ISSN 1870-4069

Simulador con heurísticas de empaquetamiento para la asignación de memoria y despacho de procesos

Hilda Castillo Zacatelco, Rafael de la Rosa Flores, Claudia Zepeda Cortés, Ana Patricia Cervantes Márquez, Esly Cuautle Aguilar

Benemérita Universidad Autónoma de Puebla, Puebla, México
{hildacz,rafa.elo31,czepedac,cervantes.patty,shyzumae}@gmail.com

Resumen. En este trabajo se persiguen varios objetivos: primero, enseñar a los estudiantes de la FCC de la BUAP la importancia de los algoritmos heurísticos de empaquetamiento; segundo, que comprendan las tareas involucradas en la asignación de memoria utilizando listas ligadas, así como el despacho de procesos que se lleva a cabo dentro de un Sistema Operativo; y por último, el desarrollo y la implementación de un simulador. El cual implementa algunos de los algoritmos heurísticos más usados de acuerdo al estado del arte revisado. El simulador consta de tres módulos principales: primero, un asignador de memoria;segundo, un despachador de procesos y por último un módulo de manejo de memoria virtual. En cada uno de los módulo que se presentan en esta propuesta es posible seleccionar entre diferentes técnicas de asignación de memoria y/o despacho de procesos, lo cual permite realizar reconfiguraciones del simulador para establecer comparativas entre los diferentes algoritmos heurísticos.

Palabras clave: heurísticas, FF, NF, BF, WF, asignación de memoria, despacho de procesos, simulador.

Simulator with Packaging Heuristics for Memory Allocation and Process Scheduling

Abstract. In this job, several objectives are pursued: first, to teach the students of the FCC of the BUAP the importance of packaging heuristic algorithms; second, that they understand the tasks involved in memory allocation using linked lists, as well as the dispatch of processes that takes place within an operating system; and by last, the development and implementation of a simulator. Which implements some of the most used heuristic algorithms according to the state of the art reviewed. The simulator

consists of three main modules: first, a memory allocator, second, a process dispatcher and by last a virtual memory management module. In each of the modules presented in this proposal it is possible to select between different memory allocation techniques and / or dispatch of processes, which allows reconfigurations of the simulator to establish comparisons between the different heuristic algorithms.

Keywords: heuristic, FF, NF, BF, WF, memory allocation, process scheduling, simulator.

1. Introducción

El problema de empaquetamiento ha sido ampliamente estudiado desde la década de los 70's y aún sigue siendo de suma importancia debido al crecimiento de la tecnología, prueba de ello es el problema de la asignación de máquinas virtuales en infraestructuras IaaS (Infraestructura como servicio) [1], el cual se puede resolver utilizando técnicas para solucionar el problema de bin packing. Por lo que se puede decir que su uso sigue vigente y abierto a mejoras.

Muchas de estas técnicas caen dentro del área de la Inteligencia Artificial (IA), que como parte de sus estrategias de solución utiliza algoritmos metaheurísticos para resolver problemas. Las metaheurísticas generalmente involucran una o varias heurísticas en sus algoritmos. Las heurísticas son algoritmos que obtienen buenas soluciones pero no demuestran si la solución es óptima, pueden ir desde muy simples a muy complejas [2]. Dentro de las simples y sobre las cuales se basan muchas otras heurísticas y metaheurísticas, se encuentran los algoritmos primer ajuste (FF, First Fit), siguiente ajuste (NF, Next Fit), mejor ajuste (BF, Best Fit) [3] y peor ajuste (WF, Worst Fit).

Estas heurísticas son utilizadas principalmente para resolver problemas de empaquetamiento, como el problema de bin packing (BPP), el de cutting stock (CSP), el de scheduling entre otros. Estos problemas pueden ser mapeados a problemas de la vida real. Éstas heurísticas a pesar de su simplicidad han demostrado dar buenos resultados.

Por otra parte, la enseñanza de los conceptos de sistemas operativos es un tema fundamental para los alumnos de Ciencias de la Computación. Su importancia radica en que un sistema operativo es un administrador de recursos [4] que utiliza diferentes algoritmos para mejorar su desempeño. Muchas de estas técnicas están basadas en algoritmos de la inteligencia artificial.

Este trabajo tiene tres objetivos importantes: primero, que los estudiantes reconozcan la importancia de las heurísticas de empaquetamiento y su aplicación en la solución de problemas reales; segundo, que comprendan las tareas que conlleva la asignación de memoria en Sistemas Operativos y el despacho de procesos, y los algoritmos que se pueden utilizar en cada caso; y por último, construir un simulador para el curso de Sistemas Operativos.

El simulador consiste de tres módulos: un asignador de memoria, un despachador de procesos y un módulo para el manejo de memoria virtual. El asignador de memoria es a su vez un conjunto de módulos que implementa una heurística de empaquetamiento para la asignación de memoria; en el caso del despachador, también son varios módulos que implementan diferentes técnicas de despacho y lo mismo con el módulo de manejo de memoria virtual en donde se implementan algoritmos de reemplazo de páginas.

