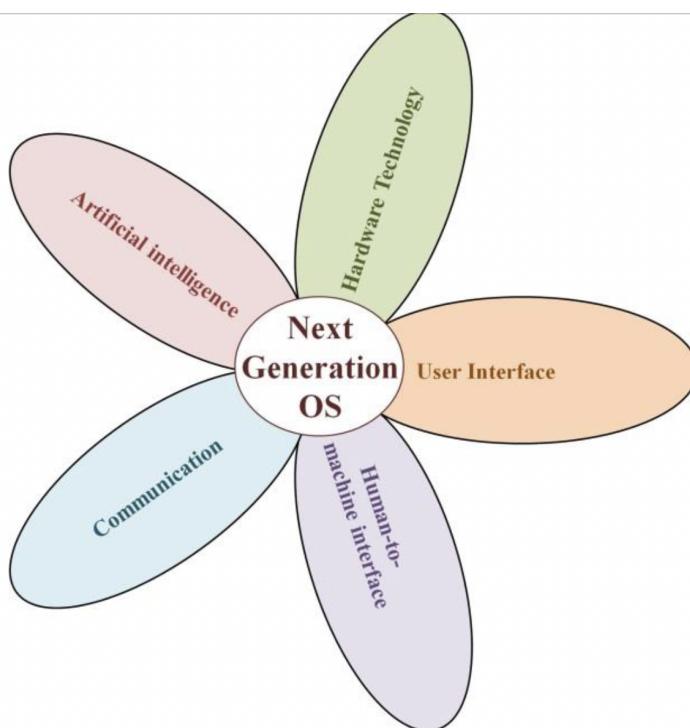




Inteligencia artificial implementada en sistemas operativos.

Introducción.

Los sistemas operativos han ayudado a emplear y administrar el uso de los recursos de una computadora. Estos sistemas permiten al usuario interactuar con los recursos brindados por el hardware y dependiendo de dicho usuario se puede emplear un uso eficiente de las componentes físicas de la computadora. Los Sistemas operativos contemporáneos usan ciertas estrategias en la planificación de procesos mediante el empleo de colas como el de multinivel o el de ruta más corta para manejar múltiples procesadores, pero cuando se tiene una planificación en un solo procesador, no se ha tenido mucho desarrollo. Hay casos, por ejemplo en el manejo de procesar aplicaciones multimedia como los videojuegos o imágenes, en donde se necesita que el sistema operativo sea lo suficientemente rápido para obtener todos los recursos que la aplicación necesita, estos luego no llegan a ser los óptimos. Es por ello, que a través de investigaciones en diversas áreas de la tecnología se ha buscado que la administración de memoria, planificación de procesos, la optimización de energía y entre otros elementos de los sistemas operativos se logre mejorar su eficiencia.





Pruebas e Implementaciones en los sistemas operativos a través de IA.

A nivel de kernel:

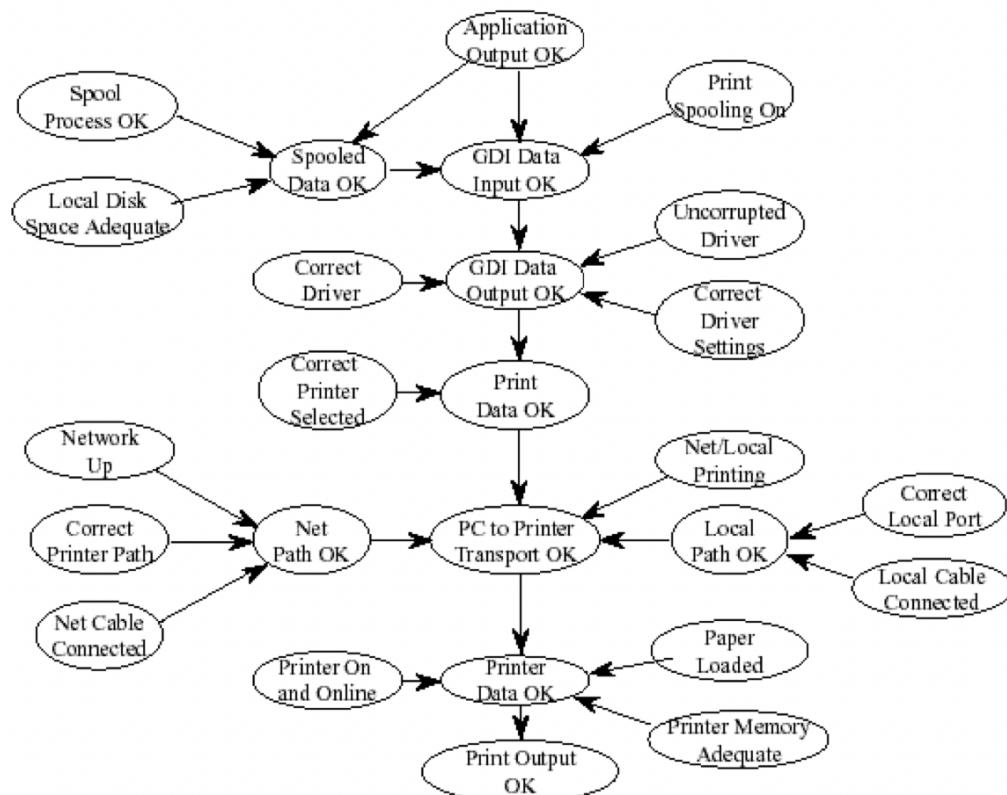
Con el paso de los años, los avances a nivel de hardware han sido de rango exponencial, mientras que a nivel de software no ha habido mucho relevancia pues a pesar de que han habido diferentes actualizaciones de los S.O o han surgido nuevos, la arquitectura en kernel sigue siendo la misma a través de años, (monolítico, microkernel, **kernels** híbridos y los exonúcleos).

Planificación de procesos: La planificación de procesos maneja un rol vital en los multiprocesos de cualquier sistema operativo. Existen muchos factores que envuelven dicha planificación como las prioridades, liberación de memoria, demanda del usuario y el procesador. Si este no está bien manejado puede ser muy complejo y demandante de tiempo según las características de cada proceso. Bien se sabe que un uso eficiente en la planificación y los cambios de contexto, es decir, el cambio entre un procesador a otro mientras se trata una proceso, son pasos importantes en el multiprocesamiento. Mediante la implementación de algoritmos de IA, veremos que tan útiles y eficientes pueden llegar a ser en un sistema operativo. Si en realidad es que vale la pena o no y hasta dónde es que se ha llegado a implementar.

Muchos estudios se han realizado respecto al tema, empleando simulaciones donde corren y prueban los siguientes algoritmos a nivel del kernel. Dichos algoritmos son los siguientes:

- **Sistemas bayesianos o redes de inferencia:** Estos sistemas son modelos gráficos que muestran las relaciones posibles entre las variables. Es decir, deben estar relacionados entre sí, por ejemplo, en las enfermedades la tifoidea o una infección estomacal es comúnmente relacionada a algo que comiste. La gripe por ejemplo no está relacionada a la comida. En las redes bayesianas es posible emplear predicciones futuras inclusive con la falta de algunos valores en los datos conocidos de entrada. Porque también es importante recalcar que los algoritmos de IA siempre necesitan datos de entrada para poder trabajar y entregar una cierta salida.

A continuación se muestra en la imagen una red bayesiana que muestra el diagnóstico de problemas de impresión de una impresora.



Diagnóstico de Problemas de Impresión (Heckerman)

Continuando con las redes bayesianas, estas pueden ser usadas para aprender las relaciones causantes, es decir, que ocasionó algo. Retomando el ejemplo de las enfermedades, podemos saber que ocasionó la fiebre, puede ser por gripe o por covid, u alguna otra enfermedad y ayudan a un mejor comprendimiento para predecir eventos futuros. Como la red bayesiana es un grafo acíclico directo, no puede ser usado cuando el modelo gráfico contiene un ciclo. Este acercamiento computacional llega a ser costoso. En general no encontré alguna atribución existente empleando datos de procesos del S.O.

