Evaluación de Variantes de la Metaheurística VNS para el Problema de Planificación de Máquinas Paralelas

Claudia Ruth Gatica, Silvia Marta Molina y Guillermo Leguizamón
LIDIC, Universidad Nacional de San Luis
Ejército de Los Andes 950 - Local 106, San Luis, Argentina,
{crgatica, smolina, legui}@unsl.edu.ar

Resumen. VNS (*Variable Neighborhood Search*) es una metaheurística de trayectoria y usa diferentes estructuras de vecindarios siguiendo algún criterio pre-establecido para realizar la búsqueda. En este trabajo se proponen variantes de VNS estándar (o simplemente VNS) para mejorar su desempeño introduciendo cambios en las secuencias de vecindarios utilizadas y/o mecanismos de exploración considerando el problema de Planificación de Máquinas Paralelas. Las variantes propuestas son: VNS+R (VNS *Random*) con selección de vecindario aleatoria; VNS+LHS (VNS *Latin Hypercube Sample*) con preselección de vecindarios a través de Cuadrados Latinos; VNS+E (VNS *Exploratory*) que intensifica la exploración del espacio de búsqueda y por último, VNS+ER (VNS *Exploratory&Random*) que combina aspectos funcionales de VNS+R y VNS+E. Los resultados muestran que las variantes que intensifican la exploración en el espacio de búsqueda con selección aleatoria de estructuras de vecindario, mejoran al desempeño de VNS, variante representada por el algoritmo VNS+ER.

Palabras claves: Planificación de Máquinas Paralelas, Tardanza Máxima, Búsqueda en Vecindarios Variables, Metaheurísticas.

1 Introducción

Las metaheurísticas de trayectoria (S-metaheurísticas, de aquí en adelante) son algoritmos de búsqueda usados para resolver problemas NP-duros tal como son los problemas de planificación. En particular VNS (Mladenović y Hansen [6]) realiza una búsqueda sistemática de estructuras de vecindarios, en donde la secuencia de exploración y tamaño de cada vecindario es crucial en el diseño del algoritmo [5].

La planificación (*scheduling*) de actividades es un proceso de toma de decisión que tiene un papel importante en los sistemas de producción y multiprocesadores, en los entornos de fabricación y distribución de información, y de transporte. En particular, este artículo considera el problema de planificación o *scheduling* de máquinas paralelas idénticas sin restricciones con el objetivo de minimizar la Tardanza Máxima

(*Maximum Tardiness*). El entorno de máquinas paralelas se ha estudiado durante varios años debido a su importancia tanto a nivel académico como a nivel industrial.

En una revisión bibliográfica se observa que la metaheurística VNS ha sido aplicada a diversos problemas académicos y del mundo real, aplicada sola o en forma híbrida en combinación con otras metaheurísticas. En [2] se presenta un VNS para proveer las soluciones iniciales a otras metaheurísticas implementadas para resolver el problema de planificación de procesos dinámicos y asignación de fechas de vencimiento (DIPPSDDA), en donde la función objetivo fue minimizar la puntualidad y la tardanza (E/T). En [4] se aplica al problema de planificar un conjunto de trabajos independientes, con tiempos de configuración dependientes de la secuencia y restricciones de compatibilidad de tareas, en un conjunto de máquinas paralelas, con el objetivo de minimizar el tiempo máximo de finalización (*makespan*). En [9] se aplica un algoritmo híbrido VNS-GSA (VNS y el *Algoritmo de búsqueda gravitacional*) para la planificación de máquina única y de máquinas paralelas, los objetivos de minimizar el *maximum earliness* y el número de tareas (*jobs*) tardías con efecto de aprendizaje basado en la posición (donde el tiempo real de procesamiento del trabajo es una función de su posición) y los tiempos de procesamientos dependientes de la configuración. En [17] se presenta VNS para el problema de planificación de máquinas paralelas idénticas, con el objetivo de minimizar el tiempo total de finalización. Para la configuración de parámetros, en [3] se presenta un VNS para el diseño de los parámetros de los controladores de amortiguamiento en sistemas de potencia multi-máquina: tal como estabilizadores del sistema de potencia (PSS) y el controlador de flujo de potencia interlínea-amortiguación de oscilación de potencia (IPFC-POD). Para el problema del mundo real llamado despacho de energía en redes distribuidas inteligentes se propone en [14] el algoritmo híbrido VNS-DEEPSO (*Differential Evolutionary Particle Swarm*), para obtener soluciones para minimizar los costos operacionales y maximizar los ingresos de la red inteligente. Este algoritmo fue diseñado con algunas mejoras que permiten la evaluación de ecuaciones formadas a partir de ecuaciones algebraicas no lineales y ecuaciones diferenciales en la que es imposible obtener su derivada con un modelo matemático exacto [3].

