6. ANALISIS DE RESULTADOS

6.1 EXPERIMENTOS

El método propuesto se validará teniendo en cuenta el balanceo de carga dinámico centralizado con el caso más genérico existente, en el que la asignación de las tareas por parte del nodo maestro a los nodos esclavos se hace de manera aleatoria como el método que propone [BEVILACQUA, 1999] en su trabajo. Una vez que cada esclavo ejecuta la tarea que le correspondió enviará una confirmación de terminación de tarea para que el maestro asigne una nueva tarea, hasta efectuar todas las listadas en el arreglo de tareas.

Por otro lado también se validará el método propuesto con una modificación del método de Choi, en el que se propone una métrica de carga basada en número de tareas afectando así el desempeño del sistema en cuanto a tiempo de ejecución. [CHOI, 2003]. En esta investigación este método se denominará método uniforme.

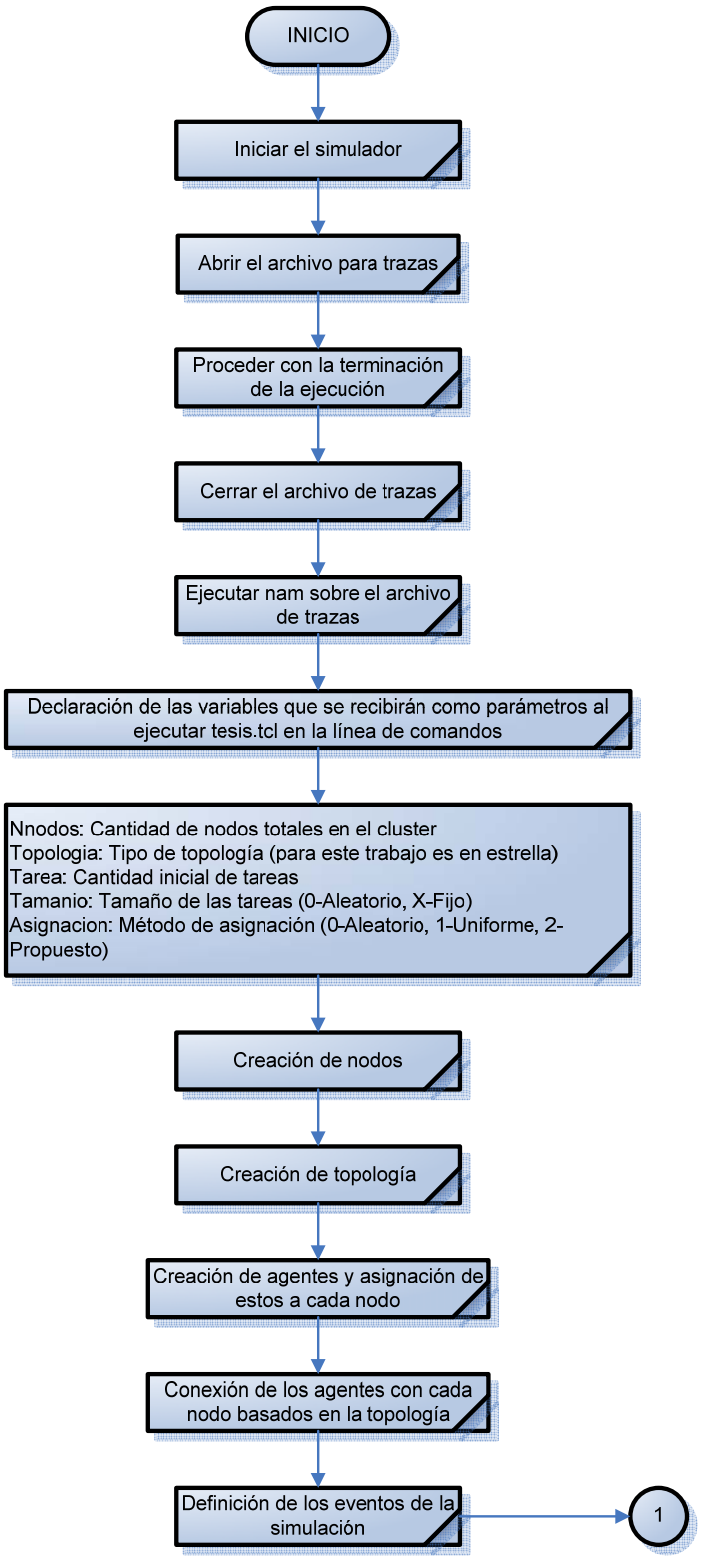
Los tres (3) métodos fueron programados en C++, integrados al simulador Network Simulator 2 (NS-2), sin embargo existen algunas variaciones en los métodos de Bevilacqua y de Choi.