Cada uno de estos módulos debe permitir utilizar diferentes algoritmos para configurarse e intercambiarse de diferentes formas para evaluar el desempeño de las diferentes combinaciones, de esta manera se pueden realizar comparaciones o incluso mejoras a los algoritmos.

De acuerdo a la experiencia de varios años en la impartición del curso de Sistemas Operativos II de la carrera de Ciencias de la Computación de la BUAP, se ha observado que la mejor forma de aprender es practicando, sin embargo, son tantos los algoritmos que en un curso de 16 semanas es casi imposible revisar a detalle cada uno de ellos. Por lo que la construcción de un simulador como herramienta de apoyo para este curso, se realizará por etapas.

El objetivo final es construir un simulador con módulos bien documentados que pueda utilizarse en los cursos de Sistemas Operativos en donde los alumnos puedan manipular el código para realizar cambios o mejoras.

En este trabajo se presenta el resultado de la primera etapa en la construcción del simulador. Por el momento se cuenta con el módulo de asignación de memoria que cuenta con la implementación de las heurísticas de empaquetamiento FF, BF, NF, WF y se agregó una heurística denominada ajuste rápido. El módulo de despacho de procesos implementa únicamente el algoritmo de Round Robin y esta en construcción el módulo de manejo de memoria virtual en donde se implementarán diferentes algoritmos de reemplazo de páginas.

Los alumnos implementaron estos módulos y compararon el desempeño de las diferente heurísticas, con una entrada de 2000 procesos. Los resultados muestran que con la implementación de estos módulos, los alumnos lograron explicar los conceptos involucrados en estos temas.

El documento está dividido en seis secciones. En la segunda sección se mencionan los trabajos relacionados y la importancia de las heurísticas de empaquetamiento. En la tercera sección se describe la arquitectura del simulador de asignación de memoria y despacho de procesos. En la cuarta sección se describe el funcionamiento de las heurísticas de empaquetamiento FF, BF, NF, WF y ajuste rápido. La quinta sección presenta los resultados en cuanto a la implementación del simulador y el desempeño de los estudiantes después de haber implementado el simulador. En la sexta sección se presentan las conclusiones y el trabajo a futuro.

2. Importancia de las heurísticas de empaquetamiento y trabajos relacionados

En los trabajos del estado del arte en donde se proponen estrategias para solucionar problemas de empaquetamiento y corte como el problema de bin packing, cutting stock o scheduling, se hace uso fundamentalmente de heurísticas.

El tema de asignación de memoria y despacho de procesos, son dos de los temas más interesantes y en muchas ocasiones difíciles de comprender para los estudiantes, por lo que muchos trabajos están enfocados en proponer estrategias o herramientas para la enseñanza de sistemas operativos, principalmente en estos temas. A continuación se mencionan algunos trabajos cuyo objetivo es la enseñanza de los sistemas operativos.

En [18] se presenta una página que describe de forma breve las técnicas de administración de memoria conforme éstas fueron evolucionando, desde particiones fijas, pasando por particiones variables hasta memoria virtual. Es un documento estático con texto plano e imágenes. En [5] Buendía et al. presentan un trabajo en donde hicieron uso de simuladores para la enseñanza de sistemas operativos, en éste concluyen que el uso de herramientas para tal fin requiere de un gran esfuerzo para crearlas por lo que el profesor al utilizar este tipo de herramientas debe cambiar su estrategia de enseñanza y su forma de evaluar.

Desde la década de los 80ś, se ha visto la necesidad de emplear herramientas que le permitan al alumno comprender las tareas de un sistema operativo, tal y como lo muestra el trabajo de Santana y Santana [6] en 1987.

En [7] se presenta un trabajo para la enseñanza de sistemas operativos mediante un simulador que contiene componentes básicos de éste programa maestro, el cual es entregado a los estudiantes para que construyen en él algunos algoritmos de sistemas operativos. En [8] se presenta un trabajo en donde se desarrolló un software que permite simular el manejo de memoria del sistema operativo MAC OS.

Dado que los estudiantes tienen diferentes formas de aprender, es importante incorporar técnicas en el salón de clases que les facilite el aprendizaje de los diferentes temas del curso, tal es el caso de elementos lúdicos. En [9] Hil et. al. presentan un trabajo que utiliza rompecabezas y juegos para la enseñanza de sistemas operativos. Por otro lado, mientras algunos utilizan juegos o simuladores, otros trabajos utilizan sistemas operativos reales para que los estudiantes aprendan procesos, la forma en la cual los estudiantes se comunican con el sistema operativo es a través de una aplicación web, que envía las solicitudes de procesos al sistema operativo, los ejecuta y retorna resultados y retroalimentación [10], esta metodología, de acuerdo a sus autores, es adecuada para cursos de sistemas operativos a distancia.

En cuanto a la importancia del uso de heurísticas en la solución de problemas reales, se tiene que a la fecha se han desarrollados trabajos interesantes, que demuestran la importancia de los algoritmos heurísticos FF, BF, NF, tanto en su uso como en su implementación. A continuación se mencionan algunos de los más relevantes.

