Relajación del Problema de Calendarización de Trabajos en un Taller de Manufactura utilizando un Grafo Bipartita

Marco Antonio Cruz-Chávez, Alina Martínez-Oropeza, José Crispín Zavala-Díaz, Martín G. Martínez-Rangel.

Centro de Investigaciones en Ingeniería y Ciencias Aplicadas, Universidad Autónoma del Estado de Morelos Av. Universidad 1001, Col. Chamilpa, 62270, Cuernavaca, Morelos, MÉXICO
{mcruz, alinam, crispin_zaval, mmtzr}@uaem.mx

Abstract. En este artículo se aborda el problema de Calendarización de Máquinas en un Taller de Manufactura. Se establecen las restricciones básicas y el modelado por medio de un grafo disyuntivo. El modelo se mapea al problema de Máquinas en Paralelo no Relacionadas a través de un grafo bipartita. Se hace un análisis de las restricciones de ambos problemas para llevar a cabo una relajación del problema de manufactura. Al llevar a cabo la relajación del problema de manufactura, se tiene que cualquier operación puede ser asignada a cualquier máquina, además de que la restricción de precedencia entre par de operaciones deja de aplicarse. La aplicación del modelo de grafo bipartita aproximado se muestra como un nuevo modelo alternativo para representar al problema de Máquinas en un Taller de Manufactura pudiendo ser éste una opción para trabajar con este tipo de problemas en lugar de utilizar el modelo de grafo disyuntivo.

Introducción

El problema de Calendarización de Trabajos en un Taller de Manufactura mejor conocido por su acrónimo JSSP (por sus siglas en inglés, Job Shop Scheduling Problem), ha sido considerado como uno de los problemas con mayor relevancia en el área de Manufactura y la Optimización, ya que la administración, así como el manejo eficiente de los recursos, es de vital importancia en las empresas. Debido a su complejidad, JSSP ha sido uno de los problemas más estudiados durante las últimas cuatro décadas, motivo por lo que es el problema que ha obtenido más avances en el área de la Calendarización, y sirve como referencia para comparar las técnicas empleadas en diversos problemas clasificados como difíciles, como es el caso del Agente Viajero, entre otros. Además de ser clasificado dentro de la Teoría de la Complejidad como NP-Duro [1], también se conoce como uno de los problemas más difíciles de resolver en ésta clasificación. . La clase NP-Completo puede definirse alternativamente como la intersección entre NP y NP-Duro. Una característica del tipo NP-Duro es que no se conoce un algoritmo determinístico que resuelva en tiempo polinomial a problemas dentro de esa clasificación [23], por lo que muchos

M.A Cruz-Chávez, J.C Zavala Díaz(Eds):CICos2009, ISBN:978-607-00-1970-8, pp. 178 - 188, 2009.

investigadores se han visto atraídos para tratar de resolverlos empleando diversos métodos de optimización como son Branch and Bound [11], Algoritmos Genéticos [12], Recocido Simulado [13], Colonia de Hormigas [3], entre otros. Algunos de estos algoritmos han sido diseñados específicamente para resolver ciertas instancias del problema.

Varios enfoques para resolver el JSSP han sido propuestos usando diversos modelos. Los modelos más utilizados son grafos disyuntivos [14], satisfacción de restricciones [15] y programación lineal entera [14]. Instancias pequeñas, medianas y grandes de JSSP modeladas con el grafo disyuntivo se pueden resolver o encontrar soluciones aproximadas al óptimo global por medio de metaheurísticas, las cuales aplican métodos de búsqueda local. Instancias pequeñas de JSSP modeladas con programación lineal entera se pueden resolver con algoritmos exactos. Los métodos de búsqueda de tipo heurísticos pueden ser muy rápidos para encontrar una solución factible para una instancia del JSSP, pero desafortunadamente no existe prueba de optimalidad de que la solución encontrada sea la óptima. Sin embargo, los métodos de búsqueda pueden proporcionar un punto de inicio. Los métodos basados en satisfactibilidad [9, 15, 17] y reglas de prioridad [20] son algunos ejemplos de métodos de búsqueda. El método de Cuello de Botella [18, 19] es un tipo especial de método de búsqueda que tiene mejor rendimiento que muchos otros, pero sólo en instancias pequeñas. Tradicionalmente, las soluciones de Programación Lineal Entera, mejor conocido por su acrónimo en inglés ILP (Integer Linear Programming) y los paquetes de Diagramas Binarios de Decisión mejor conocido por su acrónimo en inglés BDD (Binary Decisión Diagram) [10], son utilizados para obtener soluciones exactas para muchos problemas de automatización. El problema de calendarización ha sido direccionado por investigadores enfocándose en cualquiera de estos métodos. Las formulaciones basadas en ILP son más populares para calendarización aunque los calendarizadores basados en BDD ofrecen también soluciones atractivas. Una deficiencia de las soluciones basadas en BDD es que el tamaño del diagrama binario de decisión puede llegar a ser muy grande para instancias grandes del problema [21]. Cabe mencionar que la mayoría de estos métodos requieren una solución inicial factible al inicio del proceso [9].

