PI121-01

MODELO DE PLANIFICACIÓN PREDICTIVO BASADO EN UN ALGORÍTMO HÍBRIDO DE ENJAMBRE

ANDREA VIVIANA RODRÍGUEZ OLIVA

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
MAESTRÍA EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
BOGOTÁ, D.C.
2016

PI121-01

MODELO DE PLANIFICACIÓN PREDICTIVO BASADO EN UN ALGORÍTMO HÍBRIDO DE ENJAMBRE

Autor:

Andrea Viviana Rodríguez Oliva

MEMORIA DEL TRABAJO DE GRADO REALIZADO PARA CUMPLIR UNO DE LOS REQUISITOS PARA OPTAR AL TÍTULO DE

MAGÍSTER EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Director

Ingeniero Enrique González Guerrero

Comité de Evaluación del Trabajo de Grado

Ingeniero Julián Ángel

Ingeniero Rabie Mohamed Nait-Abdallah

Página web del Trabajo de Grado

http://pegasus.javeriana.edu.co/~PI121-01-PlanPredictEnjambre

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
MAESTRÍA EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
BOGOTÁ, D.C.
Febrero, 2016

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA DE SISTEMAS

Rector Magnífico

Joaquín Emilio Sánchez García S.J.

Decano Académico Facultad de Ingeniería

Ingeniero Luis David Prieto Martínez

Decano del Medio Universitario Facultad de Ingeniería

Padre Sergio Bernal Restrepo S.J.

Director Maestría en Ingeniería de Sistemas y Computación

Ingeniera Angela Cristina Carrillo Ramos

Director Departamento de Ingeniería de Sistemas

Ingeniero César Julio Bustacara Medina

En esta hoja solía venir la nota de aceptación del director y la firma de los jurados. Ahora, el director puede aprobar a través de correo electrónico. Esta hoja puede ser obviada.

Artículo 23 de la Resolución No. 1 de Junio de 1946

"La Universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado. Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque no contengan ataques o polémicas puramente personales. Antes bien, que se vean en ellos el anhelo de buscar la verdad y la justicia"

AGRADECIMIENTOS

Al Ingeniero Enrique González Guerrero, director del trabajo de grado, por asesorarme y apoyarme siempre en la realización de este trabajo.

Al Ingeniero Daniel Ramírez, director de Tecnología de Millenium Contact Center, por sus ideas.

A Carolina Patiño Bustos por su asesoría y apoyo en el desarrollo del proyecto.

A Sandra Contreras por su colaboración con la lectura y corrección de estilo de mi monografía.

Contenido

ABSTRACT

Through three hierarchical levels a predictive planning model is designed. The short-term scheduler performs task assignment in long term. The medium-term scheduler, acts as doser and selects which task should be executed immediately. The short-term scheduler is based on a hybrid swarm algorithm which allocates tasks to the workstation. Thus, the assignment is intelligently performed through data provided by an adaptive component. The result improves on average 30% compared to a ramdomized assignment.

RESUMEN

A través de tres niveles jerárquicos se diseña un modelo de planificacion predictivo. El scheduler a corto plazo realiza asignación de tareas a largo plazo. El scheduler a mediano plazo actúa como dosificador y elige cuáles tasks deben ser ejecutadas de forma inmediata. El scheduler a corto plazo se realiza con base en un algoritmo híbrido de enjambre, que asigna las workstation con los tasks. Por ende, la asignación se realiza inteligentemente a través de datos aportados por un componente adaptativo. El resultado mejora en un 30% de promedio la asignación realizada al azar.

RESUMEN EJECUTIVO

Los modelos de planificación predictiva (MPS) resuelven problemas de optimización combinatorios de planificación con variables de entrada aleatorias.

Algunos problemas que se solucionan por medio de MPS son los Multiproce-ssor Scheduling (MS) de los tipos Minimun Multiprocessor Scheduling (MMS), Minimun Precedence Constrained Scheduling (MPCS), Minimun Resource Constrai-ned Sche-duling (MRCS), multiclass queueing networks (MQNs), entre otros. Estos problemas consisten en la asignación o programación de tareas computacionales en un ambiente de multiprocesadores. Usualmente, se resuelven por medio de programación dinámica estocástica o la combinación de programación dinámica con algunas formas de búsqueda global. Sin embargo, a lo largo de la reciente década se han propuesto como soluciones técnicas las denominadas metaheurísticas. Entre las técnicas metaheurísticas más conocidas se encuentran la inteligencia en enjambre (SI), los algoritmos evolutivos (EA), la búsqueda voraz, el enfriamiento simulado (SA), la búsqueda tabú (TS), entre otras.

Se pretende que a partir de un modelo de planificación predictivo se solucione problemas de scheduling de workstation paralelas con velocidad variable. El modelo que se planeó inicialmente consistía en utilizar series de tiempo y algoritmos de enjambre. No obstante, después de un análisis exhaustivo a cada una de las técnicas, se determinó hibridar una técnica de inteligencia de enjambre seleccionada con una técnica de búsqueda local. Para llegar a este algoritmo hibridado se escogieron las técnicas de optimización por enjambre de partículas (PSO) y búsqueda tabú. Cada una de ellas presenta ventajas y desventajas que la otra técnica entra a subsanar y es precisamente el modelo de planificación predictivo se complementa e incluyéndole niveles jerárquicos a cada uno de estos se especializa en la asignación de recursos, pero en un periodo de tiempo determinado. Es así como se agregó en el diseño el scheduler a largo plazo, el scheduler a mediano plazo y el scheduler a corto plazo. El diseño en niveles del planificador se tomó del scheduler utilizado por los sistemas operativos para asignar tareas al procesador.

Un componente que se adicionó al modelo de planificación predictivo (MPS) fue el adaptador. Este componente hace que la asignación de recursos a un task se realice de forma inteligente, teniendo presente las características del task, que entra y maneja perfiles informando cuál workstation es la indicada para procesar el trabajo nuevo entrante a la cola. Cada uno de los componentes colocados en el modelo de planificación predictiva realiza funciones primordiales, pero no indispensables para el correcto funcionamiento del scheduler a corto plazo, que es el scheduler en el cual se enfoca en este trabajo.

Después de seleccionar cómo se estructura el modelo de planificación predictiva, se planea donde estará ubicado el algoritmo híbrido de enjambre y con cuáles técnicas se construirá. Luego, se procede a desarrollar el modelo a través de un paradigma de programación orientado a agentes del modelo diseñado. Para esto se utiliza una librería en Java denominada Jape, que dará el soporte a agentes en la implementación realizada. Se utilizó esta librería debido a la extensa cantidad de literatura que se encuentra de ella, ejemplos de implementación y madura-

ción en sus versiones junto con el paradigma escogido para desarrollar el modelo de planificación predictivo, el cual permite adjudicar subtemas a cada uno de los agentes. Y es así como el desarrollo complejo se puede segmentar en partes comprensibles de programación.

Como tarea inicial del proyecto de grado se realizó la selección del caso de estudio que se utilizaría para realizar pruebas con datos reales; y por ello se escogió realizar el scheduling de un Call Center, es decir, el proceso de recibir y realizar llamadas por medio de personas llamadas agentes. Mediante este proceso se establece que las variables por optimizar son: el tiempo de procesamiento, que consiste el tiempo que dura la workstation procesando el task y el tiempo de espera (el tiempo que espera un task en cola). Los datos para probar el modelo propuesto se tomaron de las bases del respectivo Call Center. A partir de estas, se construyeron los perfiles de las workstations y tasks.

De igual forma, se crea el protocolo experimental y se divide en tres partes: la primera parte, presenta una evaluación cualitativa de técnicas conocidas que se han tratado en paper para resolver los problemas de scheduling contra el modelo de planificación predictivo desarrollado; para luego en la segunda parte, se crearan datos sintéticos aleatorios, que permiten verificar el funcionamiento del modelo desarrollado para varios contextos, incluyendo el caso de estudio. Y por último, en la tercera etapa, se realizarán los experimentos con los datos obtenidos del Call Center.

Como resultado de comparación cualitativa se expone que diferentes técnicas han tratado este tipo de scheduling, sin embargo, presentan por ejemplo, poca velocidad en el momento de ejecución del algoritmo o, por el contrario, no pueden separar las tasks que deben ser ejecutados de forma inmediata o las que pueden esperar.

Como resultado de los protocolos experimentales cuantitativo se pretende primero verificar el rendimiento del algoritmo bajo diversa cantidad de tasks y workstations, para así caracterizar el algoritmo y encontrar con qué configuración este trabaja mejor. También, se pretende verificar si funciona o no el modelo realizado. Para lo anterior se midió el variable dependiente tiempo de procesamiento tanto para los datos sintéticos como para los datos reales. Es aquí donde se ve que esta variable tuvo una disminución promedio del 30%. Otra variable que se midió fue el tiempo en que se demoraba en resolver el algoritmo híbrido de enjambre el problema de schedulering. La configuración más apta sería donde las partículas y enjambres fueran de nivel bajo junto con las workstations y tasks también.

Como resultado de este trabajo de grado, se puede decir que se construyó un modelo que funciona al resolver problemas de scheduling con workstation paralelas y velocidad variable bajo cualquier tipo de contexto. Un modelo que es sencillo de implementar y que aprovecha conocimiento aportado por el pasado para realizar una mejor asignación de recursos. Debido a que solo se construyó el componente adaptativo y el scheduler a corto plazo, como trabajo futuro queda realizar e implementar el scheduler a mediano y largo plazo respectivamente.

INTRODUCCIÓN

Entre los inconvenientes más comunes en las industrias son los problemas de optimización combinatorios orientados al scheduling estocástico. Un aspecto importante de este tipo de problema es la aleatoriedad en sus variables de entrada. Estos problemas han sido estudiados a través de proyectos académicos incluyéndolos en la investigación de operaciones. Se han desarrollado propuestas como la programación dinámica estocástica para resolverlos (SDP). Sin embargo, existe en la computación una rama de sistemas inteligentes que ha abordado estos asuntos mediante la computación en enjambre y algoritmos evolutivos.

El objetivo de este trabajo de grado es desarrollar un modelo de planificación predictivo (MPS), a través de inteligencia en enjambre y series de tiempo. Como objetivos específicos se proponen: 1) Seleccionar a partir de un análisis conceptual, la técnica de inteligencia en enjambre y la técnica de análisis de series de tiempo más adecuadas para construir un MPS. 2) Desarrollar un MPS meta-heurístico híbrido, que combina las técnicas previamente seleccionadas y 3) Validar el MPS generado a través de un caso de estudio. Cada objetivo se realiza en una de las tres fases definidas en la metodología.

En la primera fase de la metodología se plantea una selección y análisis de información de diferentes autores, en el que se debate conceptualmente las ventajas y desventajas de cada una de las técnicas de inteligencia en enjambre y análisis de series de tiempo en problemas de optimización combinatorios de planificación. Es necesario estudiar las técnicas de forma individual. Se seleccionan los autores y proyectos más relevantes no sólo para conocer las técnicas, sino también para identificar los criterios de calidad, tales como: velocidad de convergencia, capacidad de generalización, aplicación en problemas reales, complejidad computacional, entre otros. Con estos criterios y su valoración apoyada en el caso de estudio, se pre-seleccionan dos técnicas en enjambre y una técnica de análisis de series de tiempo que se utilizarán para diseñar el MPS. Posteriormente, se refinan los criterios de calidad teniendo en cuenta qué ventajas poseen las técnicas ya seleccionadas -de inteligencia en enjambre al combinarlas con la técnica de análisis de series de tiempo-. Se debe tener en cuenta las posibles combinaciones para que al finalizar se seleccione la mejor combinación de técnicas a utilizar en la construcción del MPS.

En la segunda fase de la metodología se plantea la identificación y caracterización el modelo de entradas y salidas de cada una de las técnicas y la manera en que interactúan las técnicas escogidas. Se identifica y modela cada uno de los componentes como una arquitectura. Se crea el modelo de cada uno de los componentes, primero de caja negra y luego el detallado, hasta llegar al desarrollo y refinamiento de los algoritmos que rigen el comportamiento en cada módulo. A lo largo de este proceso de diseño, se realiza un refinamiento del modelo generado en referencia al caso del estudio.

En la tercera fase de la metodología se elige como caso de estudio la planificación de campañas de un Call Center, debido a que se conocen los procesos, se cuenta con personal conocedor de la operación, se poseen los datos y es un problema representativo de la planificación predictiva. Para la validación del modelo se identifican los módulos claves (kernel) para el funcionamiento

del sistema y solamente se implementan éstos de forma progresiva. Se prueban de forma individual cada uno de ellos. La implementación se realiza mediante una metodología de desarrollo ágil. Cada módulo debe tener sus correspondientes pruebas unitarias.

En esta última fase, se debe realizar una selección de la herramienta que se utilizará para la elaboración del ambiente de simulación, que a su vez debe soportar la simulación del caso de estudio previamente seleccionado; además debe ser posible parametrizar la simulación para que permita evaluar escenarios de diferentes complejidades y escalas. Para la simulación se define un protocolo de pruebas, que está orientado a evaluar tiempos de respuesta, correctitud del modelo, entre otras características. Así como para asegurar que el simulador se ajuste a la realidad, se realiza en dos etapas: identificación y priorización de los elementos más relevantes que interfieren en esa realidad, es decir, afinar el software basado en datos de la realidad. Los resultados de la simulación son evaluados estadísticamente, con el fin de obtener una aproximación cuantitativa sobre la validación del funcionamiento del modelo diseñado. La simulación ayuda a validar y caracterizar el funcionamiento del modelo diseñado.

Al finalizar las tres fases de la metodología se da cumplimiento a todos los objetivos específicos y al objetivo general. La contribución que se realiza creando un modelo de planificación predictivo es poder contruir un modelo que resuelva de forma sencilla problemas de optimización combinatorios sin importar que tan complicado, difícil o extenso el problema sea. La otra ventaja que presenta es que se realiza una asignación inteligente no, y eso permite realizar una mejor asignación del recurso.

El presente trabajo de grado se estructura de la siguiente manera: el primer capítulo define conceptualmente que son los problemas de optimización combinatorios, entre ellos el scheduling. Y se define formalmente el problema de scheduling tratado en este trabajo de grado; mientras que en el segundo capítulo se muestra el diseño conceptual que se realiza del modelo de planificación predictivo, así mismo en el tercer capítulo se documenta cómo se implementó el modelo de planificación predictivo, bajo que lenguaje de programación y con qué paradigma de programación. También se realiza una descripción acerca del sistema multiagentes que implementa el diseño contextual del capítulo dos. Y por último, el capítulo cuatro desarrolla los protocolos experimentales para probar el diseño realizado y el análisis de resultados.

I - 1. FORMULACIÓN DEL PROBLEMA Y ANTECEDENTES

Para la toma de decisiones en los negocios, no es suficiente tener reportes simples con el fin de generar una correcta planificación. La optimización debería ser un objetivo por cumplir dentro del diseño de actividades y procesos. El hecho de tomar el concepto de optimización y aplicarlo en el diseño de herramientas redunda en beneficios para las empresas y las partes interesadas (socios, empleados, gobierno). El desarrollo de la tecnología resulta en la creación y adopción de instrumentos para la competitividad, productividad, calidad y por ende, el liderazgo.

Resolver problemas de scheduling permite optimizar los tiempos de diferentes procesos, disminuir los costos asociados al tiempo de proceso o uso, facilitar la toma de decisiones basado en proyecciones acertadas, preparar escenarios para favorecer la adopción de estrategias y enfrentar los cambios en el entorno; en resumen, la optimización por medio del scheduling permite diseñar, crear, administrar y mantener mejores negocios.

1. Problemas de Optimización Combinatorios

El concepto de optimización se puede definir como el proceso de encontrar la mejor solución posible a un problema de optimización generalmente en un tiempo limitado [1].

La optimización es un método importante en diferentes campos de la ciencia, como la computación. Un problema de optimización es un dilema en el que hay varias soluciones probables, y de alguna manera, se pueden comparar entre ellas, de tal forma que este problema existe únicamente si hay una forma clara de respuestas posibles que puedan ser confrontadas.

Visto desde la forma matemática en la Ecuación 1, un problema de optimización P se formula como una 3-tupla $P = (f, SS, F)$ de la siguiente manera:

$$P = \begin{cases} opt: f(x) \\ x \in F \subset SS \end{cases}$$

Ecuación 1. Forma matematica de un problema de optimización.

Donde $f(x)$ es la función por optimizar (maximizar o minimizar), F es el grupo de soluciones posibles y SS es el espacio de soluciones.

Los problemas de optimización se pueden dividir en dos clases: los que tienen respuesta perteneciente a los números reales; y los que donde la respuesta está entre los valores enteros. En este último grupo de los que poseen como solución valores enteros, están los problemas de optimización combinatoria, en los que se busca hallar un objeto entre un conjunto limitado, o como mínimo una cantidad contable, de posibles respuestas. El elemento hallado es comúnmente un número natural, o un conjunto de ellos, un grafo o un subgrafo [2].

Como característica especial de los problemas de optimización combinatoria, siempre existe un algoritmo exacto que permite hallar la respuesta óptima. El método para encontrar la solución favorable, consiste en la investigación profunda del conjunto de respuestas, nombrado como enumeración. [3]. Este algoritmo es con frecuencia muy ineficiente, debido a que según las características de los problemas más importantes que se podrían solucionar con este método, el tiempo que se utiliza para encontrar una respuesta crece de forma exponencial acorde con el tamaño del problema.

De diferentes estudios realizados en la década de 1970, se concluyó que hay un subgrupo de problemas, en el que los algoritmos que los resuelven tienen una complejidad computacional polinómica, esto quiere decir que el tiempo de ejecución de estos algoritmos crecen de forma polinómica con la dimensión del problema. Este tipo de problemas pertenecen a la clase P y se considera por consenso que tienen respuesta de forma eficaz [4] [5]. Pero, para la mayoría de problemas que tienen un interés por su aplicación práctica para responder a problemas de la vida diaria, no existe tal tipo de algoritmo. Cuando se da esta situación, el problema pertenece a la clase NP [6].

Los problemas de optimización combinatoria útiles son difíciles de resolver porque producen resultados que mejoran de manera sustancial algún factor importante dentro de cierto proceso. Por ahora, no hay algún algoritmo que produzca una solución óptima en tiempo polinómico, entonces, no existe una solución en tiempo razonable [2]. Estos problemas se determinan acorde con la dificultad de su respuesta. Existen ciertos problemas que a pesar de no tener un algoritmo que los resuelva en tiempo polinómico, si permiten saber si algún valor corresponde a la respuesta del problema. Esta es una característica de problemas de tipo NP [7]. Los problemas de tipo NP no poseen una respuesta única; sino un grupo de posibles soluciones, donde algunas optimizan el problema. No se logra encontrar una única solución, hay que comprobar cuáles respuestas servirían para hacer la situación actual óptima y realizar las verificaciones pertinentes para comprobar la optimización.

Dentro del grupo de los problemas de tipo NP, existe también otra clase de problemas denominada como $NP - completos$. Para este grupo, no existe un algoritmo polinómico que dé una solución. Los matemáticos no han podido comprobar de manera formal que no existan. Esto significa que un problema x es reducible a otro problema y, cuando se invente un algoritmo que resuelva el problema x utilizando como elemento transformador un algoritmo para resolver el problema y. Lo que indica que debe haber un algoritmo que tome los datos de entrada del problema x, los modifique de tal forma que se puedan utilizar como entrada de un proceso de transformación que resuelva el problema y, y la respuesta para y se logre convertir en una solución para x. Al final, el proceso de transformación que soluciona el problema y puede usarse una sola vez o varias veces dentro del algoritmo que resuelva x [8].

Por último, existe una clase adicional de problemas denominados como $NP - duros$. Estos no hacen parte del grupo de NP problemas; no tienen un algoritmo polinómico que permita comprobar una solución. Para que haya un problema $NP - duro$, debe haber un problema $NP - completo$ que sea reducible a este problema. Lo que significa que debe haber un algoritmo polinómico que sirva como proceso de transformación, que solucione a un problema $NP - duro$ para que se pueda resolver un problema $NP - completo$ [1]. Según Pinedo [9] y Gatica [10] los problemas de Scheduling se considera de complejidad $NP - duro$.

En la Tabla 1 se ilustran algunos ejemplos de problemas de optimización combinatorios, con su respuesta tradicional y con otras soluciones atípicas. El concepto de soluciones tradicionales se refiere a las metodologías que se basan en el uso de algoritmos exactos. Las desventajas que resultan de resolver los problemas de tipo NP con estos métodos son: los algoritmos exactos no tienen tiempos de convergencia en tiempo polinómico; significa que el encontrar la respuesta tardaría tanto que no hace aplicable de manera razonable esta metodología. Otra desventaja es que para algunos problemas de optimización combinatoria, no existe respuesta basada en algoritmos exactos. También, que un algoritmo exacto depende de las variables del problema. En algún momento si se cambian las variables se tendría que crear un algoritmo exacto y nuevo que cumpla con la optimalidad buscada.

Problema de Optimización Combinatorio	**Solución Tradicional**	**Otras Soluciones**
El problema del vendedor viajero	Algoritmos voraces, algoritmos de divide y vencerás	Algoritmos genéticos
El problema de enrutamiento de vehículos	Algoritmos de ramificación y poda	Enfriamiento simulado

Tabla 1. Ejemplos de Problemas de Optimización Combinatorio **[11]**

1.1 Soluciones Tradicionales

Como se menciono anteriormente, las soluciones tradicionales para resolver problemas de optimización combinatoria ofrecen una solución exacta, pero no están hechos para cumplir con el parámetro del tiempo de crecimiento del problema. Esta característica convierte a las soluciones tradicionales en prohibitivas y costosas cuando se aplican en campos medianos o grandes [12].

