Un algoritmo de satisfactibilidad para el problema de Job Shop scheduling

Marco Antonio Cruz Chávez[1], Juan Frausto Solís[2], David Juárez[3]

[1] UAEM Av. Universidad 1001 Col. Chamilpa C.P. 62210 Cuernavaca Morelos, México
mcruz@buzon.uaem.mx
[2] ITESM Campus Morelos, Paseo de la reforma No 182-A Col. Lomas de Cuernavaca C.P. 62050 Cuernavaca Morelos, México
jfrausto@campus.mor.itesm.mx

Reporte Técnico
Marzo del 2006

Abstract. En este artículo proponemos un algoritmo llamado SPJS (Satisfactibilidad del problema de Job Shop) para el problema de scheduling, que permite encontrar asignaciones satisfactibles de Job Shop mediante la evaluación de cláusulas reducidas de la Lógica proposicional que representan al problema. La asignación propuesta en Job shop parte de una codificación en forma SAT que solo incluye a cláusulas que representan a las restricciones de precedencia y que se evalúan mediante la obtención de los tiempos de inicio más tardíos, los que se obtienen por medio de un algoritmo que encuentra la ruta más larga..

Palabras Clave: Scheduling, Job Shop, Satisfactibilidad.

Introducción.

Los procesos de manufactura se han vuelto muy complicados debido a que las máquinas que intervienen en estas tareas pueden ejecutar un mayor número de operaciones. También las herramientas que se utilizan en las máquinas son caras y por lo tanto estas tienen que compartirlas a distintos tiempos con diferentes operaciones con el propósito de reducir costos de operación; el problema se complica dado que existen comúnmente dependencias entre las diversas operaciones. De esta forma resulta relevante para los sistemas de manufactura, el problema de scheduling que trata de determinar en que tiempos y en qué máquinas, las operaciones serán realizadas.

Dentro del area de Scheduling, el problema de Job Shop (JSP) es el que mas aplicación práctica ofrece, pues permite incrementar la eficiencia de los procesos de manufactura [Adams et al 1988]. JSP se describe de forma general como sigue:

Se tiene un taller con un conjunto de máquinas $m = \{M_1, M_2, \ldots, M_m\}$ las cuales deben realizar un conjunto de tareas $j = \{1,2,\ldots,n\}$. Cada tarea j requiere de una serie de operaciones $O_{1\,j}, O_{2\,j},\ldots,O_{nj\,,j}$, donde la tupla (i, j) indica la operación i de la tarea j y hay nj operaciones en j. Se requiere obtener la programación de tiempos de cada tarea determinando para ello los tiempos de inicio de cada una de sus operaciones. Al conjunto de estos tiempos de inicio se le denomina un schedule. Se supone que al inicio del proceso (i.e al tiempo cero), cada máquina está disponible y que en cada momento solo puede procesar una operación a la vez. Como información de entrada, se proporciona, la secuencia de máquinas por las que cada tarea debe pasar [Adams et al 1988]; de esta forma, la operación O_{1j} está disponible desde el tiempo cero. Se conocen además los tiempos de procesamiento p_{ij} de todas las operaciones y sus restricciones de precedencia.,

