ALGORITMO VORAZ ITERATIVO CON MULTI-VECINDAD APLICADO AL PROBLEMA DE SECUENCIACIÓN DIFUSO MULTIPRODUCTO Y MULTIETAPAS

Trabajo para optar al título de:
INGENIERA INDUSTRIAL

Tatiana Porras Cortés 20092015073
Correo: tatiporras96@gmail.com

LINDSAY ÁLVAREZ POMAR
Directora del trabajo de grado

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
FACULTAD DE INGENIERÍA
PROYECTO CURRICULAR DE INGENIERÍA INDUSTRIAL
Bogotá D.C., Colombia. 2021-05-30

TÍTULO

"ALGORITMO VORAZ ITERATIVO CON MULTI-VECINDAD APLICADO AL PROBLEMA DE SECUENCIACIÓN DIFUSO MULTIPRODUCTO Y MULTIETAPAS"

RESUMEN

Este trabajo exploró el Problema de Secuenciación Difuso Multiproducto y Multietapas (FMMSP por sus siglas en inglés, denotado como $FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}$) que se observa en industrias donde no hay estandarización establecida. Este problema ya había sido tratado con cuatro algoritmos (DBSA-LS, BDBSA, MBSA, e IGA), pero todos ellos presentan variabilidad en la solución encontrada, y algunos de ellos presentan soluciones de menor calidad. Se adaptó el Algoritmo Voraz Iterativo con Multi-Vecindad (MNIG por sus siglas en inglés) para resolver el modelo FMMSP, y se encontró que el algoritmo MNIG aporta soluciones de alta calidad y con menos variabilidad que los otros cuatro algoritmos.

Se reportaron los resultados del algoritmo aplicado a una instancia del problema (la instancia o10s2u5, que es la única instancia pública de este problema), y se compararon dichos resultados con los de los otros cuatro algoritmos DBSA-LS, BDBSA, MBSA, e IGA. También se midió el tiempo de ejecución, el cual no fue medido para los otros cuatro algoritmos. El algoritmo MNIG también fue aplicado a instancias de Taillard, que son instancias del problema flow shop básico.

Palabras Clave: Algoritmo Voraz Iterativo, Multi-vecindad, Secuenciación, Números Triangulares Difusos, Multiproducto, Multietapas.

ABSTRACT

This work explored the Fuzzy Multiproduct Multistage Scheduling Problem (FMMSP, denoted as $\boldsymbol{FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}}$) that can be observed in industries without an established standardization. This problem has already been treated with four algorithms (DBSA-LS, BDBSA, MBSA, and IGA), but all of them present variability en the solution they find, and some of them present solutions of lesser quality. Here the Multi-Neighborhood Iterated Greedy algorithm (MNIG) was adapted to solve the FMMSP model, and it was found that the MNIG algorithm gives solutions of high quality and with less variability than the other four algorithms.

The results from applying the algorithm to an instance of the problem (the o10s2u5 instance, which is the only public instance of this problem), will be reported, and the results will be compared with those of the other four algorithms DBSA-LS, BDBSA, MBSA, and IGA. The execution time will also be measured, which was not measured for the other four algorithms. The MNIG algorithm will also be applied to Taillard's instances, these are instances of the basic flow shop problem.

Keywords: Iterated Greedy Algorithm, Multi-neighborhood, Scheduling, Triangular Fuzzy Numbers, Multiproduct, Multistage.

INTRODUCCIÓN

El presente trabajo de tesis de pregrado ha hecho un aporte a un pequeño a mi conocimiento sobre los sistemas flow shop flexible con tiempos de procesamiento representados con números triangulares difusos, multiproducto y multietapa. Así como mi conocimiento sobre cómo obtener secuencias con un makespan relativamente bajo para este sistema, usando una metaheurística voraz, con multi-vecindad, e iterativo. La palabra 'makespan' significa el tiempo de terminación máximo de la secuencia.

Este trabajo no solamente tiene que ser útil para mi conocimiento, sino que también puede servir para el conocimiento de quien lo lea, pues esta combinación entre modelo y algoritmo, no había sido realizada antes.

Para ser más precisos: en el presente trabajo se eligió el tema de secuenciación, mejor conocido por su nombre en inglés como "scheduling." Dentro de este tema se revisó el estado del arte (consultando las bases de datos internacionales, a las que tiene acceso la Universidad Distrital), se encontró un modelo de scheduling que ha sido poco estudiado, y por aparte se encontró un algoritmo que nunca ha sido aplicado al modelo, sino hasta ahora.

El modelo encontrado (Yan, Han, and Gu 2020) es llamado FMMSP que significa Fuzzy Multiproduct Multistage Scheduling Problem, que en español se podría traducir como Problema de Secuenciación Difuso Multiproducto y Multietapas, y su representación en notación de scheduling es $\boldsymbol{FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}}$. Es un modelo interesante de ser estudiado pues incluye números

difusos triangulares, múltiples productos, y múltiples etapas.

El algoritmo encontrado (Shao, Shao, and Pi 2020) es llamado MNIG que significa Multi-Neighborhood Iterated Greedy algorithm. En español se traduce como algoritmo Voraz Iterativo con Multi-Vecindad. Se ha encontrado que este algoritmo sirve para resolver problemas de scheduling (Shao, Shao, and Pi 2020).

Lo novedoso resultó en que dicho algoritmo jamás había sido aplicado al modelo hasta ahora. Para realizar este trabajo se hizo necesario modificar el algoritmo, pues el algoritmo original no se puede aplicar directamente al modelo. En este trabajo se comprobó que el algoritmo MNIG se puede adaptar para encontrar soluciones del modelo FMMSP, y no solo eso, pues también se comprobó que las soluciones en promedio son mejores que las de los cuatro algoritmos que ya han sido aplicados al modelo FMMSP, específicamente los algoritmos: DBSA-LS, BDBSA, MBSA, e IGA. Esto se comprobó usando la única instancia publicada por los autores del modelo FMMSP (Yan, Han, and Gu 2020).

El algoritmo MNIG fue implementado en Python, y fue específicamente diseñado para tratar el modelo FMMSP. También se le hizo benchmarking al algoritmo MNIG usando 95 instancias de Taillard, que son instancias del problema flow shop básico. A pesar de que en total son 120 instancias de Taillard, solamente se usaron 95, porque las últimas 25 instancias son tan grandes, que esta implementación del algoritmo MNIG se tarda más de 24 horas para realizar una sola iteración, y esto es porque el algoritmo MNIG en este caso fue implementado para resolver el modelo FMMSP, no para resolver el modelo flow shop básico (pero indudablemente si se usara una instancia del modelo FMMSP con muchas máquinas y productos, la ejecución sería lenta, por lo que si se quisiera usar este algoritmo para esas instancias, primero se debería optimizar, lo cual se sale del alcance de este trabajo. Acá lo que cuenta es comprobar que el algoritmo MNIG es competitivo, e inclusive aporta mejores soluciones en promedio, que los otros cuatro algoritmos).

Después de buscar en las bases de datos bibliográficas a que tiene acceso la Universidad Distrital, se encontró que el modelo FMMSP únicamente ha sido solucionado usando los algoritmos DBSA-LS, BDBSA, MBSA, e IGA. La búsqueda se realizó utilizando la búsqueda avanzada en bases de datos tales como: ScienceDirect, SCOPUS, Cengage Learning, Engineering Village, IEEE, JSTOR, y EBSCO. Se usaron y mezclaron las palabras clave: 'fuzzy,' 'multiproduct,' 'multistage,' 'scheduling,' 'triangular fuzzy number.'

PLANTEAMIENTO DEL PROBLEMA

El problema que se resolvió con esta tesis, es el de explorar el algoritmo MNIG como candidato para encontrar secuencias de un sistema de producción del tipo FMMSP, de modo tal que dichas secuencias tengan un makespan bajo, comparado con los makespan encontrados con los algoritmos DBSA-LS, BDBSA, MBSA, e IGA.

Los algoritmos DBSA-LS, BDBSA, MBSA, e IGA, ya se usan para encontrar buenas soluciones del modelo FMMSP, pero estas soluciones tienen variabilidad, que hace que empeore la solución promedio encontrada por esos algoritmos. El algoritmo MNIG es capaz de mejorar la solución promedio encontrada para el modelo FMMSP.

Elementos del Problema

Según lo anterior, los elementos del problema son:

- Se requieren secuencias con bajos makespan en el modelo FMMSP, denotado como $FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}$. Estas secuencias son llamadas 'soluciones.'

- El algoritmo MNIG tiene potencial para encontrar dichas soluciones en un tiempo prudente. Los resultados se pueden comparar a otros cuatro algoritmos que ya han sido usados en el modelo FMMSP, y el algoritmo MNIG podría mejorar la solución promedio cuando se compare con las soluciones promedio de los otros cuatro algoritmos.

Este trabajo de tesis logró resolver estos elementos del problema, de la siguiente manera, respectivamente:

- El algoritmo MNIG ha servido para encontrar soluciones del modelo FMMSP.
- Las soluciones encontradas por el algoritmo MNIG son al menos igual de buenas, y mejores en promedio, que las soluciones encontradas por los algoritmos DBSA-LS, BDBSA, MBSA, e IGA. El algoritmo MNIG mejoró la solución promedio al ser comparada con los otros cuatro algoritmos, en la instancia o10s2u5 del modelo FMMSP.

Pregunta del Problema

¿Cómo puede el algoritmo MNIG mejorar las soluciones promedio encontradas para el modelo FMMSP?

Respuesta encontrada en este trabajo: Implementando el algoritmo MNIG en Python, y aplicándolo a la instancia o10s2u5 del modelo FMMSP. No es obligatorio que el algoritmo MNIG mejorara la solución promedio, pues eso no se podía predecir sin hacer este trabajo, pero se encontró que de hecho el algoritmo MNIG sí mejora la solución promedio respecto a los algoritmos DBSA-LS, BDBSA, MBSA, e IGA.

Subpreguntas del Problema

¿Cómo puede implementarse el algoritmo MNIG, para obtener resultados del modelo FMMSP, denotado como $FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}$?

Respuesta: El algoritmo MNIG puede ser implementado en Python, usando un paradigma de programación funcional, porque el algoritmo MNIG viene subdividido en cuatro algoritmos, cada uno de los cuales puede ser encapsulado dentro de una función, de modo que cada función pueda llamar a las otras según como está escrito en la definición del algoritmo MNIG. Esto se presta para ser implementado usando un paradigma de programación funcional, en el que se crean funciones que se llaman las unas a las otras.

¿De qué forma puede la implementación del algoritmo aportar resultados útiles,como por ejemplo,una secuencia de trabajos?

Respuesta: Usando una interfaz de línea de comandos. Toda interfaz de línea de comandos básica, tiene la facultad de imprimir resultados en pantalla. Estos resultados se pueden copiar en archivos de texto para ser analizados.

¿Cómo se comparan los resultados del algoritmo con los resultados de los otros cuatro algoritmos?

Respuesta: El algoritmo MNIG sorprendió en sus resultados, pues resultó ser al menos igual de bueno, y mejor en promedio, que los algoritmos DBSA-LS, BDBSA, MBSA, e IGA.

OBJETIVOS

Objetivo General

- Encontrar soluciones a través del algoritmo MNIG para el modelo FMMSP, y determinar si las

soluciones encontradas mejoran o no la solución promedio.

Objetivos Específicos

- Implementar el algoritmo MNIG de modo que se puedan obtener resultados del modelo FMMSP, denotado como $FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}$.
- Obtener secuencias del modelo FMMSP usando la implementación realizada.
- Comparar el algoritmo MNIG aplicado al modelo FMMSP con los otros cuatro algoritmos que se han aplicado al modelo FMMSP.
- Establecer si tiene sentido o no aplicar el algoritmo MNIG al modelo FMMSP, o si es preferible buscar otros algoritmos incluidos los que ya han sido probados.
- Determinar si la solución promedio es mejorada o no por el algoritmo MNIG.
- Determinar la efectividad del algoritmo MNIG respecto al modelo flow shop general.

JUSTIFICACIÓN

Se propone este trabajo porque en el mundo académico existe una necesidad de comprobar que nuevos algoritmos den resultados comparables a viejos algoritmos, al ser aplicados a los modelos. En particular en el mundo del scheduling ocurre que un algoritmo que se usa para un tipo de modelos pueda ser adaptado para tratar otro tipo de modelos diferente. Este trabajo se justifica como un aporte de prueba de que el algoritmo MNIG sirve para resolver el modelo FMMSP $FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}$ en comparación a otros algoritmos ya establecidos, y también como prueba de si el algoritmo MNIG puede mejorar la solución promedio en comparación con la solución promedio dada por los algoritmos DBSA-LS, BDBSA, MBSA, e IGA

¿Cuál es el problema de investigación?: El problema es que la solución promedio que aportan los algoritmos DBSA-LS, BDBSA, MBSA, e IGA, al modelo FMMSP, puede ser mejorada. El algoritmo MNIG es un buen candidato para lograr esta mejora, porque es un algoritmo voraz, cuya voracidad se basa en un algoritmo de búsqueda local, que explora el espacio de búsqueda y podría ser competitivo con los otros cuatro algoritmos, a tal punto que podría llegar a reducir el valor de la solución promedio en comparación a los otros cuatro algoritmos. Este trabajo comprobó que el algoritmo MNIG sí reduce la solución promedio, para la instancia o10s2u5 del modelo FMMSP.

¿Cómo ha sido resuelto este problema de scheduling?: El modelo FMMSP ha sido resuelto usando cuatro algoritmos, específicamente los algoritmos DBSA-LS, BDBSA, MBSA, e IGA. Hasta donde se sabe (tras hacer una revisión profunda de las bases de datos internacionales a las que tiene acceso la Universidad Distrital), estos son los únicos cuatro algoritmos con que se ha solucionado el modelo FMMSP. Por ello el algoritmo MNIG fue comparado con estos cuatro algoritmos, y resultó ser que el algoritmo MNIG es competitivo con los otros cuatro al encontrar soluciones del modelo FMMSP, resolviendo la única instancia publicada por los autores del modelo FMMSP en 0.83 segundos en promedio.

¿Cuál es el aporte a la solución de éste problema de scheduling en particular?: En este trabajo, el aporte a la solución del modelo FMMSP, es que se aportó un nuevo algoritmo de comprobada efectividad para solucionar el modelo FMMSP. Este nuevo algoritmo es el algoritmo MNIG. Con la única instancia pública internacional que existe del modelo FMMSP se comprobó la competitividad del algoritmo MNIG para resolver el modelo FMMSP.