En el presente trabajo se presentan y analizan diferentes variantes del algoritmo VNS. Dichas variantes (VNS+R, VNS+LHS, VNS+E y VNS+ER) surgen a partir de cambios en el esquema básico de búsqueda y selección de vecindarios, con el fin de mejorar su desempeño sobre el problema minimización de la tardanza máxima.

La organización del presente trabajo es la siguiente. En la sección 2 se describe y formula el problema de planificación de máquinas paralelas. En la sección 3 se describen las variantes propuestas de VNS. En la sección 4 se detalla el diseño de los experimentos. En la sección 5 se muestran y analizan los resultados obtenidos. Finalmente, en la sección 6 se presentan las conclusiones.

2 EL Problema de Planificación de Máquinas Paralelas

El problema de planificación (*scheduling*) de máquina paralelas sin restricciones es un problema común en los sistemas reales de manufacturación y producción, y es también un problema de interés desde el punto de vista teórico y práctico.

En la literatura, el problema estudiado se denota como: $P_m||T_{max}$. El primer campo describe el ambiente de las máquinas P_m, el segundo contiene las restricciones, aquí se puede notar que el problema no tiene restricciones, por lo tanto, el campo está vacío, y el tercero provee la función objetivo a ser minimizada T_{max} [12], [16].
Este problema de planificación se puede expresar de la siguiente manera: existen *n* tareas para ser procesadas sin interrupción en algunas de las *m* máquinas idénticas que pertenecen al sistema P_m; cada máquina no puede procesar más de una tarea a la vez. La tarea t_j, (j=1, 2, 3, …, *n*) está disponible en el tiempo cero. La misma requiere un tiempo de procesamiento p_j positivo e ininterrumpido sobre una máquina y también tiene una fecha de vencimiento d_j en la cual su procesamiento debería estar finalizado. Para una orden (*schedule*) de procesamiento de tareas dada, el tiempo de completitud más temprano C_j y el tiempo máximo de tardanza $T_j = \{C_j - d_j, 0\}$, son fácilmente calculados. El problema es encontrar una orden (*schedule*) óptima que minimice el valor de la función objetivo de la ecuación:

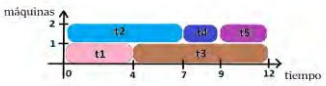
$$\textit{Maximum Tardiness}: T_{max} = max_j (T_j) \qquad (1)$$

Este problema es considerado NP-duro cuando $2 \leq m \leq n$ [12], [16]. Una instancia del problema con $n=5$ tareas y $m=2$ máquinas es mostrada en el Ejemplo 1.

Ejemplo 1: Instancia con n=5 tareas t_j, m=2, con tiempos de procesamiento p_j y fechas de vencimientos d_j; para j=1, …, 5.

t_1	t_2	t_3	t_4	t_5	t_j
(4, 15)	(7, 20)	(8,19)	(2, 6)	(3, 8)	(p_j, d_j)

Figura 1: Diagrama de Gantt de una posible solución para el Ejemplo 1.



