SCHEDULING EN LÍNEA MINIMIZANDO LA TARDANZA TOTAL PONDERADA PARA UNA MÁQUINA MEDIANTE METAHEURÍSTICA GRASP

CAMILO BERNAL NISPERUZA

Trabajo de Grado

Director: Carlos Alberto Vega Mejía
Ingeniero de Sistemas
MSc. Ingeniería Industrial

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA INDUSTRIAL
BOGOTÁ D.C.
2012

Nota de aceptación

_____

_____

_____

_____

_____

_____

_____

Firma del jurado

_____

Firma del jurado

Bogotá D.C., Julio 5 de 2012

“A mi familia, quienes han creído que en la educación está la mejor inversión de la vida.”

AGRADECIMIENTOS

Quiero agradecer a la Pontificia Universidad Javeriana y a la Universidad Nacional de Colombia, por enseñarme los conceptos que constituyen la profesión de la Ingeniería y, más allá de eso, que existe un país maravilloso por el que vale la pena prepararse y trabajar todos los días.

A mis profesores y amigos, quienes han sido un soporte valioso para transitar un camino que no ha sido fácil y que, con este trabajo, logra culminar uno de los tramos más importantes.

A mi director de proyecto, Carlos Vega, de quien aprendí a tener paciencia con cada tropiezo en la construcción de este trabajo, y a mis jurados por haberme ayudado en la concepción de la idea.

Finalmente, agradezco a mi familia que me ha apoyado en el capricho de convertirme en un doble profesional, haciendo los mayores sacrificios en estos últimos años y en quienes he encontrado el motor más importante para exigirme en la elaboración de las páginas que se presentan a continuación.

TABLA DE CONTENIDO

1 INTRODUCCIÓN 1

2 ANTECEDENETES 3

2.1 Problema de *scheduling* 3

2.2 El problema de minimizar la tardanza ponderada en una máquina 3

2.3 Variaciones al problema 1||ΣwiTi 4

2.4 Métodos de solución 4

2.5 Problemas *on-line* 5

3 PLANTEAMIENTO DEL PROBLEMA 20

4 JUSTIFICACIÓN DEL PROBLEMA 21

5 MARCO TEÓRICO 23

5.1 Entorno de la maquinaria (α) 23

5.2 Características del trabajo (β) 23

5.3 Medidas de desempeño (γ) 25

5.4 Problema 1||ΣwiTi 25

5.5 Complejidad del problema 25

5.6 Problemas *On-line* 26

5.7 Metaheurísticas 27

5.7.1 Metaheurística GRASP 27

5.8 Parámetros definidos para las instancias 28

6 OBJETIVOS 29
6.1 Objetivo General 29
6.2 Objetivos Específicos 29
7 DESARROLLO DEL PROBLEMA 30
7.1 Llegada de los trabajos en lotes 31
7.2 Metaheurística 33
7.2.1 Fase constructiva 34
7.2.2 Fase de búsqueda Local 36
8 RESULTADOS 38
8.1 Programación de los trabajos 39
8.2 Medidas de interés 44
8.3 Estadísticas descriptivas para todas las pruebas 46
8.4 Correlación de variables 52
8.4.1 40 Trabajos 55
8.4.2 50 Trabajos 56
8.4.3 100 Trabajos 58
9 EXPERIMENTO ESTADÍSTICO 61
10 CONCLUSIONES 70
11 BIBLIOGRAFÍA 72
12 ANEXO 1: Resultados Computacionales vi

13 ANEXO 2: Medidas de interés vi

14 ANEXO 3: Una mirada al problema *off-line* vii

15 ANEXO 4: Prueba de Tamhane para la interacción de los parámetros RDD y TF… vi

16 ANEXO 5: Función ‘randi’ de MATLAB vi

1 INTRODUCCIÓN

Programar actividades es una tarea que consciente o inconscientemente todos aplicamos para planear nuestras actividades diarias. Para cualquier persona se hace necesario plantear y evaluar el orden en las que éstas deben ser desarrolladas dependiendo de factores externos y el criterio único del individuo. De igual forma, la planeación de la producción en la industria requiere organizar los trabajos de una manera óptima para aprovechar al máximo unos recursos limitados (máquinas por ejemplo) y cumplir con objetivos enmarcados por la estrategia de la organización (un ejemplo es la política *Just-in-Time*). Sin embargo, a diferencia de un individuo, la industria cuenta con especializadas herramientas analíticas cuyo único propósito es apoyar a la toma de decisiones, y a lo que se conoce comúnmente dentro del campo de la investigación de operaciones como *scheduling*.

Uno de los problemas que se ha estudiado en las últimas cuatro décadas, consiste en la programación de tareas cuya entrega luego del plazo establecido conlleva penalidades y, por lo tanto, se pretenden minimizar la suma de todas ellas. Su aplicación se debe a que, tal como dicen Altunc y Keha (2009), “Con frecuencia hay ciertas penalidades contractuales aplicadas al proveedor para los trabajos tardíos. Dichas penalidades son mayores para los trabajos estratégicos”. No obstante, computacionalmente, resolver este problema resulta tan difícil que requiere de un tiempo exponencial para descubrir una solución. Además, la solución misma sería tan extensa que no podría ser descrita con una expresión de longitud limitada mediante una función polinómica de los datos de entrada (GAREY & JOHNSON, 1979). Por lo tanto, este problema se considera dentro del grupo de los posibles problemas intratables, lo cual explica el gran número de investigaciones que se han desarrollado en procedimientos más certeros y rápidos que van desde los muy sofisticadas hasta simples heurísticas (PINEDO, Scheduling Theory, Algorithms and Systems, 2002).

Adicionalmente, los investigadores han planteado diferentes características de los trabajos dependiendo de su interés, lo que ha permitido abrir el horizonte de investigación para este problema en particular. En consecuencia, el presente proyecto plantea realizar un desarrollo computacional que permita darle solución a la minimización de la tardanza total ponderada en una máquina añadiendo la característica *on-line*, es decir, que la información de los trabajos a desarrollar no se conoce con anticipación a la llegada de los mismos. Lo anterior, se plantea solucionar mediante una metaheurística llamada procedimiento de búsqueda miope aleatorizado y adaptativo (GRASP, por sus siglas en inglés), de tal forma que se intente llegar a respuestas factibles y aceptables en poco tiempo.

El proyecto se ha desarrollado con miras a descubrir una pequeña parte del mundo de la programación de la producción, considerando que éste es un universo apenas conocido,

en el que un problema tiene un sinnúmero de caminos para llegar a una o múltiples soluciones, y constituye un reto para quienes gustan de las matemáticas, la computación y la optimización de las operaciones. Las siguientes páginas están dispuestas para que el lector considere uno de esos planteamientos, e igualmente se le invita a que sea reflexivo y crítico a la vez para que siga abriendo paso a la investigación en estos temas. Sus hallazgos, que puedan traducirse en oportunidades de mejora, son consideradas de antemano un recurso valioso para continuar investigando y descubriendo que aún nos falta mucho por descubrir.

Finalmente, se le pide al lector que vuelva a pensar en el individuo que planea sus tareas diarias, las cuales pudiesen incurrir en castigos por atenderlas fuera del plazo establecido (como perder una cita si no se llega a la hora indicada) y cuyas características sólo se conocen en el instante en que aparece la necesidad de desarrollarlas. Si ese llegara a ser su caso, el presente proyecto puede darle una visión desde la perspectiva matemática con la que se plantea solucionar este problema, aunque también le muestra un panorama de lo que puede estar sucediendo en nuestras cabezas y, de esta manera, pueda asombrarse de la maravillosa herramienta que es nuestro cerebro.

2 ANTECEDENETES

2.1 Problema de *scheduling*

Vincent Wiers (1997) menciona en su tesis de doctorado que “una vez antes de que existiera la programación de la producción en la mente de los administradores, hubo una época en la que las fábricas no sabían cuándo empezaba un trabajo, dónde estaba, cómo se movía en la planta y cómo debía ser realizado”. Ese hecho fue quizá un motivo por el cual el estudio de las operaciones industriales le haya dado relevancia a la programación de la producción desde hace más de un siglo.

Se podría decir que diferentes herramientas como el diagrama desarrollado por Henry Gantt a principios del siglo XX e incluso la división del trabajo por parte de Taylor, condujeron a una nueva perspectiva de la programación de trabajos en la producción (HERRMANN, 2011). Sin embargo, el primer método analítico para darle solución a un problema de organización de trabajos se dio en el año de 1956 cuando los ingenieros James Kelley y Margen Walker desarrollaron un método basado en grafos llamado CPM (*Critical Path Method*) capaz de asignar únicamente las tareas necesarias a una computadora, de tal forma que se disminuyera el consumo de tiempo total en desarrollarlas todas y así minimizar el costo de funcionamiento. En 1959, juntos presentaron el método en un artículo titulado “*Critical Path Planning* and *Scheduling*” que poco después sirvió como punto de partida para el desarrollo de metodologías de solución a problemas similares aplicados a ambientes industriales. Finalmente, el desarrollo de computadores personales y el incremento de la capacidad de cómputo, hicieron que los problemas de *scheduling* se pudieran solucionar con procedimientos más estructurados a un gran número de problemas particulares. Hoy en día, la programación de la producción es vista como una profesión y como parte del epicentro de los proyectos exitosos. (WEAVER, 2006)

2.2 El problema de minimizar la tardanza ponderada en una máquina

Uno de los problemas de *scheduling* consiste en minimizar la sumatoria de las tardanzas ponderadas, escrita como $1||\Sigma w_l T_l$ en la notación de Graham (cit. por BRUCKER, 2007) o también llamado Single-machine TWT (*Total Weighted Tardiness*) por Sen, Sulek y Dileepan (2003). El objetivo de este problema consiste en organizar un conjunto de trabajos con características propias para ser procesados de forma independiente en una máquina, de tal forma que la suma de las penalidades por entregar trabajos en un momento posterior a la fecha convenida, sea mínima. Lo anterior supone una limitación de la máquina para cumplir con todos los trabajos dentro de los tiempos establecidos.

Este problema fue estudiado inicialmente por el investigador Salah E. Elmaghraby (SEN et al., 2003) cuando trataba de dar solución al problema de minimizar la suma de las tardanzas a un conjunto de trabajos con una fecha límite de entrega. Lo que propuso en

1968 fue darle un peso a cada uno de esas tardanzas y resolver la situación usando un modelo de redes y la metodología *branch-and-bound*. A partir de esa fecha, se han desarrollado numerosas investigaciones que tocan este mismo tema para una sola máquina (hasta el momento se han encontrado alrededor de 50 artículos en la revisión que se ha hecho al respecto) teniendo en cuenta que se trata de un problema *NP-Hard*; es decir, que un algoritmo diseñado para alcanzar una solución única exacta tardaría mucho tiempo y por lo tanto, se pueden aplicar algoritmos de aproximación tales que produzcan soluciones que, a su vez, garanticen estar dentro de un porcentaje del óptimo conocido (BRUCKER, 2007). Además, ha sido considerado como uno de los criterios más importantes en sistemas prácticos (LIAO & JUAN, 2007).

2.3 Variaciones al problema 1||ΣwiTi

En un problema de *scheduling* existe un gran número de funciones objetivo que varían dependiendo de la necesidad del autor. Asimismo, al igual que existe una gran variedad de funciones objetivo para un problema de *scheduling*, dentro de cada uno de ellos existen un número aún mayor de restricciones que han sido tenidas en cuenta por investigadores atendiendo a requerimientos específicos de un sistema. Para la función objetivo Single-Machine TWT, por ejemplo, Ying, Lin y Huan (2009) proponen una función en la que los trabajos requieren de un alistamiento previo de la máquina y éste depende del tiempo de procesamiento que requiere cada trabajo. Tasgetiren, Pan y Liang (2009) toman en cuenta los tiempos de preparación aunque, a diferencia de Ying et al. (2009), no demuestran dependencia de éstos con ningún otro parámetro. Huyn y Soukhal (2010) afirman que sus trabajos tienen la misma duración mientras que Della Croce, Desmier y Garaix (2011) evitan cualquier tiempo ocioso de la máquina.

Uno de los parámetros con mayor aparición en los artículos en mención tiene que ver con la fechas de vencimiento, por ejemplo: Kellerer y Strusevich (2006) estudian la situación en la que todos los trabajos tienen una fecha de vencimiento común; Bozeijko, Grabowsky y Wodecki (2006) las asumen independientes; mientras que Alidaee y Dragan (1997) o Gordon y Tarasevich (2009) definen ventanas de vencimiento común para los trabajos.

Finalmente, autores como Nemanee y Reodecha (2009), y Akturk e Ilhan (2011) han hecho aplicaciones reales de los problemas a una máquina de circuitos impresos (PCB) y una máquina CNC respectivamente.

2.4 Métodos de solución

En la revisión bibliográfica mencionada, se ha encontrado que los investigadores se valen de heurísticas y metaheurísticas para darle solución al problema con diversas restricciones. Cabe aclarar que la gran mayoría ellos, combinan las funcionalidades de diferentes metodologías para conseguir el objetivo y muchos otros crean sus propios procedimientos.

Para empezar, el método exacto *branch-and-bound* es el que aparece un mayor número de veces dentro de la revisión en cuanto a que, además de Elmagraby, ha sido seguido por otros autores citados por Sen et al. (2003) tales como Rinnooy Kan et al.(1975), Picard y Queyranne (1978) o Potts y Van Wassenhove (1985), teniendo continuidad con documentos recientes expuestos por Yin et al. (2011) y Schaller et al. (2011).

Por su parte, se ha encontrado otro gran grupo de metaheurísticas enmarcadas por los algoritmos de búsqueda. Tal es el caso de Grosso, Della Croce y Tadei (2004) quienes desarrollan una búsqueda dinámica al igual que lo hacen Angel y Bampis (2005). Estos últimos, a su vez, lo combinan con una búsqueda local demostrando un mejor desempeño gracias a dicha combinación. Wan y Yen (2002) aplican un algoritmo llamado temporizador junto con una búsqueda tabú, y ésta última es igualmente aplicada por Bozeijko et al. (2006) y Bilge, Kurtulan y Kiraç (2007) que lo hacen de la mano con lo que ellos han denominado vecindarios híbridos. Por último, la metodología *beam-search* es propuesta por Valente y Alves (2008) como una estrategia que permite parámetros variables logrando, según concluyen, mejores resultados que los parámetros fijos.

Finalmente, otras metaheurísticas tales como algoritmos genéticos, colonia de hormigas, GRASP (*Greedy Randomized Adaptative Search Procedure*), PSO (*Particle Swarm Optimization*) y programación dinámica han sido aplicadas por Chang et al. (2009), Liao y Juan (2007), Tasgetiren et al. (2009), Vega-Mejía y Caballero-Villalobos (2010) o Caballero-Villalobos y Alvarado-Valencia (2010); Anghinolfi y Paolucci (2009); y Kellerer y Strusevich (2006) respectivamente, demostrando en cada uno de estos la aplicabilidad de estas metaheurísticas al problema de *scheduling*.

2.5 Problemas *on-line*

En la última década se ha notado que el problema *on-line* ha tomado gran importancia entre los investigadores. Como muestra, se han encontrado artículos con esta característica que toman en cuenta como función objetivo la minimización de tiempos de entrega, tiempos de flujo, tiempos de terminación de los trabajos, entre otros. En primer lugar, se hace mención al caso de los investigadores Tian, Fu y Yuan con artículos publicados en los años (2011) (2008) (2007) quienes han estudiado el problema de minimizar el tiempo de entrega de productos que han de ser procesados por lotes y entregados a un cliente remoto. Sus artículos se han basado en la variación del parámetro de tiempos de entrega, para lo cual han encontrado que se trata de problemas totalmente diferentes cuando esa variable tiene un valor mayor o menor que el tiempo de procesamiento. Por su parte, autores como Sitters (2010), Zhang y Wirth (2009), Nong, Yuam, Fu, Lin y Tian (2008), Stee y La Putré (2005), Montoya-Torres (2003), Kaminsky y Simchi-Levi (2001), o Fiat y Woeginger (1999) tienen en cuenta la función objetivo de minimizar el tiempo de terminación de los trabajos. Sin embargo, el concepto de *on-line* varía de unos a otros dado que, por ejemplo, para Montoya-Torres (2003) los tiempos de llegadas de los trabajos son conocidos y los tiempos de procesamiento no, mientras que

para Nong et al, (2008) todas las características son desconocidas hasta que el trabajo llega a la máquina.

Aquí, al igual que en los problemas presentados para minimizar la tardanza total ponderada, las restricciones propuestas por cada uno de los autores son diferentes en cada caso, por lo cual las máquinas tienen una reacción diferente. Para terminar, se ha encontrado que la mayoría de los artículos revisados solucionan el problema mediante algoritmos propios basados en reglas simples de despacho tales como SPT (*Shortest processing time*), SRPT (*Shortest remaining processing time*), entre otros, a los cuales les realizan variaciones ajustadas al tipo de problemas que desean resolver.

Para una revisión más detallada de las publicaciones consultadas, se invita al lector a que revise las Tablas 1 y 2 que contienen la consulta de cada uno de los artículos revisados para la presente propuesta.

Tabla 1: Consulta sobre artículos que traten el problema TWT o similares

CONSULTA DE ARTÍCULOS CIENTÍFICOS					
Autores	**Año**	**Problema a tratar**	**Metodología**	**Consideraciones**	***On-line***
Schaller, J. Valente, J.	**2011**	***Scheduling* para una sola máquina en el que se minimice la tardanza ponderada al cuadrado.**	**Desarrollan un algoritmo *branch-and-bound*. Varias condiciones dominantes se incluyen dentro del algoritmo.**	**Se prueba el algoritmo con varios números de trabajos, fechas de vencimiento próximas y con rangos en las mismas.**	
Akturk, M. Ilhan T.	**2011**	***Scheduling* para una máquina CNC cuyo objetivo es minimizar la tardanza ponderada y los costos de herramienta y procesamiento**	**Desarrollan un algoritmo dinámico que considera la interacción entre los trabajos en una determinada secuencia**	**Aprovechan la propiedad que tienen los mecanismos CNC para controlar tiempos de procesamiento a expensas del desgaste de la herramienta**	**X**
Yin, Y. Wu, C. Wu, W. Cheng, S.	**2011**	**Minimizar la tardanza ponderada con efecto de aprendizaje basado en la posición.**	**Desarrollan un algoritmo ramificado para llegar a la solución óptima.**	**Desarrollan otras tres heurísticas para óptimos cercanos.**	
Yin, Y. Xu, D.	**2011**	**Solución a los problemas de *makespan*, suma de la k-ésima potencia de los tiempos de terminación, total de retrasos y suma de las prontitudes ponderadas.**	**Muestran un modelo general de programación de la producción basado en aprendizaje y deterioro que da solución a un gran número de problemas de *scheduling*.**		
Wang, J. Li, J.	**2011**	**Solucionan múltiples problemas dentro de los que se incluye la minimización de la tardanza máxima**	**Desarrollan un algoritmo dependiente de la posición y el tiempo con efectos de aprendizaje. El trabajo no es una función única de del tiempo de procesamiento sino de la posición que ocupa en la programación también.**	**Los tiempos de alistamiento dependen de los trabajos que ya han sido procesados.**	

CONSULTA DE ARTÍCULOS CIENTÍFICOS					
Autores	**Año**	**Problema a tratar**	**Metodología**	**Consideraciones**	***On-line***
Della Croce, F. Garaix, D.	**2011**	**Minimizar la tardanza total ponderada sin tiempos ociosos de máquina.**	**Si se aplica el procedimiento de búsqueda dinámica al problema resuelto por Valente, se logra la mejor heurística para minimizar la tardanza total ponderada sin tiempos ociosos en una búsqueda con haz.**	**Afirman que su algoritmo debería ser usado como un punto de referencia para otros trabajos en el que se involucren esos parámetros.**	
Caballero-Villalobos, J. Alvarado-Valencia, J.	**2010**	**Minimizar la tardanza total ponderada en una máquina**	**Desarrollan la metaheurística GRASP obteniendo los mejores resultados para las instancias de OR-library en la mayoría de los casos.**	**En la fase constructiva, la lista de elegibles es siempre constante.**	
Huyn, N. Soukhal, A.	**2010**	**Asignación de tiempos de entrega (due dates) de trabajos con igual tamaño para una máquina y máquinas paralelas**	**Reducen la complejidad de este tipo de problemas convirtiendo la situación en un problema de asignación sin enumerar los tiempos de entrega**		
Tavakkoli-Moghaddam, R. Javadi, B. Jolai, F. Ghodratnama, A.	**2010**	**Minimizar la tardanza total ponderada y el *makespan* simultáneamente.**	**Formulan programación linear difusa para multiobjetivo (FMOLP) con respecto a un nivel de satisfacción aceptado a nivel general, dado por un tomador de decisiones (DM).**	**Comparan resultados con la aproximación realizada por Wang y Liang. Demuestran que el algoritmo alcanza pequeñas funciones objetivos y altos grados de satisfacción.**	
Geiger, M.	**2010**	**Minimizar la tardanza total ponderada para una máquina.**	**Se trata de una investigación de la heurística de búsqueda investigada a fondo por el autor y la respectiva comparación con otras heurísticas**		
Cheng, T. Lee, W. Wu, C.	**2010**	**Minimizar el *makespan* aunque afirman que la solución planteada se puede aplicar a la minimización de la tardanza máxima.**	**Se considera un modelo de deterioro en el que el tiempo de procesamiento del trabajo actual es una función de los tiempos de procesamiento de los trabajos que ya han sido realizados**		