El alcance de la investigación es desarrollar un algoritmo con el enfoque de Colonia de Hormigas (ACO), el cual se encuentre paralelizado por medio de paso de mensajes utilizando lenguaje C y la librería de Paso de Mensajes “MPI”, para, de esta forma implementarlo en una estructura de cluster. El algoritmo propuesto podrá resolver el problema relajado de Calendarización en un Taller de Manufactura. Se podrá comprobar la eficiencia y eficacia por medio de problemas de prueba generados aleatoriamente. Estos benchmarks permitirán probar el rendimiento del algoritmo propuesto, mismo que será comparado con algoritmos de búsqueda local que ya han sido probados, como lo es Recocido Simulado y un algoritmo exacto como Simplex.

Las pruebas a realizar serán de acuerdo a lo que propone el área de estadística para su completa evaluación y análisis. Es decir serán utilizados los mismos problemas (benchmarks) para probar los 3 algoritmos mencionados anteriormente y de esta forma calcular eficiencia, eficacia y error relativo para el algoritmo propuesto y comparar de forma directa con los resultados de los otros dos algoritmos.

En este trabajo se presenta una alternativa para representar el JSSP utilizando un grafo bipartita, el cual puede ser resuelto como un problema de tipo P al relajar el

problema, esto da una posibilidad muy atractiva para probar algoritmos exactos y de tipo metaheurísticos aplicados en la literatura a la solución del grafo bipartita [22] y evaluar la eficiencia y eficacia de estos para el JSSP. El reto de encontrar rápidamente una solución factible es mapear una instancia del JSSP como un problema de Máquinas en Paralelo no Relacionadas, donde se relajan las restricciones del problema para poder representarlo por medio de un grafo bipartita.

El presente artículo se divide en las siguientes secciones: Definición del JSSP, donde se explica la definición general del problema así como el objetivo del mismo y las restricciones básicas; en la segunda sección, se explica el modelo del JSSP por medio de un grafo disyuntivo, dando una solución para una instancia de 3 x 3; la tercera sección aborda la definición del grafo bipartita y sus características, además de llevar a cabo una representación de la instancia utilizada para el JSSP por medio de un grafo tipo bipartita; en la cuarta sección se explica como se llevó a cabo la relajación del problema para poder representarlo por medio de un grafo bipartita y por último se mencionan las conclusiones del artículo así como los trabajos futuros.

1. Definición del JSSP

El problema se define como conjuntos de máquinas y trabajos, donde cada trabajo cuenta con cierto número de operaciones que deben ser procesadas durante determinado tiempo en determinada máquina sin interrupciones. Cada máquina puede procesar únicamente una operación en un instante de tiempo. De manera que se puede definir el problema clásico del JSSP como un conjunto finito J de n trabajos $\{J_j\}_{1 \leq j \leq n}$, los cuales deben ser procesados en un conjunto finito M de m máquinas $\{M_k\}_{1 \leq k \leq m}$. Cada trabajo es visto como una secuencia de máquinas en las cuales este puede ser procesado. El procesamiento de un trabajo J_j en una máquina M_k se conoce como O_{jk}. La operación O_{jk} requiere el uso exclusivo de una máquina M_k, donde el tiempo de procesamiento de cada operación no permite interrupciones y es conocido como t_{jk}. [3]. A continuación (*Tabla 1*) se presenta una solución al problema para una instancia de 3 x 3. Cabe mencionar que el conjunto de soluciones para este problema está dado por $(n!)^m$, es decir para un problema de 3 x 3 tendríamos 216 posibles soluciones, de las cuales, es importante mencionar que no todas son soluciones factibles, debido a que algunas de ellas violan restricciones de precendencia.