El scheduling es un problema de optimización combinatoria, en el que se busca asignar recursos a diferentes tareas a través del tiempo, y uno de los fundamentos es el optimizar esta asignación, minimizando o maximizando parámetros de recursos y tiempo. Desde el punto de vista técnico, es difícil hacer un scheduling óptimo. Las dificultades que se observan al hacer scheduling, suelen ser similares a otras que están presentes en otros tipos de problemas de optimización combinatoria y modelado estocástico, normalmente se relacionan con el modelado del problema, y la recopilación de información [9]. A continuación, se mencionan metodologías usadas para darle solución al problema del scheduling. Aunque estos métodos son exactos, han logrado ser una respuesta complaciente a algunos casos en scheduling, a pesar de las desventajas que presentan.

Reglas de Prioridad. Son probablemente la metodología más usada para resolver problemas de scheduling. Consisten en crear reglas que permitan dar prioridad a los task (asignaciones) en la cola para ser procesados por una workstation. Cuando una workstation está libre, una regla elige la siguiente task a ser procesada, por orden de primacía. A pesar de que usar estas reglas da resultados de manera veloz, la calidad del producto no es muy buena. Es muy común encontrar el uso de este método asociado con otros, como inicio para dar elementos de entrada a otras metaheurísticas. [12].

Las reglas de prioridad o primacía, se clasifican en dinámicas y estáticas. Las dinámicas están relacionadas con el tiempo, por lo que cierta task puede tener diferente nivel de prioridad a lo largo de un periodo; mientras que las reglas estáticas dependen de la información de las task y las workstations. La aplicación de estas técnicas se puede observar en el trabajo de Holthaus y Rajendram [13].

Existen unas reglas de prioridad simples, que típicamente son las más usadas y se observan en varios trabajos [13] [14] [15]. Algunos ejemplos de reglas simples son: CP ($camino\ crítico$) es usada cuando los task presenten restricciones de preferencia, se elige el task que tenga el mayor tiempo de proceso, teniendo en cuenta para el cálculo las task anteriores; EDD ($instante\ de\ finalización\ más\ temprano$) que elige el task que tenga la hora de fin más próxima o $MWKR$ ($mayor\ trabajo\ restante$) que escoge al task con un mayor tiempo de proceso faltante. Las reglas de prioridad simples sirven para buscar soluciones aptas a los problemas con un objetivo único, de cualquier forma para problemas con un número mayor de objetivos se pueden utilizar reglas compuestas, que son la combinación de reglas simples con la condición de que cada regla que se utilice debe tener una responsabilidad sobre cada objetivo.

En resumen, utilizar reglas de prioridad como metodología es una de las alternativas más comunes para solucionar problemas de scheduling, pero presenta los problemas típicos de algoritmos exactos, relativos al tiempo polinómico de crecimiento del problema que hace desatinado utilizar este método en universos medianos o grandes.

Programación Dinámica. Este algoritmo puede ser una ventaja al representar una solución favorable para el problema de scheduling porque ante los demás algoritmos exactos, logra responder de manera adecuada ante el requerimiento del crecimiento extraordinario del tiempo de ejecución de respuesta. Otro atributo de la programación dinámica es que guarda los resultados de las interaciones de las ejecuciones del algoritmo para futuros usos, esta característica permite afirmar que la programación dinámica es eficiente [16].

Así mismo, la programación dinámica está basada en la división del problema en subproblemas y la combinación de sus respuestas generadas cada una por separado. Esta metodología es aplicable cuando los subproblemas no están relacionados entre sí. Cuando se ejecuta el algoritmo, cada vez que se resuelve un subproblema el resultado es guardado en una tabla, para evitar que cuando el subproblema reaparezca la respuesta se calcule de nuevo. Esta técnica es apropiada para resolver problemas de optimización porque permite encontrar un valor máximo o mínimo, de acuerdo con lo que se requiera [1].

Por otra parte, una desventaja de la programación dinámica reside en la formulación del problema; porque cada situación requiere un análisis particular. Cuando el problema no está formulado de manera adecuada, entonces posiblemente el algoritmo diseñado no sea el preciso para realizar la búsqueda de respuestas que optimicen. La limitación más importante es la poca capacidad para soportar cantidades significativas de variables de entrada. A medida que aumenta la cantidad de entradas en el algoritmo, también aumentan las respuestas posibles y por lo tanto se complica la opción de encontrar los óptimos. Esta situación se conoce como la maldición de la dimensionalidad [17].

Algoritmo Branch & Bound (NP). Esta metodología es usada para resolver problemas de optimización, puesto que realiza un inventario arbitrario del universo de soluciones basándose en la creación de un árbol de expansión. En un algoritmo de branch & bound se realizan tres fases: la primera es la de selección, en la que se escoge un nodo entre el conjunto de los nodos vivos. Cuando se habla de nodos vivos se hace referencia a un nodo del árbol con posibilidades de ser ramificado, porque no ha sido podado, de allí se escogen los nodos vivos de acuerdo con la estrategia de búsqueda que se decida utilizar. Existen varias opciones para elegir los nodos vivos tales como guardar los nodos generados, pero que no se han examinado en una búsqueda profunda; almacenar nodos vivos, de tal forma que se analicen estos nodos en el orden en el que han sido creados, o también utilizando una función de costo para decidir en cada instante qué nodo debe explorarse para encontrar de la forma más rápida una respuesta mejor que la hallada anteriormente [16].

La segunda fase de la ejecución del algoritmo es la de ramificación. En esta parte se producen los posibles nodos hijos de los nodos seleccionados en la parte anterior. Por último, la tercera fase es la poda, en la que se desechan algunos nodos creados en la etapa anterior y en especial, favorece a reducir el espacio de búsqueda y mitigar la complejidad del algoritmo. Los nodos no podados pasan a ser parte del conjunto de nodos vivos, y se itera la fase de selección. El algoritmo se consuma cuando se encuentra la respuesta o si se termina el conjunto de nodos vivos.

Una de las ventajas de estos algoritmos es que son ejecutables en paralelo, debido a la estructura en nodos del método. Así, se pueden ejecutar las tres fases, en diferentes áreas del árbol, extrayendo nodos, ramificando y podando al mismo tiempo. Sin embargo, la mayor desventaja es el requisito de memoria, que podría ser significativamente mayor que otros algoritmos como la programación dinámica o las reglas de prioridad.

1.2 Soluciones a través de Metaheurísticas

El desarrollo de las metaheurísticas ha tenido un avance muy grande en poco tiempo. Durante los últimos 30 años, desde que se nombró como un concepto sin rigor científico hasta la actualidad, se logran encontrar respuestas de alta calidad a dilemas que en el pasado parecían intratables [20]. El concepto de Metaheurísticas fue utilizado por F. Glover en el año 1986, y con éste definía un "procedimiento maestro de alto nivel que guía y modifica otras heurísticas para explorar soluciones más allá de un simple optimo local" [18]. A partir de la definición original de Glover, se pueden encontrar otras definiciones que buscan complementar este concepto.

Según J.P. Kelly et al. [19], "las Metaheurísticas son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Las Metaheurísticas proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los procedimientos estadísticos". De acuerdo con S. Voss el al. "una metaheurística es un proceso iterativo maestro que guía y modifica las operaciones de una heurística subordinada para producir eficientemente soluciones de alta calidad. Las metaheurísticas pueden manipular una única solución completa (o incompleta) o una colección de soluciones en cada iteración. La heurística subordinada puede ser un procedimiento de alto (o bajo) nivel, una búsqueda local, o un método constructivo".

En resumen, las metaheurísticas buscan emplear un grado de aleatoriedad para encontrar la solución óptima a un problema dado. Por ahora, para varios métodos de este tipo no hay disponible una regulación conceptual que las sustente, únicamente por medio de las respuestas obtenidas en diferentes experimentos se encuentra su razón [20].

Para entender el funcionamiento de las metaheurísticas hay que conocer los conceptos de diversificación e intensificación. La diversificación conceptualiza la explotación del espacio de búsqueda, mientras que la intensificación se refiere a la exploración de la experiencia de búsqueda acumulada. Cada aplicación de una metaheurística está caracterizada por un equilibrio entre estos dos conceptos, que son importantes para identificar regiones en el espacio de búsqueda con soluciones de alta calidad, y también para no desperdiciar tiempo en regiones que ya hayan sido exploradas o donde no se encuentren mejores respuestas.

Existen diferentes maneras de clasificar y describir algoritmos metaheurísticos, y cada uno es el resultado de puntos de vista específicos. De acuerdo con esto, se pueden catalogar estos algoritmos en el grupo de los inspirados en la naturaleza, y en el conjunto de los que no están. Esta agrupación está basada en los orígenes de formación de las metaheurísticas. Algunos ejemplos de algoritmos inspirados en la naturaleza se conocen como computación evolutiva y optimización por colonia de hormigas; al contrario de los que no están inspirados en la naturaleza como son la búsqueda tabú y la búsqueda local iterativa.

Otro tipo de clasificación serían los algoritmos basados en memoria y los que no tienen memoria. Este esquema de agrupación utiliza la manera en la que la metaheurística recolecta la historia de búsqueda; por ejemplo, si para guardar la respuesta de ejecución usa memoria o no. Las metaheurísticas sin memoria usan un proceso Markov para que la información que utilizan determine la próxima acción, sea cual sea el estado actual del proceso de búsqueda.

Metaheurísticas Trayectoriales. Estos algoritmos parten de una solución inicial y forman un camino o trayectoria en el espacio de búsqueda a través de procedimientos de desplazamiento. Las características del camino trazado dan información acerca del comportamiento y efectividad de la metaheurística. A continuación algunas metaheurísticas trayectoriales:

Búsqueda Tabú: Es un proceso de búsqueda local que utiliza su memoria para salir de óptimos locales. Este método se basa en dos afirmaciones: la memoria adaptativa, que se restablece en función del tiempo y análisis de la velocidad; y la búsqueda sensible, que es un movimiento producido después de obtener conclusiones de sucesos anteriores. En un sistema que usa memoria, una mala decisión basada en una estrategia proporcionará claves ventajosas para extender la búsqueda. Al contrario, una buena decisión partiendo de la casualidad no dará información para actos posteriores. En resumen, ese es el concepto de la búsqueda tabú [21] [22]. Una de las características que diferencia la búsqueda tabú de los otros algoritmos es que en este método se busca minimizar la participación de procedimientos al azar. Esta metaheurística se basa en las metodologías sistemática y determinista que llevan a encontrar un óptimo local. Entonces se usa la historia de la búsqueda para huir de óptimos locales y para efectuar la táctica de indagación del espacio de búsqueda.

Recocido Simulado: Es una metaheurística que se basa en la similitud que existe entre un proceso de optimización combinatoria y un proceso termodinámico, conocido como recocido. El fundamento operacional en el que se basa el recocido simulado se enuncia como un algoritmo de búsqueda local que escapa de óptimos locales, permitiendo de acuerdo con algunas circunstancias, aceptar movimientos capaces de empeorar la mejor solución encontrada hasta el instante en el proceso de búsqueda. Esta metaheurística fue desarrollada por Kirkpatrick et al. [23] en la década de 1980 y fue creada mientras buscaban respuesta a problemas de circuitos electrónicos, en los que aplicaron por analogía las características que interfieren entre un proceso termodinámico y un proceso de optimización.

Metaheurísticas Poblacionales. Son los algoritmos que utilizan un conjunto de respuestas (población) en cada repetición de la metaheurística en vez de usar una solución única como las metaheurísticas trayectoriales. Estos algoritmos brindan de forma exclusiva una metodología de exploración paralela del espacio de respuestas, y su validez depende de cómo se maneje la población.

Las metaheurísticas poblacionales comienzan su ejecución desde una población inicial de soluciones. Después, iterativamente aplican la generación de una nueva población y la mejora de la población actual. En la fase de generación, una nueva población de soluciones es creada. En la etapa de reemplazo, un individuo de la población anterior es reemplazado por un individuo de la población nueva. Este proceso se itera hasta que se dé un criterio para detenerlo.

Las fases de generación y reemplazo suelen no usar algún tipo de memoria para guardar la información. Cuando esto sucede, los dos procedimientos están basados sólo en el comportamiento de la población. De otra manera, cuando se usa la memoria, la historia de búsqueda guardada es utilizada para la generación de la población nueva que reemplazará la población anterior. La mayoría de metaheurísticas poblacionales son algoritmos inspirados en la naturaleza. Algunos ejemplos populares de esta clase de algoritmos son: optimización por colonia de hormigas, computación evolutiva, colonia de abejas [24].

Algoritmos evolutivos. La metaheurística de algoritmos evolutivos proviene de los años 1950. En esta década se escribieron textos donde se proponen algoritmos evolutivos sencillos para dar respuesta a problemas combinatorios [25] [26]. En los años sesenta, se proponen las bases de la programación genética y en la década posterior se dictan los fundamentos de las estrategias evolutivas. Estas investigaciones son la base de metaheurísticas contemporáneas, tales como los algoritmos genéticos, algoritmos culturales, inteligencia de enjambre, entre otros [27]. Los algoritmos evolutivos parten de la teoría neodarwiniana de la evolución de las especies, que se puede expresar como: los individuos que poseen una superior adaptación al medio conservan una perspectiva más alta de vivir mayor cantidad de tiempo, con lo que asumirán más posibilidad de formar descendencia que obtenga sus buenas características. Al contrario, los individuos con mala adaptación, tienen inferior probabilidad de subsistir, por lo que poseerán menos oportunidades de concebir familia y, probablemente, acaben agotándose. Este algoritmo inicia una población de N

individuos de acuerdo con criterios determinados. Este grupo se altera usando la aplicación de las operaciones de selección y modificación (mutación y cruce) en continuas repeticiones hasta llegar a una posición de detención. Por último, el mejor individuo se retorna como la respuesta al problema de optimización.

Particle swarm optimization (PSO): James Kennedy and Russell Eberhart, introducen el concepto de PSO en el año 1995 [28], descubierto a través de una simulación de un modelo simplificado social en los cardúmenes de peces o una bandada de pájaros.

El concepto parte de desarrollar la simulación social utilizando una bandada de pájaros. Una población de pájaros fue inicializada aleatoriamente con una posición, con X y Y velocidades. En cada interacción el programa determina para cada pájaro una velocidad en X y Y, teniendo en cuenta el pájaro más cercano. A su vez, se introdujo una variable estocástica llamada "craziness". La variable "craziness" dio al sistema simulado un apariencia más real. Sin embargo, quedaba por fuera ¿Cómo hacer para encontrar alimento? Se definió un vector de XY coordenadas denominado "confield vector". Cada pájaro fue programado para evaluar su posición actual, según los términos de la Ecuación 2. Sin embargo, la ecuación de la posición y velocidad se puede variar dependiendo del problema y del tipo de datos que se manejen.

$$Eval = \sqrt{(presentx - 100)^2} + \sqrt{(presenty - 100)^2} (1)$$

Ecuación 2. Cálculo de velocidad para las partículas en PSO

Cada pájaro debe recordar el mejor valor y la posición XY con la cual obtuvo el mejor valor. El valor se denomina $pbest[]$ y las posiciones $pbestx[]$ y $pbesty[]$. A su vez, cada agente sabe la mejor posición global que un miembro de la bandada había encontrado y su valor. La variable es llamada $gbest$. Se crea un arreglo con las mejores posiciones X - $pbestx[gbest]$ - , Y - y $pbesty[gbest]$ - del grupo. Las velocidades X -$vx[]$- , Y -$vy[]$- se fueron ajustando según un sistema de parámetros denominado $g\_increment$. Con un incremento alto en $g\_increment$, los pájaros se acercaban cada vez más al campo de maíz [28]. A diferencia de ABC, no se establece roles ni especializaciones en la bandada de pájaros (divisiones de labor) [29].

Entre las aplicaciones y estudios realizados, en los que se utilizó PSO para encontrar la mejor solución en gradientes descendientes, se usaron redes neuronales con un porcentaje de aceptación del 89% correcto [30], mientras PSO obtuvo un 92% de aceptación [28].

2. Problemas de Scheduling

Como se vio en la sección 1, los problemas de scheduling hacen parte de los problemas de optimización combinatorios y se establecen como el objeto de estudio en esta sección. Inicialmente, se defininen términos propios de scheduling que se utilizan a lo largo de todo el informe y al final se enumeran algunos tipos de scheduling que existen.

2.1 Términos en Scheduling

A partir del año 1979 los investigadores empiezan a utilizar palabras y notaciones que hasta la época se usan para definir un scheduling y lo estructuran como un conjunto de macrocomponentes: Task (t_j) y workstations (e_i) [31]. Las tasks se definen como unidades de actividades en un proceso, y workstation, como un recurso que se le ha asignado a un task para que sea procesado. Es necesario destacar que los recursos en un problema de scheduling son limitados. [12]. En la Tabla 2 se muestra la notación de los dos macrocomponentes que forman un problema de scheduling, mientras que en la Tabla 3 se muestran algunos conceptos utilizados en Scheduling, que se trabajan a lo largo de todo el documento.

Macrocomponentes	Sinónimos	Notación	Índice	Número máximo
Task	Job	t_j	j	N
Workstation	Machine	e_i	i	M

Tabla 2. Notación de los macrocomponentes en scheduling.

Conceptos	Definición	Notación
Tiempo de procesamiento	Si un task (t_j) requiere procesamiento en una workstation (e_i), entonces p_{ij} representa el tiempo de procesamiento del task (t_j) en la workstation (e_i). El tiempo de procesamiento es igual a la variable aleatoria X_j. La cual se distribuye de forma Erlang con una tasa λ_j.	p_{ij}
Fecha de liberación	Es el momento en el cual el task (t_j) puede empezar a procesarse.	r_j
Fecha de vencimiento	Es el momento en el cual se espera que el task (t_j) sea terminado. La finalización del trabajo después de esta fecha, acarrea un costo monetario.	d_j
Número de trabajos tardíos	Refleja los tasks (t_j) que no se han podido llevar a cabo en un determinado tiempo.	$\sum w_j$
Tiempo de espera.	Tiempo que pasa un taks (t_j) en el sistema antes de iniciar su procesamiento.	v_j

Tabla 3. Conceptos utilizados en scheduling.

Se ilustra más claramente las definiciones con respecto a tiempos que se manejan en scheduling en la Figura 1.

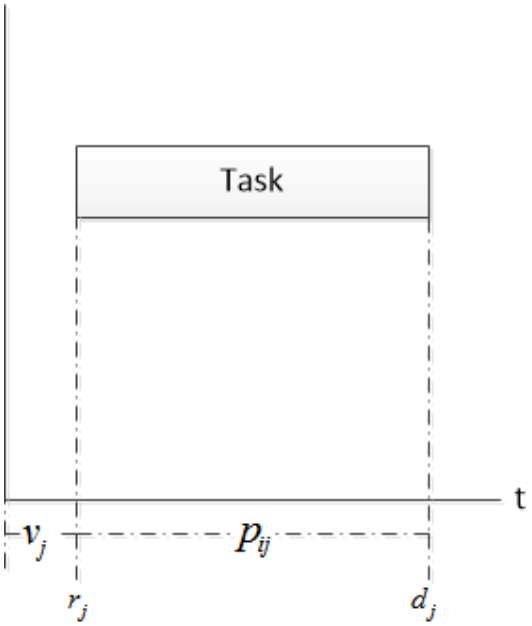


Figura 1. Relación de tiempos con respecto a la ejecución de un task (t_j).

2.2 Tipos de Scheduling

Los cálculos de scheduling son considerados complejos debido a que se requiere mayor capacidad de procesamiento para solucionarlos. En diversas áreas tanto en la industria como a nivel académico se encuentran este tipo de problemas; tal como el arribo de aviones al aeropuerto, la producción en alguna planta de procesamiento, la asignación de recursos a una aplicación que se ejecutará en un computador. Sin embargo, no todos los problemas de scheduling son similares. Dependiendo de las características que este posea existen métodos de solución que pueden facilitar su solución. A continuación se realiza una identificación de los problemas de scheduling a partir de la identificación que se efectuó Delgado [32]:

Workstation Única. Se refiere cuando en el problema de scheduling existe una única workstation que debe ser asignada a un task. Es el problema de scheduling más simple.

Workstation Idénticas en Paralelo (P_m). Hace referencia a las características de los workstations (M) que son idénticas y su ejecución se realiza en paralelo. El task (t_j) que se desea procesar solo posee una etapa de procesamiento y se puede ejecutar en cualquier workstation (e_i).

Workstation en Paralelo con Diferentes Velocidades (Q_m). Se refieren a la condición de existencia en el ambiente de workstations (M) dispuestas paralelamente cuya velocidad varía según el tiempo de procesamiento (p_{ij}) que posee cada task (t_j). Si todas las workstations poseen la misma velocidad se denominaría workstation uniformes.

Workstation No Relacionadas en Paralelo (R_m). Se refieren a la condición de existencia en el ambiente workstations (M) dispuestas paralelamente. La workstation (e_i) puede procesar la task (t_j) a una velocidad (v_{ij}). Es este ambiente la velocidad de la workstation (e_i) es dependiente del task (t_j).

Flow Show (F_m). Existen en el ambiente workstations (M) dispuestas en serie. Todos los tasks (N) deben pasar por todas las workstations secuencialmente. Cuando un task (t_j) deja una workstation (e_i), está debe esperar en la cola de la siguiente workstation (e_i).

Flexible Flow Show (FF_s). Hace referencia a una generalización del flow shop y del ambiente de workstations paralelas. En lugar de workstations (e_i) en serie existen islas con workstation (M) en serie. Cada task debe pasar por una de las islas. Cada isla funciona como workstation (M) en paralelo. Es decir, la asignación del task (t_j) se puede realizar indistintamente a cualquiera de las islas.

