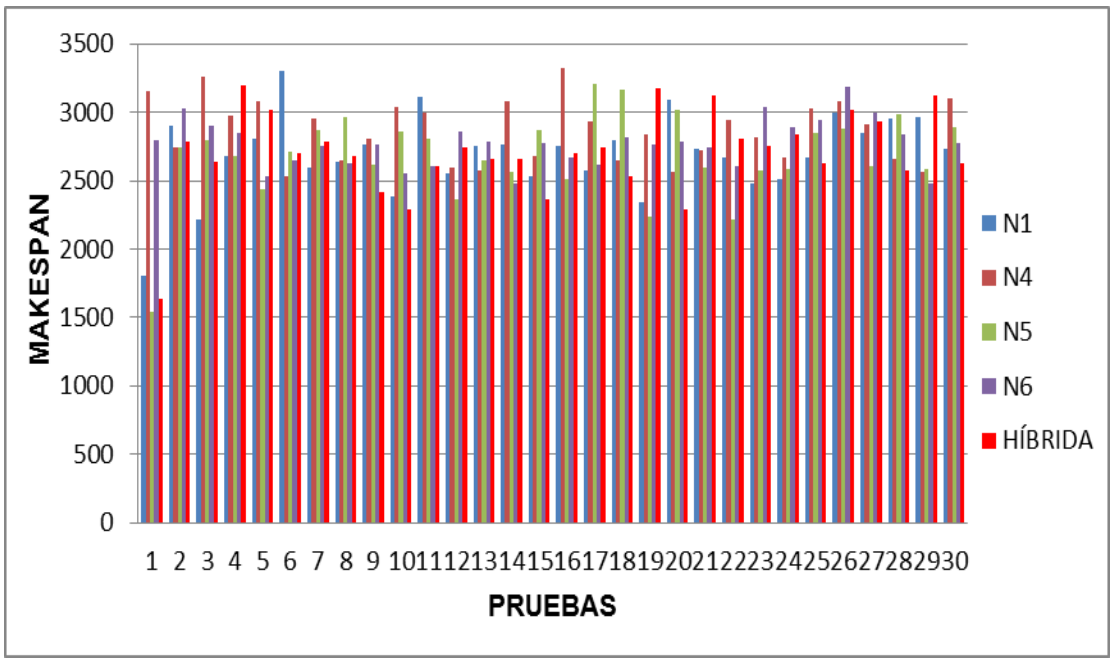

Figura 4.7. Resultados YN4

4.4 Resultados experimentales de las estructuras de vecindad

Para conocer el comportamiento de un algoritmo, es necesario recurrir al cálculo de parámetros estadísticos como son la media, desviación estándar y error relativo porcentual. La media permite obtener el promedio de las soluciones encontradas para cada instancia. La desviación estándar permite conocer que tan dispersas están las soluciones con respecto a la media aritmética.

El error relativo permite conocer el porcentaje de error de la mejor solución encontrada con respecto a una cota establecida, que para este caso es la solución óptima correspondiente a cada instancia. De acuerdo a los resultados obtenidos por dichos parámetros, se puede determinar si un algoritmo es bueno o no para cierto problema.

El error relativo porcentual se obtiene de la formula (4.1), en esta fórmula *VE* es el valor esperado, en este caso la mejor cota conocida (*makespan*) del problema, *VC* es el valor obtenido de los experimentos, el cual es el *makespan*.

$$\%Er = \frac{VE - VC}{VE}\ 100 \qquad (4.1)$$

Para la obtención de resultados de la estructura híbrida de vecindad, se seleccionaron casos de pruebas entre los *benchmarks* más conocidos y aplicados por la comunidad científica, los cuales se encuentran en la librería OR [Beasley, 03], y que presenta una recopilación de los problemas más difíciles para JSSP.

Con esta elección, fue posible realizar una comparación de los resultados obtenidos en esta investigación contra otros reportados en la literatura. Se tomaron 7 benchmarks, clasificados por la comunidad científica como pequeños, medianos y grandes. El benchmark FT10 [Fisher and Thompson,1963], el cual se tardó más de 20 años en obtener su solución y que por lo general se usa como un estándar para medir la eficiencia de algoritmos propuestos para JSSP; dos problemas medianos que constan de 15 máquinas y 15 trabajos, cada trabajo consta de 15 operaciones, los problemas son LA39 Y LA40 [Lawrence,1984], considerados los más difíciles en la categoría de tamaño mediano; cuatro problemas grandes que constan de 20 máquinas y 20 trabajos, cada trabajo consta de 20 operaciones, los problemas son YN1, YN2, YN3, YN4 [Yamada and Nakano,1992], los cuales a la fecha no se ha encontrado la solución óptima de estos cuatro problemas, solo se conocen cotas próximas al óptimo [Bauhaus,2004].

Como se puede notar, solo se eligieron tipos de problemas llamados cuadrados o simétricos, esto es, el número de máquinas es igual al número de trabajos *(m = n)*, este tipo de problemas se consideran más difíciles para encontrar o acercarse a una solución óptima global en JSSP, que cuando se tienen problemas no simétricos [Taillard, 1994], [Watson *et al*, 2003].

Se puede apreciar los resultados para los diferentes benchmarks, en la Tabla 4-13, al aplicarles la estructura de vecindad híbrida, se observan los mejores resultados así como los peores para cada uno de estos. En la primera columna se observan los benchmarks de prueba, la segunda columna corresponde a la cota del benchmark, las cotas denotadas por un asterisco identifican el óptimo encontrado para ese benchmark, la tercera columna representa la mejor solución encontrada de acuerdo a las 30 ejecuciones realizadas para cada instancia. La cuarta columna representa la peor solución encontrada, es decir la más lejana al óptimo. La columna de promedio representa el promedio total de las 30 ejecuciones para cada instancia.

Tabla 4-13 Resultados de la estructura híbrida en varios benchmarks

Benchmark	Cota	Mejor	Peor	Promedio	%Er	Σ
FT10	930*	1279	2867	2310.8	37.5	354.8
LA39	1233	2026	4495	3403.1	20.2	450.5
LA40	1222*	1823	3782	3245.5	49.1	422.1
YN1	885	1504	3159	2519.1	69.9	320.6
YN2	909	1458	3046	2549.2	60.3	301.1
YN3	892	1425	2911	2565.2	59.7	283.7
YN4	968	1632	3192	2699.8	68.5	314.8

Para realizar las pruebas experimentales, se ejecutaron 30 veces cada uno de los problemas de prueba antes mencionados, tomando como criterio de paro el número de iteraciones y el tamaño de la vecindad, de acuerdo al análisis de sintonización hecho. Como primer paso, se ejecutaron cada una de las estructuras de vecindad implementadas, incluyendo la estructura de vecindad híbrida, con el objetivo de conocer el rendimiento de cada una de ellas por separado.

Se puede observar, en las Tablas 4-14 y 4-15, en la primera columna el tipo de benchmark utilizado, la segunda columna representa la cota para cada benchmark reportado en la literatura, mejor representa la mejor solución encontrada de acuerdo al total de ejecuciones realizadas para cada instancia, peor es la solución más lejana a la cota encontrada, promedio representa el promedio de todas las operaciones, %Er es el error relativo de la mejor solución encontrada con respecto al óptimo.