En un trabajo reciente en Ahmed [11] se argumenta que la asignación de recursos (procesos, uso de la memoria y migración de procesos) sobre una granja de servidores en la nube hace que se aumente la carga de trabajo y por consiguiente el consumo de energía. Para resolverlo proponen el uso de dos algoritmos; uno heurístico y el otro una metaheurística, FF y colonia de hormigas, respectivamente. El objetivo es eficientar la carga de trabajo en la nube para reducir el consumo de energía y reducir los niveles de CO2. En este mismo sentido, Kumar [12], propone el uso de las heurísticas FF, NF, BF, entre otras, ellos tratan de solucionar el problema de virtualización de hardware en la nube, conocido como IaaS, los servicios que se proporcionan son recursos de hardware tales como: memoria, almacenamiento, tiempo de CPU, ancho de banda, entre otros. Para ello, propone utilizar el conocido problema del BPP, donde cada uno de los servidores IaaS es un contenedor y las máquinas virtuales son objetos que se desea colocar en los contenedores y de esta forma reducir el consumo de energía. Así como también, en [13] se hace referencia a la problemática de encontrar una solución óptima al asignamiento de máquinas virtuales en una plataforma IaaS. Para ello, plantean que se puede resolver utilizando el problema del BPP. En este trabajo se hace hincapié en que a la fecha no existen trabajos serios que permitan evaluar de manera formal cuál algoritmo es mejor en términos de rendimiento. Para ello proponen una metodología que compara siete algoritmos heurísticos basados en *best fit* para resolverla y utilizan el simulador CloudSim [14], el cual realiza la asignación de máquinas virtuales a máquinas físicas.

Otro trabajo interesante, es el que propone Bein [1], donde enfatizan el uso del almacenamiento de la información en los centros de datos, así como la importancia del problema del BPP para resolverlo. Para ello, realizan un panorama de las heurísticas FF y BF aplicadas al BPP y proponen un algoritmo heurístico, llamado K-binary, éste lo comparan con los algoritmos propuestos por Lee [15] y Seiden [16] y con base en sus resultados experimentales muestran que su propuesta tiene un mejor desempeño.

3. Arquitectura del simulador

El simulador propuesto consta de tres módulos principales: el que lleva a cabo la asignación de memoria, el despachador de procesos y el módulo para el manejo de memoria virtual. La entrada de datos esta dado por un archivo de texto que contiene la información necesaria de cada proceso. En este caso, cada proceso tiene asociado un nombre de proceso, el número de bloques de memoria que necesita y su tiempo de ejecución. En la Figura 1 se muestra la arquitectura del simulador y la forma en la cual intectúan los módulos. Las partes sombreadas son los módulos que aún están en etapa de construcción.

Además de los módulos principales, el simulador consta de módulos auxiliares, tal es el caso del módulo que permite la lectura de los datos de procesos por lotes y el módulo de liberación de memoria. A continuación se describe el funcionamiento general de cada módulo y su interacción con los demás módulos.

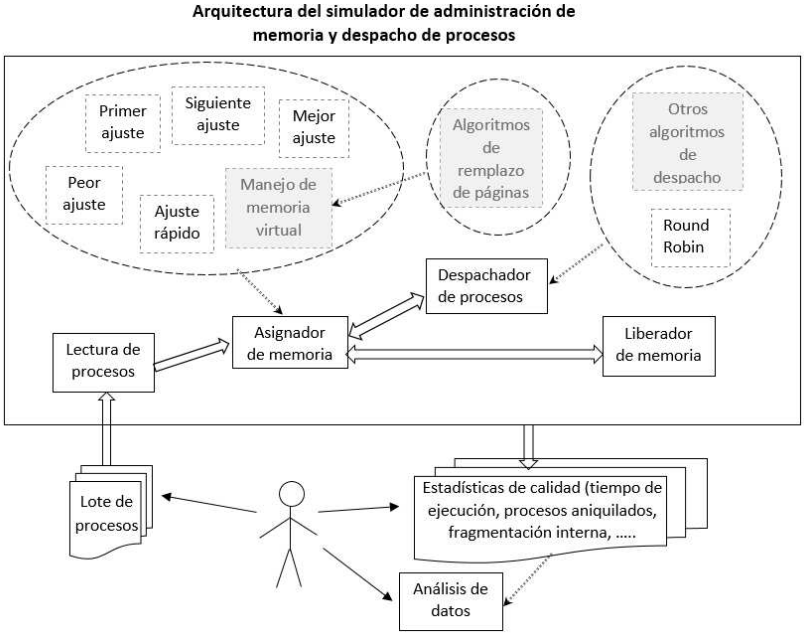

Fig. 1. Arquitectura del simulador en donde se muestran los diferentes módulos que lo integran y la interacción entre ellos.

Cada cierto tiempo aleatorio, el módulo *Lectura de procesos* lee un cierto número aleatorio de procesos y los envía al módulo *Asignador de memoria*. Éste módulo, puede elegir entre diferentes algoritmos para realizar la asignación de memoria, cada algoritmo se encuentra implementado en módulos diferentes que pueden integrarse al sistema. Manualmente se elige el módulo de la heurística o técnica de asignación de memoria que se quiere utilizar y se integra al simulador, esto es, se insertan las instrucciones para invocar a un determinado módulo y se compila el simulador. Lo mismo sucede con el módulo *Despachador de procesos* y el módulo de *Manejo de memoria virtual*, selecciona alguno de los algoritmos implementados y lo integran al simulador.

