Computational study on scheduling problem using genetic algorithm

Estudio computacional sobre problemas de programación mediante algoritmo genético

Nelly Juan[1], Sary Marin[2], Alexis Iranzo[3]
[1]Department of Mathematics, Universidad Católica del Norte, Chile
[2]Department of Statistics,Universidad de Valparaiso, Chile
[3]Department of Mathematics, Universidad de Talca, Chile

Abstracto
Un problema de programación consiste en una serie de etapas de producción, cada una de las cuales tiene varias máquinas funcionando en paralelo. Algunas etapas pueden tener solo una máquina, pero al menos una etapa debe tener varias máquinas. Cada trabajo es procesado por una máquina en cada etapa y debe pasar por una o más etapas.

Palabras clave: problema de programación, algoritmo genético, computación, optimización

Abstract
A scheduling problem consists of series of production stages, each of which has several machines operating in parallel. Some stages may have only one machine, but at least one stage must have multiple machines. Each job is processed by one machine in each stage and it must go through one or more stages.

Keywords: Scheduling problem, Genetic algorithm, computation, optimization

1.Introducción

Un problema de programación HFS consta de una serie de etapas de producción, cada una de las cuales tiene varias máquinas que funcionan en paralelo[1]. Algunas etapas pueden tener solo una máquina, pero al menos una etapa debe tener varias máquinas. Cada trabajo es procesado por una máquina en cada etapa y debe pasar por una o más etapas. Las máquinas en cada etapa pueden ser idénticos, uniformes o no relacionados. Los problemas de programación del taller de flujo híbrido puede describirse formalmente como sigue[2]: Las máquinas se organizan en etapas en series; en cada etapa k (k = 1,…., s) hay mk máquinas idénticas en paralelo; trabajo j, j = 1,…., n, tiene para ser procesado en cualquier máquina en cada etapa y el trabajo j tiene tiempos de procesamiento finitos en cada etapa (p1j, p2j,…., psj)[3]. Asumimos que todos los trabajos y máquinas están siempre disponibles durante el período programado y la preferencia no está permitida. El objetivo es encontrar un horario que minimice el máximo tiempo de finalización (makespan). Los problemas de HFS son NP-Hard cuando el objetivo es minimizar el makespan[4].

Un problema de programación del taller de flujo híbrido El modelo matemático se describe como un entero mixto. programación. El modelo se da como sigue[5].

Restricción indica que la finalización El tiempo del último trabajo en la última etapa s es Q. La restricción indica que no es posible para el trabajo j se procesará en la etapa l + 1 antes del trabajo j en la etapa l está completa. El orden de procesamiento de los trabajos f yg en la máquina i en la etapa l se define por restricciones. Estas limitaciones[6] No permita que se procese más de un trabajo en una máquina en cualquier momento. La restricción no permitir que un trabajo se procese en más de una máquina en cualquier momento. Restricciones proporcionar que las variables no sean negativas y 0-1 valores enteros[7].

2.Revisión de literature

Un problema de HFS con 5 trabajos × 3 etapas donde el etapas tienen 2, 3 y 3 máquinas respectivamente es dado.

El HFS fue estudiado por primera vez por Arthanari y Ramamurthy. Desarrollaron una rama y algoritmo enlazado para resolver problemas HFS. Gupta y Hoogeveen demostró que el El problema de programación del taller de flujo híbrido de dos etapas fue NP-Hard en el sentido más fuerte, incluso si solo hubiera una máquina en la primera etapa y dos máquinas en la segunda etapa. Los otros estudios sobre el híbrido Los problemas de programación del taller de flujo en la literatura son dado[8].

Wang propuso un híbrido eficaz algoritmo genético (HGA) para la tienda de flujo de permutación programación con búferes limitados. En la HGA, no sólo múltiples operadores genéticos basados en mecanismo evolutivo se

utilizan simultáneamente en sentido híbrido, sino también una estructura de vecindario basado en el modelo gráfico se emplea para mejorar la búsqueda local, de modo que la exploración y las habilidades de explotación pueden estar bien equilibradas[9]. Ellos investigó los efectos del tamaño del búfer y la decisión probabilidad de rendimiento de optimización utilizando simulación.