CONSULTA DE ARTÍCULOS CIENTÍFICOS					
Autores	**Año**	**Problema a tratar**	**Metodología**	**Consideraciones**	***On-line***
Tasgetiren, M. Pan, Q. Liang, Yun.	**2009**	**Minimizar la tardanza total ponderada cuya secuencia depende de los tiempos de preparación de la máquina**	**Desarrollan un algoritmo evolutivo donde convergen las heurísticas NEH y GRASP así como reglas de prioridad como EWDD y ATCS. Lo mejoran con Búsqueda local y algoritmo de construcción destrucción.**	**Lo comparan con algoritmos recientes basados en PSO y Colonia de hormigas y muestran que es mucho mejor tanto en la calidad de la solución como en el tiempo de procesamiento**	
Neammanee, P. Reodecha, M.	**2009**	**Minimizar la tardanza ponderada con tiempo de alistamiento dependientes de la secuencia y con capacidad de alimentación para una PCB**	**Se basan en un *memetic algorithm* integrando GA, Mínimo tiempo de holgura, la política "*Keep Tool Needed Soonest*" (KTNS) y búsqueda local.**	**Al final obtienen la programación de la máquina y el plan de alistamiento de la alimentación. El tiempo de computación es alto pero aseguran que muestra mejores soluciones para este tipo de problemas.**	
Chang, P. Chen, S. Mani, V.	**2009**	**Minimizar la tardanza y prontitud ponderada para una sola máquina.**	**Desarrollan un algoritmo genético híbrido que se compone del GA sencillo y propiedades de dominancia.**	**Demuestran que el criterio de propiedades dominantes consiste en intercambiar trabajos adyacentes y, según los autores, es necesario pero no suficiente para ser óptimo.**	
Ying, K. Lin, S. Huang, C.	**2009**	**Minimizar los trabajos con tardanza ponderada y sin ponderar para una secuencia con tiempos de preparación dependientes**	**Desarrollan *iterated greedy heuristic*.**	**Demuestran que es bastante efectiva comparada con otras metaheurísticas para casos similares.**	
Chou, F.	**2009**	**Minimizar la tardanza total ponderada para una máquina.**	**Resuelven el problema usando un algoritmo genético con aprendizaje experimentado (ELGA) basado en matrices dinámicas para la generación de cromosomas**	**Anuncian que ELGA es robusto debido a que la desviación estándar del porcentaje de diferencia entre las soluciones es muy pequeño**	

CONSULTA DE ARTÍCULOS CIENTÍFICOS					
Autores	**Año**	**Problema a tratar**	**Metodología**	**Consideraciones**	***On-line***
Colack, A. Burak, A.	**2009**	**Minimizar la tardanza total ponderada para una máquina.**	**Desarrollan la solución mediante el uso de programación entera y programación lineal basadas en algoritmos heurísticos. La formulación del problema se hace mediante *Interval-indexed***	**Muestran varios métodos para formar los intervalos y métodos post-procesos. Además muestran como el algoritmo mejora la selección de una población en un algoritmo genético además de poder ser usado en otros problemas combinatorios**	
Anghinolfi, D. Paolucci, M.	**2009**	**Minimizar las tardanzas ponderadas en presencia de dependencia de la secuencia de los tiempos de alistamiento**	**Proponen el uso de PSOP discreta en la que las partículas y las velocidades tienen una medida coherente. Concluyen que es la mejor aplicación que se ha hecho de DPSO**	**Realizan una investigación de los mejores algoritmos propuestos para ello.**	
Wang, X. Tang, L.	**2009**	**Minimizar la tardanza total ponderada para una máquina.**	**Proponen el algoritmo de b{búsqueda de vecindario variable basado en poblaciones. Consiste en la iteración del la búsqueda de vecindarios variables junto con el re direccionamiento de la búsqueda para lograr mayor intensificación**	**Según afirman, ese algoritmo llega a la mejor solución dentro de un tiempo razonable y, por lo tanto, es el mejor de los que se encuentran en la literatura.**	
Cheng, T. Lai, P. Wu, C. Lee, W.	**2009**	**Su problema consiste en disminuir el *makespan*.**	**Proponen un modelo de aprendizaje donde el tiempo de procesamiento del trabajo actual es una función de los logaritmos de los tiempos de procesamiento de los trabajos que ya han sido procesados.**	**Afirman que su metodología se extiende a la minimización de la máxima tardanza y tardanza total**	
Gordon, V. Tarasevich, A.	**2009**	**Minimizar la prontitud, tardanza y los costos de vencimiento ponderados con fecha de vencimiento común.**	**Plantean una solución basada en la modificación de la tasa de actividad para disminuir el valor de la función objetivo.**	**Dicen que en varios estudios esta metodología puede disminuir la complejidad del problema.**	

CONSULTA DE ARTÍCULOS CIENTÍFICOS					
Autores	**Año**	**Problema a tratar**	**Metodología**	**Consideraciones**	***On-line***
Lee I. Sung, C.	**2008**	**Minimizan el retraso (L) y los costos de tercerización ponderados para la programación de una máquina y la tardanza total con la posibilidad de tercerizar trabajos en otra idéntica.**	**La solución que proponen es la integración de heurísticas y el algoritmo *branch-and-bound*.**		
Valente, J. Alves, R.	**2008**	**Minimizar la tardanza total ponderada de trabajos con alistamientos dependientes de la secuencia.**	**Proponen la técnica de *beam-search*. Desarrollan nuevas versiones del algoritmo en el que varíe el haz y el ancho del filtro.**	**Con la metodología usada, se determina que la solución con parámetros variables es mejor que la de parámetros fijos. Sin embargo, para problemas largos, el resultado es bastante complicado.**	
Chen, W. Sheen, W.	**2007**	**Minimizar la prontitud y tardanza ponderada sujeta al número de trabajos que se entregan tarde con fecha de vencimiento común para todos los trabajos.**	**Proponen un algoritmo de pareto en el que se generan las soluciones óptimas de pareto para cada una de los trabajos tardíos.**	**Comparan con trabajos anteriores para demostrar su exactitud y mejoramiento del tiempo de procesamiento. Muestran que aproximadamente el 47,15% de los nodos en los árboles pueden ser eliminados.**	
Tang, L. Xuan, H. Liu, J.	**2007**	**Minimizar la tardanza total ponderada con restricciones de precedencia.**	**Diseñan un algoritmo de programación dinámica híbrida (progresiva y regresiva) para el problema de relajación Lagrangiano.**	**Determinan que el algoritmo propuesto puede resolver problemas con múltiples trabajos y que se desempeña mejor que una solución Lagrangiana únicamente progresiva.**	

CONSULTA DE ARTÍCULOS CIENTÍFICOS					
Autores	**Año**	**Problema a tratar**	**Metodología**	**Consideraciones**	***On-line***
Bilge, Ü. Kurtulan, M. Kiraç F.	**2007**	**Minimizar la tardanza total ponderada.**	**Proponen una búsqueda tabú con parámetros determinísticos, vecindarios híbridos y estructura de permanencia dinámica, e investiga estrategias para mejorar el desempeño en problemas específicos.**	**Afirman que su algoritmo produce una buena calidad de los resultados para un gran número de problemas encontrados en la literatura.**	
Liao, C. Juan, H.	**2007**	**Minimizar la tardanza total ponderada con tiempos de alistamiento dependientes**	**Proponen un algoritmo de colonia de hormigas para darle solución al problema.**	**Comparan con otros problemas y afirman que es mejor que varios de ellos. Además prueban con la versión del problema sin ponderanciones.**	
Bozejko, W. Grabowski, J. Wodecki, M.	**2006**	**Minimizar la tardanza total ponderada en el que cada trabajo tiene un tiempo de procesamiento y una fecha límite**	**Desarrollan un procedimiento de búsqueda local basado en el algoritmo tabú con un vecindario específico que emplea bloques de trabajo y una técnica de movimientos simultáneos.**	**Es un problema determinísitco y lo comparan con heurísticas en ese mismo sentido.**	
Kolliopoulos, S. Steiner, G.	**2006**	**Minimizar la tardanza total ponderada para una máquina.**	**Se dan dos algoritmos. El primero soluciona el problema con un número fijo de fechas de vencimiento. El segundo, es un algoritmo cuasi-polinómico en el que se pueden tener un número de fechas de vencimiento y tiempos de procesamiento arbitrarios.**		
Kellerer, H. Strusevich, V.	**2006**	**Minimizar al tardanza total ponderada en el que todas las fechas de vencimiento son iguales.**	**Desarrollan un esquema de aproximación de tiempo polinomial completo obtenido mediante un algoritmo de programación dinámica pseudopolinomial.**		

CONSULTA DE ARTÍCULOS CIENTÍFICOS					
Autores	**Año**	**Problema a tratar**	**Metodología**	**Consideraciones**	***On-line***
Ergun, Ö. Orlin, J.	**2006**	**Minimizar la tardanza total ponderada para una máquina.**	**Presentan un algoritmo que mejora la complejidad del intercambio de vecindario.**	**Afirman que el resultado mejora la complejidad de las heurísticas de búsqueda dinámica.**	
Cheng, T. Ng, C. Yuan, J. Liu, Z.	**2005**	**Minimizar la tardanza total ponderada en una sola máquina.**	**Tienen dos soluciones. 1. Asumen que las fechas de vencimiento del trabajo son funciones lineales de su tiempo de procesamiento y que la ponderación de la tardanza es proporcional al tiempo este mismo parámetro. 2. Manejan un modelo en los que las fechas de vencimiento de los trabajos tienen la misma holgura.**	**1. Encuentran un algoritmo O(nlogn) para todos los problemas. 2. Desarrollan un algoritmo con complejidad O(n2P) donde P es la suma de todos los tiempos de procesamiento.**	
Angel, E. Bampis, E.	**2005**	**Minimizar la tardanza total ponderada para la que el tiempo de procesamiento depende del momento del release time.**	**Desarrollan una búsqueda local con inicios múltiples como una extensión del trabajo realizado por Congram et al. En el que propone una técnica de búsqueda dinámica.**	**Demuestran que existe superioridad del algoritmo sobre el algoritmo dinámico tras la experimentación en diferentes casos.**	
Grosso, A. Della Croce. F. Tadei, R.	**2004**	**Minimizar la tardanza total ponderada**	**Desarrollan una búsqueda dinámica de vecindario.**	**El algoritmo lo desarrollan gracias los operadores de intercambio por parejas.**	
Sen, T. Sulek, J. Dileepan, P.	**2003**	**ESTADO DEL ARTE DEL PROBLEMA TWT**	**Presentan la historia del problema desde sus inicios que fue estudiada por primera vez en 1968 hasta 1998.**		
Wan, G. Yen, B.	**2002**	**Minimizar la tardanza y prontitud ponderada de los trabajos con distintas ventanas de vencimiento.**	**Usan búsqueda tabú junto con un algoritmo temporizador que determina la finalización de los trabajos en determinada secuencia**	**Realizan experimentos que demuestran que el desempeño del algoritmo es bastante bueno, en especial para casos de gran tamaño.**	

CONSULTA DE ARTÍCULOS CIENTÍFICOS					
Autores	**Año**	**Problema a tratar**	**Metodología**	**Consideraciones**	***On-line***
Mondal, S. Sen, A.	**2001**	**Minimizar la tardanza y prontitud ponderada. Dichos pesos son dependientes del trabajo. Se asume que la fecha de vencimiento es común.**	**Sugieren un algoritmo que usa un espacio gráfico de búsqueda.**	**Supera los inconvenientes del problema resuelto por programación dinámica (fuera de memoria) y *depth-first branch and bound formulation* para casos de gran tamaño**	
Akturk, S. Ozdemir, D.	**2001**	**Minimizar tardanza ponderada con *release time.***	**Proponen una regla de dominancia con la que aseguran que se obtiene un óptimo local.**	**Comparan la regla de dominancia con otras heurísticas para un conjunto de problemas. Además implementan la regla a dos algoritmos de búsqueda local para conducirlos a regiones con mejores soluciones.**	
Yeung, W. Orguz, C. Cheng, T.	**2001**	**Minimización de prontitud-tardanza ponderada, número ponderado de trabajos tempranos y tardíos.**	**Proponen un algoritmo de programación dinámica pseudo-polinómica.**	**Se permite que la ventana de vencimiento pueda ser una variable de decisión o un parámetro.**	
Holsenback, J. Rusell, R. Markland, R. Philipoom, P.	**1999**	**Minimizar la tardanza total ponderada.**	**Desarrollan un método para la modificación de fechas de vencimiento.**	**Afirman que su metodología es mejor que otras heurísticas conocidas, especialmente cuando se puede cumplir con el 40% de los trabajos sin retrasos**	
Liaw, C.	**1999**	**Minimizar la tardanza y prontitud total ponderadas sin considerar tiempos ociosos**	**Límites superiores e inferiores eficientes se desarrollan. Cada uno de ellos se desarrolla mediante un procedimiento diferente.**	**Tras probar problemas con más de 50 trabajos, se determina que el algoritmo se comporta bastante bien.**	
Volgenant, A. Teerhuis, E.	**1999**	**Minimizar la tardanza total ponderada.**	**Se establece una regla de prioridad cuya validez está dada para trabajos vecinos en una solución óptima. La regla se mejora mediante el uso de cuatro heurísticas.**	**Muestran resultados mejorados para problemas de gran tamaño aunque los tiempos de computación son altos.**	

CONSULTA DE ARTÍCULOS CIENTÍFICOS					
Autores	**Año**	**Problema a tratar**	**Metodología**	**Consideraciones**	***On-line***
Tsiushuang C. Xiangtong, Q. Fengshen, T.	**1997**	**Minimizar las prontitudes sujeto a la máxima tardanza de cada trabajo.**	**Derivan relaciones de precedencia entre los trabajos, las cuales, son incorporadas en un procedimiento que disminuye la enumeración de trabajos. Debido a la complejidad del problema, desarrollan una heurística.**	**Ambos algoritmos son examinados bajo un estudio computacional**	
Alidaee, B. Dragan, I.	**1997**	**Minimizar la desviación de los tiempos de finalización de n trabajos para una fecha de vencimiento común.**	**Demuestran que el algoritmo LPT que ordena por tiempo de procesamiento máximo es óptimo sin importar la fecha de vencimiento común. Generan el algoritmo que organiza los trabajos.**	**Asumen que las ponderaciones son proporcionales al tiempo de procesamiento.**	
Srindharan, V. Zhou, Z.	**1996**	***Scheduling* dinámico (*on-line*) de una máquina para minimizar tiempos de entrega tardíos y tempranos**	**Desarrollan una heurística basada en la teoría de decisión con la cual es posible programar la producción con tiempos ociosos de la máquina**	**Comparan resultados con 116 problemas cuyos óptimos son conocidos y se acercan bastante a la respuesta.**	
Andamopoulos G.I., Pappis C.P.	**1996**	**Minimizar el retraso (min Lateness) en la que existen penalidades para la prontitud y demora en entrega ponderadas**	**Se asignan fechas de entregas usando el método SLK (Slack Time) asumiendo diferentes penalidades (dependientes del trabajo, proporcionales al tiempo de procesamiento).**	**Concluyen que el algoritmo es eficiente debido a los pocos esfuerzos que requiere calcular una secuencia óptima.**	
Alidaee, B. Ramakrishnan K.	**1996**	**Minimizar la tardanza ponderada y sin ponderar.**	**Se presentan reglas de dos heurísticas, Costo sobre el tiempo (COVERT) y Urgencia Aparente (AU).**	**Se comparan con 12 reglas de la tardanza total ponderada**	

CONSULTA DE ARTÍCULOS CIENTÍFICOS					
Autores	**Año**	**Problema a tratar**	**Metodología**	**Consideraciones**	***On-line***
Szwarc, W.	**1996**	**Minimizar las prontitudes y tardanzas ponderadas con un modelo fecha de vencimiento común.**	**Identifica 3 tipos de soluciones con complejidades diferentes. Una de ellas, demuestra superioridad sobre las demás.**	**Asume que las penalidades varían de trabajo a trabajo y las proporciones entre los tiempos de procesamiento y penalidades cumplen ciertas condiciones acordadas. Hace pruebas sobre 1500 problemas cuyo número de trabajos es n= 10, 20, 50, 70 y 100.**	
Liman S. Panwalkar, S. Thongmee, S.	**1996**	**Minimizan, prontitud, tardanza y las penalidades ponderadas en un problema con ventana de vencimiento (*due window*) común**	**Proponen un algoritmo que da solución a este problema. Demuestran que en dos casos especiales, los algoritmos SPT y LPT solucionan el mismo problema.**	**Asumen que la ventana de vencimiento es un parámetro del sistema. El problema es estático y determinístico.**	
Forst, F.	**1995**	**Minimizar la tardanza total ponderada y el tiempo esperado de flujo ponderado para una sola máquina y para m-máquinas tipo taller. Los trabajos tienen fecha de vencimiento común.**	**Para ambos problemas se obtiene una secuencia óptima ordenando los trabajos en orden creciente de los tiempos de procesamiento.**	**Los tiempos de procesamiento son variables independientes y dichos trabajos tienen fecha de vencimiento común.**	

Fuente: Presentación propia del autor

Tabla 2: Consulta sobre artículos que traten el problema de *scheduling*

CONSULTA DE ARTÍCULOS CIENTÍFICOS					
Autores	**Año**	**Problema a tratar**	**Metodología**	**Otros**	***On-line***
Tian, J. Fu, R. Yuan, J.	**2011**	**Minimizar el tiempo total de entrega de los trabajos desarrollados en una máquina por lotes.**	**Formulan un algoritmo H para aquellos trabajos cuyos tiempos de entrega son mayores que sus tiempos de procesamiento.**	**Logran una razón competitiva de (√5 + 1)/2 y demuestran que ningún otro algoritmo *on-line* tiene una razón de por lo menos (√3 + 1)/2**	**X**
Liu, M. Chu, C. Xu, Y. Zheng, F.	**2010**	**Minimizar el tiempo de entrega de trabajos de una máquina.**	**Desarrollan un algoritmo llamado EX D LDT (*Extended Delayed Largest Delivery Time*) sin la posibilidad de interrumpir el proceso que se esté desarrollando.**	**Afirman que su algoritmo es óptimo con una razón competitiva dependiente de la proporción entre el tiempo procesamiento y de entregas**	**X**
Sitters, R.	**2010**	**Minimizar el tiempo de terminación total en una sola máquina en la que se permiten interrupciones de los trabajos.**	**Desarrollan un algoritmo basado en WSPT (*Weighted short processing time*) y en interrupción en el caso en que la constante c=wi/pi sea mayor que 1.**	**Prueban que su algoritmo tiene una razón competitiva de 1.57.**	**X**
Zhang, L. Wirth, A.	**2009**	**Minimizar el último tiempo de terminación de todos los trabajos en máquinas paralelas.**	**Desarrollan algoritmos para tres casos especiales: longitud de trabajos iguales, tiempos de procesamiento iguales y tiempos de alistamiento regulares iguales. Para los dos últimos desarrollan heurísticas aplicando *List Scheduling* estableciendo la razón asintótica de cada uno de ellos.**	**Demuestran que sus algoritmos resuelven los tres problemas con tasas de competitividad mejores que la simple aplicación de la heurística LS.**	**X**
Saks, M. Divakaran, S.	**2009**	**Minimizar el tiempo de flujo total de trabajos en una máquina en la que los trabajos están clasificados en dos tipos y ésta requiere tiempos de alistamiento independientes.**	**Implementan un algoritmo basado en la agrupación por lotes de los diferentes tipos de trabajo.**	**Afirman que su algoritmo es O(1)-competitivo, es decir, que siempre está a un factor constante del óptimo.**	**X**
Tian, J. Fu, R. Yuan, J.	**2008**	**Minimizar los tiempos de entrega de todos los trabajos suponiendo que los tiempos de entregas son pequeños.**	**Desarrollan un algoritmo llamado H, basado en el trabajo desarrollado por Hoogeveen et al (2000).**	**El algoritmo que desarrollan demuestra tener un índice √2-competitivo**	**X**

CONSULTA DE ARTÍCULOS CIENTÍFICOS					
Autores	**Año**	**Problema a tratar**	**Metodología**	**Otros**	***On-line***
Nong, Q. Yuan, J. Fu, R. Lin, L. Tian, J.	**2008**	**Minimizar el máximo tiempo de terminación de trabajos (*makespan*) para una máquina capaz de procesar trabajos por lotes de una misma familia de productos.**	**Realiza una investigación sobre problemas similares y desarrolla un algoritmo llamado Hb basado en FBLPT (*Full Batch Longest Processing Time*).**	**Concluyen que su algoritmo es 2-competitivo y que no hay ningún otro algoritmo que pueda disminuir ese valor.**	**X**
Chrétienne, P.	**2008**	**Minimizar el costo de los tiempos ociosos de las máquinas en operación.**	**Introducen una condición llamada "*Earliest Non-Idiling*" que le permite plantear cualquier algoritmo con la restricción de no-tiempos ociosos.**	**Demuestran que la condición propuesta no solo sirve para solucionar este problema sino que genera buenos resultados para los problemas que no cuentan con esta condición.**	**X**
Tian, J. Fu, R. Yuan, J.	**2007**	**Minimizar los tiempos de entrega de todos los trabajos en una máquina que procesa por lotes.**	**Plantean el algoritmo LDT (*Largest delivery time*) que demuestra solucionar el problema *on-line* con razón competitiva.**	**Realizan pruebas de competitividad del algoritmo, hallando su límite superior.**	**X**
Averbakh, I. Xue, Z.	**2007**	**Minimizar el tiempo de flujo de trabajos y el costo de envío de los mismos.**	**Presentan un algoritmo basado en SRPT (*shortest remaining processing time*) que soluciona el problema para un solo cliente al que se le pueden entregar los trabajos por lotes.**	**Afirman que su algoritmo es el que presenta un mejor comportamiento para un solo cliente. También realizan pruebas con un mayor número de clientes suponiendo el mismo costo de envío.**	**X**
Stee, R. Putré, H.	**2005**	**Minimizar el tiempo total de terminación en una máquina con la posibilidad de reinicio de los trabajos no terminados.**	**Desarrollan un algoritmo que evalúa la opción de interrumpir o no el que está en proceso frente a las características del trabajo que acaba de llegar. No tienen en cuenta los que están en la fila de espera para dicha comparación.**	**Afirman que su algoritmo es el primer algoritmo con reinicio de trabajos que muestra ser mejor que aquellos que no permiten interrupciones en el procesamiento de un trabajo.**	**X**

CONSULTA DE ARTÍCULOS CIENTÍFICOS					
Autores	**Año**	**Problema a tratar**	**Metodología**	**Otros**	***On-line***
Montoya-Torres, J.	**2003**	**Minimizar el tiempo total de terminación en una máquina. Es *on-line* dado que, a pesar de conocerse con anticipación los tiempos de llegada de los trabajos, no se conoce el tiempo de procesamiento hasta que son liberados.**	**Proponen un algoritmo basado en SPT (*Shortest processing* time) con un componente adicional que permite fraccionar los trabajos y optar por los que quedan con un menor factor de tiempo de procesamiento. Así, es posible anticiparse a que un trabajo**	**Comprueban que su algoritmo es √3- competitivo.**	**X**
Epstein, L. Stee, R.	**2003**	**Minimizar el tiempo total de terminación de trabajos y tiempo total de flujo (ponderados y sin ponderar) para una máquina con posibilidad de interrupciones.**	**Demuestran los límites inferiores de estos problemas haciendo uso de diferentes algoritmos conocidos para cada uno de ellos como el SPT, entre otros desarrollados en otros artículos similares.**		**X**
Kaminsky, P. Simchi-Levi D.	**2001**	**Minimizar la suma de los tiempos de terminación de todos los trabajos**	**Usan el criterio del SPT (*shortest processing* time) probando su optimalidad usando un análisis asintótico.**	**Los trabajos no permiten interrupciones.**	**X**
Hoogenveen, H. Potts, C. Woeginger, G.	**2000**	**Maximizar el número de trabajos prematuros en una máquina**	**Desarrollan una solución basada en la interrupción del procesamiento de los trabajos aunque estos deben empezar de nuevo cuando les llegue el turno nuevamente.**	**Afirman que no existe un algoritmo con mejor resultado que el que se presenta en el artículo.**	**X**
Fiat, A. Woeginger, G.	**1999**	**Minimizar el tiempo de terminación total en una máquina.**	**Plantean que debe haber una condición mínima para que el algoritmo sea competitivo.**		**X**

Fuente: Presentación propia del autor

3 PLANTEAMIENTO DEL PROBLEMA

Tras la revisión bibliográfica del problema $1||\Sigma w_i T_i$ y sus diferentes variaciones, se pudo constatar que los investigadores se preocupan por tres elementos básicos: representación de la realidad, eficacia de la respuesta y rapidez computacional. A su vez, estos aspectos se ven reflejados en el modelo adoptado para el problema, el método usado para su solución y la comparación con problemas similares.