Una vez que todo esta listo, el simulador inicia su ejecución, lee por lotes los procesos y entonces cada vez que un proceso llega al simulador, el *Asignador de memoria* asigna los bloques o páginas necesarias al proceso. Si el proceso no encuentra espacio en la memoria, entonces es aniquilado y no se ejecuta. Una vez que un proceso esta cargado en memoria se envía su información al módulo *Despachador de procesos*, quien es el encargado de elegir al siguiente proceso en ser ejecutado. Éste módulo, hace uso de una estructura de datos especial para realizar su trabajo. Esta serie de pasos se llevan a cabo con cada proceso.

Para el caso de las heurísticas de empaquetamiento, utilizan una lista ligada como estructura de datos, para almacenar la información necesaria de los procesos.

El módulo *Liberador de memoria* se encarga de liberar los bloques o las páginas de los procesos que han finalizado su ejecución. En el caso de las heurísticas FF, BF, NF y WF, como se esta utilizando una lista ligada que mapea exactamente el estado de la memoria, entonces se debe revisar si el nodo a liberar tiene huecos adyacentes a la izquierda, a la derecha o a ambos lados, en caso afirmativo se deben fusionar los nodos, en cada contrario, únicamente se cambia el nombre del proceso por un nombre que indique que se trata de un hueco. El módulo *Liberador de memoria* es común para las técnicas FF, NF, BF y WF, en el caso de Ajuste rápido y de *Manejo de memoria virtual* la liberación de la memoria se ajusta a cada caso. El módulo *Despachador de procesos* utiliza una estructura de datos propia para realizar su tarea.

Todo el tiempo el simulador esta generando estadísticas con respecto al tiempo de espera de cada proceso, el tiempo que tardo un proceso en el sistema desde que ingresa hasta que es finalizado, la fragmentación interna que se genera, el número de procesos que son aniquilados. Con estos datos se puede realizar una comparación en cuanto al desempeño de cada heurística y se puede conocer el tipo de variables de las que dependen los resultados. Los estudiantes por el momento realizan el análisis de los datos apoyandose en herramientas estadísticas y dependiendo de los resultados emiten sus conclusiones.

4. Funcionamiento de las heurísticas de empaquetamiento

En esta sección se describirá el funcionamiento de cada una de las heurísticas que fueron implementadas para la asignación de memoria. Además de las heurísticas FF, NF, BF y WF, se implementó una heurística más denominada Ajuste Rápido.

Como la asignación de memoria con listas ligadas ya no es empleada en los sistemas operativos modernos, es frecuente que el estudiante se tope con el problema de no encontrar información que le permita entender a detalle como funcionan estas heurísticas para la asignación de memoria, ya que sus búsqueda se limita a los libros de sistemas operativos. Por ejemplo en [4] se describe de forma escrita el funcionamiento y en [17] no se hace mención a ellas. Dado lo anterior, en esta sección se describe el funcionamiento de forma gráfica tal y como los estudiantes del curso de sistemas operativos las implementaron.

Estas heurísticas utilizan como estructura de datos una lista ligada en donde cada nodo tiene tres campos: el primer campo contiene el nombre del proceso o H en el caso de que sea un hueco, es decir, una cierta cantidad de bloques que están libres; el segundo campo contiene el índice donde comienza el hueco o el proceso, y el tercer campo contiene el número de bloques de que consta el hueco o proceso. En la Figura 2 se muestra un ejemplo de como se pueden mapear los bloques de la memoria a una lista ligada.

En este caso los procesos deben ser asignados sin dividirlos en bloques, aunque la memoria sí está dividida en bloques de tamaño fijo. Al inicio se supone que la memoria está vacía, es decir, no existe ningún proceso asignado, por lo que la lista ligada consiste de un nodo. Suponga que se tiene una memoria de 1Mb

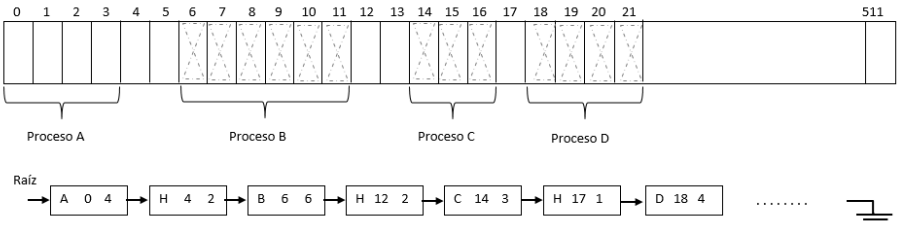

Fig. 2. Representación de procesos en la memoria utilizando listas ligadas.

dividida en bloques de 1Kb entonces al inicio se tendrá una lista ligada de un nodo con identificador H (indica "hueco"), bloque inicial 0 y número de bloques igual a 1024. Cada vez que un proceso entra al sistema, el administrador de memoria asigna el número de bloques necesarios al proceso, los bloques deben ser contiguos.

