APLICACIÓN DE LA PROGRAMACIÓN DINÁMICA PARA RESOLVER EL PROBLEMA SIMPLE DE BALANCEO DE LINEA DE ENSAMBLE.

Dynamic programming application to solve the simple assembly line balancing problem

RESUMEN
Este documento presenta en forma reducida la aplicación de un algoritmo exacto para solucionar un problema simple de balanceo de línea de ensamble, el algoritmo utilizado es la programación dinámica, se muestran las ventajas y desventajas de usar este método para solucionar problemas de tipo combinatorial.

PALABRAS CLAVES: Estación, Tarea, Tiempo de ciclo, Algoritmo, Programación dinámica.

ABSTRACT
This paper show reduced form the application of an exact algorithm to solve a simple assembly line balancing problem, the algorithm used is dynamic programming, presents the advantages and disadvantages of use this method to solve combinatorial problems..

KEYWORDS: *Workstation, Task, Cycle time, Algorithm, Dynamic programming.*

ANDRES JARAMILLO GARZON
Ingeniero Industrial.
Desarrollo Físico Comfamiliar Risaralda
Estudiante de la Maestría en Sistemas Integrados de Gestión de la Calidad
Facultad de Ingeniería Industrial
andresjg@utp.edu.co

JORGE HERNAN RESTREPO CORREA
Ingeniero Industrial, M.Sc
Profesor Asistente
Universidad Tecnológica de Pereira
jhrestrepoco@utp.edu.co

1. INTRODUCCIÓN

El problema de las líneas de ensamble ha sido estudiado por diferentes investigadores desde diferentes enfoques, tratando de dar solución a este problema utilizando algoritmos exactos y algoritmos heurísticos. La noción de línea de ensamble existe desde hace varios siglos, cuando se tenían líneas incipientes y rudimentarias para construir embarcaciones en astilleros en Venecia; pero solo fue hasta Henry Ford que se estudiaron las líneas de ensamble de manera técnica y científica, desde ese momento empieza la investigación y desarrollo sobre el equilibrado de las líneas de ensamble. La optimización de las líneas de ensamble ha sido el objetivo de las investigaciones en las cuales se han empleado diferentes técnicas: heurísticas, meta-heurísticas, algoritmos exactos, algoritmos genéticos. Entre otros.

La programación dinámica fue aplicada a este problema por primera vez en el año de 1963 por Held, Karp y Shareshian en el documento **[5],** en donde se realiza un desarrollo matemático complejo para la aplicación de la programación dinámica a la solución del problema de balanceo de líneas, considerando restricciones de precedencia, esta investigación se considera como base para las investigaciones posteriores. En el año de 1978 Schrage y Baker **[11]** desarrollan un documento en el cual toman conceptos del trabajo anteriormente descrito e introducen detalles sobre procedimientos de enumeración y se hacen comparaciones sobre tiempos de cómputo con otros métodos. Posteriormente Kao y Queyranne en 1982 **[8]** Hacen recopilación de investigaciones anteriores y describen el procedimiento en seudo código. Todos los documentos anteriormente nombrados son artículos de revistas de investigación que muestran de forma muy general la utilización del método propuesto para la solución del problema. Por esta razón surge la necesidad de ampliar la investigación y aclarar las ventajas y desventajas de la aplicación del método para resolver el problema propuesto.

El problema de balanceo de líneas de ensamble, es uno de los mas comunes en las fabricas y empresas industriales, en términos generales trata de optimizar los recursos de la línea de ensamble, ya sea minimizando estaciones de trabajo, o minimizando el tiempo de ciclo, es decir, el problema de balanceo de línea de ensamble trata de asignar las tareas en una secuencia ordenada de las estaciones, satisfaciendo las relaciones de precedencia y optimizando una función objetivo.

En este trabajo se resolverá concretamente un pequeño problema tipo SALBP-1 (será descrito posteriormente), utilizando la programación dinámica.

Fecha Recepción: 9 de Septiembre de 2010
Fecha aceptación: 15 de Noviembre de 2010

2. CLASES DE PROBLEMAS DE BALANCEO DE LINEA DE ENSAMBLE

Consiste en distribuir las tareas necesarias para ensamblar un producto a través del conjunto de estaciones que conforman la línea de ensamble, esta distribución de las tareas en las estaciones de trabajo se hace siguiendo un objetivo, puede ser maximizar la eficiencia de la línea, o minimizar el tiempo ocioso o minimizar el número de estaciones requeridas en la línea de ensamble. Un problema de balanceo de línea está compuesto por una función objetivo y un conjunto de restricciones.