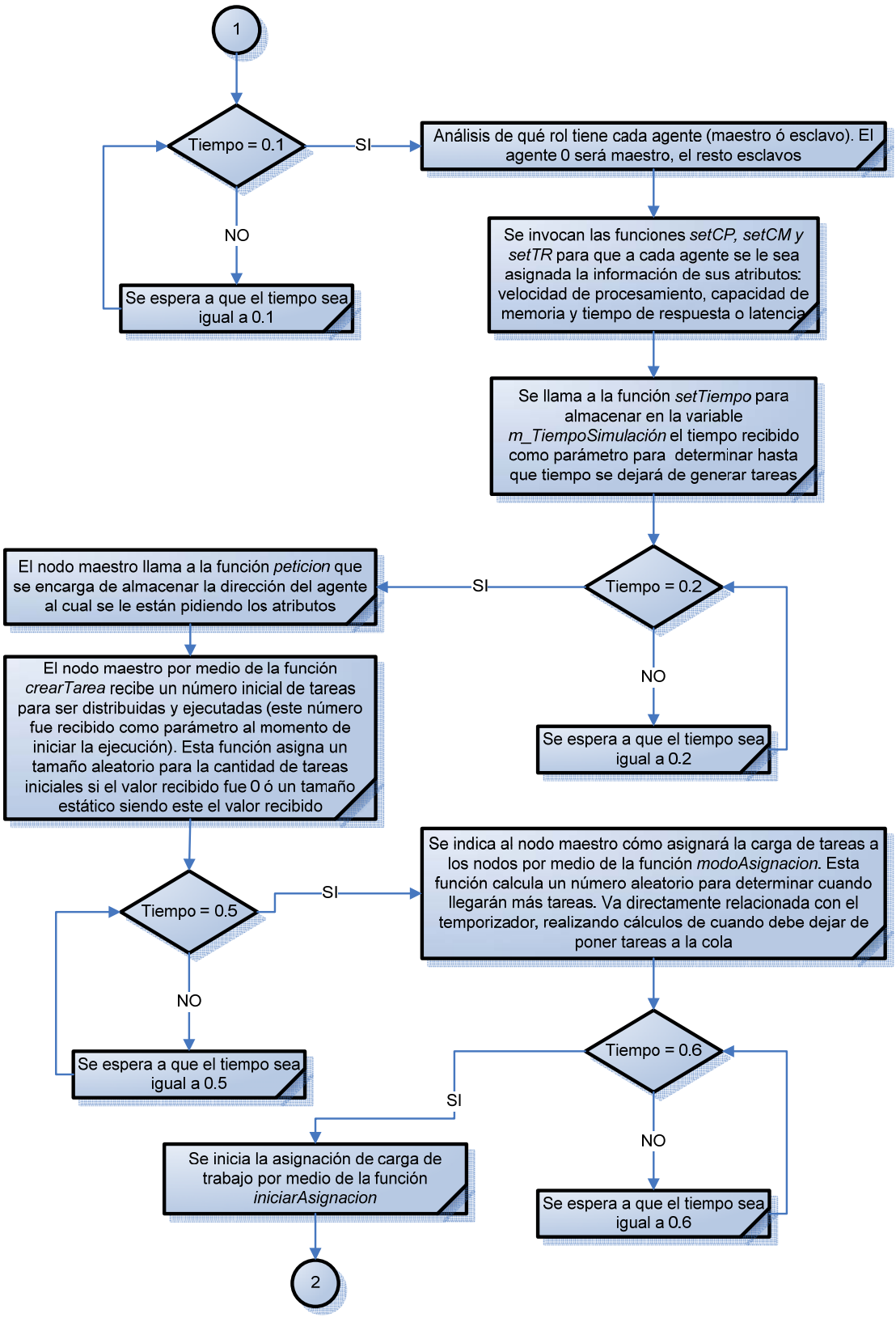
Para mayor información sobre el Network Simulator-2 se puede revisar el anexo 9.6 o visitar la página siguiente: http://www.isi.edu/nsnam/ns/