Cuando los procesos van entrando se les debe asignar memoria, es ahí donde se pueden emplear los algoritmos FF, NF, BF, WF y Ajuste Rápido. Cada una de éstas heurísticas toman los datos de entrada en cualquier orden y en lotes de procesos.

4.1. Primer ajuste (First fit)

El algoritmo inicia la búsqueda al inicio de la lista, si encuentra un hueco donde pueda colocar al proceso lo coloca, en otro caso, avanza hasta encontrar un hueco donde pueda colocar al proceso. Si el hueco es ás grande que el proceso, entonces ese nodo se divide en dos, en el primer nodo se coloca la información del proceso (nombre, número de bloque donde inicia el proceso, número de bloques que ocupa el proceso), en el segundo bloque se coloca la información de los bloques que no fueron asignados (identificador que indica que es un hueco, número de bloque donde inicia el hueco, número de bloques en que consiste el hueco). Si el hueco es exacto para el proceso, entonces únicamente se cambia el identificador de hueco por el nombre del proceso. Si se recorre toda la lista y no se encuentra hueco donde pueda colocarse el proceso, entonces el proceso es aniquilado y no se acepta en el sistema. La Figura 3 muestra los pasos para asignar memoria a tres procesos.

Los problemas que enfrenta este algoritmo es que siempre empieza en la cabeza de la lista ligada que representa la memoria; entonces en el peor de los casos, teniendo una lista ligada de una cantidad de nodos inmensa en la cual ningún hueco, excepto la cola de la lista, pueda albergar al nuevo proceso, recorrerá la lista completamente.

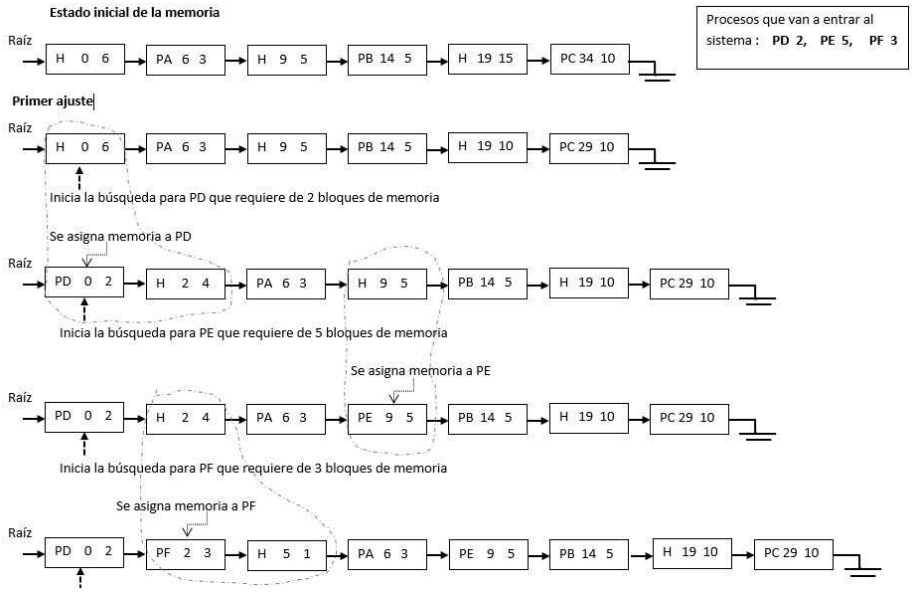

Fig. 3. Funcionamiento de la heurística primer ajuste.

4.2. Siguiente ajuste (Next fit)

Este algoritmo funciona de la misma manera que el primer ajuste cuando llega el primer proceso, esto es, inicia la búsqueda del hueco a partir de la raíz de la lista, cuando lo encuentra inserta el proceso en ese sitio y almacena el apuntador en ese sitio, para que cuando llegue un nuevo proceso, la búsqueda la inicie a partir de ahí, y así sucesivamente. Al igual que el primer ajuste, si no existe hueco para el proceso, éste será aniquilado. Los problemas que enfrenta son similares a los del primer ajuste. La Figura 4 muestra la misma entrada de procesos que para FF, pero ahora utilizando NF.

4.3. Mejor ajuste (Best fit)

Debido al problema que presentan los algoritmos anteriores en cuanto a no discriminar el hueco de memoria para asignar un nuevo proceso se crea este algoritmo, su funcionamiento es que revisará la lista ligada en búsqueda del hueco que mejor le ajuste al nuevo proceso, evitando así utilizar bloques demasiado grandes de memoria que podrían usar los siguientes procesos. Los problemas que presenta este algoritmo son dos, el primero es que debido a que tiene que buscar el mejor hueco para el proceso, esto significa recorrer toda la lista, lo que podría ocasionar un bajo desempeño en el proceso de asignación. El segundo problema es que si el hueco que más se ajusta al nuevo proceso, sigue siendo mayor se dejará un hueco de tamaño pequeño, lo puede ocasionar fragmentación