Una solución factible de un problema de balanceo de línea de ensamble debe cumplir con las siguientes condiciones:

- Cada tarea se debe asignar exactamente a una estación
- Se debe cumplir por completo con las relaciones de precedencia.
- La suma de los tiempos de las tareas de cada estación no deben exceder el tiempo de ciclo, para todas las estaciones.

El problema de balanceo de línea de ensamble (ALBP) se divide en dos categorías, los SALBPs, que son problemas simples de balanceo de línea, en los que se consideran pocas variables de entrada desconocidas para reducir la complejidad del mismo; y los GALBPs, problemas generales de balanceo de línea de ensamble, que estudian casos más reales y complejos que se presentan en la industria.

2.1 Problema general (GALBP)

Según el documento de Capacho y Pastor **[3],** los problemas generales de balanceo de líneas de ensamble, consideran los problemas que no son SALBP, es decir, problemas más complejos, cuyas características se asemejan a un problema real de balanceo de línea.

Se distinguen cuatro casos de GALBP:

UALBP: U-line assembly line balancing problem – problema de equilibrado de líneas tipo U. Los UABLP están caracterizados de manera similar a los problemas SALBP pero consideran una línea tipo U en lugar de una serial. Las líneas tipo U se consideran líneas más flexibles que las líneas tipo serial, según Scholl y Becker **[9],** en los SALBP únicamente se pueden asignar aquellas tareas cuyos predecesores han sido asignados. Las estaciones pueden ser colocadas de tal manera que, durante el mismo tiempo de ciclo, se puedan manejar a la vez dos piezas en diferentes posiciones de la línea. Esto implica que hay un mayor número de posibilidades de asignar las tareas a las estaciones, lo que resulta, en algunos casos que el problema se pueda resolver de manera más eficiente que cuando se tiene un línea simple. De manera similar a los problemas simples SALBP, se distinguen los problemas UABLP-1, UABLP-2 y UABLP-E, en donde se busca minimizar el número de estaciones, minimizar el tiempo de ciclo y maximizar la eficiencia de la línea U, respectivamente.

MALBP: *mixed-model assembly line balancing problem* Problema de equilibrado de líneas de modelos mixtos. Este tipo de problemas se presentan cuando se consideran varios modelos de un mismo producto y, por lo tanto, se tiene un conjunto de tareas básicas que se realizan en todos los modelos sin considerar tiempos de Setup. En este caso, también se tiene el problema de secuenciación de los diferentes modelos así como el problema de determinar el tamaño de los lotes de cada modelo; la secuenciación puede ser importante dado que los tiempos de tareas entre modelos pueden variar significativamente. También se tienen las versiones MALPB-1, MALBP-2 y MALBP-E**. [3]**

RALBP: *robotic assembly line balancing problem* – problema de equilibrado de líneas robotizadas. En este tipo de problemas se considera tanto la asignación de las tareas como la asignación de un grupo de robots a las estaciones de trabajo, con la finalidad de optimizar la realización de las tareas en la línea.

MOALBP: *multi-objective assembly line balancing problem* – problema de equilibrado de líneas con objetivos múltiples. En este tipo de problemas se consideran varios objetivos simultáneamente como por ejemplo: minimizar el número de estaciones, el coste total de montaje o el número de buffers; maximizar la eficiencia de línea, etc. De acuerdo con Capacho y Pastor **[3]** la mayoría de los problemas de equilibrado de líneas consideran múltiples objetivos.

Los problemas anteriores, el SALBP y el GALBP se pueden subdividir, teniendo en cuenta:
El tipo de producto que se procesa en la línea: modelo simple (SM) y modelo mixto (múltiple) (MM).
La variabilidad del tiempo de duración de las tareas: determinístico (D) y estocástico (S).

2.2 PROBLEMA SIMPLE (SALBP)

Los SALBP contienen los problemas de balanceo más simples y se caracterizan por: consideran líneas simples, sólo se consideran restricciones de precedencia, se asume que las tareas son indivisibles, los tiempos de proceso de las tareas son considerados independientes de la estación y del orden de proceso, los tiempos de proceso de las tareas son determinísticos y conocidos a priori, así como

todos los parámetros de entrada, la línea es sincrónica, se tiene un tiempo de ciclo (o un número de estaciones) fijo, la arquitectura de la línea es serial con todas las estaciones igualmente equipadas para realizar cualquiera de las tareas y la tasa de entrada de las piezas a la línea es fija.