Oğuz y Ercan propusieron una AG para programación híbrida flow-shop con multiprocesador problemas de la tarea y describió su implementación. Mejoraron un nuevo operador de cruce para ser utilizado en los GA y lo comparó con Parcialmente coincidente Crossover (PMX). También emplearon un prueba preliminar para establecer la mejor combinación de los parámetros de control que se utilizarán junto con diferentes operadores genéticos[10].

Li-Xin desarrolló una descendencia genética algoritmo para el problema de programación del taller de flujo híbrido. 230 instancias generadas aleatoriamente fueron probadas por programa de simulación. Experimentos computacionales mostrar que para la programación HFSS de tamaño pequeño problemas, la desviación promedio de GDA de la la solución óptima es 0,01%; para tamaño mediano-grande problemas, el rendimiento de GDA es 10,45% mejor que el del algoritmo NEH[11].

Xia propuso un enfoque GA para Problema de programación del taller de flujo híbrido. los El algoritmo se basa en el principio de programación de listas. desarrollando secuencias de trabajo para la primera etapa y poner en cola las etapas restantes en forma FIFO[12].

En este documento, se desarrolla una AG eficaz para Problemas de programación de HFS. La efectividad del El método propuesto se prueba con los problemas de programación HFS de Carlier y Neron del literatura. Los resultados computacionales indican que el enfoque propuesto es eficaz en términos de espacio reducido para los problemas intentados. A lo mejor de nuestro conocimiento, no hay genética algoritmos aplicados al taller de flujo híbrido que incluyen Los problemas de programación de Carlier y Neron en la literature[13].

Los GA fueron inventados por John Holland y fueron métodos de búsqueda estocásticos diseñado para buscar espacios grandes y complejos por explotación de soluciones actualmente conocidas y exploración robusta de todo el espacio de búsqueda[14].

Los GA utilizan una colección de soluciones llamadas población. Cada individuo de la población es llamado cromosoma (una cadena de símbolos) y un El cromosoma representa una solución al problema. Los cromosomas se pueden producir a través de iteraciones sucesivas, llamadas generaciones y el tamaño de la población (el número de individuos en un población) permanece fija de generación en Generacion[15]. Los cromosomas se evalúan utilizando el valor de la función de aptitud durante cada Generacion. Un conjunto de operadores genéticos como reproducción (selección) y recombinación (crossover y mutación) se aplica para crear nuevos y mejores soluciones (derivadas) de la individuos de la población actual y el las soluciones se mejoran constantemente de generación en Generacion. La estructura de los GA se muestra.

3.Discusión

La GA propuesta se basa en una permutación representación de los n puestos de trabajo. Los detalles de nuestro la implementación de los GA se indica a continuación.

Se utiliza un enfoque de codificación directa. En esto codificación, un cromosoma representa un horario directamente. La inicial la población se genera aleatoriamente. La población el tamaño se determina con la ayuda de un factorial completo diseño experimental utilizando nuestro programa GA[16].

Los esquemas de selección permiten que el algoritmo tome decisiones sesgadas que favorecen las buenas cuerdas cuando las generaciones cambian. Para este fin, algunos de los buenos las cadenas se replican mientras que algunas de las cadenas defectuosas remoto. Como consecuencia, después de la selección mecanismo está determinado, la población es probable ser “dominado” por buenas cuerdas. Varios Se han utilizado esquemas de selección en la literatura. Nos centramos en la selección de la rueda de la ruleta y Selección de torneo sin reemplazo.

La función fitness juega un papel importante en decidiendo la cadena en la próxima generación. La función de aptitud de una cuerda está definida por la Makepan (Cmax) valor de la programación.