Tabla 1. Solución al problema de JSSP para un problema de 3 máquinas y 3 trabajos con 3 operaciones cada uno.

Trabajos	**Operaciones Máquinas (Tiempo de Procesamiento)**		
	1	**2**	**3**
1	1(1)	2(2)	3(1)
2	1(3)	3(1)	2(3)
3	2(2)	1(2)	3(3)

La solución propuesta a este problema (Tabla 1), puede ser representada por medio de una Gráfica de Gantt, donde se representa tanto la calendarización de las operaciones como las unidades de tiempo que se consumen para procesarlas (*makespan*).

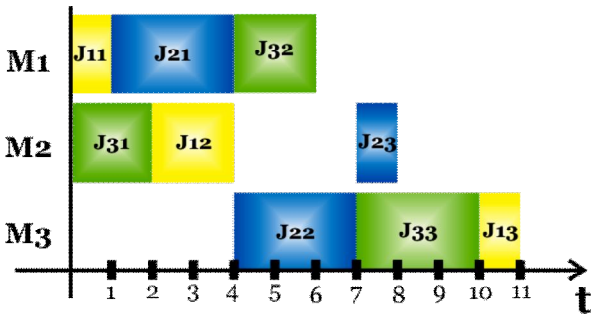

Fig. 1. Una de las 216 posibles soluciones para un problema de 3 máquinas y 3 trabajos.

Una característica importante del JSSP es que la máquina que inicia, no necesariamente empieza con el primer trabajo, sino que cualquier máquina puede iniciar o terminar, ya que el orden de procesamiento de los trabajos en las máquinas es una restricción del problema [6].

1.1. Restricciones

Para llevar a cabo la Calendarización de trabajos en un Taller de Manufactura, es necesario definir ciertas restricciones, mismas que aseguran la integridad de las operaciones y eficiencia del proceso. A continuación se enumeran las restricciones consideradas para este problema.

- Una máquina puede procesar solo un trabajo en un instante de tiempo.
- Un trabajo no debe procesarse en una misma máquina 2 veces.
- No existen restricciones entre operaciones de trabajos diferentes.
- Una vez que una operación es asignada, no puede ser interrumpida.
- Un trabajo consiste en *i* número de operaciones.
- Existe una restricción de Precedencia entre las operaciones de un mismo trabajo, lo cuál, al igual que los tiempos de procesamiento, son datos conocidos.
- Existe una restricción de capacidad de recursos, la cual es conocida como la secuencia de operaciones en una máquina, y son datos conocidos.
- Las operaciones de los trabajos tienen la misma prioridad.
- No se especifican fechas de liberación ni de vencimiento.

La calendarización de trabajos factible puede obtenerse permutando el orden de las operaciones en las máquinas, cuidando de no violar las restricciones anteriormente especificadas [6].

2. Modelo de Grafo Disyuntivo

El problema de JSSP, puede ser descrito por un grafo disyuntivo $G = (V, C \cup D)$, donde *V* es el conjunto de operaciones de los trabajos junto con dos nodos especiales *fuente* y *destino*, los cuales indican el inicio y fin de la calendarización. *C* es el conjunto de arcos conjuntivos que representan la secuencia de las operaciones (restricción de precedencia). *D* consiste en el conjunto de arcos disyuntivos (cliqués) que representan el conjunto de operaciones que deben ser procesadas en la misma máquina [2]. El tiempo de procesamiento para cada operación se encuentra definido, además de que el tiempo de procesamiento del último trabajo (C_{max}) depende directamente de la secuencia de operaciones en cada máquina. A continuación (*Figura 2*) se muestra la representación de una instancia del JSSP de 3 x 3 (*Tabla 1*) por medio de un grafo disyuntivo.

