V Congreso de Ingeniería de Organización
Valladolid-Burgos, 4-5 Septiembre 2003

Índice

Experimentación con PLEM para la satisfacción de restricciones del Car-Sequencing Problem.

Joaquin Bautista Valhondo [1], Jordi Pereira Gude[2]
[1] Doctor Ingeniero Industrial, ETSII de Barcelona UPC, Diagonal 647-08028 BCN, joaquin.bautista@upc.es
[2] Ingeniero Organización Industrial, ETSII de Barcelona UPC, Diagonal 647-08028 BCN, jorge.pereira@upc.es

RESUMEN

Se presenta un modelo de Programación Lineal Entera-Mixta: PLEM (Mixed-Integer Linear Programming: MILP) para el problema de satisfacción de restricciones del Car-Sequencing Problem. Se experimenta con diversas variantes del modelo construidas a partir de la combinación de bloques elementales de restricciones representativas del problema. Finalmente, se estudia la aplicabilidad de la PLEM al problema mediante una experiencia compuesta por ejemplares de prueba estándares en la literatura sobre el tema.

***Palabras clave**: MILP, PLEM, Car-Sequencing Problem, Secuenciación.*

1. Introducción.

Una variante del problema de secuenciación de unidades en líneas de montaje de productos mixtos es el denominado *Car-Sequencing Problem*. El problema, definido por Parello, Kabat & Wos [3,4], ha sido tratado principalmente y recibido especial atención en la comunidad científica dedicada a la Programación por Restricciones (*Constraint Programming*) [1,5], y se ha demostrado que es NP-*completo*. En el problema, se debe establecer el orden de entrada a una línea de montaje de un conjunto de unidades (*cars*) de distintos tipos.

Cada unidad de un determinado tipo requiere un subconjunto de opciones que forman parte de un conjunto completo de opciones especiales (techo solar, aire acondicionado, protectores laterales, etc.) que, normalmente, requieren un tiempo de aplicación, en la estación destinada a tal efecto, superior al tiempo estándar concedido en consonancia con la tasa de producción requerida a la línea. Por ello, desafortunadamente, cada opción está sujeta a un conjunto de restricciones de capacidad de la forma p_j/q_j, que se interpreta así: el número máximo de veces que puede aparecer la opción j en todo segmento de la secuencia de longitud q_j es igual a p_j. Aquí proponemos también una extensión natural del problema considerando restricciones de mínima carga $r_j\backslash s_j$: el número mínimo de veces que debe aparecer la opción j en todo segmento de la secuencia de longitud s_j es r_j.

En el presente trabajo nos centramos en la propuesta de un modelo de PLEM para el problema. Se realiza una experiencia computacional empleando diversas variantes del modelo y un conjunto de colecciones de ejemplares extraídas de la literatura sobre el tema, con el propósito de estudiar la aplicabilidad de la PLEM para resolver el *Car-Sequencing Problem.*

El texto que inmediatamente sigue se ha estructurado de la siguiente forma: en la sección 2 se establece la nomenclatura y los elementos de los modelos que, posteriormente, se proponen;

la sección 3 está dedicada al modelo básico de satisfacción de restricciones con el que se enfoca la resolución del problema hacia la técnica Programación por Restricciones (*Constraint Programming*: CP) para la versión genuina del *Car-Sequencing Problem*; en la sección 4 se extiende el modelo anterior a un modelo de Programación Lineal Entera-Mixta: PLEM (*Mixed-Integer Linear Programming: MILP*) proponiendo algunas funciones objetivo y restricciones adicionales; en la sección 5 se realiza una experiencia computacional explotando uno de los modelos de PLEM para resolver una colección de ejemplares de referencia en la literatura sobre el tema; y finalmente, la sección 6 se dedica a las conclusiones extraídas del trabajo.

2. Nomenclatura y elementos de los modelos para el CSP.

2.1 Datos.

Sea:

P el número de tipos de unidades-producto (*cars*).
C el número de tipos de opciones.
U el número total de unidades a secuenciar.
u_i el número de unidades de tipo i (i=1,..,P).
n_{ji} el valor representativo de la presencia (=1) o ausencia (=0) de la opción j en una unidad de tipo i (i=1,..,P ; j=1,..,C).
q_j longitud del tramo de capacidad máxima asociado a la opción j.
p_j el número máximo de veces que la opción j puede estar presente en cualquier tramo de la secuencia de longitud q_j.
s_j longitud del tramo de capacidad mínima asociado a la opción j.
r_j el número mínimo de veces que la opción j debe estar presente en cualquier tramo de la secuencia de longitud s_j.

2.2 Variables.

Sea:

x_{it} variable binaria que adopta el valor 1 si la t-ésima unidad de la secuencia es de tipo i, y adopta el valor 0, en caso contrario (i=1,..,P ; t=1,..,U).
y^-_{jt} holgura de la opción j en el tramo [t - q_j+1, t] de la secuencia (j=1,..,C; t= q_j, U).
y^+_{jt} descarga de la opción j en el tramo [t - s_j+1, t] de la secuencia (j=1,..,C; t= s_j, U).