esto es, no se permite que la operación O_{ij} se inicie antes de que termine la operación $O_{i-1,j}$. Ninguna operación puede ser interrumpida antes de que transcurra su tiempo de procesamiento (un caso donde no se permiten preemptions) [Pinedo 1995].
En la figura 1 se ilustra un problema de JSP, el cual se representa por un grafo etiquetado. El problema representado consiste de dos tareas, cada una de ellas se compone de dos operaciones (**O_{11}, O_{21} y O_{12} y O_{22}** respectivamente). Las operaciones **$O_{i\,j}$** se representan en el grafo por las etiquetas O_i el subíndice j que representa el número de la tarea se omite por simplicidad, y este está implícito en el número de renglón del grafo. Así los nodos etiquetados en el grafo por O_1 y O_2 junto a las marcas M1 y M2 corresponden a las operaciones 1 y 2 de la tarea 1. De esta forma estas dos etiquetas O_1 y O_2 se leen **O_{11}** y **O_{21}**. Nótese que los números de las operaciones **O_{ij}** están en formato oscuro para distinguirlas de las etiquetas O_i del grafo. Se tienen restricciones de precedencia Pk para cada tarea (arcos conjuntivos); así , para la tarea uno P1 indica que la operación **O_{11}** debe de terminar antes de iniciar la operación **O_{21}**; lo mismo ocurre para la tarea dos con la precedencia P2. Se cuenta con dos maquinas (M_1 y M_2) que son compartidas por las tareas, esto es, la tarea uno utiliza la maquina uno para realizar la operación **$O_{1\,1}$** y la tarea dos para la operación **$O_{2\,2}$**. De aquí podemos entender que las tareas uno y dos comparten los mismos recursos (máquinas uno y dos respectivamente), por lo que existirán restricciones de capacidad de recursos denotados por C1 y C2 en cada par de operaciones que requiera utilizar la misma máquina (arcos disyuntivos). Sobre cada nodo se encuentra un número que representa las unidades de tiempo de procesamiento necesario para la operación que ese nodo representa, por ejemplo en la figura 1, para la operación **O_{11}** se requieren 3 unidades de tiempo para que dicha operación se procesada. El principal problema es decidir que secuencia de operaciones se debe seguir en cada máquina que compone al Job Shop, respetando las restricciones de precedencia Pk y de compartición (o de capacidad) de recursos Cr. El proceso de solución para este problema se ilustrará en una sección posterior.

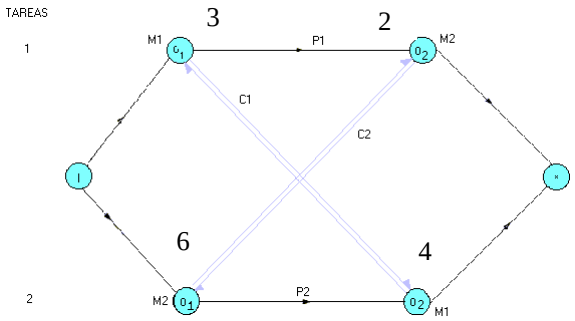


Fig. 1. El problema de Job Shop más sencillo.

Job shop scheduling es un problema de optimización catalogado como NP duro [Papadimitriou 1994], de forma que no hay hasta el momento algoritmos determinísticos que lo resuelvan en forma eficiente (polinomial). Problemas de solo 10 trabajos y 10 máquinas han podido resolverse solo después de un período de 25 años [Schutten 1998]; debido a esta dificultad, se han dado diversas propuestas de cómo plantear un problema de Scheduling, como por ejemplo CSP [Cheng and Smith, 1997; Smith and Cheng, 1993; Sadeh and Fox, 1995], SAT [Crawford and Baker, 1994, Ullman 1975], Programación Lineal [Pinedo 1995], por mencionar solo algunas; para posteriormente darle solución a través de una variedad de algoritmos como por ejemplo, Branch and Bound [Conway et

al, 1967], Simulated Anealing [Perregaard, 1995], Genetics Algorithms [Zalzala and Flemming, 1997], Reglas de prioridad [Panwalker and Iskander, 1997], Enumeración implícita [Carlier and Pinson, 1989; Lageweg et al, 1977], Shifting Bottleneck [Schutten, 1998; Adams et al, 1988], ISAMP [Crawford and Baker, 1994], etc. Actualmente muchos investigadores, han realizado trabajos sobre el análisis y métodos de solución para problemas de satisfactibilidad, en este tipo de problemas que por lo general se han distinguido por ser teóricos, nosotros abordamos un problema practico como lo es JSP.

Formulación del problema de scheduling.

Siguiendo la nomenclatura tradicional [Pinedo 1995], este problema de Scheduling se representa por:

$$Jn|prec|C\max \quad (1)$$

Donde, J indica que se trata de un problema de Job shop con n máquinas, prec señala que se tienen restricciones de procesamiento definidas por los tiempos en que cada operación esté lista (ready time) y sus tiempos de terminación obligatorios (deadline) de cada operación y Cmax es la función objetivo a minimizar llamada Makespan.
La función objetivo que se busca es la de obtener un schedule que minimice el tiempo de terminación máximo de la última tarea en el sistema ó Makespan [Schutten, 1998]. El Makespan se representa por max C_i. De esta forma la función objetivo Z es el minimizar C_i (el máximo tiempo de termino encontrado de una operación i),:

$$Z = min(max\ C_i) \quad (2)$$

Sujeto a (con i y j como par de operaciones):

