Divulgaciones Matemáticas Vol. 12 No. 1(2004), pp. 25–34

Heurístico para balancear una línea de ensamblaje simple (*SALBP*, Problema tipo-1) *

*Heuristic for balancing a simple assembly line (*SALBP*, type-1 problem)*

Marval L., Fernando J. (fmarval@sucre.udo.edu.ve)
Centeno, Manuel V. (mcenteno@sucre.udo.edu.ve)
Universidad de Oriente - Núcleo de Sucre
Dpto. de Matemáticas
Cumaná, Venezuela.
Salazar G., Juan J. (jjsalaza@ull.es)
Universidad de La Laguna - Facultad de Matemáticas
Dpto. de Estadística, I.O. y Computación
Tenerife, España.

Resumen

Este trabajo trata de planificar adecuadamente la realización de un conjunto de tareas en un conjunto de máquinas de manera que el proyecto total se ejecute en el menor tiempo posible. Estos problemas de optimización se conocen como "problemas de balancear líneas de ensamblaje", y hay de muy diversos tipos. Aquí se centra la atención en dos: el caso más sencillo, en donde no hay limitación ni de la cantidad de máquinas ni sobre el tiempo de uso máximo de cualquier máquina, conocido éste como "tiempo de ciclo", y el caso más complejo, en donde se limita el "tiempo de ciclo" y se minimiza el número de máquinas. El primero se corresponde con un problema polinomial y el segundo con un problema *NP-difícil*, según la clasificación clásica en Complejidad Algorítmica. Este segundo problema se llama de tipo-1. Una de las principales aportaciones de este trabajo es la presentación de una nueva formulación como problema de Programación Lineal Entera para

Recibido 2002/02/27. Revisado 2004/01/14. Aceptado 2004/02/10.
MSC (2000): Primary 90C08, 90C10.
* Esta investigación fue parcialmente financiada por el Consejo de Investigación de la Universidad de Oriente, proyecto CI-5-1003-0977/01

el problema de balanceo de líneas de ensamblaje tipo-1, el cual podría servir para futuros trabajos computacionales. Otra aportación destacable es un método heurístico simple para resolver el problema.
Palabras y frases clave: optimización, asignación, línea de ensamblaje, heurísticas.

Abstract

This paper concerns the realization of a set of tasks in a set of machines so that the total project is performed in the smallest possible time. This kind of optimization problems is known as "simple assembly line balancing problem", and there are several types of them. We concentrate the attention to two of them: the simplest case where neither the number of machines nor the maximal time used by any machine, called "cycle time" are limited and the most complex case where the "cycle time" is limited and the number of machines is minimized. One of the principal contributions of this work is a new version of integer lineal programming for the simple assembly line balancing problem type 1, which could be used for future investigations. Another contribution to point out is a simple heuristic method to resolve the *NP-hard* problem.
Key words and phrases: pptimization, assignment, assembly line, heuristics.

1 Introducción

Un elemento fundamental en muchos procesos industriales de producción consiste en la determinación de una oportuna asignación de tareas a máquinas y secuenciación en cada máquina de las tareas a realizar de manera que se optimicen los recursos disponibles. Este trabajo tiene como finalidad aportar herramientas para ayudar a la toma de decisiones en tales contextos. Aunque la variedad de las aplicaciones es muy amplia, se trabaja en aquéllas que consideran exclusivamente dos recursos: el tiempo de procesamiento y el número de máquinas.

Este tipo de problemas se conoce en la literatura como problema de balancear líneas de ensamblaje ($ALBP$), y la versión más simple, abreviadamente $SALBP$, puede definirse de la siguiente manera [3]:

Supongamos que una empresa dispone de un conjunto (en principio sin limitación) de máquinas y debe realizar un conjunto finito de tareas. Se asume que las máquinas son todas idénticas, y que cada una puede actuar en cada momento sobre una única tarea, sucediendo que si una máquina comienza a

procesar una tarea, entonces sólo puede pasar a procesar otra tarea cuando haya finalizado aquélla (es decir, la ejecución de una tarea no es divisible). Aunque haya máquinas suficientes, se supone también que no todas las tareas pueden realizarse en paralelo, existiendo una relación de precedencia que obliga a realizar algunas tareas después de que se ha finalizado de procesar otras tareas predeterminadas por ellas. Se conoce, además, el tiempo de procesamiento que cada tarea precisa sobre cualquier máquina para que sea ejecutada. El *SALBP* pretende determinar cómo debemos asignar las tareas a las máquinas y cómo ordenar su procesamiento en cada máquina de manera que se minimice el tiempo de procesamiento del conjunto de las tareas y se minimice el número de máquinas empleadas [3].