3. Modelo básico de satisfacción de restricciones para el CSP.

La modelización original del CSP se basa en la programación por restricciones. El objetivo consiste en encontrar una solución que satisfaga el máximo número de restricciones representativas, académicamente todas ellas, de las condiciones impuestas al problema.

Dado un ejemplar del *Car-Sequencing* (P,C,U,u_i,n_{ji},p_j/q_j), el objetivo es hallar una secuencia de U unidades que satisfaga los bloques de restricciones (1), (2), (3) y (4).

$$\sum_{i=1}^{P} x_{it} = 1 \qquad \text{para} \quad 1 \le t \le T \tag{1}$$

$$\sum_{t=1}^{U} x_{it} = u_i \qquad para \quad 1 \le i \le P \tag{2}$$

$$\sum_{i=1}^{P} \sum_{k=t-q_j+1}^{t} n_{ji} x_{ik} \le p_j \qquad para \quad 1 \le j \le C; q_j \le t \le U \tag{3}$$

$$x_{it} \in \{0,1\} \qquad para \quad 1 \le i \le P; 1 \le t \le U \tag{4}$$

En el modelo, el bloque de restricciones (1) asegura que la secuencia es compacta, esto es, que en todos los instantes de secuenciación, se asigna una única unidad de producto. El segundo bloque asegura el cumplimiento del programa de producción, es decir, en la secuencia aparecen todas las unidades de cada tipo requeridas originalmente. El tercer bloque se corresponde con las restricciones p_j/q_j, cuyo objetivo es asegurar que en todo segmento de longitud q_i de la secuencia, el número máximo de unidades de tipo i presentes en él, no excede a p_i unidades. Finalmente, el bloque (4) asegura la integridad de variables.

El problema, y este modelo, ha servido como referencia para el estudio de restricciones globales, habiendo sido tratado como ejemplo por gran parte de los paquetes de propagación de restricciones, entre los que se encuentran Chip, Eclipse e Ilog-Solver.

4. Modelo de Programación Lineal Entera-Mixta (MILP) para el problema CSP.

La primera adaptación propuesta del modelo anterior, restricciones (1),(2),(3) y (4), consiste en sustituir el bloque de restricciones (3) por el siguiente:

$$y_{jt}^{-} + \sum_{i=1}^{P} \sum_{k=t-q_j+1}^{t} n_{ji} x_{ik} = p_j \qquad para \quad 1 \le j \le C; q_j \le t \le U \tag{5}$$

donde las variables y^{-}_{jt}, representan las holguras del bloque de restricciones (3). En el presente trabajo se explicitan por su importante significación. En efecto, estas variables representan el número de veces que está ausente la opción j en todo tramo $[t - q_j+1, t]$ de longitud q_j, de la secuencia cuando se impone una presencia máxima de la opción j, igual a p_j.

Esto nos lleva también el concepto de descarga, y^{+}_{jt}, cuyo significado es el número de veces que está ausente la opción j en todo tramo $[t - s_j+1, t]$ de la secuencia de longitud s_j, cuando se impone una presencia mínima de opción j, igual a r_j. Este concepto puede asociarse a un conjunto de restricciones que establezcan el cumplimiento de estos mínimos, a las que se denomina restricciones de tipo $r_j\backslash s_j$:

$$\sum_{i=1}^{P} \sum_{k=t-s_j+1}^{t} n_{ji} x_{ik} - y_{jt}^{+} = r_j \qquad para \quad 1 \le j \le C; s_j \le t \le U \tag{6}$$

A partir de los conceptos de holgura y descarga, puede establecerse una familia de funciones objetivo para la modelización del problema mediante Programación Lineal Entera-Mixta. El objetivo, por tanto, será la optimización de la función objetivo y la satisfacción de las restricciones originales, por tanto cualquier solución obtenida para este problema será a su vez solución del problema básico de satisfacción de restricciones.

Como función objetivo general, se propone una función que involucre tanto el concepto de descarga como el de holgura:

$$[OPT]\ Z = \sum_{1\le j\le C; q_j\le t\le U} \alpha_{jt}^{-} y_{jt}^{-} + \sum_{1\le j\le C; s_j\le t\le U} \alpha_{jt}^{+} y_{jt}^{+} \qquad (7)$$

donde α_{jt}^{-} y α_{jt}^{+} representan, respectivamente, las ponderaciones de las holguras y descargas para la opción *j* en el instante de secuenciación *t*.

Esta forma general de función objetivo permite numerosas alternativas para tratar de optimizar las cargas y descargas por opciones y a lo largo del tiempo. Por ejemplo, si se desea penalizar de igual modo la ausencia de todas las opciones a lo largo del tiempo en las restricciones *p/q*, bastará con hacer [OPT]=[MIN], $\alpha_{jt}^{-}=1$ (para todo $j=1,..,C$; $t= q_j, U$) y $\alpha_{jt}^{+}=0$ (para todo $j=1,..,C$; $t= s_j, U$).