- **Árboles de decisión:** Un árbol de decisión es a grandes rasgos, un mapa de resultados de una secuencia de decisiones relacionadas. Un árbol de decisión, por lo general, comienza con un único nodo y luego se ramifica en resultados posibles. Estos resultados crean nodos adicionales que se van adhiriendo al árbol y se ramifican en otras posibilidades.

Está compuesto por nodos los cuales son de tres tipos: Dichos tipos son los siguientes:

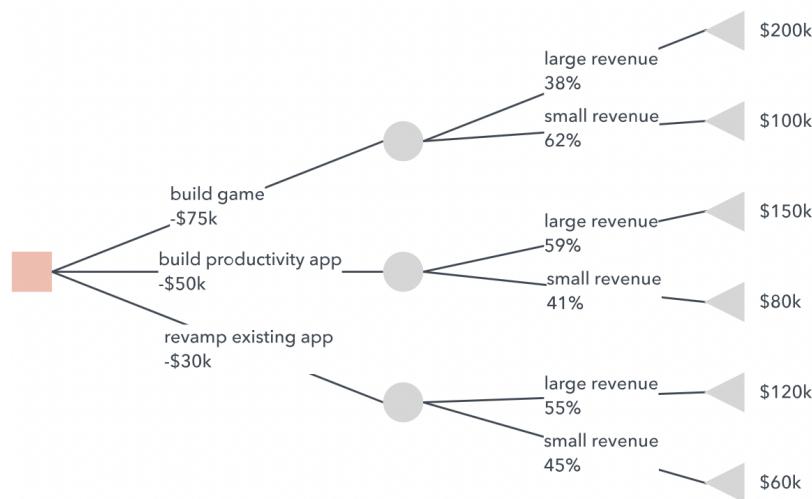
- Un nodo de probabilidad, representado con un círculo, muestra las probabilidades de ciertos resultados.
- Un nodo de decisión, representado con un cuadrado, muestra una decisión que se tomará.



- Y un nodo terminal muestra el resultado definitivo de una ruta de decisión.

Figura	Nombre	Significado
(cuadro)	Nodo de decisión	Indica una decisión que se tomará
(círculo)	Nodo de probabilidad	Muestra múltiples resultados inciertos
(triángulo)	Nodo terminal	Indica un resultado definitivo

En la siguiente imagen, se muestra un ejemplo .Tal cual es un árbol de decisión que determina aproximadamente la cantidad de beneficio en ingresos para realizar una aplicación.



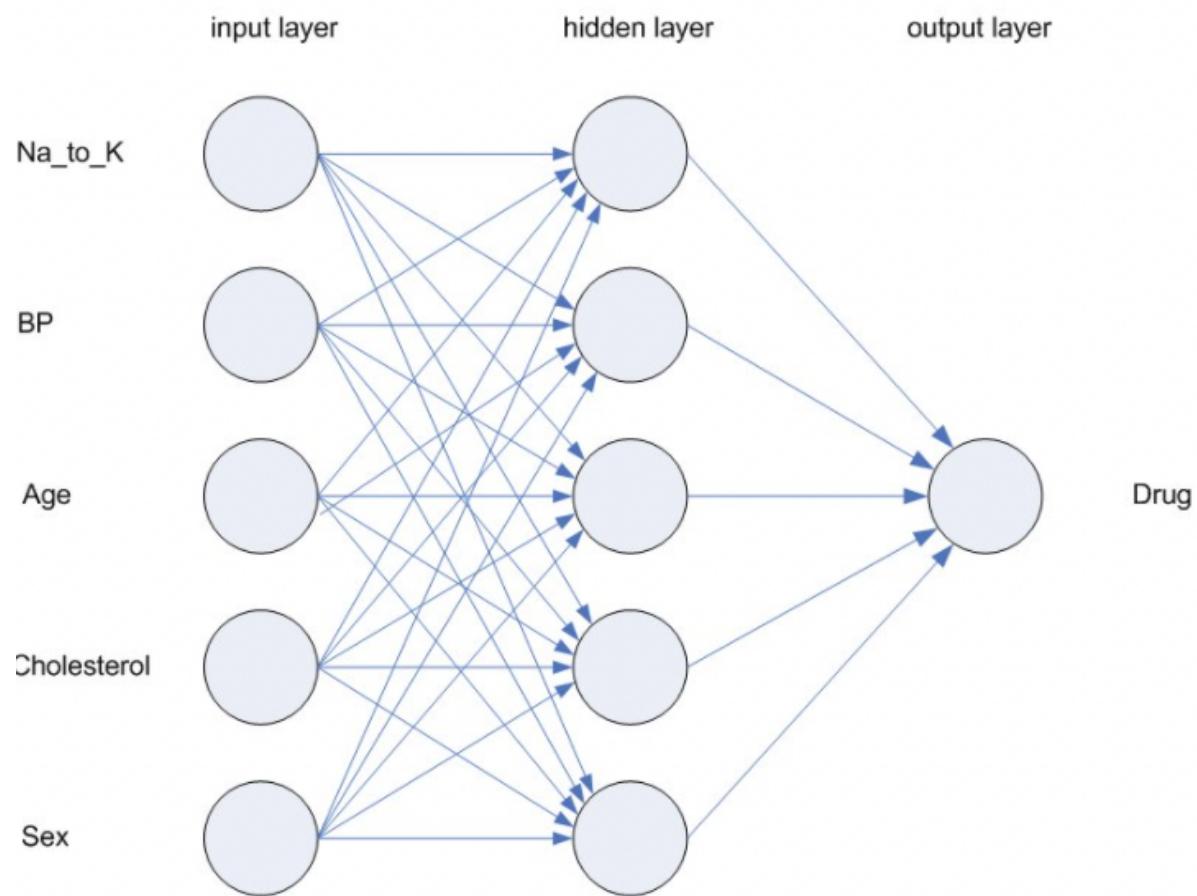


La principal desventaja de esta implementación en sistemas operativos es que es muy sensible a alteraciones ocasionadas por las pequeñas variaciones de los datos, afecta mucho la variabilidad, por lo que el árbol de decisión pierde precisión. En este caso los procesos no podrían ser plenamente eficientes en consecuencia de esta variabilidad ya que se volvería impredecible su eficiencia y no garantiza una solución global óptima .

- **Redes neuronales:** Las redes neuronales tienen una naturaleza en general adaptable para manejar las partes complejas fácilmente. Es de lo que más he encontrado de información al respecto y lo que en general han tratado de implementar y manejar en los sistemas operativos.

- ¿Qué es una red neuronal?

Una red neuronal son modelos simples de IA que tratan de emular el funcionamiento del sistema nervioso, emula el modo en que el cerebro humano procesa la información: Funciona ejecutando simultáneamente un número elevado de unidades de procesamiento interconectadas que parecen versiones abstractas de neuronas. A través de neuronas como unidad básica , generalmente se organizan en capas como se muestra en la siguiente imagen.



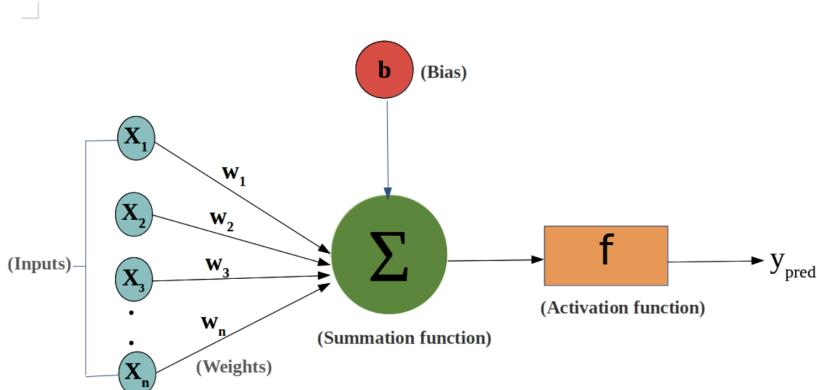
Hay tres partes normalmente en una red neuronal : una **capa de entrada**, con unidades que representan los campos de entrada; una o varias **capas ocultas**; y una **capa de salida**, con una unidad o unidades que representa el campo o los campos de destino. Las unidades se conectan con fuerzas de conexión variables (o **ponderaciones**). Los datos de entrada se presentan en la primera capa, y los valores se propagan entre cada neurona existente de la siguiente capa. Al final, se envía un resultado desde la capa de salida.