Entre los principales algoritmos existentes para resolver la versión más sencilla se encuentran el *CPM* ("critical path method") para tiempo de duración de tareas determinísticas y el *PERT* ("program evaluation review technique") para tiempos de tareas estocásticos. Ambos algoritmos fueron propuestos en los años 50 por E. I. DuPont (USA) [9].

Los problemas tratados en este trabajo se clasifican en varios tipos. El caso sencillo es cuando no hay limitación prefijada sobre el número de máquinas ni sobre el tiempo de procesamiento de ellas, y se encuentra en la clase de problemas P (problemas con algoritmo polinomiales para su solución), ya que *PERT* y *CPM* son algoritmos polinomiales [11]. En cambio, cuando aparece alguna limitación, entonces se encuentra en la clase de problemas *NP-duro* [13], ya que el problema de empaquetamiento es un caso particular del mismo [5].

Tal como se define el *SALBP* se trata de un problema de optimización *bi-objetivo*. A partir de él existen dos problemas *uni-objetivos*:

(i) *SALBP* tipo-1: establecido un tiempo máximo de procesamiento para todas las máquinas, se debe minimizar el número de máquinas empleadas.

(ii) *SALBP* tipo-2: establecido un número máximo de máquinas a emplear, se debe minimizar el tiempo de procesamiento de la máquina que más trabaja.

Dada además sus múltiples aplicaciones, el *SALBP* en diversas versiones ha recibido numerosos estudios [1, 2, 5, 6, 8, 10, 12, 14]. Como muestra de su importancia actual conviene citar que recientemente ha sido publicado un libro enteramente dedicado a él, con los últimos avances [3].

El problema *SALBP* tipo-1 ha sido tratado por algunos autores, entre los cuales destacan Johnson [10]; Saltzman [12]; Baybars [5]; Berger, Bourjolly

y Laporte [6]; Hackman, Magazine y Wee [14]; y más recientemente Klein y Scholl [1, 2], realizando algoritmos exactos (del tipo ramificar y acotar) y heurísticos.

Aquí se utilizan ciertos resultados obtenidos en [11], tesis de maestría del primer autor con asesoría de los otros dos autores. En dicho trabajo se hace uso de la herramienta *PERT/CPM* (algoritmos polinomiales) para planificar, programar y controlar proyectos. Además, en la presente investigación se muestra un modelo de programación lineal para *SALBP tipo 1*, con el cual se hacen algunos experimentos computacionales sobre datos extraidos de la literatura [1], y se concluye con la descripción y evaluación de un algoritmo voraz (greedy). Finalmente se muestran los resultados computacionales de diversos experimentos sobre los datos anteriormente mencionados.

2 Líneas de ensamblaje

Definición 2.1. Se conoce como línea de ensamblaje a un conjunto de máquinas, enlazadas por un mecanismo de transporte y una especificación detallada de cómo fluye, de una máquina a otra, el ensamblaje del producto.

Consideremos n tareas y m máquinas idénticas (recursos limitados). Cada tarea i tiene asociada una duración t_i, y además existen ciertas relaciones de precedencia entre las tareas [4, 7]. El par (i,j) denota que la tarea i debe realizarse antes que la tarea j.

Definición 2.2. Se conoce como *tiempo de ciclo* ("*cycle time*") al tiempo máximo del que dispone una máquina para realizar el trabajo que se le asigna.

Definición 2.3. Se llama *problema de balanceo de líneas de ensamblaje tipo 1* a la búsqueda de una asignación factible de las tareas a las máquinas sin que se sobrepase el tiempo de ciclo y se minimice el número de máquinas utilizadas.

Definición 2.4. Se llama *solución factible de un problema de balanceo de líneas de ensamblaje (ALBP)* a aquella solución que cumple con lo siguiente:

1. Cada tarea se asigna exactamente a una sola máquina.
2. Se deben cumplir, por completo, las relaciones de precedencia.
3. La suma de los tiempos de las tareas de cada máquina no exceden el tiempo de ciclo, para toda máquina.

Una ilustración del problema se puede ver en la Figura 1, en donde los nodos del grafo representan las tareas y los números sobre dichos nodos representan los tiempos de ejecución de las mismas.