El diagrama de Gantt de la Figura 1 describe una planificación **S** (*por schedule*) de tareas en dos máquinas m_1 y m_2, en donde S= [{m_1, (t_1, p_1, d_1), (t_3, p_3, d_3)}, {m_2, (t_2, p_2, d_2), (t_4, p_4, d_4), (t_5, p_5, d_5)}] y la distribución de las tareas= {t_1, t_2, t_3, t_4, t_5} es: t_1 y t_3 se ejecutan en m_1 y t_2, t_4 y t_5 en m_2. Por consiguiente, los tiempos de completitud respectivos son C_j=((t_1,4), (t_2,7), (t_3,12), (t_4, 9), (t_5, 12)) y los tiempos de tardanza computados en cada tarea T_j= (0, 0, 0, 3, 4), donde la tardanza máxima es T_{max}= 4.

3 La S-metaheurística VNS y las variantes propuestas

La S-metaheurísticas han mostrado su eficacia para abordar varios problemas de optimización en diferentes dominios [2], [3], [9], [14], entre otros. Éstas realizan una exploración a través de estructuras de vecindarios [5] mediante un procedimiento iterativo a partir de una solución inicial, generalmente aleatoria, y aplicando operadores que definen la trayectoria de búsqueda.

VNS considera de manera secuencial un conjunto de distintas estructuras de vecindarios N_i donde i=1, 2, 3, …, k_{max}, son definidas en el espacio de soluciones del problema. La secuencia previa es realizada en forma sistemática o aleatoria a fin de obtener soluciones de alta calidad a los efectos de escapar de los óptimos locales. Una estructura de vecindario N en el espacio de soluciones X de una solución dada x, es denotada como $N(x) \subset X$, en donde $N: X \rightarrow \wp(X)$, siendo $\wp(X)$ el conjunto potencia de x [15].

En nuestro trabajo, se entiende como secuencia de estructuras de vecindarios al orden de inspección de cada una de las estructuras de vecindarios N_i definidas como la entrada del algoritmo VNS. Un aspecto interesante de VNS es que presenta un esquema básico que sirve como marco flexible para implementar variantes heurísticas [15] y además demuestran un buen desempeño en resolver problemas complejos.

3.1 Algoritmo estándar VNS

VNS (`Algoritmo 1`) recibe como entrada un conjunto de estructuras de vecindarios definidas de antemano N_i, i=1, …, k_{max}. Cada iteración del algoritmo está compuesta de tres pasos: (1) sacudida (*shaking*), (2) búsqueda local y (3) moverse (*move*). La solución inicial S_0 es sacudida en el vecindario actual N_k, una búsqueda local (`Algoritmo 2`) es aplicada sobre la solución S_1 (se obtiene S_1 mediante el operador de perturbación N_k) para obtener una solución candidata S_2.

Algoritmo 1 VNS()	**Algoritmo 2 LS() {Búsqueda Local}**
1: **Entrada**: un conjunto de estructuras de vecindarios N_i i=1,…,k_{max} 2: $S=S_0$ 3: k=1 4: **Repetir** 5: **Mientras** $k \leq k_{max}$ **Hacer** 6: $S_1=N_k(S)$ {Shaking} 7: S_2=**LS**(S_1) {Búsqueda Local en N_k} 8: **Si** $f(S_2) \leq f(S)$ **Entonces** 9: $S = S_2$ 10: k = 1 11: **Sino** 12: k = k + 1 {Move} 13: **Fin Si** 14: **Fin Mientras** 15: **Hasta Un Criterio de Parada** 16: **Salida**: La mejor solución encontrada S	1: **Entrada**: $S=S_0$ {Solución Inicial y una estructura de vecindarios N_k} 2: **Repetir** 3: $S_1= N_k(S)$ {Genera un vecino de S_1 a partir de S aplicando N_k} 4: **Si** $f(S_1) \leq f(S)$ **Entonces** 5: $S=S_1$ 6: **Sino** 7: Detener 8: **Fin Si** 9: **Hasta Detener** 10: **Salida**: Solución final S

La solución actual S es reemplazada por el nuevo óptimo local S_2, si y sólo si, S_2 es mejor que S y el mismo procedimiento es recomenzado en N_1, si S_2 no es mejor, la búsqueda se mueve al siguiente vecindario N_{k+1}, y se genera una nueva solución en N_{k+1} que se intenta mejorar hasta agotar las estructuras de vecindario disponibles.