A la luz de estos aspectos, se ha encontrado la oportunidad de crear un algoritmo que dé solución al problema $1|\textit{on-line}|\Sigma w_i T_i$ debido a que éste, exactamente, no ha sido estudiado dentro de la literatura consultada y se cree que es posible encontrar una solución que contenga las características descritas en el párrafo anterior. Lo anterior, se fundamenta en el hecho de que la característica *on-line* aparece en la literatura como una más de las posibles restricciones del trabajo (YVES, 2010). Incluso, se pueden encontrar investigaciones sobre *scheduling on-line* que toman otras funciones objetivo como lo hacen Fiat y Woeginger (1999) para el problema de minimización de los tiempos de terminación de los trabajos en una máquina.

Adicionalmente, se plantea que la solución esté basada en el uso de la metaheurística GRASP con parámetros determinísticos, basados en el conocimiento de los parámetros del trabajo en el momento que llega un nuevo trabajo a la máquina y no antes. Dicho algoritmo debe estar implementado bajo un entorno computacional que permita observar respuestas bajo diferentes instancias y que, a su vez, éstas puedan ser comparadas con algoritmos *off-line* bajo la suposición de que estos últimos tienen toda la información de los trabajos desde el inicio.

En consecuencia se plantea la siguiente pregunta de investigación ¿Bajo qué condiciones se debe plantear un algoritmo basado en la metaheurística GRASP que sea capaz de solucionar el problema de *scheduling* $1|\textit{on-line}|\Sigma wiTi$ de tal manera que los resultados obtenidos demuestren la búsqueda de un valor mínimo de la función objetivo?

4 JUSTIFICACIÓN DEL PROBLEMA

La presente investigación se propone debido a la importancia que se ha encontrado en buscar una solución al problema 1|*on-line*|$\Sigma w_i T_i$ mediante el uso de la metaheurística GRASP ya que no se tiene evidencia de que alguien lo haya estudiado, a que los resultados que se puedan obtener de dicha investigación puede significar un avance en tratamientos de la planeación de la producción en ambientes industriales en tiempo real y a que, como se ha demostrado, estos planteamientos pueden ser llevados a ambientes reales de producción.

En primer lugar, la revisión bibliográfica desarrollada hasta el momento indica que el problema de *scheduling* cuyo objetivo es minimizar la sumatoria de tardanzas ponderadas tiene vigencia en la actualidad. Muestra de ello es que a pesar de que se ha estado estudiando por más de 40 años, existen artículos recientes interesados en el tema tal como se aprecia en la Tabla 1, en la que se pueden encontrar artículos publicados en el año 2011.

Lo anterior es una consecuencia de que el problema en cuestión es de tipo *NP-Hard*, lo que significa que no existe una metodología eficiente que permita llegar a la mejor respuesta en un tiempo de cálculo razonable debido a su complejidad computacional (YVES, 2010), en especial cuando existe un número de trabajos relativamente grande a organizar (GROSSO, DELLA CROCE, & TADEI, 2004). Por lo anterior, se abre la puerta a que los investigadores traten de llegar a soluciones cada vez mejores (entiéndase mínimas en el presente proyecto) por diferentes caminos. Por otra parte, la gran cantidad de restricciones que un problema de este tipo puede tener, conlleva a que muy pocos artículos traten las mismas características exactas (β en notación de Graham) y, por lo tanto, el horizonte de investigación se muestra bastante amplio, ya sea porque se desea mejorar lo que alguien más ha desarrollado o porque, según la revisión del presente proyecto, existe campo inexplorado.

En segundo lugar, se ha optado por el problema 1|*on-line*|ΣwiTi ya que se tiene en cuenta la presencia de un solo recurso que debe procesar todos los trabajos tal como una impresora, un baño de uso común (CABALLERO-VILLALOBOS, 2010) o una máquina que represente el cuello de botella de un proceso; además, pueden representar la base para solución de problemas más complejos con un mayor número de recursos. Por otra parte, se decide *on-line* debido a que en condiciones reales se hace muy difícil conocer con anticipación la información completa de todos los trabajos que deben ser procesados; es decir, “los trabajos llegan a lo largo del tiempo, y quien toma las decisiones no tiene ningún conocimiento del futuro del sistema” (HOOGEVEEN, POTTS, & WOEGINGER, 1999). Esto contrasta con el hecho de que los problemas estáticos deben tener todos los datos al inicio del problema o que, en caso de cambiar alguno de los parámetros iniciales del problema (la llegada de un nuevo trabajo por ejemplo), implica correr nuevamente el algoritmo y esto implica el aumento del tiempo de computación. Por último, se ha escogido la función TWT debido a que esta función sirve como indicador de servicio al cliente (VEGA-MEJÍA & CABALLERO-VILLALOBOS, 2010) y en ambientes industriales

esta medida de desempeño es bastante apreciada (CABALLERO-VILLALOBOS, Conceptos Preliminares II, 2010).

Tal como se expuso en los antecedentes del problema, existe evidencia de que la solución de un problema de *scheduling* tiene gran aplicabilidad a los procesos productivos reales (Ver Antecedentes). En tal caso, es posible llevar a cabo la experimentación de resultados con datos de situaciones reales e igualmente los resultados pueden ser implementados en esos mismos ambientes. Adicional a esto, existen evidencias de la importancia económica ya que un posible criterio de la penalidad puede ser el costo incurrido por la no entrega del trabajo a tiempo. También, una herramienta de este tipo permite evaluar la posibilidad de llevar a cabo una mejor planeación de la producción y, por lo tanto, analizar alternativas como la tercerización así como lo proponen Lee y Sung (2008) dentro de su modelo.

Por otra parte, se ha optado por la metaheurística GRASP dado que se ha demostrado que alcanza resultados muy cercanos a los mejores obtenidos para este problema de *scheduling off-line* tal como lo demuestran Vega-Mejía y Caballero-Villalobos (2010), y Caballero-Villalobos y Alvarado-Valencia (2010) para este mismo problema. Además, los algoritmos de búsqueda son muy usados entre los investigadores por lo que se tiene una sólida base de que las metaheurísticas de este tipo son útiles para la solución del problema $1||\Sigma w_i T_i$. En consecuencia, dado que la metodología GRASP consiste en un algoritmo de búsqueda en su segunda fase (Ver Marco Teórico), soporta aún más la idea de que esta metodología es aplicable a este problema.

Finalmente, los problemas de este tipo son susceptibles de mejoras por lo que el trabajo propuesto puede servir como punto de partida para posteriores investigaciones bajo cualquier resultado dado que, por las características del método de solución, se busca encontrar una buena respuesta que satisfaga las necesidades de minimizar la función objetivo pero no es garantía de que esa sea la mejor solución al problema.

5 MARCO TEÓRICO

Para Pinedo (2008) "*Scheduling* es un proceso de toma de decisiones que es usado regularmente en industrias de manufactura y servicios. Se encarga de la asignación de recursos a tareas en ciertos periodos de tiempo y cuya meta es optimizar uno a más objetivos". Por su parte, Wiers (1997) lo define de una forma más general así: "Programar se trata de hacer lo máximo en un periodo limitado de tiempo".

Para el caso industrial, un problema de programación de la producción consiste en organizar n trabajos J_i (i = 1,…, n) en m máquinas M_j (j=1,…,m) los cuales pueden ser graficados en un diagrama de Gantt (BRUCKER, 2007).

Todo problema de programación de la producción incluye tres características fundamentales: entorno de la maquinaria, características del trabajo y criterio de optimización. Debido a esta problemática, en la literatura se usa la notación de Graham α|β|γ (Citado por BRUCKER, 2007) donde cada término hace referencia a las diferentes características como se muestra a continuación bajo la descripción de Pinedo (2008) y Brucker (2007).

5.1 Entorno de la maquinaria (α)

Se representa por una cadena de caracteres $\alpha=\alpha_1\alpha_2$ donde α_1 puede tomar valores como °, P, Q, R, G, X, entre otros, significando la configuración en la que se encuentran las máquinas. Por ejemplo α_1= P significa que se trata de un problema de máquinas paralelas idénticas. En el caso que α_1=°, se dice que $\alpha= \alpha_2$ que demuestra el número de máquinas que se están tratando. Por ejemplo, para el presente proyecto, α=1 por tratarse de un problema de una sola máquina.

5.2 Características del trabajo (β)

Para está característica se tienen seis posibles elementos cuyos valores son los que se muestran en la Tabla 3. Vale la pena aclarar que la notación expuesta por Brucker (2007) no concuerda con muchos de los artículos consultados, en los cuales el parámetro β toma muchos otros valores y connotaciones. Sin embargo, dentro de la bibliografía consultada, esta fue la explicación más completa del parámetro:

Tabla 3: Resumen de los posibles valores de las características del trabajo β

β	Valor en caso de presentarse	Significado
β_1	pmtn	Posibilidad de parar un trabajo para atender otro con mayor prioridad.
β_2	prec	Los trabajos tienen relación de precedencia.
β_3	r_i	Especificación de tiempo de llegada del trabajo i.
β_4	p_{ij}, p_i	Se define el tiempo de procesamiento del trabajo i.
β_5	d_i	Se definen tiempos de entrega
β_6	s-batch, p-batch	Se definen trabajos por lotes con alistamiento

Fuente: Brucker (2007)

Tiempo de procesamiento (Processing time)(p_{ij}): Tiempo que permanece el trabajo i en la máquina j. Se omite j si el trabajo consiste en una sola operación.

Release Time **(r_i):** Momento en el que el trabajo i llega al sistema

Fecha de entrega (Due date) (d_i): Momento en el que se promete el trabajo i al cliente. Puede darse con penalidad cuando se excede ese tiempo.

Peso relativo del trabajo (weight)(w_i): Es el factor de prioridad del trabajo i.

Tiempo de inicio (Start time) (s_{ij}): Es el momento en el que se comienza la ejecución del trabajo i en la máquina j. Nuevamente, si el trabajo sólo tiene una operación, se denota mediante s_i.

Tiempo de finalización (Completion time)(C_{ij}): Momento en el que el trabajo i es completado en la máquina j.

Para Pinedo, estas restricciones pueden ser las siguientes:

Preemption **(prmp):** Esta característica implica que no es necesario que se mantenga un trabajo en la máquina una vez inicia el proceso. El programador tiene permitido interrumpir el proceso e ingresar uno diferente.

Restricciones de precedencia (prec): Aparece en los problemas de una máquina o máquinas paralelas en las que se requiere que algún trabajo sea procesado antes que otro obligatoriamente.

Familias de trabajos (fmls): Se trata de aquellos casos en las que un trabajo n pertenece a una familia particular F. Cuando se están procesando, los trabajos de la misma familia no requieren tiempos de preparación, mientras que cuando se cambia de familia sí.

Paradas (brkdwn): Una máquina puede no estar disponible en todo momento.

Restricciones de Selectividad de máquina (Mj): Los trabajos deben ser procesados en máquinas específicas.

5.3 Medidas de desempeño (γ)

Existen varias medidas de desempeño y cada una ofrece un criterio diferente para resolver el problema y debe corresponder a un problema de maximización o minimización según lo requiera la función. Las más nombradas son las siguientes:

- $F_i = C_i – r_i$: Tiempo de flujo del trabajo i.
- $L_i = máx\{0,C_i - d_i\}$: retraso del trabajo i.
- $E_i = máx\{0, d_i – C_i\}$: Adelanto del trabajo i
- $T_i = máx\{0,C_i - d_i\}$: Tardanza del trabajo i.
- $D_i = máx|C_i – d_i|$: Desviación absoluta.
- $U_i = \{0$ si $C_i \leq d_i$ o 1 en otro caso: Penalidad unitaria.

A partir de cada una de las anteriores definiciones, pueden asociarse un gran número de problemas dentro de los que sobresalen los presentados a continuación:

- $C_{máx} = máx\{C_i\}$: llamado *makespan* en los artículos y traducido como lapso o tiempo máximo de terminación de todos los trabajos.
- ΣC_i = Total del tiempo de flujo
- $\Sigma w_i Ci$ = Total del tiempo de flujo ponderado
- $L_{máx} = máx\{L_i\}$: retraso máximo
- ΣT_i = Tardanza total
- $\Sigma w_i T_i$ = Tardanza total ponderada

5.4 Problema 1||ΣwiTi

En palabras de Brucker (2007) este problema consiste en “programar i=1, …, n trabajos en una máquina de tal manera que $\Sigma w_i T_i$ es minimizada, donde $T_i= max\{0,C_i-d_i\}$ es la tardanza del trabajo j.” Además, como ya se ha mencionado, la variación del problema depende de la característica del trabajo en los que se pueden encontrar muchas variaciones (el presente proyecto muestra una de ellas).

5.5 Complejidad del problema

Cada vez que se resuelve un problema de *scheduling* (o en general un problema de optimización combinatorio) lo primero que se hace es tratar de desarrollar un algoritmo que lo solucione de manera eficiente. Si dicho algoritmo no se puede encontrar, sería de gran utilidad probar que el problema es de tipo *NP-Hard*, lo que implica que lo más probable es que no existe un algoritmo eficiente (YVES, 2010).

Un problema computacional puede ser visto como una función h la cual toma valores en un determinado rango para una entrada x. De esta manera, la eficiencia de un algoritmo puede ser medida por el tiempo de procesamiento, como por ejemplo, el número de pasos que requiere el algoritmo para ciertas entradas. Por lo tanto, es de suponerse que ante una entrada más grande (definida por la magnitud de la entrada |x|), debe haber una función que defina proporcionalidad con la salida. El tamaño de una entrada para un programa computacional es definido usualmente por la longitud de la codificación binaria de la entrada. Para el caso de una entrada a, la magnitud de codificación binaria es entonces $\log_2 a$. Por otra parte, una codificación unitaria es aquella en la que un número a es representado por igual número de bits.

Aunque en la mayoría de los casos es muy difícil saber el promedio del tiempo de procesamiento para una entrada de tamaño n, comúnmente se toma el peor caso del algoritmo estudiado. Para este propósito, la función T(n) se define como el límite superior de los tiempos de corrida del algoritmo para una entrada x con una magnitud |x|=n. Igualmente, como es difícil describir completamente el comportamiento de n, sólo se considera la tasa de mayor crecimiento de la función T(n). Se dice entonces que una función T(n) es O(g(n)) si se cumple que para las constantes $c, n_0 \in N$ tal que $T(n) \leq cg(n)$ para toda $n \geq n_0$. Por ejemplo, la función $T_1(n)=37n^3 + 4n^2 + n$ es $O(n^3)$ y la función $T_2(n) = 2^n + n^{100} + 4$ es $O(2^n)$.

Si el tiempo de corrida de un algoritmo es limitado por una función polinomial, es decir, por $O(n^k)$ con $k \in N$, se llama un algoritmo de tiempo polinomial (*polynomial-time algorithm*). Si el tiempo de corrida es limitado por una función polinomial con una entrada de codificación unitaria, se dice que es pseudo-polinomial (*pseudo-polynomial algorithm*).

Dado que un problema con tiempo de corrida polinomial como $O(n^3)$ es mucho más veloz que un problema $O(2^n)$ (la relación es que mientras un problema de tiempo polinomial puede resolver un problema con n=1000 en menos de un segundo, uno de tiempo exponencial tardaría 374 siglos en resolver el problema para uno con n=70),los problemas se clasifican en fáciles o difíciles si se pueden resolver en tiempos polinomiales (P) o no polinomiales (NP) respectivamente.

5.6 Problemas *On-line*

Muchos de los problemas de *scheduling* que se plantean en la práctica son *on-line*. Bajo esa situación, las decisiones de programación deben ser tomadas sin la información completa acerca de la situación. Esta falta de información se debe a varias fuentes: (1) Los trabajos llegan uno tras otro como una lista o incluso como una cadena a lo largo del tiempo. Las decisiones se deben tomar siempre sin el conocimiento de los futuros trabajos. (2) Los tiempos de procesamiento de los trabajos son desconocidos al inicio y durante el tiempo de corrida. Estos tiempos se conocen en el momento en el que son terminados. (3) Las fallas de la máquina y el mantenimiento de las mismas son desconocidos.

Como es de esperarse, la mejor solución computacional es, en general, muy difícil de alcanzar y se debe valer de aproximaciones como recurso para su solución. Hoy en día, un algoritmo A es comparado con un algoritmo *off-line* óptimo OPT que conoce toda la información previa al inicio del problema. Así, un algoritmo *on-line* se llama *c-competitive* si existe una constante b tal que para todos los problemas con entradas I, se mantiene la siguiente desigualdad $A \leq c\,(OPT(I)) + b$ (YVES, 2010).

5.7 Metaheurísticas

En palabras de Glover y Kochenberger (2003) "las metaheuríticas, en su definición original, son soluciones a métodos que orquestan una interacción entre procedimientos de mejoramiento local y estrategias de alto nivel para crear un proceso capaz de escapar de un óptimo local y desempeñarse como un buscador robusto de un espacio de soluciones". En otras palabras, las metaheurísticas son métodos de solución estructurados aplicables a problemas cuyo método de solución analítico resulta complejo o muy extenso. Para ello, parten de soluciones factibles que direccionan la atención hacia soluciones óptimas hasta que la respuesta obtenida sea satisfactoria aún sin asegurar que sea la mejor que se puede obtener.

5.7.1 Metaheurística GRASP

Una breve definición de esta metaheurística la realizan Resende y González (2003) como sigue:

> "Un procedimiento de búsqueda miope aleatorizado (GRASP por sus siglas en inglés) es una metaheurística para contar soluciones aproximadas (i.e. sub-óptimas de buena calidad, pero no necesariamente óptimas) a problemas de optimización combinatoria. Se basa en la premisa de que soluciones iniciales diversas y de buena calidad juegan un papel importante en el éxito de métodos locales de búsqueda.
>
> Un GRASP es un método multi-arranque, en el cual cada iteración GRASP consiste en la construcción de una solución miope aleatorizada seguida de una búsqueda local usando la solución construida como punto inicial de búsqueda local. Este procedimiento se repite varias veces y la mejor solución encontrada sobre las iteraciones GRASP se devuelve como la solución aproximada." (RESENDE & GONZÁLEZ, 2003)

GRASP consiste en dos fases: una inicial de construcción y luego una de búsqueda local. En el primer paso, lo que se busca es construir una solución. Este proceso se debe repetir hasta conseguir una solución factible para la solución. Luego, se investiga en los alrededores de la respuesta cuál es el mínimo local. Finalmente se escoge el valor mínimo entre todas las soluciones locales encontradas recordando que esta metaheurística debe ser corrida varias veces para tener un número suficiente de soluciones para comparar (RESSENDE & RIBEIRO, 2010).

A continuación se presentan el pseudocódigo de la metaheurística:

Figura 1: Pseudo-Código de la metaheurística GRASP

```
procedure GRASP(Max_Iterations, Seed)
1     Read_Input();
2     for k = 1,...,Max_Iterations do
3          Solution ← Greedy_Randomized_Construction(Seed);
4          if Solution is not feasible then
5               Solution ← Repair(Solution);
6          end;
7          Solution ← Local_Search(Solution);
8          Update_Solution(Solution, Best_Solution);
9     end;
10    return Best_Solution;
end GRASP.
```

Fuente: Resende y González (2003)

5.8 Parámetros definidos para las instancias

Una revisión preliminar de las instancias presentes en OR-Library para el problema TWT, revelan que las instancias generadas dependen de la generación aleatoria de los valores de w_i y p_i para las cuales se usan distribuciones continuas uniformes.

Adicional a esto, la fecha de entrega se genera mediante una distribución de este mismo tipo pero, a diferencia de los anteriores parámetros en los que el rango de la distribución es fijo, aquí se depende de dos factores definidos como *tightness factor* (τ) y *due date range* (RDD). los cuales toman diferentes valores entre 0 y 1.

En consecuencia, se tiene que *tightness factor* es un indicador que mide el porcentaje de trabajos que son completados luego de su fecha de entrega. Por su parte, *due date range* (RDD) se define como la razón de cambio del rango de fechas límites con respecto al valor estimado del tiempo total de terminación de los trabajos ($\hat{C}_{max}$).

A continuación se muestra la manera de calcular estos indicadores:

Ecuación 1: Resumen de los parámetros *tightness factor* y *due date range*

$$\tau = 1 - \frac{\bar{d}}{\hat{C}_{max}}$$

Donde $\bar{d}$ corresponde al valor promedio de las fechas de entrega y $\hat{C}_{max}$ corresponde a la siguiente formulación:

$$\hat{C}_{max} = \sum_{j=1}^{n} p_j + n\gamma\bar{s}$$

Donde γ es un parámetro definido empíricamente y debe ajustarse a γ ≤ 1 y $\bar{s}$ significa el tiempo promedio de los tiempos de alistamiento. Finalmente, el *due date range* se define así:

$$RDD = \frac{(d_{max} - d_{min})}{\bar{C}_{max}}$$

Fuente: Armentano y Araujo (2006)

6 OBJETIVOS

6.1 Objetivo General

Diseñar un algoritmo que dé solución al problema de programación de la producción denotado como $1|on\text{-}line|\sum w_i T_i$ con parámetros determinísticos mediante el uso de la metaheurística GRASP con el fin de determinar la secuencia en la que han de ser organizados un número finito de trabajos cuyas características sólo se conocen en el momento en el que llegan al puesto de trabajo

6.2 Objetivos Específicos

Desarrollar un procedimiento computacional basado en la metaheurística GRASP para el problema $1|on\text{-}line|\sum w_i T_i$, con el propósito de organizar un número finito de trabajos de tal manera que su tardanza total ponderada sea minimizada bajo condiciones *on-line*.

