ALGORITMO PARA LA OPTIMIZACIÓN DEL PROCESO DE SECUENCIACIÓN DE REPORTES

ALGORITHM FOR THE OPTIMIZATION OF THE REPORT SCHEDULING PROCESS

Jessica Coto Palacio[1, 2], Beatriz M. Méndez Hernández[2], Yailen Martínez Jiménez[2], Erick D. Rodríguez Bazan[3], Ann Nowé[4]

1 Delegación MINTUR Villa Clara, Cuba, jessica.coto@vcl.mintur.gob.cu, Maceo #453 e/ Caridad y Carretera Central, Santa Clara, Villa Clara

2 Universidad Central "Marta Abreu" de Las Villas, Cuba, {bmendez, yailenm}@uclv.edu.cu

3 International Centre for Theoretical Physics, Italia, erodra_g@ictp.it

4 Vrije Universiteit Brussel, Bélgica, ann.nowe@vub.ac.be

RESUMEN: *La secuenciación de trabajos es un área muy amplia en la cual muchos investigadores se han enfocado en los últimos años. En las empresas generalmente esta planificación se realiza de forma manual o semiautomática. Este trabajo propone un algoritmo para la secuenciación de trabajos en máquinas paralelas no relacionadas. El algoritmo utiliza dos variantes de solución: una heurística simple basada en una generación pseudoaleatoria y una regla de despacho basada en la máquina que más tiempo de procesamiento tiene pendiente. Para analizar el desempeño de las mismas se utiliza un caso de estudio donde los resultados obtenidos demuestran que la regla de despacho proporciona mejores resultados, conclusión que fue validada mediante pruebas estadísticas.*

Palabras Clave: Secuenciación de reportes, máquinas paralelas no relacionadas, heurísticas, reglas de despacho

ABSTRACT: *Scheduling is a wide research area in which many researchers have focused in recent years. In companies, this planning is usually done manually or semi-automatically. This work proposes an algorithm for the job sequencing in parallel unrelated machines. The algorithm uses two solution alternatives: a simple heuristic based on a pseudo-random generation and a dispatching rule based on the machine with most work remaining. To analyze the performance of the alternatives a study case is used, where the results obtained show that the dispatching rule provides better results, a conclusion that was validated using statistical tests.*

KeyWords: Report scheduling, unrelated parallel machines, heuristics, dispatching rules.

1. INTRODUCCIÓN

Los problemas de secuenciación están presentes en todas aquellas situaciones donde es necesario ejecutar un conjunto de tareas y para esto se requiere la asignación de las mismas a los recursos disponibles en intervalos de tiempo. En [1] se define la secuenciación de tareas como el proceso de seleccionar planes alternativos y asignar recursos y tiempos a un conjunto de actividades en un plan. Debido a esto, las asignaciones deben obedecer ciertas restricciones que reflejan las relaciones temporales entre las actividades y las limitaciones de los recursos compartidos. La mayoría de las investigaciones en el área de la secuenciación de

tareas se han enfocado en el desarrollo de procedimientos exactos para la generación de una solución base asumiendo que se tiene toda la información necesaria y un ambiente determinístico [2]. Este enfoque es conocido en la literatura como problemas de secuenciación de tareas en ambientes de tipo *offline.*

En ambientes *online*, por otra parte, una secuencia de trabajos $\sigma = J_1, J_2, \ldots, J_n$ tiene que procesarse en un número determinado de máquinas. Los trabajos arriban al sistema uno a uno, y cuando un nuevo trabajo llega tiene que ser inmediatamente despachado hacia una de las máquinas, sin tener conocimiento sobre trabajos futuros. El objetivo es optimizar una función objetivo determinada.

Los problemas de secuenciación pueden clasificarse teniendo en cuenta el ambiente de las máquinas, las características de los trabajos y la función objetivo. Esta clasificación se conoce comúnmente como $\alpha|\beta|\gamma$ [3], donde α representa el ambiente de las máquinas, β las características de los trabajos y γ el criterio a optimizar.