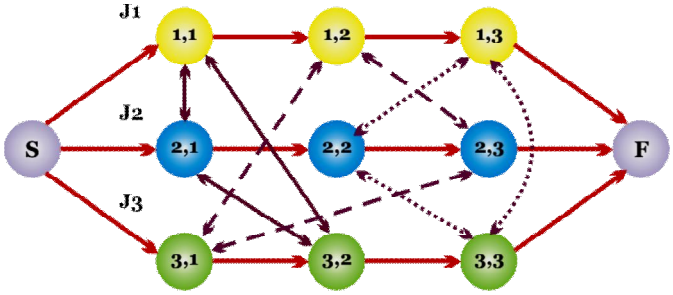

Fig. 2. Representación de un problema de 3 x 3 por medio de un grafo disyuntivo

Un arco disyuntivo puede ser orientado por cualquiera de sus dos extremos. Al momento de llevar a cabo la calendarización, se fija la orientación de los arcos disyuntivos, de esta forma se obtiene la secuencia de operaciones que serán procesadas en cada máquina. Una vez que se obtiene la secuencia de operaciones, los arcos disyuntivos cambian a arcos conjuntivos (*Figura 3*).

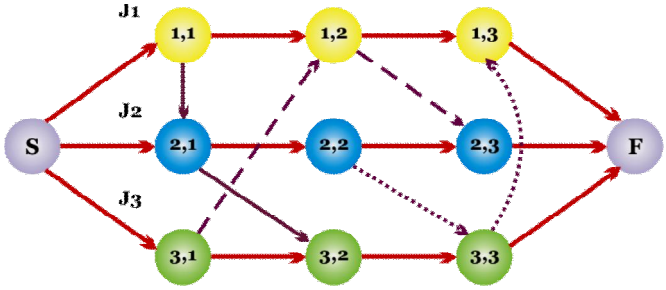

Fig. 3. Una solución para el problema JSSP con tres trabajos en tres máquinas, en donde se encuentra definida por arcos conjuntivos la secuencia de operaciones en cada máquina, respetando las restricciones de precedencia y de capacidad de recursos.

La calendarización de las operaciones, como se mencionó anteriormente, se puede representar por medio de una Grafica de Gantt. Para esta solución, tenemos la calendarización final en la Figura 1.

3. Modelo de Grafo Bipartita

Un grafo bipartita es un grafo no dirigido que tiene la característica de que su conjunto de vértices V pueden ser dividido en dos subconjuntos disjuntos $G = \{V_1 \cup V_2, A\}$, de tal forma que cada arco conecta un vértice de cada subconjunto, es decir, un arco conecta un elemento del subconjunto V_1 y uno del V_2, tomando en cuenta que no deben existir adyacencias entre elementos de un mismo subconjunto [5]. A continuación se enumeran las características básicas de un grafo bipartita.

- Los subconjuntos V_1 y V_2 son disjuntos y no vacíos.
- Cada arista de A une un vértice de V_1 con uno de V_2.
- No existen aristas uniendo dos elementos de V_1; análogamente para V_2.

Tomando en cuenta las características mencionadas anteriormente, se representó el problema de JSSP por medio de un grafo tipo bipartita (*Figura 4*), esto se realizó debido a que ya existen algoritmos de programación lineal que resuelven grafos, bipartitas, como es el caso del Método Simplex.

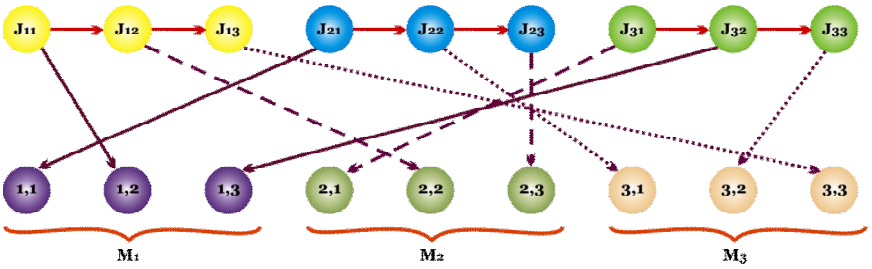

Fig. 4. Representación del problema JSSP por medio de un grafo tipo bipartita

De acuerdo a esta representación (*Figura 4*), se dice que es un grafo tipo bipartita, debido a la restricción de precedencia que prevalece representada entre las operaciones de cada trabajo. Una vez que se tiene esta representación, es necesario relajar las restricciones del problema para poder representarlo como un grafo bipartita autentico y de esta forma tratar de resolverlo por medio de algoritmos de programación lineal.