$$C_i \le s_j \quad (3)$$

$$cap_{ij} = (C_i \le s_j) \vee (C_j \le s_i) \quad (4)$$

$$s_i \ge r_i \quad (5)$$

$$C_i \le d_i \quad (6)$$

donde

$$C_i = s_i + p_i \quad (7)$$

Estas restricciones se explicaran mas adelante. Este problema se puede ver como uno de satisfacción de restricciones (CSP) que consiste en determinar si un conjunto de restricciones puede ser satisfecho. La solución de CSP es una asignación para las variables que intervienen que simultáneamente las satisfaga a todas ellas [Gu, et. al., 1997]. La mayor dificultad de solucionar este problema proviene de las interacciones entre restricciones y, en consecuencia CSP se torna difícil.

El problema de satisfactibilidad

Definición del problema de satisfactibilidad (sat): Sean las variables booleanas x_1, x_2, ..., x_n, las cuales conforman m cláusulas C_1, C_2, ..., C_m, de modo que cada cláusula es una disyunción de una o más variables x_i. Encontrar el valor de verdad de las variables x_2 de modo que la fórmula $C_1 \wedge C_2 \wedge ... \wedge C_m$ sea satisfactible (tome el valor de verdadero).

El problema SAT fue el primero en reconocerse como NPC (NP Completo) [Garey and Jonson 1979], el cual es la versión de decisión de un problema np duro (versión de optimización). Los problemas de optimización pueden llegar a ser NP duros, mientras que los de decisión podrán ser NPC [Papadimitriou 1994]. Los problemas NP duros pueden ser transformados a SAT. Se puede pensar que no se obtiene ninguna ventaja, sin embargo, la transformación a SAT de determinadas instancias de un problema NP duro puede dar por resultado un problema SAT que pueda ser abordado eficientemente. El precio de esta transformación

El problema de scheduling pertenece a la clase NP duros. Un problema A se dice NP duro sí pudiéndose transformar a SAT, mediante una transformación polinomial, no se puede decir que A sea NP completo, sin embargo A es al menos tan difícil (duro) como SAT; a estos problema también se les conoce como intratables [Papadimitriou 1994]. Los problemas de optimización (como. Scheduling) son todos NP duros. Muchos problemas de optimización NP duros son con frecuencia transformados (codificados) a SAT, debido a que para muchas instancias, el problema en su codificación SAT tiene instancias para las cuales existe un algoritmo eficiente. Dicho algoritmo será capaz de determinar una solución factible al problema de scheduling.

Para el problema de JSP [Crawford and Baker, 1994] se utiliza una codificación SAT, que es abordada usando algoritmos clásicos para SAT (TABLEAU, GSAT WSAT). Esta codificación será presentada en la sección siguiente. No obstante, como veremos adelante, en esta codificación se trabaja con todas las restricciones, lo que consume demasiado tiempo de ejecución en los algoritmos para resolver SAT. En este articulo se propone un algoritmo denominado de Codificación Reducida que permite evaluar solo un subconjunto de restricciones codificadas en SAT.

Codificación clásica de scheduling como un problema tipo SAT.

Para esta parte del artículo utilizaremos la tupla *(i,j)* como par de operaciones y nos auxiliáremos de la relación (7). JSP se expresa entonces con las siguientes restricciones [Smith 1993] y [Crawford and Baker, 1994]:

a) Restricciones de secuencias ($i \rightarrow j$):

Indica que la operación *i* debe finalizar antes de comenzar la operación *j*. Esto mismo se expresa con la ecuación (3); esto es, el tiempo de inicio de la operación s_i más su tiempo de procesamiento p_i debe de ser menor o igual que el tiempo de inicio de la operación j.

b) Restricciones de recursos:

La capacidad de recursos representada por cap_{ij} en (4) es una forma de evitar los conflictos que ocurren cuando dos operaciones requieren de una misma máquina al mismo tiempo. Esta se define como una disyunción, lo que significa que la operación *i* debe finalizar antes que inicie la operación *j* o bien *j* debe terminar antes de que inicie *i*, pero no los dos.

c) Restricciones de tiempo de liberación (ready time)

El tiempo de liberación (ready time) de una operación es el tiempo en que está lista para ser introducida al sistema, pero no necesariamente significa que ese sea su tiempo de inicio. Para una operación *i*, este tiempo se representa por r_i en (5). La tarea *i* no puede comenzar su procesamiento s_i antes de su tiempo de liberación. Si esta restricción no

existiera, entonces el procesamiento de la operación *i* podría comenzar en cualquier tiempo.

d) Restricciones de tiempo límite (deadline):

El tiempo límite (deadline) de una operación i es el tiempo (obligatorio) más tardío en que la operación i deberá ser completada (no se debe de sobrepasar) . Se representa por d_i, en (6).