El crossover se utiliza como principal operador genético. y el desempeño de un GA depende en gran medida en eso. Durante las últimas tres décadas, varios Se han propuesto operadores de cruce para el problemas de programación. En este estudio, seis crossover Se han utilizado operadores: Basado en posición Crossover (PBX),

Order Crossover (OX), Parcialmente Crossover mapeado (PMX), Crossover de ciclo (CX), Cruce de orden lineal (LOX) y basado en orden Crossover (OBX) que se utilizan ampliamente en el literatura. Estos seis operadores cruzados son brevemente explicado a continuación:

Primero, se genera una máscara aleatoria y luego genes relativos intercambiados entre padres según la máscara. Este operador se explica y detallado.

La descendencia hereda los elementos entre los dos puntos de cruce del padre seleccionado en el mismo orden y posición que aparecen en el padre. Los elementos restantes se heredan de el padre alterno en el orden en que aparecen en ese padre, comenzando con el primer posición después del segundo punto de cruce y omitiendo todos los elementos ya presentes en el apagado primavera.

Se seleccionan un sitio principal y dos sitios cruzados aleatoriamente y los elementos entre dos cadenas las posiciones en uno de los padres se heredan directamente por la descendencia. Cada elemento entre los dos los puntos de cruce en el padre alternativo se asignan a la posición ocupada por este elemento en el primer padre. Entonces los elementos restantes se heredan del padre alterno.

El ciclo entre cuerdas está multado, los símbolos en el ciclo se adaptan a una nueva cuerda, el los símbolos restantes se determinan para el nuevo cadena eliminando los símbolos y el resto los símbolos se cumplen con la nueva cadena.

Las dos sublistas se seleccionan de cadenas al azar. Sublist se elimina de string1, dejando algunos "agujeros" y luego los agujeros se deslizan desde el extremidades hacia el centro hasta que alcanzan el sección transversal. Del mismo modo, Sublist1 se elimina de cadena2. Al final, Sublist1 se inserta en los agujeros de string2 para formar offspring1 y se inserta sublist2 en los orificios de la cuerda1 para formar descendencia.

Se selecciona un conjunto de posiciones al azar; el orden de símbolos en las posiciones seleccionadas se imponen en el símbolos correspondientes en la otra cadena.

El operador de mutación juega un papel muy importante en GA y ayuda a mantener la diversidad en el población para evitar una convergencia prematura. Seis Los operadores de mutación se examinan en el GA para Minimizar la fabricación en HFS. Estos son basado en el vecindario, adyacente a dos cambios de trabajo, cambio arbitrario de dos trabajos, cambio arbitrario de tres trabajos, operador de cambio de turno y mutación de inversión.

Es bien sabido que la eficiencia de los GA depende de un alto grado en la selección del control parámetros. El proceso de búsqueda de los GA se controla con múltiples factores (parámetros de control) cuyos efectos posiblemente interactuarán entre sí. En general, hay algunos mecanismos de control para estos parámetros y en este artículo el factorial completo Se utiliza el diseño de experimentos (DOE). los La aplicación implica seis parámetros (factores), cada uno teniendo posibles valores diferentes. Estos parámetros se dan en la Tabla 2.

Los problemas de referencia dados en Carlier y Neron se consideran en el estudio. los el tamaño del problema varía de 10 trabajos × 5 etapas a 15 trabajo × 10 etapas. Los tiempos de procesamiento tienen un uniforme distribución en el rango de (3, 20). Tres las características que definen un problema son no. de trabajos, No. de etapas y no. de máquinas idénticas en cada etapa. Un total de 77 problemas se clasifican en 13 grupos según sus características. Un El problema de instancia se toma de cada uno de los grupos. Se implementa la optimización de parámetros y la mejor se encuentra el conjunto de parámetros para la instancia. los el conjunto de parámetros encontrado para una instancia está generalizado y se utiliza para los demás problemas del mismo grupo. Por lo tanto, se implementa la optimización de parámetros por 13 instancias. En el estudio, dos selecciones métodos, diez niveles de relación de selección, seis cruces métodos, diez niveles de relación de cruce, cinco mutaciones métodos, y diez niveles de tasa de mutación son implementado entre los 13 problemas. Un total número de 2x10x6x10x6x10 = 72 000 carreras son hecho entre estos problemas. Los mejores parámetros establecido en cada una de las ejecuciones replicadas para 13 puntos de referencia Los problemas se dan. El La población se selecciona como 25 para todos los parámetros problemas.