En la Tabla 4-14 se muestran los resultados de las estructuras de vecindad N1 y N4, aplicadas a varios benchmarks para el problema del JSSP, se observa que la mejor solución de entre estas dos estructuras de vecindad, fue obtenida por la estructura N1, la peor solución se obtuvo por la estructura N4, en cuanto al mejor promedio, se obtuvo por la estructura N1, mientras que el mejor error relativo fue obtenido por la estructura de vecindad N1.

Tabla 4-14 Resultados de las estructuras de vecindad N1 Y N4

		N1				**N4**			
Benchmark	Cota	Mejor	Peor	Promedio	%Er	Mejor	Peor	Promedio	%Er
FT10	930*	**1376**	2772	**2404.3**	47.9	2046	2841	2578.7	205.4
LA39	1233	2084	3864	3351.4	**24.3**	2682	**4058**	3463.4	60.0
LA40	1222*	1656	3641	3294.5	35	2647	3729	3249.1	116
YN1	885	1735	3003	2555.1	96	2123	3054	2583.8	139
YN2	909	1666	3023	2636.2	83.2	2301	2929	2656.0	153
YN3	892	1579	2984	2547.2	77	2349	3024	2641.9	163
YN4	968	1800	3304	2697.5	85.9	2528	3322	2863.7	161

Los resultados de las estructuras de vecindad N5 y N6 se muestran en la Tabla 4-15, se aprecia que el mejor resultado obtenido de entre estas dos estructuras de vecindad, fue por la estructura N5, y en cuanto al peor resultado, se obtuvo por la estructura N5 también, así como el mejor error relativo.

Tabla 4-15 Resultados de las estructuras de vecindad N5 Y N6.

		N5				**N6**			
Benchmark	Cota	Mejor	Peor	Promedio	%Er	Mejor	Peor	Promedio	%Er
FT10	930*	**1184**	3100	**2393.7**	27.3	1908	2768	2529.8	105
LA39	1233	1742	3933	3299.9	**3.9**	3052	4011	3464.4	82
LA40	1222*	1841	**4245**	3269.0	50	2559	3890	3450.4	109
YN1	885	1348	3174	2561.1	52	1983	3071	2604.6	124
YN2	909	1484	2843	2492.3	63	2290	3085	2693.2	151
YN3	892	1546	2927	2586.9	73.3	2308	2940	2612.4	158
YN4	968	1537	3202	2679.13	58.7	2478	3184	2768.76	155

Al calcular el error relativo del mejor resultado para cada una de las instancias con cada una de las estructuras de vecindad, se obtiene que la estructura de vecindad que menor porcentaje de error relativo tiene en todas las instancias de prueba tratadas, es la función de vecindad N5. En base a este resultado, se compararon los resultados obtenidos por la estructura de vecindad N5 con la implementación de la estructura de vecindad híbrida, tomando como parámetro de control el tamaño de la lista tabú y el tamaño de vecindad más grande.

De acuerdo al análisis de resultados, en la Tabla 4-16, se puede observar que la estructura híbrida de vecindad, la cual es una combinación de las cuatro estructuras de vecindad explicadas en el capítulo 3, propuesta en esta tesis tiene un porcentaje de error relativo competitivo con la mejor estructura de vecindad en todas las instancias de pruebas, que es la estructura de vecindad N5.

Tabla 4-16 Resultados de la estructura híbrida contra la estructura N5

		HÍBRIDA				**N5**			
Benchmark	Cota	Mejor	Peor	Promedio	%Er	Mejor	Peor	Promedio	%Er
FT10	930*	1279	2867	7440.2	**37.5**	1184	3100	2393.7	**27.3**
LA39	1233	2016	4495	13656.4	**20.2**	1742	3933	3299.9	**3.9**
LA40	1222*	1823	3782	17215.5	**49.1**	1841	4245	3269	**50**
YN1	885	1504	3159	39084.0	**69.9**	1348	3174	2561.1	**52**
YN2	909	1458	3046	44718.2	**60.3**	1484	2843	2492.3	**63**
YN3	892	1425	2911	29672.6	**59.7**	1546	2927	2586.9	**73.3**
YN4	968	1632	3192	30853.0	**68.5**	1537	3202	2679.1	**58.7**

4.5 Análisis de Resultados

Para comparar el desempeño de la estructura de vecindad híbrida, se realizaron pruebas experimentales con cada una de las estructuras de vecindad implementadas, así como de la estructura híbrida propuesta, para cada una de las instancias de prueba se ejecutó 30 veces el algoritmo bajo los parámetros de sintonización obtenidos. Estas pruebas fueron realizadas bajo las mismas condiciones manejadas para cada estructura de vecindad. A continuación se muestran los resultados en la Tabla 4-17.

Tabla 4-17 Resultados de las estructuras de vecindad e híbrida

BENCHMARK	%Er/N1	%Er/N4	%Er/N5	%Er/N6	%Er/HÍBRIDA
FT10	47.9	205.4	27.3	105	37.5
LA39	**24.3**	**60**	**3.9**	82	**20.2**
LA40	35	116	50	109	49.1
YN1	96	139	52	124	69.9
YN2	83.2	153	63	151	60.3
YN3	77	163	73.3	158	59.7
YN4	85.9	161	58.7	155	68.5

Los resultados obtenidos por las estructuras N1, N4 Y N6, tienen un error relativo bastante alto con respecto a los obtenidos por la estructura híbrida. Para observar de manera más precisa estos datos, en la Figura 4.8, se grafica los errores relativos de cada una de estas estructuras. Es posible observar, en base al error relativo que las mejores soluciones para el JSSP, fueron en primer lugar la estructura de vecindad N5 y en segundo lugar la estructura híbrida de vecindad propuesta en esta tesis. Por lo que se comprueba que la aplicación de la estructura de vecindad híbrida hace eficaz al problema de calendarización de trabajos en talleres de manufactura.

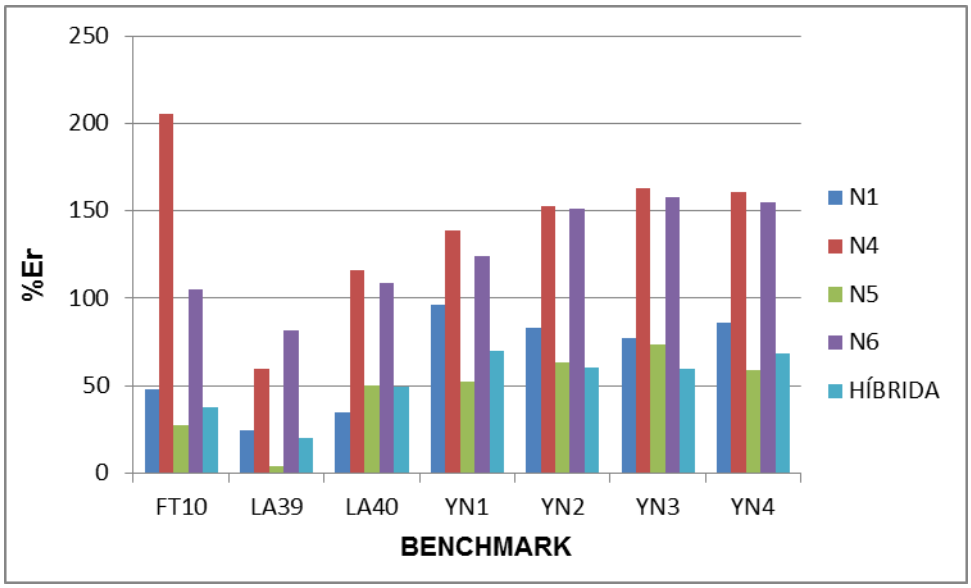

Figura 4.8. Gráfica comparativa del error relativo.