El de Bevilacqua es denominado en este trabajo método aleatorio y varía en que el nodo maestro nunca realiza trabajo de ejecución, simplemente se encarga de asignar las tareas. Una vez que cada nodo esclavo culmine con la ejecución de la tarea que se encuentra realizando envía al nodo maestro una notificación de terminación de tarea y de petición de tareas nuevas en caso de que existan.
Para el trabajo de Choi y compañía el método como se mencionó anteriormente es denominado método uniforme y se encarga de distribuir uniformemente las tareas en cuanto a tamaños a los nodos esclavos sin tener muy en cuenta la capacidad que tenga cada uno de estos. La variación radica principalmente en el hecho de que procura que cada nodo esclavo ejecute una cantidad de carga similar sin importar la capacidad real de cada nodo esclavo.

A continuación se hará una explicación del algoritmo por medio de Diagramas de flujo, en donde inicialmente se presentará la ilustración de la secuencia del algoritmo en TCL integrado con C++ (figura 6), continuando con la descripción detallada de los dos métodos de prueba: aleatorio y uniforme (figuras 7 y 8 respectivamente) y el método propuesto (figura 9).

SECUENCIA DEL ALGORITMO EN TCL INTEGRADO CON C++




1
Tiempo = 0.1
SI
NO
Se espera a que el tiempo sea igual a 0.1
Análisis de qué rol tiene cada agente (maestro ó esclavo). El agente 0 será maestro, el resto esclavos
Se invocan las funciones *setCP, setCM y setTR* para que a cada agente se le sea asignada la información de sus atributos: velocidad de procesamiento, capacidad de memoria y tiempo de respuesta o latencia
Se llama a la función *setTiempo* para almacenar en la variable *m_TiempoSimulación* el tiempo recibido como parámetro para determinar hasta que tiempo se dejará de generar tareas
Tiempo = 0.2
SI
NO
Se espera a que el tiempo sea igual a 0.2
El nodo maestro llama a la función *peticion* que se encarga de almacenar la dirección del agente al cual se le están pidiendo los atributos
El nodo maestro por medio de la función *crearTarea* recibe un número inicial de tareas para ser distribuidas y ejecutadas (este número fue recibido como parámetro al momento de iniciar la ejecución). Esta función asigna un tamaño aleatorio para la cantidad de tareas iniciales si el valor recibido fue 0 ó un tamaño estático siendo este el valor recibido
Tiempo = 0.5
SI
NO
Se espera a que el tiempo sea igual a 0.5
Se indica al nodo maestro cómo asignará la carga de tareas a los nodos por medio de la función *modoAsignacion*. Esta función calcula un número aleatorio para determinar cuando llegarán más tareas. Va directamente relacionada con el temporizador, realizando cálculos de cuando debe dejar de poner tareas a la cola
Tiempo = 0.6
SI
NO
Se espera a que el tiempo sea igual a 0.6
Se inicia la asignación de carga de trabajo por medio de la función *iniciarAsignacion*
2