La mejor selección, cruce y mutación. Los métodos para 13 problemas de referencia son brevemente descrito de la siguiente manera:

Se elige la rueda de la ruleta, donde el se calcula la aptitud media de cada cromosoma dependiendo de la aptitud total del conjunto población. Los cromosomas son aleatorios seleccionados proporcionalmente a su aptitud media. La selección de la rueda de la ruleta se resume en el siguientes pasos,

Paso 1. Deje que el tamaño de pop, el número de cadenas en popular.
Paso 2. nsum, suma de todos los valores de aptitud de las cuerdas en pop; formar ranuras nsum y asignar cadena a las ranuras de acuerdo con el valor de aptitud del cuerda.
Paso 3. Repita el paso 4 (tamaño pop -1) veces.
Paso 4. Genera un número aleatorio entre 1 y nsum, y utilícelo para indexar en las ranuras para encontrar la cadena correspondiente; agregue esta cadena a newpop
Paso 5. Agregue la cuerda con la mayor aptitud valor en pop a newpop.

(i) Seleccione un conjunto de posiciones de una cadena en aleatorio,
(ii) Produzca una nueva cadena copiando los símbolos en estas posiciones en el correspondiente posiciones en la nueva cadena,
(iii) Eliminar los símbolos ya seleccionados de la segunda cuerda. La secuencia resultante contiene solo los símbolos que necesita la nueva cadena,
(iv) Coloque los símbolos en posiciones no fijas en el nueva cadena de izquierda a derecha según el orden de la secuencia utilizada para producir uno descendencia.

Puede verse que la inversión mutación selecciona dos posiciones al azar y luego intercambia los genes en estas posiciones.

Los problemas de prueba utilizados en los experimentos son Problemas de referencia de Carlier. Los mismos problemas también fueron estudiados por Santos et Alabama. Engin y Döyen Alaykiran. Santos utilizó una rama y método enlazado, Engin utilizaron un método del sistema inmunológico artificial (AIS) que fue mejorado con el uso de pruebas de satisfacción y ajustes de duración determinada y también Alaykiran utilizaron el método de optimización de colonias de hormigas. Santos, Engin y Döyen laykiran limitaron su algoritmo con 1600 s. Si no se encontró una solución óptima dentro 1600, la búsqueda se detuvo y la mejor solución fue aceptado como programa final. Ellos calcularon los límites inferiores (LB) de los problemas y la brecha relativa de estos límites para los no óptimos instancias resueltas.

El número de iteración se selecciona como 1000 y solo uno replicado para todos los problemas de referencia. Además, el tiempo de la CPU está limitado a 1600 s. Si un óptimo la solución no se encuentra dentro de este tiempo, la búsqueda es detenido y la mejor solución se acepta como el horario final. El algoritmo se implementa en Borland Delphi y ejecutar en una PC Pentium 4 Procesador con 3 GHz y 512 MB de memoria. En, para todos los 77 problemas, la mejor Cmax valores y tiempos de CPU obtenidos por la propuesta Modelo GA, modelo AIS de Engin y Döyen, y se presentan el modelo B&B de Neron. Para todos los métodos (GA, AIS y B&B) la CPU los tiempos se dan en segundos. Los límites inferiores y Los% de desviaciones de los límites inferiores se dan al últimas tres columnas.

Como se verá en la Tabla 4, mejores resultados para problemas de tipo ayb que el tipo c y d Se han obtenido problemas. La máquina configuraciones tienen un efecto importante en la complejidad de los problemas que afectan la solución calidad. El algoritmo GA ha encontrado las soluciones óptimas para todos los problemas de tipo ayb como AIS algoritmo (47 problemas), aunque B&B ha encontrado las soluciones óptimas para 46 problemas. tipo cyd Los problemas son problemas relativamente difíciles. Neron agruparon algunos de los problemas como difíciles problemas. Por estos problemas, no pudieron llegar las soluciones óptimas en poco tiempo. La diferencia de estos problemas se debe principalmente a su configuraciones de la máquina (todos estos problemas son c od problemas de tipo d). Hay 30 problemas en eso grupo (los tipos cyd de 10x5 y 15x5 problemas). El resto de los problemas (todos los tipos a, b y problemas de tipo 10x10 c) se denominan fáciles problemas.