El estudio de las variantes del algoritmo VNS se basó en la hipótesis que un cambio en la elección de la secuencia de vecindarios podría ayudar a escapar eficientemente de mínimos locales y, por ende, mejorar su desempeño. En las

siguientes subsecciones se describen las variantes implementadas y estudiadas en el presente trabajo.

3.2 Variante VNS+R

El algoritmo VNS+R (Algoritmo 3) utiliza una secuencia de estructuras de vecindarios de entrada aleatoria, la selección de la siguiente estructura de vecindario N_i se obtiene de algunas de las N_r (Algoritmo 3, líneas 12-13).

Algoritmo 3 VNS+R()	**Algoritmo 4 VNS+E()**
1: **Entrada**: un conjunto de estructuras de vecindarios N_i i=1,…,k_{max} 2: S=S_0 3: k=1 4: **Repetir** 5: **Mientras** k ≤ k_{max} **Hacer** 6: S_1=N_k(S) 7: S_2=**LS**(S_1){Búsqueda Local en N_k} 8: **Si** f(S_2) ≤ f(S) **Entonces** 9: S = S_2 10: k = 1 11: **Sino** 12: r = random(1, k_{max}) (r =! k) 13: k = r; 14: **Fin Si** 15: **Fin Mientras** 16: **Hasta Un Criterio de Parada** 17: **Salida**: La mejor solución encontrada S	1: **Entrada**: Un conjunto de estructuras de vecindarios N_i i=1,…,k_{max} 2: S=S_0 3: k=1 4: **Repetir** 5: **Mientras** k ≤ k_{max} **Hacer** 6: S_1 = N_k(S) 7: S_2 = **LSE**(S_1) {Búsqueda Local Explorativa} 8: **Si** f(S_2) ≤ f(S) **Entonces** 9: S = S_2 10: k = 1 11: **Sino** 12: k = k + 1 13: **Fin Si** 14: **Fin Mientras** 15: **Hasta Un Criterio de Parada** 16: **Salida**: La mejor solución encontrada S

Algoritmo 5 LSE() {Búsqueda Local Explorativa}	**Algoritmo 6 VNS+ER()**
1: **Entrada**: S=S_0 {Solución Inicial y estructura de vecindario N_k} 2: **Repetir** 3: genera$_n$(S) {Genera n vecinos a partir de S aplicando N_k} 4: S_1 = rank-best-neighbor() {Ordena y selecciona el mejor vecino de S} 5: **Si** f(S_1) ≤ f(S) **Entonces** 6: S=S_1 7: **sino** 8: Detener 9: **Fin Si** 10: **Hasta** Detener 11: **Salida**: Solución final S	1: **Entrada**: un conjunto de estructuras de vecindarios N_i i=1,…,k_{max} 2: S=S_0 3: k=1 4: **Repetir** 5: **Mientras** k ≤ k_{max} **Hacer** 6: S_1 = N_k(S) 7: S_2 = **LSE**(S_1) {Búsqueda Local Explorativa en N_k} 8: **Si** f(S_2) ≤ f(S) **Entonces** 9: S = S_2 10: k = 1 11: **Sino** 12: r = random(1, k_{max}) 13: k = r; 14: **Fin Si** 15: **Fin Mientras** 16: **Hasta Un Criterio de Parada** 17: **Salida**: La mejor solución encontrada S

