Análisis Comparativo de Heurísticas para el Problema de Calendarización de Trabajos con Transferencia Cero

Beatriz Pérez Rojas
Departamento de Sistemas y Computación
Instituto Tecnológico de Puebla
am120@cs.buap.mx

María Auxilio Osorio Lama
Facultad de Ciencias de la Computación
Benemérita Universidad Autónoma de Puebla
aosorio@cs.buap.mx

Resumen

El presente trabajo aborda la solución del problema de la calendarización de trabajos con transferencia cero o flujo continuo. Para resolverlo se utilizan dos de las heurísticas más efectivas que existen actualmente y un algoritmo genético que utiliza cromosomas no binarios, cruza de subsecuencias, mutación inversa y alternación de generaciones. Finalmente se presenta un análisis comparativo de los resultados obtenidos con estos métodos con respecto a exactitud en la solución y tiempo de CPU.

Palabras Clave: Heurística, Algoritmo Genético, Calendarización

1. Introducción

Los problemas de calendarización de trabajos se presentan en muchos procesos industriales como lo son los sistemas de producción, el diseño de circuitos integrados, de computadoras y de material aislante, la logística y las comunicaciones entre otros. Esto resalta la importancia de estudiar este tipo de problemas y analizar métodos de solución que aporten soluciones aproximadas en tiempos razonables.

Específicamente, se considera el caso en el cual, un conjunto de trabajos $i \in I$ debe ser procesado secuencialmente en una serie de etapas, pero no todos los trabajos requieren pasar por todas las etapas. Esto es, cada trabajo es procesado en un subconjunto de etapas $j \in J(i)$. Además se asume una transferencia cero o un flujo continuo entre las etapas. El objetivo es obtener una asignación de tareas que minimice el tiempo T de terminación. Con el objeto de obtener una solución factible, se necesita eliminar todo antagonismo entre los tiempos de inicio de cada trabajo. Esto requiere que no se procesen dos trabajos simultáneamente en la misma etapa. Para cada par de trabajos i, k, las etapas con antagonismos potenciales son $C_{ik} = \{J(i) \cap J(k)\}$.

Este problema puede describirse con un modelo combinatorio de programación mixta-entera que minimiza el tiempo del proceso sujeto a la condición de evitar antagonismos potenciales entre cada uno de los trabajos en las diferentes etapas del proceso mediante la utilización de variables binarias. El modelo es exacto pero el tiempo de solución crece exponencialmente pues se trata de un problema NP completo.

El modelo mixto-entero (MIP) puede escribirse como:

$$
\begin{aligned}
& \textit{Minimizar } T \\
s.a \quad & T \geq t_i + \Sigma t_{ij} && \forall i \in I \\
& \qquad \text{j}\in\text{J(i)} \\
& t_i + \Sigma t_{im} \leq t_k + \Sigma t_{km} + M(1 - y_{ik}) && \forall j \in C_{ik}, \forall i,k \in I, i < k \\
& \quad m\in J(i) \quad m\leq j \qquad m\in J(k) \quad m\leq j \\
& t_k + \Sigma t_{km} \leq t_i + \Sigma t_{im} + My_{ik} && \forall j \in C_{ik}, \forall i,k \in I, i < k \\
& \quad m\in J(k) \quad m\leq j \qquad m\in J(k) \quad m\leq j \\
& y_{ik} + y_{ki} = 1 && \forall i,k \in I, i < k \\
& t_i \geq 0, i \in I, y_{ik} = \{0,1\} && \forall i,k \in I, i < k
\end{aligned}
$$

donde t_i es el tiempo del comienzo del trabajo i y t_{ij} el tiempo de proceso del trabajo i en la etapa j.