Según el ambiente de las máquinas el escenario más simple es en el que se cuenta con un solo recurso, pues los trabajos tienen una sola operación a ser procesada y solo existe una máquina que pueda ejecutarla. Cuando existen múltiples máquinas el entorno se torna más complicado, ya que pueden ser idénticas o pueden diferir en velocidad. Los posibles ambientes con máquinas paralelas se resumen de la siguiente forma [4]:

- Máquinas Paralelas Idénticas: Procesan los trabajos a la misma velocidad.
- Máquinas Paralelas Diferentes: El tiempo de procesamiento depende de la máquina.
- Máquinas Paralelas no relacionadas: El tiempo de procesamiento depende de la máquina y del trabajo.

El presente trabajo se centra en resolver un problema que se corresponde con un ambiente de máquinas paralelas no relacionadas. Esta primera propuesta de solución constituye una versión simplificada la cual cuenta con diferentes máquinas para procesar los reportes, cada una tiene características diferentes, y cada reporte puede ser procesado solo por una máquina en específico. Es decir, que al construirse una solución para este problema debe tenerse en cuenta que cada máquina tiene una cantidad de trabajos a procesar y el orden de los mismos debe determinarse teniendo en cuenta el objetivo a optimizar.

En este trabajo se proponen dos alternativas de solución: la primera basada en una heurística simple (pseudoaleatoria) y la segunda basada en una de las reglas de despacho clásicas existentes (Mayor tiempo de trabajo pendiente, *Most Work Remaining*).

El resto del artículo está estructurado de la siguiente manera: La sección 2 presenta una descripción general de los problemas de secuenciación en ambientes con máquinas paralelas no relacionadas y posibles métodos de solución, hace una descripción general del problema a resolver. En la sección 3 se presenta la propuesta de solución con un ejemplo de la salida del algoritmo, los resultados experimentales se muestran en la sección 4. El trabajo termina con conclusiones y posibles ideas de trabajo futuro.

2. SECUENCIACIÓN EN MÁQUINAS PARALELAS NO RELACIONADAS

Si se considera una situación en la cual se tienen *m* máquinas paralelas y *n* trabajos independientes con una sola operación cada uno de ellos, estamos en presencia de un ambiente de secuenciación con máquinas paralelas no relacionadas, ya que cada trabajo se puede procesar en una de las máquinas y el tiempo de procesamiento depende de esta combinación.

Las máquinas que tengan características similares pueden agruparse y un conjunto de trabajos se puede asignar a las mismas para ser procesados en aras de optimizar un objetivo determinado.

Son varios los enfoques que se han utilizado en la literatura para resolver problemas de tipo *NP-hard*, debido a su complejidad los intentos por obtener soluciones a través de algoritmos exactos pueden terminar en fallos para la mayoría de los juegos de datos [5]. De ahí que la mayoría de las investigaciones se centren en el desarrollo de heurísticas que puedan proporcionar buenas soluciones en un tiempo razonable.

Entre las heurísticas utilizadas uno de los enfoques más sencillos son las llamadas reglas de despacho. Una regla de despacho es una estrategia de secuenciación en el cual una prioridad es asignada a cada trabajo que espera a ser ejecutado en una máquina específica. Siempre que una máquina está disponible, una regla de despacho basado en prioridad examina los trabajos en espera y selecciona el de prioridad más alta pera ser procesado a continuación [6].

Algunas de las reglas de despacho más utilizadas son:

- Menor tiempo de Procesamiento (*Shortest Processing Time*, SPT): La prioridad más alta se le da a la operación en espera con el menor tiempo de procesamiento.
- Primero en entrar primero en salir (*First In First Out*, FIFO): La operación que arriba primero a la cola recibe la prioridad más alta.
- Mayor tiempo de procesamiento restante (*Most*

Work Remaining, MWKR): La prioridad más alta se le da a la operación que pertenece al trabajo con el mayor tiempo de procesamiento restante.

- Fecha de terminación más próxima (*Earliest Due Date*, EDD): El trabajo con la fecha de terminación más próxima es procesado primero.