Se distinguen cuatro casos de SALBP:

1. **SALBP-1**: consiste en asignar un conjunto de tareas a las estaciones de tal forma que se minimice el número de estaciones, dado un tiempo de ciclo (o tasa de producción). Este caso se presenta habitualmente cuando un nuevo sistema de montaje va a ser instalado y la demanda externa puede ser estimada.

2. **SALBP-2**: Este problema busca lo contrario del problema anterior, es decir, se busca minimizar el tiempo de ciclo (o maximizar la tasa de producción), dado un Número de estaciones fijo. Se considera que la línea de montaje ya existe.

3. **SALBP-E**: maximiza la eficiencia *E* de la línea, es decir, minimiza el producto de *m* (número de estaciones) por *c* (tiempo de ciclo).

4. **SALBP-F**: consiste en determinar si existe alguna solución factible para la combinación de un número *m* de estaciones y un tiempo de ciclo *c*; es decir, se quiere conocer si la línea puede operar con *m* estaciones y un tiempo de ciclo *c* dados. No se busca minimizar ni maximizar ningún valor.

El siguiente cuadro muestra en resumen las características de los problemas SALBP`s.

Versión	Tiempo de Ciclo	Número de Estaciones
SALBP-F	Fijo	Fijo
SALBP-1	Fijo	Mínimo
SALBP-2	Mínimo	Fijo
SALBP-E	Mínimo	Mínimo

Tabla 1. Versiones del SALBP

3. PROGRAMACIÓN DINÁMICA

La programación dinámica es una técnica matemática útil en la toma de una serie de decisiones relacionadas entre sí. Proporciona un método sistemático para determinar la combinación óptima de decisiones. **[2,6,9,12]**

En contraste con la programación lineal, no se cuenta con una formulación matemática estándar para el problema de programación dinámica, sino que se trata de un enfoque de tipo general para la solución de problemas, y las ecuaciones especificas que se usan se deben desarrollar para que representen cada situación individual. Entonces, se necesita cierto grado de ingenio y un buen conocimiento de la estructura general de los problemas de programación dinámica para reconocer cuando y como se puede resolver un problema por medio de estos procedimientos. Estas habilidades se pueden desarrollar mejor mediante la exposición de una gran variedad de aplicaciones de programación dinámica y con el análisis detallado de las características comunes a todas estas situaciones.

4. METODOLOGIA DE SOLUCION DE UN PEQUEÑO PROBLEMA SIMPLE DE BALANCEO DE LINEA DE ENSAMBLE

A continuación se muestra la metodología de solución utilizada para resolver el problema simple de balanceo de línea de ensamble, en dicho problema se conoce el tiempo de ciclo y se quiere encontrar el mínimo de estaciones para realizar las tareas de la línea.

4.1 Problema

Se tiene una línea de ensamble, la cual, sus características permiten abordar el problema de equilibrado como un problema SALBP. Se conoce que la tasa de producción de la línea es de 61200 unidades por hora, además, se tienen los tiempos estándares de las tareas en la siguiente tabla:

No	Tarea	T estándar (s)	Precedencia
1	Put inspection card	6	
2	Place panel and the low case	11	1
3	Put supply desk PCB on conveyor	10	
4	Wire dressing for 6 spots on PCB	6	2,3
5	Setting PCB on the desk	7	4
8	CW direction for piece with washer	8	5
6	CW direction for piece assembly	10	5
7	Remove the jigs	3	5
9	Put PCB desk in the lower case	9	6,7,8
10	connect wire	12	9

Tabla 2. Tiempos de las tareas

El siguiente diagrama explica mejor el problema:

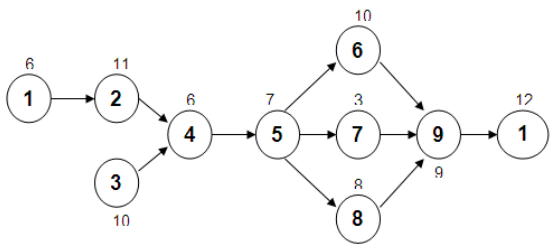

Figura 1. Grafo de precedencias.