3.3 Variante VNS + LHS

VNS+LHS tiene una etapa previa donde se procesa un conjunto de estructuras de vecindarios $\{N_1, ..., N_{kmax}\}$ para generar la secuencia a aplicar durante la búsqueda. Ésta se selecciona a partir de un conjunto de posibles secuencias de estructuras de vecindarios construidas con cuadrados latinos. Por tanto, a fin de construir dicho conjunto de posibles secuencias de estructuras de vecindarios y luego elegir la más conveniente, se propuso: 1) generar una combinación de secuencias uniformes en un espacio de diseño con el enfoque de cuadrados latinos, 2) evaluar las secuencias con la función objetivo del problema sobre un conjunto acotado de instancias y aplicando pruebas estadísticas apropiadas, y finalmente 3) elegir aquella que haya resultado más conveniente. A partir de allí, VNS+LHS se comporta como VNS.

3.4 Variante VNS+E

La variante VNS+E (`Algoritmo 4`) se implementó extendiendo en la búsqueda local (`Algoritmo 4`, línea 7) a un número mayor que 1, respecto a los posibles vecinos de la solución actual. Se realiza luego un ranking mediante la evaluación de la función objetivo y se selecciona a la solución de mejor calidad como la mejor solución con respecto a la solución actual (LSE, `Algoritmo 5`).

3.5 VNS+ER

El algoritmo VNS+ER (`Algoritmo 6`) es la última variante que se propone, la cual es obtenida a través de la combinación del VNS+E y el VNS+R, éste utiliza también la búsqueda LSE (`Algoritmo 5`).

4 Diseño de experimentos

Todos los algoritmos fueron implementados en lenguaje C++, en la biblioteca MALLBA [1]. Se establecen 20 instancias del problema, para cada una de estas instancias los algoritmos se ejecutan 30 veces y se configura como criterio de parada al número máximo de evaluaciones en 300.000. Los experimentos fueron ejecutados en un sub-cluster formado por once nodos cuyas características son las siguientes: CPUs de 64 bits cada uno con Intel Q9550 Quad Core 2.83GHz, 4GB DDR3 1333Mz de memoria, 160 Gb SATA disco duro y Asus P5Q3 placa madre.

4.1 Instancias del problema

Con el fin de encontrar instancias de prueba del problema, se realizó una revisión de la literatura en donde se aborda el problema de planificación de máquinas paralelas [3], [9] y [17] entre otros. Si bien se observó que existen instancias para el problema de planificación de máquinas paralelas, las mismas en general involucran más información dependiendo de la función objetivo estudiada y de las restricciones del

problema como, por ejemplo, tiempos dependientes de la configuración (*setup times*), o tiempos de lanzamientos al sistema (*release times*), tiempos de deterioro de tareas (*jobs*). Sin embargo, para nuestro problema sin restricciones (en las asignaciones) no fueron halladas ninguna, por tal motivo el conjunto de instancias fue construido a partir de datos obtenidos en la OR-Library [7], del problema *weighted tardiness problem*. Estos datos consistieron en pares (p_j, d_j), donde p_j es el tiempo de procesamiento y d_j es el (*due date*) la fecha de vencimiento o expiración de la tarea t_j para instancias de tamaño de 100 tareas, tales instancias son numeradas con #i (i=1, …, 125). Para los experimentos se seleccionaron 20 instancias no consecutivas en forma aleatoria por diferentes grupos. Además, debemos tener en cuenta que éstas fueron generadas con un factor de *tardiness* con dificultad incremental, de manera tal que, a mayor índice de identificación #i, es mayor el factor de *tardiness* con el que fueron generadas, es un motivo por el cual las instancias de mayor índice tienden a ser más difíciles de optimizar mediante la función objetivo T_{max}. Tales están disponibles previa solicitud (email: crgatica@email.unsl.edu.ar) .

La función de evaluación de la solución toma como entrada una instancia y calcula el valor de *maximum tardiness* de acuerdo a la ecuación (1) de la sección 2.