Capítulo 5 Conclusiones y trabajos futuros

5.1 Conclusiones

Con base en las pruebas experimentales realizadas a las diferentes estructuras de vecindad, aplicando una estructura hibrida de vecindad, se puede concluir que la estructura híbrida propuesta es competitiva en eficacia y en eficiencia con respecto al resto de las estructuras analizadas.

Al comparar los resultados obtenidos por las diferentes estructuras de vecindad, se puede decir que la estructura híbrida de vecindad es mejor en tanto que esta, aumenta la velocidad de explotación del vecindario. La estructura de vecindad N5 queda arriba de la híbrida con un error relativo de 3.9%, seguida en tercer lugar por la estructura de vecindad N1 que obtuvo un error relativo de 24.3%, en cuarto lugar la estructura de vecindad N4 con un error relativo de 60% y en último lugar la estructura N6 con un error relativo de 82%.

De acuerdo a los resultados obtenidos para las diferentes instancias del JSSP, para espacios de soluciones más complejos, estos se ven beneficiados con la aplicación de una estructura de vecindad híbrida, debido que al incorporar características de diversas estructuras, los saltos que realiza dentro del espacio de soluciones varían en magnitud, debido a que la selección de la estructura de vecindad a aplicar se realiza de manera aleatoria.

5.2 Trabajos Futuros

- Realizar un algoritmo paralelo en el que se implemente la estructura de vecindad híbrida, para mejorar su eficiencia y eficacia, probando una mayor cantidad de instancias.

- Implementar la estructura de vecindad híbrida en metaheurísticas, como por ejemplo, recocido simulado y evaluar la eficiencia y eficacia, con las mismas instancias de pruebas, así como también comparar esta eficiencia y eficacia con otras metaheurísticas.

- Realizar la investigación requerida y análisis necesario para encontrar mejores métodos, que evalúen las soluciones obtenidas por la estructura de vecindad híbrida.

REFERENCIAS

Anagnostopoulos Georgios C., Rabadi Ghaith, (2002), A Simulated Annealing Algorithm for the Unrelated Parallel Machine Scheduling Problem. School of Electrical Engineering & Computer Science, Orlando Florida.

Bauhaus, (2004), Universitat Weimar, Problem instances of JSSP, http://www.uni-weimar.de/~henning3/index.html, 2012.

Balas Egon, Vazacopoulos Alkis, (1998), Guided Local Search with Shifting Bottleneck for Job Shop Scheduling, Management Science, Vol. 44, No. 2.

Blazewicz Jacek, Domschke Wolfgang, Pesch Erwin, (1996), The job shop scheduling problem: Conventional and new solution techniques, European Journal of Operational Research 93 1-33, October 1995.

Brandimarte Paolo, (1993), Routing and Scheduling in a flexible job shop by tabu Search, Annals of Operations Research 41157-183.

Brucker, P., Jurisch, B., and Sievers, B. (1992), A branch and bound algorithm for the job shop Scheduling problem, Discrete Applied Mathematics, 49, 107-127.

Carlier J., Pinson E., (1989), An algorithm for solvin the Job Shop Problem, Management Science, Vol. 35, No. 2, February 1989.

Chong Soon Chin, Low Hean Yoke Malcolm, Sivakumar Iyer Appa, and Gay Leng Khen, (2006), A Bee colony optimization algorithm to job shop scheduling, Proceedings of the 2006 Winter Simulation Conference, L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds. 1-4244-0501-7/06/IEEE.

Cruz-**Chávez** Marco Antonio, Frausto Solís Juan, Ramos Quintana Fernando, (2004), The Problem of Using the Calculation of the Critical Path to Solver Instances of the Job Shop Scheduling Problem, International Journal of Computational Intelligence, ISSN(p): 1304-4508, ISSN(e): 1304-2386, Vol. 1, No. 4, pp. 334-337.

Cruz-Chávez, M. A, (2005), "Cooperación de Procesos para el Problema de Job Shop Scheduling Aplicando Recocido Simulado". Tesis de Doctorado. Marzo 2005.

Cruz-Chávez Marco Antonio, Martínez Rangel Martin, Hernández, J.A., Zavala Díaz José Crispín, Diaz Parra Ocotlán (2006-2007), An Algorithm of scheduling for the Job Shop Scheduling Problem, CIICAP, FCAeI, Autonomous University of Morelos State, Avenida Universidad 1001. Col. Chamilpa, C.P. 62210. Cuernavaca, Morelos, México, This work was supported by project 160 of the Fideicomiso SEP-UNAM.

Cruz-**Chávez** Marco Antonio, Ocotlán Díaz-Parra, David Juárez Romero, Eric Barreto Sedeño, Crispín Zavala Díaz, Martín G. Martínez Rangel, (2008), Un Mecanismo de Vecindad con Búsqueda Local y Algoritmo Genético para Problemas de Transporte con Ventanas de Tiempo, CICos (2008), 6to Congreso Internacional de Cómputo en Optimización y Software, ACD, ISBN(e) 978-607-00-0165-9, ISBN(i) 978-970-9750-26-3, pp 23-32, 25-27 Junio, México, 2008.

Cruz-Chávez Marco Antonio, Martínez Oropeza Alina J.C Zavala Díaz, Martínez Rangel Martin, (2009), Relajación del Problema de Calendarización de Trabajos en un Taller de Manufactura utilizando un Grafo Bipartita, CICos2009, ISBN:978-607-00-1970-8,pp. 178 – 188, Centro de Investigaciones en Ingeniería y Ciencias Aplicadas, Universidad Autónoma del Estado de Morelos Av. Universidad 1001, Col. Chamilpa, 62270, Cuernavaca, Morelos, México.

Cruz-Chávez Marco Antonio, Rodríguez Leon Abelardo, Avila Melgar Erika Yesenia, Juarez Perez Fredy, Cruz Rosales Martin H., Rivera Lopez Rafael, (2010), Gridification of Genetic Algoritm with Reduced Communication for the Job Shop Scheduling problem, CIICAP, FCAeI,FC,Autonomus University of Morelos State, Av. Universidad. Col. Chamilpa, C.P. 62209. Cuernavaca, Morelos, México. International Journal of Grid and Distributed Computing, Vol.3, No.3, September.

Cruz-Chávez Marco Antonio, Martínez-Oropeza Alina, Serna-Barquera Sergio A. (2010b). Neighborhood Hybrid Structure for Discrete Optimization, Robotics and Automotive Mechanics Conference, CERMA2010, IEEE Computer Society, ISBN 978-0-7695-4204-1, pp. 108-113.

Cruz-Chávez Marco Antonio, (2013), Neighborhood Generation Mechanism Applied in Simulated Annealing to Job Shop Scheduling Problems, International Journal of Systems Science, Taylor & Francis, ISSN: 0020-7721, Vol., No., pp., 2013. (To appear).

Dell'Amico Mauro, Trubian Marco, (1993), Appling Tabu search to the job-shop scheduling problem, Politecnico di Milano, 1-20133, Milano, Italy, Annals of Operations Research 41(1993)231-252.

Dell'Amico, M. (1993), shop problems with two machines and times lags, working paper, Politecnico di Milano.

Garey M. R., Johnson D. S. and Sethi R. (1976), The complexity of flowshop and jobshop scheduling, Math. Oper. Res. 1(1976)117-129.