La red aprende examinando los registros individuales, generando una predicción para cada registro y realizando ajustes a las ponderaciones cuando realiza una predicción incorrecta. Este proceso se repite muchas veces y la red sigue mejorando sus predicciones hasta haber alcanzado uno o varios criterios de parada.

Al principio, todas las ponderaciones son aleatorias y las respuestas que resultan de la red son, posiblemente, disparatadas e ilógicas. Es por ello que la red neuronal debe de ser entrenada.



A medida que progresá el entrenamiento, la red se va haciendo cada vez más precisa en la replicación de resultados conocidos. Una vez entrenada, la red se puede aplicar a casos futuros en los que se desconoce el resultado.



En esta imagen se comprende de mejor manera cómo es que funcionan matemáticamente las capas ocultas.

Donde

Las **entradas** son el conjunto de valores el cual se necesita para predecir el valor de salida, pueden ser como las características o atributos del conjunto de datos.

Las **cargas** son los valores reales que son asociados a cada característica y menciona la importancia de cada atributo para predecir el valor final. Qué tan importante es el atributo.

El **Bias** cambia la función de activación de cierta capa.

La **función de suma** une las cargas y entradas y encuentra la suma total.

La **función de activación** introduce los modelos no lineales a la red.

Para una red neuronal como planificador de proceso se plantea como los datos de entrada, los datos del usuario, y como salida se obtendría un pre ordenamiento de los procesos en la cola con su correspondiente cantidad necesaria para reservar memoria, así el siguiente proceso con el que cuenta mayor prioridad que el usuario vaya a acceder, ya esté listo para correr o ejecutarse. Para que funcione, como se mencionó anteriormente, la red neuronal debe de estar entrenada inicialmente, es decir, tiene que tener una etapa inicial de entrenamiento para procesar los datos de uso de proceso del usuario. Mientras se va entrenando, la red neuronal va ajustando las cargas con base en el conjunto de datos brindados con la finalidad de tener resultados precisos en la salida.

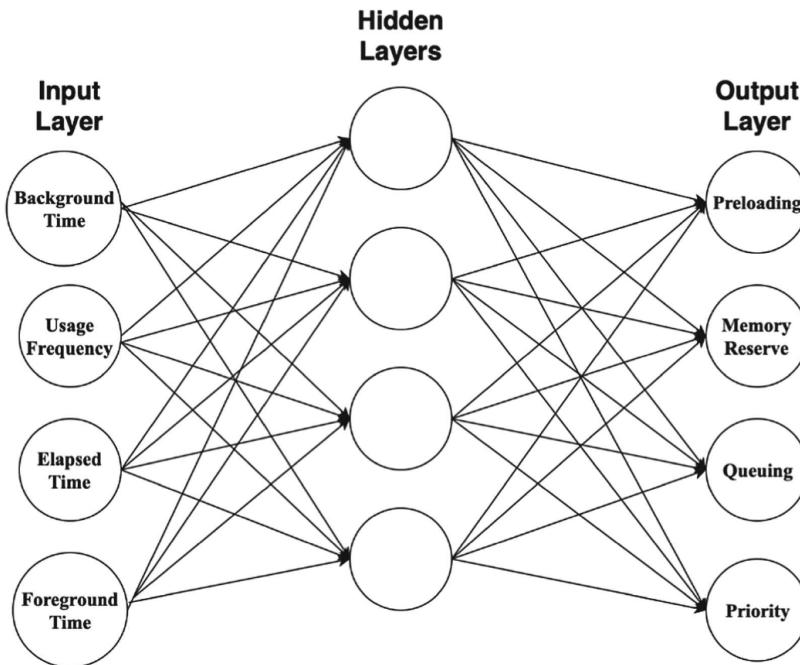


Fig. 2. Proposed System diagram

En la red propuesta de la imagen, como datos de entrada, se pueden tomar los siguientes: tiempo de primer plano, el tiempo de fondo , tiempo transcurrido, frecuencia de uso.Estas entradas pueden variar, por lo que se deben de emplear métodos de normalización de datos para reducir redundancia y mejorar en este caso la salida las cuales estos resultados son ordenados por su prioridad. La urgencia de precarga, reserva de memoria de la aplicación tanto en ram como en memoria virtual y el encolado de procesos según el uso del CPU serían las salidas presentadas de la red neuronal.

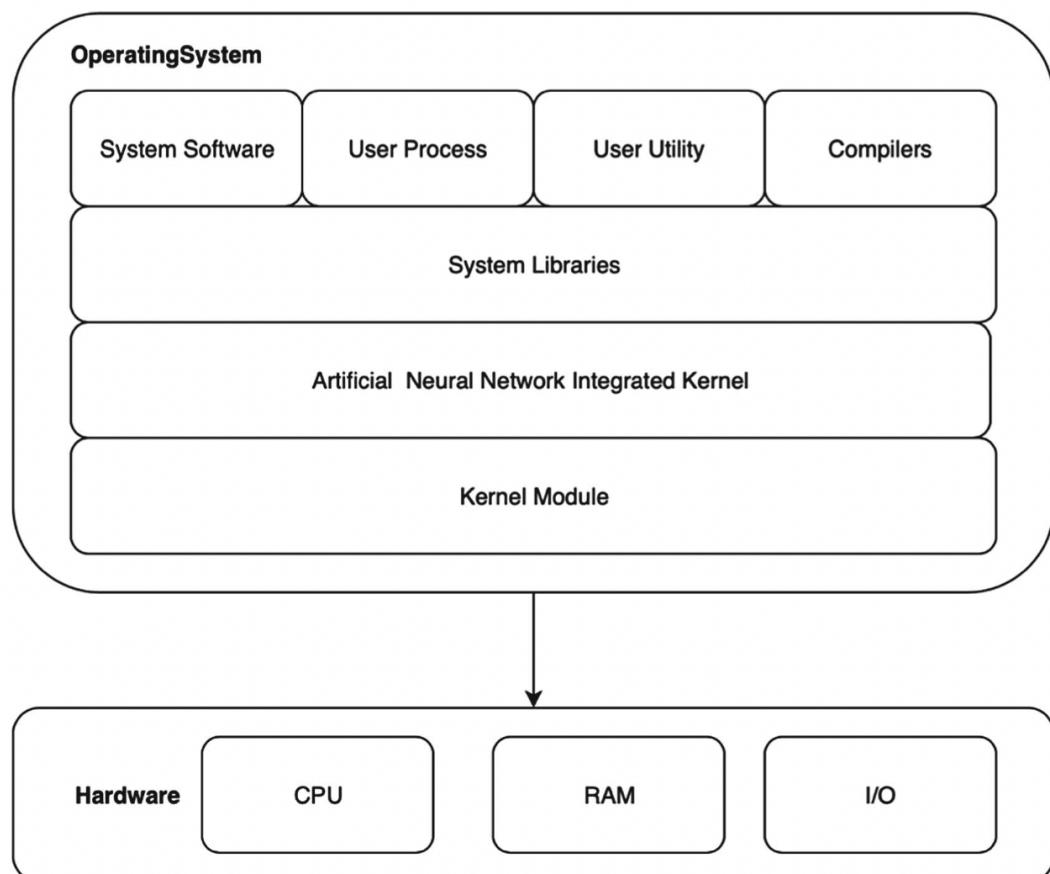
En la propuesta implementada por Gaurav Jarilawa y Harshit Argawal, 4 capas ocultas son implementadas y cada capa contiene 6 nodos los cuales los ejes son inicializados con cargas diferentes y cuentan con funciones específicas para procesar la información en la capa oculta. Algunas funciones empleadas fueron la activación de unidad lineal rectificada que se encarga de producir salidas no lineales y la función de optimización Adam la cual actualiza las cargas de la red neuronal durante el entrenamiento.