¿Por qué hay que solucionar este problema?: Porque los algoritmos DBSA-LS, BDBSA, MBSA, e IGA presentan variabilidad en sus soluciones, lo que hace que empeore la solución promedio. Debido a la

naturaleza NP-Hard de los modelos derivados de flow shop (como lo es el modelo FMMSP), aún no existe forma de garantizar la optimalidad de las soluciones encontradas por las metaheurísticas, y por eso existe cierta competencia entre diferentes metaheurísticas que usan diferentes estrategias para buscar soluciones en el espacio de búsqueda. Esta competencia sigue abierta para el modelo FMMSP, y aunque el algoritmo MNIG fue diseñado para problemas muy diferentes, en este trabajo se ha logrado adaptar el algoritmo MNIG para tratar el modelo FMMSP y encontrar soluciones al mismo, así como disminuir el valor de la solución promedio, comparado con los algoritmos DBSA-LS, BDBSA, MBSA, e IGA.

¿Qué sucede si no solucionamos este problema identificado?: Si no se solucionara este problema, no avanzaría nuestro conocimiento sobre el modelo FMMSP y sus soluciones. El modelo FMMSP es un modelo que se puede encontrar en sistemas de producción reales, nada más el hecho de que el modelo FMMSP incluye números difusos triangulares (F es por Fuzzy), muestra que el modelo FMMSP puede ser aplicado a sistemas de producción colombianos, en donde no hay una fuerte estandarización, porque si las estaciones de trabajo tienen tiempos de procesamiento fuertemente estandarizados, ya no se hacen necesarios los números triangulares difusos, y resulta más fácil y razonable usar simples números para representar los tiempos de procesamiento. En cambio, en sistemas donde no hay una fuerte estandarización, puede tener sentido usar números triangulares difusos para representar los tiempos de producción de las estaciones de trabajo, porque así se representa la incertidumbre sobre el verdadero tiempo de producción, usando las tres partes que forman un número triangular difuso: el tiempo optimista de producción, el tiempo promedio de producción, y el tiempo pesimista de producción (esto se puede ver también en la sección del marco de referencia).

ALCANCES Y LIMITACIONES DEL PROYECTO

Alcances: Los alcances de este proyecto están bien definidos. Parte del alcance consiste en escribir un algoritmo en Python, específicamente implementar el algoritmo MNIG en un programa de consola de Python. En el proyecto también se incluye la tesis de grado en la que se compararán los resultados del algoritmo implementado en este proyecto con los de otros cuatro algoritmos (DBSA-LS, BDBSA, MBSA, e IGA) aplicados al modelo FMMSP $\boldsymbol{FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}}$. Dicha tesis será entregada a la Universidad Distrital.

MARCO DE REFERENCIA

El marco de referencia se enfoca en el scheduling de los sistemas de producción tipo FMMSP, en el algoritmo MNIG que es un tipo de metaheurística, y en los otros cuatro algoritmos que son DBSA-LS, BDBSA, MBSA, e IGA.

Antecedentes y Marco Teórico

En los antecedentes encontramos dos temas pilares: el scheduling, y las metaheurísticas. Del libro de scheduling de Pinedo (Pinedo 2016) se tiene una interrelación entre los temas de los antecedentes. Se han creado multitud de heurísticas y metaheurísticas para resolver los problemas de scheduling. También ha ocurrido que muchas metaheurísticas pueden ser modificadas para tratar otros problemas diferentes de scheduling.

El scheduling es un proceso de toma de decisiones en el que se asignan recursos a los trabajos (u ordenes) a realizar. En esta asignación los trabajos se van realizando en virtud de los recursos que le han sido asignados. En otras palabras, no se puede completar un trabajo que no ha sido asignado a los recursos que lo llevan a cabo. En la decisión de cómo realizar dichas asignaciones surge la importancia del scheduling, por

el hecho de que distintas asignaciones conllevan a diferentes tiempos de realización de los trabajos. (Pinedo 2016)

La medida de desempeño más ubicua del scheduling es el tiempo de terminación máximo, también conocido como makespan, que es el tiempo en que se termina el último trabajo de la secuencia. En este sentido, el makespan representa el tiempo que toma realizar todos los trabajos. En el scheduling, lo más comúm es que la función objetivo sea minimizar el makespan, es decir, minimizar el máximo de los tiempos de terminación de los trabajos de la secuencia. (Pinedo 2016)

Respecto a las heurísticas y metaheurísticas, en ambas se renuncia a la optimalidad, se renuncia a encontrar la mejor solución posible según determinada medida de desempeño, a cambio de encontrar soluciones aceptables en un tiempo prudente. (Talbi 2009)

La notación del modelo FMMSP: $\boldsymbol{FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}}$, fue creada con base en la notación tradicional de scheduling de Pinedo. (Pinedo 2016)

En cuanto al algoritmo MNIG y la diferencia entre heurísticas y metaheurísticas, las metaheurísticas pueden abarcar un amplio abanico de problemas, mientras que las heurísticas se diseñan para un problema muy específico (Talbi 2009). Acorde a lo anterior, el algoritmo MNIG es una metaheurística pues es un algoritmo que se puede usar para diversos tipos de problemas, y aunque originalmente no haya sido diseñado pensando en el modelo FMMSP $\boldsymbol{FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}}$ se puede adaptar fácilmente sin cambiar la esencia de la metaheurística.

Marco Conceptual

Scheduling: problemas de optimización en los que se establece una secuencia de trabajos en un sistema de producción, y se busca optimizar la medida de desempeño de interés en el sistema de producción.

Existe variedad de tipos de sistemas de producción, los primera división se hace en sistemas determinísticos y estocásticos. En los sistemas determinísticos se conoce con precisión los parámetros del sistema, mientras que en los sistemas estocásticos los parámetros tienen variabilidad. (Pinedo 2016)

Luego están los sistemas de una sola máquina, que son sistemas teóricos que sirven para formular modelos más complejos, puesto que en los sistemas de producción reales rara vez se presenta el caso de una sola máquina. Después vienen los sistemas de máquinas paralelas en los que varias máquinas pueden procesar los mismos trabajos. En el siguiente nivel de complejidad se encuentran los sistemas tipo Flow Shop en los que los trabajos van fluyendo de una máquina a la siguiente en un mismo orden de máquinas. Sigue los sistemas tipo Job Shop en los que el orden de las máquinas puede ser diferente para cada trabajo. Por último los sistemas Open Shop en los que el orden de las máquinas por las que debe pasar cada trabajo puede ser parte de las decisiones de scheduling. (Pinedo 2016)

Metaheurística: Las metaheurísticas surgen como una alternativa a la situación en que la optimización de problemas toma demasiado tiempo. Cuando los problemas a optimizar son grandes y según aumentan de tamaño, el tiempo de optimización puede pasar de horas a días, a meses y años, e incluso mucho más tiempo. Los problemas realistas tienden a ser de gran tamaño, por lo que su optimización puede tomar años o más, incluso con los computadores más rápidos, en estos casos se usan metaheurísticas, que aunque no optimizan el problema, sí proveen soluciones aceptables en un tiempo razonable. (Talbi 2009)

La complejidad de una metaheurística, y en general la complejidad de un algoritmo se mide con el tiempo

que toma su ejecución en función del tamaño del problema. Dicho tiempo se denota usando la llamada notación O-grande, por ejemplo, si n es el tamaño del problema, y $O(n^2)$ es la notación O-grande de un algoritmo dado, significa que el tiempo de resolución de dicho algoritmo aumenta al cuadrado del tamaño del problema n. Esto no significa que el tiempo de resolución sea igual al cuadrado de n, sino que el tiempo de resolución va aumentando con el tamaño del problema, y se acota por una cantidad proporcional al cuadrado del tamaño del problema. (Talbi 2009)

Hay varios tipos de complejidad en la optimización de problemas y en los algoritmos. Se considera que un tiempo de resolución razonable es el llamado tiempo polinomial, así, un algoritmo de tiempo polinomial, es un algoritmo cuyo tiempo de resolución puede ser acotado con una función polinomial, y su notación O-grande respectiva será $O(p(n))$ donde $p(n)$ es una función polinomial del tamaño del problema n. (Talbi 2009)

Ocurre en los casos reales que muchas veces el tiempo de resolución de la optimización en un problema de scheduling, no puede ser acotado por una función polinomial. En estos casos la complejidad del problema es no polinomial, o NP por sus siglas en inglés. Una demostración importante, es que la optimización de modelos tipo Flow Shop y la optimización de modelos más complejos que se basen en el Flow Shop es de complejidad NP-Hard, que es una variante de los problemas no polinomiales. Lo anterior requiere que para estos modelos de scheduling se usen metaheurísticas para su resolución. (Garey, Jhonson, and Sethi 1976)

Modelo FMMSP: El modelo FMMSP, denotado como $FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}$, representa un sistema de producción en el que los trabajos van pasando por etapas, y en cada etapa hay un número de máquinas (o unidades como les llaman en la literatura). Cada trabajo puede ser procesado solamente por una unidad en cada etapa. Cuando un trabajo pasa por todas las etapas se dice que está terminado. En el modelo FMMSP se busca optimizar el tiempo de terminación máximo de los trabajos, es decir, el tiempo en el que el último trabajo de la secuencia es terminado, medido desde el inicio de la producción. (Yan, Han, and Gu 2020)

Para modelar el tiempo de producción en el modelo FMMSP se utilizan números difusos, en concreto número difusos triangulares. El hecho de que el tiempo de producción se represente mediante números difusos triangulares conlleva a que también el tiempo de terminación y el tiempo de inicio de cada trabajo sean también números difusos triangulares. Los números difusos triangulares son usados en situaciones donde se quiere representar la incertidumbre. Para el caso del modelo FMMSP se usan los números difusos triangulares para representar la incertidumbre en los tiempos de terminación, en otras palabras, un trabajo puede terminar o bien antes o bien después del momento en que se espera su terminación. En el modelo FMMSP cada número difuso triangular está compuesto a su vez por tres números. Esto es, cada tiempo de procesamiento, de terminación, o de inicio es una terna ordenada, en la que el primer número representa el tiempo optimista (bien sea de procesamiento, terminación, o inicio según el caso), el segundo número representa el tiempo esperado, y el tercer número representa el tiempo pesimista. (Yan, Han, and Gu 2020)

Los números difusos triangulares se definen de la siguiente forma: (Anand and Bharatraj 2017)

Sea $\tilde{A}$ un número difuso triangular, el símbolo encima de la A indica que se trata de un número difuso. Entonces $\tilde{A} = (a, b, c)$ donde a, b, y c son tres números reales que conforman al número difuso.

Por otra parte sea x cualquier número real, y sea $\mu_{\tilde{A}}(x)$ la función del grado de pertenencia de x al número difuso $\tilde{A}$, entonces:

$$\mu_{\tilde{A}}(x) = \begin{cases} 0, & x < a; \\ \frac{x-a}{b-a}, & a \leq x \leq b; \\ \frac{c-x}{c-b}, & b < x \leq c; \\ 0, & x > c, \end{cases} \tag{1}$$

Como se observa en la ecuación (1) la función del grado de pertenencia está entre 0 y 1. Antes de a y después de c su valor es 0, es decir que sólo tiene valor distinto de 0 entre a y c. Por otra parte, esta función del grado de pertenencia llega a un valor de 1 cuando $x = b$, entre a y b el valor de la función va ascendiendo de 0 a 1, y entre b y c desciende de 1 a 0, lo que hace que efectivamente la función forme un triangulo que inicia en a, tiene el máximo en b y termina en c. (Anand and Bharatraj 2017)

Haciendo uso de los números difusos triangulares, a continuación se define el modelo FMMSP $FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}$: (Yan, Han, and Gu 2020)

Sea NT el número de órdenes o trabajos a procesar, sea I el conjunto de los trabajos, tal que $I = \{1, 2, \ldots, i, \ldots, NT\}$ donde i es cualquier trabajo del conjunto I.

Sea L el número de etapas por las que deben pasar los trabajos, sea S el conjunto de las etapas, tal que $S = \{1, 2, \ldots, s, \ldots, L\}$ donde s es cualquier etapa del conjunto S.

En cada etapa s se presentan n_s unidades de procesamiento paralelas, sea U_s el conjunto de unidades de procesamiento de la etapa s, tal que $U_s = \{1, 2, \ldots, n_s\}$.

El tiempo de procesamiento del trabajo i en la unidad u es un número difuso triangular, por lo que se puede representar como: $\tilde{T}_{i,u} = (T^1_{i,u}, T^2_{i,u}, T^3_{i,u})$. Notar que u es cualquier unidad de todas las etapas, y no de una etapa en particular, es decir si se númeran todas las unidades o "máquinas" sin distinguir la etapa, entonces u es uno de estos números, o sea u es cualquier máquina de todo el sistema de producción. Las unidades que pertenecen a una etapa en particular se denotan con U_s.

El tiempo de inicio del trabajo i en la etapa s también es un número difuso triangular y se representa como $\widetilde{Ts}_{i,s}$, y el tiempo de terminación del trabajo i en la etapa s siendo un número difuso triangular se representa como $\widetilde{Tf}_{i,s}$

Para definir completamente el modelo hacen falta pocas variables, entre ellas una variable binaria, llamada $Z_{i,u,s}$. Si la variable binaria es igual a 1 indica que el trabajo i se procesará en la unidad u de la etapa s, en caso contrario si la variable es igual a 0 significa que el trabajo i no será procesado en la unidad u de la etapa s. Sea F el tiempo de terminación del último trabajo de la secuencia, por tanto F es la función objetivo a minimizar, F también es llamado el makespan. Y por último sea M un número muy grande, normalmente conocido como gran M.