4 Relajación del JSSP

Para llevar a cabo la relación del JSSP, fueron analizadas cada una de las restricciones, llegando a la conclusión de que algunas de las restricciones básicas del problema, pueden ser tomadas de forma implícita, es decir, que no necesariamente tienen que estar representadas de forma gráfica en el grafo, por lo que, de forma general, la representación del grafo bipartita para esta instancia del JSSP queda de la siguiente manera (*Figura 5*).

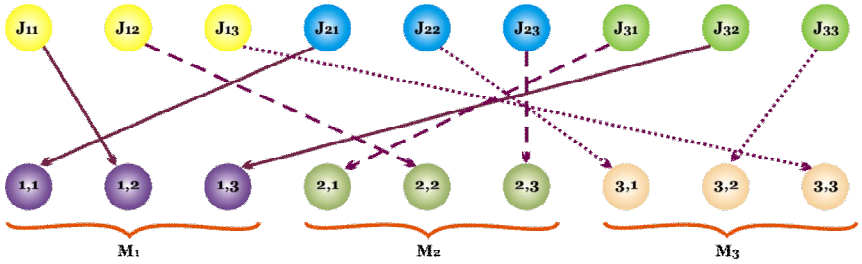

Fig. 5. Representación del JSSP relajado por medio de un Grafo bipartita para una instancia de 3 x 3.

Tomando en cuenta esta representación donde se llevó a cabo la relajación del problema, obtenemos un mapeo al problema de Máquinas en Paralelo no Relacionadas (UPM). De acuerdo a la literatura, el problema de UPM puede ser definido como el conjunto de ***n*** trabajos independientes que necesitan ser calendarizados en ***m*** máquinas en paralelo no relacionadas, de forma que se cumpla la función objetivo que es minimizar el tiempo total de completación de todos los trabajos, para ello se debe considerar que una máquina no puede procesar más de un trabajo a la vez y los trabajos no pueden ser interrumpidos una vez que han sido asignados.

La diferencia del problema de UPM con el JSSP radica en que un trabajo puede ser procesado en cualquier máquina debido a que las máquinas en las que los trabajos serán procesados son multipropósito. El Tiempo de Procesamiento P_{ij} de un trabajo j en una máquina i, se encuentra en función del trabajo a procesar y de la máquina al que fue asignado, esta es una característica fundamental del problema, ya que al ser máquinas no relacionadas, el tiempo de procesamiento de un mismo trabajo varía de una máquina a otra, debido a que sus recursos y capacidades pueden ser diferentes.

De acuerdo a esta definición, para la instancia utilizada para el JSSP, tendríamos un grafo bipartita no dirigido (Figura 6), donde cada trabajo podría ser procesado en cualquier máquina, por lo que de esta forma relajaríamos la restricción de precedencia y tendríamos representada de manera gráfica únicamente la restricción de capacidad de recursos.

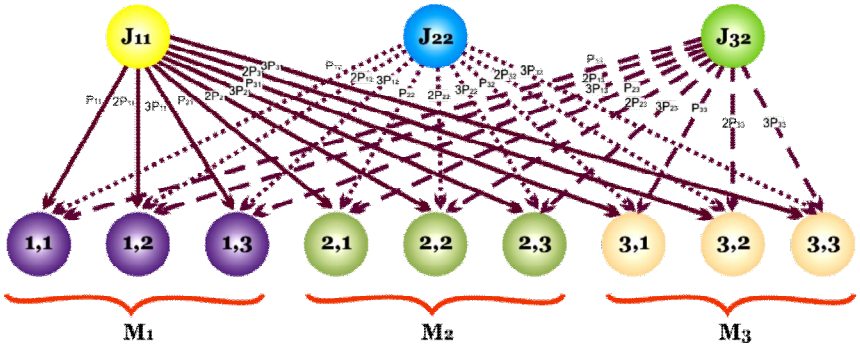

Fig. 6. Grafo Bipartita no dirigido para una instancia de 3 x 3 para el problema de Máquinas en Paralelo no Relacionadas.

En el grafo bipartita mostrado anteriormente, se utiliza kP_{ij} para identificar donde se va a procesar cada trabajo. Donde k es la posición en la máquina i donde se va procesar el trabajo j. Por lo que para el caso de un grafo no dirigido, tenemos que un trabajo puede ser asignado a cualquier máquina en cualquier posición.