Para la traducción del problema a SAT, deben obtenerse los tiempos de inicio de cada operación ya que para dos operaciones consecutivas i,j, se tiene que decidir si *i* se asigna antes que *j* o bien *j* se asigna antes que *i*. Para cada par de operaciones (*i, j,)*, en cada restricción, se realiza un mapeo al conjunto booleano={verdadero, falso} por lo que se introducen las siguientes variables booleanas [Crawford and Baker, 1994]:

Ec.	Equivale	Traducción SAT	Num	Significado
$i \rightarrow j$	$C_i \leq s_j$	$pr_{i,j}$ = *verdadero*	(**8**)	*i* precede a *j*
$cap_{i\,j,}$	$(C_i \leq s_j) \vee (C_j \leq s_i)$	$pr_{i,j}$ *v* $pr_{j,i}$ =*verdadero*	(**9**)	*i* precede a *j* o *j* precede a *i*
r_i.	$s_i \geq r_i$	$sa_{i,ri}$ = *verdadero*	(**10**)	*i* comienza al tiempo de liberación de *i* (es decir ri) o mas adelante
d_i	$C_i \leq d_i$	$eb_{i,di}$ = *verdadero*	(**11**)	*i* termina al tiempo de término di

Las cláusulas (8) a (11) se denomina conjunto C o conjunto de restricciones. Generalizando estas cláusulas y la formulación del problema, [Crawford and Baker, 1994] determinó el siguiente conjunto S de coherencias mostrado en (12) a (15) y que completa la codificación en SAT. El modelo es del tipo $C \wedge S$ que describe un conjunto de soluciones para JSP, y que mediante valores propuestos de las variables p_{ij} se puede resolver el problema SAT encontrado y determinar un schedule factible. Para evaluar las coherencias, estas requieren estar en forma normal conjuntiva (FNC). En *S*, t significa el tiempo en que la operación puede comenzar o terminar.

Coherencias	FNC	Cláusula
$sa_{i,t} \rightarrow sa_{i,t-1}$	$\neg sa_{i,t} \vee sa_{i,t-1}$	(12)
$eb_{i,t} \rightarrow eb_{i,t+1}$	$\neg eb_{i,t} \vee eb_{i,t+1}$	(13)
$sa_{i,t} \rightarrow \neg cb_{i,t+pi}$	$\neg sa_{i,t} \vee \neg cb_{i,t+pi}$	(14)
$sa_{i,t} \vee pr_{i,j} \rightarrow sa_{j,t+pi}$	$\neg sa_{i,t} \vee \neg pr_{i,j} \vee sa_{j,t+pi}$	(15)

Algoritmo de codificación SAT reducida

Cuando en una codificación SAT de un problema algunas de sus cláusulas son siempre verdaderas y por lo tanto no intervienen en la solución, estas pueden ser eliminadas y al conjunto reducido de cláusulas se le da el nombre de codificación SAT reducida.

Para nuestro método de solución las restricciones de capacidad de recursos, Cr, establecen el conjunto de variables independientes, mientras que las restricciones de precedencia, Pk, son datos del problema que se tienen que cumplir. Como se observa en la figura 1 las restricciones Cr son establecidas mediante arcos disyuntivos que indican una selección de dos opciones, o la operación i de una tarea j, Oij, es ejecutada inmediatamente después de la operación x de la tarea y, Oxy, o viceversa. De esta forma la asignación a uno de los sentidos de una restricción de capacidad la convierte en una restricción de precedencia entre las dos operaciones que une. Por ejemplo, en la figura 1, si a C1 le asignamos un sentido de arriba hacia abajo, la restricción resultante implicará una precedencia entre O11 y O22. De esta forma, haciendo una asignación al conjunto de restricciones Cr se obtiene un grafo conjuntivo a partir del cual se pueden establecer los tiempos de inicio más tardios para cada operación (el tiempo más tarde en el cual la operación puede comenzar para esa asignación). Encontrando entonces el tiempo de inicio más tardío para cada operación, es posible determinar los tiempos de liberación y de término. Usando los valores de los tiempos de liberación y de término encontrados, éstos serán siempre verdaderos, de modo que en cada cláusula disyuntiva donde aparezcan, esta será trivialmente verdadera. Como consecuencia, el conjunto de cláusulas será reducido solo a las cláusulas (15).