Para problemas difíciles, el algoritmo GA propuesto encontró valores de LB para 18 de los 24 problemas mientras AIS encontró valores de LB para 17 de los 24 problemas. También para estos problemas difíciles, GA encontró un mejor Makepan valor que los métodos AIS y B&B. por solo dos problemas, GA no pudo alcanzar el AIS hace el valor de pan. Estos problemas están representados en negrita.

El% de desviación promedio de LB para GA es más pequeño que los métodos AIS y B&B ". Hay 53 problemas fáciles. Ambos métodos, AIS y B&B, no pudo alcanzar los valores de LB para 6 de los problemas. Pero el% de desviación promedio de LB para El algoritmo GA es más pequeño que AIS y B&B métodos'. En, el porcentaje de los resueltos problemas y los valores de% de desviación promedio para Se presentan problemas fáciles y difíciles.

Como se ve claramente el menor la desviación pertenece a GA. AIS es el segundo con un 0.08% de diferencia. B&B es el peor de todos con desviación casi dos veces mayor que las demás.

También se comparan los resultados computacionales con el estudio anterior de Alaykýran. El% de desviaciones promedio de LB debido a la los tipos de disposición de la máquina se calculan para GA soluciones y comparado con la solución de Algoritmo AS de Alaykýran. los Los resultados computacionales se dan.

4.Conclusión

Como se ve en la Tabla 6, la AG propuesta encontró las soluciones óptimas para todos los problemas de tipo ayb, aunque el algoritmo AS de Alaykıran no pudo encontrar soluciones óptimas. También para tipo d problemas, el GA propuesto encontró un promedio menor % de desviación de LB que Alaykıran AS algoritmos. Pero para problemas de tipo c Alaykıran El algoritmo AS encontró un menor % de desviación promedio de LB que el propuesto GEORGIA.

La GA propuesta no se puede comparar con la AIS y B&B según los tiempos de la CPU porque el configuración de los ordenadores, en los que Los problemas considerados fueron resueltos, son diferentes De uno a otro. Se usa el tiempo de CPU 1600 sólo unos parámetros de parada de GA.

En este artículo, proponemos una GA eficaz para HFS problemas de programación con el objetivo de minimizando la fabricación. El problema considerado es un Problema NP-Hard. La mayoría de los estudios para resolver eso problema son métodos aproximados en lugar de una método exacto, que garantiza una solución óptima. Los problemas de prueba son problemas de evaluación comparativa utilizados en la literatura. Las desviaciones porcentuales de se calculan los límites inferiores. Los hallazgos son en comparación con otro estudio que probó el mismo problemas. Obtuvimos mejores soluciones con el algoritmo GA propuesto. Cuando todos los problemas son considerado; la desviación promedio de la GA El algoritmo es 1,50%, mientras que las desviaciones medias de AIS y B&B son 1,657% y 3,6%, respectivamente. También se puede ver que el Los tiempos de CPU de GA son mucho más pequeños que los de AIS y B&B. La GA propuesta es un buen problema técnica de resolución para un problema de programación y puede utilizarse para otros problemas industriales.

Referencias

[1] Ojeda, D., Divo, E., Kassab, A., Cerrolaza, M. “Locating defects in cortical bone using the boundary element method and genetic algorithms”, (2008) Boletin Tecnico/Technical Bulletin, 46 (1), .

[2] Tovar, J.J., Almanza, O., Montes-Vides, L. “Determination of anisotropy hti by seismic inversion using genetic algorithm in the valle medio of magdalena basin”, (2016) Boletin de Geologia, 38 (4), pp. 107-117.