Para representar la solución utilizada para el JSSP como grafo bipartita, es necesario pasar del grafo no dirigido mostrado en la Figura 6, a un grafo dirigido, es decir, asignar cada trabajo a una máquina en una posición, tomando en cuenta tanto las restricciones del problema, como las características del grafo bipartita, por lo que la representación de la solución quedaría de la siguiente manera (*Figura 7*).

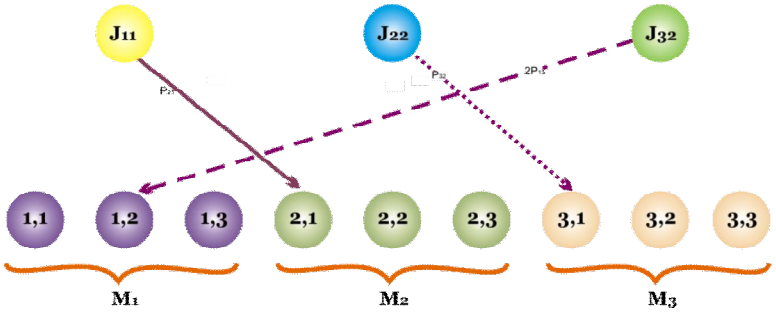

Fig. 7. Grafo Bipartita Dirigido, donde para una instancia de 3 x 3, cada trabajo será asignado a una máquina en *k* posición, de esta forma se evita la sobrecarga o máquinas ociosas.

Tomando esta solución para la instancia propuesta, tendríamos, de acuerdo a los tiempos de procesamiento establecidos en la Tabla 1, una calendarización (*Figura 8*), la cuál como ya se mencionó, se representa por medio de una grafica de Gantt.

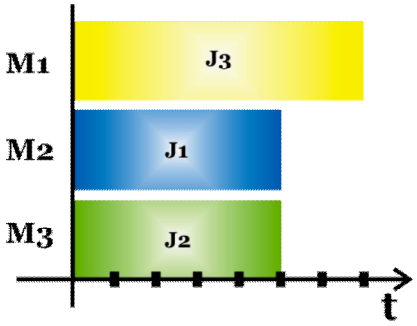

Fig. 8. Calendarización de una instancia de 3 x 3 para el problema de Máquinas en Paralelo no Relacionadas

5. Conclusiones

Uno de los problemas más estudiados en el área de la Optimización es el problema de Calendarización de Trabajos en un Taller de Manufactura (JSSP), el cuál ha atraído la atención un sin número de investigadores alrededor del mundo debido a su dificultad y su clasificación como problema intratable. Por lo que en últimos años se ha puesto especial interés en la utilización de algoritmos híbridos y heurísticos para tratar de resolver este problema.
Se han utilizado diversos modelos para la representación del problema como es el caso del modelo de grafos disyuntivos. En este artículo se realizó una relajación al JSSP, con lo que se consiguió un mapeo al problema de Máquinas en Paralelo no Relacionadas y con ello una modelado por medio de un grafo bipartita.

6. Trabajos Futuros

- Desarrollar un algoritmo capaz de resolver el grafo bipartita sin relajación para el JSSP.
- Comparar los resultados del algoritmo antes mencionado con un algoritmo de grafos disyuntivos y uno de programación lineal.

7. Referencias

1. M. Garey, R. Jonson, D. S. and SEIT, R.: The Complexity of Flor Shop and Job Shop Scheduling, in Mathematics of Operation Research, Vol. 1, No. 2. 1976. pp. 117-129.