El tiempo de inicio más tardío de una operación se determina a través de la ruta critica (ruta más larga desde el inicio a la operación en cuestión). El tiempo de inicio más tardío de una operación se conforma con la suma de los tiempos de procesamiento de las operaciones sobre la ruta crítica sin tomar su propio tiempo. Conociendo los tiempos de inicio mas tardíos de cada operación, se revisa la satisfactibilidad solo del conjunto de cláusulas (15) para cada par de operaciones.

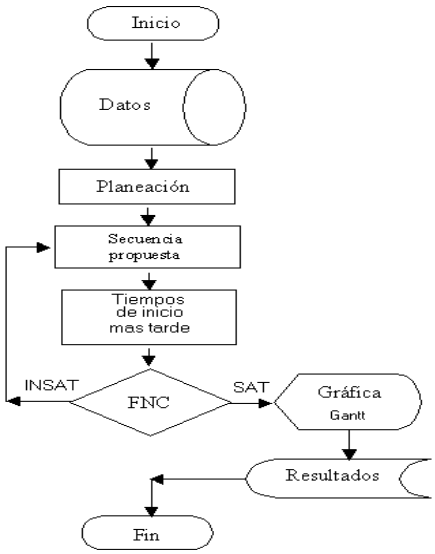


Fig. 2. Diagrama de flujo del algoritmo SPJS el cual evalúa una formula SAT para ver si la secuencia de operaciones propuestas es satisfactible.

La interpretación que se le da a (15) en forma de implicación es: Asegurar que si la operación *i* inicia a o después del tiempo *t* y además *j* sigue de *i*, entonces *j* no podrá

iniciar hasta que *i* haya finalizado. Puesto que tenemos los tiempos de inicio podemos evaluar las variables $Sa_{i,t}$ y $Sa_{j,t+pi}$, donde t representa el tiempo de inicio encontrado para la operación i. Sustituyendo en (15) el tiempo de inicio para *i*, resulta que la primera de estas dos variables es siempre verdadera, mientras que la segunda dependerá de que $t + pi$ no rebase el tiempo de inicio previamente encontrado para *j*. Siendo pr_{ij} supuesta verdadera, el valor de verdad de la implicación (15) dependerá solo de su lado derecho. Entonces, para un par de operaciones (*i*,*j*), cuyos tiempos de inicio y de procesamiento conocemos, el lado derecho de (15) es falso, solo si el tiempo de inicio de *j* (i.e, $t + pi$) es mayor al tiempo de inicio calculado mediante el proceso de ruta critica.

La figura 2 muestra el flujo del algoritmo SPJS en el cual el paso más importante es la evaluación de la formula SAT puesta en FNC formada por el conjunto de cláusulas obtenidas de (15) para cada par de operaciones con restricciones de capacidad de recursos. En la introducción de datos se requiere al número de operaciones; número de máquina, tiempo de procesamiento y restricción de precedencia que le corresponde a cada operación. La planeación designa sobre la base de los datos anteriores a las restricciones de capacidad de recursos para cada operación. La forma en que se propone las secuencias para ser evaluadas, es en forma aleatoria. De acuerdo con la secuencia propuesta se obtienen los tiempos de inicio de cada operación y se prueba la satisfactibilidad de la formula FNC, en caso de ser falsa se vuelve a proponer otra secuencia de operaciones, de lo contrario se obtiene una asignación satisfactible. El resultado obtenido (Makespan, tiempos de inicio de las operaciones y sus secuencias) se almacena en un archivo para su consulta. Para una interpretación fácil de resultados, se incluye un módulo que muestra la asignación de las tareas y el Makespan, por medio de la gráfica de Gantt (ver ejemplos de prueba). Si se tratara de optimizar estos resultados, entonces debera de obtenerse varias asignaciones SAT y elegir a la que tenga el mejor Makespan.

Obtención de los tiempos de inicio más tardíos.