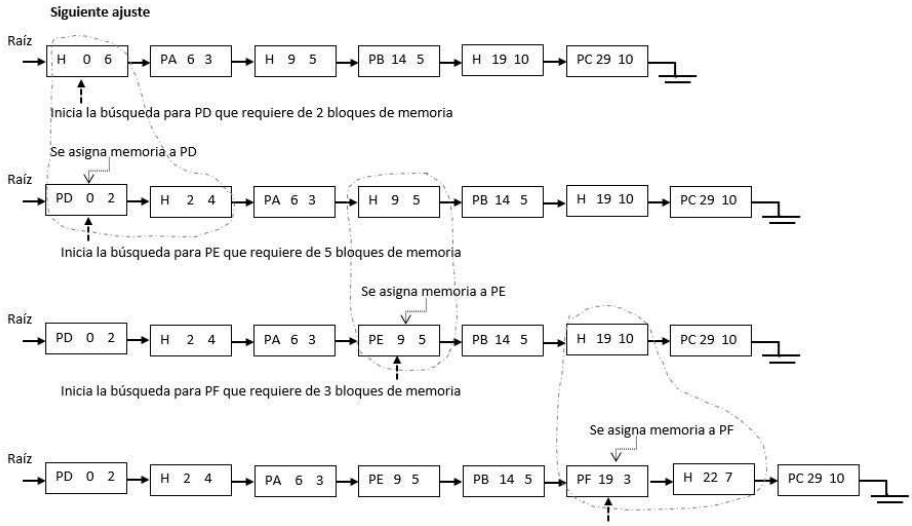

Fig. 4. Funcionamiento de la heurística siguiente ajuste.

externa. En la Figura 5 se muestra el mismo ejemplo que para FF pero ahora con BF.

4.4. Peor ajuste (Worst fit)

Con la finalidad de evitar dejar huecos muy pequeños, se creó el algoritmo del peor ajuste. Al contrario del mejor ajuste, el peor ajuste busca el hueco de memoria más grande con respecto al tamaño del nuevo proceso, la lógica detrás de esto, es que al elegir el hueco más grande, el hueco resultante al asignarle un nuevo proceso tendrá espacio suficiente para albergar a otro y habrá un menor desperdicio de memoria. La Figura 6, otra vez con el ejemplo de FF pero ahora con WF.

4.5. Ajuste rápido

Este algoritmo fue creado para ajustar de manera rápida un nuevo proceso a un bloque que tenga un tamaño similar al del proceso. En este algoritmo los bloques de la memoria son clasificados por tamaño, de esta forma la asignación de la memoria es una tarea rápida. La estructura de datos cambia a un arreglo de listas en los cuales se almacenan los bloques vacíos de memoria por intervalos de tamaño, esto permite que, al llegar un nuevo proceso, dependiendo de su tamaño, se busque en el índice correspondiente al intervalo de su tamaño y se elige el primer bloque que se encuentre en ahí, en caso de que la lista este vacía, se va al siguiente índice, hasta encontrar un hueco al que se le pueda asignar. La

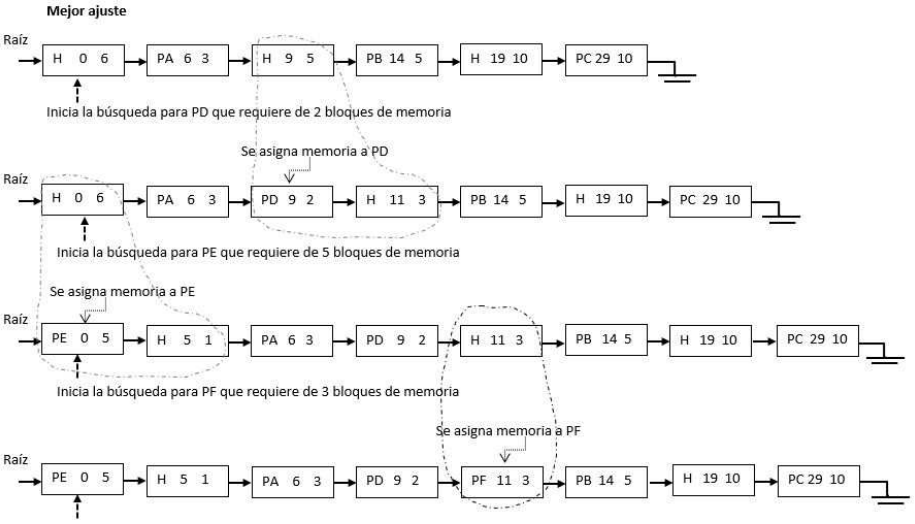

Fig. 5. Funcionamiento de la heurística mejor ajuste.

Figura 7 muestra la forma en la que las estructuras se configuran después de asignar un proceso.

El problema del ajuste rápido, es que no es tan rápido para liberar la memoria, debido a que toda la memoria no se representa en una sola lista ligada, al terminar un proceso no pueden checarse de manera sencilla quiénes son sus vecinos izquierdo y derecho y si estos son huecos. En lugar de esto tiene que buscarse dentro de todo el arreglo de listas donde se encontrarían estos vecinos.