En [7] los autores proponen un algoritmo de optimización y uno de aproximación que se basan en la dualidad y la relajación para resolver este problema *NP-hard*, la idea por detrás de la relajación es reemplazar un conjunto de restricciones complejas por una sola restricción que es una agregación ponderada de estas restricciones.

En [8] los autores aplican metaheurísticas en la solución del problema de secuenciación para minimizar el *makespan* o tiempo de completamiento de las tareas, reportando que el algoritmo genético no obtiene buenos resultados por sí solo, pero cuando se le incorpora un método de descenso entonces los resultados son comparables con los del recocido simulado.

Francis Sourd propone en [9] dos algoritmos de aproximación para el problema en cuestión, el primero se basa en una exploración parcial y heurística de un árbol de búsqueda. En el segundo se implementa un procedimiento de mejora de vecindarios grandes en un algoritmo ya existente. En este trabajo se reporta que la eficiencia computacional de los algoritmos es equivalente a la mejor heurística de búsqueda local, similar trabajo se propone en [10]. En algunos trabajos se introduce el uso de reglas de despacho en ambientes *online*, por ejemplo en [11] los autores enfocan el problema en presencia de trabajos con tiempo de procesamiento estocástico y cuya entrada al sistema ocurre de forma aleatoria siguiendo distribuciones específicas.

También existen enfoques que dividen el proceso de solución en dos pasos o etapas, por ejemplo en [12-16] se presentan propuestas de solución que siguen esta idea y van combinando desde heurísticas simples hasta *beam-search* [13,14]. Otros utilizan enfoques más complejos, por ejemplo, algoritmos genéticos, enumeración parcial y planos cortantes [15-18], otros utilizan búsqueda local y aprendizaje [19, 20], algunos más avanzados llegan a utilizar *spectral graph partitioning* [21] y minería de datos [22], es importante mencionar que en muchos de estos casos las funciones de evaluación o de calidad se basan en ideas de las mencionadas reglas de despacho.

2.1 Descripción del problema

En los problemas de secuenciación de tareas de la vida cotidiana se procesan trabajos siguiendo un determinado orden, de forma tal que se logre optimizar los tiempos en los que transitan por el sistema. La ejecución de los reportes generados por los clientes de los supermercados a través de compras por diferentes vías, es un problema de secuenciación de tareas. Muchos de los reportes deben cumplir una serie de condiciones y el sistema debe ser capaz de priorizar su ejecución en caso de ser necesario.

Existe una cantidad de *m* tipos de máquinas, donde cada tipo de máquina cuenta con un número determinado de recursos. También se tienen *n* reportes en lotes, agrupados por el tipo de reporte, y que deben ser procesados por un tipo de máquina. Estos reportes cuentan además con un ID, un tiempo medio de procesamiento, una cantidad determinada de pedidos y un tiempo total de procesamiento que está dado por la cantidad de pedidos. Los reportes pueden ser procesados al mismo tiempo, ellos son independientes, siempre que haya un espacio disponible. Si las máquinas están ocupadas, los reportes deben esperar que exista la disponibilidad. Existe un recurso que no presenta límite de capacidad, solo el límite que impone el sistema para no sobrecargarse. En este recurso se ejecutan los reportes que no necesitan ser ejecutados por una máquina determinada. Entre los recursos no existe ningún tipo de relación.

La Tabla I muestra un pequeño ejemplo donde se puede apreciar el formato de los datos que necesitan ser procesados.

Tabla I: Formato de los datos

Trabajo	Tiempo total	Número de ejecuciones	Tiempo por reporte	Máq.
T1	2227,71	1	2227,715	7
T2	1368,48	1	1368,487	7
T3	784,220	1	784,220	2
T4	319,311	1	319,311	1
T5	300,079	1	300,079	1
T6	292,257	1	292,257	2
T7	95,557	1	95,557	1

Los reportes son ejecutados a determinadas horas del día, lo que provoca que puedan ocurrir llegadas de lotes inesperados al sistema, los cuales deben ser integrados y procesados como cualquier otro con sus características peculiares.