4.2 Solución

Identificamos el problema como un SALBP, tipo 1, ya que el tiempo de ciclo es dado, (17 segundos), se va a emplear la programación dinámica para encontrar el mínimo número de estaciones de la línea **[4,5]:**

- Se inicia con el conjunto vacío: v = {} ; que tendrá un tiempo t{v} = 0
- Desde j = 1 hasta j = N; Se generan todos los subconjuntos de dimensión "j" teniendo en cuenta las relaciones de precedencia, (no importa el orden). Cada subconjunto generado es igual a uno o varios subconjuntos de dimensiones "j-1" ya calculado en la iteración anterior mas una tarea nueva. Entonces se calcula el tiempo de cada subconjunto v como sigue:

$$t\{v\} = t\{J_1,...,J_j\} = \min_{J_k \in v}(t\{v - J_k\} + \Delta J_k)$$

- Al final se obtiene t{v} para v = {J1, J2,…,JN}, es decir, el de todas las tareas. Para calcular la secuencia óptima se empieza por el final y se va viniendo la tarea que se fue añadiendo y con base a la subsecuencia de dimensión inferior que se añadió. Una vez se tenga la secuencia óptima se forman las estaciones a partir de los tiempos de las tareas en bloques de tamaño máximo C.

Teniendo en cuenta la metodología anterior se va a resolver el problema propuesto:

Inicialización:
V= {} T {}=0
Subconjuntos con 1 tarea
V= {1} T {1}=6
V= {3} T {3}=10

Subconjuntos con 2 tareas
V= {1,2} T {1,2}= T {1}+Δ2= 6+11=17
V= {1,3} T {1,3}= min {T {1}+Δ3; T {3}+Δ1}=
min {6+10; 10+6}=16

Subconjuntos con 3 tareas
V={1,2,3}
T{1,2,3}= min{ T{1,2}+Δ3; T{3,1}+Δ2}=
min {17+ {17-17+10}; 16+ {17-16+11} }=27

Subconjuntos con 4 tareas
V={1,2,3,4} T{1,2,3,4}= T{1,2,3}+Δ4= 27+6=33

Subconjuntos con 5 tareas
V={1,2,3,4,5} T{1,2,3,4,5}= 33+{34-33+7}=41

Subconjuntos con 6 tareas
V={1,2,3,4,5,6} T{1,2,3,4,5,6}= T{1,2,3,4,5}+Δ6 = 51
V={1,2,3,4,5,7} T{1,2,3,4,5,7}= T{1,2,3,4,5}+Δ7 = 44
V={1,2,3,4,5,8} T{1,2,3,4,5,8}= T{1,2,3,4,5}+Δ8 = 49

Subconjuntos con 7 tareas
V={1,2,3,4,5,6,7}
min{ T{1,2,3,4,5,6}+Δ7; T{1,2,3,4,5,7}+Δ6}=
min {51+ {51-51+3}; 44+ {51-44+10}}= 54

V={1,2,3,4,5,6,8}
min{ T{1,2,3,4,5,6}+Δ8; T{1,2,3,4,5,8}+Δ6}=
min {51+ {51-51+8}; 49+ {51-49+10}}= 59

V={1,2,3,4,5,7,8}
min{ T{1,2,3,4,5,7}+Δ8; T{1,2,3,4,5,8}+Δ7}=
min {44+ {51-44+8}; 49+ {51-49+3}}= 54

Subconjuntos con 8 tareas
V={1,2,3,4,5,6,7,8}
min{ T{1,2,3,4,5,6,7}+Δ8; T{1,2,3,4,5,6,8}+Δ7;
T {1,2,3,4,5,7,8}+Δ6;}=
min{ 54+8; 59+3; 54+10}= 62

Subconjuntos con 9 tareas
V={1,2,3,4,5,6,7,8,9}
T{1,2,3,4,5,6,7,8,9}= T{1,2,3,4,5,6,7,8 }+Δ9
= 62+ {68-62+9}=77

Subconjuntos con 10 tareas
V={1,2,3,4,5,6,7,8,9,10}
T{1,2,3,4,5,6,7,8,9,10}= T{1,2,3,4,5,6,7,8,9 }+Δ10
= 77+ {85- 77+12}=97