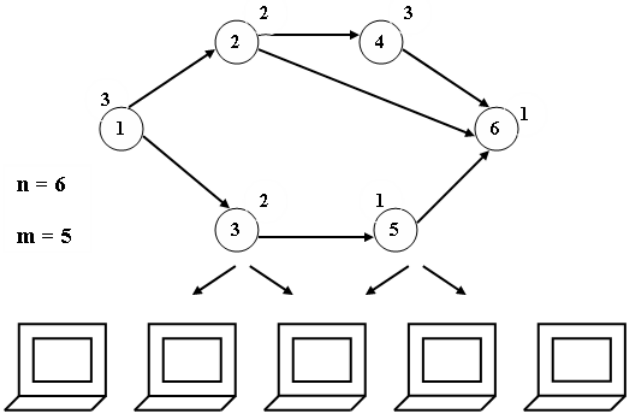

Figura 1: Precedencia entre tareas y posibles máquinas a utilizar.

Algunos problemas relacionados son:

1. *PERT/CPM*: minimiza (*ii*) sin limitar (*i*).
2. *Planificación*: en donde las máquinas suelen ser no-idénticas, y se pueden considerar otras características.
3. *Empaquetamiento:* es el *SALBP* tipo-1 sin relaciones de precedencia (el peso de los objetos es t_i, y el tamaño del contenedor es c.

2.1 Un modelo matemático.

Se sabe de la literatura que los modelos de programación lineal para resolver problemas de optimización no son los más adecuados, pero también se sabe que la representación de tales problemas a través de estos modelos puede servir como herramienta para encontrar nuevos métodos de resolución de los mismos. Nosotros en este trabajo estamos presentando una nueva formulación para el SALBP tipo-1, la cual es una variante del modelo de Patterson y Albracht [3].

Esta nueva formulación se caracteriza por generar $nm + 1$ variables enteras, las cuales son 0 o 1, donde n es el número de tareas involucradas y m ($m \leq n$), el número de máquinas iniciales; además genera $n(m + 1) + m + |P|$ restricciones, siendo $|P|$ la cardinalidad de P, teniéndose así que el modelo es $O(n^2)$.

Sea N el conjunto de n tareas a realizar, y t_i el tiempo de proceso para cada tarea $i \in \{1, 2, \ldots, n\}$. Sean además, c el tiempo de ciclo para cada máquina, W_h el conjunto de tareas que pueden asignarse a la máquina h, debido a las restricciones de precedencia, $T(W_h)$ la carga asignada a la máquina h, P el conjunto de todas las precedencias entre tareas, $Pred(i)$ el conjunto de tareas predecesoras inmediatas de i, x_{ih} variable entera no-negativa la cual representa que la tarea i es asignada a la máquina $h-$ésima.

Un modelo matemático para $SALBP$ tipo-1 es.

$$\text{Minimizar} \quad z \tag{1}$$

Subject to

$$hx_{ih} - z \leq 0 \quad \forall i = 1, 2, \cdots, n, \qquad \forall h = 1, 2, \cdots, m \tag{2}$$

$$\sum_{h=k}^{m} x_{ih} - \sum_{l=k}^{m} x_{jl} \leq 0 \quad \forall (i, j) \in P, \quad \forall k = 1, 2, \cdots, m \tag{3}$$

$$\sum_{h=1}^{m} x_{ih} = 1 \quad \forall i = 1, 2, \cdots, n \tag{4}$$

$$\sum_{i=1}^{n} t_i x_{ih} \leq c \quad \forall h = 1, 2, \cdots, m \tag{5}$$

$$x_{ih} \in \{0, 1\} \quad \forall i = 1, 2, \cdots, n; \quad \forall h = 1, 2, \cdots, m \tag{6}$$

donde, x_{ih} es uno si la tarea i se asigna a la máquina h, y cero en otro caso.

El objetivo (1) es minimizar el número de máquinas; (2) limita la máquina en la cual se puede realizar la tarea i; (3) impone el orden parcial a las tareas; (4) indica que cada tarea debe realizarse en una única máquina; (5) limita la carga máxima por cada máquina, y (6) representa un conjunto de variables booleanas para indicar cuando una tarea es asignada o no a una máquina específica.

2.2 Algoritmo

Con el modelo dado anteriormente, hemos buscado soluciones por medio de un algoritmo exacto codificado en el paquete de optimización XPRESSMP

versión 13, y una vez vista la limitación en el tiempo de ejecución con el mismo, hemos desarrollado el siguiente algoritmo voraz (en la literatura se conoce como *greedy*).