3. PROPUESTA DE SOLUCIÓN

Como se explicó anteriormente, el problema de secuenciar *n* trabajos independientes compuestos por una sola operación en *m* máquinas paralelas no relacionadas con el objetivo de minimizar el tiempo de completamiento de todos los trabajos es un problema combinatorio. Por lo tanto, el tiempo necesa-

rio para obtener la solución óptima utilizando ya sea un modelo matemático, una técnica de enumeración completa o una técnica de ramas y cotas crecerá exponencialmente con respecto al aumento del tamaño del problema. Es por esto que el uso de una heurística para superar esta situación es altamente inevitable.

Para dar solución al problema descrito en la sección 2.1 se proponen dos alternativas. La primera consiste en la generación de una secuencia de números los cuales deben cumplir con una serie de condiciones: 1) la suma de los números la primera vez que se generan es el límite que impone el sistema, 2) cada número se corresponde con un tipo de máquina y no es mayor que la cantidad de máquinas existentes del tipo correspondiente ni de la cantidad de reportes que se encuentran en la cola de espera y 3) tiene en cuenta para las generaciones diferentes de la primera la cantidad de reportes que se encuentran dentro del sistema en el momento de generar los números.

En la segunda variante se utiliza la regla de despacho, MWKR, y se genera una sucesión de números que cumplen las tres condiciones enumeradas anteriormente y que, además, da prioridad a la máquina que mayor tiempo de procesamiento pendiente tenga. A continuación, se muestra el pseudocódigo del algoritmo propuesto.

```
Input: PriorityQueue reports, machines
Output: Reports Assignment (Λ)
int [] a = distribution();
repeat
    for i = 1 to machines.size() do
        for j = 1 to a[i] do
            machines[i].pool[j] = reports[i].pop()
        end for
    end for
until reports.isEmpty()
return Λ
```

Figura. 1: Pseudocódigo del algoritmo propuesto

La entrada del algoritmo es un documento como el que se muestra en la Tabla I. Estos datos son leídos y almacenados en diferentes colas de prioridad en dependencia del recurso que se necesite usar. Existe una cola de prioridad por cada tipo de recurso. Las colas de prioridad están ordenadas por el tiempo medio de procesamiento de un reporte (de menor a mayor). El método *distribution()* genera una secuencia de números enteros cuya longitud es igual a la cantidad de tipos de máquinas con que cuenta el problema, esta secuencia es generada de acuerdo a una de las dos variantes explicadas previamente.

La idea general del algoritmo es ir procesando los reportes que se encuentran esperando en las colas de prioridad de cada máquina. Mientras haya reportes que procesar se van introduciendo en las máquinas de acuerdo a la cantidad que genera el método *distribution*() usando la técnica pseudoaleatoria o la regla de despacho.

El algoritmo devuelve la mejor secuenciación encontrada después de procesar todos los reportes. El objetivo a optimizar en este caso fue el *makespan*. Una posible solución del problema sería la presentada en la Figura 2, donde mediante un diagrama de Gantt se muestra como quedarían secuenciados los reportes en las diferentes máquinas. El objetivo en este caso es minimizar el tiempo de completamiento de las tareas o *makespan*. Este resultado se corresponde con los datos mostrados en la Tabla I del epígrafe anterior.

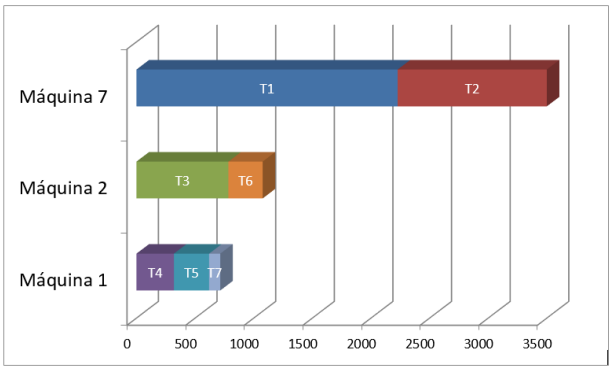

Figura. 2: Diagrama de Gantt para el ejemplo presentado en la Tabla 2.