Para la obtención de la ruta mas larga se aplica un algoritmo que genera un árbol de búsqueda de esa ruta para cada una de las operaciones involucradas en el sistema. Por cada ruta encontrada (rama) se almacena el valor de la ruta y se elimina la rama actual para generar la siguiente rama. En caso de que la nueva rama tenga un camino parecido a la anterior, se eliminan los hijos que no estén en la ruta de la nueva rama. Se prosigue en este camino hasta terminar todo el árbol de búsqueda, conservando siempre el valor de la ruta mas larga, esta será el tiempo de inicio más tardío para la operación en cuestión.
En la figura 3. se observa un ejemplo sencillo para la ejemplificación de la obtención de la ruta más larga. Para la ruta de la operación dos a través de las operaciones que le preceden se obtiene el árbol mostrado en la figura 4. En este se muestran todas las posibles rutas que se pueden seguir desde la operación dos a la operación ficticia cero, algunas rutas tienen marcados

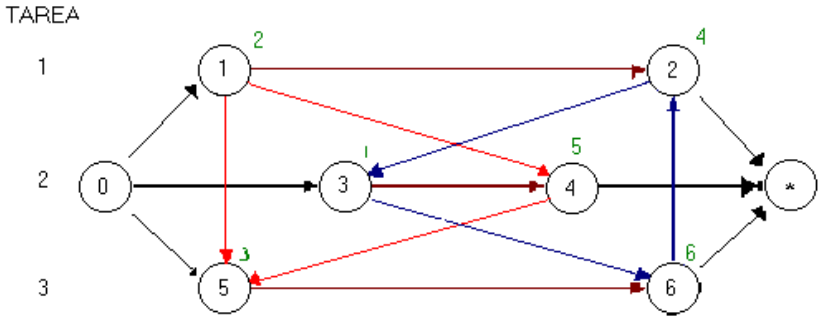
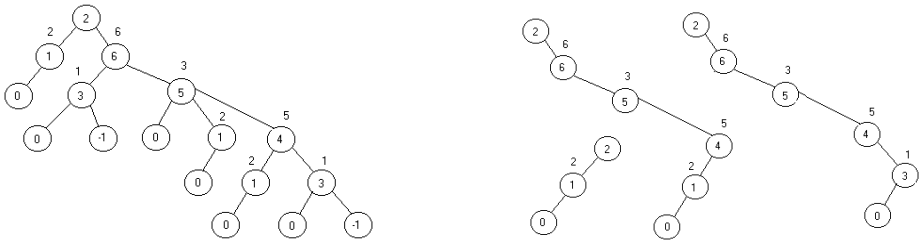


Fig. 3. Un ejemplo sencillo de un problema de Job shop que involucra 2 máquinas y tres tareas con dos operaciones cada una.

sus hijos con una etiqueta de uno negativo (-1) lo que indica que es una ruta que no puede llegar al inicio, por lo tanto se desecha. Las rutas que se toman en cuenta son las que llegan a la operación ficticia cero. La ruta mas larga es 16 como se puede ver en la figura 4(a).



(a) (b)

Fig 4. Path for the operation two of the figure 3. (a) Tree of complete routes. (b) Branches of some routes found in the tree of the parenthesis (a).

A fin de evitar que el árbol crezca de forma exponencial en memoria, se realizan podas cada vez que se genera una rama. La figura 4(b) muestra esto, de izquierda a derecha se comienza generando la primer rama. Esta rama se compone de las operaciones 2-1-0. La conservación de la rama cuando vuelve a utilizarse se ve en la siguiente rama de la figura 4(b). La rama 2-6-5-4-1-0 una vez obtenida su valor de ruta, comienza a eliminarse de abajo hacia arriba. En cada nodo antes de eliminarse se pregunta si se tiene un hijo derecho, en caso de ser cierto se suspende la eliminación y a partir de ese nodo con hijo derecho se continúa una nueva búsqueda de ruta encontrándose la 2-6-5-4-3-0. De lo anterior se puede entender que la rama mas larga encontrada y que pudiera estar en memoria será de complejidad espacial O(n), donde n representa al número de operaciones involucradas en el problema.

Ejemplos de prueba.

Se muestran dos ejemplos sencillos de prueba. El programa con que se implantó el algoritmo por ser no determinístico genera diversos resultados, aquí se presenta solo uno de cada problema y no precisamente el mejor.