4.2 Ajustes de parámetros

Una de las ventajas de los algoritmos VNS consiste en su escaso número de parámetros para configurar, más allá de establecer el criterio de parada y de definir el conjunto de estructuras de vecindarios de entrada, estos son: cantidad de estructuras de vecindarios k_{max}=6, número máximo de iteraciones en la búsqueda local es igual 1.000, número de vecinos adicionales en la búsqueda local exploratoria es igual a 10, número máximo de evaluaciones de la función objetivo es igual a 300.000.

Las estructuras de vecindarios N_k están dadas por los operadores de perturbación N-swap (N_1), 2-opt (N_2), 3-opt (N_3), 4-opt (N_4), Shift (N_5) y Scramble (N_6). Una descripción de tales operadores se encuentra en la bibliografía [5]. La representación de cada solución es una permutación de enteros.

4.3 Métricas de evaluación del desempeño de los algoritmos

Las métricas de evaluación definidas para el estudio experimental fueron: *Bench*, el valor de referencia u óptimo conocido hasta el momento [18]; *Best*, el mejor valor de la función objetivo obtenido; *Ebest*=(*Best*-*Bench*/*Bench*)*100, el error porcentual; *Mbest*, el valor medio de *Best*; *Mebest*, el valor medio de *Ebest*; *Mediana*, es el valor central del conjunto de valores *Best* ordenados de menor a mayor; *Mevals*, el valor medio de ejecuciones donde se encontró el valor *Best*; *M. Times Runs*, el tiempo promedio de ejecución en nanosegundos.

5 Análisis de resultados

En esta sección se presenta el análisis de resultados de los experimentos computacionales realizados considerando las 20 instancias de tamaño *n*=100 tareas y

m=5 máquinas del problema de *planificación de máquinas paralelas idénticas sin restricciones* para minimizar la función objetivo de *maximum tardiness* (T_{max}).

La Tabla 1 sintetiza los resultados obtenidos por VNS y las variantes VNS+R, VNS+LHS, VNS+E y VNS+ER. Se muestran los valores *Bench, Best* y el error porcentual *Ebest.* Las entradas en la tabla marcadas en negrita corresponden a valores iguales o menores a los valores de referencia *Bench*, en estos casos los valores del error porcentual *Ebest* son cero o negativo y están marcados en itálica y subrayados.

Si miramos en las columnas correspondientes a VNS+R los valores (*Best* y *Ebest*) y los comparamos con los valores de VNS, vemos que no se obtuvieron mejores valores, salvo en las instancias del problema 6 y 41. La columna de la variante VNS+LHS, muestra mejores valores en varias instancias del problema con respecto a VNS, de la misma manera con las variantes VNS+E y VNS+ER notamos mejores valores, por lo tanto, se cumple la hipótesis planteada.

Utilizamos el test estadístico no paramétrico *Ranking de Friedman* [10] y [11] para realizar una comparación de las medias experimentales obtenidas tomando como métrica de evaluación al valor promedio de la *Mediana*, de esta manera obtuvimos un *ranking* de los algoritmos.

Tabla 1: Comparación de los valores *Bench* vs. *Best* y *Ebest* de los algoritmos propuestos.