Open Show (O_m). Existen tasks ($\mathrm{t_j}$) que deben ser procesadas por cada uno de las workstations ($\mathrm{e_i}$). Sin embargo, no todos los tiempos de procesamiento ($\mathrm{p_{ij}}$) son iguales y hasta algunos pueden ser cero. No existe orden para la asignación de los tasks (t_j) a las workstations (e_i).

Job Show (J_m). Existe tasks (t_j) que deben ser procesadas por los workstations (e_i), con una ruta ya predeterminada.

3. Formulación del Problema de Scheduling HMO

Como se plantea en el subcapítulo anterior, existen diferentes tipos de scheduling que se pueden abordar de diversas maneras. El tipo de scheduling abordado en este estudio, es no convencional; es decir, posee particularidades que no se tienen presentes en otros tipos de scheduling. Debido a esas particularidades es probable que su solución a través de técnicas convencionales sea más dispendiosa. A continuación, se desarrolla el concepto formal del scheduling tratado en este estudio y el porqué se le denomina scheduling HMO.

El scheduling de estudio se puede definir formalmente según la notación $\alpha/\beta/\gamma$ como:

Campo α. Indica cómo se comportan las workstations en el ambiente donde se desarrolla el scheduling. En el caso de estudio se trata el campo α como workstations(M) no relacionadas estocásticas (R_m); es decir, son workstations en paralelo, que pueden procesar los tasks (N) con diferentes velocidades. Los tiempos en los cuales la workstation (e_i) no está disponible se llaman *downtime.* Y son estocásticas debido a que no se conoce inicialmente el número de tasks(N) y tiempos de procesamiento (p_{ij}).

Campo β. Se refiere a las características de los tasks y las restricciones de programación. En este estudio se hace referencia a: restricciones de procedencia ($prec$), debido a que los tasks deben ser completados antes que otros puedan empezar su procesamiento.

Campo γ. Define la función objetivo. En general, lo que se intenta minimizar es una función de tiempo [1]. En el caso de estudio se mínimaza el tiempo de procesamiento (p_{ij}). En la Tabla 4 se extracta la formulación formal del problema de scheduling tratado.

Notación problema	**Ambiente** (α)	**Restricciones** (β)	**Función optimización** (γ)
$R_m \mid prec \mid p_{ij}, v$	Máquinas no relacionadas	Precedencia	Tiempos de procesamiento y Tiempos de espera

Tabla 4. Formulación del problema de scheduling.

No obstante, existen particularidades en el scheduling estudiado que no pueden ser tratadas de forma tradicional. Las particularidades que presenta el scheduling por desarrollar son:

Workstation Humanos (H). Se define en el scheduling estudiado que las workstations (M) son seres humanos. Está particularidad implica incluir en el desarrollo del modelo de planificación predictivo (MPS), características propias de las personas que procesaran el task($\mathrm{t_j}$). Así mismo, los seres humanos van perfeccionando la forma en la que atienden cada uno de los tasks ($\mathrm{t_j}$) que son asignados a ellos. Estas características en particular ayudan a realizar el scheduling de forma inteligente previendo que workstation (e_i) estás mejor capacitada para procesar un trabajo y de esta forma, obtener un menor tiempo en su procesamiento. La capacidad de aprender y mejorar según la experiencia se denomina adaptación.

Mezcla (M). Se define en el scheduling estudiado que los tasks (N) entrantes al modelo de planificación predictivo (MPS), deben ser asignados de forma que se realice una clasificación acerca de qué momento es el adecuado para ser procesados por una workstation (e_i). Esta clasificación se realiza teniendo en cuenta datos históricos acerca del tasks ($\mathrm{t_j}$) y divide los tasks ($\mathrm{t_j}$) en inmediatos y diferidos. Es así como los tasks inmediatos deben entrar de forma instantánea al componente encargado de asignarlos a alguna workstation (e_i), mientras que los tasks diferidos aguardarán en la cola antes de ser asignados. Este mecanismo de clasificación se comporta como un dosificador, que reduce los costos computacionales en la ejecución del algoritmo de scheduling.

Online (O). Al hacer la clasificación entre tasks inmediatos y tasks diferidos, el algoritmo de scheduling debe responder asignando los task inmediatos a una workstation (e_i) de forma online. Aunque en la Tabla 4 el problema tratado en este estudio se define como un problema de workstations no relacionadas paralelas, esa definición no es del todo correcta. Debido a que la velocidad de la workstation (e_i) depende tanto de las características de sí misma, como de las características de los tasks ($\mathrm{t_j}$) que se procesan en ella.

El scheduling tratado en este estudio es de difícil manejo si se implementara en una solución que intente reunir todos los requerimientos. La solución planteada para problemas de scheduling HMO, se produce al tener que reunir bajo un mismo sistema todos los requerimientos planteados y la ejecución e implementación del algoritmo para que se vuelva sencilla.

Para probar la solución planteada en problemas de scheduling HMO se propone como caso de estudio el scheduling de llamadas entrantes y salientes en un Call Center. Las características presentes en el scheduling HMO se cumplen en el caso de estudio. Las workstations (M) serán las personas encargadas de contestar o realizar las llamadas (t_j). De igual forma, la característica mezcla está presente cuando se decide si la llamada debe ser asignada de forma inmediata a una workstation (e_i) o debe esperar en la cola (en general esta clasificación sucede si la llamada a realizar es de entrada o de salida). Si la llamada es de entrada debe ser ejecutada inmediatamente, pero si, la llamada es de salida, esta debe tener una medida de contactabilidad alta, sino debe esperar en la queue (Q).

II – 2. DISEÑO GENERAL DEL PLANIFICADOR PREDICTIVO

A partir de la formulación del problema definida en el capitulo I -1. se realiza el diseño del modelo de planificación predictivo (MPS). Este capítulo consiste en una definición de cómo se aborda el problema de scheduling, reduciendo un problema complejo a partes simples interconectadas entre ellas. Luego se explica cómo se utiliza la jerarquización en el problema de scheduling HMO y se describe las entrada, las salidas y los subcomponentes de los componentes que hacen parte del MPS.

1. Jerarquías de Planificación

En el modelo general propuesto dentro de la investigación se tienen presentes tres niveles de scheduling (planificación). Cada nivel de scheduling es realizado por un componente scheduler (planificador). Los niveles de scheduling son acordes con la frecuencia de ejecución de los tasks (t_j) y se pueden ver de forma jerárquica. Precisamente, en este informe se realiza énfasis en el scheduler a corto plazo, al diseñar e implementar el scheduler correspondiente a este nivel.

Scheduler a Largo Plazo (SLP). Es el componente encargado de asignar recursos a los tasks que no se presentan con frecuencia, pues la decisión de ejecutar estos tasks no es prioritaria en el proceso. A su vez, la decisión que tome este scheduler se complementara con el scheduler a corto plazo. El tipo de tareas que asigne el scheduler a largo plazo depende de la formulación del problema. Según el caso de estudio la asignación que realiza este scheduler será los turnos de los agentes en el Call Center que se incluye como trabajo futuro.

Scheduler a Mediano Plazo (SMP). También denominado Mid Term Scheduler. Se define como el encargado de seleccionar los tasks (t_j) que según su prioridad deben ser ejecutados inmediatamente o al contrario, deben aguardar en la queue. Los tasks (t_j) salientes deben ingresar al scheduler a corto plazo. Una forma de ver al scheduler a mediano plazo es como un dosificador, calcula el nivel de ocupación del scheduler a corto plazo (SCP), y decide, cuáles tasks (t_j) deben ser ejecutadas inmediatamente o cuáles pueden esperar en la queue para su posterior ejecución. Según el caso de estudio el scheduler a mediano plazo dosificara las llamadas y priorizara su ejecución dependiendo de factores tales como: tipo de llamada (entrante, saliente), ocupación de los agentes, entre otros.

Scheduler a Corto Plazo (SCP). También denominado Short Term Scheduler o Dispatcher. Se define como el scheduler encargado de la asignación de recursos en tiempo real, según el número de tasks salientes del scheduler a mediano plazo (n_{SMP}). El comportamiento de tasks (t_j) entrantes es estocástico. Es el scheduler más relevante entre los tres tipos de schedulers tratados.

2. Arquitectura General del Planificador

La arquitectura general está descrita en el sistema que se divide en cuatro componentes: 1) componente adaptativo, 2) componente scheduler a mediano plazo, 3) componente scheduler a corto plazo y 4) componente scheduler a largo plazo. En la Figura 2. se muestra el detalle de los componentes mencionados. Se definen como componentes auxiliares el componente adaptativo y el monitor de contexto. El scheduler a largo plazo aunque se muestra en la figura no se detalla en este informe.

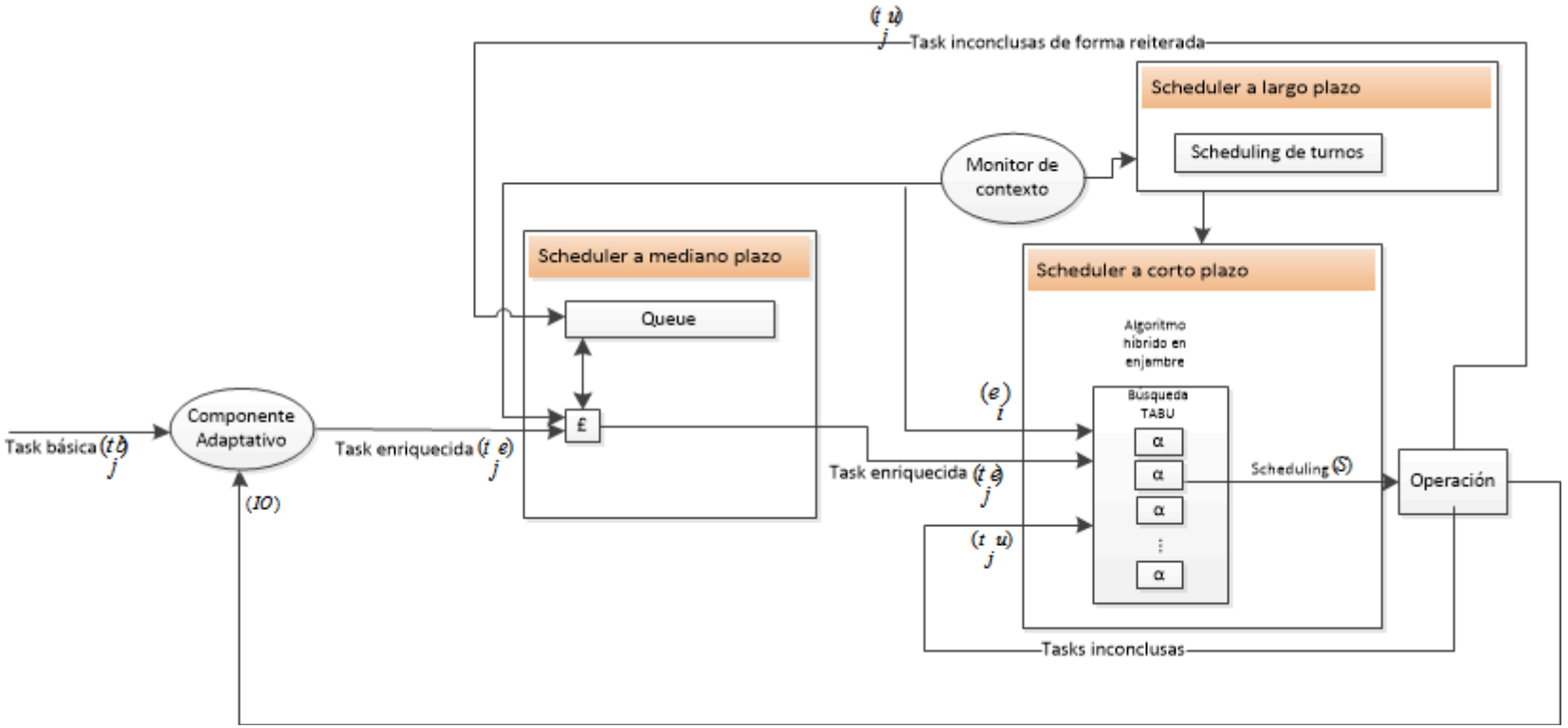

Task inconclusas de forma reiterada
Scheduler a largo plazo
Scheduling de turnos
Monitor de contexto
Scheduler a mediano plazo
Queue
Scheduler a corto plazo
Algoritmo híbrido en enjambre
Búsqueda TABU
Componente Adaptativo
Task básica
Task enriquecida
Task enriquecida
Scheduling
Operación
Tasks inconclusas

Figura 2.

Componente Adaptativo. Es el responsable de aportar las características deseables de la workstation (e_i) que procesa un task básico ($t_j b$). La salida es un task enriquecido ($t_j e$) que se construye a partir del perfil (pf) correspondiente al task básico ($t_j b$) y por medio del mecanismo de adaptación y adaptador se mejora. Se define perfil (pf) como el conjunto de características que posee una persona [33].

Scheduler a Largo Plazo. Es el responsable de disponer en periodos de tiempo determinado la cantidad de workstation (e_i) que son necesarias para procesar las tasks entrantes (t_j) al sistema. Este concepto se conoce como creación de turnos, pero en este caso de estudio esta función se llamara dimensionamiento de la malla de turnos de los agentes. Este scheduler no se trata con detalle en este trabajo.

Scheduler a Mediano Plazo. Es la herramienta docificadora que define cuales tasks (t_j) transiciona al scheduler a corto plazo o cuáles deben aguardar en una queue para ser procesados posteriormente. Esta decisión es apoyada por el monitor del contexto, siendo el componente que aporta información acerca de los cambios en el ambiente donde se realiza el scheduling.

Scheduler a Corto Plazo. Es el encargado de definir a través de un algoritmo híbrido en enjambre cuál workstation (e_i) será la que asume la labor de procesar el task enriquecido ($t_j e$). El espacio de búsqueda del algoritmo se forma por el task enriquecido ($t_j e$) aportado por el componente adaptativo y las salidas del monitor del contexto.

En el caso de estudio, se definen los tasks básicos ($t_j b$) como llamadas entrantes y salientes en las líneas telefónicas. Los tasks básicos ($t_j b$) entran al componente adaptativo, donde por medio del manejador de perfiles se busca el perfil de la persona que está llamando o a la cual se desea llamar. Los perfiles guardados son datos históricos de las personas contactadas, recolectados por el subcomponente constructor de perfiles. El mecanismo de adaptación recibe el perfil acorde con la llamada y concluye cuáles características deseadas debe tener un agente (e_i) para atender esta llamada. Se define agente como la persona encargada de atender las llamadas tanto de tipo de salida como de entrada, dentro de un Call Center, y serán estos agentes en el caso de estudio sinónimo de workstation (e_i). El subcomponente adaptador es el encargado de orquestar el envío de información a otros subcomponentes, y también es el encargado de enviar a la salida del componente adaptativo el task enriquecido ($t_j e$). A las características definidas por el mecanismo de adaptación, más alguna información relevante del task básico ($t_j b$), se define como task enriquecido ($t_j e$).

Posteriormente, el task enriquecido ($t_j e$) pasa al scheduler de mediano plazo. Según la información aportada por el task enriquecido ($t_j e$) y la información (IC) del monitor del contexto, el scheduler de mediano plazo decidirá a través del subcomponente divisor de task enriquecidos (£), cuáles llamadas deben pasar al scheduler de corto plazo y cuáles deben esperar en la queue. La razón por la cual hay llamadas que deben esperar, es por las características del perfil de la persona contacto (disposición de tiempo para atender llamadas, capacidad de dar información en determinado momento, ocupación de los agentes (e_i), entre otros) y el tipo de llamada, si es entrante o saliente. A la queue también entran llamadas que no han podido ser gestionadas por el scheduler a corto plazo de forma reiterada ($t_j i$) .

Las llamadas seleccionadas para ser procesadas por el scheduler a corto plazo (n_{SMP}), son la suma del task enriquecido ($t_j e$), los task enriquecidos que aguardaban en la queue ($t_j eq$), y los task inconclusos ($t_j i$). Las llamadas (n_{SMP}) son tratadas por el subcomponente algoritmo híbrido de enjambre (AHE). El algoritmo híbrido de enjambre decide cuál agente (e_i) que cumpla con las características del task enriquecido ($t_j e$) y a su vez que minimice el tiempo de procesamiento

(p_{ij}), atienda la llamada (t_j). La información y características de los agentes disponibles (IC) son aportadas por el componente monitor del contexto. Como salida del scheduler a corto plazo se tiene una asignación de recursos (m) para atender una serie de llamadas (n_{SMP}) que se denomina scheduling (s).

El scheduling (s) es ejecutado por la operación. Al terminar el proceso, el constructor de perfiles perfecciona el perfil de la persona contacto y el perfil del agente, por medio de la entrada de nuevos datos recolectados en la operación (IO).

En las siguientes secciones se detalla cada uno de los componentes mostrados en la Figura 2.

3. Componente Adaptativo

El componente adaptativo es la primera parte del sistema diseñado. Posee varias entradas, subcomponentes y salidas, que permiten entregar a la siguiente etapa del proceso el task enriquecido ($t_j e$); fundamental para lograr el scheduling óptimo (s). Los subcomponentes cumplen las labores de procesar las variables de entrada, con el objetivo primordial de brindar las mejores características para la designación del workstation (e_i), como base para ejecutar los pasos posteriores. En la Figura 3 se muestran los detalles internos de cómo se estructura el componente adaptativo.

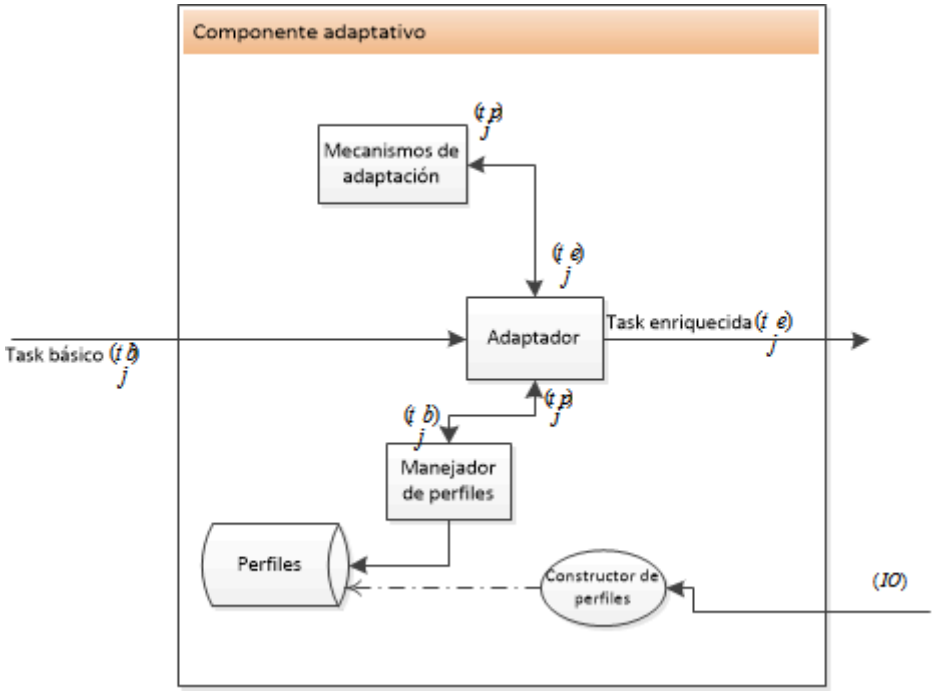


Figura 3. Detalle interno del componente adaptativo.

La variable de entrada que participa en el componente adaptativo se puede entender de la siguiente manera:

Task Básico ($t_j b$). Formalmente se define como la tupla: [Número de Teléfono, Tipo de Teléfono] para el caso de estudio. Esta variable es la entrada al componente adaptativo y su valor determina la búsqueda del perfil (pf) con el cual se construye el task enriquecido ($t_j e$).

Es importante anotar que también hay una retroalimentación de perfiles, procedente de la ejecución del último paso del sistema; se verá más adelante dentro del caso de estudio.

La variable de salida que se produce en el componente adaptativo se delimita conceptualmente así:

Task Enriquecido ($t_j e$). Formalmente se define como la tupla: [Teléfono, Percentil 1 hora de llamada, Percentil 2 hora de llamada, Percentil 3 hora de llamada, Tema 1, Tema 2, Mínima calificación de servicio, Máxima calificación de servicio, Mínimo nivel de tolerancia, Máximo nivel de tolerancia] para el caso de estudio. Esta variable es la salida del componente adaptativo y su valor corresponde a características deseables que debe tener el workstation (e_i), encargado de procesar el task básico ($t_j b$). En la Tabla 5 se definen los conceptos que hacen parte del task enriquecido ($t_j e$) según el caso de estudio.

Procedencia	**Concepto**	**Definición**
Perfil persona contacto (pf_c)	Teléfono	Número telefónico del cual se comunica o al cual se desea llamar la persona contacto.
Componente adaptativo o Perfil persona contacto (pf_c)	Percentil 1 hora de llamada	Primer percentil de la variable hora de contacto.
	Percentil 2 hora de llamada	Segundo percentil de la variable hora de contacto.
	Percentil 3 hora de llamada	Tercer percentil de la variable hora de contacto.
	Tema 1	Es el tema general por el cual la persona se comunicó a la línea o la línea se comunicó con la persona contacto.
	Tema 2	Es el tema específico por el cual la persona se comunicó a la línea o la línea se comunicó con la persona contacto.
Componente adaptativo	Mínima calificación de servicio	Valor que se da entre 0 y 1 al servicio que proporciona el agente. Este valor es el límite mínimo.
	Máxima calificación de servicio	Valor que se da entre 0 y 1 al servicio que proporciona el agente. Este valor es el límite máximo.
	Mínimo nivel de tolerancia	Valor que se da entre 0 y 1 al nivel de tolerancia que posee el agente. Este valor es el límite mínimo.
	Máximo nivel de tolerancia	Valor que se da entre 0 y 1 al nivel de tolerancia que posee el agente. Este valor es el límite máximo.