Teniendo las definiciones anteriores, el modelo FMMSP es el siguiente:

$$minF = max(\widetilde{Tf}_{i,L}) \qquad \forall i \in I \tag{2}$$

$$\sum_{u \in U_s} Z_{i,u,s} = 1 \qquad \forall i \in I, s \in S \tag{3}$$

$$\widetilde{Tf}_{i,s} = \widetilde{Ts}_{i,s} + \sum_{u \in U_s} (Z_{i,u,s} * \tilde{T}_{i,u}) \qquad \forall i \in I, s \in S \tag{4}$$

$$\widetilde{Tf}_{i,s} + M * (2 - Z_{i,u,s} - Z_{j,u,s}) \leq \widetilde{Ts}_{j,s} \qquad \forall i \in I, j \in I, i < j, s \in S, u \in U_s \tag{5}$$

$$\widetilde{Tf}_{i,s} \leq \widetilde{Ts}_{i,s+1} \qquad \forall i \in I, s \in S \tag{6}$$

La ecuación (2) es la función objetivo de minimizar el máximo de los tiempos de terminación de los trabajos en la etapa L, que es la última etapa. Luego la ecuación (3) establece que el trabajo i en la etapa s solamente se pueda realizar en una sola unidad u. La ecuación (4) muestra el tiempo de terminación siendo igual al tiempo de inicio más el tiempo de procesamiento. La inecuación (5) es una restricción para el tiempo de inicio del trabajo j si este se realiza en la misma unidad que un trabajo anterior i. Por último la inecuación (6) es una restricción al tiempo de inicio del trabajo i en la etapa siguiente $s+1$ que lo hace ser mayor o igual que el tiempo de terminación del trabajo i en la etapa s. (Yan, Han, and Gu 2020)

Teniendo definido el modelo $FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}$, para resolverlo hace falta saber exactamente qué se entiende por el máximo entre dos números difusos triangulares. Como en el modelo se calcula el máximo de los tiempos de terminación, y dichos tiempos son números difusos triangulares, entonces se requiere definir los operadores de comparación $<$, y $>$ para números difusos triangulares. Solo es necesario definir un operador pues el otro es su negación. El operador $>$ se define de la siguiente manera: (Yan, Han, and Gu 2020)

Sean $\tilde{A} = (a^1, a^2, a^3)$ y $\tilde{B} = (b^1, b^2, b^3)$. Para establecer que $\tilde{A} > \tilde{B}$ se dan los tres casos siguientes

$$\begin{gathered} Si \left(\frac{a^1 + 2 * a^2 + a^3}{4} \right) > \left(\frac{b^1 + 2 * b^2 + b^3}{4} \right), entonces\ \tilde{A} > \tilde{B} \\ Si \left(\frac{a^1 + 2 * a^2 + a^3}{4} \right) = \left(\frac{b^1 + 2 * b^2 + b^3}{4} \right), y\ a^2 > b^2, entonces\ \tilde{A} > \tilde{B} \\ Si \left(\frac{a^1 + 2 * a^2 + a^3}{4} \right) = \left(\frac{b^1 + 2 * b^2 + b^3}{4} \right), y\ a^2 = b^2, a^3 - a^1 > b^3 - b^1, entonces\ \tilde{A} > \tilde{B} \end{gathered} \tag{7}$$

Algoritmo MNIG: El algoritmo MNIG se entiende dividido a su vez en cuatro algoritmos, cada uno realizando un rol diferente y al ser aplicados como se indica logran una solución. (Shao, Shao, and Pi 2020)

El primer algoritmo es llamado DNEH_SMR, sirve para obtener una secuencia de trabajos inicial promisoria en la región de búsqueda. DNEH_SMR significa “Distributed NEH with Small-Medium Rule” a su vez NEH proviene de “Nawaz, Enscore, Ham” que es un reconocido algoritmo para minimizar el makespan. (Shao, Shao, and Pi 2020)

El algoritmo MNIG usa su propia notación, que en ocasiones usa los mismos símbolos del modelo FMMSP $FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}$. En este trabajo se le dará prelación a los símbolos del modelo FMMSP, puesto que el algoritmo MNIG se está adaptando al modelo, así también se adaptarán los símbolos de modo que sean congruentes con los del modelo FMMSP.

Para entender el algoritmo DNEH_SMR se requieren algunos símbolos extra. Una secuencia de trabajos factible se denota como π, entonces π es una solución factible al problema. En el algoritmo DNEH_SMR se divide en dos la secuencia, el punto en que se divide es llamado k, y es definido como $k = \lfloor \frac{L}{2} \rfloor$. Luego en el algoritmo DNEH_SMR se promedian los tiempos de cada grupo de etapas: antes de k y después de k. (Shao, Shao, and Pi 2020)

En el modelo FMMSP $FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}$ no se consideran los tiempos de procesamiento por etapa sino por unidad (o máquina) por lo que se hace necesario modificar ligeramente el algoritmo DNEH_SMR. Para hacer un tiempo de procesamiento equivalente por etapa, partiendo de los tiempos por unidad, se promedian los tiempos de las unidades en la etapa, es decir $\forall u \in U_s$, de este modo defínase $\widetilde{Ta}_{i,s} = \frac{1}{n_s} * \sum_{u \in U_s} \widetilde{T}_{i,u}$. Ta es por $Taverage$. Esta modificación es válida porque en cada etapa las máquinas trabajan en paralelo, es decir, cuando un trabajo pasa por una etapa, solamente es procesado por una máquina, no por todas las máquinas, entonces el tiempo promedio por etapa representa el tiempo que en promedio pasan los trabajos en cada etapa.

Teniendo $\widetilde{Ta}$ se definen los tiempos de procesamiento promedio por grupo de etapas, para el primer grupo antes de k, $\widetilde{T}'_{i,1} = \frac{1}{k} * \sum_{l=1}^{k} \widetilde{Ta}_{i,l}$, y para el segundo grupo después de k, $\widetilde{T}'_{i,2} = \frac{1}{L-k} * \sum_{l=k+1}^{L} \widetilde{Ta}_{i,l}$. Nótese el símbolo prima para indicar que son los tiempos promedio de los grupos de etapas. (Shao, Shao, and Pi 2020)

Del algoritmo DNEH_SMR resulta una secuencia factible y además no aleatoria, pues intenta crear una solución inicial de calidad. Como parte de ello se crea una secuencia ordenando los trabajos, en orden ascendente del tiempo promedio en el grupo 2 $\widetilde{T}'_{i,2}$, sea dicha secuencia π_{re1} donde $re1$ indica que es el primer reordenamiento de la secuencia. Para el segundo reordenamiento, π_{re2} se consigue al tomar el primer trabajo y luego el trabajo de en medio de π_{re1}, eliminando los trabajos tomados de π_{re1}. Si el número de trabajos en el que se busca el trabajo de en medio es par, se puede tomar el trabajo que está antes de la mitad (pues cuando el número de trabajos que quedan es par existen dos trabajos en el medio, tomar el primero). El hecho de que se van eliminando los trabajos significa que solo hay que repetir la instrucción: tomar el primero, tomar el de en medio, tomar el primero, tomar el de en medio, hasta agotar los trabajos y así dar forma a π_{re2}. (Shao, Shao, and Pi 2020)

Dado que el modelo FMMSP es un modelo para sistemas del tipo flow shop flexible, cada trabajo sigue una secuencia de máquinas, es decir, en este modelo cada solución es una secuencia de trabajos en donde cada trabajo tiene una secuencia de máquinas. Para elegir la secuencia de máquinas de un trabajo, en este documento se usará la regla de asignar un trabajo a la máquina que lo procese más pronto, incluyendo la posibilidad de esperar a que un trabajo anterior sea procesado en una máquina y que se asigne el siguiente trabajo a esa máquina, según lo que resulte en el menor tiempo final de procesamiento. Esto se representa en la notación del modelo con: $Z_{i,u,s}$, porque el uso de la variable $Z_{i,u,s}$ en el modelo FMMSP muestra que es posible procesar dos trabajos diferentes al mismo tiempo en una misma etapa, usando dos máquinas diferentes.

Por último en el algoritmo, se toma cada trabajo, y se reinserta en cada posición y se deja en la posición en

la que resulte el mínimo makespan. Sea π_{re3} la secuencia resultante de estas reinserciones. El algoritmo DNEH_SMR retorna la secuencia π_{re3} como su resultado. (Shao, Shao, and Pi 2020)

Teniendo los elementos anteriores, a continuación se presenta el algoritmo DNEH_SMR en su versión de pseudocódigo de Python:

Algoritmo 1 DNEH_SMR en pseudocódigo Python

---

def DNEH_SMR(todas las $\widetilde{T}_{i,u}$):

1: Computar $k = \lfloor \frac{L}{2} \rfloor$

2: Computar $\widetilde{Ta}_{i,s} = \frac{1}{n_s} * \sum_{u \in U_s} \widetilde{T}_{i,u} \forall i \in I, s \in S$

3: Computar $\widetilde{T}'_{i,1} = \frac{1}{k} * \sum_{l=1}^{k} \widetilde{Ta}_{i,l}$, y $\widetilde{T}'_{i,2} = \frac{1}{L-k} * \sum_{l=k+1}^{L} \widetilde{Ta}_{i,l} \ \forall i \in I$

4: Ordenar los trabajos según el orden ascendente de $\widetilde{T}'_{i,2}$, guardando la secuencia en π_{re1}

5: Crear la secuencia π_{re2}, quitando el primer trabajo y luego el de en medio de π_{re1}, y ponerlos en π_{re2}, repetir con los trabajos restantes hasta que ya no queden trabajos en π_{re1}

6: Copiar π_{re2} en π_{re3}

7: for j in π_{re3}:

8: Reinsertar j en cada posición de π_{re3} y medir el makespan

9: Dejar j en la posición con el mínimo makespan sin mover los trabajos de iteraciones anteriores

10: return π_{re3}

Por claridad en el paso 4 del algoritmo DNEH_SMR, si por ejemplo la secuencia π_{re1} es la secuencia $[1, 6, 2, 5, 4, 3]$, la secuencia π_{re2} correspondiente será entonces $[1, 5, 6, 4, 2, 3]$. (Shao, Shao, and Pi 2020)

Con el algoritmo DNEH_SMR se obtiene una solución inicial π_{re3}. Este primer algoritmo, y los dos siguientes, algoritmos 1, 2, y 3 se integran en el algoritmo 4. Desde ya decir que el algoritmo 4 es el algoritmo MNIG. Para llevar a cabo el algoritmo MNIG hace falta mostrar otros dos algoritmos, los algoritmos 2 y 3 que serán trabajados a continuación.

Para hacer el algoritmo MNIG se necesita poder destruir y reconstruir la secuencia π_{re3} reconstruyendo de

modo que se vaya mejorando la función objetivo. La destrucción y reconstrucción de la secuencia se logran con el algoritmo 2. También se necesita hacer la búsqueda local con multi-vecindad, en la que se hacen inserciones e intercambios de trabajos en la secuencia, buscando la mejora en la función objetivo. Esta búsqueda local con multi-vecindad se realiza con el algoritmo 3. (Shao, Shao, and Pi 2020)

En el algoritmo 2, la destrucción y reconstrucción de una secuencia de trabajos tiene por objeto la mejora en la función objetivo. Para hacer este algoritmo de destrucción y reconstrucción, se comienza por remover d trabajos de la secuencia de manera aleatoria. Con estos trabajos removidos se forma la secuencia π_d. Con los trabajos restantes se forma la secuencia π_r. Luego se toma cada trabajo en π_d y se reinserta en cada posición de π_r, se mide el makespan a pesar de que esta secuencia tenga menos trabajos que la original y se deja el trabajo en la posición que tenga el mínimo makespan. (Shao, Shao, and Pi 2020)

Luego cada trabajo en π_r se reinserta en la secuencia bajo construcción, se mide el makespan, y si éste es menor que el makespan inicial entonces se inserta el el trabajo en la posición que haya generado el nuevo y mejor makespan. Se hace lo mismo con los restantes trabajos en π_r sin reemplazar los que ya hayan sido insertados. Por último se repite este proceso con los demás trabajos en π_d. La secuencia resultante es retornada por el algoritmo. (Shao, Shao, and Pi 2020)

Llámesele π_{input} a la secuencia a ser destruida y reconstruida, y sea π_{desrec} la secuencia retorna por el algoritmo 2 tras ser destruida y reconstruida. En consideración de lo dicho se presenta el algoritmo de destrucción y reconstrucción en su versión de pseudocódigo de Python:

Algoritmo 2 Destrucción y reconstrucción de las secuencias

---

def destruction_reconstruction(π_{input}, d):

1: Remover aleatoriamente d trabajos de la secuencia π_{input} y ponerlos en la secuencia π_d. Poner el resto de trabajos en la secuencia π_r

2: for j in π_d:

3: Reinsertar j en cada posición de π_r y medir el makespan como $fitness1$

4: Insertar j en la posición que tenga el mínimo $fitness1$

5: for j2 in π_r excepto los trabajos insertados de π_d:

6: Reinsertar j2 en cada posición de π_r y medir el makespan como $fitness2$

7: if ($fitness2$ < $fitness1$):

8: Insertar j2 en la posición con el menor $fitness2$

9: $\pi_{desrec} = \pi_r$

10: return π_{desrec}

El algoritmo 3 es el algoritmo de búsqueda local con multi-vecindad. En este algoritmo se toma una secuencia a ser procesada π_{input}, y el algoritmo retorna una secuencia factible mejorada $\pi_{multivec}$. En el algoritmo se le hacen dos tipos de operaciones a la secuencia: inserciones e intercambios de trabajos. Insertar un trabajo significa tomar un trabajo y ponerlo en otra posición, desplazando los demás trabajos hacia la derecha. Intercambiar dos trabajos implica tomar dos trabajos y cambiarlos de posición, el uno en la posición del otro. Sea π_{modif} la secuencia tras ser modificada con alguna de estas operaciones, y π_{temp} una secuencia para guardar un resultado temporal. Para este algoritmo, sea $C_{max}(\pi) = F$ para dejar a F en función de la secuencia π. C_{max} es un símbolo común para el makespan. (Shao, Shao, and Pi 2020)

En el algoritmo de búsqueda local con multi-vecindad hay dos funciones clave, que realizan las operaciones de inserción y de intercambio de trabajos. La operación de inserción es realizada con la función $Insertion_{inner}(\pi)$, en esta función se toma cada trabajo de π y se inserta en cada posición, midiendo el makespan y al final, si el makespan mejora, se deja insertado el trabajo en la posición que tenga el menor makespan. La otra función es $Swap_{inner}(\pi)$ en la que se intercambian dos trabajos de π, se mide el makespan, y al final dejando intercambiados los trabajos en donde se presente el menor makespan. Si después de aplicar estas funciones se encuentra mejora, la función vuelve a ser aplicada, hasta que no se presente mejora en el makespan. (Shao, Shao, and Pi 2020)

Con lo anterior, aquí el algoritmo de búsqueda local con multi-vecindad en su versión de pseudocódigo de Python:

Algoritmo 3 Búsqueda local con multi-vecindad

---

def local_search(π_{input}):

1: |$\pi_{modif} = \pi_{input}, l_{max} = 2, l = 1$

2: while (l<=l_{max}):

3: if (l== 1):

4: |$\pi_{temp} = Insertion_{inner}(\pi_{modif})$

5: else if (l== 2):

6: |$\pi_{temp} = Swap_{inner}(\pi_{modif})$

7: if ($C_{max}(\pi_{temp}) < C_{max}(\pi_{modif})$):

8: |$\pi_{modif} = \pi_{temp}, l = 1$

```
9:      else:
10:         |l = l + 1
11: |π_multivec = π_modif
12: return π_multivec
```

Teniendo los tres primeros algoritmos, el cuarto y último es la síntesis de los tres más algunas adiciones, por ello es llamado el algoritmo MNIG. Sea π_{result} la secuencia final resultado de la aplicación del algoritmo MNIG. Para aplicar el algoritmo MNIG es necesario definir un criterio de terminación del algoritmo, pues en caso contrario el algoritmo nunca pararía de ejecutarse. En la literatura el criterio de terminación propuesto (ajustado para el modelo FMMSP $FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}$) es $N^2 * L * 0.1ms$ (Shao, Shao, and Pi 2020).