Validar el algoritmo formulado con el uso de diferentes instancias publicadas para el caso *off-line* del mismo problema, con el propósito de establecer la capacidad del programa para generar respuestas factibles y medir la diferencia porcentual entre ambos tipos de soluciones basados en los resultados de estudios previos.

Determinar estadísticamente, mediante el diseño de un experimento cuyos factores consisten en el parámetro α de GRASP y los parámetros propios de cada instancia probable (Número de trabajos, *tightness factor* -τ- y *due date range* –RDD-), con cuál combinación de parámetros se obtienen los mejores resultados.

7 DESARROLLO DEL PROBLEMA

El problema que se ha querido tratar en los siguientes capítulos se refiere a aquel en el que los trabajos llegan en instantes diferentes de tiempo y cuyo objetivo consiste en minimizar la tardanza total ponderada. Además, se considera el supuesto que los trabajos se agrupan en lotes mientras la máquina esté ocupada y la conformación de los mismos no da lugar a que la máquina quede inactiva en ningún instante de tiempo, en otras palabras, los trabajos que llegan a la estación mientras ésta se encuentra ocupada se agrupan con los demás trabajos en la misma situación y, una vez liberado el recurso, todos los elementos del lote toman ese instante como su fecha de liberación (*release time*). Dicha perspectiva del problema *on-line* es llamada paso-a-paso (*step-by-step*), descrita por Fiat y Woenginger (1999), en la que la solución del problema se concentra en la información disponible al momento de correr el método de solución y no toma en cuenta los trabajos que llegan después de iniciado el proceso sino hasta que éste ha culimnado. En consecuencia, todo lo anterior se logra mediante un procedimiento lógico que permita la generación de la situación y la programación de los trabajos en el menor tiempo posible.

El algoritmo se desarrolla en dos etapas, una de recepción de trabajos y otra de secuenciación. En la primera, se hace una reorganización de las instancias disponibles para el problema *off-line* mediante la cual cada trabajo se asigna a un lote de trabajo de forma aleatoria de tal manera que cada grupo llegue en un instante diferente. La segunda etapa, consiste en el desarrollo de la metaheurística GRASP que, a su vez, consta de dos fases (construcción y búsqueda) con las que se lleva a cabo la organización de los trabajos tratando de minimizar la tardanza total ponderada en el menor tiempo posible.

En consecuencia, el producto de este proyecto es ofrecerle al usuario una función cuyos datos de entrada sean los trabajos (con sus correspondientes parámetros, incluido el lote al que pertenece), el número de iteraciones (IT) que se desea correr el algoritmo y el parámetro miope (α) propio de la metaheurística. Como resultado, ésta le entregará una secuencia determinada que respeta el orden de llegada de los lotes y procurando administrar de la mejor manera el objetivo buscado en un tiempo de procesamiento pequeño, ojalá imperceptible.

Todo lo anterior, se ha desarrollado en el programa MATLAB® R2011b de la empresa *The MathWorks Inc*, el cual, cuenta con un lenguaje de programación propio que incorpora muchas características de C. En muchos aspectos es considerado un lenguaje de más alto nivel que otros similares como Pascal, Fortran o C, en el sentido de que evita pérdida de tiempo en formalismos y sintaxis (PATRAP, 2002). Por lo tanto, se considera que además de la facilidad de manejo, se está tratando un problema matemático en una plataforma diseñada para tal fin. A continuación se muestra en detalle cada una de las etapas descritas:

7.1 Llegada de los trabajos en lotes

Como ya se ha mencionado, la bibliografía consultada referente al problema de tardanza ponderada se analiza bajo características *off-line* en sus diferentes variaciones y no se ha encontrado una investigación similar hasta el momento. Por esta misma razón, las instancias disponibles para el problema no contemplan que éstos lleguen en un momento diferente a cero y, por lo tanto, en principio no podrían ser usadas para el desarrollo *on-line*.

En vista de este hecho, se ha planteado modificar las instancias disponibles del problema *off-line* de tal forma que se simulen lotes de llegada en instantes diferentes y así tener un punto de comparación con resultados existentes. Para ello, se desarrolló una función en MATLAB® que permite seleccionar aleatoriamente tamaños de lotes y, análogamente, los trabajos que contiene cada uno de ellos.

El algoritmo planteado, toma la instancia a modificar y de forma aleatoria escoge un nuevo orden en el que se presentan los trabajos. Luego, con ayuda de otra variable aleatoria, selecciona el número de trabajos que contiene cada lote y les asigna un valor hasta que todos los trabajos pertenezcan a un grupo. Así, el resultado de este proceso es entregar un archivo de la librería OR-Library a la que se agregan un parámetro nuevo definido como lote al que pertenece cada trabajo.

Para las secciones siguientes se debe entender que el tiempo de llegada del lote a la máquina puede ocurrir desde un instante siguiente al que se empieza a procesar el lote anterior o poco antes de que éste sea terminado en su totalidad. En todo caso, el tiempo de inicio del último lote conformado será la suma de los tiempos de procesamiento de los trabajos de los lotes anteriores.

Con esto se concluye, que a partir de la distribución por lotes se puede contar con un punto de partida para el desarrollo *on-line* desde la perspectiva definida. La Figura 2 presenta el pseudocódigo de esta función.

Figura 2: Pseudocódigo usado de la división por lotes de los datos originales

```
[Instancia_L] = Lotes (Instancia)
N=Instancia
Instancia_L=[]
Mientras Tamaño(N)>0
        k=Aleatorio_Entre (1,Tamaño (N))
        Instancia_L=[Instancia_L;N(k,:)]
        Eliminar N(k,:)
Fin Mientras
L = Instancia_L
T2=Tamaño(L)/2
%T2 Busca asegurar que haya al menos dos lotes en los datos modificados
Borrar Instancia_L
j=1
Mientras Tamaño(L)>0
        T=Tamaño(L)
        R=Aleatorio entre (1, min(T,T2))
        v=Vector Columna de j's tamaño R
        Instacia_L = [Instancia_L; L, v]
        Borrar L(1:R,:)
        J=j+1
Fin Mientras
```

Fuente: Presentación propia del autor

De lo anterior se quiere rescatar tres puntos importantes. En primer lugar, la distribución en lotes tiene como objetivo asegurar que el algoritmo de programación basado en GRASP sea aplicable cada vez que el lote tenga más de un trabajo, de lo contrario, estos se programarían bajo el criterio de orden de llegada (esto ocurre cuando los lotes son de tamaño 1).

En segundo lugar, se ha definido la variable T2 como aquella que asegura la existencia de al menos dos lotes de trabajos en la nueva instancia. Esto, le permite al algoritmo trabajar de forma *on-line*, bajo los supuestos descritos y procurando por la conformación de grupos con suficientes trabajos a fin de ofrecerle un reto a la segunda etapa del procedimiento. Para ilustrar, se tiene que el mínimo número de lotes permitido es 2, en cuyo caso, cada uno tendría la mitad de los trabajos de la instancia inicial.

Además, esta forma de presentación, en la que se mantienen los parámetros propios de cada trabajo (p, r, d, w, T), contempla un escenario que no ocurre en el problema *off-line* debido a que aquí algún trabajo puede llegar a la máquina con la fecha de entrega vencida, en cuyo caso, la fase constructiva de GRASP debe estar en función de darle prioridad.

Finalmente, se debe tener en cuenta que esta etapa del proyecto sólo se desarrolló con fines de evaluación de las instancias y que, en la práctica, esta etapa del proyecto no debería estar presente. Cabe anotar que en la realidad, cada trabajo presenta su propio tiempo de llegada al sistema y, desde el punto de vista de este proyecto, éste se agrupará con otros que lleguen a la máquina mientras ésta se encuentre ocupada. Una vez liberada de carga, los trabajos que han ido llegando conformarán un lote en ese momento. En

consecuencia, en el uso práctico, las instancias del problema no deben sufrir ninguna reorganización y sólo requieren de la aplicación de la segunda etapa del algoritmo propuesto.

7.2 Metaheurística

Para el problema en mención, la metaheurística GRASP se desarrolla en sus fases de construcción y búsqueda local lote a lote, conservando el orden de llegada de los mismos. En la primera fase, se llega a una solución básica mediante la selección de los trabajos bajo la función WSPT (Tiempo de procesamiento más corto) estudiada por Niño y Caballero (2011). Posteriormente, en la búsqueda local, se examina la vecindad de la solución básica mediante el uso de la metodología *2-optimal* usada por Vega y Caballero (2010) para el problema *off-line*, en la que se hacen permutaciones entre dos trabajos de la solución básica. Finalmente, el valor de la función objetivo que se obtiene luego de las fases, es almacenada en una variable solución y ésta, a su vez, es remplazada cada vez que aparece una mejor programación en una iteración posterior, entendida como de menor tiempo de procesamiento. El pseudocódigo planteado se muestra a continuación:

Figura 3: Pseudocódigo de la metaheurística GRASP

```
[Z, SF, FW1] = SGRASP (alfa, it, Instancia_L)
vi=it
Mientras vi>0
        I =Instancia_L
        m = 1
        T = máx(I(:, Lote))
        Mientras m <= T
                Mientras I (1, Lote) = m
                        C1 = [C1; Instancia]
                        Borrar Instancia (1, Lote)
                Fin Mientras
                [C2, S_Optimo] = Fase Constructiva (C1)
                Si Tamaño(S_Optimo)>1
                        [SL] = Búsqueda Local ( S_Optimo)
                Fin Si
                        m=m+1
        Fin Mientras
        FW1= suma (S (Tardanza Ponderada))
        Si vi = it
                SF=S;
        Fin Si
        FW1=[FW1;FW2]
        Si FW2<min(FW1)
                SF = SL
        Fin Si
        vi=vi-1
Fin Mientras
Z=min(FW1)
```

Fuente: Presentación propia del autor

7.2.1 Fase constructiva

Esta fase, que se aplica a cada lote por separado, tiene la consigna de darle a cada trabajo una posición dentro de la de la solución inicial, basada en un criterio determinado por la función de costo, el parámetro alfa y un factor aleatorio. Para esto, se siguen los siguientes pasos de forma iterativa:

i. A cada trabajo se le calcula la función de costo WSPT determinada como la más apropiada en la mayoría de los casos para el problema de tardanza total ponderada cuando se le aplica la metaheurística GRASP según los contrastes con otras funciones estudiadas por Niño y Caballero (2011). La función se define como:

Ecuación 2: Función de costo para la fase constructiva

$$f_c(j) = \begin{cases} \dfrac{t + p_j - d_j}{w_j} & si\ el\ trabajo\ j\ está\ atrasado \\ \dfrac{p_j}{w_j} & si\ el\ trabajo\ j\ no\ está\ atrasado \end{cases}$$

Fuente: Niño y Caballero (2011)

ii. Se calcula la lista de trabajos seleccionables (RCL) que contiene todos los trabajos que cumplan con el siguiente criterio:

$$f_c(j) \leq mínimo\ [f_c(j)] + \ \alpha * (máximo[f_c(j)] - mínimo\ [f_c(j)])$$

iii. Aleatoriamente, se escoge uno de los trabajos entre la RCL y el tiempo de procesamiento del trabajo se suma al valor del tiempo total de procesamiento del problema.

iv. Si aún quedan trabajos por programar en el lote, se regresa al punto 1. De lo contrario, se culmina con la fase constructiva.

La anterior secuencia se realiza para cada uno de los lotes una vez que se haya culminado la búsqueda local del lote anterior y hasta que se ha terminado la programación de todos los trabajos. Además, se hace notar que la repetición de los cuatro pasos descritos obliga a que cada vez las funciones de costo y el criterio de selección sean evaluados nuevamente tanto por el tiempo transcurrido como por la eliminación de un nuevo trabajo. Esto obliga a que todos los trabajos pasen al menos en una ocasión por la lista de seleccionables y, por supuesto, sean escogidos para conformar la solución básica en algún momento. El pseudocódigo de esta fase se presenta a continuación:

Figura 4: Pseudocódigo de la fase Constructiva

```
[S_Optimo] = Fase Constructiva (C1)
Tiempo =0
Mientras Tamaño(C1)>0
        RCL=[]
        J=1
        Mientras j<=Tamaño(C1)
                C1(j, FCosto)=WSPT(Tiempo)
                J=j+1
        Fin Mientras
        Mientras j<=Tamaño (C1)
                Si C1(j , Fcosto) <= min(C2(j, Fcosto)) + alfa * (max(C2(j,Fcosto)) –
                min(C2(j,Fcosto))
                        RCL = [RCL; C1(j,: )]
                Fin Si
                j = j+1
        Fin Mientras
        R= Aleatorio Entre (1,Tamaño(RCL))
        Tiempo=Tiempo + RCL (R, Tprocesamiento))
        Tardanza = Tiempo – RCL (R, Tlímite)
        S_Optimo=[S_Optimo; RCL(R, : )]
        Eliminar Fila CL = RCL
        Borrar RCL
Fin Mientras
```

Fuente: Presentación propia del autor

7.2.2 Fase de búsqueda Local

Una vez que se ha salido de la fase constructiva, el algoritmo ejecuta un nuevo proceso iterativo llamado *2-optimal* mediante búsqueda intensiva para cada lote. El procedimiento consiste en seleccionar el primer elemento de la solución básica y permutarlo con cada uno de los trabajos que lo siguen. Cada vez que esto ocurre, se recalcula la tardanza sufrida por los trabajos permutados y los que se encuentran en el intermedio de éstos (los demás no sufren ningún cambio).

Una vez que se hace una permutación efectiva, entendida como la disminución del valor de la función objetivo, se toma el nuevo orden como respuesta y se realizan las mediciones correspondientes. De la misma manera, se selecciona el siguiente trabajo de la solución básica que no haya sido permutado y se repite el procedimiento con los trabajos siguientes. Todo este proceso se lleva a cabo hasta que todos los trabajos del lote hayan sido intercambiados en al menos una ocasión.

Con esta búsqueda, se pretende mejorar la respuesta básica evitando hacer todas las permutaciones posibles en las que, con seguridad, se lograría el valor óptimo de la función para el lote. Sin embargo, a cambio de esto, el tiempo computacional puede tomar un valor considerable si se tiene en cuenta que, por ejemplo, un lote de 10 trabajos tiene más de 3 millones de combinaciones.

Figura 5: Pseudocódigo de la fase Búsqueda local

```
[SL]=Búsqueda Local (S_Optimo)
I=1
J=1
FW=0
FW1=Suma(S_Optimo(Tardanza Ponderada))
SL = S_Optimo
Mientras i<Tamaño(S_Optimo)
        J=j+1
        Mientras j<Tamaño (S_Optimo)
                S= Cambiar SL(i,: ) con SL(j,: )
                Para m=i hasta j
                        Calcular Tardanza Ponderada SL(m)
                Fin Para
                FW=Suma Tardanzas Ponderadas (SL)
                Si FW<=FW1
                        SL=S
                        FW1 = FW
                        I=j
                        J=Tamaño(S_optimo)
                Fin Si
                J=j+1
        Fin Mientras
        I=i+1
Fin Mientras
```

Fuente: Presentación propia del autor

Resumiendo, se ha mostrado cómo se diseñó el algoritmo de acuerdo a la modificación de los datos de entrada, convirtiendo las instancias disponibles para el problema *off-line* en trabajos organizados por lotes para ser analizados desde la perspectiva *on-line*.

También, se ha hecho una descripción de cómo se planteó la metaheurística GRASP, haciendo mención en que sus fases son aplicadas a cada lote por separado y la respuesta del problema será la unión de la secuencia de todos los lotes y sus respectivos valores. Finalmente, las repeticiones de este último procedimiento hace que en cada iteración exista la posibilidad de encontrar una mejor respuesta al problema y, aprovechando la facultad de procesamiento de funciones matemáticas de MATLAB®, se pueda hacer este procedimiento un número considerable de veces.

8 RESULTADOS

La librería OR-Library ofrece un total de 375 archivos que sirven como datos de entrada para el problema de tardanza total ponderada, distribuidos en tres grupos iguales en los que hay 40, 50 o 100 trabajos[1]. Las instancias son generadas a partir de las variables RDD y TF (Ver Marco Teórico), mientras que la función definida en la sección anterior cuenta con dos parámetros denotados como alfa (α) y el número de iteraciones (IT). Los primeros, establecen las características de los parámetros de los trabajos, mientras que los segundos, definen la forma en la que se van a procesar los datos.

De lo anterior, se puede inferir que existe un sinnúmero de posibles combinaciones con los que se podría probar cada una de las 375 instancias disponibles. Sin embargo, de antemano se sabe que a un mayor número de iteraciones, existe una mayor probabilidad de obtener un mejor resultado contrastando, a su vez, con la necesidad de procesamiento de la metaheurística en poco tiempo.

En vista de lo anterior, se hizo una prueba preliminar con 15 archivos de 40, 50 y 100 trabajos (5 de cada uno con el mismo RDD y TF definidos al azar) y un alfa de 0.05 para determinar el número de iteraciones en las que la función objetivo promedio tiene una variación porcentual pequeña con respecto a la respuesta arrojada por un número muy grande de IT. La variación calculada se realiza con la Ecuación 3:

Ecuación 3: Cálculo de la desviación de la respuesta i con it=j con respecto al mínimo valor calculado en la prueba

$$Desviación\ de\ la\ F_{obj}\ i\ con\ respecto\ a\ la\ mejor\ respuesta\ de\ F_{obj}\ para\ todas\ las\ it\ calculadas = \frac{(Promedio\ de\ la\ F_{obj}\ con\ it = j) - Mín\ (Valor\ promedio\ de\ la\ F_{obj}\ \forall it\ )}{Mín\ (Valor\ promedio\ de\ la\ F_{obj}\ \forall it)}\ X\ 100\%$$

Fuente: Presentación propia del autor

Por facilidad, se escogió una escala logarítmica para determinar el número de iteraciones y se promediaron las tardanzas ponderadas en instancias con igual número de trabajos. La Tabla 4 muestra los resultados obtenidos:

[1] Desarrollado por los profesores Richard K. Congam y Chris Potts (Universidad de Sothampton), y publicado por el profesor J E Beasley (Imperial College).

Tabla 4: Comparación de valores objetivos con distintos valores de iteraciones

Número de Trabajos	Iteraciones	Valor Promedio	Variación con respecto al mejor valor obtenido obtenido de la Fobj con el algoritmo planteado
40	1	198818.93	1.49%
	10	197408.67	0.77%
	100	196617.37	0.37%
	1000	196048.50	0.08%
	10000	195894.40	0.00%
50	1	313348.23	2.78%
	10	309791.77	1.61%
	100	307013.63	0.70%
	1000	305851.80	0.32%
	10000	304877.13	0.00%
100	1	1193669.63	2.59%
	10	1179923.83	1.41%
	100	1173975.73	0.90%
	1000	1167428.53	0.34%
	10000	1163511.70	0.00%

Fuente: Presentación propia del autor

Con el anterior cálculo se pudo confirmar, en primer lugar, que a un mayor número de iteraciones la función objetivo tiende a ser menor debido al aumento de la probabilidad de encontrar una mejor respuesta. Por otra parte, comparando porcentualmente los valores promedio de las funciones objetivo, se encontró que la variación entre los distintos valores analizados es muy pequeña (menor al 3% de variación con respecto al mejor valor obtenido con esta metaheurística), con lo cual queda abierta la posibilidad a escoger un valor muy pequeño para disminuir tiempo de procesamiento sin que la respuesta sea muy diferente al menor valor encontrado para un número de iteraciones bastante grande.

Sin embargo, aprovechando que el tiempo de procesamiento para 100 iteraciones es de una fracción de segundo y que, para los tres casos, su diferencia porcentual con respecto al mejor valor encontrado es de menos del 1%, se escogió este valor para IT y con el cual se obtuvieron los resultados que se presentan en las siguientes secciones.