Un algoritmo voraz para el *SALBP* tipo-1 es:

Paso 1. para toda $h = 1, 2, \ldots, n$, hacer $W_h = \emptyset$ y $T(W_h) = 0$
Paso 2. para toda tarea $j = 2, 3, \ldots, n$, identificar Pred(j)
Paso 3. para toda tarea $i = 1, 2, \ldots, n$, hacer asignada(i) $\leftarrow falso$
Paso 4 $h = 1$
Repetir para $j = 2$ hasta n
mientras Pred(j) $\neq \emptyset$ hacer
seleccionar $i \in$ Pred(j) tal que $t_i \geqslant t_k \ \ \forall k \in$ Pred(j)
mientras asignada(i) $\neq$ *verdad* hacer
si $t_i \leqslant c - T(W_h)$ entonces $T(W_h) \leftarrow T(W_h) + t_i$,
asignada(i) $\leftarrow$ *verdad*, Pred(j) $\leftarrow$ Pred(j) - $\{i\}$
si $t_i > c - T(W_h)$ entonces
para $l = h - 1$ hasta 1 hacer
si $t_i \leqslant T(W_l)$ entonces $T(W_l) \leftarrow T(W_l) + t_i$,
asignada(i) $\leftarrow$ *verdad*, Pred(j) $\leftarrow$ Pred(j) - $\{i\}$
si no
$h \leftarrow h + 1$, $T(W_h) \leftarrow T(W_h) + t_i$,
asignada(i) $\leftarrow$ *verdad*, Pred(j) $\leftarrow$ Pred(j) - $\{i\}$
si no $h \leftarrow h + 1$, $T(W_h) \leftarrow T(W_h) + t_i$,
asignada(i) $\leftarrow$ *verdad*, Pred(j) $\leftarrow$ Pred(j) - $\{i\}$

Pred(j) $\leftarrow$ Pred(j) - $\{i\}$
Paso 5. Escribir h, número de máquinas usadas

En el algoritmo anterior se comienza con las máquinas vacías, se identifican las tareas predecesoras de cada tarea, luego se selecciona, para cada tarea, de sus predecesores, la de mayor duración, y se coloca en la máquina disponible en ese momento; si tal tarea no puede ser colocada allí, porque viola el tiempo de ciclo, entonces se abre una nueva máquina, y se asigna en ella dicha tarea. El proceso se continua hasta que se hayan agotado todas las tareas.

2.3 Algunos resultados computacionales.

Con la nueva formulación lineal entera, mostrada en 2.1, hemos hecho experimentos computacionales, los cuales nos demuestran que a partir de $n = 45$, el tiempo de computador para resolver el problema es muy elevado. Como ejemplo, en la tabla 1, vemos que para Warnecke de 58 tareas, el tiempo de

consumo es de 5003 segundos, equivalente a unas 1.39 horas, y sin finalizar el proceso. Esto muestra la necesidad de la implementación de los métodos heurísticos para resolver el problema. Nosotros, en nuestro trabajo, presentamos un heurístico simple voraz para encontrar una cota superior para el problema SALBP tipo-1.

Los resultados computacionales que se obtienen con el algoritmo exacto SALOME [1], y los obtenidos por el nuevo algoritmo heurístico se muestran en la tabla 1. En la misma se señalan los datos extraídos de la literatura, correspondiente a autores que han trabajado en el tema.

Los algoritmos fueron implementados en lenguaje C, y ejecutados en un PC Pentium 133MHz /32 Mb RAM.

Como se observa en la tabla 1, con nuestro heurístico, el consumo en segundos del computador fue inapreciable. Además, en los casos en los que no encontró la solución óptima, la desviación fue de sólo de una máquina. Esto demuestra los buenos resultados del algoritmo que presentamos.