Este es un buen criterio, pero tiene dos problemas, primero no tiene en cuenta todas las partes del modelo FMMSP y segundo está en unidades de milisegundo. Para el primer problema, se puede incluir lo que falta del modelo FMMSP, las unidades (o máquinas). Sea $UT = \sum_{s=1}^{L} |U_s|$, UT es el total de unidades. Lo segundo puede ser un problema porque si el algoritmo para según una cantidad de tiempo arbitraria en milisegundos, ello fuerza a escribir el código de modo que cada iteración dure el menor tiempo posible. Pero eso es una desventaja porque no se puede hacer que el algoritmo dé resultados parciales, pues imprimir resultados parciales toma tiempo, a pesar de que sea poco tiempo. Otra forma de hacerlo sería teniendo un criterio de terminación en número de iteraciones, o sea que el algoritmo pare luego de un número de iteraciones. Según lo anterior, un criterio de terminación para el algoritmo MNIG aplicado al modelo FMMSP podría ser $Iterations = N^2 * L * UT$.

Antes de proceder con el pseudocódigo del algoritmo MNIG falta decir que este algoritmo explora en la región de búsqueda, intentando no quedarse en óptimos locales. A este efecto, existe un criterio de aceptación de nuevas soluciones que incluye la posibilidad de aceptar soluciones peores con una baja probabilidad. Este criterio define un nuevo parámetro derivado T, cuya ecuación es $T = T_0 * \frac{\sum_i \sum_s \widetilde{Ta}_{i,s}}{10*N*L}$ Donde T_0 es un parámetro diferente de cero y no muy grande. (Shao, Shao, and Pi 2020)

Pseudocódigo de Python del algoritmo MNIG:

Algoritmo 4 MNIG

```
def MNIG(todas las T̃_{i,u}, d, T_0):
1:  |π_re3 = DNEH_SMR(todas las T̃_{i,u})
2:  |π_result = π_re3, π_temp = π_re3, iter = 1
3:  while (iter <= N^2 * L * UT):
```

```
4:      |π_temp = local_search(π_temp)
5:      if (C_max(π_temp) < C_max(π_re3)):
6:          |π_re3 = π_temp
7:          if (C_max(π_temp) < C_max(π_result)):
8:              |π_result = π_temp
9:      else:
10:         if (random(0,1) < e^(-(C_max(π_temp)-C_max(π_re3))/T)):
11:             |π_re3 = π_temp
12:     |π_temp = destruction_reconstruction(π_temp, d)
13: return π_result
```

Algoritmos BSA en general: Los algoritmos BSA, por sus siglas en inglés que significan Backtracking Search Algorithm, son algoritmos que al ser aplicados a problemas de Scheduling, constan de 5 pasos: inicialización, selección-I, mutación, cruzamiento, y selección-II. (Yan, Han, and Gu 2020)

La inicialización genera la población de secuencias. El paso selección-I actualiza la población de secuencias si es necesario, usando la memoria de la información histórica que guarda un algoritmo BSA. Dicha memoria también se usa para construir o cambiar la dirección de búsqueda de soluciones. El paso mutación genera una nueva población de secuencias mutando o alterando la población recibida del paso anterior. El paso cruzamiento cruza las secuencias unas con otras para obtener nuevas secuencias, descartando las secuencias no factibles. Por último, el paso de selección-II compara el makespan de las nuevas secuencias respecto a la población de secuencias del paso selección-I, y las mejores secuencias son elegidas para la iteración siguiente. (Yan, Han, and Gu 2020)

Tres de los cuatro algoritmos de comparación en este trabajo, son algoritmos de tipo BSA.

Algoritmo DBSA-LS: El algoritmo DBSA-LS, por sus siglas en inglés que significan Discrete Backtracking Search Algorithm with Local Search, es un algoritmo que busca aumentar la velocidad de convergencia del algoritmo BSA. Para ello, el algoritmo DBSA-LS utiliza la mejor secuencia de cada iteración, conocida como la secuencia Gbest. Aparte de esto, el algoritmo DBSA-LS utiliza búsqueda local para mejorar la de exploración de soluciones. (Yan, Han, and Gu 2020)

Algoritmo BDBSA: El algoritmo BDBSA, por sus siglas en inglés que significan Basic Discrete Backtracking Search Algorithm, es un algoritmo tipo BSA que aplica los mismos cinco pasos del algoritmo BSA a problemas de tiempo discreto. Este algoritmo puede ser aplicado a problemas de scheduling en los que los trabajos inician y terminan en unidades de tiempo discretos. (Yan, Han, and Gu 2020)

Algoritmo MBSA: El algoritmo MBSA, por sus siglas en inglés que significan Modified Backtracking Search Algorithm, es una forma modificada del algoritmo BSA. Esta modificación no cambia los cinco pasos

esenciales del algoritmo BSA. (Yan, Han, and Gu 2020)

Algoritmo IGA: El algoritmo IGA, por sus siglas en inglés que significan Interactive Genetic Algorithm, es un algoritmo genético que tiene algunas similaridades con el algoritmo BSA. Por ejemplo, en el algoritmo IGA las secuencias son modificadas mediante mutaciones aleatorias, y las mejores secuencias según el makespan pasan a la siguiente iteración. Esto también ocurre en el algoritmo BSA. (Yan, Han, and Gu 2020)

Esto termina la presentación del modelo FMMSP denotado como $FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}$, el algoritmo MNIG, y los cuatro algoritmos DBSA-LS, BDBSA, MBSA, e IGA. Con este marco de referencia se llevó a cabo el proyecto de grado.

HIPÓTESIS

A través del algoritmo MNIG se pueden encontrar soluciones para el modelo FMMSP $FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}$. Tras su aplicación la efectividad del algoritmo MNIG es medible y es comparable a la efectividad de los otros cuatro algoritmos: DBSA-LS, BDBSA, MBSA, e IGA.

DISEÑO METODOLÓGICO

El diseño metodológico de este proyecto abarca tres partes principales: escribir el algoritmo MNIG en Python, ejecutar el algoritmo MNIG en una instancia del modelo FMMSP, y comparar los resultados del algoritmo MNIG con los resultados de los otros cuatro algoritmos que ya han sido aplicados al modelo FMMSP (DBSA-LS, BDBSA, MBSA, e IGA).

Escritura del algoritmo MNIG en Python

El algoritmo MNIG se escribió en el lenguaje Python como una aplicación de consola. El interprete de Python empieza por ejecutar un archivo de texto plano de Python, en que se encuentran los parámetros del modelo $FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}$, en particular los tiempos de procesamiento, el número de etapas, el número de unidades por etapa. A su vez, los siguientes parámetros son pasados al programa por medio de la interfaz de línea de comandos, para así tener mayor facilidad para modificar estos parámetros y hacer pruebas: el parámetro de número de iteraciones N, el número de trabajos a remover en el algoritmo de destrucción y reconstrucción d, y el parámetro T_0.

La jerarquía de archivos del programa es:

```
MNIG_to_FMMSP/
  __main__.py
  API/
    functions.py
  algorithms/
    __init__.py
    DNEH_SMR.py
    destruction_reconstruction.py
    local_search.py
```

En esta estructura, cada algoritmo tiene su propio archivo, y el archivo que tiene el algoritmo MNIG como síntesis de los demás es el archivo `__main__.py` desde este archivo se llama a los demás algoritmos.

La carpeta `API/` contiene el archivo `functions.py`, en el cual se definen funciones comunes a todos los

algoritmos, por ejemplo la funcion `makespan()` que puede ser usada desde cualquier algoritmo que lo necesite, para calcular el makespan de una secuencia dada.

Usando la opción `--debug`, el programa va imprimiendo en la terminal resultados parciales de cada iteración, específicamente la secuencia encontrada con el mínimo makespan en esa iteración. Al finalizar la ejecución del programa, queda impreso en pantalla la secuencia final así como su makespan.

Como se observa en la estructura de archivos, la carpeta `algorithms/` es tratada como un paquete mediante el archivo `__init__.py` se podría decir que es el paquete con los algoritmos necesarios para ejecutar el algoritmo principal MNIG.

Ejecución del algoritmo MNIG en instancia del modelo FMMSP

Aquí se explica por qué sólo se trabajará una instancia y no varias del modelo FMMSP $FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}$. La razón es que sólo existe una instancia públicamente a nivel internacional. Ello porque aunque en la literatura se habla de varias instancias (de unas 15 instancias), sólo en 1 se dan los datos, en el resto se dice que fueron generadas con números aleatorios uniformes con ciertas condiciones, pero instancias creadas aleatoriamente no pueden ser replicadas con exactitud. (Yan, Han, and Gu 2020)

Por lo anterior se ejecutó el algoritmo MNIG en la única instancia conocida del modelo FMMSP $FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}$. En la literatura las instancias son nombradas con 3 letras y un número por letra, por ejemplo la instancia que se trabajará es "o10s2u5." En esta cadena de caracteres, "o" significa "ordenes" (o trabajos), "s" significa etapas (stages en inglés), "u" significa "unidades" (o máquinas). Por ende la instancia sobre la que se probará el algoritmo es de 10 trabajos, 2 etapas, y 5 unidades. La siguiente tabla contiene los datos de la instancia: (Yan, Han, and Gu 2020)

Tiempos de procesamiento de la instancia o10s2u5

Trabajo	**Unidad 1 Etapa 1**	**Unidad 2 Etapa 1**	**Unidad 3 Etapa 2**	**Unidad 4 Etapa 2**	**Unidad 5 Etapa 2**
1	(10,12,13)	(11,12,14)	(9,10,12)	(6,7,9)	(8,9,10)
2	(7,8,10)	(8,9,10)	(5,6,8)	(4,5,6)	(4,5,6)
3	(10,11,12)	(9,10,12)	(2,3,4)	(5,6,8)	(5,6,7)
4	(8,9,10)	(6,7,8)	(7,8,9)	(4,5,6)	(5,6,8)
5	(6,7,8)	(8,9,10)	(4,5,6)	(6,7,8)	(6,7,9)
6	(4,5,6)	(2,3,4)	(15,16,19)	(13,14,15)	(15,16,20)
7	(11,13,15)	(1,2,3)	(11,13,14)	(10,11,13)	(9,10,12)
8	(10,11,12)	(18,19,23)	(5,6,7)	(6,7,9)	(7,8,10)
9	(5,6,8)	(4,5,6)	(14,15,16)	(19,21,25)	(10,12,13)
10	(15,17,20)	(12,14,15)	(16,17,20)	(17,18,21)	(18,19,21)

Recordar que los tiempos de procesamiento vienen en ternas en la tabla pues son números difusos triangulares. El algoritmo fue ejecutado sobre esta instancia o10s2u5 y se compararon los resultados con los de los otros cuatro algoritmos.

Comparación del algoritmo MNIG con otros algoritmos

Tras aplicar el algoritmo MNIG a la instancia del modelo FMMSP y obtener resultados, se comparó con los otros cuatro algoritmos. Estos son: DBSA-LS, BDBSA, MBSA, e IGA. Estos cuatro algoritmos ya fueron aplicados a la instancia del modelo FMMSP. En la siguiente tabla se presentan los resultados de esos cuatro algoritmos: (Yan, Han, and Gu 2020)

Resultados del makespan de los cuatro algoritmos en la instancia o10s2u5

Algoritmo	**Mejor valor**	**Valor medio**	**Peor valor**
DBSA-LS	(36,44,52)	(38,45,52)	(42,48,53)
BDBSA	(36,44,52)	(38.6,45.5,52)	(40,46,54)
MBSA	(40,47,55)	(44,50,57)	(42.2,49.3,54.6)
IGA	(39,45,52)	(40.1,46.9,54.5)	(43,49,57)

Con esta tabla se compararon directamente los resultados del algoritmo MNIG al ser aplicado al modelo FMMSP $FFc|\tilde{T}_{i,u}, batch(b), Z_{i,u,s}|C_{max}$.

Instancias de Taillard

A nivel internacional, se reconocen algunas instancias con las que se puede medir la efectividad de las metaheurísticas. Para el caso flow shop, las instancias de Taillard son instancias que se han venido usando desde 1993 cuando fueron creadas, son en total 120 instancias. La más pequeña cuenta con 20 trabajos y 5 máquinas. La instancia más grande es de 500 trabajos y 20 máquinas. Se aplicó el algoritmo MNIG a las primeras 95 de estas instancias, aunque el algoritmo MNIG no esté diseñado para resolver el modelo flow shop básico. Se podría aplicar el algoritmo MNIG a las 25 instancias, pero hacerlo requeriría más de 24 horas por cada instancia, es decir, al menos 25 días de ejecución continua, y probablemente más. Aparte para analizar el algoritmo MNIG aplicado al problema flow shop básico, no necesariamente es obligatorio probar con todas las instancias.

RESULTADOS

Se corrió varias veces el algoritmo MNIG implementado en Python, para resolver la instancia o10s2u5 del modelo FMMSP. Las siguientes dos tablas presentan un resumen de los resultados encontrados al resolver la instancia o10s2u5.