Variantes		**VNS**		**VNS+R**		**VNS+LHS**		**VNS+E**		**VNS+ER**	
Inst.	*Bench*	*Best*	*Ebest*	*Best*	*Ebest*	*Best*	*Ebest*	*Best*	*Ebest*	*Best*	*Ebest*
1	**548**	**547**	*-0,182*	553	0,912	**542**	*-1,095*	**542**	*-1,095*	**539**	*-1,642*
6	**1594**	**1576**	*-1,129*	**1584**	*-0,627*	**1571**	*-1,443*	**1574**	*-1,255*	**1574**	*-1,255*
11	**2551**	**2548**	*-0,118*	2560	0,353	**2544**	*-0,274*	**2547**	*-0,157*	**2549**	*-0,078*
19	**3703**	3728	0,675	3741	1,026	3712	0,243	3717	0,378	**3703**	*0,000*
21	**5187**	**5187**	*0,000*	5207	0,386	**5184**	*-0,058*	**5181**	*-0,116*	**5177**	*-0,193*
26	**84**	99	17,857	113	34,524	99	17,857	**84**	*0,000*	**75**	*-10,714*
31	**1134**	1171	3,263	1170	3,175	1135	0,088	**1135**	*0,088*	**1132**	*-0,176*
36	**2069**	2111	2,030	2081	0,580	**2061**	*-0,387*	**2071**	*0,097*	**2061**	*-0,387*
41	**3651**	**3626**	*-0,685*	**3649**	*-0,055*	**3608**	*-1,178*	**3647**	*-0,110*	**3608**	*-1,178*
46	**4439**	4445	0,135	4456	0,383	4443	0,090	4440	0,023	**4443**	*0,090*
56	**617**	667	8,104	708	14,749	**609**	*-1,297*	**617**	*0,000*	**609**	*-1,297*
61	**1582**	1620	2,402	1743	10,177	1589	0,442	1615	2,086	**1580**	*-0,126*
66	**2360**	2403	1,822	2474	4,831	2364	0,169	**2359**	*-0,042*	**2359**	*-0,042*
71	**3786**	3829	1,136	3897	2,932	3797	0,291	3802	0,423	**3787**	0,026
86	**1194**	1259	5,444	1289	7,956	1214	1,675	1225	2,596	1209	1,256
91	**2204**	2284	3,630	2397	8,757	2244	1,815	2252	2,178	**2221**	*0,771*
96	**3185**	3196	0,345	3214	0,911	3196	0,345	3196	0,345	**3186**	*0,031*
111	**1365**	1621	18,755	1689	23,736	1486	8,864	1462	7,106	1437	5,275
116	**2222**	2392	7,651	2513	13,096	2279	2,565	2318	4,320	2288	2,970
121	**2999**	3230	7,703	3265	8,870	3150	5,035	3065	2,201	3046	1,567

Los resultados indican que existen diferencias significativas entre los algoritmos comparados donde el algoritmo VNS+ER muestra mejor desempeño que los restantes. Para comparar cada algoritmo entre sí y verificar si existen diferencias entre ellos, utilizamos un procedimiento *post-hoc*, que al igual que los test antes mencionados éste también es proveído por la herramienta *Controltest*, [8].

Al aplicar el procedimiento post-hoc de comparaciones de a pares entre el algoritmo de control (VNS+ER) respecto a VNS y VNS+R los valores *p-values* son menores a 0,05 por lo tanto, las parejas son significativamente diferentes, en cambio los *p-values* del VNS+LHS y VNS+E son mayores a 0,05 lo que implica que tales

algoritmos no son diferentes estadísticamente al algoritmo de control. Para finalizar el análisis de los resultados experimentales concluimos que la variante VNS+R no mejora el desempeño de VNS, en cambio las otras tres variantes VNS+LHS, VNS+E y VNS+ER si lo hacen, siendo la de mejor desempeño la variante VNS+ER en primer lugar, VNS+E en segundo y por último VNS+LHS según el ranking de *Friedman*. Finalmente, se observan (aunque no reportados aquí) tiempos de ejecución promedio de manera incremental según el siguiente orden: VNS+LHS, VNS+R, VNS, VNS+E y VNS+ER.

6 Conclusiones

En este trabajo presentamos la S-metaheurística VNS para el problema de *planificación de máquinas idénticas paralelas sin restricciones* que minimiza la función objetivo de *Tardanza Máxima* T_{max}. El objetivo del estudio consistió en proponer variantes heurísticas de la versión VNS estándar que mejoren el desempeño de la misma. Los resultados experimentales obtenidos muestran que el algoritmo VNS+ER fue el que obtuvo mejores resultados, seguido VNS+E y evidenciando diferencias estadísticamente significativas con VNS. A pesar de los buenos resultados obtenidos se observó una gran variabilidad de VNS+ER en el número promedio de evaluaciones de la función objetivo usados para alcanzar el óptimo. Es decir, estos dependen de la instancia del problema y de esta manera puede usar muy pocas o todas las evaluaciones permitidas dentro del máximo establecido por el criterio de parada. Esto nos permite concluir que en trabajos futuros será necesario realizar nuevos experimentos para analizar tanto un conjunto más grande de instancias del problema y de mayores dimensiones, como así también un análisis profundo del comportamiento de los algoritmos propuestos, en especial de VNS+ER.