Tabla 1. Resultados computacionales para SALBP-1

nombre	n	*npre*	ciclo	m(H)	Ti(H)	Salo	Ti(Salo)	SML	Ti(ML)
Mertens	7	6	18	2	0.00	2	0.00	2	0
Jaeschke	9	11	9	5	0.00	5	0.00	5	0
Jackson	11	13	10	5	0.00	5	0.00	5	1
Mansoor	11	11	45	5	0.00	5	0.00	5	1
Laporte	20	19	138	7	0.00	7	0.00	7	34
Heskia	28	39	138	9	0.00	8	0.06	8	35
Sawyer30	30	39	41	9	0.00	8	0.06	8	236
Kilbridge	45	62	79	8	0.00	7	0.05	7	810
Hahn	53	82	2004	8	0.00	8	0.05	8	2167
Warnecke	58	70	92	18	0.01	17	0.17	-	5003
Tonge-70	70	86	182	20	0.01	20	0.11	–	–
Wee-Mag	75	86	108	32	0.01	32	0.05	–	–
Arc83	83	113	8412	10	0.02	10	0.11	–	–
Arc111	111	177	16723	10	0.02	9	0.16	–	–
Barthold	148	175	403	15	0.03	14	0.38	–	–

n: Número de tareas.
npre: Número de precedencias entre tareas.
ciclo: Tiempo de ciclo que utiliza cada máquina para realizar las tareas asignadas a la misma.
m(H): Número de máquinas que da como solución el heurístico.
Ti(H): Tiempo que registra el heurístico en la ejecución en segundos.

Salo: Número mínimo de máquinas que encuentra el algoritmo SALOME.
Ti(Salo): Tiempo que tarda la ejecución con SALOME en segundos.
SML: Solución hallada con el modelo lineal entero
Ti(ML): Tiempo empleado por el modelo lineal entero en segundos.

3 Conclusiones

En este trabajo hemos estudiado un problema de optimización relacionado con el balanceo de líneas de ensamblaje, en el cual se consideran ciertas limitaciones. Desde el punto de vista de la Complejidad Computacional este problema es *NP-difícil*. Una aportación importante de este trabajo, es la nueva formulación lineal entera para *SALBP-tipo-1*, que resolvemos usando XPRESSMP. Otra aportación destacable es un método heurístico para resolver el problema, y que ha sido implementado y analizado por nosotros en este trabajo. Queda abierto para trabajos futuros desarrollar algún algoritmo exacto que reduzca los tiempos de cálculo de XPRESSMP sobre el modelo lineal entero presentado aquí.

Referencias

[1] Armin S., Klein R. SALOME: A bi-directional branch-and-bound procedure for assembly line balancing. INFORMS *J. of Computing*, 9 (1997) 319-334.

[2] Armin S., Klein R. Balancing assembly lines effective - A computational comparison *European J. of Oper. Res.*, 114 (1999) 50-58.

[3] Armin S. *Balancing and Sequencing of Assembly Lines*. Second revised edition. Physica-Verlag Heidelberg. Germany, 1999.

[4] Baase S., Gelder A. *Computer Algorithms: Introduction to Design and Analysis*. Third Edition. Addison Wesley, 2000.

[5] Baybars I. A survey of exact algorithms for the simple assembly line balancing problem. *Magnt. Sc.* 32 (1986) 909-931.

[6] Berger I., Bourjolly J., y Laporte G. Branch-and-bound algorithms for the multi-product assembly line balancing problem. *European J. of Oper. Res.*, 58 (1992) 215-222.

[7] Chartrand G., Oellermann O. *Applied and Algorithmic Graph Theory*. International Series in Pure and Applied Mathematics. McGraw-Hill International Editions. International Edition, 1993.

[8] Fatih H., Rachamadugu R., y Papachristou Ch. Designing paced assembly lines with fixed number of stations. *European J. of Oper. Res.*, 102 (1997) 488-501.

[9] Hillier F., Lieberman G. *Introducción a la Investigación de Operaciones*. Cuarta edición (segunda edición en español). McGRAW-HILL, 1989.

[10] Johnson R. Optimally balancing large assembly lines with FABLE. *Magnt. Sc.*, 34 (1988) 240-257.

[11] Marval F. Balanceando una línea de ensamblaje simple (Problema tipo-1). Tesis de Maestría en Matemáticas, Universidad de Oriente, Cumaná, Venezuela. 2001.

[12] Saltzman M., y Baybars I. A two process implicit enumeration algorithm for the simple assembly line balancing problem. *European J. of Oper. Res.* 32 (1987), 118-129.

[13] Thatcher J., y Miller R. Reducibility Among Combinatorial Problems. *Complexity of Comp. Appl.* (1972), 85-104

[14] Wee T., Hackman S., y Magazine M. Fast, Effective Algorithms for Simple Assembly Line Balancing Problems. *Oper. Res.*, 37 (1988) 916-924.