Tabla 5. Task Enriquecido ($\mathbf{t_j e}$).

Los subcomponentes pertenecientes al componente adaptativo se describen de la siguiente forma:

Adaptador. Definido formalmente como $t_j e = Adaptador(t_j b, pf)$. Es el subcomponente principal del componente adaptativo. Se encarga de recibir el task básico ($t_j b$), orquesta la ejecución de labores que realiza cada subcomponente, y de manera posterior envía como salida del proceso el task enriquecido ($t_j e$).

Para el caso de estudio, el adaptador recibe el task básico $(t_j b)$, enviando los datos al manejador de perfiles, donde se ejecuta la labor correspondiente y posteriormente toma la respuesta para ser remitida al mecanismo de adaptación. Cuya respuesta es la salida del adaptador $(t_j e)$.

Manejador de Perfiles. Definido formalmente $pf = perfilDataBase(t_j b)$. Subcomponente encargado de buscar los perfiles relacionados con el task básico $(t_j b)$. Las salidas deben ser un perfil (pf) o un perfil generalizado, este último se refiere al perfil que se construye ajustado a los datos históricos cuando no se encuentra el perfil relacionado con el task básico $(t_j b)$. Al manejador de perfiles ingresan los datos provenientes del adaptador.

En el caso de estudio, el manejador de perfiles recibe el task básico $(t_j b)$, enviado por el adaptador y busca o crea el perfil de la persona contacto (pf_c) que más se ajuste, y retorna la respuesta al adaptador.

Perfiles. Definido formalmente como la tupla: [Género, Edad, Incumplimientos, Calificación, Tipo de persona, Percentiles, Cuantas llamadas, Tiempo promedio de atención, Tipo de llamada, Tema 1, Tema 2] Agrupación de características y métricas que están relacionadas con los task básicos $(t_j b)$. Los perfiles son creados a partir de datos históricos que se recolectan después de la ejecución del scheduling (s).

Para el caso de estudio, el perfil de la persona contacto está delimitado a las variables a continuación: género, edad, incumplimientos, calificación, tipo de persona, percentil 1 hora de llamada, percentil 2 hora de llamada, percentil 3 hora de llamada, cuántas llamadas, tiempo promedio de atención, tipo de llamada, tema 1 y tema 2. En la Tabla 6 se definen las características y métricas que forman el perfil de la persona contacto (pf_c).

Concepto	Definición
Género	Masculino o femenino. Es la condición orgánica de la persona contacto [33].
Edad	Tiempo que ha vivido la persona contacto [33].
Incumplimientos	Falta de cumplimiento [33]. Se refiere si ha existido algún incumplimiento en un trámite que solicitó la persona contacto.
Calificación	Es la puntuación obtenida de la persona contacto según el trato dado a los agentes.
Tipo de persona	Se refiere si la persona se encuentra pensionada o es beneficiario de alguna pensión por derecho.
Percentil 1 hora de llamada	Primer percentil de la variable hora de contacto.
Percentil 2 hora de llamada	Segundo percentil de la variable hora de contacto.
Percentil 3 hora de llamada	Tercer percentil de la variable hora de contacto.
Cuantas llamadas	Suma de las llamadas que la persona contacto ha realizado a la línea.
Tiempo promedio de atención	Tiempo promedio que se ha demorado un agente en atender la llamada.
Tipo de llamada	Si la llamada se realiza a la línea o es la línea quien realiza la llamada.
Tema 1	Es el tema general por el cual la persona se comunicó a la línea o la línea se comunicó con la persona contacto.
Tema 2	Es el tema específico por el cual la persona se comunicó a la línea o la línea se comunicó con la persona contacto.

Tabla 6. Perfil de la persona contacto $(\boldsymbol{pf_c})$.

Constructor de Perfiles. Construye o actualiza automáticamente perfiles en el sistema a partir de la información recolectada después de ser ejecutado el scheduling (s). El constructor de perfiles entrará en funcionamiento después que el proceso de scheduling (s) haya concluido. En su interior contará con metadatos para la definición correcta de los perfiles.

Para el caso de estudio, el constructor de perfiles toma los datos recolectados de la operación y los convierte de acuerdo con los metadatos en perfiles de las personas contacto (pf_c). Realiza funciones estadísticas para determinar características de los perfiles.

Mecanismo de Adaptación. Se define formalmente como $t_j e = ProcessJoRich(pf, t_j b)$. Contiene reglas y funciones los cuales buscan al ser ejecutadas enriquecer los task básicos $(t_j b)$.

En el caso de estudio, el mecanismo de adaptación toma el perfil de la persona contacto (pf_c) y a través de un conjunto de reglas se determina que características deseables debe poseer la workstation (e_i) que procesara el task enriquecido ($t_j e$).

4. Scheduler a Mediano Plazo

El scheduler a mediano plazo es la segunda etapa del sistema de planificación. Sus entradas provienen de diferentes orígenes, pero todas dependen del resultado de actividades presentes en otras etapas del sistema. Los subcomponentes son decisorios para el desarrollo del scheduling, porque se toman disposiciones que determinan si se envía al scheduler de corto plazo el task enriquecido ($t_j e$).

Las variables de entrada que participa en el scheduler a mediano plazo se especifican de la siguiente manera:

Task Enriquecido ($t_j e$). Formalmente se define como la tupla: [Teléfono, Percentil 1 hora de llamada, Percentil 2 hora de llamada, Percentil 3 hora de llamada, Tema 1, Tema 2, Mínima calificación de servicio, Máxima calificación de servicio, Mínimo nivel de tolerancia, Máximo nivel de tolerancia] para el caso de estudio. Esta variable es la entrada al scheduler de mediano plazo y su valor corresponde a características deseables que debe tener el workstation (e_i), encargado de procesar el task básico ($t_j b$).

Tasks Inconclusos de Forma Reiterada ($t_j u$). Se definen formalmente como la tupla: [Teléfono, Percentil 1 hora de llamada, Percentil 2 hora de llamada, Percentil 3 hora de llamada, Tema 1, Tema 2, Mínima calificación de servicio, Máxima calificación de servicio, Mínimo nivel de tolerancia, Máximo nivel de tolerancia] para el caso de estudio. Se contextualizan como task enriquecidos ($t_j e$) que pasaron al scheduler de corto plazo para su ejecución, sin embargo, por motivos extrínsecos no se procesaron o se interrumpió su procesamiento de forma reiterada. Es decir; el scheduler a corto plazo intento varias veces ejecutar los task enriquecidos ($t_j e$), no obstante, no fue posible la finalización exitosa.

Para el caso de estudio los tasks inconclusos de forma reiterada ($t_j u$) representan primero, a las personas contacto con las cuales no fue posible la comunicación telefónica después de reiterados intentos de marcación. Segundo, las llamadas que en algún momento del proceso de ejecución se cortó la comunicación en repetidas ocasiones.

Información del Monitor del Contexto (IC). Se define formalmente como la tupla: [Día, Mes, Año, Nombre día semana, Festivo, Hora, Minuto, Segundo, Cantidad agentes disponibles] para el caso de estudio. Informa al scheduler a mediano plazo sobre factores ambientales o del contexto que pueden influir en la decisión de si se envía el task enriquecido ($t_j e$) al scheduler a corto plazo

o si el task enriquecido ($t_j e$) debe esperar en la queue. También informa la cantidad de workstation (e_i) disponibles.

Para el caso de estudio el monitor del contexto informa variables tales como el día, si es festivo, la hora en formato 24 horas, entre otros. Teniendo en cuenta la anterior información se decide cuales tasks enriquecidos ($t_j e$) deben pasar el scheduler a corto plazo. La información del monitor del contexto (IC) lo usa el subcomponente divisor de task enriquecidos (£) para calcular la probabilidad de contactabilidad de las llamadas tipo saliente y para retener las llamadas en la queue si no existen recursos disponibles para procesarlas.

La variable de salida que se produce en el scheduler a mediano plazo se contextualiza así:

Tasks enriquecidos del scheduler a mediano plazo (n_{SMP}). Se definen formalmente como la tupla: : [Teléfono, Percentil 1 hora de llamada, Percentil 2 hora de llamada, Percentil 3 hora de llamada, Tema 1, Tema 2, Mínima calificación de servicio, Máxima calificación de servicio, Mínimo nivel de tolerancia, Máximo nivel de tolerancia] para el caso de estudio. Son la suma de los task enriquecido ($t_j e$) que el divisor de task enriquecidos (£) direcciona inmediatamente al scheduler a corto plazo y los task enriquecidos que aguardaban en la queue ($t_j eq$).

Los subcomponentes pertenecientes al scheduler a mediano plazo se describen de la siguiente forma:

Queue. Cola que retiene los tasks enriquecidos ($t_j e$), donde el divisor de task enriquecidos (£) decide que deben esperar para ser enviados a el scheduler a corto plazo. También contiene los tasks Inconclusos de Forma Reiterada ($t_j u$).

Para el caso de estudio, la queue contiene en general llamadas de tipo saliente cuya probabilidad de que la persona contacto atienda sea baja, con las condiciones dadas por el monitor del contexto (IO). También contiene las llamadas inconclusas reiteradamente.

Divisor de Tasks Enriquecidos (£). Subcomponente encargado de decidir teniendo en cuenta la información que incluye el task enriquecido ($t_j e$) y la información proveniente del monitor del contexto (IO), cuales tasks enriquecidos ($t_j e$) provenientes del componente adaptativo se envían al scheduler a corto tiempo o cuales se envían a la queue. Los tasks enriquecidos ($t_j e$) que esperan en la queue se denominan tasks diferidos. Mientras los tasks enriquecidos que son enviados al scheduler a corto plazo se denominan tasks inmediatos. El divisor de tasks enriquecidos también es el encargado de decidir cuales tasks enriquecidos ($t_j e$) o Tasks Inconclusos de forma reiterada ($t_j u$) que se encuentran en la queue, pueden salir y pasar al scheduler de corto plazo. La última función descrita se realiza continuamente.

Para el caso de estudio el divisor de task enriquecidos (£) separa las llamadas según su probabilidad de contactabilidad. La probabilidad de contactabilidad se calcula de los datos provenientes del task enriquecido ($t_j e$) y la información del monitor del contexto (IO). Las llamadas tipo entrada generalmente pasan directamente al scheduler a corto plazo. En el caso tal que no existan recursos disponibles para el procesamiento de la llamada, la llamada debe ser ingresada a la queue. Así se evita sobrecargar al scheduler a corto plazo.

Componente Monitor del Contexto. Definido formalmente como $e_i = findContext()$. Componente encargado de consultar información relacionada con el ambiente y enviarla al scheduler a mediano plazo y a corto plazo.

5. Scheduler a Corto Plazo

El scheduler a corto plazo es la tercera y última etapa del sistema de planificación. También cuenta con las variables de entrada que participa en el scheduler a corto plazo se especifican de la siguiente manera:

Tasks Enriquecidos del Scheduler a Mediano Plazo (n_{SMP}). Se definen formalmente como: [Teléfono, Percentil 1 hora de llamada, Percentil 2 hora de llamada, Percentil 3 hora de llamada, Tema 1, Tema 2, Mínima calificación de servicio, Máxima calificación de servicio, Mínimo nivel de tolerancia, Máximo nivel de tolerancia] para el caso de estudio. Son la suma de los task enriquecido ($t_j e$) que el divisor de task enriquecidos (£) direcciona inmediatamente al scheduler a corto plazo, los task enriquecidos que aguardaban en la queue ($t_j eq$) y los tasks reiterativos ($t_j i$). Los tasks reiterativos ($t_j i$) son aquellos tasks a los cuales se les asigna recursos para su ejecución; sin embargo, por motivos intrínsecos no finalizaron correctamente. Estos tasks (t_j) son enviados de nuevo a la entrada del scheduler a corto plazo, para asignarles recursos nuevamente. En el caso de que la ejecución falle de forma frecuente, estos tasks (t_j) se convierten en tasks inconclusos de forma reiterada ($t_j u$).

La variable de salida que se produce en el scheduler a corto plazo se contextualiza así:

Scheduling (s). Se define formalmente como [Tasks(t_j), Workstation (e_i)]. Es la asignación óptima de recursos también denominados workstation (e_i) a una serie de tasks (t_j) que se desean ejecutar. El scheduling debe ser implementado dentro de un proceso productivo.
Para el caso de estudio, el scheduling es una matriz $M \times N$ que representa las llamadas y el agente al cual fue asignado para su proceso.

Los subcomponentes pertenecientes al scheduler a corto plazo se describen de la siguiente forma:

Algoritmo Híbrido de Enjambre (AHE). Definido formalmente como $[\text{Tasks}(t_j), \text{Workstation } (e_i)] = ahe(t_j e, IC)$. Subcomponente encargado de encontrar el scheduling optimo (s). Básicamente se compone de dos subcomponentes internos quienes realizan la labor de buscar cuál task (t_j) debe ser ejecutado por cuál workstation (e_i) optimizando el tiempo de procesamiento (p_{ij}) cuando deben ser ejecutados un límite máximo de tasks (N).

Para realizar el algoritmo híbrido de enjambre se tuvo en cuenta las siguientes restricciones:

a) Cada task (t_j) debe ser asignado a una workstation (e_i) y viceversa.
b) Un task (t_j) no puede ser asignado a una workstation (e_i) no disponible.
c) Cuando un task (t_j) es asignado a una workstation (e_i) para su procesamiento. No es permitido que exista expropiación por parte de otro task (t_j).
d) Tasks enriquecidos ($t_j e$) deben ser los más cercanas posibles a la workstation real que procesa el trabajo.

Los subcomponentes pertenecientes al híbrido de enjambre (AHE) se muestran en la Figura 4 y se definen de la siguiente forma:

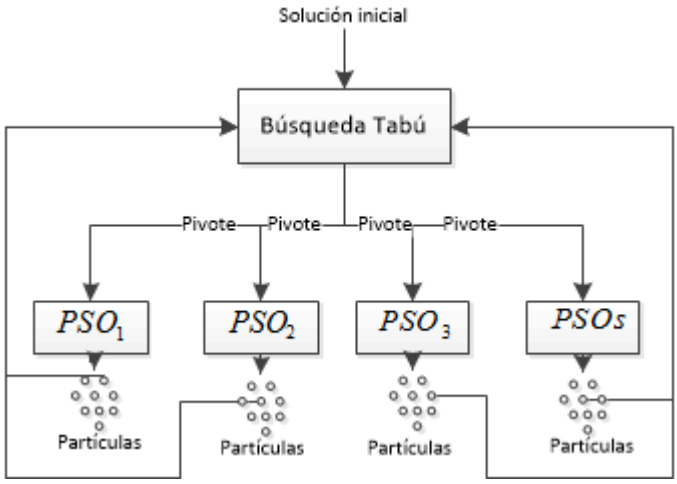


Figura 4. Subcomponentes del algoritmo híbrido de enjambre (***AHE***).

Búsqueda Tabú (TS). Definida formalmente como $[\text{Tasks}(t_j), \text{Workstation}\ (e_i)] = ahe(t_j e, IC)$. Metaheurística basada en búsqueda local que explora el espacio de búsqueda más allá de una optimización local. El termino Búsqueda Tabú fue acuñado en el año 1978 por Glover. La Búsqueda Tabú está basada en la incorporación de memoria adaptativa y exploración sensible. Implementada en gran cantidad de aplicaciones tales como: scheduling, telecomunicaciones, ruteo, entre otros [34].

La Búsqueda Tabú empieza con una solución generalmente creada al azar. Solución que se sitúa en una vecindad[1]. Cada solución se calcula por una vecindad dada moviéndose así en un espacio de soluciones. En este caso, una vecindad contendrá un enjambre que se encargara de la búsqueda. Entonces, el número de vecindades es igual a número de enjambres que se desean crear. Debido a que la Búsqueda Tabú usa estructuras de memoria para almacenar las mejores soluciones de las vecindades, permite no caer en un óptimo local y realizar estrategias de exploración. Es decir, las estructuras de memoria restringen los movimientos hacia las soluciones allí guardadas [1]. La Búsqueda Tabú implementada guarda la solución completa encontrada en una vecindad por el algoritmo PSO, este tipo de estructura de memoria se denomina explicita.

La Búsqueda Tabú al ser un algoritmo de búsqueda local posee la desventaja de que la mayoría de tiempo de búsqueda lo consumen buscando en espacio de soluciones muy pequeñas [1]. Para solucionar este inconveniente se opta por utilizar un algoritmo de enjambre de partículas (PSO).

El criterio de parada de la Búsqueda Tabú implementada está dado por el número de iteraciones que se desea realizar el proceso (IT). No obstante, existen otros criterios de parada que se podrían tener presentes tales como: establecer un tiempo máximo de ejecución del algoritmo, no mejora de la solución después de tres o más iteraciones, entre otros.

Los elementos que componen la Búsqueda Tabú son:

a) Solución inicial. La solución con la cual inicia la Búsqueda Tabú. Se implementa aleatoriamente.

[1] Dado un espacio de búsqueda S para un problema, se define una función $dist(x, y)$. A partir de esta función se puede definir una vecindad $N(s) \subseteq S$ de la solución [1].

b) Soluciones pivote. La Búsqueda Tabú crea soluciones a partir de la solución inicial y teniendo en cuenta las soluciones contenidas en las estructuras de memoria Tabú. La cantidad de soluciones creadas son según la cantidad de enjambres (E) que se desean crear.
c) Estructura de memoria Tabú. Lista donde se almacena las soluciones ya encontradas en una vecindad por el algoritmo PSO. Al general las soluciones pivote, son tenidas en cuenta como condición de rechazo las soluciones que sean idénticas a las ya contenidas en la estructura de memoria. En el algoritmo implementado la estructura de memoria Tabú se define con una longitud constante.
d) Scheduling. Al finalizar su ejecución el algoritmo implementado, su resultado será el scheduling; es decir, un vector $(N|M) \times 2$, el cual representa lo tasks (t_j) que fueron asignados a un workstation (e_i) respectivo.

Optimización por Enjambre de Partículas (PSO) (α). Definido formalmente como $[\text{Tasks}(t_j), \text{Workstation } (e_i)] = ahe(Pivote, IC, t_j e)$. PSO pertenece a la familia de algoritmos en enjambres, que se basan en el comportamiento colectivo de las sociedades. En la naturaleza se encuentra gran cantidad de ejemplos tales como las hormigas, los pájaros, el krill, entre otros [1]. PSO se basa en el comportamiento de una bandada de pájaros y su principal componente son las partículas. Uno de los primeros usos que se le dio a PSO fue el remplazo del algoritmo Backpropagration [28].

Entre las principales características de PSO se pueden destacar: asume un intercambio de información local provocando así que el comportamiento de las partículas este influenciado por las partículas vecinas, cada partícula decide su mejor dirección teniendo en cuenta por las que paso en el pasado y suele tener una rápida convergencia [35].

En el algoritmo implementado, PSO no forma sus partículas aleatoriamente sino a partir de una solución enviada por la Búsqueda Tabú, que se denominara solución pivote. La solución pivote se genera teniendo en cuenta las estructuras de memoria de la Búsqueda Tabú. PSO realiza estrategias de explotación del espacio de búsqueda, complementando así la Búsqueda Tabú [35]. La mejor solución del enjambre es enviada a la Búsqueda Tabú para que sea guardada en las estructuras de memoria.

Los elementos que componen la Optimización por Enjambre de Partículas son:

a) Solución inicial. La solución con la cual inicia su ejecución PSO es la solución enviada por la Búsqueda Tabú denominada solución pivote.
b) Función objetivo. Se define como una ecuación cuya finalidad es de calcular la calificación de la solución según las limitaciones o restricciones impuestas. La función objetivo está dada por la Ecuación 3.

$$\min f(s) = \sum_{j|i} d\left(\overrightarrow{CR}_{ij}, \overrightarrow{CD}_{ij}\right) \times \beta_1 + p_{ij} \times \beta_2 + v_j^2 \times \beta_3$$

Ecuación 3. Función objetivo de la Optimización por Enjambre de Partículas.

Donde:

- $\overrightarrow{CR}_{ij}$. Vector de características del workstation (e_i) real. Se denomina workstation (e_i) real como aquel recurso asignado para realizar una task (t_j) en las soluciones que se van creando cuando se ejecuta el algoritmo PSO. Este vector cambia según la solución propuesta por PSO.
- $\overrightarrow{CD}_{ij}$. Task enriquecidos $(t_j e)$ del workstation (e_i) deseada. Este vector es inmutable para un task (t_j) en todo el proceso.

- p_{ij}. Tiempo de procesamiento.
- v_j. Tiempo de espera de un task en el sistema.
- β_1, β_2 y β_3. Ponderación que se le asigna a cada uno de los términos por los cuales se multiplica, según su importancia en el proceso de scheduling.