Primer ejemplo: Es el mostrado en la figura 3. Se obtiene un Makespan de 12, los datos de planeación y resultados aparecen en las tablas 1 y 2 respectivamente. La gráfica de Gantt

obtenida por el programa se presenta en la figura 6. donde se puede ver que se respetan las precedencias de la solución propuesta en la tabla 2 entre tres trabajos cuyas operaciones se ejecutan en dos maquinas.

Planeación de JSP			
Op	Tproc	Prec	Cap. Rec.
1	2	0	4, 5,
2	4	1	3, 6,
3	1	0	2, 6,
4	5	3	1, 5,
5	3	0	1, 4,
6	6	5	2, 3,

Table 1. Planeación del JSP primer ejemplo.

Capacidad de recursos propuestos a JSP y sus tiempos de inicio de acuerdo a las precedencias				
Op.	Tproc.	Inicio tardío	Prec.	Cap. Rec
1	2	0	0	
2	4	2	1	3,
3	1	0	0	
4	5	5	3	1, 5,
5	3	2	0	1,
6	6	6	5	2, 3,

Table 2. Resultados de una asignacion factible al JSP del primer ejemplo.

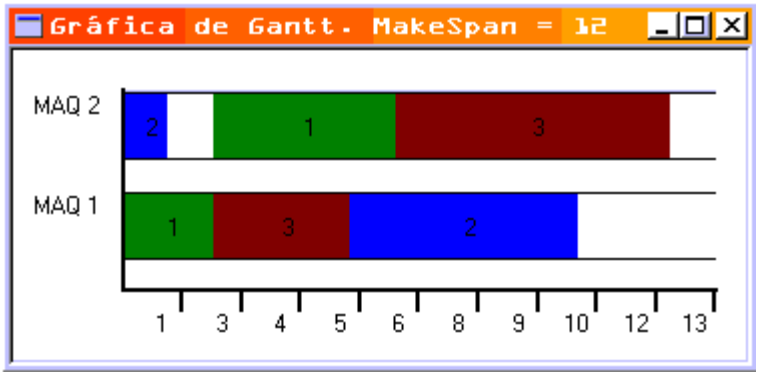


Fig. 4. Diagrama de Gantt de asignación de trabajos y sus precedencias para el primer ejemplo.

Segundo ejemplo: Este es un benchmark propuesto por Muth y Thompson en 1963 y utilizado comúnmente para pruebas. El Job shop es de 6x6, seis máquinas y seis trabajos, cada una de las máquinas efectúan 6 operaciones. Solo se muestra la gráfica de Gantt en la figura 7 y su Makespan obtenido. Como puede verse en la gráfica los trabajos respetan sus precedencias.

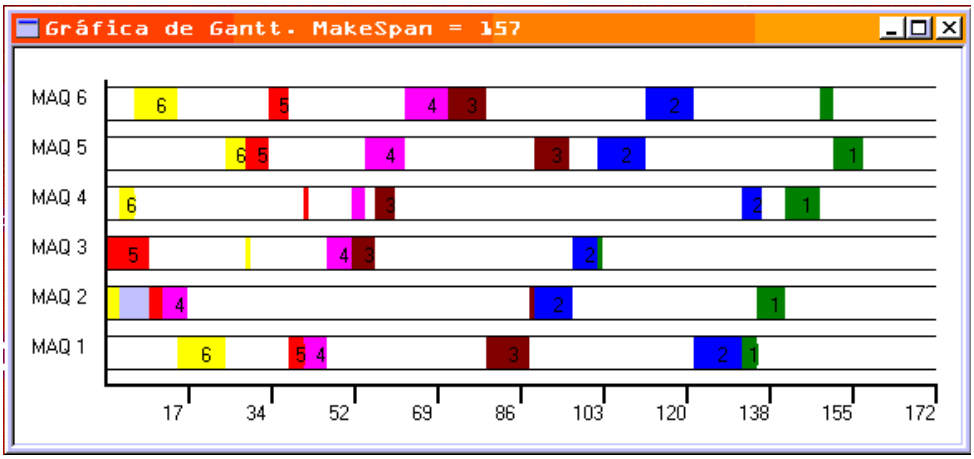


Fig. 5. Diagrama de Gantt que muestra la asignación de trabajos y sus precedencias para el segundo ejemplo.