Debido al tiempo exponencial de solución que necesitan los problemas NP completos, desde hace 4 décadas se han venido proponiendo heurísticas que encuentran soluciones aproximadas de forma casi inmediata, Zanakis y Evans [28]. Para problemas de calendarización, Baker[1], Silver[25] y Taillard[26] publicaron compendios de métodos heurísticos, mientras que autores como Campbell[4], Chandrasekharan[6], Dannenbring[8], Giffler y Thompson[16], Nawaz et al.[23] y Wismer[27], popularizaron sus heurísticas especializadas.

Dada la relevancia del problema de calendarización, desde un principio, los algoritmos genéticos, con diversas variantes se han aplicado exitosamente a este problema. Se han publicado compendios por Cheng et al [7], y exitosas aplicaciones particulares por Bean [3], Davis [9], Falkenauer y Bouffoix[13], Gen et al. [15], Nakano y Yamada[22].

Algunos autores como Dorndorf[11], Fang et al.[14], Kibayashy y Yamamura[19], Michalewicz[20],Osorio et al.[24], han ido al área de la computación evolutiva, buscando estructuras eficientes de almacenamiento de cromosomas y variantes en las operaciones de cruza y mutación que permitan obtener mejores resultados en este tipo de problemas.

Sin embargo, para atacar problemas reales, algunas veces deben tomarse en cuenta las limitaciones de los recursos y otros objetivos adicionales a la minimización de la duración total del proceso. En esta área, Esquivel *et al*[12] ha utilizado, con éxito, la multiplicidad para resolver el problema de optimización multicriterio resultante.

Recientemente, Miyashita[21] también combina el objetivo de la minimización de tiempos con el de la optimización de la asignación de los recursos involucrados en los diferentes procesos, y resuelve los problemas de calendarización utilizando agentes múltiples. Cada agente utiliza Programación Genética para aprender y realizar los ordenamientos requeridos.

2. *Heurística de CHANDRASEKHARAN RAJENDRAN (CR)*

Esta heurística fue diseñada por Chandrasekharan[6] utilizando el concepto de retardo mínimo de la primera máquina entre el trabajo inicial i y k, con la condición de transferencia cero entre cada uno. Además de las variables ya descritas, se utiliza σ, el conjunto de trabajos que ha sido calendarizado e $[i]$, el trabajo que se encuentra en la i-ésima posición de una secuencia o arreglo. La duración de la secuencia se calcula con

$$M = \sum_{i=2}^{n} d_{[i-1][i]} + \sum_{j=1}^{m} t_{[n]j} \qquad (1)$$

Un análisis de la ecuación (1) revela que la duración de los trabajos adyacentes que se encuentran dentro de la secuencia y el tiempo de procesamiento del trabajo que se encuentra en la ultima posición de la secuencia, tienen influencia en la duración de cualquier secuencia de trabajos. Con lo anterior es evidente que para minimizar la duración de una secuencia de trabajos se necesita:

i. Que los trabajos adyacentes en una secuencia de trabajos sean lo más parecido posible en tiempo de procesamiento para que se minimice el tiempo de espera entre los trabajos.
ii. El ultimo trabajo debe tener tiempos de procesamiento cortos.

Una característica adicional para minimizar tiempos de procesamiento es tomado del problema de calendarización de dos máquinas de Johnson[18], en donde los trabajos con tendencia creciente (o tendencia positiva) en sus tiempos de procesamiento son procesados (o calendarizados) antes que los trabajos con tendencia decreciente (o tendencia negativa) en sus tiempos de procesamiento para minimizar la duración.

A continuación se describe el trabajo de la heurística mediante un algoritmo:

Paso 1. Formar $A=\{i|\ P_i \geq (1+m)/2, i \in J\}$ y $B=\{i|\ P_i < (1+m)/2, i \in J\}$.

Paso 2. Obtener $A'= \{\ [i]|\ T_{[i]} \leq T_{[i-1]},\ i=1,2,...,n(A)$ e $[i] \in A\}$ y $B'= \{\ [i]|\ T_{[i]} \geq T_{[i-1]},\ i=1,2,...,n(B)$-1 e $[i] \in B\}$. Lo anterior se realiza ordenando los trabajos en A de forma ascendente de acuerdo al valor de T_i y en B en forma descendente de acuerdo al valor de T_i. Se unen los conjuntos A y B para obtener un arreglo de trabajos inicial I.