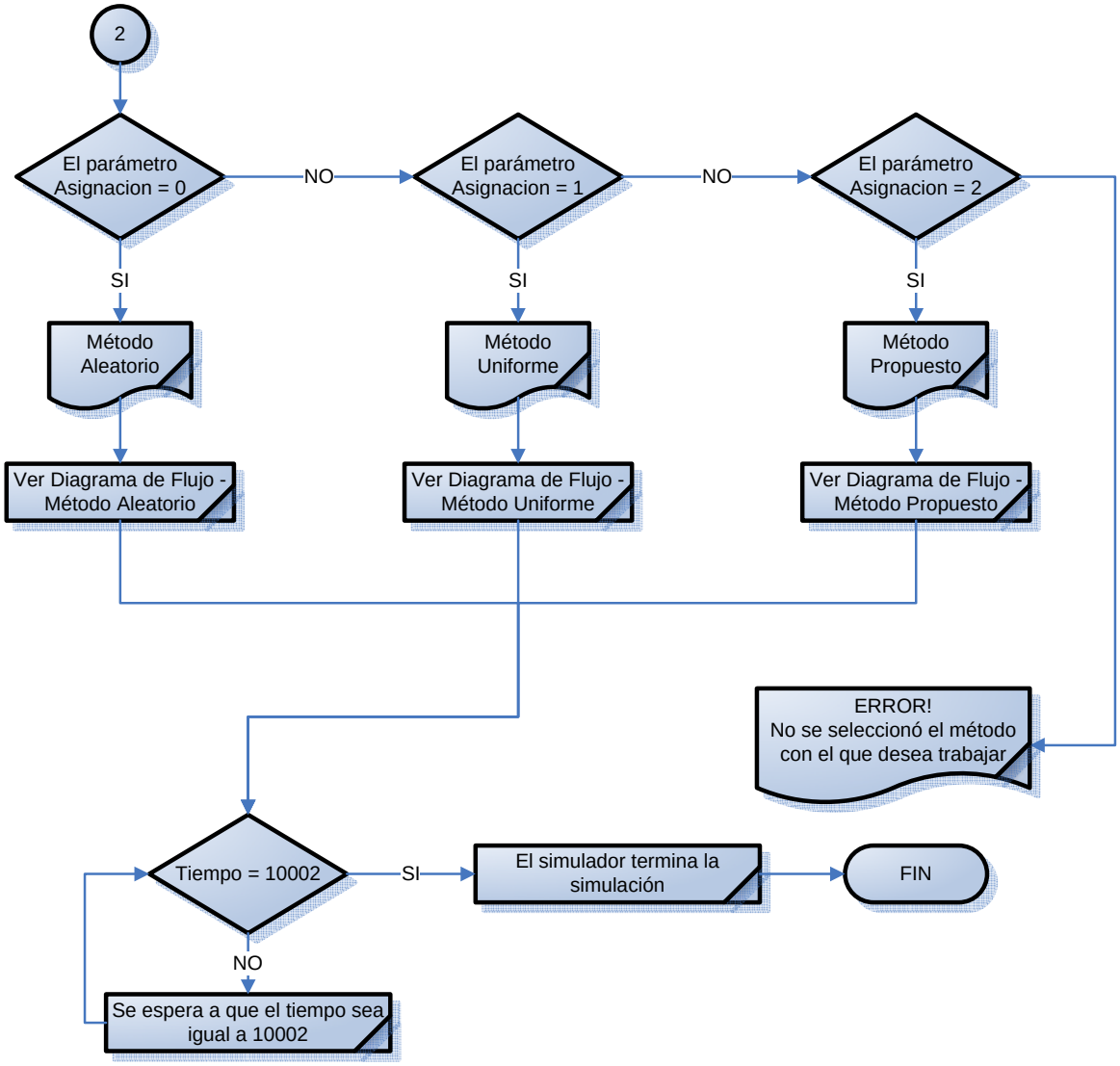


Figura 6. Diagrama de flujo - Secuencia del Algoritmo en TCL Integrado con C++

MÉTODO ALEATORIO

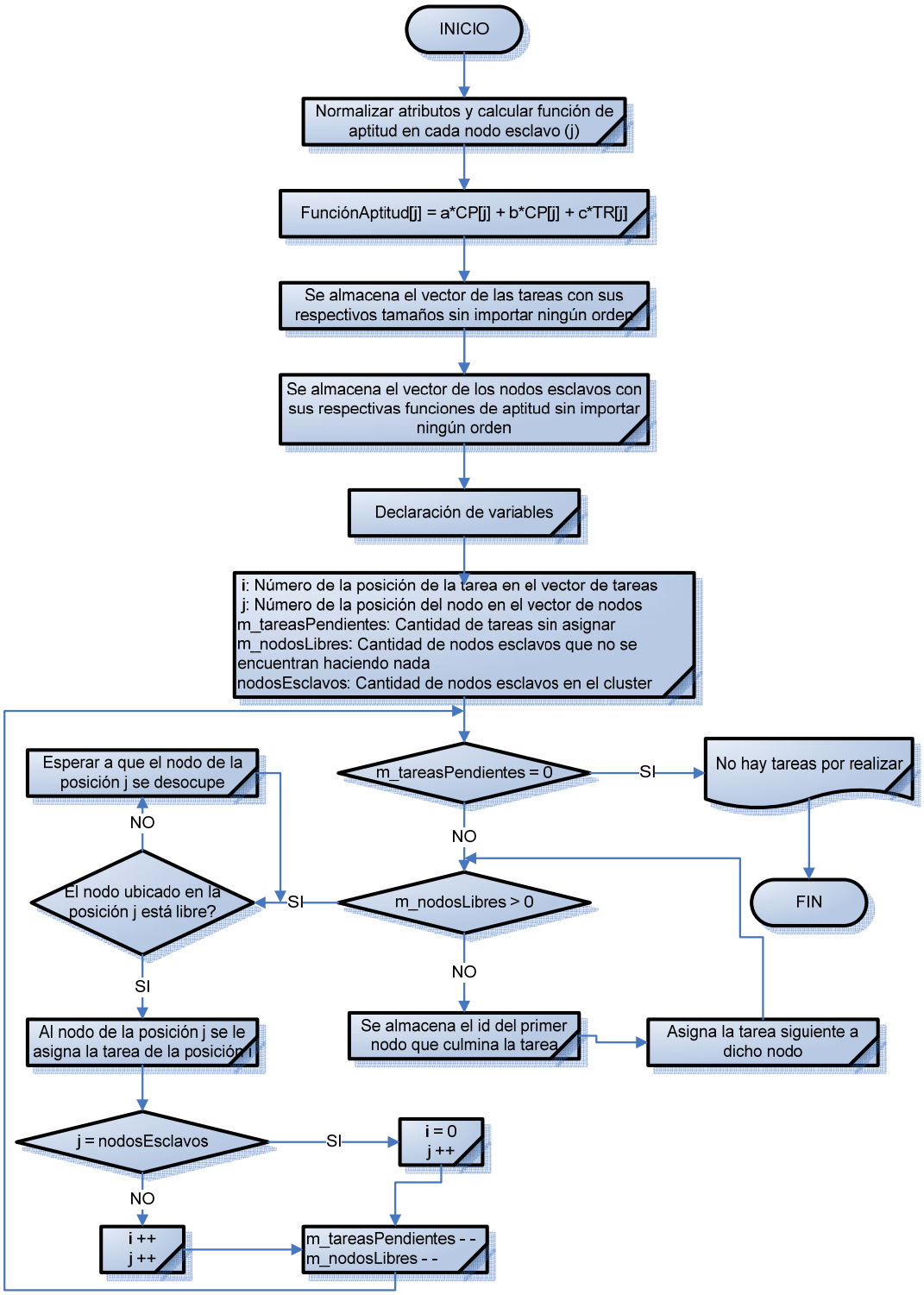
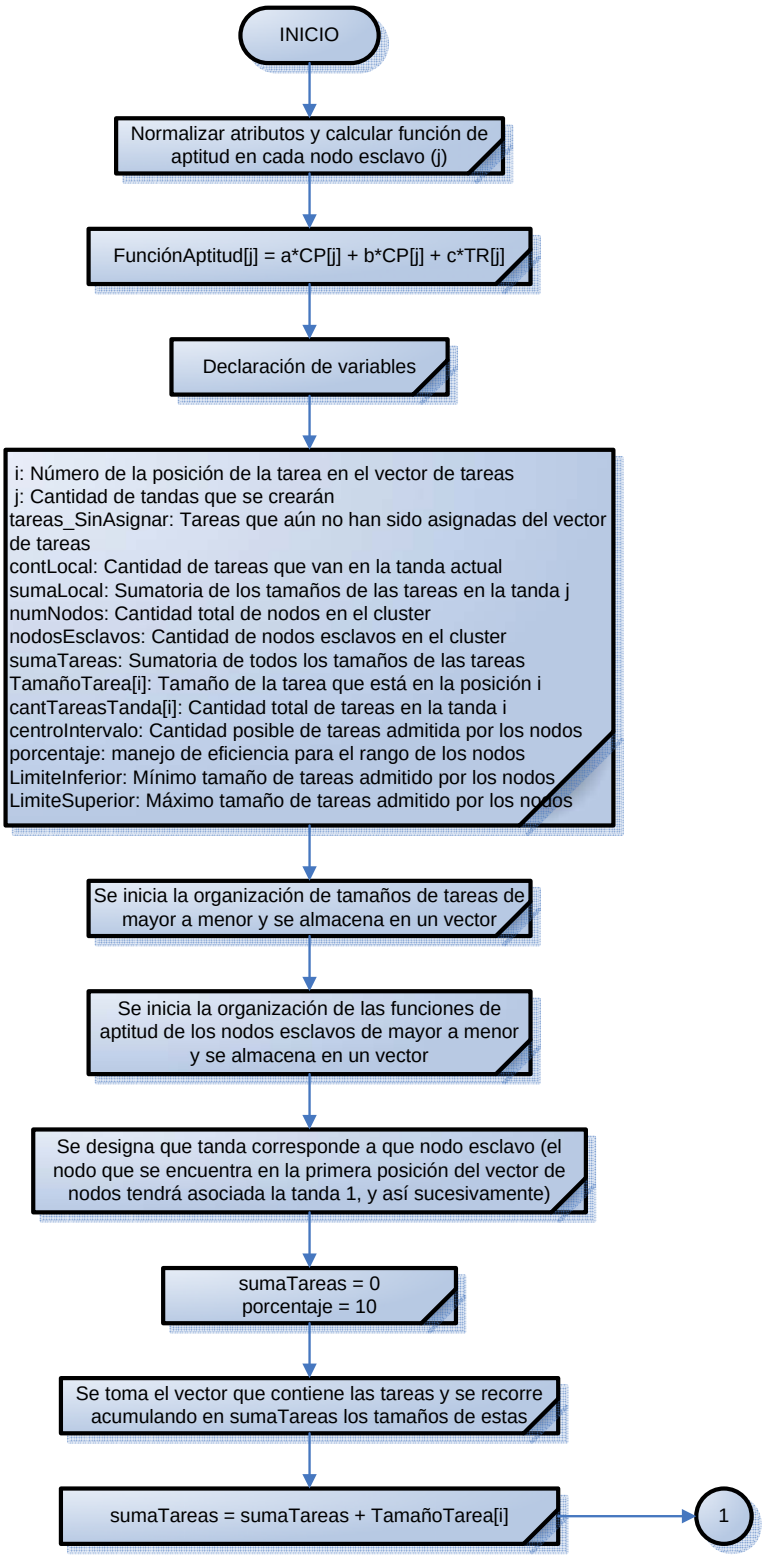