Conclusiones

Dada la importancia de los problemas prácticos del tipo planeación-scheduling y a su gran dificultad de obtener soluciones por ser del tipo NP-Completo, es de relevancia el dirigir nuestra mirada a SAT pues con una investigación más profunda se puede encontrar un camino para una importante simplificación en el trato de estos problemas. En este enfoque logramos simplificar y reducir el número de cláusulas necesarias para la evaluación de un problema Job shop codificado en SAT, esto es importante en problemas grandes. También se pudo incrementar la eficiencia algorítmica en cuanto a complejidad espacial para la obtención de los tiempos de inicio al realizar una poda constante en el árbol de búsqueda.

Trabajos a futuro.

Todavía se puede eficientar el método que asigna las secuencias de operaciones de tal forma que se puedan proponer secuencias más probables de ser satisfactibles. Por el lado de rutas más largas, se puede buscar o proponer nuevos algoritmos más eficientes para la obtención de la ruta más larga en problemas grandes.

Referencias

[Adams et al. 1988] J. Adams, E. Balas y D. Zawack, “The Shifting Bottleneck Procedure for job shop scheduling”, Management Science Vol. 34, No 3, March 1988.

[Cheng and Smith, 1997] Cheng-Chung Cheng and Stepehen F. Smith, “Applyng constraint satisfaction techniques to job shop scheduling”, Annals of Operations Research 70, 327-357, 1997.

[Conway et al, 1967] R. W. Conway, W.L. Maxwell and L. W. Miller, Theory of scheduling, Addison-Wesley, USA, ISBN 0-201-01189-1,1967.

[Crawford and Baker, 1994] J.M. Crawford and A.B. Baker, “Experimental Results on the Application of Satisfiability Algorithms to Scheduling Problems”, Computational Intelligence Research Laboratory, 1994.

[Garey and Johnson, 1979] M.R. Garey and D.S. Johnson, Computers and intractability: a guide to the theory of NP-completeness, W.H. Freeman and Company, USA, 340 pp., 1979.

[Gu, et. al., 1997] J.Gu, P.W. Purdom, John Franco and B. W. Wah, "Algortthms for the satisfiability (SAT) problem: a survey",

[Jonson 1999] G. Jonson, “Separating the Insolvable and Merely Difficult”, New York Times, New York, UMI Publication No. 05618128, Jul 13, 1999.

[Lageweg et al, 1977] B.J. Lageweg, J.K. Lenstra and A.H.G. Rinnooy Kan, “Job-Shop scheduling by Implicit Enumeration”, Management Science, Vol 24, N0 4, USA, December 1977.

[Papadimitriou, 1994] C.H. Papadimitriou, Computational complexity, Addison Wesley Pub. Co. , USA. ISBN 0-201-53082-1, 523 pp., 1994.

[Perregaard, 1995] M. Perregaard, Branch-and-bound methods for the Processor Job Shop and Flow Shop scheduling problems, Master Thesis, Supervisor: Jens Clausen, DIKU, November 13, 1995.

[Pinedo 1995] M. Pinedo, Scheduling Theory, Algorithms, and Systems, Prentice Hall, U.S.A., 1995.

[Sadeh and Fox, 1995] Norman M. Sadeh and Marks S. Fox, “Variable and value ordering heuristics for the Job Shop scheduling constraint satisfaction problem”, tecnical report CMU-RI-TR-95-39, Appear in the Artificial Intelligence Journal, 1995.

[Schutten 1998] J.M.J. Schutten, “Practical job shop scheduling”, Annals of Operations Research 83(1998)161-177.

[Smith and Cheng, 1993] S. F. Smith, and C. C. Cheng, “Slack-Based heuristics for constraint satisfaction scheduling”. In Proccedings of the Eleventh National Conference on Artificial Intelligence, 134-144, 1993.

[Ullman 1975] J. D. Ullman, “Np-complete scheduling problems,” Journal of Computer System sciences. 10, 384-393, 1975.

[Zalzala and Flemming, 1997] A.M.S. Zalzala and P.J. Flemming. Genetic Algorithms in engineering systems, Zalzala A.M.S. (Ali M. S.), ed., London Institution of Electrical Engineers 1997.

M. Pinedo, Scheduling Theory, Algorithms, and Systems, Prentice Hall, U.S.A., 1995.