Paso 3. Remover el primer trabajo de I y tomarlo como trabajo semilla. De esta forma se forma la calendarización parcial σ con n', el numero de trabajos en σ es igual a 1. Actualizar el contenido de I.

Paso 4. Remover el primer trabajo que se encuentra en I e insertarlo en la *p-ésima* posición de la calendarización parcial σ, donde $(n'+1)/2 \leq p \leq (n'+1)$.Evaluar todas las calendarizaciones parciales resultantes en cuanto a su duración usando(4) y seleccionar la calendarizacion parcial con la duración menor. La calendarización parcial seleccionada es asignada a σ con $n'=n'+1$:

Paso 5. Regresar al paso 4 si I no es un conjunto vacío, de otra manera parar y la calendarización parcial es considerada completa. Calcular la duración de esta utilizando (4).

3. *Heurística NEH*

Es una de las heurísticas clásicas es la creada por Nawaz, Enscore y Ham[23], llamada NEH. El algoritmo es el siguiente:

Paso 1.Ordenar los *n* trabajos en forma descendente de acuerdo al tiempo total de procesamiento de cada uno en las maquinas.
Paso 2.Tomar los dos primeros trabajos y calendarizarlos de manera que se minimice la duración de la calendarizacion parcial.
Paso 3.Para *k*=3 hasta *n* hacer
Paso 4.Insertar el *k-ésimo* trabajo en alguna posición de las *k* posiciones disponibles, de manera que el lugar en que se quede fijo minimice la duración de la calendarizacion parcial.

La complejidad del paso (1) es O(n $log(n)$); del paso (2) es O(m). Cada calendarización parcial creada en el paso (4) se necesita O(km) operaciones.

4. *Algoritmo Genético de Cromosomas no Binarios*

Las características del algoritmo genético utilizado se tomaron del Algoritmo Genético propuesto por Kobayashi, Ono y Yamamura[19]. El orden de los trabajos se codifica como una secuencia donde cada trabajo es representado por un número. Cada secuencia se representa como un cromosoma no-binario, lo cual lo hace diferente de los algoritmos genéticos clásicos.

La población es proporcional al tamaño de la secuencia de trabajos a calendarizar con una relación igual a n^2. Las secuencias de la población inicial se generan de forma aleatoria, resolviendo de esta manera, la parte combinatoria. El operador fundamental del algoritmo es la cruza por intercambio de subsecuencias, la cual se realiza si las subsecuencias a intercambiar contienen el mismo conjunto de trabajos. En la Fig. 1 se muestra un ejemplo con cromosomas de longitud 6.

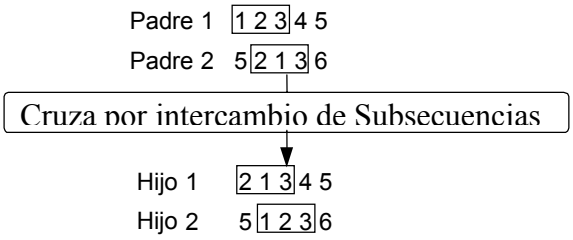

Figura 1. *Ejemplo de cruza por intercambio de subsecuencias (SXX).*

Con el objetivo de mantener la diversidad de soluciones o individuos en la población se utilizan dos estrategias. La primera es la mutación inversa, donde aleatoriamente se toman dos alelos del cromosoma y se invierten sus posiciones. La segunda es la alternación que consiste en seleccionar aleatoriamente un par de individuos de la población en la *t-esima* generación, realizar la cruza por subsecuencia de tareas de estos dos individuos y generar el número de hijos correspondientes (esto es porque no todas las subsecuencias de tareas tienen los mismos elementos). Para reemplazar a los individuos seleccionados para cruza se toman en cuenta a los hijos y

los padres. Los dos individuos con la mejor aptitud son los que sustituirán a los padres en la generación *t+1*. La Fig. 2 muestra gráficamente este criterio.