En general, la finalidad del método a nivel de kernel es envolverse a sí mismo para que pueda participar en tomar decisiones de núcleo a través del sistema operativo.



Puntos negativos: Algunos detalles a considerar en su implementación es saber dónde se va almacenar esta información del usuario procesada, a través del bloque de memoria? Eso implicaría una alta cantidad de espacio requerido, por lo que llega a ser costoso. La finalidad de realizar esto es conseguir que se pueda retroalimentar sin la necesidad de consumir poder de cómputo del CPU o GPU. Se debe buscar una manera en la que entrene el algoritmo a través de que esté en estado de suspenso, o que no esté en uso por el usuario. La falta de flexibilidad del sistema operativo es también un punto que se pierde, bueno podría ser positivo o negativo dependiendo del enfoque, debido al comportamiento único de un solo tipo de usuario. Con el algoritmo entrenado el kernel anticipa las necesidades del usuario mientras está cargando el sistema. Hay resultados encontrados en donde muestran resultados muy positivos implementando este tipo de sistemas, al final de la presentación anexaré las fuentes donde explican a detalle y se encuentran las pruebas experimentales y sus menciones.

La arquitectura en este caso también se ve afectada pues el Kernel tendría que estar envuelto por el sistema operativo como en la imagen y tener la red neuronal integrada.



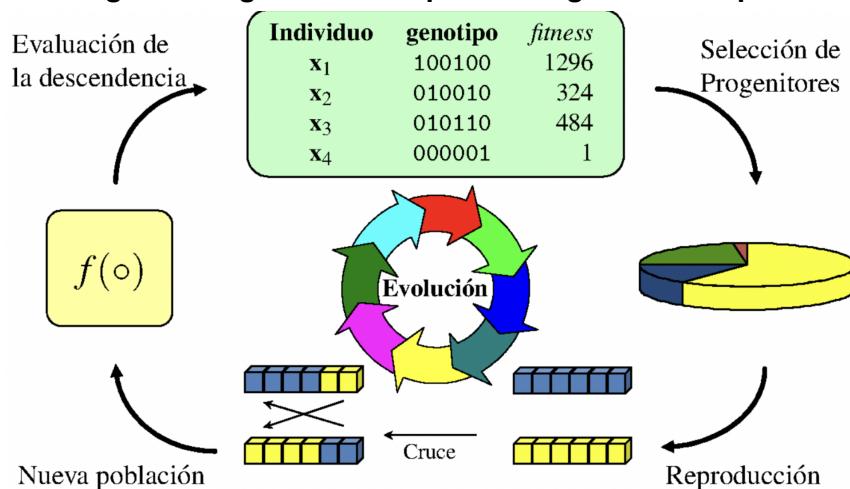


En conclusión mediante el empleo de una red neuronal en el SO, es necesario saber que proceso o programa debe planificar con base en el comportamiento del usuario en cierto periodo de tiempo. Es decir, existe una restricción el cual debe de ser exclusivamente para algún tipo de usuario en específico para que el sistema operativo sepa cuáles procesos son importantes o no. Un ejemplo cómputo orientado a un solo propósito son en servidores web o en videojuegos.

Otro aspecto que también se puede eficientar es en la multiprogramación para emplear un manejo de memoria eficiente teniendo la mayor cantidad de procesos posibles en la memoria principal para un uso óptimo del procesador.

Algoritmos genéticos: Los planificadores a través de algoritmos genéticos son otra opción en IA, estos algoritmos se basan en realizar búsquedas estocásticas inspiradas en la selección natural, evolución y genética .

Los algoritmos genéticos emplea las siguientes etapas:



Vamos a mencionar a detalle brevemente las etapas del algoritmo genético para una mejor comprensión.

Etapa de selección

Un algoritmo genético puede utilizar muchas técnicas diferentes para seleccionar a los individuos que deben copiarse para crear la siguiente generación.

Selección elitista: Selección de los miembros más aptos de cada generación.



Selección proporcional a la aptitud: los individuos más aptos tienen más probabilidad de ser seleccionados, pero no la certeza.

Selección por rueda de ruleta: la probabilidad de que un individuo sea seleccionado es proporcional a la diferencia entre su aptitud y la de sus competidores.

Selección escalada: al incrementarse la aptitud media de la población, la fuerza de la presión selectiva también aumenta y la función de aptitud se hace más discriminadora.

Selección por torneo: se eligen subgrupos de individuos de la población, y los miembros de cada subgrupo compiten entre ellos.

Selección por rango: a cada individuo de la población se le asigna un rango numérico basado en su aptitud, y la selección se basa en este ranking.

Selección generacional: la descendencia de los individuos seleccionados en cada generación se convierte en toda la siguiente generación.

Selección jerárquica: los individuos atraviesan múltiples rondas de selección en cada generación. Las evaluaciones de los primeros niveles son más rápidas y menos discriminatorias, mientras que los que sobreviven hasta niveles más altos son evaluados más rigurosamente.

y pues bueno aquí en la diapositiva están brevemente la mayoría de tipos de selección.

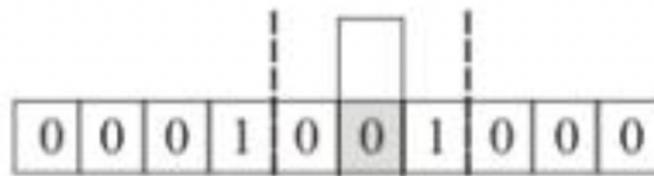
Etapa de cambio

Una vez que la selección ha elegido a los individuos aptos, éstos deben ser alterados aleatoriamente con la esperanza de mejorar su aptitud para la siguiente generación.

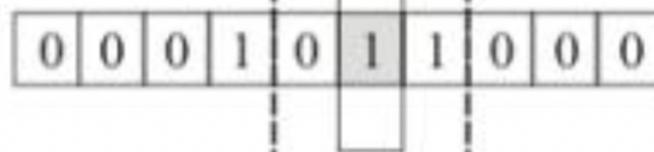
Mutación: Al igual que una mutación en los seres vivos cambia un gen por otro, una mutación en un algoritmo genético también causa pequeñas alteraciones en puntos concretos de la codificación del individuo.



Parent1



Child1



Cruzamiento: consiste en seleccionar a dos individuos para que intercambien segmentos de su código genético, produciendo una "descendencia" artificial cuyos individuos son combinaciones de sus padres.

cruzamiento de un punto, se establece un punto de intercambio en un lugar aleatorio del genoma de los dos individuos, uno de los individuos contribuye todo su código anterior a ese punto y el otro individuo contribuye todo su código a partir de ese punto

cruzamiento en dos puntos, en el que se intercambian los genes que aparecen en el intervalo de genes delimitados por dos puntos

cruzamiento uniforme, el valor de una posición dada en el genoma de la descendencia corresponde al valor en esa posición del genoma de uno de los padres o al valor en esa posición del genoma del otro parent, elegido con un 50% de probabilidad.