2. Yamada Takeshi and Nakano Ryohei. Genetic Algorithms for Job-Shop Scheduling Problems. Proceedings of Modern Heuristic for Decision Support, pp. 67-81, UNICOM Seminar, London 1997.
3. Martínez-Rangel Martín. Una Revisión del Problema de Calendarizar las Tareas en un Taller de Trabajo (Job Shop Scheduling Problem) mediante el Modelo de Grafos Disyuntivo y el Modelo de Programación Entera Mixta. Disponible online: www.uaem.mx/posgrado/mcruz/surveymartin.pdf.
4. Díaz-Pérez Arturo. Análisis y Complejidad de Algoritmos, Complejidad Computacional. Disponible online: *http://delta.cs.cinvestav.mx/~adiaz/anadis/Complexity2k5.pdf*. pp. 11.
5. Rosen Kenneth H. Discrete Mathematics and its Applications. Fifth Edition. Ed. Mc. Graw Hill. International Edition 2003. pp. 549.
6. Cortés Rivera Daniel. Un Sistema Inmune Artificial para resolver el problema de Job Shop Scheduling. Tésis para obtener el grado de Maestro en Ciencias con especialidad en Tecnología eléctrica con opción en Computación, CINVESTAV, Marzo 2004.
7. Morikawa, K., Furuhashi, T. y Uchikawa, Y. Single Populated GA and its Application to Job Shop Scheduling. Proceedings of the 1992 International Conference on Industrial Electronics, Control, Instrumentation and Automation. Vol. 1-3, 1992, ch. 286_vl.003, pp. 1014-1019.
8. Peña Víctor y Zumelzu Lillo. Estado del Arte del Job Shop Scheduling Problem. Disponible on-line: http://www.alumnos.inf.utfsm.cl/~vpena/ramos/ili295/ia-jobshop.pdf. 2006.
9. Frausto-Solis Juan y Cruz-Chávez Marco Antonio, A Reduced Codification for the Logical Representation of Job Shop Scheduling Problems, Lecture Notes in Computer Science, Springer Verlag Pub., Berlin Heidelberg, ISSN: 0302-9743, Vol. 3046 (4), pp. 553 - 562, 2004.
10. Ogrenci-Memik Seda y Fallah Farzan. Accelerated SAT-based scheduling of control/data flow graphs Computer Design: VLSI in Computers and Processors, pp395–400, Proc IEEE, 16-18 Sept. 2002.
11. Carlier, J. And Pinson, E.: An algorithm for solving the job-shop problem, in Management Sciences, Vol. 35, No.2 (1989) 164-176.
12. Zalzala,P.J. and Flemming: Genetic algorithms in engineering systems, in A.M.S. Inst. of Electrical Engineers (1997).
13. Yamada, T. And Nakano, R.: Job-Shop Scheduling by Simulated Annealing Combined with Deterministic Local Search, In Metaheuristics Int. Conference, Colorado, USA, (1995) 344-349.
14. Conway, R.W., Maxwell, W.L and Miller, L.W.: Theory of Scheduling. Addison-Wesley, Reading, Massachusetts (1967).
15. Smith, S.F. and Cheng, C.C.: Slack-Based Heuristics for Constraint Satisfaction Scheduling, in Proc. Of the 11th National Conf. on Artificial Intelligence, Washington, D.C., (1993) 139-145.
16. Cruz-Chávez M. A., Rivera-López R., A Local Search Algorithm for a SAT Representation of Scheduling Problems, Lecture Notes in Computer Science, Springer-Verlag Pub., Berlin Heidelberg, ISSN: 0302-9743, Vol.4707, No. 3, pp. 697-709, 2007.

17. Crawford, J.M. and Baker, A.B.: Experimental Results on the Application of Satisfiability Algorithms to Scheduling Problems, in Proc. Of the 12th National Conf. on Artificial Intelligence, Austin, TX, (1994) 1092-1098.
18. Adams, E., Balas, E. And Zawack, D.: The shifting Bottleneck Procedure for Job Shop Scheduling, in Management Science, Vol. 34, No. 3 (1988) 391-401
19. Schutten, M.J.: Practical job shop scheduling, in Annals of Operations Research, Vol. 83, (1988) 161-177.
20. Yoshida T. and Touzaki, H., A Study on Association amount Dispatching Rules in Manufacturing Scheduling Problems, IEEE, 0-7803-5670, 1999.
21. Rune-Jensen Anders, Bech Lauritzen Lau and Laursen Ole. Optimal Task Graph Scheduling with Binary Decision Diagrams. Disponible on line: http://citeseerx.ist.psu.edu/viewdoc/download?doi= 10.1.1.84.1116&rep=rep1&type=pdf .2004.
22. Cruz-Chávez M. A., Juárez-Pérez F., Ávila-Melgar E. Y., Martínez-Oropeza A., Simulated Annealing Algorithm for the Weighted Unrelated Parallel Machines Problem, Electronics, Robotics and Automotive Mechanics Conference, CERMA2009, IEEE-Computer Society, ISBN , pp, September 22 - 25, México, 2009 (a publicar).
23. Pinedo Michael. Scheduling. Theory, Algorithms, and Systems, Third Edition. Springer Science-Business Media, LLC. ISBN: 978-0-387-78934-7, e-ISBN: 978-0-387-78935-4.