Referencias

1. Alba, E., Almeida F., Blesa M., Cotta C., Diaz M., Dorta I., Gabarró J., González J., León C., Moreno L., Petit J., Roda J., Rojas A., Xhafa F., MALLBA: A library of skeletons for combinatorial optimization. Proceedings of the Euro-Par, pp. 927-932, 2002.
2. C. Erden, H. I. Demir and A. H. Kökçam: Solving Integrated Process Planning, Dynamic Scheduling, and Due Date Assignment Using Metaheuristic Algorithm. Hindawi Mathematical Problems in Engineering Volume 2019, 2019.
3. E. L. Franca Senne and A. A. Chaves: A VNS Heuristic for an Industrial Parallel Machine Scheduling Problem, ICIEOM – CIO, Valladolid, Spain, 2013.
4. E. Fortesa, L.H. Macedob Percival, B. de Araujo, R. Romero: A VNS algorithm for the design of supplementary damping controllers for small signal stability analysis. International Journal of Electrical Power & Energy Systems, vol. 94, no. 1, pp. 41–56, 2018.
5. E. G. Talbi: Metaheuristics from design to implementation, by John Wiley & Sons, C., 2009.
6. P. Hansen and N. Mladenovic, VNS: principles and applications. Eur. J. Oper. Res., 130, 449–467,1999.
7. J. E. Beasley, OR-Library: distributing test problems by electronic mail, Journal of the Operational Research Society, pp 1069-1072, 1990.

8. J. Derrac, S. García, D. Molina, and F. Herrera: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation, 2011.
9. J. Pei, B. Cheng, X. Liu, P. M. Pardalos and M. Kong: Single-machine and parallel-machine serial-batching scheduling problems with position-based learning effect and linear setup time. A. Oper Res. 272:217–241, 2019.
10. M. Friedman: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of American Statistical Association, 3:674–701, 1937.
11. M. Friedman: A comparison of alternative test of significance for the problem of the rankings. Annals of Mathematical Statistics, 11:86–92, 1940.
12. M. Pinedo, Scheduling: Theory, Algorithms and System, Prentice Hall, 1995.
13. Muestreo de hipercubo latino, https://es.wikipedia.org/wiki/Muestreo_de_hipercubo_latino.
14. P. J. García-Guarín; J. Cantor-López; C. Cortés-Guerrero; M. A. Guzmán-Pardo; S. Rivera-Rodríguez: Implementación del algoritmo VNS-DEEPSO para el despacho de energía en redes distribuidas inteligentes, INGE CUC, vol. 15, no. 1, pp. 142-154, 2019.
15. P. Hansen, N. Mladenović, R. Todosijević, S. Hanafi: Variable neighborhood search: basics and variants, EURO J. on Comp. Opt, 2016.
16. T. Morton and D. Pentico: Heuristic Scheduling Systems. John Wiley and Sons, NY, 1993.
17. W. Cheng, P. Guo, Z. Zhang, M. Zeng, and J.Liang: VNS for Parallel Machines Scheduling Problem with Step Deteriorating Jobs. Hindawi Publishing Corporation Mathematical Problems in Engineering Volume 7, 2012.
18. E. Ferretti and S. Esquivel: A Comparison of Simple and Multirecombinated Evolutionary Algorithms with and without Problem Specific Knowledge Insertion, for Parallel Machines Scheduling, International Transaction on Computer Science and Engineering, volume 3, number 1, 207-221, 2005.