8.1 Programación de los trabajos

Con el algoritmo presentado en el Capítulo 3, se simuló el problema para cada una de las instancias en 21 niveles del parámetro alfa definido entre 0 y 1 con variaciones de 0.05. Como ejemplo, se muestra el siguiente ejercicio cuya base es el archivo con nombre T040I097 que se refiere a la instancia número 97 de 40 trabajos encontrada en los archivos de OR-library, el cual, se sometió a 100 iteraciones con un parámetro alfa de 0.05. A continuación, se presenta el resumen de resultados:

Tabla 5: Instancia de prueba

		114686		
N	**P**	**R**	**W**	**D**
1	70	0	8	76
2	43	0	4	782
3	42	0	6	0
4	48	0	6	0
5	8	0	2	231
6	70	0	3	340
7	18	0	1	381
8	57	0	3	3
9	49	0	6	0
10	72	0	4	730
11	92	0	9	93
12	11	0	10	0
13	100	0	7	273
14	68	0	1	114
15	74	0	9	0
16	46	0	7	0
17	1	0	4	0
18	30	0	8	0
19	52	0	4	0
20	97	0	8	0
21	16	0	1	624
22	84	0	4	0
23	93	0	9	588
24	3	0	3	0
25	38	0	2	157
26	8	0	1	0
27	44	0	3	0
28	8	0	6	0
29	89	0	8	0
30	34	0	1	21
31	26	0	10	0
32	57	0	3	760
33	19	0	1	506
34	48	0	9	69
35	41	0	4	0
36	99	0	7	0
37	6	0	9	0
38	87	0	7	127
39	68	0	6	0
40	86	0	5	184

Fuente: OR-library

Tabla 6: Presentación de la instancia por Lotes

N	P	R	W	D	L
32	57	0	3	760	1
36	99	0	7	0	1
2	43	0	4	782	1
24	3	0	3	0	1
15	74	0	9	0	1
31	26	0	10	0	1
14	68	0	1	114	1
39	68	0	6	0	1
37	6	0	9	0	1
3	42	0	6	0	1
13	100	0	7	273	2
35	41	0	4	0	2
38	87	0	7	127	2
5	8	0	2	231	2
25	38	0	2	157	2
23	93	0	9	588	2
21	16	0	1	624	2
22	84	0	4	0	2
40	86	0	5	184	2
27	44	0	3	0	3
20	97	0	8	0	3
6	70	0	3	340	3
19	52	0	4	0	3
34	48	0	9	69	3
9	49	0	6	0	3
12	11	0	10	0	3
17	1	0	4	0	3
26	8	0	1	0	3
33	19	0	1	506	3
8	57	0	3	3	3
29	89	0	8	0	3
28	8	0	6	0	3
30	34	0	1	21	3
18	30	0	8	0	3
1	70	0	8	76	3
11	92	0	9	93	4
4	48	0	6	0	4
10	72	0	4	730	4
16	46	0	7	0	4
7	18	0	1	381	4

Fuente: Presentación propia del autor

Tabla 7: Presentación final de los datos

N	P	R	W	D	Fc	Tiempo	T	WT
37	6	0	9	0	0.67	6	6	54
24	3	0	3	0	3.00	9	9	27
31	26	0	10	0	3.50	35	35	350
15	74	0	9	0	16.89	109	109	981
2	43	0	4	782	10.75	152	0	0
3	42	0	6	0	41.83	194	194	1164
32	57	0	3	760	19.00	251	0	0
39	68	0	6	0	69.67	319	319	1914
36	99	0	7	0	50.00	418	418	2926
14	68	0	1	114	372.00	486	372	372
23	93	0	9	588	10.33	579	0	0
13	100	0	7	273	60.29	679	406	2842
21	16	0	1	624	16.00	695	71	71
35	41	0	4	0	227.25	736	736	2944
40	86	0	5	184	136.80	822	638	3190
38	87	0	7	127	93.57	909	782	5474
5	8	0	2	231	385.00	917	686	1372
22	84	0	4	0	248.25	1001	1001	4004
25	38	0	2	157	441.00	1039	882	1764
12	11	0	10	0	105.00	1050	1050	10500
18	30	0	8	0	143.75	1080	1080	8640
1	70	0	8	76	130.50	1150	1074	8592
28	8	0	6	0	201.00	1158	1158	6948
34	48	0	9	69	125.44	1206	1137	10233
9	49	0	6	0	224.00	1255	1255	7530
29	89	0	8	0	161.88	1344	1344	10752
17	1	0	4	0	373.50	1345	1345	5380
19	52	0	4	0	373.25	1397	1397	5588
20	97	0	8	0	180.13	1494	1494	11952
27	44	0	3	0	536.00	1538	1538	4614
6	70	0	3	340	408.00	1608	1268	3804
33	19	0	1	506	1178.00	1627	1121	1121
8	57	0	3	3	554.00	1684	1681	5043
26	8	0	1	0	1692.00	1692	1692	1692
30	34	0	1	21	1705.00	1726	1705	1705
4	48	0	6	0	311.00	1774	1774	10644
11	92	0	9	93	191.67	1866	1773	15957
16	46	0	7	0	273.14	1912	1912	13384
7	18	0	1	381	1621.00	1930	1549	1549
10	72	0	4	730	313.50	2002	1272	5088

Fuente: Presentación propia del autor

En la Tabla 5, se puede apreciar cómo se presentan los datos en la librería con las características de tiempo de procesamiento, tiempo de liberación, ponderación y fecha de vencimiento. En la siguiente tabla, se muestran los mismos datos, con la diferencia de que ahí cada trabajo pertenece a uno de los cuatro lotes sin que exista relación entre éste y su numeración inicial. Al final, los últimos datos muestran el orden en el que fueron programados los trabajos del problema *on-line* con sus respectivas funciones de costo, tiempo de salida del trabajo del sistema, tardanza y tardanza ponderada. De esta última tabla, también se hace un reconocimiento al hecho de que la metaheurística GRASP está operando según los criterios establecidos en cada una de sus fases. En particular, se

observa en la columna de la función de costo (fc) que los valores se encuentran aparentemente de forma ascendente (asociado a la fase constructiva) aunque algunos de ellos se salen de ese patrón (asociado a la fase de búsqueda local).

Para demostrar esta relación, hay que hacer notar que la fase constructiva es bastante miope debido al valor de alfa (recordando que para el problema α=0.05), haciendo que la RCL siempre esté conformada por los menores valores de la función de costo. Posteriormente, la búsqueda *2-optimal* hace permutaciones entre los trabajos del mismo lote hasta encontrar un mejor valor de la función objetivo, lo cual se refleja en que un valor muy alto de la función de costo no parezca respetar el patrón de ascendencia como el caso del trabajo número 33 del tercer lote. Por supuesto, este hecho no es fácilmente apreciable en la medida en la que el valor de alfa aumenta ya que el número de elementos de la lista de elegibles en la fase constructiva es mayor y la aleatoriedad aumenta (Ver Anexo 1).

Por otra parte, el algoritmo responde al hecho anunciado anteriormente de que la forma de procesamiento por lotes debe atender la situación en la que algunos trabajos se encuentran vencidos desde el momento en el que llegan al puesto de trabajo. En respuesta a ello, la función de costo adoptada trata de compensar la tardanza y el peso del trabajo para que sumen lo menos posible al resultado. Como consecuencia, se observa que los lotes del problema anterior se programan en su mayoría según la lógica de que los trabajos de vencimiento igual a 0 y los de mayor peso deben estar en las primeras posiciones. Para ilustrar esa característica del procedimiento, se invita al lector a que haga seguimiento a los trabajos número 14 y 12 en los cuadros anteriores (aunque queda abierta la posibilidad a que éste escoja el que desee).

Para el primer caso, el trabajo llega en el lote número 1 con otros 9 trabajos, presenta la menor ponderación entre éstos (w=1) y su fecha de vencimiento es la penúltima del grupo (d = 140, seguido por el trabajo 32 –w=3, d = 760- y el trabajo 2 –w=4, d=782 -). Como resultado, la respuesta lo ubica en el último lugar del lote con la mayor función de costo calculada en el momento de su programación. Por su parte, el trabajo número 12, llega en el lote 3 con otros 15 trabajos, presenta la mayor ponderación y un vencimiento en el momento 0, de tal manera que el algoritmo lo ubica en el primer lugar del lote.

En resumen, el anterior resultado representa la funcionalidad del algoritmo para generar secuencias bajo una restricción que se le ha añadido al problema *off-line*. Los datos consignados en el Anexo 1 muestran los resultados obtenidos en la simulación de todas las instancias para los niveles descritos.

8.2 Medidas de interés

Además de obtener un orden específico y una función objetivo para cada dato de entrada, se realizaron otras mediciones de interés para el proyecto tales como el valor promedio de la función objetivo en las 100 iteraciones, el tiempo de procesamiento en la secuenciación, el número de lotes, la brecha (GAP) computacional y la desviación con respecto al mejor resultado. Los dos últimos, se calcularon mediante la diferencia entre el resultado obtenido y el mejor valor del problema on-line para el GAP computacional y la desviación absoluta y relativa de la respuesta tomando en cuenta los indicadores descritos por Vallada y Ruiz (2009).

Ecuación 4: Cálculo del GAP computacional

$$GAP\ computacional\ de\ la\ instancia\ i = Valor\ de\ la\ Fobj\ obtenida\ para\ la\ instancia\ i - Mejor\ respuesta\ obtenida\ para\ el\ la\ instancia\ i\ bajo\ criterio\ offline.$$

Fuente: Presentación propia del autor

Ecuación 5: Cálculo de la desviación

$$RPD\ (Relative\ Percentage\ Deviation) = \frac{Valor\ de\ la\ Fobj\ obtenido - Mejor\ valor\ conocido\ para\ el\ problema\ offline}{Mejor\ valor\ conocido\ para\ el\ problema\ offline}\ x\ 100\%$$

$$RDI\ (Relative\ Deviation\ Index) = \frac{Respuesta\ del\ método\ de\ solución - Mejor\ valor\ conocido\ para\ el\ problema}{Peor\ valor\ conocido\ para\ el\ problema - Mejor\ valor\ conocido\ para\ el\ problema}\ x\ 100\%$$

Fuente: Vallada y Ruiz (2009)

El valor RPD se usó debido a que para los problemas que se están tratando, y sus respectivas características, no existe un estudio que haya estudiado el límite superior del mismo. Además, ha sido usado por autores como Caballero-Villalobos y Alvarado-Valencia (2010) para establecer el desempeño de su algoritmo con respecto a los mejores valores reportados. Por su parte, el porcentaje de desviación relativo no aplica para aquellos valores cuya mejor solución del problema *off-line* sea cero ya que el indicador es infinito a menos que la respuesta del algoritmo sea ese mismo valor (caso que no sucede). Por su parte, el índice relativo sí permite hacer una comparación de todas las instancias por lo que, para este caso en particular, se ha usado como peor valor el máximo valor que se obtuvo en alguna de las iteraciones corridas.

Finalmente, se hace aclaración que, si bien el RPD se define como una desviación porcentual, en la presentación de resultados se ha dejado este valor como una tasa (sin multiplicar por 100%) debido a que esta medida supera en gran número de casos el valor de 10, lo cual supone un porcentaje de desviación mayor al 1000%.

En el caso del ejemplo anterior, los valores se muestran en la siguiente tabla resumen:

TABLA 8: Cálculos resumen para la instancia T040I097

Cálculos para T040I097 (alfa = 0,05 ; it = 100)	
Número de lotes	4
Promedio de trabajos por cada lote	10
Mínimo número de trabajos por lote	5
Máximo número de trabajos por lote	16
Tardanza total ponderada	180165.00
GAP computacional	65479.00
Desviación al mejor medida con RDI	0.94
Desviación al mejor medida con RPD	0.57
Valor promedio de la función objetivo para 100 iteraciones	184539.44
Tiempo de procesamiento (segundos)	0.28

Fuente Propia del Autor

Antes de hacer el análisis estadístico para los diferentes resultados de todas las instancias, se rescatan algunos hechos importantes de la tabla anterior que se pueden aplicar a cualquier problema probado y que son mostrados en detalle en el Anexo 2.

A partir de la función objetivo se encuentra que su valor es muy superior a la mejor respuesta obtenida para el problema *off-line*. Por ejemplo, aquí la variación es de más del 57% aunque esto es atribuible principalmente a la condición *on-line* y las restricciones que eso conlleva. Para demostrar esta tolerancia al resultado, se verá en el Anexo 3 que el algoritmo probado para las instancias *off-line* (un solo lote), se obtienen desviaciones mucho más pequeñas y donde se demuestra que la forma en la que se concibe el algoritmo también representa parte de la desviación. Igualmente, sucede para el caso RDI donde la respuesta es muy cercana al máximo valor, lo que supone que la respuesta que arroja el algoritmo es muy cercana al peor valor que entrega el mismo. No obstante, para este caso particular, se puede ver que la diferencia entre el mínimo valor obtenido y valor promedio es muy pequeña, y algo similar sucede cuando se mide la diferencia entre éste último y el máximo obtenido que, a su vez, es producto de un ejercicio de minimización. En consecuencia, la diferencia entre el numerador y denominador del RDI es muy pequeña y se tienen valores muy cercanos a la unidad.

Por su parte, el valor promedio de las 100 iteraciones representa la importancia de correr la metaheurística en varias ocasiones ya que, para este caso, la tardanza total ponderada de la respuesta es 4374,44 unidades menor que la media registrada por todas las eventuales respuestas, en cuyo caso, es un reflejo adicional del funcionamiento del algoritmo ya que permanece en busca de un mejor valor.

Por último, se exalta el bajo tiempo de programación que para este caso es de tan sólo 28 centésimas de segundo. Esto significa que en un tiempo casi imperceptible para una

persona, se están desarrollando 100 respuestas diferentes atribuibles a la plataforma de programación y la máquina en la que se corrió el problema[2]. Esto soporta la idea de utilizar una metaheurística para solucionar un problema de tipo *NP-Hard*.

8.3 Estadísticas descriptivas para todas las pruebas

Los datos que se presentaron en la Tabla 8, fueron replegados para cada una de las instancias con miras a desarrollar un análisis estadístico de cada resultado. Dicho análisis comprende las medidas de tendencia central, dispersión y los histogramas que cada una arroja. Con la ayuda del programa SPSS®, se obtuvieron las siguientes tablas y figuras.

Tabla 9: Estadísticas descriptivas de los resultados de interés para 40 Trabajos

Estadísticos para instancias de 40 trabajos						
		Tardanza Ponderada	Desviación RPD	Desviación RDI	Tiempo de Procesamiento	Número de Lotes
N	Válidos	2625	2247	2625	2625	2625
	Perdidos	0	378	0	0	0
Media		89455.83	17.78	.798	.327	5.72
Mediana		76478.00	1.87	.820	.308	6.00
Desv. típ.		64635.93	68.16	.132	.076	1.53
Varianza		4.18E+09	4645.97	.017	.006	2.36
Rango		290071	967.41	.920	.688	9.00

Fuente: Presentación propia del autor

Tabla 10: Estadísticas descriptivas de los resultados de interés para 50 Trabajos

Estadísticos para instancias de 50 trabajos						
		Tardanza Ponderada	Desviación RPD	Desviación RDI	Tiempo de Procesamiento	Número de Lotes
N	Válidos	2625	2268	2625	2625	2625
	Perdidos	0	357	0	0	0
Media		132894.95	113.17	.809	.444	5.96
Mediana		120766.00	2.01	.820	.424	6.00
Desv. típ.		95258.18	826.24	.120	.097	1.65
Varianza		9.07E+09	682667.87	.014	.009	2.72
Rango		402955	14267.23	.850	1.049	11.00

Fuente: Presentación propia del autor

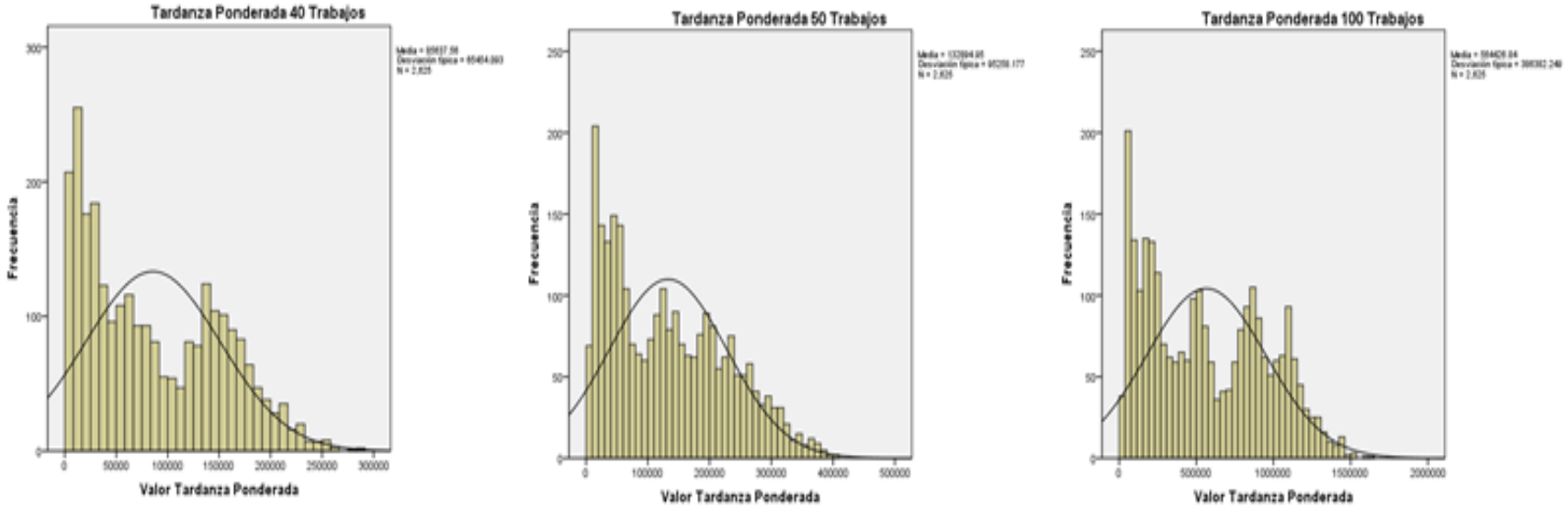
[2] El problema se desarrolló en un computador con procesador Intel Core i5, 3GB de memoria RAM y 500GB de disco duro.

Tabla 11: Estadísticas descriptivas de los resultados de interés para 100 Trabajos

Estadísticos para instancias de 100 trabajos						
		Tardanza Ponderada	Desviación RPD	Desviación RDI	Tiempo de Procesamiento	Número de Lotes
N	Válidos	2625	2268	2625	2625	2625
	Perdidos	0	357	0	0	0
Media		564426.84	151.39	.843	1.310	6.60
Mediana		514694.00	2.09	.860	1.274	6.00
Desv. típ.		386382.25	736.31	.095	.284	1.81
Varianza		1.49E+11	542149.46	.009	.080	3.29
Rango		1624587	10609.43	.650	2.388	11.00

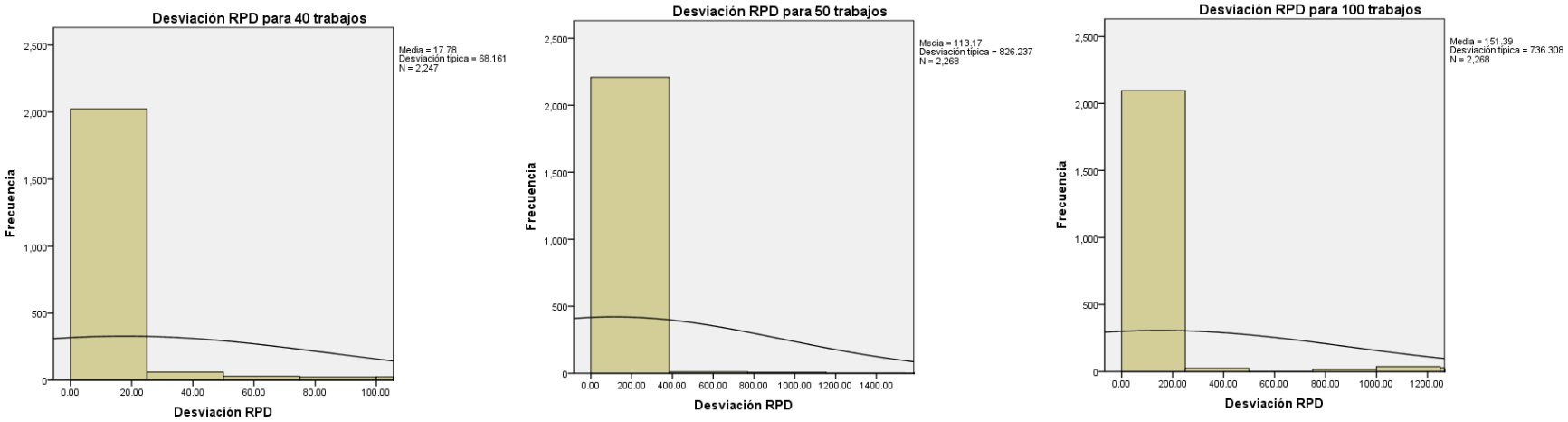
Fuente: Presentación propia del autor

Figura 6: Histogramas de la tardanza total ponderada para el problema *on-line*



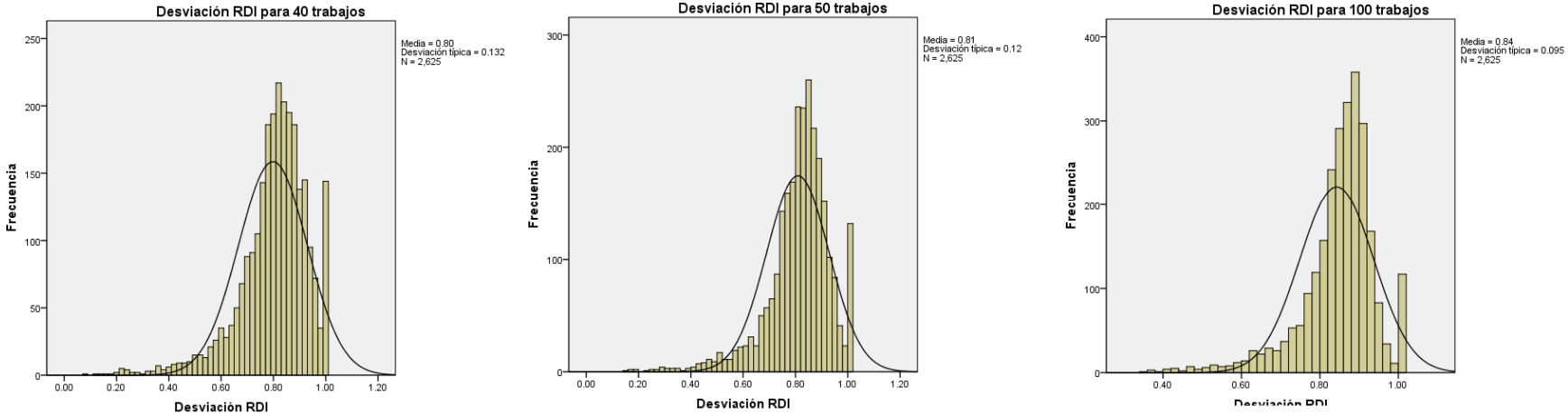
Fuente: Presentación propia del autor

Figura 7: Histogramas de la desviación RPD de las respuestas con respecto al problema *off-line*



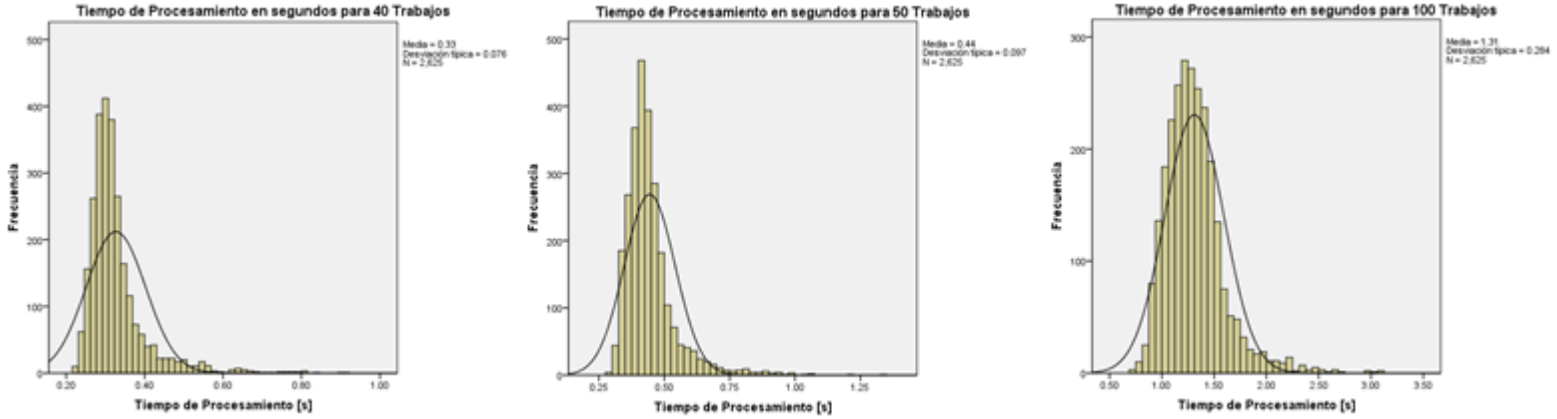
Fuente: Presentación propia del autor

Figura 8: Histogramas de la desviación RDI de las respuestas con respecto al problema *off-line*



Fuente: Presentación propia del autor

Figura 9: Histograma de los tiempo de procesamiento



Fuente: Presentación propia del autor

Figura 10: Histograma de los números de lotes en los que se reparten las instancias originales

Fuente: Presentación propia del autor

Con respecto a la información mostrada en las tablas y gráficos anteriores, se descubren varias características del algoritmo, encontrando similitudes entre los distintos tipos de instancias disponibles para el problema.