Resumen de la solución obtenida por el algoritmo MNIG

Algoritmo	**Mejor valor**	**Valor medio**	**Peor valor**	**Tiempo promedio**
MNIG	(36,44,52)	(36,44,52)	(36,44,52)	0.83 segundos

Secuencias encontradas con la solución (36, 44, 52)

Secuencia
[7, 8, 6, 10, 1, 9, 5, 4, 2, 3]
[7, 8, 6, 10, 5, 9, 1, 4, 2, 3]
[6, 1, 9, 7, 8, 10, 4, 5, 2, 3]
[1, 6, 9, 10, 7, 8, 5, 4, 2, 3]
[6, 5, 10, 1, 9, 7, 8, 4, 2, 3]
[6, 10, 1, 8, 9, 4, 5, 7, 2, 3]

Secuencia
[6, 5, 7, 10, 8, 1, 4, 9, 2, 3]
[9, 7, 1, 6, 8, 10, 4, 5, 2, 3]
[5, 8, 7, 6, 10, 4, 1, 9, 2, 3]
[7, 1, 6, 10, 5, 9, 8, 4, 2, 3]

Todas estas secuencias tienen un makespan de (36, 44, 52).

Análisis de resultados

El algoritmo MNIG se corrió más de 30 veces en la instancia o10s2u5 del modelo FMMSP, y se midió el tiempo de ejecución de esas 30 veces para promediarlo. Se encontró un tiempo de ejecución promedio de 0.83 segundos, con un máximo de 0.85 segundos y un mínimo de 0.82 segundos.

Para poder comparar el tiempo de ejecución, debe tenerse en cuenta que el algoritmo MNIG fue ejecutado en una máquina Ubuntu Linux, con un CPU Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz.

De la primera tabla de resultados, se observa que el algoritmo MNIG aporta consistentemente la misma solución (36, 44, 52), de la instancia o10s2u5. En todas las ejecuciones, el algoritmo MNIG aportó la misma solución, por lo que la peor y mejor solución, son la misma. Esto muestra una mejora por parte del algoritmo MNIG en comparación de los otros cuatro algoritmos.

Comparación de los algoritmos

Algoritmo	Mejor valor	Valor medio	Peor valor	% de mejora del MNIG
MNIG	(36,44,52)	(36,44,52)	(36,44,52)	0.00%
DBSA-LS	(36,44,52)	(38,45,52)	(42,48,53)	2.22%
BDBSA	(36,44,52)	(38.6,45.5,52)	(40,46,54)	3.08%
MBSA	(40,47,55)	(44,50,57)	(42.2,49.3,54.6)	12.44%
IGA	(39,45,52)	(40.1,46.9,54.5)	(43,49,57)	6.58%

El algoritmo MNIG tiene como mejor valor, valor medio, y peor valor, la misma solución (36, 44, 52). Es importante denotar que la solución (36, 44, 52) se alcanzaba con tan solo una iteración. Consistentemente basta una sola iteración del algoritmo MNIG para encontrar la solución (36, 44, 52). Se analizó cuál parte del programa es la que convierte cualquier secuencia en una secuencia con makespan (36, 44, 52), y resultó ser la parte voraz del algoritmo MNIG, que es el algoritmo `destruction_reconstruction.py`. Al algoritmo `destruction_reconstruction.py` se le puede dar cualquier secuencia de la instancia o10s2u5, y retornará una secuencia con makespan (36, 44, 52) en una sola iteración. Como se verá más adelante en las intancias de Taillard, la parte más lenta del algoritmo MNIG también es el algoritmo `destruction_reconstruction.py`, tanto que en las instancias más grandes de Taillard, puede tardar más de 24 horas en hacer una iteración.

La columna denominada '% de mejora del MNIG,' contiene los porcentajes con los que el algoritmo MNIG mejora la solución promedio, comparado a los demás algoritmos. Por ejemplo, el algoritmo MNIG mejora la solución promedio un 2.22% respecto al algoritmo DBSA-LS. Para calcular este porcentaje, se usa la ponderación que permite comparar cual es mayor, menor, o igual, entre dos números triangulares difusos. Por ejemplo, para calcular ese 2.22%, la solución (36, 44, 52) se pondera como (36 + 2*44 + 52)/4 = 44, y (38 + 2*45 + 52)/4 = 45. Luego, la mejora porcentual es (45 - 44)/45 = 2.22%, y así con los demás porcentajes.

Estos porcentajes muestran que el algoritmo MNIG mejora la solución promedio al menos un 2.22% y máximo un 12.44%, en comparación con los algoritmos DBSA-LS, BDBSA, MBSA, e IGA, para la instancia o10s2u5 del modelo FMMSP.

Resultados de las instancias de Taillard

El algoritmo MNIG se aplicó a las primeras 95 instancias de Taillard. Se decidió no aplicarlo a las últimas 25 instancias (son 120 en total), porque esta implementación del algoritmo MNIG se tardaba más de 24 horas por instancia.

Resultados de las instancias de Taillard

Instancia	**Makespan**	**Secuencia**	**% sobre cota superior**
1	1297	[9, 15, 6, 4, 11, 13, 1, 5, 17, 14, 2, 8, 19, 7, 18, 12, 3, 16, 10, 20]	1.49%
2	1366	[6, 7, 18, 15, 9, 12, 8, 20, 14, 13, 16, 5, 4, 19, 10, 11, 3, 17, 1, 2]	0.52%
3	1100	[3, 16, 14, 20, 7, 4, 18, 10, 12, 1, 11, 19, 5, 9, 17, 6, 13, 8, 2, 15]	1.76%
4	1315	[13, 11, 17, 15, 19, 9, 16, 2, 20, 7, 12, 10, 1, 5, 3, 8, 6, 18, 14, 4]	1.70%
5	1250	[12, 3, 19, 10, 9, 4, 6, 5, 16, 17, 13, 2, 7, 11, 15, 1, 14, 18, 8, 20]	1.13%
6	1210	[14, 11, 17, 13, 5, 20, 6, 9, 4, 7, 8, 12, 15, 18, 1, 10, 16, 19, 2, 3]	1.26%
7	1251	[5, 11, 20, 13, 1, 9, 17, 19, 2, 7, 16, 6, 15, 4, 3, 14, 8, 18, 12, 10]	0.97%
8	1206	[17, 2, 16, 9, 5, 14, 12, 1, 3, 19, 7, 13, 10, 18, 8, 20, 4, 15, 6, 11]	0.00%
9	1255	[4, 7, 20, 12, 16, 1, 15, 17, 18, 3, 8, 10, 2, 6, 9, 13, 14, 5, 11, 19]	2.03%
10	1127	[5, 16, 11, 6, 13, 10, 8, 14, 1, 19, 12, 3, 17, 18, 2, 20, 7, 4, 15, 9]	1.71%
11	1597	[5, 2, 12, 9, 10, 15, 4, 14, 19, 17, 13, 3, 18, 6, 8, 20, 11, 7, 1, 16]	0.95%
12	1687	[17, 19, 15, 12, 13, 11, 20, 10, 1, 5, 9, 7, 2, 4, 16, 8, 14, 3, 6, 18]	1.69%
13	1538	[4, 7, 16, 12, 13, 9, 5, 1, 11, 2, 6, 14, 3, 10, 17, 15, 18, 20, 19, 8]	2.81%
14	1397	[3, 12, 20, 16, 8, 18, 4, 1, 10, 9, 2, 7, 6, 15, 13, 11, 14, 17, 19, 5]	1.38%
15	1443	[16, 8, 18, 4, 13, 14, 12, 6, 19, 1, 20, 15, 3, 11, 17, 2, 9, 7, 5, 10]	1.69%
16	1414	[18, 19, 8, 3, 17, 16, 11, 14, 6, 5, 20, 13, 4, 7, 12, 10, 9, 15, 2, 1]	1.22%
17	1510	[19, 4, 6, 7, 9, 13, 17, 18, 1, 2, 10, 20, 5, 16, 12, 14, 11, 3, 8, 15]	1.75%
18	1566	[10, 19, 7, 9, 4, 16, 8, 3, 18, 14, 1, 15, 20, 5, 13, 6, 11, 17, 2, 12]	1.82%
19	1630	[14, 12, 20, 4, 2, 8, 1, 19, 3, 13, 10, 11, 18, 16, 15, 7, 17, 5, 6, 9]	2.32%
20	1618	[5, 13, 7, 17, 8, 6, 16, 20, 19, 4, 10, 2, 15, 14, 18, 1, 9, 12, 11, 3]	1.70%
21	2327	[16, 18, 14, 10, 8, 9, 15, 3, 2, 11, 6, 12, 13, 5, 1, 20, 4, 17, 7, 19]	1.31%
22	2114	[18, 11, 13, 3, 10, 4, 20, 19, 12, 5, 6, 14, 15, 16, 1, 9, 8, 7, 17, 2]	0.67%
23	2357	[19, 5, 17, 16, 4, 1, 2, 15, 13, 10, 14, 3, 20, 9, 6, 18, 8, 11, 12, 7]	1.33%
24	2235	[14, 18, 5, 2, 8, 6, 4, 20, 15, 13, 1, 12, 7, 3, 17, 19, 9, 16, 11, 10]	0.54%
25	2313	[10, 2, 9, 3, 16, 17, 15, 1, 18, 13, 4, 20, 7, 19, 11, 12, 14, 5, 8, 6]	0.96%
26	2242	[6, 14, 18, 13, 3, 5, 16, 8, 15, 9, 1, 10, 11, 20, 2, 17, 4, 7, 12, 19]	0.72%
27	2297	[10, 12, 5, 9, 11, 20, 4, 15, 6, 19, 2, 14, 1, 18, 16, 7, 17, 8, 3, 13]	1.06%
28	2224	[4, 10, 17, 5, 16, 14, 7, 2, 13, 12, 3, 11, 6, 20, 19, 8, 18, 15, 9, 1]	1.09%
29	2258	[1, 8, 14, 2, 13, 17, 4, 6, 3, 7, 18, 11, 9, 15, 20, 10, 12, 16, 5, 19]	0.94%
30	2205	[3, 7, 12, 6, 13, 18, 10, 17, 11, 5, 15, 19, 1, 8, 20, 2, 9, 16, 4, 14]	1.24%

Instancia	Makespan	Secuencia	% sobre cota superior
31	2729	[26, 39, 20, 30, 49, 17, 22, 4, 1, 43, 28, 46, 47, 25, 31, 40, 34, 29, 10, 24, 36, 11, 14, 6, 2, 32, 42, 38, 5, 27, 18, 35, 8, 44, 13, 9, 45, 33, 21, 7, 15, 16, 12, 50, 23, 41, 48, 19, 37, 3]	0.18%
32	2863	[50, 37, 33, 8, 2, 46, 24, 10, 9, 47, 17, 45, 42, 5, 41, 26, 7, 22, 40, 14, 18, 43, 29, 35, 27, 15, 39, 31, 20, 25, 6, 32, 3, 34, 48, 49, 30, 16, 28, 19, 11, 1, 36, 21, 4, 23, 44, 13, 12, 38]	1.02%
33	2646	[4, 13, 45, 3, 18, 26, 9, 24, 40, 20, 22, 2, 37, 25, 38, 42, 49, 33, 41, 10, 34, 29, 35, 5, 32, 16, 1, 19, 46, 36, 27, 28, 17, 50, 6, 11, 31, 8, 15, 39, 7, 23, 48, 14, 43, 47, 21, 44, 30, 12]	0.95%
34	2755	[42, 30, 50, 14, 12, 25, 13, 19, 35, 24, 7, 23, 38, 36, 5, 20, 47, 10, 31, 8, 28, 1, 3, 26, 11, 9, 22, 29, 46, 16, 45, 2, 41, 49, 17, 48, 15, 40, 33, 18, 27, 6, 37, 34, 39, 4, 32, 21, 43, 44]	0.15%
35	2887	[48, 5, 32, 1, 28, 21, 37, 12, 27, 40, 25, 11, 4, 35, 31, 34, 14, 43, 3, 44, 20, 22, 47, 16, 2, 29, 9, 26, 8, 23, 18, 36, 17, 41, 46, 24, 50, 7, 13, 45, 15, 39, 38, 19, 42, 49, 6, 10, 33, 30]	0.84%
36	2836	[47, 29, 35, 34, 30, 32, 15, 5, 23, 21, 28, 33, 24, 10, 13, 39, 49, 17, 37, 6, 2, 18, 20, 46, 8, 4, 19, 1, 11, 3, 38, 45, 40, 44, 48, 22, 41, 7, 9, 50, 43, 14, 36, 42, 31, 27, 25, 26, 16, 12]	0.25%
37	2736	[22, 43, 9, 49, 45, 5, 21, 24, 46, 40, 30, 33, 12, 39, 18, 3, 41, 15, 7, 35, 37, 14, 26, 10, 25, 8, 23, 32, 44, 20, 4, 38, 29, 17, 31, 19, 6, 11, 36, 34, 47, 27, 2, 48, 50, 42, 13, 1, 16, 28]	0.40%
38	2704	[8, 25, 35, 14, 31, 39, 37, 5, 28, 18, 41, 10, 1, 17, 33, 45, 43, 47, 16, 49, 48, 36, 21, 46, 20, 2, 19, 30, 29, 7, 23, 3, 34, 4, 24, 44, 15, 12, 27, 11, 26, 13, 9, 6, 32, 42, 22, 50, 38, 40]	0.78%
39	2579	[1, 15, 36, 38, 48, 5, 17, 41, 47, 44, 33, 46, 3, 18, 30, 43, 11, 49, 9, 20, 25, 34, 2, 24, 32, 29, 16, 28, 22, 42, 19, 7, 50, 40, 13, 26, 37, 39, 21, 8, 27, 23, 35, 12, 6, 31, 45, 14, 10, 4]	1.06%
40	2783	[6, 35, 16, 24, 1, 18, 48, 44, 17, 38, 34, 4, 40, 19, 30, 49, 3, 39, 11, 8, 42, 50, 31, 36, 27, 2, 7, 15, 47, 28, 45, 12, 29, 9, 14, 37, 21, 43, 23, 10, 5, 41, 33, 13, 46, 25, 20, 32, 26, 22]	0.04%
41	3079	[22, 19, 26, 18, 44, 37, 20, 34, 31, 42, 10, 9, 15, 28, 30, 46, 33, 16, 38, 40, 36, 4, 32, 12, 17, 7, 3, 2, 48, 29, 41, 23, 14, 47, 21, 6, 25, 49, 43, 35, 27, 11, 1, 13, 50, 8, 5, 45, 24, 39]	1.78%
42	2963	[35, 49, 3, 33, 28, 12, 18, 10, 42, 34, 32, 31, 23, 14, 11, 7, 50, 1, 39, 38, 43, 37, 6, 27, 46, 47, 9, 24, 19, 8, 21, 20, 36, 16, 30, 25, 13, 29, 5, 4, 26, 17, 48, 44, 40, 15, 41, 2, 45, 22]	2.46%
43	2959	[10, 24, 4, 28, 33, 38, 3, 25, 47, 11, 29, 42, 18, 37, 7, 44, 45, 1, 32, 26, 35, 20, 40, 30, 31, 17, 9, 39, 27, 43, 41, 13, 50, 16, 5, 6, 14, 49, 2, 15, 19, 46, 48, 36, 21, 8, 12, 34, 22, 23]	3.32%
44	3116	[20, 5, 39, 13, 22, 31, 41, 35, 28, 19, 14, 2, 25, 8, 36, 9, 18, 48, 26, 6, 46, 1, 4, 23, 11, 47, 21, 37, 45, 40, 3, 34, 16, 24, 29, 10, 12, 43, 15, 38, 49, 17, 27, 32, 44, 7, 50, 33, 30, 42]	1.70%