Geyik Faruk, Cedimoglu Hakki Ismail, (2004), The strategies and parameters of tabu search for job-shop scheduling, Journal of Intelligent Manufacturing, Kluwer Academic Publishers, Manufactured in the Netherlands, 15,439-448.

Goncalves J. F., Magalhaes-Mendez J. J., Resendes M. G.C. (2005), A hybrid genetic algorithm for the job shop scheduling problem, European Journal of Operational Research 167, 77–95.

González T. and Sahni, S. (1978), Flowshop and jobshop schedules: Complexity and approximation, Operations Research 20, 36-52.

Glover, F. (1990). Tabu Search: Part II. ORSA Journal on Computing.

Glover Fred, Laguna Manuel, (1997), Tabu Search, Kluwer Academic Publishers, Massachusetts, ISBN 0-7923-9965-X.

Juárez Chávez Jazmín Y., (2011), Algoritmo de recocido simulado para maximizer la Resistencia mecánica en aceros microaleados, Tesis de Maestría, Julio 2011.

Kolonko M., (1999), Some new results on simulated annealing applied to the job shop scheduling problem, European Journal of Operational Research 113, 123-136.

Kuo-Ling H., Ching-Jong L., Ant colony optimization combined with taboo search for the job shop Scheduling problem, Computers & Operations Research 35 (2008) 1030 – 1046.

Lawler, E.L, Lenstra, J.K., Rinnooy Kan, A.H.G., and Shmoys, D.B. (1993), Sequencing and Scheduling: algorithms and complexity, in: A.H.G.Rinnooy Kan and P.H. Zipkin (eds.) Handbooks in Operations Research and Management Science, Vol, 4: Logistics of production and Inventory Elsevier, Amsterdam.

Lenstra J.K. (1977), Sequencing by Enumerative Methods, Mathematical Center Tract 69, Mathematisch Centrum, Amsterdam.

Lenstra, J.K., and Rinnooy Kan, A.H.G. (1979), Computational complexity of discrete optimization problems, Annals of Discrete Mathematics 4, 121-140.

Matsuo, H., Sh, C.J, and Sullivan, R.S. (1988), Acontrolled Search simulated annealing method for the general jobshop Scheduling problem, working paper 03-04-88, University of Texas at Austin.

Martínez Morales A., (2006), "Algoritmo Basado en Tabú Search para el Cálculo del Índice de Transmisión de un Grafo. Departamento de computación Facultad de ciencias y tecnología". FARAUTE de Ciencias y Tecnología, Vol. 1, No. 1, páginas 31-39. Universidad de Carabobo, Valencia, Estado Carabobo, Venezuela.

Martínez, O. A. (2010). "Solución al Problema de Máquinas en Paralelo no Relacionadas mediante un Algoritmo de Colonia de Hormigas", Tesis de Maestría, Agosto 2010.

Martínez B. B., (2011) "Solución al problema del árbol de expansión mínima aplicando recocido simulado con búsqueda tabú", Tesis de Maestría, Julio 2011.

Martínez Gil Francisco A., Martín Quetglás Gregorio. Introducción a la Programación Estructurada en C. ISBN. 84-370-5666-7. Universidad de Valencia. pp. 209-212. Ed. Maite Simon.2003.

Meeran† Sheik, Anant Singh Jain (1998), A state-of-the-art review of job-shop scheduling techniques, Department of Applied Physics, Electronic and Mechanical Engineering University of Dundee, Dundee, Scotland, UK, DD1 4HN.

Muth J.F. and G.L. Thompson (1963). Industrial Scheduling. Prentice-Hall, Engle-wood Cli_s, N.J.

Nakano Ryohei, Yamada Takeshi, Conventional Genetic Algorithm for Job Shop Problems, (1991), Comm. And Info. Proc. Labs, NTT 1-2356 Take, Yokosuka, 238-03, Japan.

Nowicki Eugeniusz, Smutnicki Czeslaw (1996), A fast Taboo Search Algorithm for the Job Shop Problem, Technical University of Wroctaw, Institute of Engineering Cybernetics,ul. Janiszewskiego1 1/17, 50-372 Wroctaw, Poland, Management Science, Vol. 42, No. 6 (Jun.), pp. 797-813

Nowicki Eugeniusz, Smutnicki Czeslaw, (2005), An advanced Tabu Search Algorithm for the Job Shop Problem, Springer Science + Business Media, Inc. Netherlands. Journal of Scheduling 8: 145-159.

Pannell, D. J. (2009). Sensitivity analysis: strategies, methods, concepts, examples.

Papadimitriou and Steiglitz, Algorithms and Complexity, Prentice Hall, Englewood, (1982), ISBN 0-486-40258-4.

Pinedo, M. (2008). Scheduling Theory, Algorithms and Systems.Vol.1 3rd Edition, ISBN 9780387789347.

Roy and Sussman, (1964), Les problemas d'ordonnancement avec contraintes disjontives, Note D.S. no 9 bis, SEMA, Paris, France, December 1964.

Sha D.Y., Cheng-Yu Hsu, (2006), A hybrid particle swarm optimization for job shop scheduling problem, Computers & Industrial Engineering 51, 791–808.

Sotskov,Y.N. and Shaklevich, N.V. (1995), Np-hardness of shop schedulings problems with three jobs, Discrete Applied Mathematics 59, 237-266.

Taillard Eric D., (1994), Parallel Taboo Search Techniques for the Job Shop Scheduling Problem, ORSA Journal on Computing, Vol. 6, No. Spring, Operations Research Society of America, ISBN 0899-1499/94/0602-0108.

Van Laarhoven, P.J.M., and Aarts, E.H.L., (1987), Simulated Annealing: Theory and Applications, Reidel, Dordrecht, Netherlands.

Van Laarhoven, P.J.M., Aarts, E.H.L, and Lenstra, J.K. (1992), Job Shop Scheduling by simulated annealing, Operations Research 40,113-125.

Watson Jean Paul, J. Christopher Beck, Adele E. Howe, L. Darrel Whitley, (2003), Problem difficulty for tabu Search in job shop Scheduling, Artificial intelligence, 143(2): 189-217.

Yamada R. and Nakano (1992), A genetic algorithm applicable to large-scale job-shop instances, R. Manner, B. Manderick (eds.), Parallel instance solving from nature 2, North-Holland, Amsterdam, 281-290.

Yong Zhang Chao, Li PeiGen, Guan ZaiLin, Rao YunQing, (2007), A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem, Computers & operations Research 34, 3229-3242.

Zalzala A.M.S., Fleming P.J., (1997), Genetic algorithms in engineering systems, The institution of Electrical Engineers, London, United Kingdom, pag 135, ISBN 0 85296 902 3.

Zhang Ch. Y., Li P., Rao Y., Guan Z. (2008), A very fast TS/SA algorithm for the job shop Scheduling problem , Computers & Operations Research 35, 282– 294.