5. Resultados de la construcción del simulador y del desempeño de los estudiantes

Este simulador se ha ido construyendo poco a poco con ayuda de los estudiantes del curso de Sistemas Operativos II del periodo de Primavera 2018 de la FCC-BUAP. Por el momento se han construido: el módulo *Lectura de procesos*, el módulo *Asignador de memoria* del cual se han implementado las heurísticas de empaquetamiento descritas en la sección anterior, el módulo *Despachador de procesos* del cual se ha implementado el algoritmo de Round Robin y el módulo *Liberador de memoria* adaptado a las técnicas de asignación de memoria construidas hasta el momento. Los lenguajes de programación que los estudiantes utilizaron en su mayoría fueron lenguaje C y Java. En la Figura 1 se muestran sombreados aquellos módulos que están en fase de construcción.

Se hicieron pruebas con 2000 procesos simulados. Los parámetros que maneja el simulador son los siguientes: se maneja un quantum de 2 segundos, se simulan 50 Mb con un tamaño de bloque de 1 Kb, los procesos tienen asociados entre 1 y 100 bloques cada uno, y los procesos tienen un tiempo de ejecución de entre

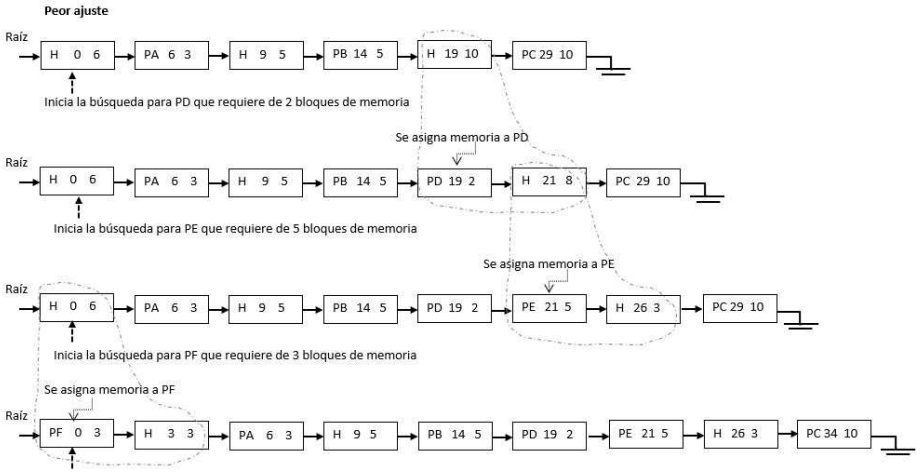

Fig. 6. Funcionamiento de la heurística peor ajuste.

1 y 10 segundos. Los resultados de la evaluación que hicieron los estudiantes a los algoritmos de asignación de memoria, difieren uno de otro. Los estudiantes al realizar su reporte, explican el porqué de sus resultados, y mencionan algunas de las variables que pueden influir como el lenguaje de programación utilizado, la cantidad de procesos que entra cada cierto tiempo, el tamaño de los procesos, el tiempo de ejecución de los procesos, etc. Esto indica que los estudiantes han logrado un conocimiento significativo en éste tema.

La estrategia de solicitar a los alumnos la implementación de heurísticas de empaquetamiento para la asignación de memoria, se ha utilizado en varias ocasiones (al menos en tres ocasiones) en los cursos de Sistemas Operativos II, y se ha encontrado que los alumnos logran comprender mejor los conceptos relacionados con la asignación de memoria y con las heurísticas de empaquetamiento. En todas estas ocasiones se ha planteado a los estudiantes un escenario en el cual se debe asignar memoria a tres procesos, mediante heurísticas de empaquetamiento y el 90 % de los estudiantes han logrado resolver satisfactoriamente el problema. Además se ha aplicado un cuestionario con respecto a conceptos de las tareas que debe llevar a cabo un asignador de memoria y también el 90 % de los estudiantes contestan correctamente.

Además de la construcción de los simuladores, se está construyendo una página [18] en donde se describen brevemente cada una de las heurísticas aplicadas a la asignación de memoria y donde se pueden descargar dos de los simuladores construidos por los alumnos.

6. Conclusiones

Las heurísticas y/o metaheuristicas de la IA pueden utilizarse en los diferentes módulos del sistema operativo, de ahi la importancia de que los estudiantes las

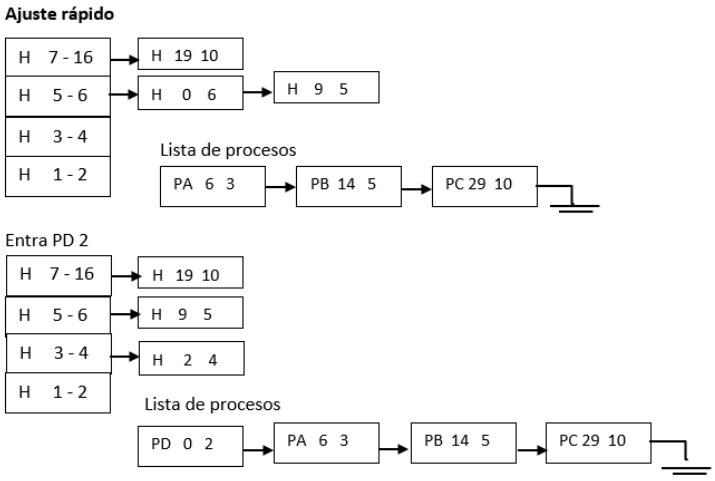

Fig. 7. Funcionamiento de la heurística ajuste rápido.

conozcan.