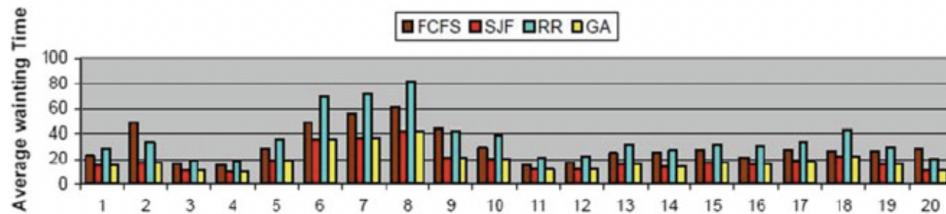
AG:

```
Crea población inicial
Evalúa los cromosomas de la población inicial
Repite hasta que se cumpla la condición de parada
    Selección de los cromosomas más aptos en la nueva población
    Cruzamiento de los cromosomas de la población
    Mutación de los cromosomas de la población
    Evaluación de los cromosomas de la población
    Devuelve la mejor solución (la más apta) en la población
```

Respecto a la implementación de este algoritmo, se muestra a continuación se muestra gráficamente el resultado de unas pruebas donde muestra el



rendimiento en tiempo del algoritmo genético con otros algoritmos planificadores realizados .



Según los resultados obtenidos por los que implementaron dicho algoritmo mostraron resultados similares al algoritmo de trabajo más corto SJN en algunos casos. El algoritmo de Shortest Job first es un algoritmo que se basa en saber y elegir cual es el proceso con menor tiempo de ejecución. Sin embargo, el SJF no es posible implementarlo en procesos a corto plazo. No existe manera actualmente de saber la longitud de la siguiente rafaga del CPU, es por ello que el rendimiento puede ser mejorado con estos algoritmos, inclusive hay estudios en que se muestra la implementación de un algoritmo genético y tiene los mismos resultados que un planificador de prioridad solo que consume mayor tiempo de ejecución. Este problema también puede ser arreglado mediante hardware o con algoritmos más complicados relacionados. Según lo que mencionan los investigadores, presenta una mayor flexibilidad que los otros algoritmos.



Algoritmo	Método	Ventajas	Desventajas	Mejoras
Job Shop Scheduling	-Encola la labor de manera lineal	-Manera más simple de planificar -Compatible con todos los sistemas	-Crea un sistema estático. -Limita el multiprocesamiento.	-Usar funciones de multicapas para incrementar la productividad.
Preemptive Scheduling	-Asigna tiempo quantum para procesar cuando el proceso es tomado por el CPU Posteriormente es regresado a la cola.	-Un solo proceso no puede monopolizar el CPU. -Basado en prioridades de quantum.	-No es el adecuado para procesos basados del usuario de segundo plano	-Los procesos basados en el usuario deben ser programados para sobreescribir tiempo quantum
Multithreading	-Más de un hilo trabaja concurrentemente en un proceso.	-Mejora el rendimiento del sistema. -Mejor uso de los recursos del CPU.	-Debuggear se vuelve complejo -Incrementa el bloqueo mutuo.	-El algoritmo del banquero puede ser empleado para evitar el bloqueo mutuo pero al emplearlo ,el costo de recursos es mayor.
Genetic algorithm	-Funciones de selección, cruzamiento y mutaciones para seleccionar el proceso más adecuado	-Supera las limitaciones del algoritmo de proceso más corto.	-El cruzamiento puede obtener resultados no esperados. -Los resultados son basados en precisiones probabilísticas.	-Creación de un programa más simple para evitar complejidad. -emplear una función de cruce adaptable.
Decision Tree Algorithm	-Los nodos con atributos son separados y el resultado es generado por una serie de decisiones.	-Es fácil de implementar -Se puede emplear parámetros no lineales.	-Resultado variable debido a la variabilidad de atributos. -No produce una solución óptima global.	-Limitar el nivel del árbol podría reducir el reajuste del árbol resultante.
Bayesian System	-Basada en probabilidades	-Puede emplearse incluso con datos faltantes.	-Computacionalmente costoso. -No se puede emplear cuando se tengan ciclos.	-Su precisión es muy buena en conjuntos pequeños de datos pero puede tener alto índice de error.
Neural Network	-Trabaja mediante una estructura de capas y su salida o resultado es generado por la función de activación.	-Crea un sistema operativo de usuario específico -Adaptable a diferentes situaciones	-Hardware muy especializado y costoso para procesar datos. -Difícil de interpretar la estructura de la red neuronal.	-Creando un mejor sistema de compresión y alimentación de la red para un rápido computo.

Un aspecto negativo a ello que mencionan los investigadores es que la respuesta al sistema obtenida de la inteligencia artificial a nivel kernel requiere bastante tiempo de proceso y llega a ser impredecible.



IA en Sistemas Operativos Distribuidos

Los sistemas operativos distribuidos, a grandes rasgos, son sistemas cuyos componentes de hardware y software están conectados por una red distribuida de comunicaciones, se comunican y coordinan sus acciones con la finalidad de lograr un objetivo. Cada máquina conectada posee sus componentes de hardware y software que el usuario siente como un solo sistema. El usuario accede a los recursos de vía remota de la misma manera a la que se accede a los recursos locales.

Estos sistemas son de alta fiabilidad pues si un componente del sistema truena o se descompone otro componente lo puede suplir, es lo que comúnmente se le denomina como **tolerancia a fallos**.

Los sistemas operativos distribuidos han logrado un gran desarrollo en el área de cómputo con el fin de crear aplicaciones mucho más simples gracias a la implementación de IA. Las Redes de conocimiento distribuido (Distributed Knowledge Network), las Distributed Natural Language Summarization(DNLS) búsqueda paralela usando machine learning (PSML) son algunas de las integraciones en donde se ha beneficiado con los sistemas distribuidos. Respecto a las mencionadas un DKN provee la información requerida en tiempo para ayudar a la toma de decisiones de algún dominio en particular. Consiste en emplear sistemas agentes que se enfocan en la recuperación, extracción y asimilación de información a través de los algoritmos de IA mencionados anteriormente como redes neuronales, sistemas bayesianos, entre otros.

En DNS por ejemplo, los agentes proveen resúmenes de información encontrada de diferentes fuentes para algún tema en específico y en el PSML se realizan búsquedas partidas en pequeñas partes a través de búsqueda paralela o árboles de búsqueda distribuidos controlado por el S.O Distribuido.

En general, en las investigaciones que se encontraron respecto al tema, los autores que realizaron las publicaciones sugieren que la amalgama o combinación de IA con sistemas distribuidos puede funcionar efectivamente en grandes arquitecturas de sistemas distribuidos.

- **planificación de procesos y manejo de recursos:** Es claro que el objetivo siempre será minimizar el tiempo total.pudiendo maximizar el empleo de múltiples procesadores. Los balances de carga de los procesadores siempre debería de ser equitativo.Sin embargo, esto en un sistema distribuido está planteado como un problema de tipo NP , es por ello que a continuación presento técnicas de IA que han sido aplicadas exitosamente.
 - **Algoritmos genéticos:** Como vimos anteriormente los algoritmos genéticos son diseñados para encontrar la mejor solución a través de un criterio predefinido.Los investigadores Nikravan y Kashani demostraron que la implementación de dicho algoritmo en la planificación de procesos. Su algoritmo mapea cada planificación como un cromosoma que muestra la ejecución ordenada de todos los procesos existentes del procesador.Cabe resaltar que un cromosoma corresponde a una solución al problema Ellos asumieron que el sistema distribuido es no uniforme , es decir, que los procesadores pueden ser diferentes y que no existen prioridades algunas entre los cpus. También asumieron que el mecanismo del balance de carga en la planificación de procesos estaba centralizado y sin migración.