4. RESULTADOS EXPERIMENTALES

En esta sección se describen los principales resultados obtenidos al evaluar el comportamiento del algoritmo con las dos variantes o alternativas propuestas, para dicha evaluación se hizo uso del caso de estudio que se describe.

4.1 Caso de estudio

El sistema impone un máximo de 130 trabajos que pueden ser ejecutados a la vez. Se cuenta con 10 tipos de máquinas las cuales tienen los siguientes límites de trabajos a procesar al mismo tiempo: máquina 1 (4), máquina 2 (10), máquina 3 (15), máquina 4 (60), máquina 5 (10), máquina 6 (2), máquina 7 (6), máquina 8 (20), máquina 9 (1), máquina 10 (130).

La décima máquina es la encargada de procesar los reportes que no tienen ningún recurso asignado por eso su límite de capacidad es el mismo del sistema.

Los reportes están organizados de menor a mayor de acuerdo al tiempo de procesamiento. La selección se realiza para cada máquina en el mismo

instante de tiempo, teniendo en cuenta la disponibilidad de recursos y la cantidad de recursos que esperan en cola. En caso de no existir capacidad suficiente en las máquinas para ejecutar en su totalidad un determinado lote de reportes, se pasará a procesar la cantidad admitida por la máquina y el resto se mantendrá en espera hasta la existencia de disponibilidad. En una misma máquina se pueden ejecutar trabajos distintos, debido a que la capacidad de un recurso puede ser mayor que el número de ejecuciones de determinados reportes.

Tabla III: Datos del caso de estudio a analizar

Trabajo	Tiempo total	Número de ejecuciones	Tiempo por reporte	Máq
T1	39449,1	30	1314,97	7
T2	43791,68	32	1368,49	7
T3	15904,71	9	1767,19	
T4	90255,72	51	1769,72	
T5	107204,4	60	1786,74	
T6	7335,9	30	244,53	2
T7	10214,82	42	243,21	2
T8	25,75	1	25,75	1
T9	2008,8	40	50,22	1
T10	77,46	3	25,82	3
T11	677,88	12	56,49	3

En la Tabla II se muestran un conjunto de 11 trabajos o reportes que deben ser procesados en las máquinas correspondientes de acuerdo a las características de cada uno. En un primer instante se generan la cantidad de reportes que van a ser procesados por cada máquina, teniendo en cuenta no sobrepasar la cantidad admitida en cada máquina ni tampoco los 130 del sistema.

En una primera iteración del algoritmo, la posible combinación de números podría ser: M1 - 3, M2 - 8, M3 - 13, M7 - 6 y M10 - 100 trabajos. Luego se selecciona por máquina el primer trabajo en cola y se resta el número de ejecuciones a la cantidad que será procesada en la máquina en ese instante. En caso de existir más disponibilidad en la máquina que el número de ejecuciones, por ejemplo en la máquina 1, se toma la totalidad del primer trabajo en cola y una parte del segundo, quedando en cola para futuras iteraciones. La Figura 3 muestra de forma visual como quedaría esta asignación. Las máquinas 4, 5, 6, 8 y 9 quedan vacías al no ser requeridas por los trabajos del caso de estudio.

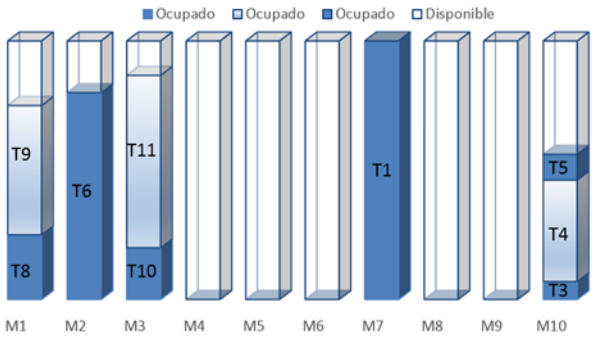

Figura 3. Ejemplo de asignación en las máquinas

Luego de realizar esta asignación inicial, los trabajos que permanecen en cola esperando por la disponibilidad de las máquinas son los siguientes:

Tabla IIII: Trabajos que permanecen en cola

Trabajo	Tiempo total	Número de ejecuciones	Tiempo por reporte	Máq
T1	35504,19	27	1314,97	7
T2	43791,68	32	1368,49	7
T5	83976,78	47	1786,74	
T6	5379,66	22	244,53	2
T7	10214,82	42	243,21	2
T9	1908,36	38	50,22	1
T11	112,98	2	56,49	3

Cada vez que una máquina termina de procesar y vuelve a tener capacidad disponible, el algoritmo genera una combinación de números o asignaciones teniendo en cuenta todos los *slots* o capacidades que están libres en todo el sistema, y se procede a distribuir en las mismas los reportes que esperan en las colas correspondientes, este proceso se repite hasta que todos los reportes hayan sido ejecutados. El algoritmo da como salida en que máquina y en que intervalo de tiempo fue procesado cada reporte, como se muestra en la Figura 4.

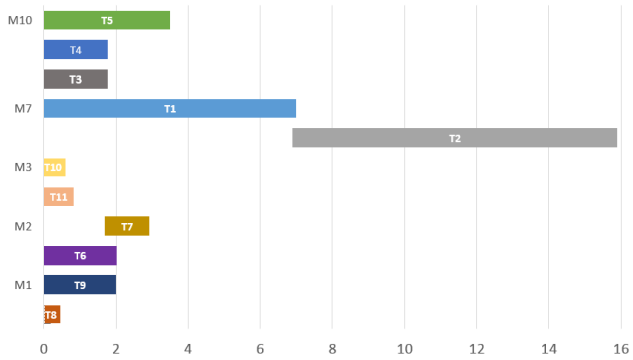

Fig. 4. Ejemplo de solución usando la alternativa *Random. Makespan=15.13*

En la imagen se puede ver que hay múltiples trabajos ejecutándose de forma simultánea en algunos instantes de tiempo en las máquinas, esto es debido a que las mismas tienen múltiples *slots* que pueden ejecutarlos en paralelo.

En este caso los valores en el gráfico fueron divididos por mil para una mejor visualización de los tiempos de completamiento y las asignaciones.

Como se puede observar, la máquina 7 es la que más tiempo demora en terminar los reportes que debe procesar, es necesario recordar que hasta aquí las combinaciones de números aleatorios que rigen la cantidad máxima de reportes que puede procesar cada máquina se han generado sin seguir una heurística específica, es decir, de forma aleatoria.

A continuación, se muestra un ejemplo utilizando la segunda alternativa de solución (la regla de despacho MKWR explicada en la sección 3). Una posible salida del algoritmo cuando se utiliza esta alternativa sería la mostrada en la Figura 5, como se puede apreciar el resultado es de 14.78, lo cual mejora el resultado de la estrategia aleatoria cuyo valor de *makespan* fue de 15.13.

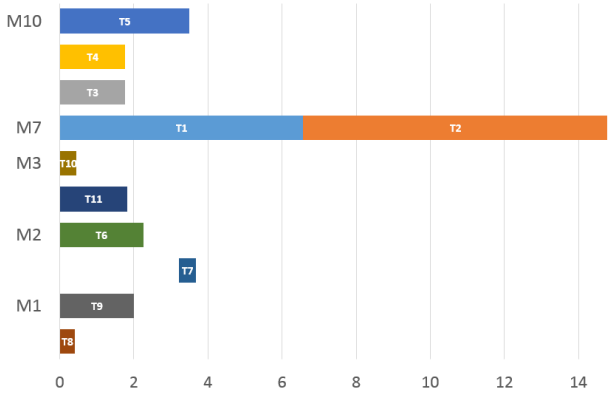

Fig. 5. Ejemplo de solución usando la alternativa *MWKR. Makespan=14.78*

Para comparar el desempeño de ambas alternativas se realizaron diez corridas de cada una de ellas sobre el mismo juego de datos, obteniéndose como resultado el *makespan*, que en este caso no es más que el tiempo en segundos que toma secuenciar todos los reportes. La tabla IV muestra un resumen de los resultados obtenidos.