APÉNDICE A

Estructuras de datos utilizadas para el problema del JSSP

Para generar la representación simbólica, se hace uso de dos estructuras de datos para guardar y manipular la información de las operaciones que representan cada uno de los nodos del grafo disyuntivo, ya que estas estructuras de datos permiten el uso eficiente de la información. A continuación se explican cada una de las estructuras utilizadas en el algoritmo propuesto.

```
struct maq{           // Restricciones de capacidad
           int j;     // Numero de trabajo
           int t;     // Tiempo de procesamiento
           int s;     // Tiempo de inicio
           int sr;    //tiempo inicio hacia atrás
           int nop;   //número de operación
           int max;   //método Rc
           int min;   //método Rc
           int flag;  //método Rc*/
};
```

La estructura *maq* permite almacenar la información sobre la secuencia de trabajos en cada máquina, en esta estructura se guarda el trabajo al que pertenece cada operación, el tiempo de ejecución y el tiempo de inicio de esta operación. Así mismo, se almacena información sobre los recorridos hacia adelante y hacia atrás de la ruta crítica.

```
struct trab{      // Maneja restricciones de precedencia
       int m;     // número de maq
       int t;     // Tiempo de procesamiento
       int s;     // Tiempo de inicio
       int sr;    //tiempo inicio hacia atrás
       int nop;   //número de operación
       int max;   //método Rc
       int min;   //método Rc
       int hol;   //método Rc
       int flag;  //método Rc
};
```

La estructura *trab* almacena toda la información de una operación de acuerdo al trabajo que pertenezca, su tiempo de procesamiento, en que máquina se lleva a cabo esa operación, su tiempo de inicio, que numero de operación es, la holgura de cada operación e información de los recorridos hacia adelante y hacia atrás de la ruta crítica.

APÉNDICE B

Función Temporal de la estructura de vecindad N1

CODIGO	COSTO	VECES QUE SE EJECUTA
`numB = 1 + rand() % ibloq ;`	C_1	O(1)
`numOpe = 1 + rand() % ((bloques[numB][0])-1;`	C_2	O(1)
`m= bloques[numB][bloques[numB][0] + 1];`	C_3	O(1)
`for(int i=1;i<n;i++)`	C_4	N
`if(opm[m][i].nop==bloques[numB][numOpe]){`	C_5	$\sum_{i=1}^{n} i$
`aux=opm[m][i].nop;`	C_6	$\sum_{i=1}^{n} i$
`if(!buscaListaTabu(i,m)){`	C_7	$\sum_{i=1}^{n} i$
`*posI = i; *maq = m;`	C_8	$\sum_{i=1}^{n} i$
`opm[m][i].nop = opm[m][i+1].nop;`	C_9	$\sum_{i=1}^{n} i$
`opm[m][i+1].nop = aux;`	C_{10}	$\sum_{i=1}^{n} i$
`aux2 = opm[m][i].t;`	C_{11}	$\sum_{i=1}^{n} i$
`opm[m][i].t = opm[m][i+1].t;`	C_{12}	$\sum_{i=1}^{n} i$
`opm[m][i+1].t = aux2;`	C_{13}	$\sum_{i=1}^{n} i$
`aux=opm[m][i].j;`	C_{14}	$\sum_{i=1}^{n} i$
`opm[m][i].j = opm[m][i+1].j;`	C_{15}	$\sum_{i=1}^{n} i$
`opm[m][i+1].j = aux;`	C_{16}	$\sum_{i=1}^{n} i$

NOTA: Por inducción matemática sabemos que:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \frac{n^2+n}{2}$$

Del análisis del código anterior se desprende que:

$T(n)= n\ (c_1)\ +\ (\ \frac{n^2+n}{2}\ \ (16c))$

Desarrollamos la multiplicación primero:

$(\ \frac{n^2+n}{2}\ \ (16c))\ =\ \frac{16\,n^2c\,+16\,nc}{2}$

Continuando el desarrollo:

$T= n(c) + 8\ n^2\ c + 8\ nc$

T= c (9n + 8)

$$\boldsymbol{T(n)= 9n\ +\ 8}$$

La función temporal T(n) = 9n + 8 para la estructura de vecindad N1, es de tipo Polinomial y consta de una cota superior en la cual la constante está determinada por **c**. La premisa de la función temporal para el peor de los casos es:

$$O\big(g(n)\big) = \{f(n) \exists c, n_0, \forall n \geq n_0, 0 \leq f(n) \leq cg(n)\}$$

$$f(n) = 9n + 8 = O(n) si\ \exists c: \quad 0 \leq 9n + 8 \leq cn \quad n \geq n_0 = 6$$

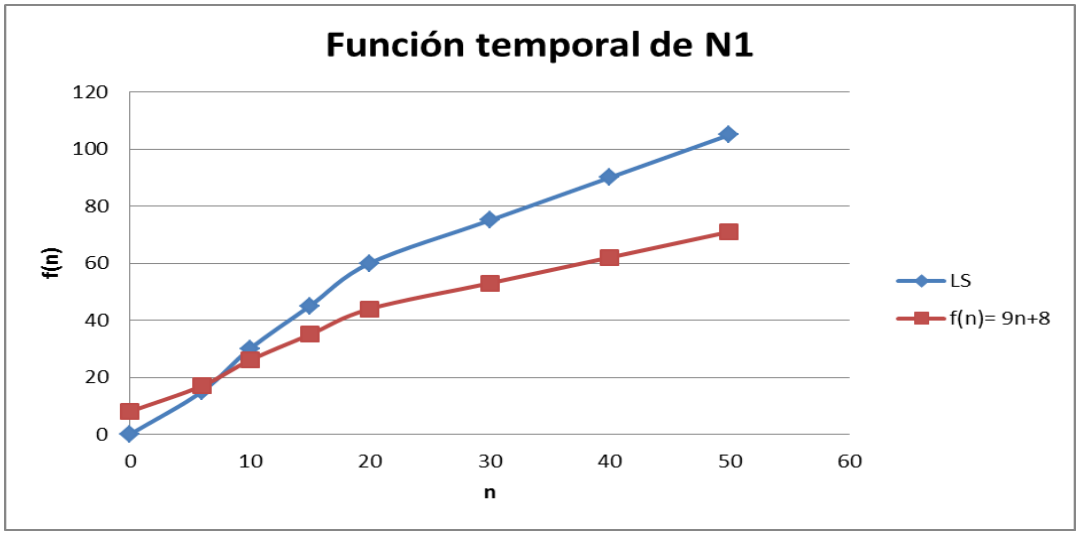

Figura B.1. Gráfica de la función temporal de la estructura N1

De acuerdo a la evaluación de la complejidad asintótica, para la estructura de vecindad N1, la complejidad es de *O*(*n*), con un valor para la constante *c* de 15, para un valor inicial de n_0 =6.

Función Temporal de la estructura de vecindad N4

CODIGO	COSTO	VECES QUE SE EJECUTA
numB = 1 + rand() % ibloq ;	C_1	O(1)
v++;	C_2	O(1)
if(v>5)goto sal;	C_3	O(1)
b++;	C_4	O(1)
if(b>30)goto end;	C_5	O(1)
}while(bloques[numB][0] < 3);	C_6	O(log n)
numOpe=2 + rand() % ((bloques[numB][0]) – 2;	C_7	O(1)
m= bloques[numB][bloques[numB][0] + 1];	C_8	O(1)
mov = 1+ rand() % 2;	C_9	O(1)
for(int i=1;i<nj;i++)	C_{10}	N
if(opm[m][i].nop == bloques[numB][1]){	C_{11}	$\sum_{i=1}^{n} i$
if(!buscaListaTabu(i,m)){	C_{12}	$\sum_{i=1}^{n} i$
*posI = i; *maq = m;	C_{13}	$\sum_{i=1}^{n} i$
aux=i;	C_{14}	$\sum_{i=1}^{n} i$
break;	C_{15}	$\sum_{i=1}^{n} i$
aux2= aux + numOpe – 1;	C_{16}	$\sum_{i=1}^{n} i$
pos = aux + bloques[numB][0]-1;	C_{17}	$\sum_{i=1}^{n} i$
datos1 = opm[m][aux2].nop;	C_{18}	$\sum_{i=1}^{n} i$
datos2 = opm[m][aux2].t;	C_{19}	$\sum_{i=1}^{n} i$
datos3 = opm[m][aux2].j;	C_{20}	$\sum_{i=1}^{n} i$
datos4 = opm[m][pos].nop;	C_{21}	$\sum_{i=1}^{n} i$