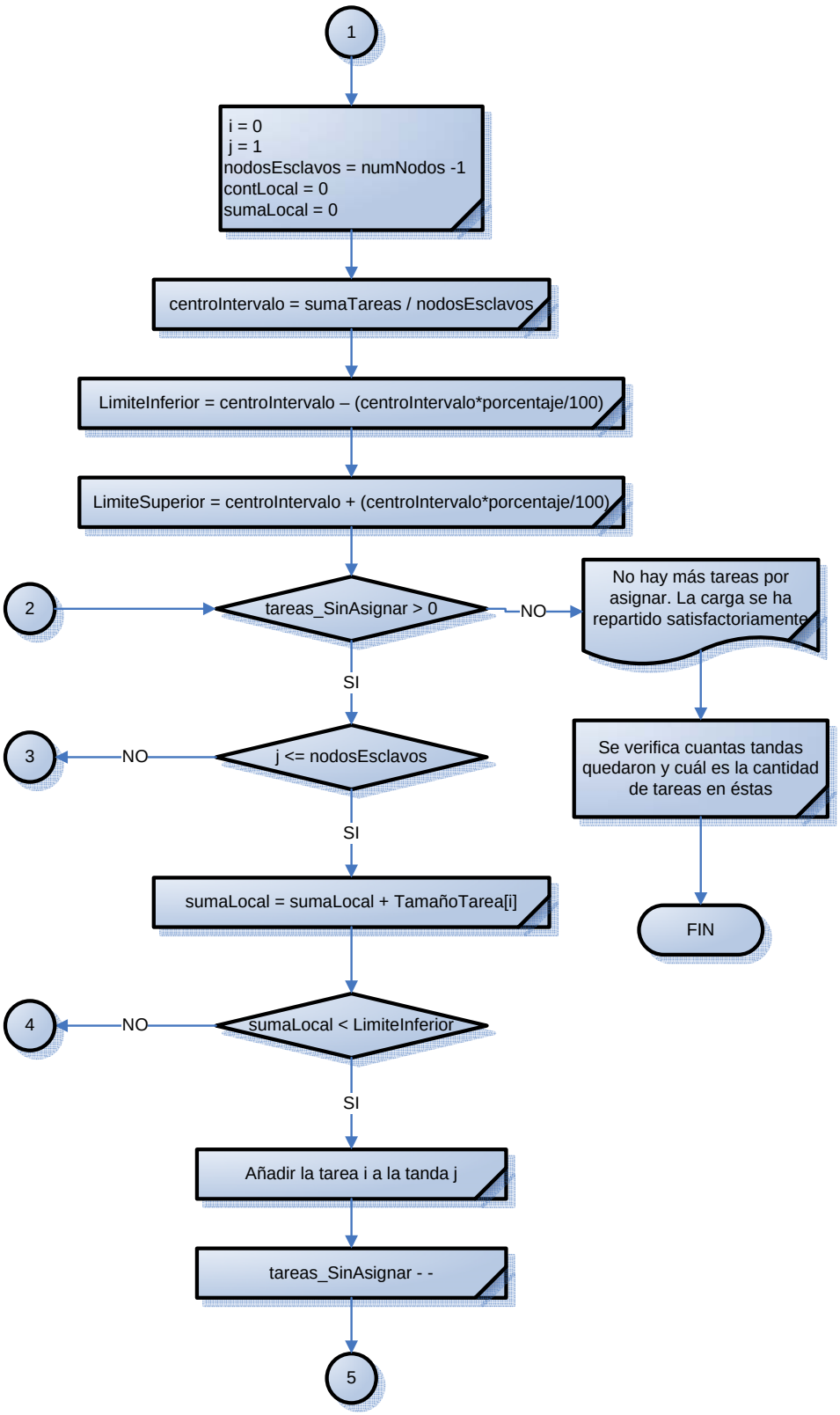


Figura 7. Diagrama de flujo – Método Aleatorio

MÉTODO UNIFORME




1
i = 0
j = 1
nodosEsclavos = numNodos -1
contLocal = 0
sumaLocal = 0
centroIntervalo = sumaTareas / nodosEsclavos
LimiteInferior = centroIntervalo – (centroIntervalo*porcentaje/100)
LimiteSuperior = centroIntervalo + (centroIntervalo*porcentaje/100)
2
tareas_SinAsignar > 0
NO
No hay más tareas por asignar. La carga se ha repartido satisfactoriamente
SI
3
NO
j <= nodosEsclavos
Se verifica cuantas tandas quedaron y cuál es la cantidad de tareas en éstas
SI
sumaLocal = sumaLocal + TamañoTarea[i]
FIN
4
NO
sumaLocal < LimiteInferior
SI
Añadir la tarea i a la tanda j
tareas_SinAsignar - -
5