En primer lugar, la tardanza ponderada presenta valores de tendencia central alejados del cero ideal y con una desviación estándar bastante elevada que representa el 72.3%, 71.7% y 68.5% de la media para las instancias de 40, 50 y 100 trabajos, respectivamente. Lo anterior, significa que existe una gran dispersión de la respuesta para los diferentes datos y que, a su vez, se ve reflejado en los histogramas donde los resultados presentan frecuencias similares en los valores más bajos y más altos de la variable respuesta. Esto último, hace suponer la influencia de algunas características del problema en la variable respuesta y que será estudiada en el siguiente capítulo.

En segundo lugar, se destaca que la tasa de desviación RPD con respecto a la mejor solución encontrada para el problema *off-line*, tiene valores muy elevados para todas las estadísticas, y que esto está ligado a la cantidad de trabajos que se procesen. Así, por ejemplo, para 40 trabajos se espera que la diferencia entre las respuestas sea 17.4 veces más que la mejor solución encontrada para el problema *off-line*, mientras que para 100 trabajos esa distancia supera las 150 veces. No obstante, la mediana y los histogramas reflejan que la gran mayoría de los datos se encentran en los valores más pequeños del indicador. En particular, este valor muestra que al menos la mitad de los resultados obtenidos se encuentran 1.87, 2.01, 2.09 veces alejados del resultado off-line para 40, 50 y 100 trabajos respectivamente. Por supuesto, lo anterior no deja de ser un indicador alto y se hace notar que ese hecho está relacionado con la naturaleza del problema que aquí se plantea ya que, al no tener ningún punto de comparación con un problema similar, no se obtiene el mismo resultado cuando la totalidad de los trabajos llegan simultáneamente que cuando existe la restricción de un tiempo de llegada distinto a 0.

En ese mismo sentido, el índice RDI refleja una tendencia hacia los valores más altos del indicador. Aquí, no se puede establecer que el algoritmo es malo debido a que, en la aplicación de la ecuación, se está suponiendo que éste es el mismo que arroja los peores resultados posibles. Por el contrario, se puede decir que para el gran número de todos los casos (con el RPD no se puede concluir de la misma manera), el mínimo valor se encuentra alejado cerca del 20% del peor resultado conocido para el mismo problema. En otras palabras, se está consiguiendo el objetivo de seleccionar el mejor valor de las iteraciones y éste, en promedio, se acerca un 20% más al mejor valor obtenido para el problema *off-line* que la peor respuesta obtenida con el mismo algoritmo.

Además de esto, existe la satisfacción de haber obtenido tiempos de procesamiento muy pequeños para todas las corridas del algoritmo debido a que éstos presentan distribuciones que se centran en valores cercanos a 1 segundo (inferiores valores para 40 y 50 trabajos, y superiores para 100 trabajos), al igual que se tiene la certeza de que

ninguna medición superó los 4 segundos (Ver Anexo 2). Este hecho, demuestra la importancia de haber desarrollado el proyecto bajo la perspectiva de un lenguaje matemático que, como ya se había mencionado, enfoca un problema de la materia con funciones adaptadas para ello. Así, en contraste a lo sucedido con lo visto hasta el momento, este parámetro pudiera competir con otros algoritmos desarrollados para el problema *off-line*, aunque hay que mencionar que este es un punto que poco se menciona en los artículos estudiados ya que ahí se habla en mayor medida la capacidad de procesamiento de la máquina que del lenguaje de programación.

Para concluir estos comentarios, se hace referencia a la cantidad de lotes que, a diferencia de las anteriores descripciones parece distribuirse de la misma manera para los tres casos, con una media centrada entre 5 y 7 lotes para los diferentes tipos de datos. Lo anterior, se le atribuye a la función aleatoria 'randi' del programa MATLAB® que es la única que no es controlada por usuario en la simulación (Ver Anexo 5).

8.4 Correlación de variables

El tiempo de procesamiento es una de las medidas más importantes en los problemas de *scheduling*, al punto que es un determinante para la comparación de problemas (Ver Marco Teórico). Así, en lo concerniente al problema tratado, se tiene que a pesar de que los tiempos de procesamiento son bastante pequeños, es posible que existan parámetros que determinen en parte el valor del tiempo de procesamiento.

En ese caso, se pensó que las únicas variables que pudiesen aumentar el tiempo de procesamiento son el valor de alfa y el número de lotes. En el primer caso, a un mayor valor, la lista de trabajos elegibles (RCL) se incrementa y la máquina consume mayor tiempo en conformarla. Por otra parte, a un mayor número de lotes, menor cantidad de trabajos se deben programar y los pasos que se deben efectuar son menores.

En consecuencia, se realizó un análisis de varianza para contrastar la hipótesis nula de que las medias de los tiempos de procesamiento son iguales obteniendo las siguientes tablas para 40, 50 y 100 trabajos de forma independiente.

Tabla 12: Análisis de varianza para el tiempo de procesamiento para 40 trabajos

Pruebas de los efectos inter-sujetos

Variable dependiente:Tiempo_de_Procesamiento

Origen	Suma de cuadrados tipo III	gl	Media cuadrática	F	Sig.
Modelo corregido	2.698[a]	169	.016	3.147	.000
Intersección	24.540	1	24.540	4836.482	.000
Alfa	.435	20	.022	4.285	.000
Número_de_Lotes	.988	9	.110	21.639	.000
Alfa * Número_de_Lotes	.784	140	.006	1.104	.197
Error	12.456	2455	.005		
Total	295.198	2625			
Total corregida	15.155	2624			

a. R cuadrado = .178 (R cuadrado corregida = .121)

Fuente: Presentación propia del autor

Tabla 13: Análisis de Varianza para el tiempo de procesamiento con 50 trabajos

Pruebas de los efectos inter-sujetos

Variable dependiente:Tiempo_de_Procesamiento

Origen	Suma de cuadrados tipo III	gl	Media cuadrática	F	Sig.
Modelo corregido	4.702[a]	184	.026	3.092	.000
Intersección	33.447	1	33.447	4047.076	.000
Alfa	.489	20	.024	2.961	.000
Número_de_Lotes	1.976	11	.180	21.733	.000
Alfa * Número_de_Lotes	.982	153	.006	.777	.979
Error	20.166	2440	.008		
Total	542.145	2625			
Total corregida	24.867	2624			

a. R cuadrado = .189 (R cuadrado corregida = .128)

Fuente: Presentación propia del autor

Tabla 14: Análisis de varianza para el tiempo de procesamiento de 100 trabajos

Pruebas de los efectos inter-sujetos

Variable dependiente:Tiempo_de_Procesamiento

Origen	Suma de cuadrados tipo III	gl	Media cuadrática	F	Sig.
Modelo corregido	74.735[a]	207	.361	6.395	.000
Intersección	424.699	1	424.699	7522.203	.000
Alfa	7.736	20	.387	6.851	.000
Número_de_Lotes	39.080	11	3.553	62.926	.000
Alfa * Número_de_Lotes	15.017	176	.085	1.511	.000
Error	136.462	2417	.056		
Total	4718.716	2625			
Total corregida	211.197	2624			

a. R cuadrado = .354 (R cuadrado corregida = .299)

Fuente: Presentación propia del autor

De las anteriores tablas, se rechazó la hipótesis nula con un nivel de significancia del 5%, encontrando que sí existe una influencia de los parámetros como se había pensado. En especial, a un mayor número de trabajos, los tiempos de procesamiento se explican mejor por los parámetros medidos, aunque dicha explicación no es muy contundente como se refleja en el coeficiente de correlación de las tablas anteriores.

De esta manera, se realizaron nuevas pruebas de correlación para establecer estadística y gráficamente cómo se comporta la relación Alfa vs Tiempo de procesamiento y Número de lotes vs Tiempo de procesamiento, tal como se muestra a continuación:

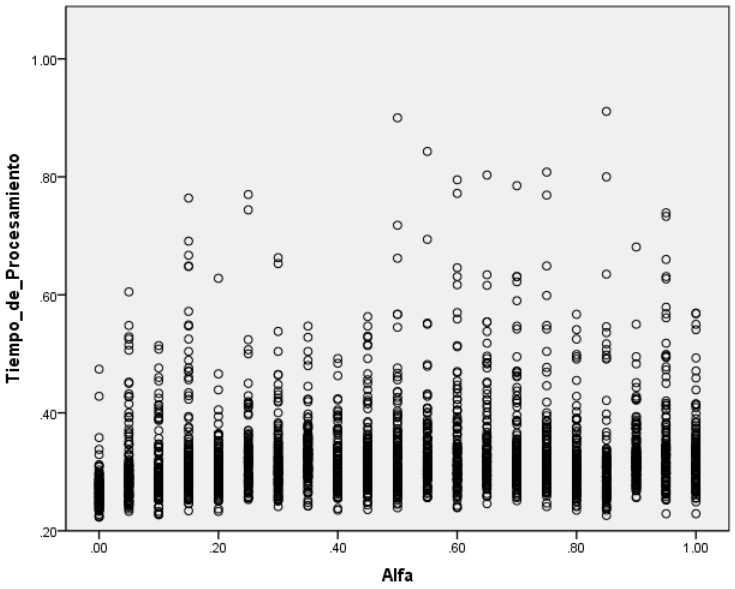
8.4.1 40 Trabajos

Tabla15: Correlación entre el tiempo de procesamiento y el parámetro alfa

Correlaciones			
		Alfa	Tiempo de Procesamiento
Alfa	Correlación de Pearson	1	.139**
	Sig. (bilateral)		0
	N	2625	2625
Tiempo de Procesamiento	Correlación de Pearson	.139**	1
	Sig. (bilateral)	0	
	N	2625	2625
**. La correlación es significativa al nivel 0,01 (bilateral).			

Fuente: Presentación propia del autor

Figura 10: Gráfico de correlación



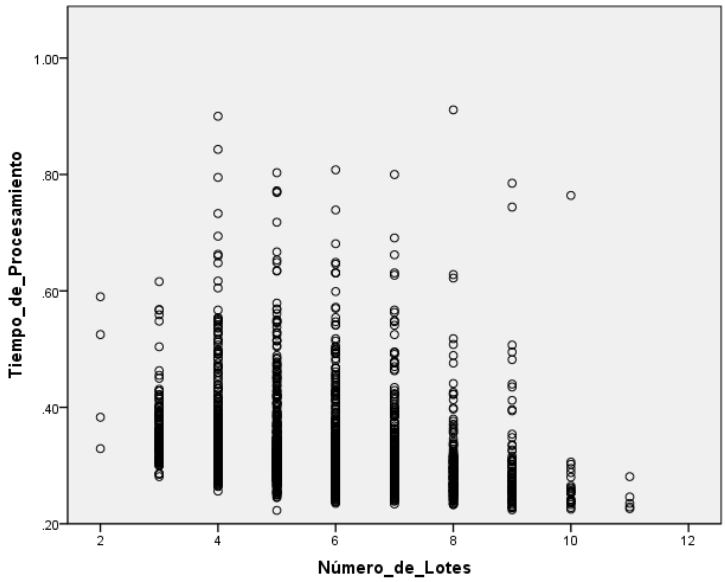
Fuente: Presentación propia del autor

Tabla16: Correlación entre el tiempo de procesamiento y el número de lotes

Correlaciones			
		Número de Lotes	Tiempo de Procesamiento
Número de Lotes	Correlación de Pearson	1	-.255**
	Sig. (bilateral)		0
	N	2625	2625
Tiempo de Procesamiento	Correlación de Pearson	-.255**	1
	Sig. (bilateral)	0	
	N	2625	2625
**. La correlación es significativa al nivel 0,01 (bilateral).			

Fuente: Presentación propia del autor

Figura 11: Gráfico de Correlación



Fuente: Presentación propia del autor

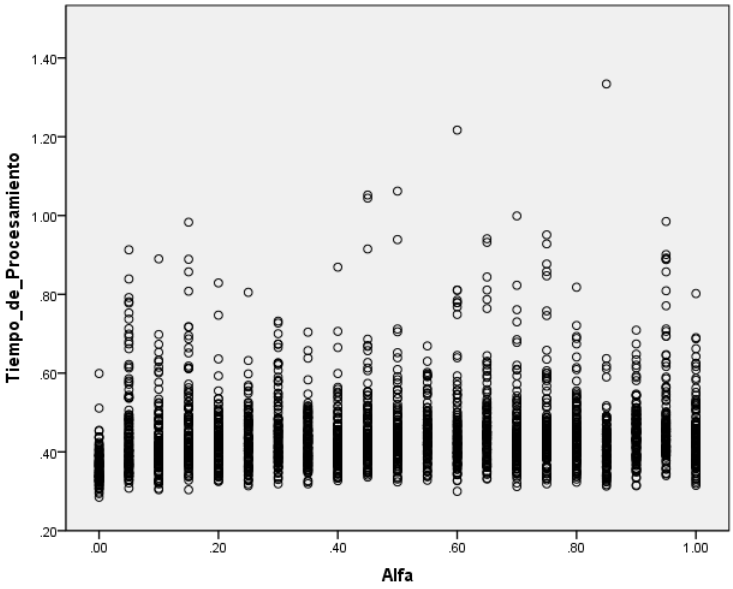
8.4.2 50 Trabajos

Tabla 17: Correlación entre el tiempo de procesamiento y el parámetro alfa

Correlaciones			
		Alfa	Tiempo de Procesamiento
Alfa	Correlación de Pearson	1	.119**
	Sig. (bilateral)		0
	N	2625	2625
Tiempo de Procesamiento	Correlación de Pearson	.119**	1
	Sig. (bilateral)	0	
	N	2625	2625
**. La correlación es significativa al nivel 0,01 (bilateral).			

Fuente: Presentación propia del autor

Figura 12: Gráfico de Correlación



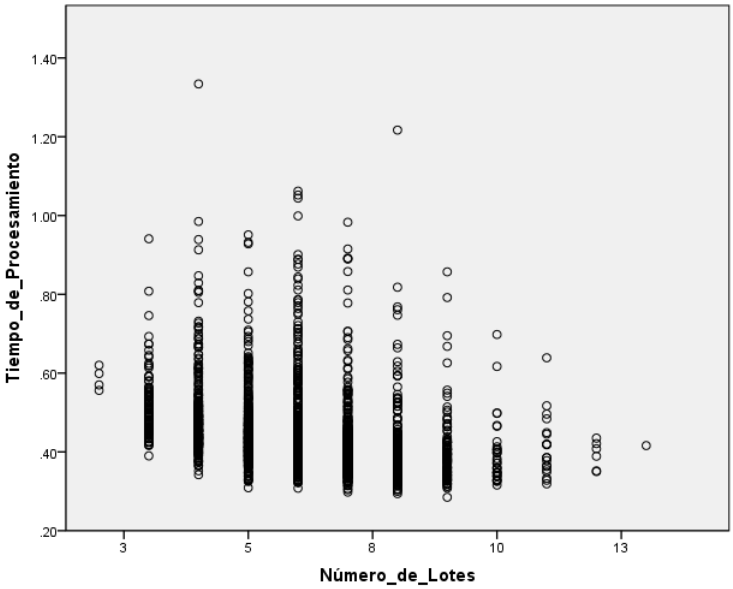
Fuente: Presentación propia del autor

Tabla 18: Correlación entre el tiempo de procesamiento y el número de lotes

Correlaciones			
		Número de Lotes	Tiempo de Procesamiento
Número de Lotes	Correlación de Pearson	1	-.284**
	Sig. (bilateral)		0
	N	2625	2625
Tiempo de Procesamiento	Correlación de Pearson	-.284**	1
	Sig. (bilateral)	0	
	N	2625	2625
**. La correlación es significativa al nivel 0,01 (bilateral).			

Fuente: Presentación propia del autor

Figura 13: Gráfico de Correlación



Fuente: Presentación propia del autor

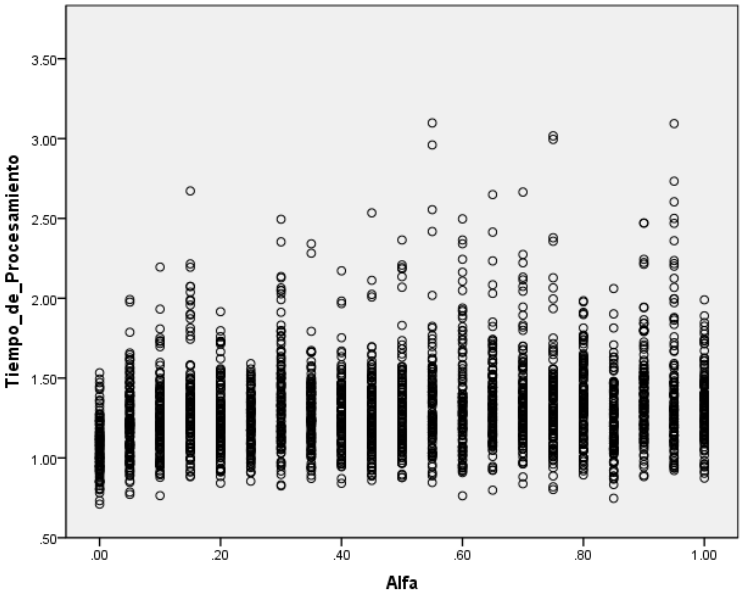
8.4.3 100 Trabajos

Tabla19: Correlación entre el tiempo de procesamiento y el parámetro alfa

Correlaciones			
		Alfa	Tiempo de Procesamiento
Alfa	Correlación de Pearson	1	.184**
	Sig. (bilateral)		0
	N	2625	2625
Tiempo de Procesamiento	Correlación de Pearson	.184**	1
	Sig. (bilateral)	0	
	N	2625	2625
**. La correlación es significativa al nivel 0,01 (bilateral).			

Fuente: Presentación propia del autor

Figura 14: Gráfico de correlación



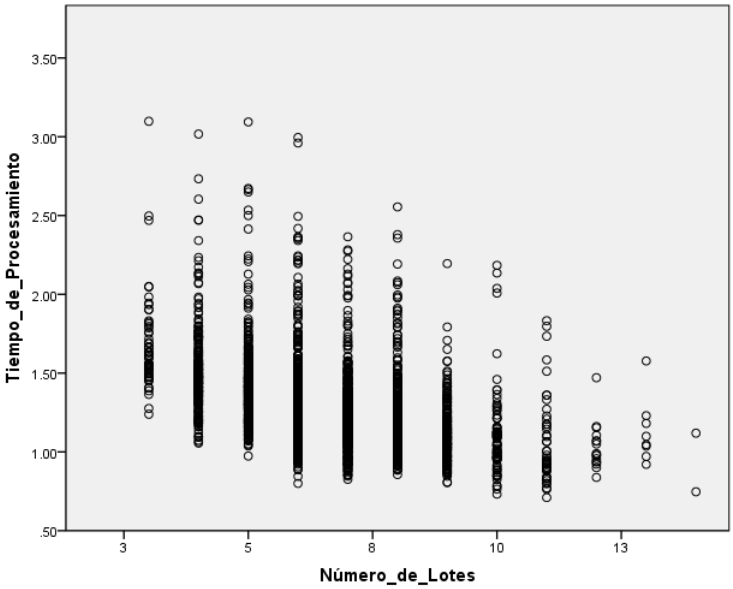
Fuente: Presentación propia del autor

Tabla 20: Correlación entre el tiempo de procesamiento y el número de lotes

Correlaciones			
		Número de Lotes	Tiempo de Procesamiento
Número de Lotes	Correlación de Pearson	1	-.439**
	Sig. (bilateral)		0
	N	2625	2625
Tiempo de Procesamiento	Correlación de Pearson	-.439**	1
	Sig. (bilateral)	0	
	N	2625	2625
**. La correlación es significativa al nivel 0,01 (bilateral).			

Fuente: Presentación propia del autor

Figura 15: Gráfico de correlación



Fuente: Presentación propia del autor

De todo lo anterior, se concluye, que el tiempo de procesamiento del algoritmo planteado está ligado linealmente al parámetro alfa de manera positiva y de manera negativa con el número de lotes en el que se divide el trabajo. Este último, en particular, con una mayor correlación que explica la variable de salida.

En resumen, un menor tiempo de procesamiento se podría lograr si el usuario del programa mantiene un parámetro miope muy pequeño con una división en lotes bastante considerable. Sin embargo, debido a la forma en la que se concibe el algoritmo, si se llega al extremo de las dos variables independientes, se tendría un proceso completamente miope que no considera muchos trabajos en la RCL y, por la otra parte, debido a que se respeta el orden de llegada de los lotes, si la división fuera máxima (cada trabajo es un lote), no habría cambio en la forma en la que llegan los trabajos y en la que son programados. Por lo tanto, se recomienda que si se desea poco tiempo de procesamiento, se considere un valor de alfa mayor a cero y un número de lotes menor al número total de trabajos.

9 EXPERIMENTO ESTADÍSTICO

Sin perder el horizonte del problema, que es disminuir la tardanza total ponderada, son muy dicientes las tablas e histogramas del capítulo 8.3, en el sentido que los valores no se concentran en un lugar del rango sino que existen frecuencias similares para valores muy diferentes. En esa misma dirección, las frecuencias parecen comportarse de manera irregular a pesar que la distribución por lotes es similar para cada caso.