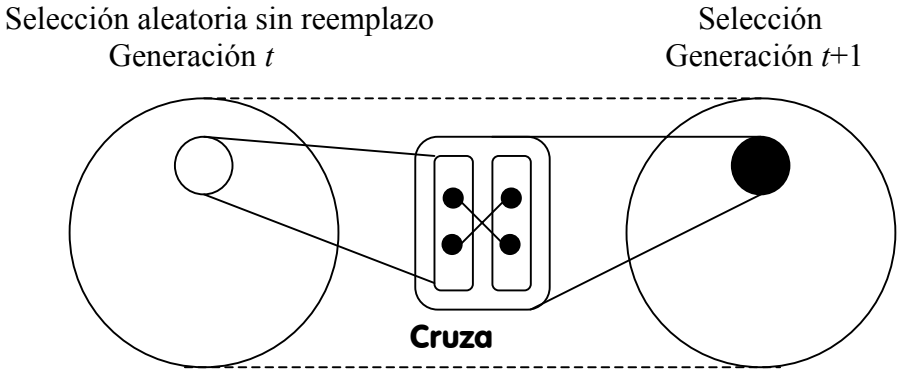

Figura 2. *Estrategia para alternar generaciones*

El método para evaluar a cada individuo desde que se genera la población inicial y durante la evolución se describe en la sección 5 como Función de Evaluación. El algoritmo genético utilizado se presenta en la Fig. 3. Los parámetros para ejecutar el algoritmo son: población de tamaño n^2, probabilidad de cruza del 70% y probabilidad de mutación del 5%.

```
Inicio
   Genera población inicial
   Evalúa la población
   Mientras no se cumpla criterio de terminación
      Selecciona a los individuos para realizar la cruza
       Cruza a los individuos por subsecuencias de tareas y genera con
         los padres y los hijos generados en la cruza una subpoblacion.
          Selecciona a los mejores individuos de la subpoblación para
         incorporarlos a la poblacion en lugar de los cromosomas
         utilizados para la cruza.
      Ejecuta mutación.
   Fin mientras
Fin
```

Figura 3. *Pseudocódigo del Algoritmo Genético Utilizado*

5. Función de Evaluación

Como función de evaluación para el algoritmo genético y las heurística CR y NEH se utilizó el algoritmo propuesto por Nawaz, Enscore y Ham[22]**,** que evalúa cada una de las secuencias generadas y les asigna un valor numérico.

En este algoritmo, la solución para un problema de calendarización de trabajos está dada por una secuencia de trabajos J_{i1}, J_{i2},...,J_{ip},...J_{in}, donde el trabajo J_{ip} es el *p-ésimo* trabajo procesado. Como no se permite tiempo de espera para un trabajo al pasar de una máquina a otra, por lo que cuando termina de procesarse en la máquina M_{jk} debe pasar inmediatamente a M_{jk+1}, esto requiere que M_{jk+1} este libre en ese momento.

Ya que el trabajo *ip* es procesado inmediatamente antes que *ip+1* en la secuencia es conveniente para la notación reemplazarlos por las variables enteras *x* y *y* respectivamente. Ahora definamos la siguiente notación:

d_{xy}^{k} =espera de J_y para entrar en M_{jk} porque M_{jk} esta aun procesando J_x.

T_x^k =tiempo en el que M_{jk} comienza a procesar J_x.

r_x^k = el tiempo requerido para que J_x sea procesado en M_{jk}

El tiempo total de procesamiento para J_x está dado por $R_x = \sum_{k=1}^{k=m} r_x^k$ y el tiempo total de espera para J_y por $d_{xy} = \sum_{k=1}^{k=m} d_{xy}^k$ en donde $d_{xy}^k \geq 0$ y $r_x^k \geq 0$.