datos5 = opm[m][pos].t;	C_{22}	$\sum_{i=1}^{n} i$
datos6 = opm[m][pos].j;	C_{23}	$\sum_{i=1}^{n} i$
if(mov == 1){	C_{24}	$\sum_{i=1}^{n} i$
opm[m][aux2].nop = opm[m][aux].nop;	C_{25}	$\sum_{i=1}^{n} i$
opm[m][aux].nop = datos1;	C_{26}	$\sum_{i=1}^{n} i$
opm[m][aux2].t = opm[m][aux].t;	C_{27}	$\sum_{i=1}^{n} i$
opm[m][aux].t = datos2;	C_{28}	$\sum_{i=1}^{n} i$
opm[m][aux2].j = opm[m][aux].j;	C_{29}	$\sum_{i=1}^{n} i$
opm[m][aux].j = datos3;	C_{30}	$\sum_{i=1}^{n} i$
}else{	C_{31}	$\sum_{i=1}^{n} i$
opm[m][aux2].nop = opm[m][pos].nop;	C_{32}	$\sum_{i=1}^{n} i$
opm[m][pos].nop = datos1;	C_{33}	$\sum_{i=1}^{n} i$
opm[m][aux2].t = opm[m][pos].t;	C_{34}	$\sum_{i=1}^{n} i$
opm[m][pos].t = datos2;	C_{36}	$\sum_{i=1}^{n} i$
opm[m][aux2].j = opm[m][pos].j;	C_{37}	$\sum_{i=1}^{n} i$
opm[m][pos].j = datos3;	C_{38}	$\sum_{i=1}^{n} i$

NOTA: Por inducción matemática sabemos que:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \frac{n^2+n}{2}$$

Del análisis del código anterior se desprende que:

$T(n)= n\ (c_1)\ +\ (\ \frac{n^2+n}{2}\ \ (38c))$

Desarrollamos la multiplicación primero:

$(\ \frac{n^2+n}{2}\ \ (38c))\ =\ \ \frac{38\ n^2c + 38\ nc}{2}$

Continuando el desarrollo:

$T= n(c) + 19\ n^2\ c + 19\ nc$

T= c (20n + 19)

$T(n)= 20n + 19$

La función temporal T(n) = 20n + 19 para la estructura de vecindad N4, es de tipo Polinomial y consta de una cota superior en la cual la constante está determinada por **c**. La premisa de la función temporal para el peor de los casos es:

$$O\big(g(n)\big) = \{f(n) \exists c, n_0, \forall n \geq n_0, 0 \leq f(n) \leq cg(n)\}$$

$$f(n) = 20n + 19 = O(n) si\ \exists c: \quad 0 \leq 20n + 19 \leq cn \quad n \geq n_0 = 10$$

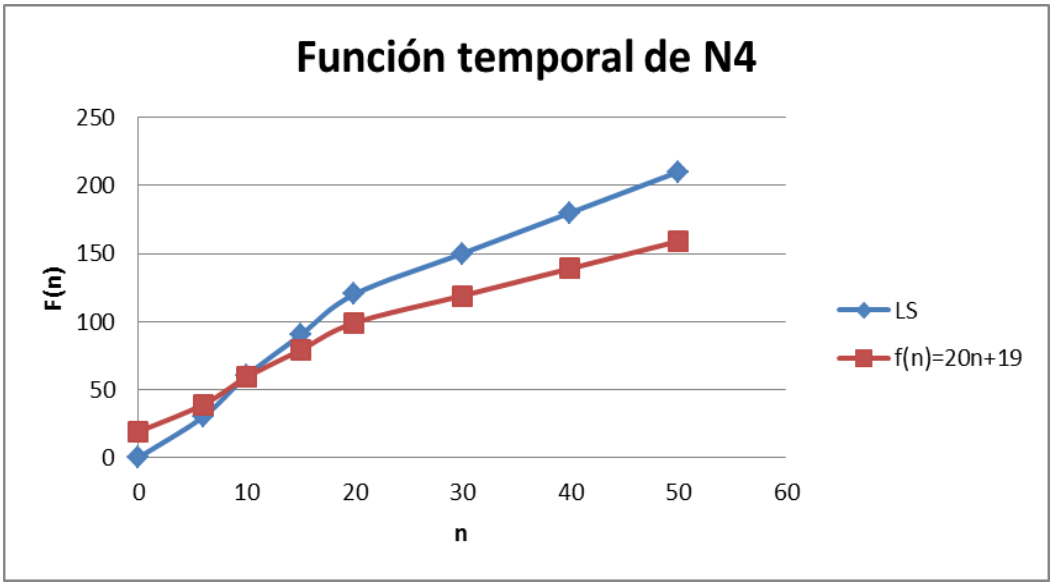

Figura B.2. Gráfica de la función temporal de la estructura N4

De acuerdo a la evaluación de la complejidad asintótica, para la estructura de vecindad N4, la complejidad es de *O*(*n*), con un valor para la constante *c* de 30, para un valor inicial de n_0 = 10.

Función Temporal de la estructura de vecindad N5

CODIGO	COSTO	VECES QUE SE EJECUTA
`if(ibloq == 1)goto sal;`	C_1	O(1)
`numB2 = 1 + rand() % ibloq;`	C_2	O(1)
`}while(numB == numB2);`	C_3	O(1)
`numOpe = bloques[numB][0]-1;`	C_4	O(1)
`m= bloques[numB][bloques[numB][0]+1];`	C_5	O(1)
`for(int i=1;i<nj;i++){`	C_6	N
`if(opm[m][i].nop==bloques[numB][numOpe]){`	C_7	$\sum_{i=1}^{n} i$
`aux = opm[m][i].nop;`	C_8	$\sum_{i=1}^{n} i$
`if(!buscaListaTabu(i,m)){`	C_9	$\sum_{i=1}^{n} i$
`*posI = i; *maq = m;`	C_{10}	$\sum_{i=1}^{n} i$
`opm[m][i].nop = opm[m][i + 1].nop;`	C_{11}	$\sum_{i=1}^{n} i$
`opm[m][i + 1].nop = aux;`	C_{12}	$\sum_{i=1}^{n} i$
`aux2 = opm[m][i].t;`	C_{13}	$\sum_{i=1}^{n} i$
`opm[m][i].t = opm[m][i + 1].t;`	C_{14}	$\sum_{i=1}^{n} i$
`opm[m][i + 1].t = aux2;`	C_{15}	$\sum_{i=1}^{n} i$
`aux3 = opm[m][i].j;`	C_{16}	$\sum_{i=1}^{n} i$
`opm[m][i].j = opm[m][i + 1].j;`	C_{17}	$\sum_{i=1}^{n} i$
`opm[m][i + 1].j = aux3;`	C_{18}	$\sum_{i=1}^{n} i$
`break;`	C_{19}	$\sum_{i=1}^{n} i$
`numOpe2 = bloques[numB2][1];`	C_{20}	$\sum_{i=1} i$