Por tal razón, se planteó un experimento que busca probar la hipótesis de que no existe una diferencia significativa de las medias a pesar de partir de que la naturaleza de los trabajos es distinta (RDD y TF) o el tratamiento del algoritmo sea diferente. Estos últimos, son tres de los cuatro parámetros sobre los cuales el usuario tiene conocimiento previo a la simulación del algoritmo, y sabiendo que el faltante es el número de iteraciones que, por razones ya enunciadas, se excluye de esta revisión. Para ello se ha definido una significancia del 5%. A continuación se presenta el modelo que se quiere probar y las respectivas pruebas de hipótesis que se desean analizar:

Ecuación 6: Modelo planteado para el análisis de varianza mejorado

$$Y_{ijkl} = \mu + \alpha_j + \delta_k + \tau_l + \alpha\delta_{jk} + \alpha\tau_{jl} + \delta\tau_{Jk} + \alpha\delta\tau_{jkl} + \varepsilon_{ijkl}$$

Donde:
μ = Media poblacional
α = Efecto del j-ésimo nivel del factor Alfa
δ = Efecto en el k-ésimo nivel del parámetro Due Date Range
τ = Efecto factor en el l-ésimo nivel del parámetro Tightness Factor
αδ = Efecto de la interacción entre el j-éslimo nivel de Alfa y el k-ésimo nivel de RDD
ατ = Efecto de la interacción del j-ésimo nivel de Alfa y el l-ésimo nivel de TF
δτ =Efecto de la interacción entre el j-ésimo nivel de RDD y el k-ésimo nivel de TF
αδτ=Efecto de la triple interacción entre los niveles de Alfa, RDD y TF.
ε = Error aleatorio de cada nivel en la l-ésima observación

Fuente: Presentación propia del autor

Ecuación 7: Pruebas de hipótesis planteadas

$H_0: \alpha_1 = \alpha_2 = \cdots = \alpha_{21}$	$H_0: \alpha\gamma_{1,1} = \alpha\gamma_{1,2} = \cdots = \alpha\gamma_{21,5}$
$H_1: \alpha_1 \neq \alpha_2 \neq \cdots \neq \alpha_{21}$	$H_1: \alpha\gamma_{1,1} \neq \alpha\gamma_{1,2} \neq \cdots \neq \alpha\gamma_{21,5}$
$H_0: \delta_1 = \delta_2 = \cdots = \delta_5$	$H_0: \delta\tau_{1,1} = \delta\tau_{1,2} = \cdots = \delta\tau_{5,5}$
$H_1: \delta_1 \neq \delta_2 \neq \cdots \neq \delta_5$	$H_1: \delta\tau_{1,1} \neq \delta\tau_{1,2} \neq \cdots \neq \delta\tau_{5,5}$
$H_0: \gamma_1 = \gamma_2 = \cdots = \gamma_5$	$H_0: \alpha\delta\tau_{1,1,1} = \alpha\delta\tau_{1,1,2} = \cdots = \alpha\delta\tau_{21.5.5}$
$H_1: \gamma_1 \neq \gamma_2 \neq \cdots \neq \gamma_5$	$H_1: \alpha\delta\tau_{1,1,1} \neq \alpha\delta\tau_{1,1,2} \neq \cdots \neq \alpha\delta\tau_{21,5,5}$
$H_0: \alpha\delta_{1,1} = \alpha\delta_{1,2} = \cdots = \alpha\delta_{21,5}$	
$H_1: \alpha\delta_{1.1} \neq \alpha\delta_{1,2} \neq \cdots \neq \alpha\delta_{21,5}$	

Fuente: Presentación propia del autor

Para comprobar la relación de dicho modelo, se realizó un análisis de varianza con una probabilidad de Error Tipo I del 5% en el programa SPSS®, obteniendo las siguientes tablas como resultado:

Tabla 21: Análisis de Varianza para Instancias de 40 Trabajos

Pruebas de los efectos inter-sujetos

Variable dependiente:Tardanza_Ponderada

Origen	Suma de cuadrados tipo III	gl	Media cuadrática	F	Sig.
Modelo corregido	1.016E13	524	1.939E10	50.721	.000
Intersección	2.101E13	1	2.101E13	54951.770	.000
Alfa	5.522E9	20	2.761E8	.722	.807
RDD	3.126E10	4	7.815E9	20.443	.000
TF	9.848E12	4	2.462E12	6440.539	.000
Alfa * RDD	1.516E10	80	1.895E8	.496	1.000
Alfa * TF	1.002E10	80	1.252E8	.328	1.000
RDD * TF	2.100E11	16	1.313E10	34.341	.000
Alfa * RDD * TF	3.982E10	320	1.244E8	.326	1.000
Error	8.028E11	2100	3.823E8		
Total	3.197E13	2625			
Total corregida	1.096E13	2624			

a. R cuadrado = .927 (R cuadrado corregida = .909)

Fuente: Presentación propia del autor

Tabla 22: Análisis de varianza para 50 trabajos

Pruebas de los efectos inter-sujetos

Variable dependiente:Tardanza_Ponderada

Origen	Suma de cuadrados tipo III	gl	Media cuadrática	F	Sig.
Modelo corregido	2.225E13	524	4.246E10	57.009	.000
Intersección	4.636E13	1	4.636E13	62252.446	.000
Alfa	8.540E9	20	4.270E8	.573	.933
RDD	5.828E10	4	1.457E10	19.566	.000
TF	2.150E13	4	5.374E12	7216.766	.000
Alfa * RDD	2.321E10	80	2.901E8	.390	1.000
Alfa * TF	2.817E10	80	3.521E8	.473	1.000
RDD * TF	5.565E11	16	3.478E10	46.701	.000
Alfa * RDD * TF	7.420E10	320	2.319E8	.311	1.000
Error	1.564E12	2100	7.447E8		
Total	7.017E13	2625			
Total corregida	2.381E13	2624			

a. R cuadrado = .934 (R cuadrado corregida = .918)

Fuente: Presentación propia del autor

Tabla 23: Análisis de Varianza para instancias de 100 trabajos

Pruebas de los efectos inter-sujetos

Variable dependiente:Tardanza_Ponderada

Origen	Suma de cuadrados tipo III	gl	Media cuadrática	F	Sig.
Modelo corregido	3.770E14	524	7.196E11	102.850	.000
Intersección	8.363E14	1	8.363E14	119531.575	.000
Alfa	1.028E11	20	5.138E9	.734	.793
RDD	7.072E11	4	1.768E11	25.272	.000
TF	3.689E14	4	9.222E13	13182.018	.000
Alfa * RDD	2.254E11	80	2.817E9	.403	1.000
Alfa * TF	3.207E11	80	4.009E9	.573	.999
RDD * TF	6.263E12	16	3.914E11	55.946	.000
Alfa * RDD * TF	5.336E11	320	1.668E9	.238	1.000
Error	1.469E13	2100	6.996E9		
Total	1.228E15	2625			
Total corregida	3.917E14	2624			

a. R cuadrado = .962 (R cuadrado corregida = .953)

Fuente: Presentación propia del autor

De los anteriores análisis de varianza se concluye que la hipótesis nula se rechaza para un nivel de significancia del 5% para los tratamientos de RDD, TF y, por ende, existe evidencia de que esos parámetros influyen en la variable respuesta. Además, la prueba es contundente al establecer que los niveles para estos parámetros y su interacción determinan en gran medida el modelo al establecerse un valor-p muy pequeño.

Por su parte, el coeficiente de correlación se encuentra por encima del 95% y por lo tanto, se planteó un nuevo experimento cuyos tratamientos correspondieran únicamente a los mencionados y su interacción, con el fin de poner a prueba la misma hipótesis bajo un modelo más sencillo mostrado en la Ecuación 8. Dicho experimento se ha llamado el mejor análisis de varianza donde se ingresa la suma de los cuadrados del tratamiento alfa y sus interacciones al error como se refleja en el cambio del modelo planteado para el ejercicio:

Ecuación 8: Modelo planteado para el análisis de varianza mejorado

$$Y_{ijk} = \mu + \delta_j + \tau_k + \delta\tau_{jk} + \varepsilon_{ijk}$$

Donde:

μ = Media poblacional

δ = Efecto en el j-ésimo nivel del parámetro Due Date Range

τ = Efecto factor en el k-ésimo nivel del parámetro Tightness Factor

δτ =Efecto de la interacción entre el j-ésimo nivel de RDD y el k-ésimo nivel de TF

ε = Error aleatorio de cada nivel en la l-ésima observación

Fuente: Presentación propia del autor

Tabla 24: Mejor Análisis de Varianza para 40 Trabajos

Pruebas de los efectos inter-sujetos

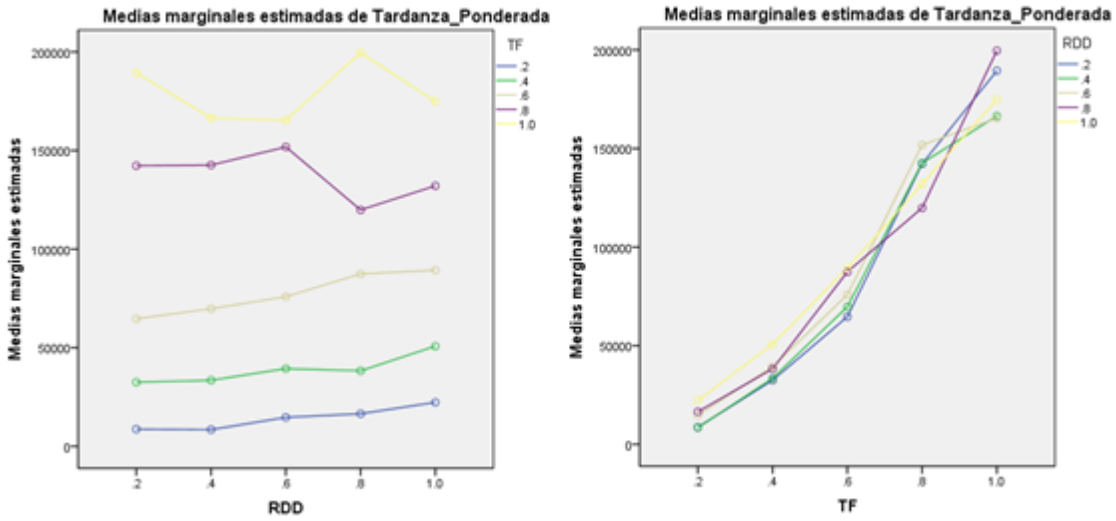
Variable dependiente:Tardanza_Ponderada

Origen	Suma de cuadrados tipo III	gl	Media cuadrática	F	Sig.
Modelo corregido	1.009E13	24	4.204E11	1251.608	.000
Intersección	2.101E13	1	2.101E13	62541.219	.000
RDD	3.126E10	4	7.815E9	23.267	.000
TF	9.848E12	4	2.462E12	7330.049	.000
RDD * TF	2.100E11	16	1.313E10	39.084	.000
Error	8.733E11	2600	3.359E8		
Total	3.197E13	2625			
Total corregida	1.096E13	2624			

a. R cuadrado = .920 (R cuadrado corregida = .920)

Fuente: Presentación propia del autor

Figura 16: Gráfica de contrastes



Fuente: Presentación propia del autor

Tabla 25: Mejor Análisis de Varianza para 50 Trabajos

Pruebas de los efectos inter-sujetos

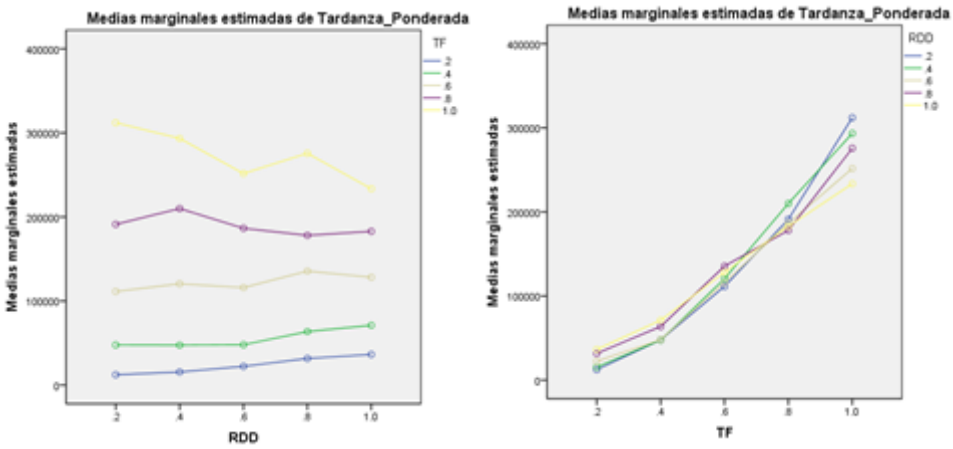
Variable dependiente:Tardanza_Ponderada

Origen	Suma de cuadrados tipo III	gl	Media cuadrática	F	Sig.
Modelo corregido	2.211E13	24	9.214E11	1410.768	.000
Intersección	4.636E13	1	4.636E13	70986.503	.000
RDD	5.828E10	4	1.457E10	22.311	.000
TF	2.150E13	4	5.374E12	8229.283	.000
RDD * TF	5.565E11	16	3.478E10	53.253	.000
Error	1.698E12	2600	6.531E8		
Total	7.017E13	2625			
Total corregida	2.381E13	2624			

a. R cuadrado = .929 (R cuadrado corregida = .928)

Fuente: Presentación propia del autor

Figura 17: Gráfica de contrastes



Fuente: Presentación propia del autor

Tabla 26: Mejor Análisis de Varianza para 100 Trabajos

Pruebas de los efectos inter-sujetos

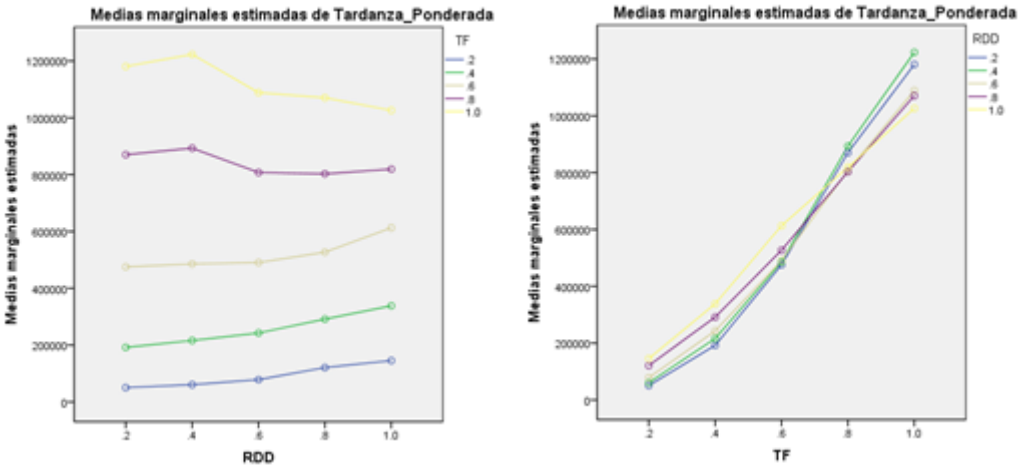
Variable dependiente:Tardanza_Ponderada

Origen	Suma de cuadrados tipo III	gl	Media cuadrática	F	Sig.
Modelo corregido	3.759E14	24	1.566E13	2565.041	.000
Intersección	8.363E14	1	8.363E14	136967.496	.000
RDD	7.072E11	4	1.768E11	28.958	.000
TF	3.689E14	4	9.222E13	15104.862	.000
RDD * TF	6.263E12	16	3.914E11	64.107	.000
Error	1.587E13	2600	6.106E9		
Total	1.228E15	2625			
Total corregida	3.917E14	2624			

a. R cuadrado = .959 (R cuadrado corregida = .959)

Fuente: Presentación propia del autor

Figura 18: Gráfica de contrastes



Fuente: Presentación propia del autor

Con lo anterior se observa que los valores RDD y TF, propios de la creación de la instancia, son los únicos que influyen en la variable respuesta en vista de que pueden explicar el modelo *on-line* en más del 90% para todos los casos. Así, se puede rechazar la hipótesis nula de igualdad de medias entre las réplicas debido a que sí existe influencia de estos parámetros y su interacción.

Para ser más objetivo, se hace referencia a las gráficas donde se ve con claridad que para los diferentes tipos de trabajos planteados, los menores valores de RDD y TF (0.2 para cada uno), existe una clara mejor respuesta. Además, las gráficas separadas de RDD muestran que sí existe interacción de las mismas.

No obstante, a partir de las gráficas no se puede concluir que la pareja RDD=0.2 y TF=0.2 sea la única con la que se obtienen las mínimas tardanzas ponderadas. Por lo tanto, resta determinar qué niveles de los parámetros son los significativamente diferentes y si el modelo se puede explicar mediante la función descrita en la Ecuación 4. Sin embargo, al efectuar las pruebas de homogeneidad de varianzas, se encontró que la variable respuesta no tiene varianzas homogéneas (Tabla 26) y por lo tanto se debe recurrir a una prueba no paramétrica para determinar qué medias son iguales.

Tabla 27: Contrastes de Levene para homogeneidad de varianzas

Contraste de Levene sobre la igualdad de las varianzas error[a]

Variable dependiente:Zobjetivo

F	gl1	gl2	Sig.
41.879	24	2600	.000

Contrasta la hipótesis nula de que la varianza error de la variable dependiente es igual a lo largo de todos los grupos.

a. Diseño: Intersección + RDD + TF + RDD * TF

Contraste de Levene sobre la igualdad de las varianzas error[a]

Variable dependiente:Zobjetivo

F	gl1	gl2	Sig.
52.306	24	2600	.000

Contrasta la hipótesis nula de que la varianza error de la variable dependiente es igual a lo largo de todos los grupos.

a. Diseño: Intersección + RDD + TF + RDD * TF

Contraste de Levene sobre la igualdad de las varianzas error[a]

Variable dependiente:Zobjetivo

F	gl1	gl2	Sig.
66.569	24	2600	.000

Contrasta la hipótesis nula de que la varianza error de la variable dependiente es igual a lo largo de todos los grupos.

a. Diseño: Intersección + RDD + TF + RDD * TF

Fuente: Presentación propia del autor

Cabe anotar, que este mismo procedimiento se realizó para valores normalizados de la media y sus raíces sin obtener resultados diferentes para dicha prueba. En ese caso, se aplicó la prueba de Tamhane (que supone la no-homogeneidad de varianzas) para la interacción y de esta manera determinar cuáles de los otros valores de RDD y TF no tienen una diferencia significativa con el mejor encontrado y, así, concluir cuáles son los mejores valores que más convienen al problema.

Los resultados de la prueba resumidos en el Anexo 4, dejan como resultado que para la interacción, se va a obtener mejores valores si se seleccionan los siguientes pares para RDD y TF respectivamente: 0.2-0.2, 0.4-0.2, 0.6-0.2 y 0.8-0.2 para 40 trabajos, 0.2-0.2, 0.4-0.2 y 0.6-0.2 para 50 trabajos, y 0.2-0.2 y 0.4-0.2 para 100 trabajos, debido a que no existe una diferencia significativa entre sus medias. Así, como se puede ver en las gráficas de interacción RDD vs Tardanza Ponderada, todas las respuestas pintadas con color azul en son estadísticamente iguales.

Desde el punto de vista práctico, esto sugiere que más allá de establecer los valores a los que tiene acceso el interesado en el algoritmo, éste debe ser consciente de que es la misma instancia la que define una mejor o peor respuesta. Por tal motivo, el programa no interfiere significativamente en la respuesta del problema a cualquier valor de alfa que el usuario desee implementar.

10 CONCLUSIONES

El presente proyecto considera un problema de gran estudio en los últimos años, al que se le agregó una característica que asemeje situaciones que se presentan en la realidad. De esta forma, se abordó un problema *on-line* para el que se demostró que la metaheurística GRASP ofrece un método de solución aceptable en la medida en la que busca siempre minimizar la función objetivo, considerando escenarios nuevos que no se contemplan generalmente en el problema *off-line*.

En el caso del algoritmo desarrollado para el presente proyecto, se pudo establecer que éste parte de soluciones aceptables desde la primera iteración y que el aumento del parámetro IT sólo incrementa la probabilidad de obtener una mejor respuesta aunque esta última no es muy diferente. Por su parte, la consideración del parámetro alfa, al igual que el número de lotes, sólo se encuentran relacionados con el tiempo de procesamiento haciendo que estos sean directamente proporcionales de forma lineal, positiva y negativamente, respectivamente.

Los resultados comparativos con la mejor respuesta encontrada para el problema *off-line* arroja como resultado que este algoritmo se encuentra alejado de dicha respuesta. Sin embargo, ese resultado es comprensible si se tiene en cuenta que, en primer lugar, la restricción de la llegada en lotes a través del tiempo de los trabajos agrega mayor dispersión a los datos. Además, que el mismo algoritmo corrido para el problema *off-line* sólo arroja los mejores valores encontrados para las 375 instancias en muy pocas ocasiones. Por lo tanto, es alentador el hecho de que el problema tratado busque acercarse a las mejores respuestas y deja abierta la puerta a encontrar mejoras como se planteará más adelante.

A partir de las pruebas estadísticas, se ha podido establecer que las mejores respuestas del algoritmo sólo están influenciadas por la naturaleza de los trabajos más que por los parámetros del mismo algoritmo. Por lo tanto, desde la perspectiva del presente trabajo de grado se tiene que, en promedio, un mayor valor de la variable respuesta estará definido por cómo se distribuyen las tardanzas y, para el caso en el que éstos estén dados por los parámetros RDD y TF, se obtendrán mejores resultados para la combinación 0.2 – 0.2 y sus similares para los diferentes tipos de instancias.

El avance de la tecnología y los lenguajes de programación han hecho que el tiempo de solución de los problemas sea cada vez más pequeño y se puedan atender problemas más grandes en un menor tiempo. Por ejemplo, uno de los trabajos más citados en este documento es el procedimiento elaborado por Caballero-Villalobos y Alvarado-Valencia (2010) en el que concluyen con éxito que su algoritmo no tarda más de 2 minutos en realizar la secuencia de 100 trabajos. Aquí, guardando las proporciones, ese tiempo no supera los 4 segundos, lo que significa una ventaja considerable a la hora de generar

respuestas inmediatas o de establecer una búsqueda más intensiva (mayores iteraciones) para encontrar mejores respuestas.