Considere un trabajo arbitrario *x* que es procesado primero sin transferencia. Se debe asumir que el tiempo de inicio de este trabajo es 0. Entonces T_x^1=0 y los tiempos de inicio sucesivos están dados por $T_x^{k+1} = T_x^k + r_x^k$, *k*=1,2,...,*m*. T_x^{m+1} es el tiempo en que el trabajo *x* termina de procesarse.

Dependiendo del progreso de J_x, el siguiente trabajo J_y tendrá cierto retardo d_{xy}^k y sus tiempos de inicio están dados por: $T_y^k = T_y^{k-1} + r_y^{k-1} + d_{xy}^k = E_y^k + d_{xy}^k$, $k=1,2,...,m$. E_y^k representa el tiempo de inicio más temprano para J_y en M_{jk}. Ya que J_y no puede entrar a la *k-ésima* máquina hasta que J_x este libre, la desigualdad $T_y^k \geq T_x^{k+1}$. El retardo para que se comience a procesar el trabajo J_y es $d_{xy} = \sum_{k=1}^{k=m} d_{xy}^k$. Este trabajo puede procesarse a través de todas las máquinas sin transferencia.

Ya que J_y no tiene transferencia entre una maquina y otra durante su procesamiento, este puede ser considerado en la posición previa designada para J_x y entonces, se puede calcular un nuevo conjunto de retardos d_{xy} para el siguiente trabajo. Este procedimiento es una manera simple de evaluar d_{xy}^k para todos los pares posibles de trabajos. Dados *n* trabajos, el numero de pares a evaluar es *n!/(n-2)!=n(n-1).*

El procedimiento descrito se puede automatizar utilizando el siguiente algoritmo:

Paso 1. Inicializar.
Paso 2. Seleccionar un par de trabajos no considerados previamente.
Paso 3. Calcular $T_x^{k+1} = T_x^k + r_x^k$ y $T_y^k = T_y^{k-1} + r_y^{k-1} + d_{xy}^k = E_y^k + d_{xy}^k$ *(k=1,2,...,m)*
Paso 4. Se satisface la desigualdad $T_y^k \geq T_x^{k+1}$? Sino, calcular d_{xy}^k>0 con
$d_{xy}^k = max\left[0; T_x^{k+1} - E_y^k\right]$ *(k=1,2,...m)* e incremente T_y^x con el valor de d_{xy}^k. Si esta desigualdad se cumple pasar directamente a 5.
Paso 5. *k=k+1.*
Paso 6. Es ¿*k=n*? Sino ir al paso 3, de lo contrario calcular d_{xy}
Paso 7. ¿Se han considerado todos los pares? Si no ir a 2, de lo contrario parar.

5. *Análisis de Resultados y Conclusiones*

Las heurísticas y el algoritmo genético utilizados en este trabajo se programaron en lenguaje C de Solaris para SUN Ultra10. Para las pruebas se utilizaron instancias con 6, 7, 8, 9, 10 y 20 trabajos y 5, 10, 15, 20 y 25 etapas. Para cada combinación de número de trabajos y etapas se generaron 30 problemas diferentes, obteniendo aleatoriamente, con una distribución uniforme de (1,99), la duración de cada etapa. Se reporta el promedio de estas 30 pruebas para cada combinación.

En la primera fase del experimento se obtuvo el valor exacto, utilizando el modelo mixto entero planteado en la introducción y resolviéndolo con el software para optimización CPLEX 6.5.2. Estos valores exactos pudieron ser obtenidos en un tiempo menor a 1500 segundos de CPU para 6, 7, 8, 9 y 10 trabajos con las 5, 10, 15, 20 y 25 etapas propuestas. Para las instancias con 20 trabajos, se definió el mismo tiempo de 1500 segundos de CPU y se tomó el mejor entero disponible en la búsqueda. También se utilizó como tiempo límite del algoritmo genético en las instancias con 20 trabajos.