`m2= bloques[numB2][bloques[numB2][0]+1];`	C_{21}	$\sum_{i=1}^{n} i$
`for(int i=1;i<nj;i++){`	C_{22}	N
`if(opm[m2][i].nop == numOpe2){`	C_{23}	$\sum_{i=1}^{n} i$
`aux4 = opm[m2][i].nop;`	C_{24}	$\sum_{i=1}^{n} i$
`opm[m2][i].nop = opm[m2][i + 1].nop;`	C_{25}	$\sum_{i=1}^{n} i$
`opm[m2][i + 1].nop = aux4;`	C_{26}	$\sum_{i=1}^{n} i$
`aux5 = opm[m2][i].t;`	C_{27}	$\sum_{i=1}^{n} i$
`opm[m2][i].t = opm[m2][i + 1].t;`	C_{28}	$\sum_{i=1}^{n} i$
`opm[m2][i + 1].t = aux5;`	C_{29}	$\sum_{i=1}^{n} i$
`aux6= opm[m2][i].j;`	C_{30}	$\sum_{i=1}^{n} i$
`opm[m2][i].j = opm[m2][i + 1].j;`	C_{31}	$\sum_{i=1}^{n} i$
`opm[m2][i +1].j = aux6;`	C_{32}	$\sum_{i=1}^{n} i$
`break;`	C_{34}	$\sum_{i=1}^{n} i$

NOTA: Por inducción matemática sabemos que:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \frac{n^2+n}{2}$$

Del análisis del código anterior se desprende que:

$T(n) = n\,(c_1) + \left(\frac{n^2+n}{2}\ (34c)\right)$

Desarrollamos la multiplicación primero:

$\left(\frac{n^2+n}{2}\ (34c)\right) = \frac{34\,n^2c + 34\,nc}{2}$

Continuando el desarrollo:

$T = n(c) + 17\,n^2\,c + 17\,nc$

T= c (18n + 17)

$$\boldsymbol{T(n) = 18n + 17}$$

La función temporal T(n) = 18n + 17 para la estructura de vecindad N5, es de tipo Polinomial y consta de una cota superior en la cual la constante está determinada por **c**. La premisa de la función temporal para el peor de los casos es:

$$O\big(g(n)\big) = \{f(n) \exists c, n_0, \forall n \geq n_0, 0 \leq f(n) \leq c_2 g(n)\}$$

$$f(n) = 18n + 17 = O(n) si\ \exists c: \quad 0 \leq 18n + 17 \leq cn \quad n \geq n_0 = 8$$

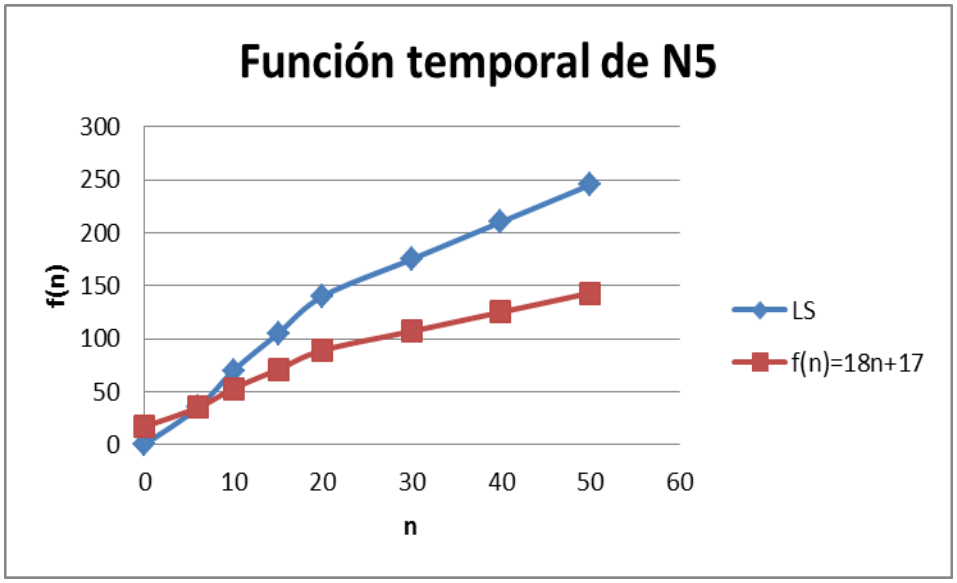

Figura B.3. Gráfica de la función temporal de la estructura N5

De acuerdo a la evaluación de la complejidad asintótica, para la estructura de vecindad N5, la complejidad es de *O* (*n*), con un valor para la constante *c* de 35, para un valor inicial de n_0 = 8.

Función Temporal de la estructura de vecindad N6

CODIGO	COSTO	VECES QUE SE EJECUTA
numB = 1 + rand() % ibloq ;	C_1	O(1)
v++;	C_2	O(1)
if(v>10)goto sal;	C_3	O(1)
b++;	C_4	O(1)
if(b>30)goto end;	C_5	O(1)
}while(bloques[numB][0] < 3);	C_6	O(log n)
numOpe= 2 + rand()%((bloques[numB][0]) - 2);	C_7	O(1)
m= bloques[numB][bloques[numB][0] + 1];	C_8	O(1)
mov = 1 + rand() % 2;	C_9	O(1)
for(int i=1;i<nj;i++)	C_{10}	N
if(opm[m][i].nop == bloques[numB][1]){	C_{11}	$\sum_{i=1}^{n} i$
if(!buscaListaTabu(i,m)){	C_{12}	$\sum_{i=1}^{n} i$
*posI = i; *maq = m;	C_{13}	$\sum_{i=1}^{n} i$
aux=i;	C_{14}	$\sum_{i=1}^{n} i$
break;	C_{15}	$\sum_{i=1}^{n} i$
aux2= aux + numOpe - 1;	C_{16}	$\sum_{i=1}^{n} i$
pos= aux + bloques[numB][0]-1;	C_{17}	$\sum_{i=1}^{n} i$
datos1 = opm[m][aux2].nop;	C_{18}	$\sum_{i=1}^{n} i$
datos2 = opm[m][aux2].t;	C_{19}	$\sum_{i=1}^{n} i$
datos3 = opm[m][aux2].j;	C_{20}	$\sum_{i=1}^{n} i$

datos4 = opm[m][pos].nop;	C_{21}	$\sum_{i=1}^{n} i$
datos5 = opm[m][pos].t;	C_{22}	$\sum_{i=1}^{n} i$
datos6 = opm[m][pos].j;	C_{23}	$\sum_{i=1}^{n} i$
if(mov == 1){	C_{24}	$\sum_{i=1}^{n} i$
for(int i = aux2; i > aux; i--){	C_{25}	N
opm[m][i].nop = opm[m][i-1].nop;	C_{26}	$\sum_{i=1}^{n} i$
opm[m][i].t = opm[m][i-1].t;	C_{27}	$\sum_{i=1}^{n} i$
opm[m][i].j = opm[m][i-1].j;	C_{28}	$\sum_{i=1}^{n} i$
opm[m][aux].nop = datos1;	C_{29}	$\sum_{i=1}^{n} i$
opm[m][aux].t = datos2;	C_{30}	$\sum_{i=1}^{n} i$
opm[m][aux].j = datos3;	C_{31}	$\sum_{i=1}^{n} i$
}else	C_{32}	$\sum_{i=1}^{n} i$
for(int i = aux2; i < pos; i++);	C_{33}	N
opm[m][i].nop = opm[m][i+1].nop;	C_{34}	$\sum_{i=1}^{n} i$
opm[m][i].t = opm[m][i+1].t;	C_{35}	$\sum_{i=1}^{n} i$
opm[m][i].j = opm[m][i+1].j;	C_{36}	$\sum_{i=1}^{n} i$
opm[m][pos].nop = datos1;	C_{37}	$\sum_{i=1}^{n} i$
opm[m][pos].t = datos2;	C_{38}	$\sum_{i=1}^{n} i$
opm[m][pos].j = datos3;	C_{39}	$\sum_{i=1}^{n} i$