Como se ha tratado a lo largo de los capítulos, el presente proyecto ofrece un extenso número de oportunidades para mejorar la herramienta o extender su aplicación a nuevos problemas. Con el algoritmo propuesto, se está aportando una nueva posibilidad de ver un caso específico de programación con ayuda de un procedimiento recientemente incorporado a los métodos de solución (teniendo en cuenta que este problema ha sido estudiado desde hace más de 40 años). Así, por ejemplo, es posible probar otras funciones de costo o alterativas para la selección de trabajos en la fase constructiva, y también se podría analizar otros tipos de heurísticas para la búsqueda local. Finalmente, lo anterior es extensivo para otros problemas de *scheduling* cuyas características sean diferentes.

11 BIBLIOGRAFÍA

AKTURK, M., & ILHAN, T. (2011). Single CNC machine scheduling with controllable processing times to minimize total weighted tardiness. *Computers & Operational Research, 38*, 771-781.

ALIDAEE, B., & DRAGAN, I. (1997). A note on minimizing the weighted sum of taridy and early completion penalties in a single machine: A case of small common due date. *European Journal of Operational Research, 96*, 559-563.

ALTUNC, A., & KEHA, A. (2009). Interval-indexed formulation based heuristics for single machine total weighted tardiness problem. *Computers&Operations Research, 36*, 2122-2131.

ANGEL, E., & BAMPIS, E. (2005). A multi-start dynasearch algorithm for the time dependent single-machine total weighted tardiness scheduling problem. *European Journal of Operational Research, 162*, 281-289.

ANGHINOLFI, D., & PAOLUCCI, M. (2009). A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times. *European Journal of Operational Research, 193*, 73-85.

ARMENTANO, V., & ARAUJO, O. (2006). Grasp with memory-based mechanism for minimizing tardiness in single machine scheduling with setup times. *Journal of Heuristics, 12*(6), 427-446.

BILGE, Ü., KURTULAN, M., & KIRAÇ, F. (2007). A tabu search algorithm for the single machine total weighted tardiness problem. *European Journal of Operational Research, 176*, 1423-1435.

BOŻEJKO, W., GRABOWSKI, J., & WODECKI, M. (2006). Block approach/tabu search algorithm for the single machine total weighted tardiness problem. *Computers & Industrial Engineering, 50*, 1-14.

BRUCKER, P. (2007). *Scheduling Algorithms* (5 ed.). Berlín: Springer.

CABALLERO-VILLALOBOS, J. (2010). Conceptos Preliminares II. *Notas de Clase*.

CABALLERO-VILLALOBOS, J., & ALVARADO-VALENCIA, J. (2010). Greedy Randomized Adaptative Search Procedure (GRASP), una alternativa valiosa en la minimización de la tardanza total ponderadaa en una máquina. *Ingeniería y Universidad, 14*(2), 275-295.

CHANG, P., CHEN, S., & MANI, V. (2009). A hybrid algorithm with dominance properties for single machine scheduling with dependent penalties. *Applied Mathematical Modeling, 33*, 579-596.

CONGRAM, R. P. (s.f.). *OR-LIBRARY.* Recuperado el Noviembre de 2011, de http://people.brunel.ac.uk/~mastjjb/jeb/orlib/wtinfo.html

DELLA CROCE, F., DESMIER, E., & GARAIX, T. (2011). A note on "Beam search heuristics for the single machine early/tardy scheduling problem with no machine idle time". *Computers & Industrial Engineering, 60*, 183-186.

FIAT, A., & WOENGINGER, G. (1999). On-line scheduling on a single machine: minimizing the total completion time. *Acta Informatica, 36*, 287-293.

GAREY, M., & JOHNSON, D. (1979). *COMPUTERS AND INTRACTABILITY. A Guide to the Theory of NP-Completeness.* Nueva York: W.H. Freeman and Company.

GLOVER, F., & KOCHENBERG, G. (2003). *Handbook of Metaheuristics.* Dordrecht: Kluwer Academic Publishers.

GORDON, V., & TARASEVICH, A. (2009). A note: Common due date assignment for a single machine scheduling with the rate-modifyng activity. *Computers & Operations Research, 36*, 325-328.

GROSSO, A., DELLA CROCE, F., & TADEI, R. (2004). An enhanced dynasearch neighborhood for the single-machine total weighted tardiness scheduling problem. *Operations Research Letters, 32*, 68-72.

HERRMANN, J. (2011). The legacy of Taylor, Gantt, and Johnson: How to Improve Pproduction Scheduling. *Virtual Pro, 111*, 5.

HOOGEVEEN, H., POTTS, C., & WOEGINGER, G. (1999). On-line scheduling on a single machine: Maximizing the number of early jobs. *Operations Research Letters, 27*, 193-197.

HUYNH, N., & SOUKHAL, A. (2010). Due dates assigment and JIT scheduling with equal-size jobs. *European Journal of Operational Research, 205*, 280-289.

KAMINSKY, P., & SIMCHI-LEVI, S. (2001). Asymptotical analysis of an on-line algorithm for the single machine completion time problem with release dates. *Operations Research Letters, 29*, 141-148.

KELLERER, H., & STRUSEVICH, A. (2006). A fully polynomial approximation scheme for the single machine weighted total tardiness problem with a common due date. *Theoretical Computer Science, 369*, 230-238.

LEE, I., & SUNG, C. (2008). Minimizing due date related measures for a single machine scheduling problem with outsourcing allowed. *European Journal of Operational Research, 186*, 931-952.

LIAO, C., & JUAN, H. (2007). An ant colony optimization for single-machine tardiness scheduling with sequence-dependent setups. *Computers & Operational Research, 34*, 1899-1909.

MONTOYA-TORRES, J. (2003). Competitive analysis of a better on-line algorithm to minimize total completion time on a single-machine. *Journal of Global Optimization, 27*, 97-103.

NEAMMANEE, P., & REODECHA, M. (2009). A memetic algorithm-based heuristic for a scheduling problem in printed circuit board assembly. *Computers & Industrial Engineering, 56*, 294-305.

NIÑO, A., & CABALLERO, J. (2011). Evaluaciones de funciones de utilidad de GRASP en la programación de la producción para minimizar la tardanza total ponderada en una máquina. *Ciencia, Investigación, Academia, Desarrollo, 14 No. 2*, 51-58.

NONG, Q., YUAN, J., FU, R., LIN, L., & TIAN, J. (2008). The single-machine parallel-batching on-line scheduling problem with family jobs to minimize makespan. *International Journal of Production Economics, 111*, 435-440.

PATRAP, R. (2002). *Getting Started with Matlab. A Quick Introduction for Scientist an Engineers.* Neva York: Oxford University Press.

PINEDO, M. (2002). *Scheduling Theory, Algorithms and Systems* (2 ed ed.). New Jersey: Prentice Hall.

PINEDO, M. (2008). *Scheduling. Theory, Algorithms, and Systems* (Tercer Edición ed.). Nueva York: Springer.

RESENDE, M., & GONZÁLEZ, J. (2003). GRASP: Procedimientos de búsqueda miopes aleatorizados y adaptatiivos.

RESSENDE, M., & RIBEIRO, M. (2010). Greedy Randomize Adaptative Search Procedures: Advances, Hybridizations, and Applications. *Handbook of Metaheuristics*, 284.

SCHALLER, J., & VALENTE, J. (2011). Minimizing the weighted sum of squared tardiness on a single machine. *Computers & Operational Research, 39*, 919-918.

SEN, T., SULEK, J., & DILEEPAN, P. (2003). Static scheduling research to minimize weighted and unweighted tardiness: A state-of-art survey. *International Journal of Production Economics, 83*, 1-12.

SITTERS, R. (2010). Competitive analyisis of preemptive single-machine scheduling. *Operational Research Letters, 38*, 585-588.

STEE, R., & LA POUTRÉ, H. (2005). Minimizing the total completion time on-line on a single machine, using restarts. *Journal of Algorithms, 57*, 95-129.

TALBI, E. (2009). *METAHEURISTICS. From Design to Implementation.* New Jersey: Wiley.

TALBI, E. (2009). *METAHEURISTICS. From Design to Implementation.* New Jersey: Wiley.

TASGETIREN, M., PAN, Q., & LIANG, Y. (2009). A discrete diferential evolution algorithm for the single machine total weighted tardiness with sequence dependent setup times. *Computers & Operations Research, 36*, 1900-1915.

TIAN, J., FU, R., & YUAN, J. (2007). On-line scheduling with delivery time on a single batch machine. *Theoretical Computwer Science, 374*, 49-57.

TIAN, J., FU, R., & YUAN, J. (2008). A best on-line algorithm for the single machine scheduling with small delivery times. *Theoretical Computer Science, 393*, 287-293.

TIAN, J., FU, R., & YUAN, J. (2011). An on-line algorithm for the single machine unbounded parallel-batching scheduling with large delivery times. *Information Processing Letters, 111*, 1048-1053.

VALENTE, J., & ALVES, R. (2008). Beam search algorithms for the single machine total weighted tardiness scheduling problem with sequence-dependent setups. *Computers & Operations Research, 35*, 2388-2405.

VALLADA, E., & RUIZ, R. (2009). Genetic Algorithms with path relinking for the minimum tardiness permutation flowshop problem. *Omega*, 57-67.

VEGA-MEJÍA, C., & CABALLERO-VILLALOBOS, J. (2010). Uso combinado de GRASP y Path-Relinking en la programación de la producción para minimnizar la tardanza ponderada en una máquina. *Ingeniería y Universidad, 14*, 79-96.

WAN, G., & YEN, B. (2002). Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. *European Journal of Operational Research, 142*, 271-281.

WEAVER, P. (2006). A brief history of scheduling: Back to the future. *MYPROMAVERA 06*, 1-10. Camberra.

WIERS, V. (1997). *Human-computer interaction in production scheduling: Analysis and design of decision support systems for production scheduling tasks.* Tesis DoctoraL, Eindhoven.

YIN, Y., WU, C., WU, W., & CHENG, S. (2011). The single-machine total weighted tardioness scheduling problem with position-based learning effects. *Computers & Operations Research, 39*, 1109-1116.

YING, K., LIN, S., & HUANG, C. (2009). Sequencing single-machine tardiness problems with sequence dependent setup times using an iterated greedy heuristic. *Expert Systems with Applications, 36*, 7087-7092.

YVES, R. (2010). *Introduction to scheduling* (1 ed.). Boca Ratón: CRC Press.

ZHANG, L., & WIRTH, A. (2009). On-line scheduling of two parallel machines with a single server. *Computers & Operational Research, 36*, 1529-1553.

12 ANEXO 1: Resultados Computacionales

En las tablas de los archivos anexos, se puede encontrar la programación de las 375 instancias de los distintos trabajos para los diferentes valores de alfa. La forma en la que se leen es la siguiente:

Tabla 28: Lectura del Anexo 1

Trabajo Original	Valor de alfa	Número del trabajo del problema
	Valor de alfa	Lote de llegada
	Valor de alfa	Secuencia de la respuesta
	Valor de alfa	Función de Costo (WSPT)

Fuente: Presentación propia del autor

13 ANEXO 2: Medidas de interés

Las tablas dispuestas en el Anexo 2, muestran los valores medidos en cada una de las simulaciones que se hizo. En cada tabla, se puede ver ordenadamente las siguientes variables con su respectiva descripción para un mejor entendimiento a la hora de leer los datos.

Tabla 29: Descripción de las variables que se presentan en el Anexo 2

Medida	Descripción
Nombre de la Instancia	T(Número de Trabajos)I(Número del archivo)
Alfa	Valores entre 0 y 1
Iteraciones	Siempre 100
RDD	Valores entre 0.2 y 1
TF	Valores entre 0.2 y 1
Tardanza Total Ponderada	Función Objetivo
Mejor Solución	Obtenida del archivo de la instancia
Diferencia	Función Objetivo - Mejor Solución
Desviación RPD	Diferencia / Mejor Solución
Máximo valor estimado	Máximo valor de la función objetivo en las 100 iteraciones
Desviación RDI	Diferencia/(Máximo valor - Mejor Solución)
Valor Promedio de las Iteraciones	Suma de las respuestas /100
Tiempo de Procesamiento	Tiempo en el que se ejecuta la metaheurística
Número de Lotes	Valor aleatorio de MATLAB®
Promedio de Trabajos por lote	Total de trabajos de la instancia /Número de lotes

Fuente: Presentación propia del autor

14 ANEXO 3: Una mirada al problema *off-line*

Como se llegó a mencionar en el desarrollo, los puntos de comparación del problema adoptado y las variables respuesta son muy diferentes debido a la distribución en lotes. Sin embargo, para tener una mejor perspectiva de la operación del algoritmo, se realizó la simulación de los mismos datos suponiendo ahora que los trabajos llegan en un único lote. El resultado de este experimento y sus correspondientes medidas, se presentan en las tablas del archivo adjunto.

En éste, se puede ver que el algoritmo no es perfecto en el sentido de que en tan solo 5 ocasiones llega al mejor resultado obtenido para cada problema. No obstante, a diferencia del algoritmo *on-line*, las desviaciones se encuentran en un estrecho margen para la gran mayoría, lo que implica que, si bien el algoritmo tiene oportunidades de mejora, el procedimiento actual está bien encaminado y lo que se debe hacer es buscar métodos más efectivos para la operación de la fase constructiva y la búsqueda local. Para la primera, por ejemplo, se podría pensar en un parámetro alfa adaptativo al problema, mientras que para la segunda, se podría establecer una búsqueda más intensiva.

A pesar de ello, el algoritmo deja muy buenas señales de funcionamiento y se cree que la tarea se ha completado satisfactoriamente.

Haciendo un análisis comparativo entre los problemas *on-line* y *off-line* del mismo algoritmo, se pudieron encontrar resultados inesperados para la respuesta. Por ejemplo, se encontró que en ningún caso la tardanza total ponderada de los problemas fue la misma, como se esperaría. En contraste, hubo 47, 49 y 40 datos para 40, 50 y 100 trabajos, respectivamente, en los que la respuesta *on-line* fue mejor que la *off-line*. Si bien esto representa un pequeño porcentaje de la totalidad de las mediciones (menos del 2%) no deja de ser interesante que se haya podido alcanzar este resultado.

Para ilustrar mejor lo que sucedió, se tomaron estos valores y se buscó la relación con los distintos parámetros y se dibujaron los histogramas correspondientes. De ahí, se pudo concluir que el parámetro que muestra una mejor explicación de lo sucedido es el parámetro alfa dado que a un mayor valor, más datos tuvieron la característica encontrada. Igualmente, sucede con el valor RDD, aunque no es muy notoria la tendencia para los datos con 50 trabajos. Lo anterior, se atribuye en gran medida a que el valor de alfa implica la incorporación de más trabajos a la lista de seleccionables conforme esta aumenta y por lo tanto

Por su parte, los valores del parámetro TF, no parecieron haber influido en este hecho al notarse una distribución muy similar a lo largo de todos sus valores. Finalmente, el número de lotes se ve con tendencia hacia el valor de 5. Sin embargo, no se considera que esa tendencia tenga mucha relación con la desviación y, en lugar de ello, se piensa que se debe más a la distribución de trabajos en lotes.

Tabla 30: Resumen de respuestas obtenidas en el problema *on-line* con respecto al *off-line* con el mismo algoritmo

Mejores Respuestas en On-line			
Lotes	**40 Trabajos**	**50 Trabajos**	**100 Trabajos**
2	0	0	0
3	3	4	1
4	9	15	3
5	20	7	7
6	7	8	8
7	6	10	11
8	1	3	9
9	1	1	1
10	0	1	0
SUMA	**47**	**49**	**40**

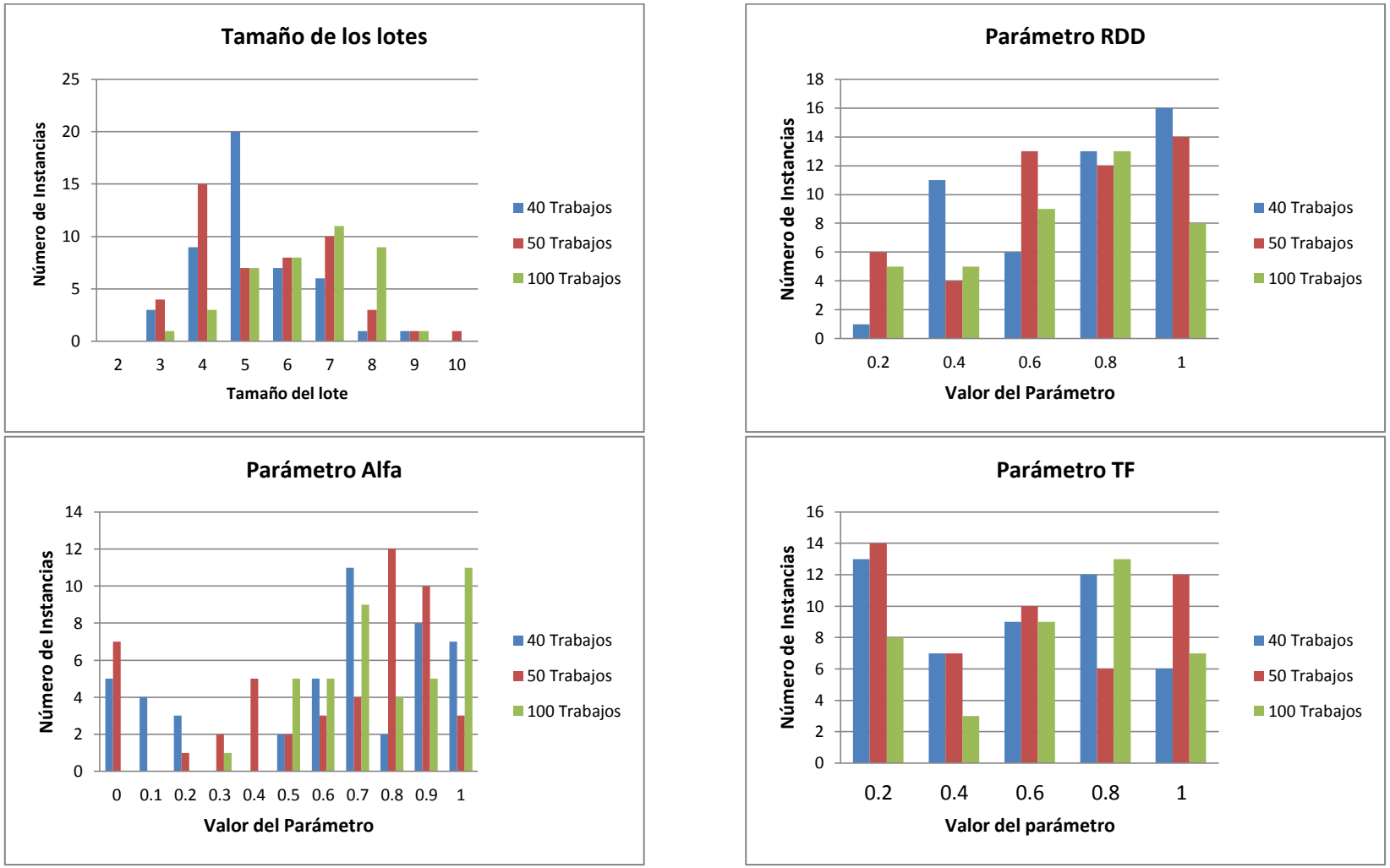
Mejores Respuestas en On-line			
Alfa	**40 Trabajos**	**50 Trabajos**	**100 Trabajos**
0	3	6	0
0.05	2	1	0
0.1	3	0	0
0.15	1	0	0
0.2	1	1	0
0.25	2	0	0
0.3	0	1	0
0.35	0	1	1
0.4	0	2	0
0.45	0	3	0
0.5	0	2	2
0.55	2	0	3
0.6	2	1	3
0.65	3	2	2
0.7	7	2	3
0.75	4	2	6
0.8	1	5	2
0.85	1	7	2
0.9	5	3	2
0.95	3	7	3
1	7	3	11
SUMA	**47**	**49**	**40**

Mejores Respuestas en On-line			
RDD	**40 Trabajos**	**50 Trabajos**	**100 Trabajos**
0.2	1	6	5
0.4	11	4	5
0.6	6	13	9
0.8	13	12	13
1	16	14	8
SUMA	**47**	**49**	**40**

Mejores Respuestas en On-line			
TF	**40 Trabajos**	**50 Trabajos**	**100 Trabajos**
0.2	13	14	8
0.4	7	7	3
0.6	9	10	9
0.8	12	6	13
1	6	12	7
SUMA	**47**	**49**	**40**

Fuente: Presentación propia del autor

Figura 19: Histogramas de las comparaciones del algoritmo desde las perspectivas *on-line* y *off-line*



Fuente: Presentación propia del autor

15 ANEXO 4: Prueba de Tamhane para la interacción de los parámetros RDD y TF.

En el archivo anexo, el lector se encontrará el resumen del resultado de la prueba de Tamhane desarrollado por el programa SPSS®. En éste, se hace una matriz cuyas columnas y filas están tituladas con las interacciones y en su interior, se han marcado con la letra 'X' aquellos pares que no muestran tener una diferencia significativa.

En la Tabla 20, se muestra un ejemplo de cómo debe hacerse la lectura de la tabla presentada en el anexo. Para ilustrar, se tiene que las interacciones 1-2 y 3-4, no presentan diferencias significativas mientras que la interacción 5 presenta diferencias con todas las demás.

Tabla 31: Ejemplo de lectura de la prueba

		Interacciones				
Interacciones		**1**	**2**	**3**	**4**	**5**
	1		X			
	2	X				
	3				X	
	4			X		
	5					

Fuente: Presentación propia del autor.

16 ANEXO 5: Función 'randi' de MATLAB

La función aleatoria del programa MATLAB® fue utilizada para determinar la manera en la que se distribuyen los trabajos por lotes. Además, a ese hecho se le atribuye en gran medida la diferencia de los datos frente al problema *off-line*. Por lo tanto, ha parecido importante agregar a este proyecto una breve descripción de cómo funciona la función aleatoria 'randi' para entender de mejor manera cómo fueron intervenidas las instancias originales.

La función 'randi' genera enteros de manera pseudoaleatoria de una distribución uniforme discreta. En general, la función r=randi(s,imax,n) regresa una matriz de tamaño n x n que contiene valores enteros entre 1 y n suponiendo que entre ellos existe una función de distribución uniforme.

Además, la secuencia de los números producidos por randi son determinados por el estado interno de la cadena aleatoria interna llamada s. De esta manera, 'randi' usa un valor uniforme a partir de s para generar el valor entero. Una vez que se reinicia la variable s para un estado fijo, se permite el reinicio del proceso.

Algunos ejemplos de uso de la función son:

```
>> randi(100,1)
ans =
91


>> randi(100, 2, 3)
ans =
   96    16    96
   97    98    49

>> randi([1,10],2,3)
ans =
    2     1     1
    8     3     1
```