La función objetivo propuesta se compone de tres términos, los cuales están ponderados según las constantes de multiplicación β_1, β_2 y β_3. A continuación se explica cada uno de los términos:

1) $d(\overrightarrow{CR_{ij}}, \overrightarrow{CD_{ij}})$. Distancia entre $\overrightarrow{CR_{ij}}$ y $\overrightarrow{CD_{ij}}$. A menor distancia serán más similares los vectores.
2) p_{ij}. Tiempo en que una workstation (e_i) procesa un task (t_j) . En este cálculo se tiene en cuenta si el task en una pasada iteración (k) no fue seleccionado para ser ejecutado por una workstation (e_i).

   p_{ij} es calculado según la Ecuación 4.

   $$p_{ij} = p_{ij} + \left|(p_{ij} - pe_{ij})_{k-1}\right|$$

   Ecuación 4. Cálculo tiempo de procesamiento $\boldsymbol{p_{ij}}$.

   Donde:

   - k. Número de iteración en que se halla el task (t_j) para su asignación a una workstation (e_i).
   - p_{ij}. Tiempo de procesamiento del task (t_{jk}) en la workstation (e_i).
   - $\left|(p_{ijk} - p_{ij(k-1)})\right|$, Tiempo de procesamiento ejecutado del trabajo (t_{jk-1}) desde que se asignó el task (t_{jk}) en la misma workstation(e_i), hasta la hora de liberación (r_j) del task (t_{jk-1}).

3) v_j^2. Penalización cuadrática con respecto al tiempo de espera (v) del task (t_j) .

c) Partículas (P). Unidad mínima del algoritmo PSO. Un conjunto de partículas se llama enjambre. Cada partícula representa una solución en el espacio de búsqueda.

   En el algoritmo implementado la partícula se compone de tres vectores:

   1) Vector $x_i[t_j, e_i]$. Representa la posición que posee la partícula en el espacio de búsqueda. La posición representa el task (t_j) a que workstation (e_i) ha sido asignado para su procesamiento.
   2) Vector velocidad. Almacena el gradiente según el cual se moverá la partícula [35].
   3) Vector función $pBest$. Almacena la mejor solución encontrada hasta el momento.

d) Cálculo de velocidad. La velocidad indica hacia qué dirección se debe mover la partícula. Para el algoritmo implementado se utiliza la Ecuación 5 para calcular la velocidad.

   $$v_i \leftarrow v_i + \varphi_1(pBest_i - x_i) + \varphi_2(g_i - x_i)$$

   Ecuación 5. Cálculo de velocidad para algoritmo PSO implementado **[35]**.

   Donde:

- v_i. Velocidad actual de la partícula.
- φ_1 y φ_2. Números aleatorios en un intervalo de [0,1].
- $pBest_i$. Mejor posición hasta el momento.
- g_i. Mejor posición global hasta el momento.
- x_i. Posición actual de la partícula.

Después de explicar por separado las características de cada algoritmo que compone el Algoritmo híbrido de enjambre (AHE), se enseña en Algoritmo 1 el seudocódigo donde se aprecia la conexión que existe entre los dos algoritmos y cómo se colaboran para su correcta ejecución.

Inicialización de variables

Int M //número total de workstations
Int N //número total de tasks
Int P //número total de partículas
Int E //número total de enjambres
Int S //longitud lista tabú
Int D=2 //dimensión
Int IT//número de iteraciones de la lista tabú
Int IE//número de iteraciones en el enjambre

Array $\mathrm{x_{gbest}}$ //vector de solución global
Array T // vector lista tabú
Array e_i // vector workstations
Array t_j // vector tasks
Array $\overrightarrow{CD_{ij}}$ // vector de task enriquecidos ($t_j e$) del workstation (e_i) deseada.
Array $\overrightarrow{CR_{ij}}$ // vector de características del workstation (e_i) real
Array SP// vector de solución pivote
Array VP// vector de partículas

Función Busqueda Tabú

```
Begin
    While termination ≤ IT to
        VP = GenerarPivotesVecindad(E, T);
        for e ≔ 1 to E do
            {x_lbest, v_*} = pso(VP, N, M, D, CD_ij, CR_ij);
        end for
        if x_lbest is the best than x_gbest then
            x_gbest = x_lbest ;
            T[n_i] = {x_lbest, v_*, 1};
        else
            T[n_i] = {x_lbest, v_*};
        end if
    end while
end
```

Función PSO

$\{\mathrm{x_{lbest}} :: \mathrm{Array}[1 \ldots \mathrm{m}], \mathrm{v_*}\} = \mathrm{pso}(\mathrm{v_p} :: \mathrm{Array}, \mathrm{pn}: : \mathrm{int}, \mathrm{D} :: \mathrm{int},)$
begin

```
        v_* = InicializarParticulas(v_p, pn);
        repeat
        for P := 1 to pn do
                v_*[p] = funcionObjetivo(v_*[p], C, i, j);
                for d := 1 to D to
                        v_*[p] = ActualizarVelocidad(v_*[p]);
                        v_*[p] = ActualizarPosicion(v_*[p]);
                end for
        end for
        until termination
        x_lbest = SeleccionarMejor(v_*);
End
```

Algoritmo 1. Seudocódigo del algoritmo híbrido de enjambre.

III – 3. IMPLEMENTACION DEL DISEÑO GENERAL DEL PLANIFICADOR PREDICTIVO

Después de diseñar conceptualmente el modelo del planificador predictivo, en este capítulo se procede a implementar un prototipo funcional a través de un paradigma de programación orientado a agentes en lenguaje de programación Java. El uso de agentes genera paralelización, comunicación y autonomía. Cada agente es una entidad situada en un medio, que trabaja de manera independiente para lograr un objetivo [36].

El prototipo realizado no implementa el MPS en su totalidad; sino, el componente adaptativo y el scheduler a corto plazo. Como trabajo futuro se puede construir el MPS en su totalidad. En este capítulo se detalla la implementación del prototipo iniciando con el modelamiento que se hizo mediante agentes, la descripción de cada agente creado y el diagrama de secuencia.

1. Arquitectura de Agentes

En la arquitectura de agentes se describe cada uno de los agentes de forma particular, cómo se clasifican y cómo interactúan en el sistema multiagente [37]. Los agentes creados en el sistema multiagente son homólogos a los subcomponentes del scheduler a corto plazo y del sistema adaptativo. A nivel global, el sistema multiagente asume la meta de resolver óptimamente un problema de scheduling. En la Figura 5 se muestra el diagrama de clases de los agentes. Las clases de los agentes surgen de las submetas y de las comunicaciones. A continuación, se describen los agentes creados:

Agente Adaptador. Este agente es homólogo del subcomponente adaptador del componente adaptativo. Entre las submetas que realiza este agente se encuentran: recibir el task básico $(t_j b)$, enviar el task básico $(t_j b)$ al agente manejador de perfiles, cuya respuesta es enviada al agente mecanismo de adaptación (pf), este último, agente mecanismo de adaptación, devuelve una respuesta que será la salida del componente adaptativo, lo que se denomina task enriquecido $(t_j e)$. La forma de conceptualizar al agente adaptador es viéndolo como el encargado de articular el proceso que se realiza en el componente adaptativo. Las submetas del agente adaptador se pueden detallar en la Tabla 7.

Agente Adaptador		
1	Recibir el task básico $(t_j b)$	El agente se encuentra en espera de un mensaje. En caso de recibir el mensaje pasa inmediatamente al rol 2.
2	Enviar tasks básico $(t_j b)$	El agente envía el task básico como mensaje al agente manejador de perfiles.
3	Recibir perfil (pf) del tasks básico $(t_j b)$	El agente se encuentra en espera de un mensaje. En caso de recibir el mensaje pasa al rol 4.
4	Envía perfil (pf) del tasks básico $(t_j b)$	El agente envía el perfil (pf) del task básico $(t_j b)$ al agente mecanismos de adaptación.
5	Recibe el tasks enriquecido $(t_j e)$.	El agente se encuentra en espera de un mensaje. En caso de recibir el mensaje pasa al rol 5.

6	Envía el tasks enriquecido ($t_j e$).	El agente envía el tasks enriquecido ($t_j e$) al agente búsqueda tabú. Retorna al rol 1.

Tabla 7. Submetas del agente adaptador.

En el prototipo funcional la entrada del agente adaptador es un arreglo de llamadas ($t_j b$), y su salida será el task enriquecido ($t_j e$). Se comunica con el agente manejador de perfiles y el agente mecanismos de adaptación.

Agente Manejador de Perfiles. Este agente es homólogo del subcomponente manejador de perfiles del componente adaptativo. Entre las submetas que realiza este agente se encuentran: recibir el task básico ($t_j b$), enviarlo al agente de consulta bases de datos para buscar un perfil (pf) almacenado. En caso de que el task básico ($t_j b$) no posea ningún perfil guardado, el manejador de perfiles será el encargado de crear un perfil generalizado. Las submetas del agente manejador de perfiles se pueden detallar en la Tabla 8.

Agente Manejador de Perfiles		
1	Recibe el tasks básico ($t_j b$).	El agente se encuentra en espera de un mensaje. En caso de recibir el mensaje pasa al rol 2.
2	Envía el tasks básico ($t_j b$) al agente base de datos.	El agente envía el tasks básico ($t_j b$) al agente base de datos.
3	Recibe el perfil (pf) del tasks básico ($t_j b$)	El agente se encuentra en espera de un mensaje. En caso de recibir el mensaje el agente evalúa si debe crear un perfil generalizado, sino se encuentra ninguno almacenado del task básico $(t_j b)$. Pasa al rol 4.
4	Envía perfil (pf) del tasks básico ($t_j b$)	El agente envía el perfil (pf) del tasks básico ($t_j b$) al agente adaptador. Retorna al rol 1.

Tabla 8. Submetas del agente manejador de perfiles.

En el prototipo funcional la entrada del agente manejador de perfiles proviene del agente adaptador y es un arreglo de llamadas (N), y su salida será un perfil (pf) o un perfil generalizado. Presenta comunicación con el agente adaptador.

Agente Mecanismo de Adaptación. Este agente es homólogo del subcomponente mecanismo de adaptación del componente adaptativo. Entre las submetas que realiza este agente se encuentran: recibe el perfil (pf) y a través de unas reglas y funciones, determina las características de una workstation (e_i) que se ajusten de mejor manera para ejecutar un task ($t_j b$) con el perfil (pf)dado. Las submetas del agente mecanismo de adaptación se pueden detallar en la Tabla 9.

Agente Mecanismos de Adaptación

1	Recibe el perfil (pf) del tasks básico $(t_j b)$	El agente se encuentra en espera de un mensaje. En caso de recibir el mensaje a través de un sistema de reglas crea el tasks enriquecido $(t_j e)$ y pasa al rol 2.
2	Envía el tasks enriquecido $(t_j e)$.	El agente envía el tasks enriquecido $(t_j e)$ al agente adaptador. Retorna al rol 1.

Tabla 9. Submetas del agente mecanismos de adaptación.

En el prototipo funcional la entrada del agente mecanismo de adaptación proviene del adaptador y es un perfil (pf). Su salida será un task enriquecido $(t_j e)$. Presenta comunicación con el agente adaptador.

Agente Búsqueda Tabú. Este agente es homólogo del subcomponente búsqueda tabú del scheduler a corto plazo. Entre las submetas que realiza este agente se encuentran: recibe el task enriquecido $(t_j e)$, crea la solución inicial del scheduling aleatoriamente, a partir de esta crea las soluciones pivotes y las envían a cada agente PSO. Luego recibe las soluciones del agente PSO y son guardadas en las estructuras de memoria tabú. Las submetas del agente búsqueda tabú se pueden detallar en la Tabla 10.

Agente Búsqueda Tabú		
1	Recibe el tasks enriquecido $(t_j e)$.	El agente se encuentra en espera de un mensaje. En caso de recibir el mensaje crea soluciones pivote e inicializa la estructura de memoria tabú. Pasa al rol 2.
2	Envía las soluciones pivote a los agentes PSO	El agente envía las soluciones pivote a los agentes PSO.
3	Recibe el pBest de los agentes PSO	El agente se encuentra en espera de un mensaje. En caso de recibir el mensaje, guarda el mensaje, que es la mejor solución de un enjambre de partículas en la lista tabú. Si el número de iteraciones en el agente tabú es mayor a uno, se realiza un proceso iterativo que crea las soluciones pivote y pasa de nuevo al rol 2. Al realizar el número de iteraciones deseadas se extrae la mejor solución de la estructura de memoria Tabú. Pasa al rol 4.
4	Envía el gBest a la operación	El agente envía el gBest a la operación. Pasa al rol 1.

Tabla 10. Submetas del agente Búsqueda Tabú.

En el prototipo funcional la entrada del agente búsqueda tabú proviene del adaptador y es un task enriquecido $(t_j e)$. Este presenta dos salidas: la primera salida es una solución pivote a cada uno de los agentes PSO creados, mientras que la segunda salida es la solución óptima del problema de scheduling y a su vez, tiene comunicación entre el agente adaptador y los agentes PSO.

Agente PSO. Este agente es homólogo del subcomponente PSO del scheduler a corto plazo. Las tareas que realiza son: recibe la solución pivote y a partir de esta se calcula el algoritmo PSO. La solución se retorna al agente búsqueda tabú. El número de agentes PSO equivale al número de enjambres que se desean crear. Las submetas del agente PSO se pueden detallar en la Tabla 11.

Agente PSO		
1	Recibe las soluciones pivote del agente búsqueda tabú	El agente se encuentra en espera de un mensaje. En caso de recibir el mensaje se inicializan las partículas del enjambre a partir de la solución pivote. Se calcula el algoritmo PSO. Pasa al rol 2.
2	Envía el pBest de los agentes búsqueda tabú.	El agente envía la mejor solución del enjambre (pBest) al agente búsqueda tabú. Pasa al rol 1.

Tabla 11. Submetas del agente PSO.

En el prototipo funcional la entrada del agente PSO es una solución pivote, a partir de la cual se crean las partículas del enjambre. Su salida es una posible solución al problema de scheduling.

Agente Monitor del Contexto. Este agente es homólogo del componente monitor del contexto. Entre las submetas que realiza este agente se encuentran: enviar información del contexto al agente búsqueda tabú. Las submetas del agente monitor se pueden detallar en la Tabla 12.

Agente Monitor del Contexto		
1	Recibe mensajes solicitando información del contexto (IC)	El agente se encuentra en espera de un mensaje. En caso de recibir el mensaje pasa al rol 2.
2	Envía solicitud al agente base de datos.	El agente envía mensaje al agente de base de datos.
3	Recibe la información solicitada (IC) del agente base de datos.	El agente se encuentra en espera de un mensaje. En caso de recibir el mensaje pasa al rol 4.
4	Envía información (IC) a los agentes que la solicitaron.	El agente envía información (IC) a los agentes que la solicitaron.

Tabla 12. Submetas del agente monitor del contexto.

En el prototipo funcional el agente monitor de contexto no presenta entradas. Su salida es la Información del contexto (IC). Se comunica con el agente búsqueda tabú.

Agente Base de Datos. Agente especializado en hacer consultas a la base de datos. Entre las submetas que realiza este agente se encuentran : enviar consultas a la base de datos según el mensaje enviado por los otros agentes y retorna los datos. Las submetas del agente base de datos se pueden detallar en la Tabla 13.

Agente Base de Datos		
1	Recibe mensajes solicitando información de la base de datos.	El agente se encuentra en espera de un mensaje. En caso de recibir el mensaje consulta a la base de datos la información solicitada por el agente. Pasa al rol 2.

2	Envía información a los agentes que la solicitaron.	El agente envía la información extraída de la base de datos, a los agentes que la solicitaron. Pasa al rol 1.

Tabla 13. Submetas del agente base de datos.

En el prototipo funcional el agente base de datos recibe mensajes de consulta de los agentes monitor del contexto y manejador de perfiles. El monitor de contexto consulta cantidad de agentes adheridos a la línea de atención. El manejador de perfiles solicita el perfil almacenado en las bases de datos. Se comunica con el manejador de perfiles y el monitor de contexto.

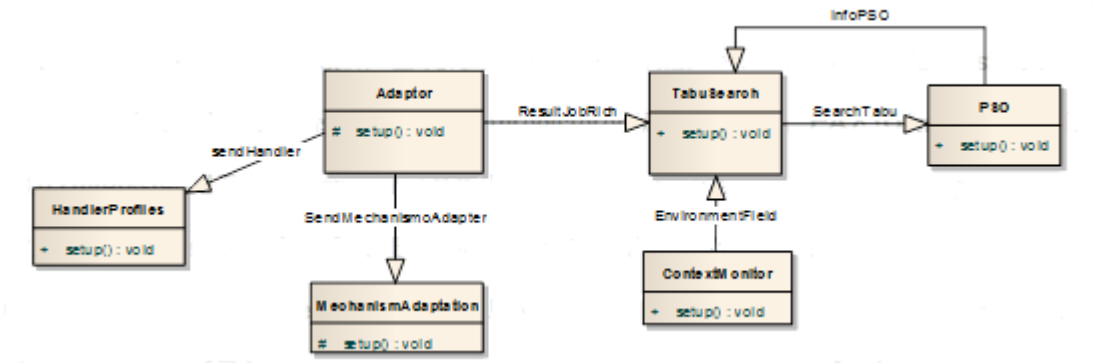


Figura 5. Modelo de clases de agentes.

2. Diagramas de Secuencia

En esta sección se realiza el diagrama de secuencia para mostrar la interacción entre los agentes en una secuencia de tiempo. Se muestran los agentes que intercambian mensajes y cuales mensajes. El diagrama de secuencia del prototipo funcional implementado se muestra en la Figura 6.

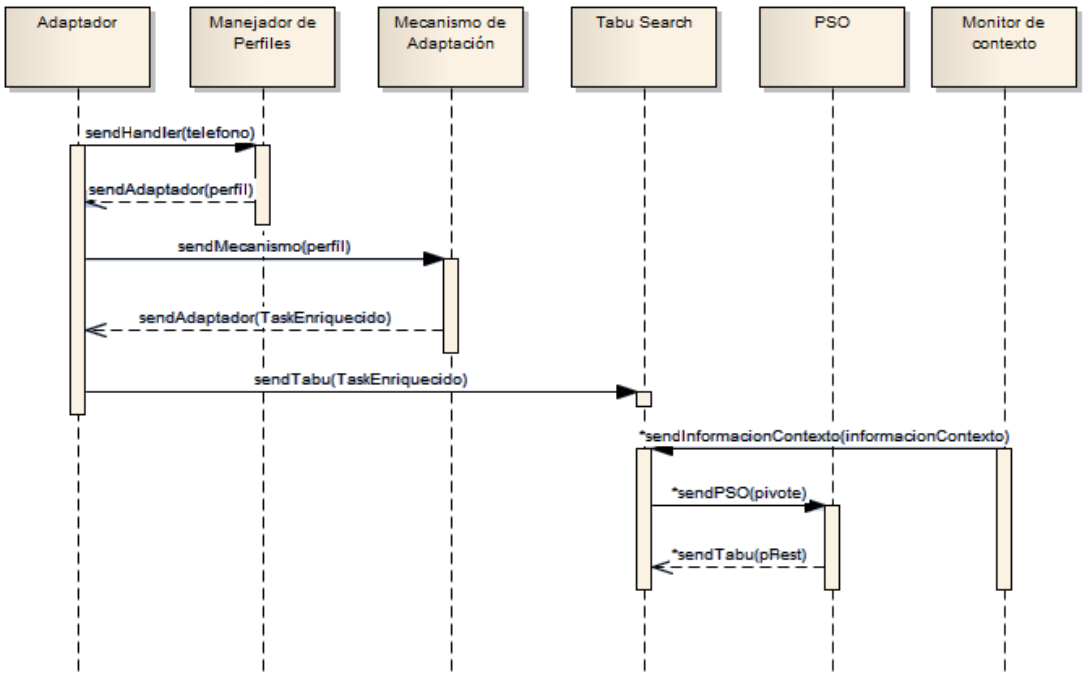


Figura 6. Diagrama de secuencia del sistema multiagentes implementado.

3. Características del Prototipo Implementado.

Para la implementación del modelo de planificación predictiva se utilizó JADE. Una plataforma de software en lenguaje de programación Java para desarrollar sistemas inteligentes o aplicaciones que sigue los estándares FIPA [38] [39] .

Al crear el sistema multiagentes a través de JADE se obtuvieron las siguientes características [38]:

1) Interface gráfica para manejar o probar los agentes creados.
2) Herramientas de depuración de la aplicación.
3) Soporte para la ejecución de múltiples, paralelos y concurrentes actividades.
4) Eficiencia en el manejo de mensajes ACL.
5) Documentación extensa.

Para crear un agente en JADE lo único que se debe realizar es crear una clase que extienda de la clase Agent y sobrescribir el método *setup()* de la clase. El modelo computacional empleado por JADE es multitárea, en el que las tareas son ejecutadas concurrentemente. El ciclo de vida de los agentes JADE los controla dando un identificador a cada agente y registrándolo en el directorio de páginas amarillas. Cuando el agente muere, automáticamente es retirado del directorio [38].

IV – 4. ANALISIS Y EVALUACIÓN DEL PLANIFICADOR PREDICTIVO

Después de implementar el prototipo, se realizan una serie de experimentos que servirán como primero para encontrar con qué configuraciones del modelo de planificación predictivo (MPS) se mejora el resultado de las variables dependientes; y, segundo, medir el rendimiento del algoritmo bajos distintas configuraciones; y, tercero comprobar que el algoritmo si mejora el tiempo de procesamiento (p_{ij}) y el tiempo de espera (v_i).

En este capítulo se aborda la tarea de analizar el rendimiento del modelo de planificación predictivo a través de tres perspectivas. Para lo anterior, la evaluación consta de dos etapas: una parte cuantitativa que se basa en rendimiento, y una parte cualitativa orientada a las cualidades que debe tener el scheduling en el contexto del caso general presentado y en particular, al caso de estudio. Como primero, se desarrolla un análisis cualitativo comparando técnicas conocidas para la solución de scheduling contra el modelo diseñado. Y como segundo, se realiza un protocolo experimental basado en datos sintéticos. Y como tercero, y para finalizar, se implementa un protocolo experimental basado en datos obtenidos del caso de estudio.