NOTA: Por inducción matemática sabemos que:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \frac{n^2+n}{2}$$

Del análisis del código anterior se desprende que:

$T(n)= n\ (c_{10}) + n\ (c_{24}) + n\ (c_{33}) + \left(\frac{n^2+n}{2}\ (39c)\right)$

Desarrollamos la multiplicación primero:

$\left(\frac{n^2+n}{2}\ (39c)\right) = \frac{39\,n^2c + 39\,nc}{2}$

Continuando el desarrollo:

$T= n(c) + 19\ n^2\ c + 19\ nc$

$T= c\ (20n^2 + 17)$

$$\boldsymbol{T(n)= 20n^2 + 19}$$

La función temporal T(n) = $20n^2$ + 19 para la estructura de vecindad N6, es de tipo Polinomial y consta de una cota superior en la cual la constante está determinada por ***c***. La premisa de la función temporal para el peor de los casos es:

$$O\big(g(n)\big) = \{f(n) \exists c, n_0, \forall n \geq n_0, 0 \leq f(n) \leq c_2 g(n)\}$$

$$f(n) = 20n^2 + 19 = O(n) si\ \exists c: \quad 0 \leq 20n + 19 \leq cn^2 \quad n \geq n_0 = 7$$

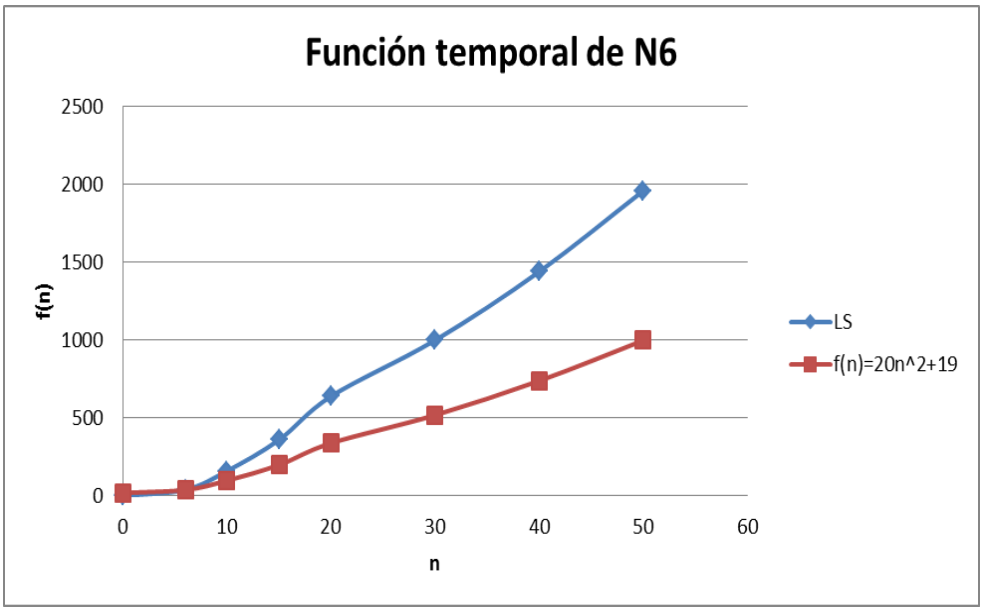

Figura B.4. Gráfica de la función temporal de la estructura N6

De acuerdo a la evaluación de la complejidad asintótica, para la estructura de vecindad N6, la complejidad es de *O* (n^2), con un valor para la constante *c* de 40, para un valor inicial de n_0 = 7.

GLOSARIO DE TÉRMINOS

Determinístico: Es un algoritmo en el que una misma entrada obtiene siempre el mismo resultado.

Algoritmo no Determinístico: Algoritmo en el que a una misma entrada se obtienen diferentes salidas, de modo que no es posible saber de manera previa, el resultado que de la ejecución de un algoritmo de este tipo.

Algoritmo Secuencial: Algoritmo programado de manera estructurada que al ser ejecutado utiliza un solo procesador.

Análisis de Sensibilidad: Evaluación del comportamiento de las variables críticas de un problema con la finalidad de establecer un rango numérico, dentro del cual, la solución obtenida por el algoritmo sigue siendo buena, además de que permite conocer que tan sensible es el algoritmo a cambios en los valores de ciertas variables propias del problema.

Benchmark: Palabra del inglés utilizada para nombrar a los problemas de prueba utilizados por diversos investigadores, para medir el rendimiento de un sistema o algoritmo para un problema dado.

Búsqueda Local: Técnica iterativa de que permite explorar el espacio de soluciones de un problema dado, a partir de una solución inicial, por medio de movimientos, de modo que vaya mejorando la solución obtenida de acuerdo a la función objetivo. Este tipo de técnicas son utilizadas para mejorar la calidad de las soluciones obtenidas por un algoritmo, así como para reducir el tiempo en que se obtienen dichas soluciones.

Búsqueda Tabú (TS): Técnica basada en la inteligencia artificial, empleando el concepto de memoria e implementándolo mediante estructuras simples, con el objetivo de dirigir la búsqueda de la solución final en función de los resultados alcanzados [Glover, 1989]. TS considera dos tipos de memoria que interaccionan entre sí, a corto y largo plazo respectivamente. Este tipo de datos darán lugar a estrategias de intensificación y/o diversificación dentro del ámbito local o global.

Complejidad Temporal: Es el número de pasos necesarios (tiempo) para obtener una solución a una instancia de un problema dado.

Estructura de Vecindad: Tipo de movimiento utilizado para explorar el espacio de soluciones de un problema dado.

Heurística: Procedimiento intuitivo bien definido que proporciona buenas soluciones aproximadas a problemas difíciles de resolver sin garantizar la optimalidad, en un tiempo computacional razonable.

Instancia*:* Es un problema de prueba que puede ser definido como el tamaño de entrada de los datos del problema.

Lista tabú: Contiene los movimientos que se realizan entre operaciones para encontrar una solución vecina, esta lista tiene la función de evitar que una solución sea visitada nuevamente en base a una repetición del movimiento.

Metaheurísticas: Métodos aproximados que mejoran procedimientos heurísticos, los cuales son diseñados para ser aplicados a problemas considerados difíciles de resolver, donde las heurísticas no son eficientes.

Óptimo Global: Es la mejor solución de un espacio de soluciones f(x).

Óptimo Local: Representa la mejor solución de f(x) en un entorno x.

Sintonización: Es la proporción adecuada en cuanto a los valores obtenidos mediante el análisis de sensibilidad aplicado a los parámetros de control, tomando en cuenta el problema y el método de optimización utilizado, de modo que el algoritmo muestre una mejora tanto en eficiencia como en eficacia.

Tiempo polinomial: En las ciencias computacionales, el tiempo viene expresado generalmente en función del tamaño de la entrada. Se considera que un algoritmo puede ser resuelto en tiempo polinomial sí su función de complejidad es de orden $O(p(n))$, es decir, que puede ser representado por un polinomio.

Vecindad: Es el conjunto de soluciones que pueden ser alcanzadas desde una solución dada por medio de un movimiento (inserción, eliminación, permutación).