La implementación del simulador de asignación de memoria y despacho de procesos en donde se utilizan heurísticas de empaquetamiento fueron de vital importancia para que los alumnos del curso de Sistemas Operativos comprendieran y describieran las diferentes tareas que se tienen que realizar en el proceso de la asignación y liberación de memoria utilizando listas ligadas, así como las tareas que debe llevar a cabo un despachador de procesos. Después de que los estudiantes implementaron el simulador fueron conscientes del tipo de problemas que pueden resolverse mediante estas heurísticas y de las variables que podrían alterar los resultados.

En cuanto a la construcción del simulador, se implementaron los módulos de lectura de procesos, de asignación de memoria con heurísticas, de despacho con Round Robin y el liberador de memoria, pero queda como trabajo a futuro implementar el resto de módulos. Se pretende implementar algunos algoritmos para el remplazo de páginas y de despacho de procesos, y de esta forma elegir entre estos algoritmos y reconfigurar el simulador para evaluar su desempeño.

El simulador esta dirigido a la enseñanza aprendizaje de los sistemas operativos en cuanto a los temas de asignación de memoria y despacho de procesos, para que los estudiantes puedan evaluar el desempeño de las diferentes técnicas y que además puedan manipular el código para analizarlo y realizar los cambios pertinentes, e incluso puedan proponer mejoras.

Referencias

1. Bein, D., Bein, W., Venigella, S. Cloud storage and online bin packing. In: Intelligent Distributed Computing V. pp. 63–68. Springer, Berlin, Heidelberg. (2011)

2. Calandín, G., Andrés C.: Métodos y Algoritmos para resolver problemas de corte unidimensional en entornos realistas. Aplicación a una empresa del Sector sider urgico. Tesis doctoral, Universidad de Valencia, Departamento de Organización de Empresas, España (2008)
3. Garey, M.R., Graham, R.L., Ullman, J.D.: An analysis of some packing algorithms. BeliTelephone Laboratories, Inc. and Princeton University (1979)
4. Tanenbaum, A., Baer, P., Gagne, G.: Fundamentos de Sistemas Operativos. 7a edición. Mc Graw Hill, España (2005)
5. Buendía, F., Cano, Juan-Carlos, Sahuquillo, Julio: Uso de simuladores y herramientas Web para la enseñanza de Sistemas Operativos. In: Actas del SImposio Nacional de Docencia en la Informática, SINDI2005 (AENUI), 121–128 (2018)
6. Santana, M., Santana, M.: Uso de minix para la enseñanza de sistemas operativos. Pro Mathematica 1(2), 95–115 (1987)
7. Trefftz, H., Cardona, J.F.: Enseñanza de sistemas operativos con un simulador didáctico fácilmente extensible. In: Encuentro Internacional de Educación en Ingeniería ACOFI 2015, pp. 1–8. Asociación Colombiana de Facultades de Ingeniería, Colombia (2015)
8. Paginación de memoria en OS X, https://revistas.udistrital.edu.co/ojs/index.php /REDES/article/view/5924/7427. Última visita 19 de abril de 2018
9. Hill, J., Ray, C. K., Blair, J. R., Carver Jr, C. A.: Puzzles and games: addressing different learning styles in teaching operating systems concepts. ACM SIGCSE Bulletin 35 (1), pp. 182–186, ACM (2003)
10. Buendía, F., Cano, J. C.: Webgene $_{OS}$: A Generative and Web-Based Learning Architecture to Teach Operating Systems in Undergraduate Courses. IEEE Transactions on Education, 49(4), pp. 464–473 (2006)
11. Ahmed, A., Ibrahim, M.: Analysis of Energy Saving Approaches in Cloud Computing using Ant Colony and First Fit Algorithms. Analysis, 12, (2017)
12. Kumar, A., Sathasivam, C., Periyasamy, P.: Virtual machine placement in cloud computing. Indian Journal of Science and Technology, 9 (29), (2017)
13. Mann, Z. A., Szabó, M.: Which is the best algorithm for virtual machine placement optimization?. Concurrency and Computation: Practice and Experience, 29(10) (2017)
14. Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., Buyya, R. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and experience, 41(1), pp. 23–50 (2011)
15. Lee, C. C., Lee, D. T.: A simple on-line bin-packing algorithm. Journal of the ACM (JACM), 32(3), pp. 562–572 (1985)
16. Seiden, S. S., Van Stee, R., Epstein, L.: New bounds for variable-sized online bin packing. SIAM Journal on Computing, 32(2), pp. 455–469 (2003)
17. Silberschatz, A.S.: Sistemas Operativos Modernos. 3ra edición. Pearson / Prentice Hall, México (2009)
18. Administración de memoria, http://algoritmosdeubicacion.mex.tl/. Última visita 19 de abril de 2018