4. Análisis Cualitativo

A lo largo de la literatura varios investigadores han publicado los resultados obtenidos en la utilización de métodos convencionales como no convencionales para solucionar problemas de optimización combinatorios, y en este caso concreto, los problemas de scheduling. A continuación se tratará una comparación acerca de investigaciones que han trabajado problemas de scheduling con características iguales o similares a las que se tratan en este estudio.

Earliest Deadline First. Invstigación sobre algoritmo dinámico para tratar problemas de scheduling inicialmente en sistemas operativos. Ha sido propuesto hibridado con algoritmos metaheurísticos, tales como algoritmos de colonias de hormigas (ACO). Una de las limitaciones de este algoritmo es que su tiempo de ejecución se incrementa exponencialmente si el sistema empieza a sobrecargarse con tareas. Y cuando es hibridado toma aún mayor tiempo en su ejecución [40].

Algoritmos Genéticos. Se consideran métodos sistemáticos para la solución de problemas de búsqueda y optimización. Aplican métodos de selección natural y evolución biológica. Los algoritmos genéticos son parametrizados por medio de estructuras denominadas cromosomas. Cada cromosoma se considera una solución en un espacio de soluciones [41]. Una limitación de los algoritmos genéticos es que son demasiado lentos en su ejecución para ser utilizados online. Existe una versión denominada algoritmos genéticos acelerados que buscan mejorar el tiempo de ejecución [42]. Se considera que son adaptativos porque realizan cambios con el fin de acoplarse a los cambios que surgen en el entorno [41].

Programación Dinámica. Consiste en una técnica para optimizar las decisiones que tienen que ver con un sistema que evoluciona. Este método fue utilizado con frecuencia para resolver problemas de optimización. Una de las ventajas de este método es la forma de abordar el problema al dividirlo en subproblemas o etapas en los que se debe tomar una decisión. Sin

embargo, los problemas tratados con programación dinámica deben ser resueltos hacia atrás. Este método también puede resolver problemas no deterministas [43].

En la Tabla 14 se encuentra una comparación no rigurosa entre los métodos anteriormente numerados y el modelo de planificación predictiva (MPS) desarrollado en este estudio. Se califica el método dependiendo de las características mencionadas en la primera columna de la Tabla 14. La calificación comprende el siguiente rango: si tiene (+) el método tiene poco presente esa característica, si es (++) tiene medianamente esa característica y si es (+++) esa característica es considerada una fortaleza del método.

	Earliest deadline first	**Algoritmos genéticos**	**Programación dinámica**	**Modelo de planificación predictivo (MPS)**
Mezcla (M)	+	+	+++	+++
Online (O)	+++	+	+	++
Workstation Humanos (H).	+	++	++	+++
Rendimiento	+	+	+	++
Adaptativo	+	++	+	+++

Tabla 14. Comparación cualitativa entre algoritmos utilizados para resolver problemas de scheduling.

5. Protocolo Experimental con Datos Sintéticos

El protocolo experimental con datos sintéticos se realiza para verificar si el modelo de planificación predictivo diseñado puede servir en otras situaciones en las que se desea realizar un scheduling, no exclusivamente en el caso de estudio.

Con los datos sintéticos se pretende evaluar el rendimiento del modelo de planificación predictivo implementado, y a su vez, elegir la mejor configuración para obtener una respuesta satisfactoria en el scheduling, sin sacrificar tiempo de ejecución del algoritmo. Se escoge un diseño completamente al azar. Para analizar efectos estadísticamente significativos o activos se realiza para las variables dependientes propuestas un estudio factorial de cuatro factores de tres niveles cada uno [44].

5.1 Definición de Variables

Es esta subsección se realiza la selección de las variables que se utilizan para realizar el protocolo experimental.

Variables Intervinientes. Para la elaboración del protocolo experimental se definen cuatro variables independientes, que a continuación se contextualizan de la siguiente manera:

a) Nivel de experticia del Workstation (e_i). Equivalente al perfil (pf) almacenado de la workstation y actúa directamente en el cálculo de tiempo de procesamiento (p_{ij}), adjudicado a cada workstation en el protocolo experimental con datos sintéticos.

b) Iteraciones en la lista tabú (IT). Número de iteraciones que se realiza de la lista tabú para encontrar en optimo global (x_{gbest}). Está variable se calibra dejando como valor final de número de iteraciones igual a cuatro.
c) Iteraciones en los enjambres (IE). Número de iteraciones que se realiza en el enjambre para encontrar la mejor solución hasta el momento ($pBest$). Está variable se calibra dejando como valor final de número de iteraciones igual a seis.

Variables Independientes. Para la elaboración del protocolo experimental se definen cuatro variables independientes. Se contextualizan de la siguiente manera:

a) Workstations (M). Definido como el número máximo de recursos que pueden ser asignado a unas tasks (N) para que sean procesadas.
b) Tasks (N). Definido como unidades de actividades en un proceso.
c) Enjambres (E). Nube de soluciones candidatas que exploran una región del espacio de búsqueda.
d) Partículas (P). Una de las muchas soluciones candidatas que hacen parte de un enjambre.

Debido a la gran cantidad de valores que pueden tomar las variables independientes definidas, se opta por definir rangos esperados de operación y examinarlos de forma cualitativa, entre alto, bajo y medio. La definición de rangos se puede observar en la Tabla 15.

Variables Independientes	Rangos	Alto	Medio	Bajo
Workstations (**M**)	1 a 208	208	150	50
Tasks (**N**)	1 a 452	452	297	100
Enjambres (**E**)	1 a 10	10	5	3
Partículas (P)	1 a 20	20	12	2

Tabla 15. Rangos esperados de operación.

Variables Dependientes. Para la elaboración del protocolo experimental se definen tres variables dependientes. Se contextualizan de la siguiente manera:

a) Tiempo de procesamiento (p_{ij}). Tiempo que demora una task (t_j) en ser procesada, desde que una workstation (e_i) empieza su ejecución hasta que finaliza. Los datos se expresan en segundos (s).
b) Tiempo de espera (v). Tiempo que pasa un taks (t_j) en el sistema antes de iniciar su procesamiento. Los datos se expresan en segundos (s).
c) Tiempo de ejecución del algoritmo híbrido de enjambre (AHE). Es el tiempo que demora el algoritmo híbrido de enjambre (AHE) en calcular la solución al problema de scheduling. Los datos se expresan en segundos (s). Esta variable se estudia en la subsección 5.4 Efectos de Dosificación en el Algoritmo Hibrido de Enjambre (AHE).

Después de elegir las variables para realizar el protocolo experimental, se procede a analizar cada una de las variables dependientes.

5.2 Análisis Variable Tiempo de Procesamiento ($\mathbf{p_{ij}}$)

Para el análisis de las variables dependientes se propone un diseño factorial de cuatro factores de tres niveles (alto, medio, bajo). Inicialmente, se analiza la variable tiempo de procesamiento (p_{ij}) y después la variable tiempo de espera (v_j).

Las hipótesis que se desean probar están dadas por la Ecuación 6. Las hipótesis planteadas se utilizarán para analizar las variables dependientes, tiempo de procesamiento (p_{ij}) y tiempo de espera (v_j). H_0 define la hipótesis nula y H_A la hipótesis de investigación.

$$H_0: Efecto\ workstation\ (M) = 0$$
$$H_A: Efecto\ workstation\ (M) \neq 0$$

$$H_0: Efecto\ tasks\ (N) = 0$$
$$H_A: Efecto\ tasks\ (N) \neq 0$$

$$H_0: Efecto\ Partículas\ (P) = 0$$
$$H_A: Efecto\ Partículas\ (P) \neq 0$$

$$H_0: Efecto\ Enjambre\ (E) = 0$$
$$H_A: Efecto\ Enjambre\ (E) \neq 0$$

$$H_0: Efecto\ workstation\ (M) \times Efecto\ tasks\ (N) = 0$$
$$H_A: Efecto\ workstation\ (M) \times Efecto\ tasks\ (N) \neq 0$$

$$H_0: Efecto\ workstation\ (M) \times Efecto\ Partículas\ (P) = 0$$
$$H_A: Efecto\ workstation\ (M) \times Efecto\ Partículas\ (P) \neq 0$$

$$H_0: Efecto\ workstation\ (M) \times Efecto\ Enjambre\ (E) = 0$$
$$H_A: Efecto\ workstation\ (M) \times Efecto\ Enjambre\ (E) \neq 0$$

$$H_0: Efecto\ tasks\ (N) \times Efecto\ Partículas\ (P) = 0$$
$$H_A: Efecto\ tasks\ (N) \times Efecto\ Partículas\ (P) \neq 0$$

$$H_0: Efecto\ tasks\ (N) \times Efecto\ Enjambre\ (E) = 0$$
$$H_A: Efecto\ tasks\ (N) \times Efecto\ Enjambre\ (E) \neq 0$$

$$H_0: Efecto\ Partículas\ (P) \times Efecto\ Enjambre\ (E) = 0$$
$$H_A: Efecto\ Partículas\ (P) \times Efecto\ Enjambre\ (E) \neq 0$$

$$H_0: Efecto\ workstation\ (M) \times Efecto\ Partículas\ (P) \times Efecto\ Enjambre\ (E) = 0$$
$$H_A: Efecto\ workstation\ (M) \times Efecto\ Partículas\ (P) \times Efecto\ Enjambre\ (E) \neq 0$$

$$H_0: Efecto\ workstation\ (M) \times Efecto\ tasks\ (N) \times Efecto\ Partículas\ (P) = 0$$
$$H_A: Efecto\ workstation\ (M) \times Efecto\ tasks\ (N) \times Efecto\ Partículas\ (P) \neq 0$$

$$H_0: Efecto\ workstation\ (M) \times Efecto\ tasks\ (N) \times Efecto\ Enjambre\ (E) = 0$$
$$H_A: Efecto\ workstation\ (M) \times Efecto\ tasks\ (N) \times Efecto\ Enjambre\ (E) \neq 0$$

$$H_0: Efecto\ tasks\ (N) \times Efecto\ Partículas\ (P) \times Efecto\ Enjambre\ (E) = 0$$
$$H_A: Efecto\ tasks\ (N) \times Efecto\ Partículas\ (P) \times Efecto\ Enjambre\ (E) \neq 0$$

Ecuación 6. Hipótesis del diseño factorial realizado para las variables dependientes.

Para probar o rechazar las hipótesis planteadas en la Ecuación 6 se realiza para cada una de las variables dependientes un diseño factorial de cuatro factores y tres niveles. La Tabla 16 muestra el diseño factorial resultante para la variable tiempo de procesamiento (p_{ij}). La Tabla 16 se estructura en seis columnas: 1) Fuente. Son los efectos y las combinaciones que posiblemente influyan en el resultado de la variable dependiente. 2) Grados de libertad (GL). Son parámetros que definen las distribuciones T, J_i-cuadrada y F, y se determinan a partir de los tamaños muestrales involucrados. 3) SC Sec. Se define como la suma de cuadrados, se explica como el nivel de fluctuación del efecto 4) SC Ajust. Se define como la suma de cuadrados ajustados, se realiza el ajuste para relacionar mejor el modelo a la población de estudio. 5) CM Ajust. Se define como la varianza o cuadrados medios, que se obtiene dividiendo la suma de cuadrados entre los grados de libertad. 6) F. Prueba F de Fisher. Se utiliza para comparar los efectos de la variación total y por último 7) P. Estadístico de prueba. Es un número calculado a partir de los datos y la hipótesis nula, cuya magnitud pemite discernir si se rechaza o no la hipótesis nula H_0. También se conoce como significancia observada o calculada. Se rechaza la significancia observada si el valor es mayor a la significancia dada $\alpha = 0.05$.

La significancia dada se define como el riesgo máximo que el experimentador esté dispuesto a correr con respecto a rechazar o no la hipótesis nula H_0.

A partir de las definiciones anteriores se analizan los resultados en los que se obtiene que debido a que el valor de P es menor que el valor de significancia fijado, serían relevantes los efectos workstation (M) y task (N) y se rechaza la hipótesis nula (H_0). Lo anterior significa, que el número de workstation$s$ (M) y el número de tasks (N) influye en la variable dependiente tiempo de procesamiento (p_{ij}).

Con respecto a las demás hipótesis planteadas se aceptan las hipótesis nulas (H_0). Esto significa que ni el efecto, ni las combinaciones planteadas afectan significativamente la variable dependiente. En la Tabla 16 se subraya en la columna P los efectos que son significativos o activos para la variable dependiente.

De la columna F de la Tabla 16 se aprecia que el efecto más importante es el task (N) seguido del workstation (M).

Fuente	**GL**	**SC Sec.**	**SC Ajust.**	**CM Ajust.**	**F**	**P**
Workstation	2	11917788603	11917788603	5958894301	3,86	0,043

Task	2	6.90E+16	6.90E+16	3.45E+16	223,45	0,000
Partículas	2	1018485751	1018485751	509242875	0,33	*0,724*
Enjambres	2	1285916806	1285916806	642958403	0,42	*0,666*
Workstation * Task	4	2273358486	2273358486	568339622	0,37	*0,828*
Workstation * Partículas	4	5629404656	5629404656	1407351164	0,91	*0,481*
Workstation * Enjambres	4	1074901527	1074901527	268725382	0,17	*0,949*
Task * Partículas	4	2113785212	2113785212	528446303	0,34	*0,846*
Task * Enjambres	4	2806963652	2806963652	701740913	0,45	*0,768*
Partículas * Enjambres	4	7978739856	7978739856	1994684964	1,29	*0,315*
Workstation * Task * Partículas	8	7130106195	7130106195	891263274	0,58	*0,782*
Workstation * Task * Enjambres	8	10605721289	10605721289	1325715161	0,86	*0,569*
Workstation * Partículas * Enjambres	8	12175777292	12175777292	1521972162	0,99	*0,482*
Task * Partículas * Enjambres	8	13796362603	13796362603	1724545325	1,12	*0,403*
Error	16	24712083569	24712083569	1544505223		
Total	80	7.95E+16				

Tabla 16. Diseño factorial de cuatro factores y tres niveles para la variable de tiempo de procesamiento ($\mathbf{p_{ij}}$).

Sin embargo, teniendo presente que los efectos más significativos son los workstation (M) y task (N) se procede a realizar un diagrama de interacción por cada uno de los niveles que poseen los efectos. La Figura 7, Figura 8 y Figura 9 muestran como al variar el efecto partículas (P) con el efecto enjambre (E) se encuentra una mejor respuesta del tiempo de procesamiento $(\mathrm{p_{ij}})$, aunque no sea significativa $(\alpha = 0.05)$ la variación.

Por ejemplo, la Figura 7 es el resultado de las relaciones tasks (N), partículas (P) y Enjambres (E) cuando las workstation (M) presentan un nivel alto. La Figura 7 presenta tres partes, la primera parte es la relación entre tasks (N) con un nivel alto con todas las posibles combinaciones de partículas (P) y Enjambres; la segunda parte es la relación entre tasks (N) con un nivel medio con todas las posibles combinaciones de partículas (P) y enjambres y la tercera

parte es la relación entre tasks (N) con un nivel bajo con todas las posibles combinaciones de partículas (P) y enjambres (E).

En la primera parte de la Figura 7 el algoritmo presenta una mejor respuesta de la variable dependiente tiempo de procesamiento (p_{ij}) cuando el algoritmo híbrido de enjambre se caracteriza con partículas (P) en un nivel medio y un enjambre (E) en un nivel bajo. Esta respuesta no es coherente con lo considerado, ya que se presumía que las mejores respuestas obtenidas para la variable dependiente se conseguirían con una configuración de enjambre (E) y partículas (P) altas. A mayor número de partículas (P) mayor número de soluciones y a mayor cantidad de enjambres (E) mayor cantidad de espacios de búsqueda exploradas (exploración, explotación).

La segunda parte de la Figura 7 el algoritmo presenta una mejor respuesta de la variable dependiente cuando el algoritmo híbrido de enjambre se configura con enjambres (E) a nivel medio y partículas (P) a nivel medio o bajo.

La tercera parte de la Figura 7 el algoritmo presenta respuestas muy similares con todas las combinaciones posibles de enjambres (E) y de partículas (P).

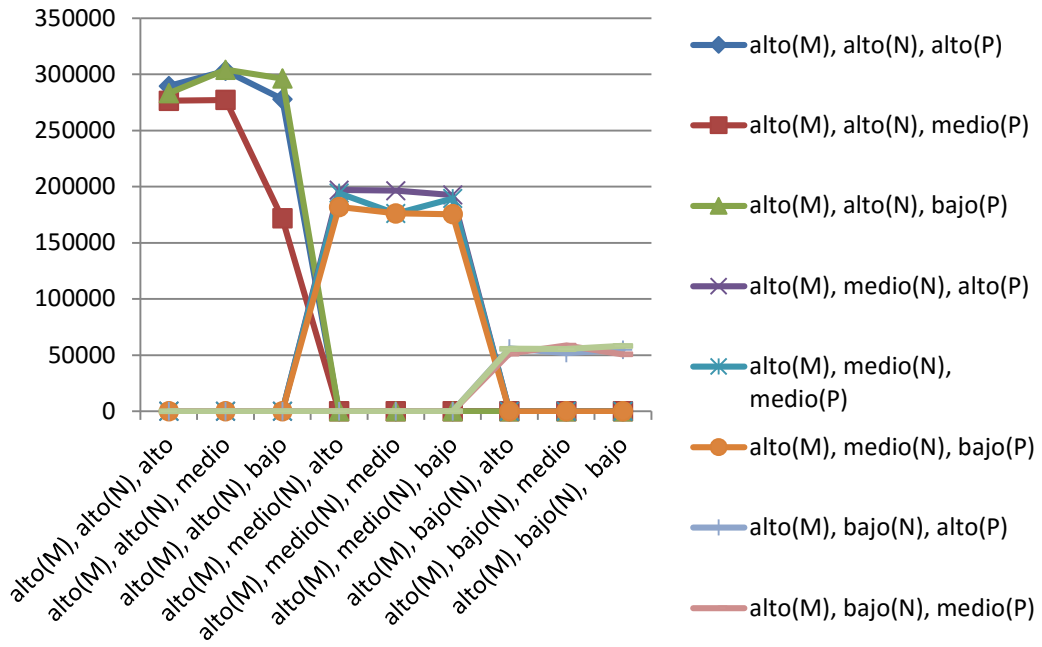


Figura 7. Diagrama de interacción para enjambres por partículas en workstation (M) con nivel alto para la variable tiempo de procesamiento (p_{ij}).

La Figura 8 es el resultado de las relaciones tasks (N), partículas (P) y Enjambres (E) cuando las workstation (M) presentan un nivel medio. La Figura 8 presenta tres partes, la primera parte es la relación entre tasks (N) con un nivel alto con todas las posibles combinaciones de partículas (P) y Enjambres; la segunda parte es la relación entre tasks (N) con un nivel medio

con todas las posibles combinaciones de partículas (P) y enjambres y la tercera parte es la relación entre tasks (N) con un nivel bajo con todas las posibles combinaciones de partículas (P) y enjambres (E).

En la primera y tercera parte de la Figura 8, en donde se grafica las respuestas del algoritmo híbrido de enjambre no presentan mayor variación con respecto a diferentes combinaciones. La segunda parte presenta una mejoría cuando las partículas (P) están en un nivel bajo y los enjambres (E) en un nivel medio.

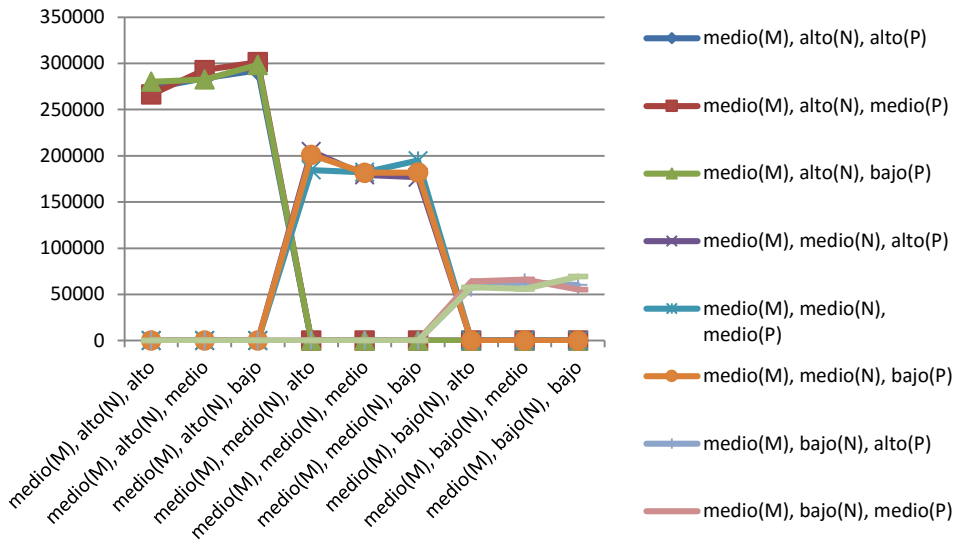


Figura 8. Diagrama de interacción para enjambres por partículas en workstation (M) con nivel medio para la variable tiempo de procesamiento (p_{ij}).

La Figura 9 es el resultado de las relaciones tasks (N), partículas (P) y Enjambres (E) cuando las workstation (M) presentan un nivel bajo. La Figura 9 presenta tres partes, la primera parte es la relación entre tasks (N) con un nivel alto con todas las posibles combinaciones de partículas (P) y Enjambres; la segunda parte es la relación entre tasks (N) con un nivel medio con todas las posibles combinaciones de partículas (P) y enjambres y la tercera parte es la relación entre tasks (N) con un nivel bajo con todas las posibles combinaciones de partículas (P) y enjambres (E).