Instancia	Makespan	Secuencia	% sobre cota superior
45	3160	[1, 40, 11, 49, 33, 16, 42, 35, 12, 48, 45, 30, 23, 43, 46, 5, 20, 6, 17, 39, 31, 14, 27, 10, 8, 15, 3, 24, 18, 34, 38, 22, 2, 41, 21, 36, 4, 7, 29, 25, 26, 47, 28, 44, 19, 37, 9, 13, 32, 50]	5.83%
46	3159	[16, 42, 2, 24, 28, 27, 3, 10, 20, 45, 33, 46, 5, 48, 44, 11, 19, 30, 25, 38, 9, 36, 15, 29, 47, 34, 14, 32, 43, 39, 40, 23, 6, 31, 8, 17, 37, 22, 49, 26, 50, 13, 7, 1, 18, 41, 21, 35, 12, 4]	5.09%
47	3297	[9, 37, 1, 2, 42, 30, 12, 49, 7, 6, 15, 28, 11, 17, 21, 33, 13, 3, 50, 19, 41, 45, 5, 26, 48, 14, 16, 8, 39, 10, 44, 35, 20, 36, 27, 23, 18, 47, 4, 43, 32, 31, 22, 25, 40, 46, 34, 29, 24, 38]	6.12%
48	3174	[6, 18, 47, 11, 45, 33, 28, 15, 50, 48, 41, 14, 2, 29, 4, 46, 13, 22, 9, 40, 26, 24, 43, 25, 34, 35, 1, 27, 12, 32, 7, 10, 44, 30, 21, 36, 37, 8, 42, 5, 31, 16, 20, 39, 3, 38, 19, 17, 23, 49]	4.44%
49	3097	[18, 44, 22, 13, 4, 46, 6, 23, 42, 17, 50, 37, 39, 48, 33, 27, 38, 11, 20, 41, 40, 14, 16, 21, 45, 3, 31, 19, 8, 12, 32, 10, 34, 49, 5, 26, 35, 29, 9, 30, 25, 7, 2, 24, 28, 1, 47, 36, 43, 15]	6.72%
50	3195	[6, 1, 11, 27, 17, 9, 7, 20, 26, 46, 32, 49, 44, 40, 10, 48, 29, 23, 28, 43, 38, 36, 16, 25, 30, 24, 5, 15, 41, 50, 8, 34, 19, 35, 12, 14, 22, 31, 21, 37, 3, 18, 45, 39, 47, 13, 4, 42, 33, 2]	3.36%
51	4089	[35, 17, 31, 24, 33, 12, 30, 20, 50, 44, 8, 40, 45, 15, 6, 13, 34, 26, 43, 42, 7, 23, 38, 14, 46, 29, 4, 27, 36, 5, 11, 39, 16, 2, 49, 10, 32, 25, 48, 41, 22, 37, 19, 28, 47, 1, 9, 21, 18, 3]	5.52%
52	3990	[37, 11, 45, 18, 40, 1, 17, 12, 21, 42, 20, 14, 36, 41, 29, 47, 8, 38, 3, 50, 43, 2, 24, 5, 15, 39, 10, 13, 22, 16, 4, 28, 23, 9, 44, 34, 33, 26, 19, 31, 25, 32, 30, 6, 35, 48, 49, 7, 46, 27]	7.40%
53	3845	[28, 22, 24, 31, 3, 39, 49, 4, 8, 16, 5, 9, 17, 18, 2, 19, 10, 33, 46, 30, 34, 29, 15, 48, 50, 12, 11, 41, 21, 1, 43, 36, 42, 20, 37, 35, 23, 32, 25, 26, 47, 14, 38, 45, 6, 13, 27, 40, 44, 7]	4.83%
54	3978	[20, 34, 3, 32, 50, 11, 30, 13, 35, 1, 29, 45, 19, 4, 33, 8, 46, 48, 31, 17, 43, 44, 49, 36, 21, 14, 42, 6, 15, 5, 47, 26, 9, 27, 2, 18, 39, 12, 7, 38, 37, 22, 23, 10, 24, 40, 25, 41, 28, 16]	6.02%
55	3893	[48, 2, 20, 19, 5, 4, 27, 21, 44, 43, 28, 31, 40, 18, 34, 23, 45, 50, 38, 33, 9, 29, 11, 30, 46, 16, 25, 22, 41, 49, 10, 26, 36, 42, 13, 8, 47, 14, 17, 1, 24, 39, 3, 6, 7, 37, 12, 35, 32, 15]	7.10%
56	3971	[21, 50, 40, 18, 11, 39, 47, 16, 14, 36, 35, 20, 45, 24, 42, 4, 22, 5, 49, 34, 30, 6, 31, 7, 2, 27, 9, 33, 37, 46, 32, 10, 26, 8, 3, 13, 29, 12, 15, 17, 28, 41, 44, 1, 25, 19, 48, 23, 43, 38]	7.38%
57	3968	[20, 15, 4, 45, 38, 46, 19, 1, 47, 33, 11, 14, 8, 5, 18, 50, 27, 23, 12, 49, 39, 40, 29, 2, 34, 31, 25, 37, 9, 3, 28, 42, 32, 6, 22, 41, 26, 13, 43, 10, 17, 48, 21, 16, 7, 30, 36, 44, 35, 24]	6.78%
58	3994	[47, 21, 26, 24, 40, 41, 49, 32, 29, 31, 22, 1, 17, 37, 36, 15, 48, 2, 18, 13, 16, 34, 45, 35, 19, 5, 9, 30, 27, 38, 50, 8, 20, 43, 12, 25, 42, 4, 33, 14, 7, 10, 6, 28, 11, 3, 46, 39, 44, 23]	7.68%

Instancia	**Makespan**	**Secuencia**	**% sobre cota superior**
59	4008	[3, 15, 28, 32, 6, 8, 17, 11, 42, 41, 12, 1, 46, 37, 40, 5, 45, 21, 9, 25, 13, 27, 47, 43, 4, 30, 31, 7, 2, 26, 23, 10, 16, 22, 33, 50, 19, 34, 24, 14, 39, 38, 48, 20, 49, 36, 18, 29, 44, 35]	6.45%
60	3923	[33, 12, 19, 15, 37, 50, 29, 2, 8, 38, 3, 18, 25, 14, 47, 48, 20, 23, 40, 31, 32, 1, 16, 9, 30, 10, 11, 41, 22, 42, 35, 27, 44, 6, 49, 36, 5, 24, 21, 45, 28, 17, 39, 43, 26, 7, 13, 46, 4, 34]	3.87%
61	5527	[42, 50, 69, 11, 27, 49, 64, 38, 25, 22, 56, 90, 81, 7, 43, 70, 85, 29, 12, 93, 14, 2, 15, 16, 5, 24, 73, 8, 59, 78, 86, 77, 28, 46, 87, 1, 36, 100, 95, 71, 13, 19, 4, 45, 44, 41, 53, 92, 51, 21, 37, 48, 31, 57, 99, 80, 55, 23, 94, 54, 10, 79, 83, 91, 6, 60, 82, 74, 52, 84, 34, 47, 40, 62, 68, 72, 33, 89, 20, 98, 63, 61, 35, 75, 58, 39, 18, 96, 3, 26, 17, 9, 67, 65, 66, 30, 76, 97, 32, 88]	0.62%
62	5316	[7, 94, 80, 67, 48, 6, 22, 81, 40, 86, 63, 97, 24, 39, 29, 1, 27, 84, 43, 38, 92, 64, 62, 47, 4, 50, 49, 14, 3, 69, 26, 54, 11, 21, 59, 30, 90, 78, 13, 12, 33, 58, 15, 75, 32, 19, 89, 100, 41, 95, 9, 73, 93, 55, 68, 5, 98, 87, 8, 99, 36, 91, 60, 66, 44, 2, 74, 88, 79, 76, 10, 35, 83, 65, 45, 53, 37, 57, 20, 42, 51, 28, 82, 71, 77, 34, 61, 16, 96, 70, 56, 18, 23, 31, 25, 46, 72, 17, 85, 52]	0.91%
63	5221	[14, 25, 74, 99, 80, 62, 37, 1, 30, 6, 54, 100, 70, 82, 46, 7, 81, 47, 69, 83, 32, 5, 15, 42, 3, 12, 18, 63, 2, 31, 85, 84, 87, 36, 40, 65, 44, 79, 33, 93, 57, 72, 41, 96, 68, 94, 9, 39, 45, 86, 38, 91, 98, 23, 76, 77, 10, 20, 22, 53, 27, 24, 49, 75, 66, 8, 55, 29, 21, 34, 67, 19, 89, 88, 59, 43, 92, 61, 4, 58, 26, 71, 64, 17, 35, 56, 78, 48, 97, 95, 73, 50, 51, 16, 90, 52, 11, 28, 60, 13]	0.89%
64	5044	[20, 92, 10, 6, 66, 75, 25, 36, 12, 18, 47, 70, 49, 28, 98, 55, 30, 32, 79, 21, 9, 82, 54, 40, 87, 67, 41, 57, 44, 39, 2, 51, 77, 27, 16, 74, 72, 4, 69, 45, 65, 33, 89, 53, 62, 1, 35, 52, 81, 90, 50, 96, 11, 63, 73, 19, 15, 100, 38, 22, 26, 84, 8, 95, 88, 31, 34, 68, 59, 56, 43, 76, 91, 80, 46, 60, 7, 83, 24, 29, 61, 37, 99, 23, 17, 48, 3, 5, 94, 93, 64, 85, 58, 71, 86, 13, 78, 97, 14, 42]	0.60%
65	5311	[21, 78, 54, 52, 86, 59, 12, 48, 51, 94, 80, 71, 45, 34, 30, 89, 96, 11, 24, 72, 26, 65, 37, 10, 14, 67, 38, 5, 69, 70, 64, 33, 84, 44, 39, 20, 74, 92, 19, 35, 47, 87, 60, 50, 61, 6, 93, 99, 55, 1, 66, 9, 41, 32, 95, 15, 56, 75, 42, 28, 57, 17, 2, 76, 22, 40, 36, 46, 85, 79, 88, 43, 27, 18, 49, 25, 53, 7, 8, 3, 91, 73, 90, 98, 4, 68, 16, 29, 82, 31, 83, 23, 77, 100, 62, 97, 81, 13, 58, 63]	1.16%
66	5161	[73, 49, 81, 22, 6, 72, 47, 38, 88, 37, 68, 85, 24, 10, 80, 63, 74, 79, 82, 3, 83, 59, 50, 19, 51, 34, 15, 84, 11, 62, 93, 30, 40, 92, 21, 7, 96, 69, 99, 100, 16, 94, 45, 71, 5, 67, 75, 42, 89, 32, 12, 23, 48, 2, 14, 8, 33, 76, 1, 91, 55, 53, 39, 98, 65, 43, 25, 70, 57, 44, 66, 31, 95, 17, 35, 52, 28, 46, 56, 4, 41, 61, 29, 78, 90, 13, 27, 77, 87, 20, 86, 54, 18, 97, 58, 60, 64, 26, 9, 36]	0.51%