		Porcentaje de error promedio			Segundos Tiempo de CPU promedio (segundos)			
Trabajos	Etapas	Algoritmo Genético	Heurística CR	Heurística NEH	Mixto Entero	Algoritmo Genético	Heurística CR	Heurística NEH
6	5	0	2.7201	1.6425	0.3653	0.1353	0.7947	0.007
	10	0	2.5416	1.7277	0.4543	0.4734	0.0636	0.0023
	15	0	2.4376	0.7581	0.6176	0.4006	0.0626	0.0041
	20	0	2.6628	0.8603	0.6463	0.4148	0.8576	0.0041
	25	0	2.5599	1.6822	0.8666	0.6389	0.8576	0.0016
7	5	0	3.2227	1.0712	1.6313	2.8046	1.1173	0.0056
	10	0	2.8448	1.3840	2.3616	2.7078	0.0663	0.0043
	15	0	2.4989	1.3182	2.8886	3.0147	0.0693	0.0046
	20	0	1.5993	1.3429	4.2603	3.0699	1.2273	0.0053
	25	0	2.4711	1.1289	5.0412	3.7419	1.1526	0.0063
8	5	0	2.3494	1.4904	9.3113	14.4819	0.1226	0.0096
	10	0	2.6648	1.8107	18.5983	10.3849	0.0986	0.0056
	15	0	3.3812	1.4951	19.7456	16.9749	0.0826	0.0061
	20	0	2.5527	1.8983	26.5943	10.1983	1.6773	0.0086
	25	0	2.6839	1.6998	33.1723	12.9775	1.9793	0.0046
9	5	0	2.8989	2.3279	64.2763	7.3027	0.2837	0.0096
	10	0	1.8738	1.2432	97.8153	7.4148	0.2086	0.0083
	15	0.2	2.9201	1.8082	154.6556	11.5925	0.1283	0.0066
	20	0	3.0904	1.9397	209.4166	10.0614	1.5483	0.0096
	25	0	3.0069	1.5506	252.9793	10.0913	1.8546	0.0091
10	5	0	3.2485	2.76467	607.8415	22.3848	0.0106	0.0063
	10	0	3.6033	2.1414	778.5428	22.5065	0.0113	0.0099
	15	0	3.3693	2.63697	941.8305	16.6587	0.0113	0.0293
	20	0	2.9854	1.81104	1320.6354	24.6333	0.0123	0.0075
	25	0	3.5489	2.58736	1234.8461	19.7487	1.4333	0.0067

Tabla 1. *Porcentaje de error relativo y tiempo de CPU promedio de los resultados para problemas de 6, 7, 8, 9, y 10 trabajos*

Las heurísticas de Chandrasekharan Rajendran y NEH se probaron registrando el tiempo utilizado y la solución obtenida en cada instancia.

Para cada una de las 30 instancias con 6,7,8,9 y10 trabajos, se realizaron 10 corridas con el algoritmo genético, registrando el tiempo promedio en el cual se llegaba al valor óptimo obtenido con el modelo MIP en CPLEX. En la Tabla 1 se muestra el porcentaje de error promedio de las soluciones obtenidas con cada heurística, y el tiempo promedio que necesitó el algoritmo genético para llegar a la solución óptima.

Para problemas de 20 trabajos, no fue posible obtener la solución óptima en un tiempo menor al límite fijado (1500 segs. CPU) con CPLEX, por lo que se reporta la solución obtenida y el tiempo de CPU que necesitaron las heurísticas especializadas para resolver los problemas con 5, 10, 15, 20 y 25 etapas. También se reporta la mejor solución que se obtuvo en 1500 segundos de CPU por CPLEX y por el algoritmo genético. La Tabla 2 muestra los resultados para estas instancias.