Para encontrar la secuencia óptima del algoritmo, se empieza a construir por el final, seleccionando aquella tarea que dio el valor mínimo:

Inicio de la Secuencia: (, , , , , , , , ,)

Subconjunto de 10 tareas con V={1,2,3,4,5,6,7,8,9,10}:
Min (97) se alcanza al añadir Δ10 -> (, , , , , , , , , 10) ; quedan {1,2,3,4,5,6,7,8,9}

Subconjunto de 9 tareas con V={1,2,3,4,5,6,7,8,9}:
Min (77) se alcanza al añadir Δ9 -> (, , , , , , , , 9, 10); quedan {1,2,3,4,5,6,7,8}

Subconjunto de 8 tareas con V={1,2,3,4,5,6,7,8}:
Min (62) se alcanza al añadir Δ7 -> (, , , , , , , 7, 9, 10) ; quedan {1,2,3,4,5,6,8}

Subconjunto de 7 tareas con V={1,2,3,4,5,6,8}:
Min (59) se alcanza al añadir Δ8 -> (, , , , , , 8, 7, 9, 10) ; quedan {1,2,3,4,5,6}

Subconjunto de 6 tareas con V={1,2,3,4,5,6}:

Min (51) se alcanza al añadir Δ6 -> (, , , , , 6, 8, 7, 9, 10) ; quedan {1,2,3,4,5}

Subconjunto de 5 tareas con V={1,2,3,4,5}:
Min (41) se alcanza al añadir Δ5 -> (, , , , 5, 6, 8, 7, 9, 10) ; quedan {1,2,3,4}

Subconjunto de 4 tareas con V={1,2,3,4}:
Min (33) se alcanza al añadir Δ4 -> (, , , 4, 5, 6, 8, 7, 9, 10) ; quedan {1,2,3}

Subconjunto de 3 tareas con V={1,2,3}:
Min (27) se alcanza al añadir Δ3 -> (, , 3, 4, 5, 6, 8, 7, 9, 10) ; quedan {1,2}

Subconjunto de 2 tareas con V={1,2}:
Min (17) se alcanza al añadir Δ2 -> (, 2, 3, 4, 5, 6, 8, 7, 9, 10) ; quedan {1}
Subconjunto de 1 tarea con V={1}:
Min (6) se alcanza al añadir Δ1 -> (1, 2, 3, 4, 5, 6, 8, 7, 9, 10) ; quedan {}

Es decir una de las secuencias óptimas es la (1, 2, 3, 4, 5, 6, 8, 7, 9, 10) y para asignar las estaciones se miran los tiempos desde el principio:
(1, 2 // 3, 4 // 5, 6 // 8, 7 // 9 // 10)
Es decir, en este problema se forman 6 estaciones y no hay solución posible para 5 estaciones con el tiempo de ciclo de 17.

NOTA: En este problema hay otra secuencia alternativa al existir empate de mínimos:
(1,2,3,4,5,6,7,8,9,10).

En el siguiente cuadro se muestra de manera ordenada la solución del problema:

No	Estación	Tarea	T estándar	Precedencia
1	1	Put inspection card	6	
2		Place panel and the low case	11	1
3	2	Put supply desk PCB on conveyor	10	
4		Wire dressing for 6 spots on PCB	6	2,3
5	3	Setting PCB on the desk	7	4
6		CW direction for piece assembly	10	5
8	4	CW direction for piece with washer	8	5
7		Remove de jigs	3	5
9	5	Put PCB desk in the lower case	9	6,7,8
10	6	Connect wire	12	9

Tabla 3. Solución del problema

Para el problema propuesto, se tiene un mínimo de 6 estaciones de trabajo, ahora se va a calcular la eficiencia de la línea:

$$E = \frac{\sum_{i=1}^{n} t_i}{MC}$$

E **:** Eficiencia de la Línea
M **:** Número de estaciones utilizado
C **:** Tiempo de ciclo
t_i **:** Tarea i

$$E = \frac{82}{(6)(17)} * 100\% = 80{,}39\%$$

5. CONCLUSIONES

Esta investigación se realizó con el ánimo de conocer la funcionalidad y practicidad del uso de la programación dinámica para solucionar el problema de equilibrado de líneas de ensamble. Como se observó en el trabajo, el problema de balanceo de línea de ensamble abarca muchos temas, y en esta investigación solo se profundizo en la solución de problemas tipo SALBP-1, es decir, el más sencillo de los modelos entre los problemas de equilibrado de líneas.