Tabla IVV: Resultados obtenidos por las alternativas propuestas al resolver el caso de estudio

Ejecuciones	Aleatorio	MWKR
1	19082,27	14785,79
2	15288,23	14890,65
3	16626,05	14961,45
4	16682,54	14852,48
5	19292,49	14995,32
6	14899,84	14785,79
7	15051,28	14854,98
8	15238,01	14890,65
9	16769,60	14975,54
10	14913,46	14890,79

Aunque a simple vista se puede apreciar que existen diferencias entre los resultados obtenidos por las alternativas, se aplica la prueba de Wilcoxon para analizar si las mismas son significativas. El resultado se muestra en la Figura 6.

Rangos

		N	Rango promedio	Suma de rangos
MWKR - Random	Rangos negativos	10[a]	5,50	55,00
	Rangos positivos	0[b]	,00	,00
	Empates	0[c]		
	Total	10		

a. MWKR < Random

b. MWKR > Random

c. MWKR = Random

Estadísticos de contraste[b]

	MWKR - Random
Z	-2,803[a]
Sig. asintót. (bilateral)	,005

a. Basado en los rangos positivos.

b. Prueba de los rangos con signo de Wilcoxon

Figura 6. Resultados de las pruebas estadísticas

Como se puede apreciar, se obtiene una significación de 0.005, la cual es menor que 0.05 por tanto se rechaza la hipótesis nula y se acepta la alternati-

va, lo cual significa que existen diferencias significativas entre los resultados de ambas alternativas.

Mirando los rangos se puede apreciar que en los 10 casos la heurística MWKR obtiene mejores resultados que la alternativa aleatoria.

5. CONCLUSIONES Y TRABAJO FUTURO

En el presente trabajo se implementó un algoritmo para optimizar un problema de secuenciación de tareas en máquinas paralelas no relacionadas. Se utilizaron dos estrategias para elegir la cantidad de trabajos a procesar por cada tipo de máquina en un instante de tiempo determinado: una variante aleatoria y otra mediante el uso de la regla de despacho *MWKR*.

Se realizaron 10 corridas para cada una de las estrategias y los resultados obtenidos fueron validados mediante pruebas estadísticas, concluyendo que la alternativa utilizando la regla de despacho es capaz de obtener resultados que superan significativamente los obtenidos por la estrategia aleatoria, al ser todos los rasgos de la prueba de Wilcoxon negativos se puede comprobar la superioridad de la misma.

Se propone como trabajo futuro aplicar el algoritmo propuesto a juegos de datos con mayor número de reportes para analizar su desempeño y escalabilidad. Además, sería interesante añadir un algoritmo de aprendizaje reforzado que obtenga las mejores combinaciones de asignaciones para optimizar el tiempo de completamiento de las tareas.

6. REFERENCIAS BIBLIOGRÁFICAS

1. **Zhang W.** Reinforcement Learning for Job Shop Scheduling. Architecture. Oregon State University; p. 190; 1996.

2. **Herroelen W, Leus R.** Project scheduling under uncertainty: Survey and research potentials. Eur. J. Oper. Res. 165(2):289–306; 2005.

3. **Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG**. Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann. Discret. Math.5:287–326; 1979.

4. **Martínez Jiménez Y**. A Generic Multi-Agent Reinforcement Learning Approach for Scheduling Problems. PhD, Vrije Universiteit Brussel. p. 128; 2012.

5. Sivasankaran P, Sornakumar T, Panneerselvam R. Efficient Heuristic to Minimize Makespan in Single Machine Scheduling Problem with Unrelated Parallel Machines. Intelligent Information Management. March 2010:188–98; 2010.

6. Nhu Binh HO, Joc Cing TAY. Evolving Dispatching Rules for solving the Flexible Job-Shop Problem. IEEE Congr. Evol. Comput. p. 2848–55. 2005.