		Solución promedio				Tiempo de CPU promedio (segundos)	
Trabajos	Etapas	Mixto Entero	Algoritmo Genético	Heurística CR	Heurística NEH	Heurística CR	Heurística NEH
20	5	2075.8	1457.5	1540.57	1506.47	0.5	0.52766
	10	2882.07	2027.47	2142.9	2097.73	0.51	0.54066
	15	3525.73	2473.87	2608.77	2563.87	0.99717	0.56266
	20	4136.93	2951.73	3091.47	3048.33	0.93449	0.55033
	25	4769.93	3391.7	3561.63	3499.23	1.43553	0.58733

Tabla 2. *Solución promedio para problemas de 20 trabajos, por todos los métodos y tiempos de CPU promedio de las heurísticas especializadas.*

Los elementos importantes para comparar el método exacto (CPLEX) con las heurísticas especializadas y el algoritmo genético son la calidad de las soluciones obtenidas y el tiempo de CPU que se necesitó para obtenerlas. Para problemas pequeños, el método exacto es el mejor, ya que garantiza la solución óptima, aunque sus tiempos de solución sean mayores hasta en un 5000% que los del algoritmo genético y hasta en un 100,000% que el de las heurísticas especializadas.

Conforme el número de trabajos crece, el tiempo de solución crece exponencialmente y es imposible obtener soluciones exactas, por lo que es necesario utilizar métodos heurísticos. Con 1500 segundos de tiempo límite para el algoritmo genético y CPLEX, el algoritmo genético obtuvo soluciones que mejoraron en un 40%, en promedio, a las obtenidas con CPLEX.

Las heurísticas especializadas proporcionaron resultados en tiempos menores a 1.5 segundos. El algoritmo genético con un límite de 1500 segundos de CPU, mejoró los resultados de la heurística CR en 5.5% y los de la heurística NEH en un 3.5%, en promedio. La heurística NEH superó los tiempos de solución de la heurística CR hasta en un 144%.

El uso de estas heurísticas debe estar de acuerdo a las prioridades de la persona que tiene a su cargo la toma de decisiones. Si lo importante es obtener una calendarizacion

factible rápida, la opcion es utilizar alguna de las heurísticas especializadas propuestas, pero si minimizar el tiempo total del proceso es un factor crucial la mejor opción es el algoritmo genético en los casos en los que no se puedan utilizar los métodos exactos, ya que aporta una mejor solución que las heurísticas especializadas o CPLEX, en los 1500 segundos de CPU utilizados como límite.

Como conclusión puede decirse que las heurísticas especializadas aportan resultados rápidos, sólo inferior en un 5%, en promedio, a los aportados por el algoritmo genético, y que el algoritmo genético proporciona los mejores resultados en todos los casos estudiados.

Referencias

[1] Baker, K., Introduction to Sequencing and Scheduling, John Wiley & Sons, New York 1974.

[2] Barr, R., B. Golden, J. Kelly, M. Resende, W. Stewart, ”Designing and Reporting on Computational Experiments with Heuristic Methods”, Journal of Heuristics, vol. 1.No. 1, p.137-155, 1995.

[3] Bean, J., “Genetic algorithms and random keys for sequencing and optimization, ORSA Journal on Computing, vol. 6. No. 2, p. 154-160, 1994.

[4] Campbell, H.G., R.A. Duder y M.L. Smith, “A heuristic algorithm for the n-job, m-machine sequencing problem”, Mangement Science, Vol. 16, p.630-637, 1970.

[5] Coello, C., “Introducción a los Algoritmos Genéticos”, Soluciones Avanzadas, No.1 1995.

[6] Chandrasekharan, R. “A No-wait Flowshop Sheduling Heuristic to Minimize Makespan”, Journal of Operations Reserach, vol. 45, No.4, p. 472-478, 1994.

[7] Cheng, R., M. Gen, y Y. Tsujimura. A tutorial survey of Job-Shop scheduling problems using genetic algoritms: part I, representation, International Journal of Computers and Industrial Enginnering, vol. 30, no. 4, pp. 983-997, 1996.

[8] Dannenbring, D. G. “ An Evaluation of flowshop sequencing heuristics”, Management Science, Vol. 23, p. 1174-1182, 1970.