El problema que se soluciono en el presente trabajo contiene 10 tareas elementales, es decir, es un problema pequeño, pero que sirve de modelo para aplicar la metodología de la programación dinámica, ya que el principal objetivo de este trabajo era mostrar como es el procedimiento de solución de un problema simple de balanceo de línea de ensamble tipo SALBP-1.

Se observa que a medida que el problema se convierte más complejo, es decir, crece en número de tareas y de restricciones, la solución, obtenida por el algoritmo de programación dinámica tiende a mostrar mejores resultados, pero su tiempo de cálculo es muy elevado.

En consecuencia a medida que el problema crece en su número de operaciones, intentar una solución con programación dinámica se vuelve complejo y poco práctico. Aunque es importante señalar que la programación dinámica es un algoritmo que busca siempre una solución optima, el tiempo computacional, es decir, el tiempo de elaboración de los cálculos se extiende y crece de forma exponencial a medida que el problema aumenta en número de tareas u operaciones.

Mediante la revisión bibliográfica, se encontró un artículo que presenta la solución de un problema de balanceo de línea de ensamble utilizando la programación dinámica, pero la complementa con una heurística para ayudar a hacerla más eficiente, el documento escrito por Bautista y Pereira **[1]** consiste en una heurística que apoya a la programación dinámica para resolver problemas de balanceo de línea más cercanos a la realidad, que contengan numerosas operaciones y tareas.

La programación dinámica es un algoritmo exacto, el uso de algoritmos exactos normalmente se da en problemas con un número de tareas reducido, ya que para problemas grandes, el número de variables y restricciones hace

inabordable el problema por cuestión de tiempo de cálculo.

En la revisión de la bibliografía, se observo que la solución de los problemas de balanceo de línea de ensamble: SALBP y GALBP, todavía son objeto de estudio e investigación, cada día se desarrollan heurísticas y se aplican nuevos algoritmos para intentar obtener soluciones optimas con tiempos de cálculo reducidos; esto aun no ha sucedido, es decir, se han explorado diferentes formas de modelar y resolver el problema, y ningún autor menciona cual es la mejor forma de modelar, ni cuál es la mejor técnica de solución. La solución a los problemas de balanceo de líneas de ensamble todavía es un tema abierto, en el cual se pueden realizar investigaciones y desarrollar técnicas más eficientes, ya que dependiendo del tipo de problema, se pueden utilizar diferentes técnicas de optimización o de aproximación para solucionar el problema.

6. BIBLIOGRAFÍA

[1] BAUTISTA J, Pereira J. A dynamic programming based heuristic for the assembly line balancing problem, European Journal of Operational Research. 2008.

[2] BAZARAA, JARVIS. Programación lineal y flujo de redes. Editorial Limusa– Noriega editores. 1994

[3] CAPACHO B, L, PASTOR, R. Generación de secuencias de montaje y equilibrado de línea, Universidad Politécnica de Catalunya, Abril 2004

[4] EGUIA S, Ignacio. Método de asignación y secuenciación de tareas en el diseño de una cadena de montaje monomodelo usando programación dinámica. Universidad de Sevilla

[5] HELD M, KARP R.M, SHARESHIAN R, Assembly line balancing — dynamic programming with precedence constraints, Operation Research. Vol 11 442–459. 1963

[6] HILLIER, LIEBERMAN. Investigación de operaciones. Mc Graw Hill. 2001

[7] JARAMILLO G, Andrés, Aplicación de la programación dinámica para resolver el problema de balanceo de línea de ensamble simple, Tesis de grado Ingeniero Industrial, Universidad Tecnológica de Pereira, Marzo de 2009.

[8] KAO, E.P.C, QUEYRANNE, On dynamic programming methods for assembly line balancing, Operations Research. Vol 30 (2) 375–390. 1982

[9] PRAWDA, W Juan. Métodos y modelos de investigación de operaciones. Editorial Limusa. 1976

[10] SCHOLL, A. Balancing and sequencing of Assembly lines. Physica-Verlag, 1999.

[11] SCHRAGE L. BAKER K.R, Dynamic programming solution of sequencing problems with precedence constraints, Operations Research. Vol. 26 444–449. 1978

[12] TAHA, Hamdy A. Investigación de operaciones, Séptima edición, editorial Pearson. 2004