Instancia	Makespan	Secuencia	% sobre cota superior
67	5355	[17, 85, 37, 71, 96, 86, 75, 48, 9, 73, 35, 15, 2, 6, 45, 89, 28, 8, 78, 91, 36, 26, 83, 54, 32, 68, 95, 16, 59, 27, 33, 40, 30, 52, 13, 51, 60, 53, 87, 12, 38, 98, 55, 14, 77, 50, 3, 23, 41, 97, 93, 74, 80, 67, 19, 57, 65, 81, 63, 61, 92, 94, 84, 46, 18, 10, 70, 20, 34, 64, 69, 49, 29, 39, 82, 72, 7, 24, 100, 22, 11, 5, 4, 25, 42, 58, 56, 90, 47, 31, 76, 43, 62, 88, 79, 44, 1, 99, 66, 21]	2.08%
68	5141	[80, 91, 73, 79, 77, 71, 97, 56, 62, 44, 31, 24, 49, 18, 57, 2, 87, 55, 100, 54, 47, 89, 25, 45, 23, 59, 64, 13, 99, 92, 51, 35, 14, 41, 42, 82, 46, 98, 76, 26, 85, 17, 52, 22, 83, 69, 32, 74, 21, 68, 53, 39, 72, 60, 65, 29, 38, 48, 50, 11, 67, 75, 30, 8, 33, 86, 9, 66, 6, 70, 61, 12, 78, 10, 20, 27, 81, 19, 96, 16, 4, 63, 84, 15, 88, 95, 93, 28, 1, 40, 58, 43, 94, 34, 37, 5, 7, 36, 3, 90]	0.69%
69	5536	[34, 98, 75, 62, 64, 33, 44, 31, 67, 79, 14, 39, 16, 71, 11, 37, 88, 10, 28, 5, 38, 70, 46, 87, 58, 2, 15, 6, 99, 36, 85, 22, 94, 74, 91, 8, 66, 57, 95, 55, 26, 89, 18, 19, 77, 76, 97, 45, 35, 63, 86, 68, 12, 83, 100, 56, 20, 84, 51, 59, 54, 42, 40, 96, 61, 4, 48, 23, 43, 73, 49, 72, 47, 92, 52, 41, 30, 69, 50, 90, 60, 80, 24, 9, 29, 25, 7, 53, 82, 81, 65, 17, 93, 32, 3, 21, 27, 1, 78, 13]	1.50%
70	5386	[24, 19, 73, 69, 31, 30, 36, 51, 83, 82, 74, 78, 77, 64, 4, 84, 39, 99, 59, 20, 94, 100, 33, 6, 15, 41, 91, 70, 79, 48, 55, 90, 81, 18, 25, 11, 86, 14, 57, 62, 26, 42, 43, 21, 56, 8, 22, 61, 96, 49, 3, 87, 16, 88, 76, 17, 29, 23, 63, 53, 28, 80, 50, 93, 89, 71, 67, 40, 10, 75, 13, 72, 97, 95, 54, 34, 12, 66, 65, 35, 98, 1, 46, 45, 68, 52, 5, 7, 32, 37, 44, 9, 60, 27, 58, 47, 92, 2, 85, 38]	1.09%
71	5999	[56, 17, 61, 39, 55, 85, 95, 38, 24, 79, 9, 70, 20, 44, 35, 94, 47, 64, 78, 57, 33, 50, 34, 87, 73, 66, 42, 74, 29, 6, 31, 89, 49, 88, 28, 90, 41, 81, 27, 98, 77, 93, 4, 99, 80, 7, 53, 100, 60, 43, 18, 97, 8, 51, 10, 82, 86, 69, 1, 54, 76, 3, 2, 83, 96, 16, 52, 14, 84, 59, 91, 58, 71, 21, 25, 68, 23, 48, 11, 15, 19, 67, 32, 26, 36, 22, 92, 65, 30, 37, 62, 46, 5, 45, 13, 72, 40, 63, 75, 12]	3.97%
72	5481	[54, 98, 22, 87, 12, 5, 49, 60, 32, 31, 10, 72, 6, 1, 74, 52, 55, 80, 24, 27, 59, 3, 62, 93, 69, 34, 66, 82, 35, 78, 96, 25, 90, 43, 85, 65, 46, 81, 37, 29, 88, 41, 9, 71, 28, 95, 75, 76, 68, 8, 13, 36, 20, 73, 44, 97, 16, 15, 89, 45, 61, 58, 64, 47, 11, 4, 86, 77, 83, 50, 38, 14, 21, 26, 42, 40, 56, 79, 91, 67, 48, 92, 63, 17, 18, 99, 51, 57, 53, 33, 19, 100, 94, 70, 30, 39, 2, 84, 23, 7]	2.47%
73	5926	[49, 90, 58, 11, 32, 52, 7, 68, 62, 2, 100, 50, 37, 47, 3, 83, 65, 76, 56, 86, 64, 75, 66, 14, 40, 12, 20, 57, 94, 16, 96, 61, 54, 77, 84, 1, 4, 67, 8, 44, 87, 24, 13, 27, 30, 82, 23, 51, 59, 5, 33, 91, 38, 74, 25, 93, 22, 60, 69, 99, 53, 78, 73, 28, 29, 80, 72, 31, 95, 45, 71, 70, 10, 34, 92, 35, 85, 9, 43, 21, 81, 19, 41, 17, 18, 55, 98, 39, 6, 42, 88, 26, 79, 89, 48, 36, 63, 97, 15, 46]	4.39%

Instancia	Makespan	Secuencia	% sobre cota superior
74	6086	[42, 71, 96, 77, 47, 23, 5, 92, 99, 78, 97, 59, 72, 93, 53, 56, 26, 58, 34, 4, 40, 87, 12, 45, 28, 30, 14, 55, 15, 66, 52, 81, 2, 86, 74, 49, 10, 18, 80, 11, 51, 8, 1, 70, 38, 17, 67, 9, 19, 76, 16, 57, 79, 29, 48, 20, 73, 27, 88, 75, 83, 95, 54, 63, 61, 22, 24, 36, 37, 35, 25, 31, 60, 64, 43, 44, 62, 90, 100, 91, 6, 32, 3, 39, 21, 89, 65, 84, 68, 46, 50, 94, 33, 82, 7, 13, 98, 69, 41, 85]	5.09%
75	5651	[79, 56, 75, 33, 66, 87, 27, 4, 58, 83, 48, 45, 51, 76, 63, 96, 30, 94, 88, 42, 44, 80, 12, 55, 81, 14, 68, 59, 100, 86, 64, 38, 62, 34, 99, 73, 39, 8, 97, 7, 84, 6, 18, 1, 2, 10, 54, 60, 90, 49, 91, 36, 17, 67, 20, 61, 69, 74, 95, 53, 21, 89, 24, 5, 28, 19, 25, 47, 52, 11, 9, 41, 40, 65, 72, 35, 43, 70, 23, 98, 92, 31, 71, 13, 22, 46, 85, 57, 77, 16, 29, 78, 50, 26, 15, 93, 37, 32, 3, 82]	3.35%
76	5417	[78, 53, 2, 76, 69, 55, 9, 86, 41, 22, 62, 35, 47, 30, 68, 85, 73, 32, 59, 79, 4, 5, 80, 39, 54, 71, 1, 50, 93, 94, 7, 26, 77, 99, 66, 75, 74, 52, 60, 49, 88, 97, 87, 89, 21, 63, 12, 20, 10, 3, 6, 11, 46, 42, 34, 70, 16, 98, 8, 57, 64, 37, 15, 81, 51, 27, 58, 28, 84, 91, 90, 14, 72, 67, 92, 96, 29, 31, 13, 43, 83, 95, 38, 48, 24, 44, 36, 65, 19, 23, 17, 56, 33, 45, 40, 18, 61, 100, 25, 82]	2.15%
77	5749	[95, 3, 50, 36, 5, 64, 20, 57, 70, 98, 32, 30, 92, 22, 62, 25, 68, 56, 86, 74, 40, 41, 99, 11, 47, 82, 51, 42, 87, 78, 14, 21, 71, 35, 65, 34, 43, 69, 18, 53, 37, 16, 31, 60, 44, 55, 72, 24, 46, 2, 66, 15, 88, 9, 100, 39, 28, 90, 54, 26, 77, 12, 79, 23, 4, 8, 29, 59, 63, 93, 10, 52, 1, 83, 38, 27, 96, 84, 13, 76, 19, 7, 33, 45, 91, 17, 67, 97, 89, 94, 6, 81, 75, 48, 61, 58, 80, 49, 85, 73]	2.68%
78	5807	[67, 6, 92, 78, 48, 75, 9, 26, 55, 52, 13, 56, 43, 22, 19, 58, 20, 12, 99, 30, 88, 28, 68, 2, 32, 100, 21, 82, 35, 95, 14, 5, 89, 63, 18, 76, 72, 65, 59, 77, 91, 93, 4, 85, 96, 87, 97, 50, 83, 69, 64, 74, 45, 29, 60, 51, 66, 42, 86, 81, 90, 3, 23, 10, 33, 17, 73, 41, 80, 62, 38, 8, 27, 34, 16, 11, 40, 1, 31, 54, 39, 71, 44, 7, 57, 36, 47, 98, 61, 24, 53, 70, 37, 46, 15, 79, 49, 25, 94, 84]	3.27%
79	6033	[87, 40, 61, 39, 86, 100, 10, 43, 68, 54, 38, 35, 57, 78, 80, 3, 29, 82, 13, 77, 15, 95, 69, 41, 64, 6, 18, 81, 19, 60, 12, 37, 24, 70, 27, 5, 14, 34, 46, 25, 93, 72, 92, 22, 32, 26, 53, 1, 58, 83, 71, 8, 7, 52, 42, 9, 51, 59, 97, 21, 30, 91, 17, 48, 45, 28, 49, 56, 16, 89, 23, 84, 62, 94, 4, 73, 98, 44, 31, 66, 88, 65, 2, 11, 74, 55, 20, 96, 75, 36, 50, 33, 47, 76, 63, 79, 67, 85, 99, 90]	2.69%
80	5969	[63, 66, 37, 71, 92, 83, 98, 13, 11, 42, 22, 35, 97, 8, 49, 57, 38, 54, 26, 9, 45, 80, 78, 27, 60, 39, 69, 20, 25, 5, 43, 2, 31, 82, 75, 79, 50, 70, 30, 85, 58, 87, 93, 65, 21, 6, 46, 76, 72, 18, 33, 41, 32, 48, 64, 15, 94, 40, 16, 81, 59, 55, 19, 17, 7, 1, 95, 88, 29, 10, 12, 56, 100, 90, 89, 86, 28, 51, 52, 47, 4, 67, 96, 34, 44, 74, 61, 62, 3, 24, 23, 73, 84, 14, 77, 36, 99, 53, 68, 91]	2.12%

Instancia	Makespan	Secuencia	% sobre cota superior
81	6550	[47, 22, 54, 79, 10, 40, 78, 71, 39, 86, 43, 3, 51, 83, 82, 16, 37, 61, 99, 74, 1, 33, 48, 89, 65, 12, 91, 90, 63, 88, 31, 9, 21, 28, 59, 93, 53, 96, 35, 15, 50, 58, 85, 75, 17, 30, 23, 56, 26, 62, 81, 34, 42, 80, 24, 18, 76, 19, 55, 44, 57, 20, 67, 95, 6, 29, 8, 2, 41, 4, 5, 32, 94, 11, 60, 14, 97, 100, 45, 92, 27, 46, 49, 70, 36, 52, 38, 87, 68, 13, 98, 77, 66, 7, 73, 72, 64, 25, 84, 69]	4.20%
82	6560	[49, 19, 62, 46, 68, 70, 97, 14, 18, 2, 23, 72, 71, 50, 66, 54, 36, 5, 69, 67, 10, 34, 89, 31, 30, 33, 90, 38, 100, 35, 59, 79, 63, 41, 26, 95, 27, 60, 85, 15, 48, 53, 96, 22, 86, 40, 9, 7, 16, 61, 51, 83, 8, 75, 73, 3, 64, 84, 44, 78, 1, 58, 76, 92, 32, 43, 52, 47, 56, 93, 82, 12, 25, 28, 65, 57, 21, 13, 45, 87, 4, 91, 24, 29, 98, 11, 42, 20, 94, 17, 77, 6, 74, 80, 99, 37, 88, 81, 39, 55]	5.11%
83	6628	[87, 48, 57, 23, 29, 8, 96, 10, 24, 58, 20, 31, 64, 70, 66, 88, 72, 5, 37, 1, 13, 49, 32, 90, 54, 97, 43, 95, 17, 67, 51, 25, 9, 91, 44, 61, 16, 52, 3, 53, 28, 19, 75, 45, 86, 12, 41, 77, 39, 40, 98, 4, 47, 42, 60, 15, 27, 11, 18, 65, 36, 76, 89, 21, 94, 2, 69, 84, 80, 56, 81, 30, 68, 59, 22, 100, 34, 93, 83, 74, 38, 78, 26, 99, 50, 62, 92, 35, 85, 33, 6, 82, 46, 7, 71, 55, 79, 14, 73, 63]	4.72%
84	6484	[74, 5, 57, 60, 8, 1, 84, 68, 38, 46, 55, 50, 13, 79, 88, 51, 69, 9, 72, 58, 23, 52, 44, 43, 21, 71, 56, 15, 99, 2, 11, 91, 16, 96, 61, 89, 10, 98, 93, 26, 54, 22, 45, 49, 78, 86, 87, 18, 14, 67, 94, 37, 41, 77, 92, 27, 70, 19, 35, 33, 31, 95, 62, 63, 76, 47, 4, 90, 85, 29, 32, 48, 53, 75, 25, 39, 20, 66, 7, 6, 12, 34, 73, 17, 28, 42, 3, 24, 64, 100, 59, 83, 82, 30, 81, 65, 40, 97, 36, 80]	2.82%
85	6597	[14, 38, 61, 97, 64, 56, 53, 69, 33, 32, 3, 18, 5, 99, 30, 71, 98, 72, 89, 51, 10, 52, 65, 1, 58, 94, 28, 7, 21, 13, 40, 90, 8, 92, 78, 41, 9, 100, 50, 95, 16, 70, 19, 47, 88, 46, 12, 79, 77, 80, 11, 17, 81, 22, 20, 31, 2, 39, 74, 49, 29, 54, 25, 37, 73, 55, 34, 76, 15, 45, 68, 59, 66, 82, 35, 75, 86, 85, 83, 60, 23, 4, 6, 93, 48, 43, 36, 26, 24, 27, 67, 87, 91, 44, 42, 57, 96, 84, 63, 62]	3.45%
86	6677	[56, 83, 86, 54, 39, 8, 68, 37, 70, 12, 82, 55, 2, 29, 10, 100, 52, 34, 65, 89, 14, 72, 85, 38, 77, 88, 50, 58, 94, 69, 62, 1, 66, 99, 64, 92, 76, 25, 23, 24, 98, 3, 67, 49, 81, 43, 27, 40, 80, 96, 15, 87, 35, 6, 9, 75, 93, 53, 78, 63, 22, 36, 32, 97, 44, 33, 18, 31, 7, 60, 57, 74, 20, 48, 4, 13, 91, 26, 16, 71, 61, 79, 84, 51, 73, 17, 5, 47, 11, 21, 45, 28, 41, 42, 95, 19, 59, 46, 30, 90]	3.73%
87	6640	[57, 83, 67, 7, 85, 59, 93, 80, 32, 19, 33, 73, 16, 23, 50, 41, 51, 20, 94, 28, 48, 47, 14, 24, 96, 11, 22, 62, 72, 55, 39, 89, 36, 31, 21, 40, 12, 98, 91, 42, 5, 97, 78, 34, 53, 27, 25, 43, 88, 95, 10, 35, 65, 1, 13, 9, 69, 64, 79, 6, 18, 90, 2, 84, 56, 92, 54, 29, 61, 45, 82, 38, 100, 75, 70, 46, 52, 30, 15, 76, 81, 3, 49, 17, 8, 4, 37, 58, 60, 77, 86, 71, 66, 26, 87, 99, 44, 68, 74, 63]	4.63%