[9] Davis, L., editor, *Handbook of genetic Algorithms*, Van Nostrand Reinhold, New York, 1991

[10] Davis, L. Job Shop Scheduling with Genetic Algorithms, Proceedings of the first International Conference on Genetic Algorithms, Grefenstette, J. Editor, Lawrence Erlbaum Associates, p. 136-140, Hillsdale, NJ, 1985.

[11] Dorndorf, U. And E. Pesh, Evolution based learning in a Job Shop Scheduling environment, Computers and Operations Research, vol. 22, p. 25-40, 1995.

[12] Esquivel S., Leiva H.,.Gallard R., "Multiplicity in genetic algorithms to face multicriteria optimization", Proceedings of the 1999 Congress on Evolutionary Computation (IEEE). Washington DC, pp 85-90, 1999.

[13] Falkenauer, E. Y S. Bouffoix, “A genetic algorithm for job shop”, en Proceedings of the IEEE International Conference on Robotics and Automation, p. 824-829, 1991.

[14] Fang, H., P. Ross, y D. Corne, “A promising genetic algorithm approach to job shop scheduling, rescheduling, and open shop scheduling problems”, Proceeding of the Second Annual Conference on Evolutionary Programming, Evolutionary Programming Society, La Jolla, p.375-382, 1993.

[15] Gen, M., Y. Tsujimura, y E., Kubota, “Solving job-shop scheduling problem using genetic algorithms”, Proceedings of the 16th International Conference on Computers and Industrial Engineering, p. 576-570, Japan, 1994.
[16] Giffler, B. y G. Thompson, “Algorithms for solving production scheduling problems”, Operations Research, vol. 8, no. 4, pp. 487-503, 1960.
[17] Goldberg, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, Reading, Massachusetts, 1989.
[18] Johnson, S.M, “Two and three-satage production schedules with set-up times included”, Naval Res. Logist. Vol. 1, p. 61-68, 1954.
[19] Kobayashy, Ono, Yamamura, “ An Efficient Genetic Algorithm for Job Shop Scheduling Problems”, Proceedings of the Sixth International Conference on Genetic Algorithms : University of Pittsburg, p. 506-511, 1995.
[20] Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*, Springer Verlag, New York, 1996.
[21] Miyashita, Kazuo, “Job-Shop Scheduling with Genetic Programming”, en Proceedings of the Genetic and Evolutionary Computation Conference (GECCO’2000), pp. 505-512, Morgan Kaufmann Publishers, 2000.
[22] Nakano, R. y T Yamada, “Conventional genetic algorithms for job-shop problems”, Proceedings of the Fourth International Conference on Genetic algorithms, Morgan Kaufmann Publishers, San Mateo, CA, p. 477-479, 1991.
[23] Nawaz, M., Enscore Jr., E., and Ham, I. “A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem”, OMEGA, The international Journal of Management Science Vol. 11, No. 1 p.91-95, 1983.
[24] Osorio, María, R. Pérez y F. Pérez, “A Comparative Strudy Between Vector and Matrix Representations of Chromosomes in TSP”, Advances in Information Science an Soft Computing, A. Zemliak y N. Mastorakis Editores, WSEAS, 2002.
[25] Silver, E.A., R.V. Vidal, D. De Werra, “A Tutorial on Heuristic Methods”, European Journal of Operational Research, Vol 5, p. 328-365, 1980.
[26] Taillard, E. “Some efficient heuristics methods for the flow shop sequencing problem”, European Journal of Operational Research, vol. 47, p. 65-74, 1990.
[27] Wismer, D. A., “Solution of the Flowshop-Scheduling Problem with No Intermedial Queues”, Systems Control. Operations Research, Vol. 20, p. 689-1972.
[28] Zanakis,S.H., J.R. Evans,”Heuristic Optimization : Why, When, and How to Use It”, Interfaces, Vol 11, No. 5, October 1981.