- **Procedimiento:** El algoritmo se basa en codificar un cromosoma como un arreglo de “n” dígitos donde “n” es el número de procesos. Los índices mostrados por el número de procesos y el dígito puede tomar de 1 a “m” valores correspondientes a los procesadores donde el proceso fue asignado al mismo procesador. Se ordena de izquierda a derecha para determinar la ejecución al procesador. Cada individuo de la primera generación o población inicial se genera aleatoriamente seleccionando un proceso no planificado y asignando a un correspondiente procesador. Circularmente se repite esto hasta que todos los procesos sean asignados. Esto significa que los procesadores fueron seleccionados respectivamente del primero al último y volvieron al primero de nuevo. Un número predefinido de individuos se generan repitiendo este método. La función de ajuste se encarga de encontrar una manera óptima en costo para planificar , mientras se considera el balance de cargas , los procesadores utilizados y el costo de la comunicación. Se representa de la siguiente manera:

$$\text{fitness}(T) = \frac{\text{avg. processors utilization} \times \text{avg. acceptableprocessors}}{\text{maximal finishing time of all processes} \times \text{comms cost to spread}}$$

La ecuación de ajuste filtra los procesos planificados por el menor tiempo de ejecución, menor costo de comunicación , mayor utilización del procesador y mayor promedio de colas del procesador. Los individuos ajustados se seleccionan por métodos probabilísticos para producir descendientes. El cruce generalmente es empleado para intercambiar parte en los cromosomas resultantes, (Y pues bueno se realiza todo el proceso del algoritmo y se llega al objetivo específico) pero puede ser omitido periódicamente para mantener los individuos idóneos. Este cruce fue igual generado aleatoriamente, similarmente la mutación de igual manera se puede omitir para que el desempeño sea bueno en los individuos seleccionados. La siguiente generación se forma seleccionando los individuos resultantes de la nueva población y se crean nuevos individuos esto se repite hasta que se llegue al objetivo específico.

Resultados: Los resultados experimentales que se obtuvieron a través de la simulación indicó que a mayor aumento del número de procesadores mayor manejo de cada uno de ellos. En consecuencia, el algoritmo fue capaz siempre de encontrar



una manera de planificar tal que el tiempo total de finalización permaneciera lineal con respecto al número de procesos, en lugar de exponencial en el caso de que la planificación no fuera óptima. También se observó que al aumentar el número de generaciones de los cromosomas y el tamaño de la población inicial , se aumentaba la calidad de la planificación

- **Algoritmo de búsqueda A*Este** algoritmo es una técnica de búsqueda en grafos muy popular que puede encontrar la ruta al menor costo , para un nodo como meta específica. El costo de un nodo $f(n)$ es la suma de la distancia del nodo inicial al nodo $g(n)$ objetivo o marcado como meta. La distancia estimada es denominada heurística. el cual debe de mantenerse consistente para encontrar una ruta óptima en la solución. Conceptualmente este algoritmo trabaja siguiendo la ruta de menor costo, manteniendo una cola ordenada de rutas alternas e intercambiandolas como si la ruta fuera de un mayor costo hasta que el objetivo sea alcanzado

Procedimiento

Los investigadores Shen y Tsai fueron los primeros en demostrar el uso de este algoritmo para planificación de tareas en un sistema distribuido. El tiempo de respuesta del proceso para cada procesador bajo una asignación de proceso particular se calculó sumando el tiempo total dedicado a la ejecución del módulo y el tiempo total de demora en la comunicación entre procesadores en ese procesador. Por lo tanto, el tiempo total requerido para completar toda la tarea de acuerdo con una asignación en particular fue el tiempo máximo de respuesta del procesador. Es por ello que la asignación de procesos óptima fue la que minimizó el tiempo máximo de respuesta del procesador, que se denominó acertadamente el criterio minimax.

Los procesadores y sus interconexiones se representaron en el gráfico del procesador. De manera similar, un gráfico de procesos representaba los módulos y las comunicaciones entre módulos del proceso en particular. La búsqueda A * se realizó utilizando $g (n)$ calculado con el criterio minimax y $h (n)$ se estableció en 0 cuando el nivel del árbol era mínimo o se estableció en un valor basado en un cálculo de costo especial.

La búsqueda terminó cuando todos los módulos se asignaron a procesadores, lo que arrojó una asignación de tareas óptima.

Resultados:

Los resultados experimentales han validado con éxito la aplicabilidad de un sistema operativo distribuido mediante la búsqueda A*. El investigador Ramakrishnan mejoró aún más este enfoque al observar



que el orden en el que se consideran los procesos para su asignación tuvo un impacto significativo en su desempeño. Otros investigadores como Kafil y Ahmad desarrollaron una versión paralela o distribuida del programador de tareas basado en búsquedas A * que no solo se benefició de la aceleración sino que también tenía menores requisitos de memoria que sus predecesores.

En tolerancias al fallo: La tolerancia al fallo es algo muy común que ocurre en los sistemas distribuidos. La tolerancia al fallo es una propiedad que permite a un sistema seguir funcionando correctamente en caso de **que se caiga** uno o varios de sus componentes conectados. Esto puede ocurrir debido a problemas en hardware o software. Las fallas por hardware pueden ocurrir por varias localizaciones en un sistema distribuido, por ejemplo en memoria, procesador, red , fuente de poder, entre otros. Las fallas de software pueden manifestarse debido a un pobre diseño de aplicaciones, protocolos, etc. Es la responsabilidad del sistema que deba de continuar operando correctamente cuando se presente algún fallo. Muchos sistemas distribuidos manejan para proveer esta característica, redundancia detección de errores o algún mecanismo corrección de errores para varias capas del sistema. El uso de arreglos redundantes de discos incosteadables(RAID) para almacenamiento es un ejemplo de tolerancia al fallo empleando redundancia. El problema de muchos de las técnicas existentes es que son explícitamente ingenieras para manejar un número finito de escenarios posibles y son limitados por la perspectiva del ingeniero o programador. Esto puede ocasionar que se presentan algunas fallas que no las haya contemplado el desarrollador o que no se hayan tratado correctamente. Es por ello, que se han implementado algunas técnicas de inteligencia artificial que han sido exitosamente. El investigador Wildstrom demostró cómo las técnicas de aprendizaje de IA pueden ser usadas para proveer tolerancia al fallo a través de una reconfiguración de hardware. El agente puede detectar cambios en los recursos del procesador , disponibilidad de memoria así como el total de bases de datos conectadas. Cuando la carga de trabajo cambia, entonces el agente decide si alterar la configuración basada en un modelo aprendido a través de datos estadísticos adquiridos.

En seguridad

Actualmente, la implementación de un eficiente y seguro sistema operativo se ha vuelto una necesidad. Para lograr la máxima seguridad con IA en un sistema de cómputo se requiere de grandes cambios además de tratar cada proceso según su tipo. Dichos procesos son clasificados en propios y no propios.

Los procesos de un sistema operativo generado por virus, worms, entre otros pueden ser clasificados como no propios ,Los procesos generados por el sistema o algún software son denominados como propios. Con la ayuda de una red neuronal artificial a través de machine learning, la identificación de los procesos no propios ha crecido rápidamente en el campo de IA.