En la primera parte de la Figura 9 el algoritmo presenta una mejor respuesta de la variable dependiente tiempo de procesamiento (p_{ij}) cuando el algoritmo híbrido de enjambre se caracteriza con partículas (P) en un nivel medio y un enjambre (E) en un nivel medio. Esta respuesta no es coherente con lo considerado. Ya que se presumía que las mejores respuestas obtenidas para la variable dependiente se conseguiría con una configuración de enjambre (E) y partículas (P) altas.

La segunda parte de la Figura 7 el algoritmo presenta una mejor respuesta de la variable dependiente cuando el algoritmo híbrido de enjambre se configura con enjambres (E) a nivel medio o bajo y partículas (P) a nivel medio.

La tercera parte de la Figura 7 el algoritmo presenta respuestas muy similares con todas las combinaciones posibles de enjambres (E) y de partículas (P). A excepción de cuando las partículas (P) se encuentran a nivel bajo, con esta caracterización se incrementa considerablemente la variable tiempo de procesamiento (p_{ij}).

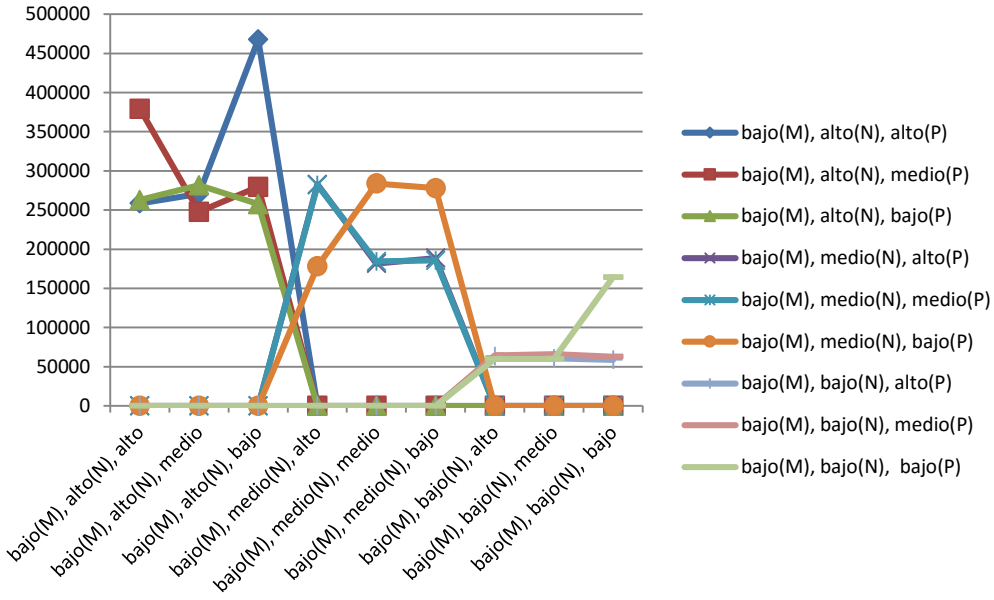


Figura 9. Diagrama de interacción para enjambres por partículas en workstation (M) con nivel bajo para la variable tiempo de procesamiento (p_{ij}).

Si se debe decidir con qué configuración se logra optimizar la variable tiempo de procesamiento (p_{ij}), es necesario estudiar otras variables, tales como: tiempo de espera (v_j) y tiempo de ejecución del algoritmo híbrido de enjambre (AHE).

5.3 Análisis Variable Tiempo de Espera (v_j)

A continuación se analiza el experimento que se desarrolla para estudiar la variable tiempo de espera (v_j). Se escogió un diseño factorial de cuatro factores y tres niveles. Se pretende con este experimento caracterizar el algoritmo y definir bajo que efectos se presenta una mejor respuesta en la variable dependiente.

La Tabla 17 muestra el diseño factorial resultante para la variable tiempo de espera (v_j). Según los resultados mostrados se obtiene que debido a que el valor de P es menor que el valor de

significancia fijado en $\alpha = 0.05$ es significante el efecto combinado (workstation (M) * enjambres (E)) y se rechaza la hipótesis nula de la Ecuación 6. Esto quiere decir que la combinación entre workstation (M) y enjambres (E) influye en la variable dependiente tiempo de espera (v_j).

Con respecto a las demás hipótesis planteadas se aceptan las hipótesis nulas. Esto significa que ni el efecto, ni las combinaciones planteadas afectan significativamente la variable dependiente. En la Tabla 17 se marca la columna P. La marca verde representa que el efecto es significativo o activo y la marca roja representa que el efecto no es significativo para la variable dependiente.

Fuente	**GL**	**SC Sec.**	**SC Ajust.**	**CM**	**F**	**P**
Workstation	2	0,914	0,914	0,457	0,35	*0,709*
Task	2	0,099	0,099	0,049	0,04	*0,963*
Partículas	2	0,765	0,765	0,383	0,29	*0,749*
Enjambres	2	1,284	1,284	0,642	0,49	*0,619*
Workstation * Task	4	1,235	1,235	0,309	0,24	*0,913*
Workstation * Partículas	4	9,901	9,901	2,475	1,90	*0,159*
Workstation * Enjambres	4	12,494	12,494	3,123	2,40	0,033
Task * Partículas	4	5,160	5,160	1,290	0,99	*0,440*
Task * Enjambres	4	1,975	1,975	0,494	0,38	*0,820*
Partículas * Enjambres	4	1,975	1,975	0,494	0,38	*0,820*
Workstation * Task * Partículas	8	2,173	2,173	0,272	0,21	*0,985*
Workstation * Task * Enjambres	8	4,914	4,914	0,614	0,47	*0,858*
Workstation * Partículas * Enjambres	8	15,580	15,580	1,948	1,50	*0,233*
Task * Partículas * Enjambres	8	16,988	16,988	2,123	1,63	*0,192*
Error	16	20,790	20,790	1,299		
Total	80	96,247				

Tabla 17. Diseño factorial de cuatro factores tres niveles para la variable de tiempo de espera (**v**)**.**

5.4 Efectos de Dosificación en el Algoritmo Hibrido de Enjambre (AHE)

Este experimento se realiza para comprobar los beneficios que presenta el modelo de planificación predictivo (MPS) por contar con un scheduler a mediano plazo. El cual impide la sobrecarga del scheduler a corto plazo y la reducción del espacio de búsqueda del algoritmo híbrido de enjambre, a través de la dosificación de tasks (N) al tipificar las tasks (N) en inmediatas y deferidas. Para el análisis de esta variable se realiza un diseño factorial de cuatro factores y tres niveles que permitirá caracterizar el algoritmo según la respuesta obtenida en la variable dependiente tiempo de procesamiento (p_{ij}). La respuesta esperada es un menor tiempo de ejecución del algoritmo si se presentan tasks (N) bajas y workstations (M) bajas.

La Tabla 18 muestra el diseño factorial resultante para la variable tiempo de ejecución del algoritmo híbrido de enjambre (AHE). Según los resultados mostrados se obtiene que debido a que el valor de P es menor que el valor de significancia fijado en $\alpha = 0.05$ son significantes los efectos workstation (M), task (N), partículas (P) y los enjambres (E). Así mismo, los efectos combinados de (workstation (M) * enjambres (E)), (workstation (M) * task (N)), (workstation (M) * partículas (P)), (task (N) * partículas (P)) y (workstation (M) * task (N) * partículas (P)) son significativos y se rechaza la hipótesis nula de la Ecuación 6. Esto quiere decir que todos los efectos anteriormente nombrados influyen en la variable dependiente tiempo de ejecución del algoritmo híbrido de enjambre (AHE).

De acuerdo con las demás hipótesis planteadas se aceptan las hipótesis nulas. Esto significa que ni el efecto, ni las combinaciones planteadas afectan significativamente la variable dependiente. En la Tabla 18 se subraya la columna P donde el efecto es significativo para la variable dependiente.

De la columna F de la Tabla 18 se aprecia que el efecto más importante es el workstation (M), workstation (M) * task (N), enjambres (E), task (N), workstation (M) * enjambres (E), workstation (M) * task (N) * partículas (P), task (N) * partículas (P), partículas (P) y por último workstation (M) * partículas (P).

Fuente	**GL**	**SC Sec.**	**SC Ajust.**	**CM Ajust.**	**F**	**P**
Workstation	2	3,247,144	3,247,144	1,623,572	391,54	0,000
Task	2	251,215	251,215	125,608	30,29	0,000
Partículas	2	54,828	54,828	27,414	6,61	0,008
Enjambres	2	304,698	304,698	152,349	36,74	0,000
Workstation*Task	4	754,016	754,016	188,504	45,46	0,000
Workstation*Partículas	4	54,796	54,796	13,699	3,30	0,037
Workstation*Enjambres	4	272,746	272,746	68,187	16,44	0,000
Task*Partículas	4	157,851	157,851	39,463	9,52	0,000
Task*Enjambres	4	23,331	23,331	5,833	1,41	*0,277*
Partículas*Enjambres	4	5,108	5,108	1,277	0,31	*0,868*
Workstation*Task*Partículas	8	343,552	343,552	42,944	10,36	0,000

Workstation*Task*Enjambres	8	22,277	22,277	2,785	0,67	*0,710*
Workstation*Partículas*Enjambres	8	11,476	11,476	1,434	0,35	*0,934*
Task*Partículas*Enjambres	8	28,136	28,136	3,517	0,85	*0,576*
Error	16	66,346	66,346	4,147		
Total	80	5,597,519				

Tabla 18. Diseño factorial de cuatro factores y tres niveles para la variable de tiempo de ejecución del algoritmo híbrido de enjambre ($\boldsymbol{AHE}$).

Se procede a realizar un diagrama de interacción por cada uno de los niveles que poseen los efectos workstation (M). La Figura 10, Figura 11 y Figura 12 muestran cómo al variar el efecto partículas (P) con el efecto enjambre (E) se encuentra una mejor respuesta del tiempo de procesamiento (p_{ij}), siendo significativa ($\alpha = 0.05$) la variación.

Por ejemplo, la Figura 10 es el resultado de las relaciones tasks (N), partículas (P) y Enjambres (E) cuando las workstation (M) presentan un nivel alto, a su vez presenta tres partes; la primera parte, es la relación entre tasks (N) con un nivel alto con todas las posibles combinaciones de partículas (P) y Enjambres; la segunda parte, es la relación entre tasks (N) con un nivel medio con todas las posibles combinaciones de partículas (P) y enjambres; y, la tercera parte, es la relación entre tasks (N) con un nivel bajo con todas las posibles combinaciones de partículas (P) y enjambres (E).

En la primera parte de la Figura 10, el algoritmo presenta una mejor respuesta de la variable dependiente tiempo de ejecución del algoritmo cuando el algoritmo híbrido de enjambre se caracteriza con partículas (P) en un nivel bajo y un enjambre (E) en un nivel medio. Esta respuesta no es coherente con lo considerado, ya que se presumía que las mejores respuestas obtenidas para la variable dependiente se conseguirían con una configuración de enjambre (E) y partículas (P) bajas. A menor número de partículas (P) menor número de cálculos y a menor cantidad de enjambres (E) menos espacios de búsqueda para explotar.

La segunda parte de la Figura 10, el algoritmo presenta una mejor respuesta de la variable dependiente cuando el algoritmo híbrido de enjambre se configura con enjambres (E) a nivel bajo y partículas (P) a nivel bajo.

La tercera parte de la Figura 10, el algoritmo presenta una mejor respuesta cuando los enjambres están en un nivel bajo y las partículas en un nivel bajo. Las respuestas obtenidas de la segunda y tercera parte son consistentes con lo considerado para este experimento.

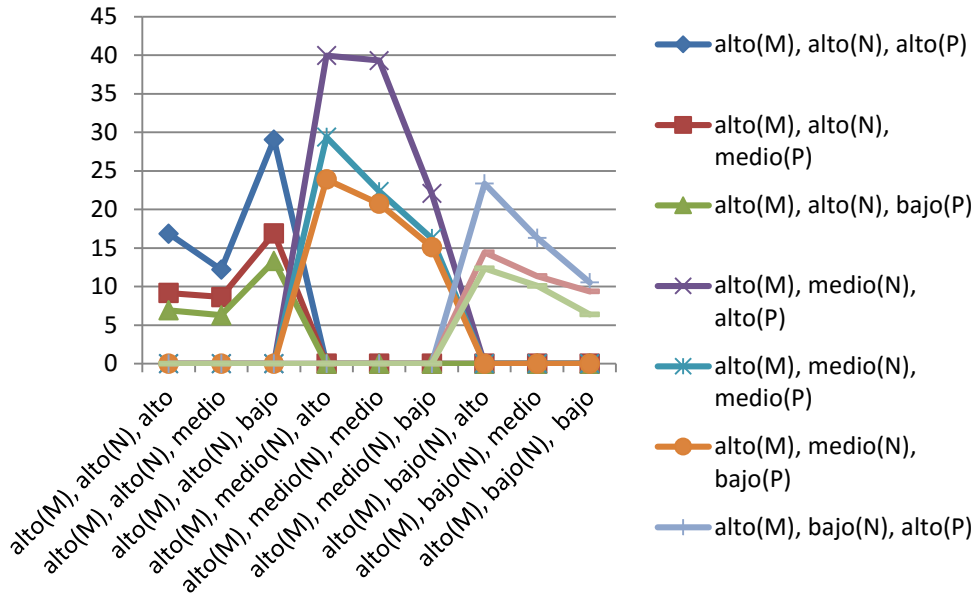


Figura 10. Diagrama de interacción para enjambres por partículas para la variable tiempo de ejecución del algoritmo híbrido de enjambre (AHE) para workstation (M) con nivel alto.

En la Figura 11 se ve el resultado de las relaciones tasks (N), partículas (P) y Enjambres (E) cuando las workstation (M) presentan un nivel medio, a su vez la Figura 7 presenta tres partes; la primera parte, es la relación entre tasks (N) con un nivel alto con todas las posibles combinaciones de partículas (P) y Enjambres; la segunda parte, es la relación entre tasks (N) con un nivel medio con todas las posibles combinaciones de partículas (P) y enjambres y; la tercera parte, es la relación entre tasks (N) con un nivel bajo con todas las posibles combinaciones de partículas (P) y enjambres (E).

En la primera parte de la Figura 11 el algoritmo presenta una mejor respuesta de la variable dependiente tiempo de ejecución del algoritmo cuando el algoritmo híbrido de enjambre se caracteriza con partículas (P) en un nivel bajo y un enjambre (E) en un nivel bajo. Esta respuesta es coherente con lo considerado. Se presume que las mejores respuestas obtenidas para la variable dependiente se conseguira con una configuración de enjambre (E) y partículas (P) bajas.

La segunda parte de la Figura 11 el algoritmo presenta una mejor respuesta de la variable dependiente cuando el algoritmo híbrido de enjambre se configura con enjambres (E) a nivel bajo y partículas (P) a nivel bajo.

La tercera parte de la Figura 11 el algoritmo presenta una mejor respuesta cuando los enjambres están en un nivel bajo y las partículas en un nivel bajo. Las respuestas obtenidas de la segunda y tercera parte son consistentes con lo considerado para este experimento.

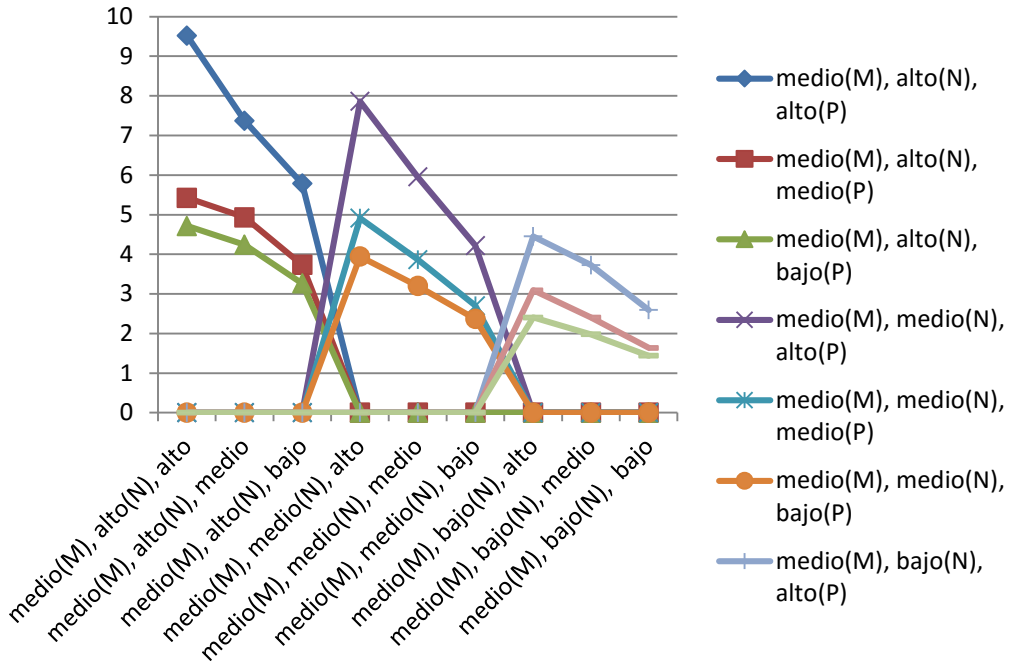


Figura 11. Diagrama de interacción para enjambres por partículas para la variable tiempo de ejecución del algoritmo híbrido de enjambre (***AHE***) para workstation (**M**) con nivel medio.

Y en la Figura 12 se ve el resultado de las relaciones tasks (N), partículas (P) y Enjambres (E) cuando las workstation (M) presentan un nivel bajo, así mismo la Figura 12 presenta tres partes, la primera parte, es la relación entre tasks (N) con un nivel alto con todas las posibles combinaciones de partículas (P) y Enjambres; la segunda parte, es la relación entre tasks (N) con un nivel medio con todas las posibles combinaciones de partículas (P) y enjambres y la tercera parte, es la relación entre tasks (N) con un nivel bajo con todas las posibles combinaciones de partículas (P) y enjambres (E).

En todas las partes de la Figura 12 la respuesta es consistente con lo considerado. Se estima que a mejor número de enjambres (E) y menor número de partículas (P) el algoritmo híbrido de enjambre encuentra de forma rápida la solución al scheduling propuesto.

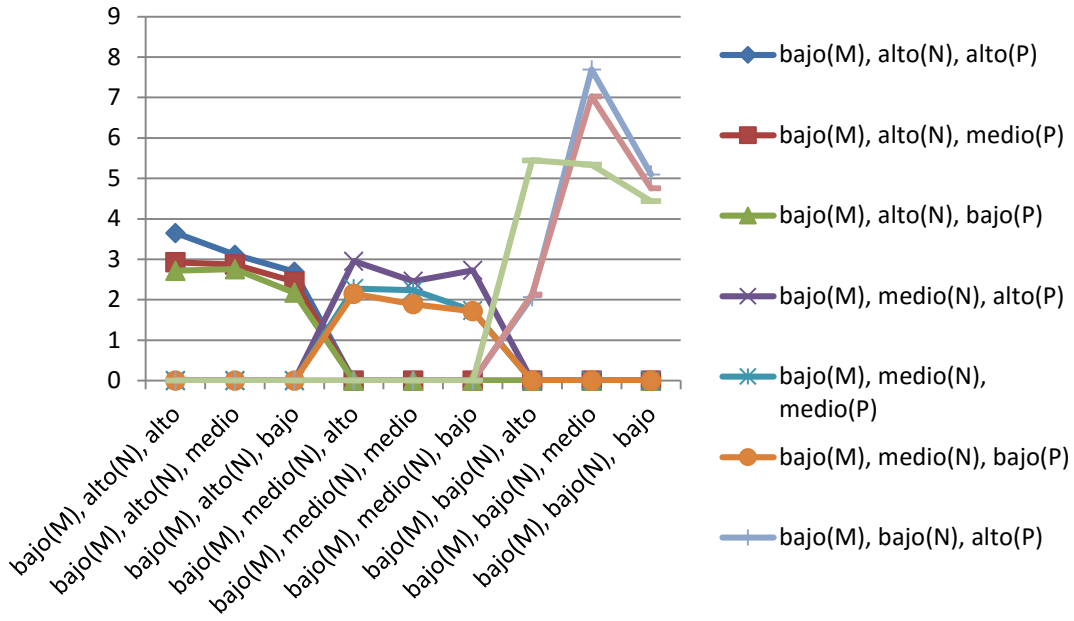


Figura 12. Diagrama de interacción para enjambres por partículas para la variable tiempo de ejecución del algoritmo híbrido de enjambre (AHE) para workstation (M) con nivel bajo.

Se concluye, para este experimento que los resultados obtenidos son consistentes con las respuestas esperadas, donde se razonaba que a menor cantidad de enjambres (E) y partículas (P), el algoritmo demora menos en su ejeción.

6. Protocolo Experimental con Datos del Caso Real

El protocolo experimental con datos del caso real se realiza para comprobar que existe mejora con respecto a la variable dependiente seleccionada. Para analizar efectos estadísticamente significativos o activos se ejecuta para las variables dependientes propuestas un estudio factorial de dos factores de tres niveles [44]. A continuación se eligen las variables intervinientes, dependientes e independientes.