7. Van De Velde S. L. Duality-based algorithms for scheduling unrelated parallel machines. ORSA Journal of Computing. Vol 5:182–205; 1993.

8. Glass C. A., Potts C. N., Shade P. Unrelated parallel machine scheduling using local search. Mathematical and Computer Modeling. Vol 20:41–52; 1994.

9. Sourd F. Scheduling tasks on unrelated machines: Large neighbourhood improvement procedures. Journal of Heuristics. Vol 7:519–31; 2001.

10. Alharkan I.M. Algorithms for Sequencing and Scheduling. Industrial Engineering Department, King Saud University, 2010.

11. Al-Turki U, Andijani A, Arifulsalam S. A New Dispatching Rule for the Stochastic Single-Machine Scheduling Problem. SIMULATION: Transactions of The Society for Modeling and Simulation International. Vol 80:165–70; 2014.

12. Monien B., Woclaw A. Scheduling unrelated parallel machines computational results. Experimental Algorithms, Springer, Berlin / Heidelberg. Vol 4007:195–206; 2006.

13. Gairing M, Monien B, Woclaw A. A faster combinatorial approximation algorithm for scheduling unrelated machines. Theoretical Computer Science. Vol 380:87–99; 2007.

14. Ghirardi M, Potts C.N.: Makespan minimization for scheduling unrelated parallel machines: A recovering beam search approach. European Journal of Operations Research. Vol 165:457–67; 2005.

15. Hariri A. M. A., Potts C. N.: Heuristics for scheduling unrelated parallel machines. Computer and Operations Research. Vol 18:323–31; 1991.

16. Mokotoff E, Chretienne P. A cutting plane algorithm for the unrelated parallel machine scheduling problem. European Journal of Operational Research. Vol 141:515–25; 2002.

17. Mokotoff E, Jimeno J. L.: Heuristics based on partial enumeration for the unrelated parallel processor scheduling problem. Annals of Operations Research. Vol 117:133–50; 2002.

18. Murata T, Gen M. Performance Evaluation of Solution-Based GA and Rule-Based GA for Scheduling Problems. Annals of Operations Research. 2000.

19. Piersman N, Van Dijk W. A local search heuristic for unrelated parallel machine scheduling with efficient neighbourhood search. Mathematical and Computer Modeling. Vol 24:11–9; 1996.

20. Shahzad A, Mebarki N. Learning Dispatching Rules for Scheduling: A Synergistic View Comprising Decision Trees, Tabu Search and Simulation. Computers [Internet]. Available from: www.mdpi.com/journal/computers; 2016.

21. Šori K, Vojvodić Rosenzweig V. SGP heuristics for one machine scheduling problem. Proceedings of 7th International Symposium on Operations Research in Slovenia (SOR 2003) p:1–6, 2003.

22. Zahmani M.H, Atmani B., Bekrar A. Efficient dispatching rules based on data mining for the single machine scheduling problem. Computer Science & Information Technology (CS & IT). 199–208; 2015.

7. SÍNTESIS CURRICULARES DE LOS AUTORES

Jessica Coto Palacio nació en Santa Clara, el 2 de mayo de 1992, entre 2010 y 2015 estudió Ingeniería Informática en la Universidad Central "Marta Abreu" de Las Villas, donde desarrolló su investigación y tesis de pregrado en temas relacionados con la Inteligencia Artificial y problemas de optimización. En julio del año 2015 se gradúa de la mencionada carrera y comienza a trabajar en la Delegación del MINTUR de Villa Clara. Actualmente se encuentra desarrollando su tesis de maestría en colaboración con el grupo de investigación de Inteligencia Artificial de la Universidad Central de Las Villas, específicamente en la solución de problemas de secuenciación de tareas a través de la aplicación de técnicas basadas en aprendizaje reforzado. Es coautora de varias publicaciones en revistas y memorias de eventos nacionales e internacionales. Tiene un registro de software, y es coautora de un premio CITMA provincial.