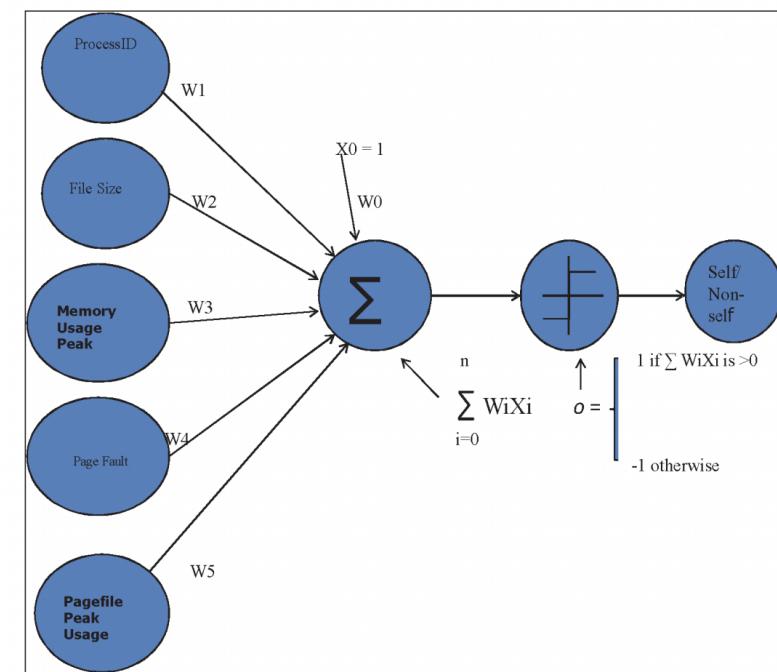


Procedimiento

En una computadora, todos los virus y ataques crean su propio proceso en el sistema. Un proceso tiene tantos atributos como el id de proceso, prioridad nombre, versión , descripción, tamaño, entre otros. Estos parámetros cuentan con un rango máximo o mínimo de valores los cuales son definidos según la arquitectura y la organización del sistema junto con el sistema operativo. En la prueba que se realizó por ciertos investigadores. Se escogieron 5 atributos y que sus valores no fueran nulos para fines prácticos. Posteriormente, con la red neuronal y el empleo de un algoritmo denominado de gradiente descendente se pudo determinar si es un proceso propio o no propio. Para ello, como datos de entrenamiento se emplearon cada uno de los datos del proceso del sistema. Acá en la imagen se puede apreciar los parámetros, que se pudieron emplear en el algoritmo.

CurProcess										
File Options Process Module Help										
Process Name /										
Process Name /	ProcessID	File Size	Base Addre...	Visible Wind...	Hidden Wind...	Mem Usage	Mem Usage ...	Page Faults	Pagefile Usage	Pagefile Pea...
ccSvchst.exe	3352	143,928	0x013D0000	0	3	4192 K	15596 K	17717	5532 K	5668 K
CProcess.exe	5748	36,352	0x00400000	1	5	8668 K	8872 K	2982	4916 K	4916 K
Dwm.exe	3528	92,672	0x00E20000	0	1	28612 K	40032 K	166621	53828 K	62032 K
Explorer.EXE	3628	2,616,320	0x00B50000	4	62	41888 K	43360 K	41009	31020 K	31652 K
FlashUtil0c.exe	4452	257,440	0x00400000	0	0	6108 K	6108 K	1762	1876 K	1876 K
FSRremo5.EXE	2728	20,480	0x00400000	0	3	6312 K	6320 K	1625	2540 K	2600 K
hkcmd.exe	3908	175,640	0x00400000	0	14	9064 K	9112 K	2484	2784 K	2856 K
IASstorIcon.exe	3964	284,696	0x001F0000	0	5	21592 K	21608 K	7500	22380 K	23424 K
ICO.exe	3872	69,632	0x00400000	0	2	5316 K	5336 K	1566	1616 K	1616 K
iexplore.exe	3928	673,040	0x00380000	1	23	34464 K	36684 K	16270	14904 K	17324 K
iexplore.exe	3484	673,040	0x00380000	0	41	166892 K	216580 K	136327	166876 K	226084 K
igfxpers.exe	3952	169,496	0x00400000	0	2	7416 K	7468 K	2024	2172 K	2248 K
igfxtray.exe	3884	141,848	0x00400000	0	2	5872 K	5880 K	1614	1748 K	10512 K
jusched.exe	4052	252,296	0x00400000	0	2	10108 K	11132 K	3221	3220 K	3320 K
MCPLaunch.exe	4032	49,976	0x008F0000	0	0	4364 K	4380 K	1159	1300 K	1324 K
Pelmiced.exe	1636	159,744	0x00400000	0	5	7492 K	7648 K	7362	2744 K	2804 K
RtHDVCpl.exe	3836	7,866,912	0x00400000	0	4	9680 K	10180 K	2826	7996 K	8548 K
Module Name /	Base Addre...	Module Size	Version	Description	Company	Product Name	Modified Date	File Size		
DeltaBac.dll	0x00000000	0x00001F000	12.1.2015.2015	Sumantac Environment Denta	Sumantac Compara	Sumantac Environment Dent	2/0/2013 1:41:24 AM	116,176		

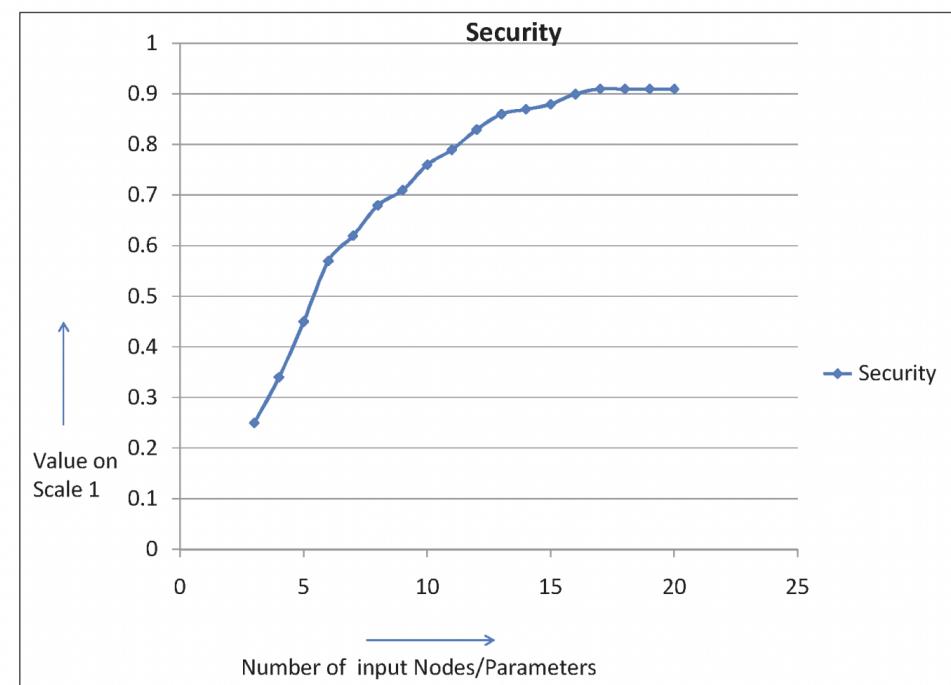
- **Algoritmo de Gradiente descendente:**El algoritmo de gradiente descendente es uno de los tantos algoritmos de inteligencia artificial empleado para ajustar los parámetros de forma iterativa para minimizar una función. Es comúnmente empleado principalmente para resolver problemas de optimización.



Resultados:

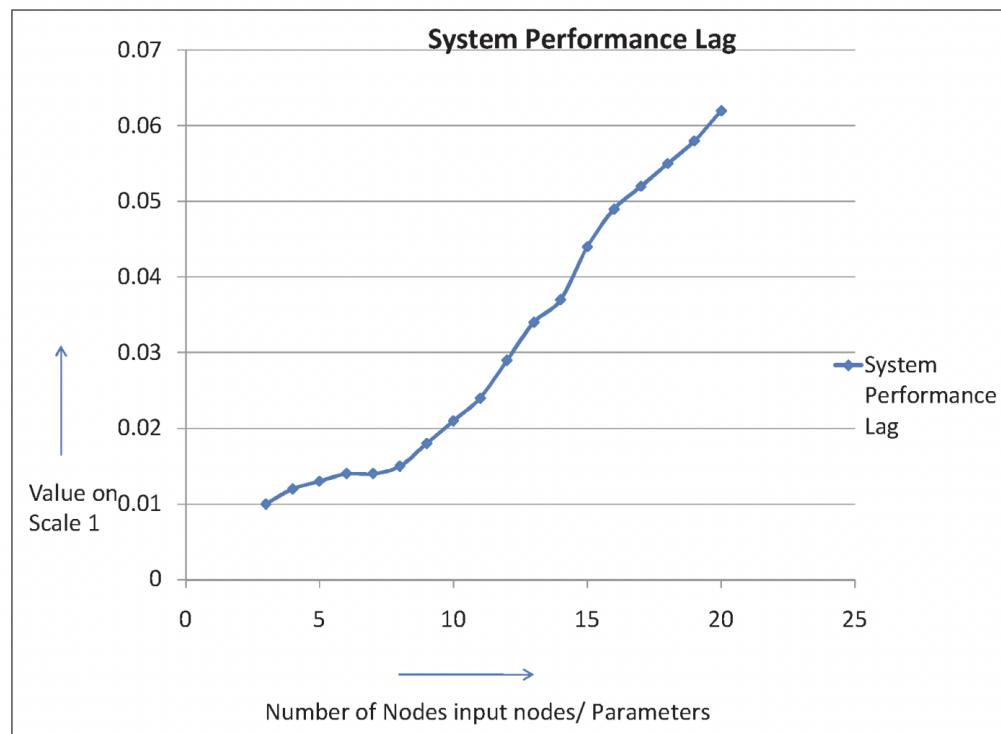
<i>ProcessID</i>	<i>File Size (KB)</i>	<i>Mem Usage Peak (KB)</i>	<i>Page Faults</i>	<i>Pagefile Peak Usage (KB)</i>	<i>Self</i>	
P1	L	M	M	VH	H	Yes
P2	H	M	VL	L	M	Yes
P3	M	L	VL	VL	L	Yes
P4	H	M	M	M	M	No
P5	M	M	L	M	M	Yes
P6	H	L	L	M	M	No
P7	L	M	L	L	M	Yes
P8	H	L	H	VH	H	Yes
P9	H	M	VL	VL	L	No
P10	H	L	L	L	M	Yes
P11	L	M	M	H	H	Yes
P12	M	L	VH	VH	H	Yes
P13	H	M	VL	L	M	No
P14	H	H	M	M	H	Yes
P15	H	M	L	M	M	Yes
P16	H	H	M	M	H	No
P17	M	L	L	VH	M	Yes
P18	L	L	VL	VL	L	Yes
P19	H	H	VL	L	M	No
P20	H	M	L	L	M	No
P21	H	M	M	M	M	Yes
P22	M	M	L	M	M	Yes
P23	M	H	M	M	M	Yes
P24	M	M	M	M	M	Yes
P25	M	H	M	M	M	Yes

Al probar con eficacia el algoritmo, se realizaron diferentes pruebas respecto a su eficacia en el ámbito de seguridad.



En esta imagen podemos apreciar una gráfica de seguridad contra el número de entradas de la red neuronal para demostrar el nivel de eficiencia según su cantidad. En la gráfica se denota que si el número de entradas a la red neuronal aumenta, la seguridad también aumenta. Se tuvo como resultado también que cuando las entradas o los nodos incrementan después de 15 entonces se vuelve constante.

En la imagen debajo, se presenta otra gráfica resultado de relacionar el número de entradas del nodo y su decremento en rendimiento del sistema. Se infirió que a mayor número de parámetros utilizados en la red, el rendimiento del sistema decae. Es por ello, que los investigadores infieren que se debe determinar un número de entradas idóneo con la finalidad de no alterar tanto el rendimiento del sistema .



Se presenta una tabla en donde se demuestra que el acercamiento de una red neuronal es mejor que los acercamientos existentes de seguridad actuales, conocidos como los antivirus. El acercamiento con la red neuronal, necesita menos tiempo para determinar cuales son procesos no propios que los otros acercamientos que escanean todos los archivos y datos de la computadora. La precisión y su índice de detección de igual manera es también alto debido a que escanea todos los procesos del sistema , sin embargo esto lo vuelve lento.

Table 4: Comparison of Proposed Approach with Some Anti-virus tools

Anti-Virus	Parameters	Scan Time	Accuracy	Detection Rate	Performance Lag	Signature based Detection	Regular Updating Required
AVG Anti-virus	High	High	Normal	Yes	Yes	Yes	Yes
Norton Anti-virus	High	High	High	Yes	Yes	Yes	Yes
Avast Anti-virus	High	Medium	Normal	Yes	Yes	Yes	Yes
Microsoft Security Essentials	Average	High	High	Yes	Yes	No	Yes
Proposed Approach	Low	Very High	Very High	Yes	No	No	No

En **sistemas embebidos** se han realizado a nivel de hardware muchas implementaciones con la finalidad de optimizar energía y aumentar velocidad de procesamiento.



Conclusión

Por último, podemos concluir que las investigaciones en inteligencia artificial mantienen mejoras prometedoras o facilitan el desarrollo de los sistemas operativos. Otro punto importante a destacar es el rápido crecimiento en diversas áreas de las tecnologías como en semiconductores, inteligencia artificial, detección y cognición permiten el desarrollo de cómputo más sofisticado y en consecuencia también en los sistemas operativos. Cabe resaltar que se va volviendo más complejo el comprendimiento de los sistemas operativos pero es por ello que el desarrollo en las áreas mencionadas al principio, se espera que los avances en los sistemas operativos faciliten diversas labores de trabajo como en administración de bases de datos, comunicación entre computadoras, procesamiento, entre otros. Unas posibles características de la siguiente generación de sistemas operativos es probable que para ordenar a la computadora ciertos comandos en la línea de comandos (CLI) o la interfaz gráfica también se emplee un comando de memoria como Siri en apple o Cortana en Microsoft. Actualmente las órdenes que se emplean a dichos asistentes virtuales son muy superficiales. Inclusive puede ser que a través de gestos se pueda controlar y manejar eficientemente el sistema operativo. En la comunicación de máquina a máquina es probable que en un futuro sean conectadas entre sí a través de un punto global para todos los ordenadores. La inteligencia artificial puede que en un futuro esté integrada para eficientar los procesos de un tipo de usuario en específico en el manejo de recursos, dependiendo los procesos que utiliza. En el área del hardware, es importante también debido a que se puede contener IA en un chip el cual puede eficientar la velocidad de cómputo en combinación con multiprocesadores. En el caso de que se llegue a emplear por ejemplo machine-learning, se debería de emplear algún modo de retroalimentación en los algoritmos para agilizar la planificación de procesos, entre otras cosas. La idea es que al recibir un dato(en este caso de los procesos) como entrada, correr algoritmos de análisis para hacer que el aprendizaje de los algoritmos pudiera ser “offline” o cuando esté suspendida la máquina para que el usuario no tenga repercusiones en rendimiento de su computadora. Dicho aspecto es altamente complejo, pues el consumo de energía y memoria puede ser altamente demandante.

En el caso de los sistemas distribuidos, la combinación entre IA y sistemas operativos puede llegar a producir más autonomía y sistemas robustos ya que IA provee el cerebro al sistema distribuido para controlarlos.



Bibliografía

- Amit Kumar*, Shishir Kumar**
<http://www.publishingindia.com/GetBrochure.aspx?query=UERGQnJvY2h1cmVzfC8yMjgwLnBkZnvvMjl4MC5wZGY=>
- Conclusion: <https://nube.iiec.unam.mx/s/5wZSfK92zN4DepW>
- Harshit Agarwal and Gaurav Jariwala: Analysis of Process Scheduling Using Neural Network in Operating System:
https://www.researchgate.net/publication/338907007_Analysis_of_Process_Scheduling_Using_Neural_Network_in_Operating_System
- New Optimal Solutions for Real-Time Scheduling of Operating System Tasks Based on Neural Networks.https://ksiresearch.org/seke/seke17paper/seke17paper_25.pdf
- Nadeesha O.Ranasinghe: Artificial Intelligence in Dsitrubuted Operating Systems.<https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.647.123&rep=rep1&type=pdf>
- <https://fdoperez.webs.ull.es/doc/Conocimiento5.pdf>
- <https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-neural-model>