Variables Intervinientes. Las variables intervinientes escogidas para el protocolo experimental con datos del caso real son iguales al protocolo experimental con datos sintéticos. Sin embargo, la calibración y fuente de donde se extrae la información son bases de datos reales de un Call Center. La bases de datos se dividen en datos de agentes, datos de personas contacto, datos de telefonía y datos de tipificación. También, se añaden dos variables intervinientes.

a) Workstations (M). Definido como el número máximo de recursos que pueden ser asignado a unas tasks (N) para que sean procesadas.
b) Tasks (N). Definido como unidades de actividades en un proceso.
c) Nivel de experticia del Workstation (e_i). A través de minería de datos a las bases de datos del Call Center se realizan los indicadores de experticia para los workstations (M).

d) Iteraciones en la lista tabú (IT). Número de iteraciones que se realiza de la lista tabú para encontrar en optimo global (x_{gbest}). Está variable se calibra al dejar como valor final de número de iteraciones igual a cinco.
e) Iteraciones en los enjambres (en) . Número de iteraciones que se realiza en el enjambre para encontrar el la mejor solución hasta el momento ($pBest$). Está variable se calibra al dejar como valor final de número de iteraciones igual a cinco.

Variables Independientes. Para la elaboración del protocolo experimental se definen dos variables independientes. Se contextualizan de la siguiente manera:

a) Enjambres (E). Nube de soluciones candidatas que exploran una región del espacio de búsqueda.
b) Partículas (P). Una de las muchas soluciones candidatas que hacen parte de un enjambre.

Se definen rangos esperados de operación y se examinan de forma cualitativa, entre alto, bajo y medio. La definición de rangos se puede observar en la **Tabla 19**.

	Rangos	**Alto**	**Medio**	**Bajo**
Enjambres (E)	1 a 10	10	5	3
Partículas (P)	1 a 20	20	12	2

Tabla 19. Rangos esperados de operación.

Variables Dependientes. Para la elaboración del protocolo experimental se definen dos variables dependientes y se contextualizan de la siguiente manera:

a) Tiempo de procesamiento (p_{ij}). Tiempo que demora una task (t_j) en ser procesada, desde que una workstation (e_i) empieza su ejecución hasta que finaliza. Los datos se expresan en segundos (s)
b) Tiempo de ejecución del algoritmo híbrido de enjambre (AHE). Es el tiempo que demora el algoritmo hibrido de enjambre (AHE) en calcular la solución al problema de scheduling. Los datos se expresan en segundos (s).

Resultados del Protocolo Experimental. Para el análisis de las variables dependientes se propone un diseño factorial de dos factores de tres niveles (alto, medio, bajo). A continuación, se analiza la variable tiempo de procesamiento (p_{ij}).

Las hipótesis que se desean probar están dadas por la Ecuación 7.

$$H_0: Efecto\ Partículas\ (P) = 0$$
$$H_A: Efecto\ Partículas\ \ (P) \neq 0$$

$$H_0: Efecto\ Enjambre\ (E) = 0$$
$$H_A: Efecto\ Enjambre\ \ (E) \neq 0$$

$$H_0: Efecto\ Partículas\ (P) \times Efecto\ Enjambre\ (E) = 0$$

$$H_A: Efecto\ Partículas\ (P) \times Efecto\ Enjambre\ (E) \neq 0$$

Ecuación 7. Hipótesis del diseño factorial realizado para la variable dependiente.

Para probar o rechazar las hipótesis planteadas en la Ecuación 7 se realiza para la variable tiempo de procesamiento (p_{ij}) un diseño factorial de dos factores y tres niveles. La Tabla 20 muestra el diseño factorial resultante.

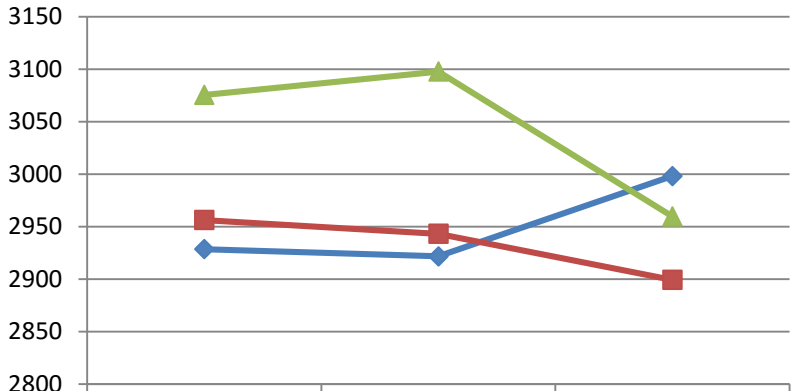
Fuente	**GL**	**SC**	**CM**	**F**	**P**
Partículas	2	21641,1	10820,6	3,10	0,154
Enjambres	2	2417,4	1208,7	0,35	0,726
Error	4	13946,6	3486,6		
Total	8	38005,1			

Tabla 20. Diseño factorial de dos factores tres niveles para la variable tiempo de procesamiento (p_{ij}).

Según los resultados mostrados se obtiene que debido a que ningún valor de *P* es menor que el valor de significancia fijado en $\alpha = 0.05$ se aceptan todas las hipótesis nulas. Esto quiere decir, que no son relevantes o significativos los factores partículas y enjambres para la variable tiempo de procesamiento (p_{ij}). Estos resultados coinciden con los resultados obtenidos en el protocolo experimental con datos sintéticos.

Se procede a realizar un diagrama de interacción por cada uno de los niveles que poseen los efectos. Es así como la Figura 13 muestra cómo al variar el efecto partículas (P) con el efecto enjambre (E) se encuentra una mejor respuesta del tiempo de procesamiento (p_{ij}), aunque no sea significativa $(\alpha = 0.05)$ la variación.

Por ejemplo, se encuentra una mejor solución del tiempo de procesamiento (p_{ij}) cuando las partículas (P) están en un nivel medio y los enjambres (E) en un nivel bajo. Estos resultados coinciden con los resultados obtenidos en el protocolo experimental con datos sintéticos.



	alto	medio	bajo
alto (P)	2928,491345	2921,589869	2998,139028
medio (P)	2956,256967	2943,05305	2899,251757
bajo (P)	3075,34598	3097,625553	2959,507797

Figura 13. Diagrama de interacción para enjambres por partículas para la variable tiempo de procesamiento ($\mathbf{p_{ij}}$).

Para la variable Tiempo de ejecución del algoritmo híbrido de enjambre (AHE) se realiza un diseño factorial de dos factores y tres niveles. En la Tabla 21 se muestra el diseño factorial resultante. Según los resultados expuestos se obtiene que debido a que el valor de P es menor que el valor de significancia fijado en $\alpha = 0.05$ son significantes los efectos partículas (P) y los enjambres (E) y se rechaza la hipótesis nula de la Ecuación 7. Esto quiere decir que todos los efectos anteriormente nombrados influyen en la variable dependiente tiempo de ejecución del algoritmo híbrido de enjambre (AHE).

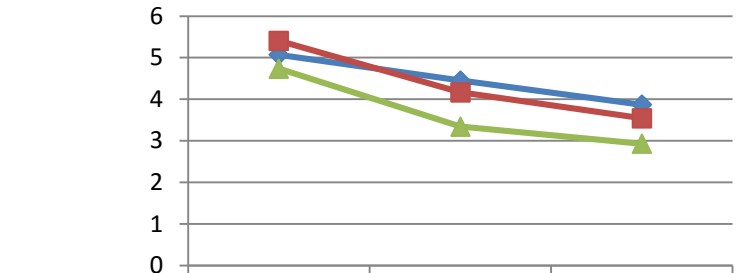
Con respecto a las demás hipótesis planteadas se aceptan las hipótesis nulas. Esto significa que ni el efecto, ni las combinaciones planteadas afectan significativamente la variable dependiente. En la Tabla 21 se marca la columna *P*se subraya el efecto significativo. En la columna F de la Tabla 21 se aprecia que el efecto más importante es enjambres (E) seguido de las partículas (P).

Fuente	**GL**	**SC**	**CM**	**F**	**P**
Partículas	2	113,111	0,56556	10,81	0,024
Enjambres	2	409,790	204,895	39,17	0,002
Error	4	0,20922	0,05230		
Total	8	543,823			

Tabla 21. Diseño factorial de dos factores tres niveles para la variable tiempo de ejecución del algoritmo híbrido de enjambre (AHE).

Se realiza un diagrama de interacción por cada uno de los niveles que poseen los efectos. La Figura 14 muestra como al variar el efecto partículas (P) con el efecto enjambre (E) se encuentra una mejor respuesta del tiempo de ejecución del algoritmo híbrido de enjambre (AHE), aunque no sea significativa ($\alpha = 0.05$) la variación.

Por ejemplo, se encuentra una mejor solución del tiempo de ejecución del algoritmo híbrido de enjambre (AHE) cuando las partículas (P) están en un nivel bajo y los enjambres (E) en un nivel bajo. Es consecuente con lo esperado debido a que menor cantidad de partículas (P) y menor cantidad de enjambres (E) el cómputo de todos los algoritmos se reduce.



	alto	medio	bajo
alto (P)	5,07	4,451	3,869
medio (P)	5,406	4,165	3,544
bajo (P)	4,735	3,344	2,93

Figura 14. Diagrama de interacción para enjambres por partículas para la variable tiempo de ejecución del algoritmo híbrido de enjambre (AHE).

Debido a que se debe escoger una configuración que logre optimizar la variable tiempo de procesamiento (p_{ij}) es necesario estudiar el comportamiento de esta variable con respecto al caso de estudio. En el caso de estudio la asignación de workstation (e_i) a las task (t_j) se realiza a través de una cola primero en entrar, primero en salir (FIFO).

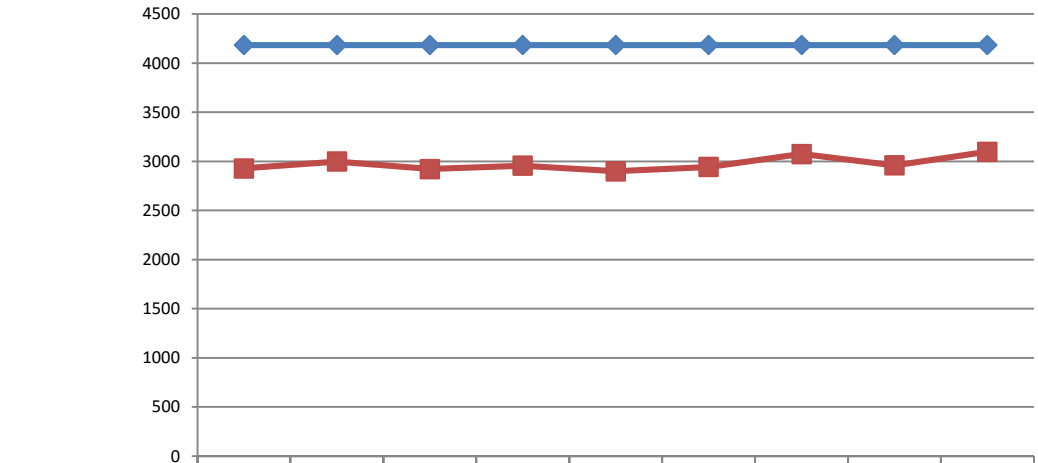
El cálculo real de la variable tiempo de procesamiento (p_{ij}) se realiza utilizando los tiempos persistidos en la base de datos de una cantidad determinada de task (t_j) de la operación de un Call Center. Con las mismas llamadas se simula la asignación de recursos a través del modelo de planificación predictivo (MPS). En la **Figura 15** se observa la mejora porcentual en promedio del 28% que presenta el uso del modelo desarrollado. Se ve que no existe mejora significativa si se varían los enjambres (E) o las partículas (P). El resultado concuerda con el estudio realizado con el diseño factorial, en el que se obtuvo que el mejor valor de la variable tiempo de procesamiento (p_{ij}) es partículas (P) en su nivel medio combinado con enjambres (E) en su nivel bajo.

Si se debe implementar en el mundo real el algoritmo híbrido de enjambre se elegiría la siguiente configuración teniendo en cuenta el resultado de los diseños experimentales:

1. Partículas (P) a nivel medio.
2. Enjambres (E) a nivel bajo.

3. Tasks (N) a nivel bajo.
4. Workstations (M) a nivel bajo.

La configuración elegida da la seguridad de obtener una buena respuesta al problema de scheduling en un tiempo corto de ejecución del algoritmo.



	alto	bajo	medio	alto	bajo	medio	alto	bajo	medio
	alto (P)			medio (P)			bajo (P)		
Real (FIFO)	4183	4183	4183	4183	4183	4183	4183	4183	4183
Calculado (AHE)	2928,49	2998,14	2921,59	2956,26	2899,25	2943,05	3075,35	2959,51	3097,63

Figura 15. Valores de la variable tiempo de procesamiento (p_{ij}) para datos reales del caso de estudio vs los valores simulados a través del modelo de planificación

Se concluye, a partir de los experimentos que el modelo de planificación predictivo (MPS), si se implementara, existiría una reducción en el tiempo de procesamiento (p_{ij}) del tasks y la solución del problema se realiza en un corto tiempo de ejecución.

CONCLUSIONES

1. Se analiza rigurosamente las técnicas de inteligencia de enjambre, en las que se selecciona el algoritmo de optimización por partículas (PSO). A su vez, se estudian las series de tiempo por las que se concluye que seleccionar la técnica adecuada en series de tiempo depende exclusivamente de los datos. Se escoge PSO debido a su flexibilidad, facilidad de hibridar y rápida convergencia. Por tal razón, se puede concluir una técnica de búsqueda local denominada búsqueda Tabú, que se hibrida PSO con búsqueda Tabú. Así mismo se realizo en su totalidad el modelo de planificación predictivo (MPS), con sus niveles jerárquicos, sus componentes y la forma en que su orquestan.

2. Se desarrolla el modelo de planificación predictivo (MPS), a través del paradigma de programación orientada a agentes. Es así como se genera el scheduler a corto plazo, cuyo componente principal es el algoritmo híbrido de enjambre. Adicional, se encuentra que se debe crear un simulador, que permite realizar el protocolo experimental.

3. A través del desarrollo del protocolo experimental se concluye que el modelo de planificación predictivo (MPS) logra el decremento de tiempo de procesamiento de un task en un 30% en promedio y, que al poseer un dosificador el scheduler a mediano plazo evita la sobrecarga del sistema y se obtiene un menor tiempo de ejecución del algoritmo. La asignación que realiza el algoritmo híbrido de enjambre es de forma inteligente, al utilizar información pasada para determinar las características del task y prever cuál workstation está capacitada para optimizar el tipo de procesamiento.

4. Con el desarrollo del modelo planteado, se encontró que es confiable, flexible, de implementación fácil y rápida de componentes por separado que puestos a prueba confirman que son aptos según el experimento llevado a cabo para los diferentes contextos.

BIBLIOGRAFÍA

[1] A. Duarte Muñoz, J. J. Pantrigo Fernández y M. Gallego Carrillo, Metaheurísticas, Madrid: Editorial Dykinson, 2007.

[2] C. Blum y A. Roli, «Metaheuristics in combinatorial optimization,» *ACM Computing Surveys,* vol. 3, nº 35, pp. 268-308, 2003.

[3] C. H. Papadimitriou y K. Steiglitz, Combinatorial Optimization, New York: Dover Publications, 1998.

[4] J. K. Lenstra, A. G. Rinnooy y K. P. Van Emde Boas, «An appraisal of computational complexety for operations research,» *European Journal of Operational Research,* p. 11, 1982.

[5] M. R. Garey y S. J. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, San Francisco: Freeman, 1979.

[6] M. Pérez y F. Sancho, Máquinas Moleculares Basadas en ADN, Sevilla: Universidad de Sevilla, 2003.

[7] C. E. Maldonado, «Heurística y producción de conocimiento nuevo en la perspectiva CTS,» de *Estética, ciencia y tecnología* , Bogotá, Editorial Pontificia Universidad Javeriana, 2005, pp. 98-127.

[8] R. Baños, «Universidad de Almería - Doctorado en Ciencias de la Computación,» Diciembre 2006. [En línea]. Available: http://cms.ual.es/UAL/en/estudios/doctorado/tesis/tesisdoctoral/DOCTORADO8705?plan=8021&anyo=2006-07&exp_numord=10&tes_codnum=1. [Último acceso: 24 Septiembre 2013].

[9] M. Pinedo, Scheduling: Theory, Algorithms and Systems, Prentice Hall, 1995.

[10] C. R. Gatica y S. C. Esquivel, *Algoritmos Híbridos para el problema de scheduling de máquinas paralelas,* San Luis, 2010.

[11] A. Punnen, The Traveling Salesman Problem and its Variations, Kluwer, 2004.

[12] M. Á. González Fernández, «Soluciones Metaheurísticas al "Job-Shop Scheduling Problem with Sequence-Dependent Setup Times",» Oviedo, 2011.

[13] O. Holthaus y C. Rajendram, «Efficient dispatching rules for scheduling a job shop,» *International Journal of Production Economics,* vol. 1, nº 48, pp. 87-105, 1997.

[14] S. Lawrence, «Resourse constrained proyect scheduling: an experimental investigation of heuristic scheduling techniques (supplement),» Technical Report, Graduate School of Industrial Administration, Carnegie Mellon University, 1984.

[15] C. Artigues, P. López y P. Ayache, «Schedule generation schemes for the job shop problem with sequence-dependent set up times: dominance properties and computational analysis.,» *Annals of Operations Research,* nº 138, pp. 21-52, 2005.

[16] R. Guerequeta y A. Vallecillo, Técnicas de diseño de algoritmos, Málaga: Servicio de publicaciones de la Universidad de Málaga, 2000.

[17] R. Bellman, Adaptive Control Processes: A Guided Tour, Princeton: Princeton University Press, 1961.

[18] F. Glover, «Heuristic for integer programming using surrogate constraints,» *Decision Sciences,* nº 8, pp. 533-549, 1977.

[19] J. P. Kelly y I. H. Osman, Meta-heuristic: Theory and Applications, Kluwer Academic Publisher, 1996.

[20] S. Luke, Essentials of Metaheuristics, Department of Computer Science George Mason University, 2013.

[21] F. Glover, M. Laguna y R. Martí, Principles of Tabu Search, Taylor & Francis Group, LLC, 2007.

[22] R. Martí, «Procedimientos metaheurísticos en optimización combinatoria,» *Matemátiques,* vol. 1, nº 1, pp. 3-62, 2000.

[23] S. Kirkpatrick, J. C. Gelatt y M. P. Vecchi, «Optimization by simulated annealing,» *Science,* nº 220, pp. 671-680, 1983.

[24] E.-G. Talbi, Metaheuristics: from design to implementation, New Jersey: John Wiley & Sons, Inc., 2009.

[25] A. S. Fraser, «Simulation of genetic systems by automatic digital computers,» *Australian Journal of Biological Science,* vol. 10, pp. 484-491, 1957.

[26] G. E. Box, «Evolutionary operation: a method of increasing industrial productivity,» *Applied Statistics,* vol. 10, pp. 81-101, 1957.

[27] J. H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, 1975.

[28] J. K. a. R. C. E. J. F. Kennedy, Swarm Intelligence, Morgan Kaufmann, 2001.

[29] D. Karaboga y B. Basturk, «A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm,» *Springer Science,* 2007.

[30] R. C. Eberhart, Neural network PC tools, Academic Press, 2001.

[31] J. P. Witenberg, Métodos y modelos de investigación de operaciones, 1981.

[32] J. E. M. Delgado, «Optimización de la Programación (scheduling) en Talleres de Mecanizado,» Madrid, 2012.

[33] «Real Academia Española.,» 2013. [En línea]. Available: http://lema.rae.es/drae/?val=perfil. [Último acceso: 04 11 2013].

[34] M. L. R. M. F. Glove, Principles of Tabu Search, Taylor & Francis Group, LLC, 2007.

[35] J. M. G. N., «Algoritmos Basados en Cúmulos de Partículas Para la Resolución de Problemas Complejos,» 2006.

[36] C. L. L. B. P. S. Arias., «Paradigma Orientado a Agentes,» 2010.

[37] L. Hernández López, «Heurísticas para el control deliberativo en una arquitectura de agentes inteligentes de tiempo real,» Universidad Politécnica de Valencia. Departamento de Sistemas Informáticos y Computación., 2004.

[38] F. Bellifemine, G. Caire, T. Trucco y G. Rimassa, «Jade Programmers's Guide,» 2010.

[39] «Java Agent DEvelopment Framework,» Italian M.I.U.R, 01 03 2013. [En línea]. Available: http://jade.tilab.com/. [Último acceso: 02 12 2013].

[40] K. Kotecha y A. Shah, «Adaptive scheduling algorithm for real-time operating system,» *Evolutionary Computation,* 2008.

[41] M. G. Pose, «Introducción a los Algoritmos Genéticos,» [En línea]. Available: http://sabia.tic.udc.es/mgestal/cv/AAGGtutorial/aagg.html. [Último acceso: 2 Enero 2016].

[42] K. Gkoutioudi y H. Karatza, «A Simulation Study of Multi-criteria Scheduling in Grid Based on Genetic Algorithms,» *Parallel and Distributed Processing with Applications (ISPA),* pp. 317,324, 2012.

[43] J. M. S. Leyes, G. Joan Baptista Fonollosa y A. S. Torrents, Métodos cuantitativos de organización industrial II, Univ. Politèc. de Catalunya, 2009.

[44] H. Gutiérrez Pulido y R. de la Vara Salazar, Análisis y Diseño de Experimentos, Mexico: McGraw Hill, 2012.

[45] R. L. A. a. M. W. Sasieni, Fundamentos de investigación de operaciones, Limusa Noriega, 1994.

[46] J. Kennedy y E. Russell C., Swarm Intelligence, Morgan Kaufmann, 2001.