Instancia	Makespan	Secuencia	% sobre cota superior
88	6936	[28, 100, 58, 81, 62, 44, 90, 51, 18, 53, 57, 87, 45, 47, 68, 23, 98, 3, 76, 78, 41, 99, 83, 25, 24, 6, 22, 52, 64, 89, 93, 54, 36, 55, 30, 70, 67, 59, 38, 50, 39, 33, 72, 82, 12, 49, 69, 31, 85, 97, 14, 1, 29, 9, 7, 4, 10, 17, 27, 42, 74, 21, 5, 60, 79, 13, 46, 32, 63, 15, 91, 75, 84, 71, 73, 43, 16, 19, 86, 88, 37, 77, 94, 2, 8, 96, 48, 95, 56, 40, 11, 92, 26, 35, 65, 80, 34, 66, 20, 61]	7.02%
89	6632	[88, 43, 89, 92, 41, 66, 78, 65, 33, 29, 7, 45, 16, 26, 62, 27, 15, 95, 94, 63, 97, 30, 2, 3, 81, 60, 76, 48, 21, 34, 19, 64, 58, 46, 32, 61, 80, 17, 51, 18, 53, 44, 74, 82, 25, 68, 75, 42, 100, 57, 79, 99, 38, 28, 77, 47, 49, 40, 59, 72, 11, 20, 73, 39, 14, 54, 8, 6, 1, 4, 12, 70, 90, 50, 85, 22, 93, 24, 23, 13, 84, 86, 71, 9, 10, 31, 55, 83, 37, 69, 67, 56, 87, 35, 52, 96, 5, 91, 36, 98]	4.31%
90	6785	[83, 65, 6, 77, 38, 67, 36, 43, 46, 17, 40, 42, 35, 88, 92, 71, 75, 52, 14, 87, 16, 41, 99, 23, 10, 70, 74, 21, 48, 51, 15, 56, 54, 20, 12, 22, 57, 69, 37, 4, 47, 94, 61, 90, 86, 68, 5, 82, 96, 78, 63, 76, 33, 34, 9, 64, 60, 13, 32, 2, 7, 30, 26, 62, 95, 27, 45, 49, 24, 3, 100, 84, 25, 8, 80, 79, 72, 73, 1, 85, 18, 89, 39, 28, 19, 93, 91, 98, 53, 97, 50, 11, 81, 31, 29, 66, 55, 44, 59, 58]	4.95%
91	11012	[66, 13, 147, 14, 152, 131, 71, 174, 59, 84, 193, 150, 121, 134, 70, 16, 127, 89, 49, 142, 189, 148, 143, 162, 54, 107, 106, 42, 175, 132, 1, 190, 41, 108, 88, 115, 10, 27, 166, 29, 179, 194, 38, 163, 192, 181, 75, 74, 86, 145, 151, 35, 160, 188, 23, 11, 79, 91, 126, 196, 56, 85, 118, 6, 139, 172, 2, 164, 48, 62, 173, 169, 191, 24, 161, 140, 68, 176, 167, 83, 185, 19, 200, 67, 111, 46, 156, 87, 124, 198, 18, 77, 141, 117, 153, 36, 28, 197, 113, 7, 81, 17, 43, 15, 94, 44, 96, 73, 120, 60, 76, 82, 159, 9, 119, 129, 32, 155, 51, 187, 157, 58, 165, 33, 53, 20, 123, 186, 138, 149, 80, 98, 144, 154, 109, 5, 30, 69, 50, 110, 95, 40, 99, 52, 55, 37, 8, 182, 125, 146, 199, 31, 57, 21, 103, 170, 184, 47, 90, 133, 3, 136, 104, 45, 12, 100, 64, 101, 65, 22, 34, 135, 128, 177, 130, 97, 93, 78, 25, 72, 137, 195, 102, 180, 105, 116, 171, 63, 168, 61, 112, 122, 26, 92, 4, 39, 183, 114, 158, 178]	1.32%
92	10625	[68, 117, 1, 45, 184, 114, 157, 7, 54, 156, 87, 133, 171, 20, 146, 9, 119, 5, 110, 165, 128, 42, 21, 43, 83, 63, 194, 131, 176, 59, 94, 147, 84, 191, 90, 139, 35, 170, 19, 198, 66, 49, 13, 192, 50, 52, 32, 46, 144, 3, 60, 169, 115, 57, 27, 155, 74, 112, 150, 75, 81, 187, 123, 200, 65, 61, 55, 15, 71, 37, 177, 102, 140, 145, 130, 2, 181, 89, 168, 80, 25, 134, 132, 48, 154, 95, 97, 6, 160, 17, 56, 161, 197, 153, 174, 99, 38, 58, 189, 120, 72, 190, 107, 113, 47, 179, 137, 22, 138, 62, 31, 159, 136, 188, 158, 14, 166, 178, 67, 44, 16, 26, 40, 64, 23, 108, 182, 164, 91, 148, 135, 92, 193, 163, 172, 121, 141, 183, 185, 33, 4, 125, 151, 11, 126, 180, 85, 18, 124, 70, 142, 88, 34, 152, 101, 111, 30, 73, 103, 175, 129, 104, 173, 10, 116, 105, 76, 98, 195, 93, 24, 118, 167, 77, 143, 29, 149, 122, 96, 69, 82, 12, 162, 186, 109, 199, 28, 106,	1.25%

Instancia	Makespan	Secuencia	% sobre cota superior
		127, 41, 78, 8, 51, 86, 36, 39, 196, 53, 100, 79]	
93	11159	[183, 83, 62, 136, 133, 48, 155, 6, 151, 166, 46, 30, 144, 132, 99, 100, 114, 111, 127, 125, 130, 147, 141, 51, 94, 126, 20, 57, 196, 1, 4, 14, 49, 107, 63, 74, 8, 87, 113, 29, 18, 70, 69, 12, 178, 186, 102, 128, 139, 148, 77, 117, 188, 98, 104, 7, 187, 157, 197, 103, 35, 175, 78, 39, 152, 52, 146, 105, 37, 164, 124, 84, 68, 184, 73, 191, 86, 32, 131, 27, 58, 192, 185, 121, 65, 59, 140, 110, 172, 200, 171, 17, 36, 3, 115, 85, 80, 177, 150, 169, 93, 189, 97, 95, 198, 44, 190, 165, 129, 195, 159, 67, 108, 21, 92, 75, 10, 120, 156, 38, 181, 81, 167, 145, 106, 154, 25, 193, 109, 23, 153, 72, 149, 135, 15, 182, 34, 66, 31, 11, 82, 28, 138, 45, 60, 16, 161, 137, 55, 174, 2, 123, 91, 13, 118, 199, 143, 168, 88, 33, 71, 24, 41, 61, 50, 116, 40, 54, 112, 163, 180, 173, 160, 79, 53, 47, 119, 76, 122, 42, 43, 96, 9, 170, 158, 90, 134, 194, 101, 179, 56, 176, 162, 22, 89, 64, 26, 5, 19, 142]	2.17%
94	11010	[43, 168, 69, 73, 191, 183, 38, 197, 55, 42, 68, 132, 147, 95, 120, 78, 158, 37, 29, 49, 101, 151, 17, 145, 14, 138, 146, 127, 134, 195, 48, 118, 10, 5, 133, 176, 88, 102, 157, 70, 59, 190, 173, 90, 113, 28, 192, 107, 25, 47, 83, 3, 66, 162, 56, 184, 172, 20, 7, 85, 35, 36, 135, 185, 137, 89, 177, 74, 196, 193, 72, 63, 9, 152, 2, 149, 167, 65, 23, 27, 93, 51, 30, 163, 104, 110, 26, 182, 108, 154, 126, 58, 114, 131, 194, 39, 148, 91, 119, 62, 169, 106, 124, 45, 155, 100, 200, 189, 75, 46, 112, 186, 94, 6, 175, 32, 16, 141, 98, 179, 19, 92, 166, 50, 115, 116, 12, 178, 81, 71, 180, 181, 15, 80, 165, 123, 60, 117, 24, 136, 187, 64, 103, 1, 128, 40, 96, 97, 139, 31, 67, 8, 111, 86, 150, 76, 170, 34, 61, 41, 87, 160, 121, 11, 153, 79, 174, 122, 129, 130, 140, 164, 44, 144, 57, 18, 4, 105, 52, 77, 199, 159, 82, 99, 161, 188, 142, 22, 171, 33, 54, 125, 198, 156, 109, 53, 143, 21, 84, 13]	1.11%
95	10625	[99, 163, 61, 193, 170, 45, 25, 180, 106, 142, 198, 192, 21, 144, 80, 76, 185, 135, 29, 43, 75, 59, 63, 115, 138, 129, 95, 83, 11, 5, 116, 46, 188, 20, 71, 8, 169, 74, 158, 191, 87, 153, 73, 177, 40, 148, 100, 37, 154, 130, 189, 152, 41, 111, 162, 49, 66, 134, 122, 131, 18, 60, 151, 166, 82, 147, 104, 47, 113, 114, 194, 110, 14, 190, 150, 53, 137, 44, 17, 89, 62, 67, 187, 23, 56, 7, 68, 26, 199, 178, 97, 3, 128, 173, 81, 38, 54, 133, 98, 77, 161, 78, 39, 196, 168, 9, 140, 143, 120, 72, 86, 55, 93, 27, 156, 85, 92, 1, 96, 145, 124, 126, 32, 102, 64, 123, 127, 181, 34, 197, 112, 70, 58, 10, 186, 103, 94, 42, 33, 36, 108, 160, 164, 15, 159, 157, 50, 30, 184, 57, 119, 79, 200, 141, 121, 13, 175, 2, 52, 105, 179, 165, 12, 24, 91, 172, 109, 118, 171, 174, 4, 176, 16, 101, 51, 48, 167, 125, 155, 65, 132, 19, 182, 6, 22, 69, 149, 183, 88, 107, 139, 136, 31, 146, 28, 117, 195, 35, 84, 90]	0.96%

El porcentaje sobre cota superior, tiene un promedio de: 2.58%, y una desviación estándar de: 2.07%. Este porcentaje significa el porcentaje que el makespan encontrado por el algoritmo MNIG, tiene por sobre la cota superior de cada instancia de Taillard.

Cada instancia de Taillard tiene una cota inferior y una cota superior para el makespan. En este trabajo ninguna de las soluciones encontradas por el algoritmo MNIG se encontró dentro del rango creado entre la cota inferior y la cota superior, sino que todas las soluciones fueron iguales o estuvieron por encima de la cota superior.

El porcentaje sobre cota superior mide el porcentaje de diferencia entre la solución encontrada por el algoritmo MNIG, y la cota superior, de una instancia dada. Por ejemplo en la instancia de Taillard 1, la cota superior es 1278, y la solución encontrada por el algoritmo MNIG tiene el valor de 1297, por lo que la diferencia porcentual es (1297 - 1278)/1278 = 1.49%. Este porcentaje de 1.49% es el que se observa en la tabla.

Se debe aclarar que para hallar estos porcentajes, el algoritmo MNIG fue aplicado con 10 iteraciones a las 95 instancias. Con un mayor número de iteraciones, se pueden encontrar mejores soluciones. Por ejemplo para la instancia de Taillard 1, se probó con 250 iteraciones, y se encontró un makespan de 1278, que es igual que la cota superior para esa instancia. Sin embargo no se aplicaron 250 iteraciones para las 95 instancias, porque hacerlo habrían tomado al menos 25 veces el tiempo que tomó realizar las 95 instancias con 10 iteraciones cada una (esto tomó un tiempo de más de 10 días seguidos de ejecución).

Estos resultados muestran que el algoritmo MNIG también se puede usar para encontrar soluciones en el sistema flow shop básico, pero como este algoritmo fue diseñado para el modelo FMMSP, esta implementación del algoritmo MNIG es demorado para resolver el problema flow shop básico, sobre todo en instancias de gran cantidad de trabajos y máquinas (por ejemplo 500 trabajos y 20 máquinas, instancias de este tamaño requerirían varios días de ejecución por iteración, en esta implementación del algoritmo).

CONCLUSIONES Y RECOMENDACIONES

A futuro se podría mejorar el algoritmo para tratar otros problemas de scheduling. Con este trabajo se puede concluir que no hay impedimento alguno para adaptar el algoritmo MNIG para tratar problemas job shop, u otros problemas de scheduling. Para ello se requiería entre otras cosas, cambiar la función `makespan` que en este caso calcula el makespan en un sistema del tipo FMMSP.

Otra posible mejora para esta implementación del algoritmo MNIG, es la de mejorar el tiempo de ejecución para el problema flow shop básico, y de este modo poder probar las últimas instancias de Taillard sin tener que esperar más de un día por iteración. Es interesante notar cómo el mismo algoritmo MNIG resuelve la instancia o10s2u5 del modelo FMMSP en menos de un segundo (en promedio en 0.83 segundos).

Como conclusión final, este trabajo evidencia que el algoritmo MNIG sí es competitivo para resolver el problema FMMSP, quod erat demonstrandum.

GLOSARIO DE ANGLICISMOS

Scheduling: Rama de estudio de la secuenciación de trabajos en un sistema productivo.

Makespan: Tiempo de terminación máximo de una secuencia.

Greedy algorithm: Tipo de algoritmo que en español se le llama algoritmo voraz.

Fuzzy number: Número difuso.

REFERENCIAS

Anand, M. Clement, and Janani Bharatraj. 2017. “Theory of Triangular Fuzzy Number.” In.

Garey, M., D. Jhonson, and R. Sethi. 1976. “The Complexity of Flowshop and Jobshop Scheduling.” *Mathematics of Operations Research* 1 (2): 117, 129.

Pinedo, Michael. 2016. *Scheduling, Theory, Algorithms, and Systems*. 5th ed. New York, USA: Springer.

Shao, Weishi, Zhongshi Shao, and Dechang Pi. 2020. “Modeling and Multi Neighborhood Iterated Greedy Algorithm for Distributed Hybrid Flow Shop Scheduling Problem.” *Knowledge-Based Systems*. https://doi.org/https://doi.org/10.1016/j.knosys.2020.105527.

Taillard, E. 1993. “Benchmarks for Basic Scheduling Problems.” *EJOR* 64 (2): 278, 285.

Talbi, El-Ghazali. 2009. *Metaheuristics, From Design to Implementation*. 1st ed. USA: Wiley.

Yan, Xueli, Yuxin Han, and Xingsheng Gu. 2020. “An Improved Discrete Backtracking Searching Algorithm for Fuzzy Multiproduct Multistage Scheduling Problem.” *Neurocomputing*. https://doi.org/https://doi.org/10.1016/j.neucom.2020.02.066.