[3] del Rocio Martínez-Torres, M., Palacios-Florencio, B., Toral-Marín, S.L., Barrero-García, F.J. “Applying genetic algorithms for the identification of Websites' structure”, (2011) Revista Espanola de Documentacion Cientifica, 34 (2), pp. 232-252.

[4] Rodríguez García, M.D.P., Cortez Alejandro, K.A., Méndez Sáenz, A.B., Garza Sánchez, H.H. “Sectorial portfolio analysis with genetic algorithms: Case applied to the Mexican Stock Exchange”, (2015) Contaduria y Administracion, 60 (1), pp. 87-112.

[5] Moncayo, E., Tchegliakova, N., Montes, L. “Seismic investment data pre-stack using a genetic algorithm: Applied to zone channel”, (2011) Revista Brasileira de Geofisica, 29 (3), pp. 511-525.

[6] Uruena, C.G., Garzón-Alvarado, D.A., González, J.M.M. “Genetic algorithms applied to Biomedical Engineering”, (2011) Revista Cubana de Investigaciones Biomedicas, 30 (3), pp. 402-411.

[7] León, M., de Lameda, B.L., Lameda, C., Chacon, J.G., Martínez, M.S., Rojas, J., Contreras-Velásquez, J., Graterol-Rivas, M., Durán, S.W., Aguirre, M., Vera, M., Cerda, M., Garicano, C., Hernández, J.D., Arias, V., Graterol, R., Chacín, M., Bermúdez, V. “Application of fuzzy logic and genetic algorithms for classification of malignant neoplastic diseases treatments”, (2016) Archivos Venezolanos de Farmacologia y Terapeutica, 35 (2), pp. 36-41.

[8] Cabrera-Hernández, L., Morales-Hernández, A., Casas-Cardoso, G.M. “Diversity measures for building multiple classifier systems using genetic algorithms”, (2016) Computacion y Sistemas, 20 (4), pp. 729-747.

[9] Pavón, R. “Summary of Thesis: Bayesian Networks for adjusting parameters of genetic algorithms used in constraint satisfaction problems of geometric”, (2010) Inteligencia Artificial, 14 (45), pp. 5-8.

[10] Pérez B., R.J., Alfonso, E.M., Fernández, S.J. “Parameter estimation and validation of power transformers top oil temperature model by applying genetic algorithms”, (2009) Revista Tecnica de la Facultad de Ingenieria Universidad del Zulia, 32 (3), pp. 266-275.

[11] Salazar, A.F., López, J.F., Tavizón, A., Araiza-Vázquez, M.J. “Study of a genetic algorithm for academic management”, (2019) Formacion Universitaria, 12 (4), pp. 63-72.

[12] Ramos-Herrera, M.D.C., Acosta-González, E. “Factors determining exchange rate stability in member and candidate States of the European Union: An analysis based on genetic algorithms”, (2017) Cuadernos de Economia, 40 (112), pp. 68-82.
[13] Hernández-Díaz, A.M., Cecilia, J.M., García-Román, M.D. “Adjustment of shear models for structural concrete using genetic algorithms”, (2017) Revista Internacional de Metodos Numericos para Calculo y Diseno en Ingenieria, 33 (1-2), pp. 52-57.
[14] Neto, J.A.F., dos Santos Junior, E.C., Paleo, U.F., Barros, D.M., de Oliveira Moreira, M.C. “Optimal subdivision of land in agrarian reform projects: An analysis using genetic algorithms”, (2011) Ciencia e Investigacion Agraria, 38 (2), pp. 168-178.
[15] Méndez Sayago, J.A. “Adapting genetic algorithms to simulate the strategic behavior of contaminating agent companies regarding retribution tax charges”, (2008) Cuadernos de Administracion, 21 (35), pp. 161-187.
[16] Barrera, F.Y. “Search of optimal neural net structures using genetic algorithms and simulated annealing. Verify with the benchmark PROBEN1”, (2007) Inteligencia Artificial, 11 (34), pp. 41-61.