5. Experiencia Computacional.

La colección de instancias que se ha utilizado como referencia procede de la colección de problemas para el Car Sequencing Problem disponible en la librería CSPLib, véase [2], que incluye instancias de diversos problemas de satisfacción de restricciones. La colección para el CSP está formada por ocho juegos de problemas. El primer juego está formado por nueve instancias de la literatura, de las cuales tres problemas se conocen como factibles, tres son infactibles y otros tres continúan abiertos. Los otros siete juegos están conformados, cada uno, por diez instancias factibles con un porcentaje de consumo de recursos entre el 60% y el 90%, en aumentos de 5 según juego.

Estas instancias han sido tratadas en dos experiencias computaciones distintas. Cada una de ellas ha sido realizada en un ordenador, con el objetivo de explorar las diferencias resultantes del ajuste de los parámetros de un mismo solver. El mismo programa fue implementado en C, utilizando como Solver de programación lineal la librería ILOG-CPLEX en su versión 6.5.

La primera experiencia computacional corresponde a un PC con microprocesador Pentium II a 450 MHz., 128 Mb. de RAM y sistema operativo Windows 98. Los parámetros de configuración son los originales.

La segunda experiencia computacional utiliza una estación de trabajo Sun Ultra Enterprise 450, equipada con cuatro procesadores Sparc a 400 MHz. con una memoria conjunta de 1 Gb. de RAM bajo sistema operativo Solaris 7. La versión de CPLEX utilizada sólo hace uso de uno de los microprocesadores, y los parámetros han sido configurados a lo largo de diversos experimentos con diferentes problemas de similares características.

Los experimentos realizados en el presente trabajo consideran ambos: [OPT]=[MAX], $\alpha_{jt}^{-}=1$ (para todo $j=1,..,C$; $t= q_j, U$) y $\alpha_{jt}^{+}=0$ (para todo $j=1,..,C$; $t= s_j, U$), cuyo significado es maximizar la holgura de todas las opciones a lo largo del tiempo en las restricciones *p/q* (es decir, maximizar la holgura total).

La tabla 1 del apéndice 1 presenta los resultados obtenidos por el ordenador PC, así mismo la tabla 2 del mismo apéndice muestra los resultados obtenidos por el ordenador SUN.

6. Conclusiones.

Se ha propuesto un modelo de PLEM para hallar soluciones del problema de satisfacción de restricciones CSP. Se ha realizado una experiencia computacional compuesta por dos experimentos que emplean dos ordenadores distintos y un mismo juego de ejemplares del problema que son referencia en la literatura especializada en el problema. En los dos experimentos se emplea el mismo Solver de programación lineal compatible con los respectivos ordenadores empleados. Los resultados obtenidos en ambos experimentos son altamente satisfactorios: se ha hallado solución en todos los ejemplares de la literatura para los que se conoce una solución obtenida por paquetes de programación por restricciones, entre los que se encuentran Chip, Eclipse e Ilog-Solver, especializados en el estudio y tratamiento de restricciones globales; los tiempos empleados para obtener soluciones por el Solver de programación lineal son competitivos con los requeridos por los paquetes de programación por restricciones.

Referencias

[1] Dincbas, M., H. Simonis, y P. van Hentenryck (1988) “Solving the car sequencing problem in constraint logic programming”. En *European Conference on Artificial Intelligence, ECAI-88*, pp. 290-295. Munich.

[2] I.P. Gent y T. Walsh (1999) "CSPLib: a benchmark library for constraints", *Technical report APES-09-1999*

[3] Parello, B.D. (1988) “CAR WARS: the (almost) birth of an expert system” *AI Expert*, 3(1): 60-64. Citado en [1].

[4] Parello, B.D., W.C. Kabat and L. Wos (1986) “Job-shop scheduling using automated reasoning” *Journal of Automated Reasoning*, 2(1): 1-42. Citado en [1].

[5] Regin, J.C. and J.F. Puget (1997) “A filtering algorithm for global sequencing constraints” En *Principles and Practice of Constraint Programming*, CP-97, pp. 32-46. LNCS 1330.

Anexo 1: Resultados del experimento para la resolución del CSP mediante PLEM.

Ejemplar	Descarga	Tiempo(seg)		Ejemplar	Descarga	Tiempo(seg)		Ejemplar	Descarga	Tiempo(seg)		Ejemplar	Descarga	Tiempo(seg)
4/72	76	366		65/02	495	188		75/02	356	368		85/02	317	583
6/76	*			65/03	453	511		75/03	359	169		85/03	551	237
10/93	*			65/04	490	477		75/04	361	254		85/04	272	1166
16/81	52	1063		65/05	486	202		75/05	374	235		85/05	240	379
19/71	*			65/06	448	198		75/06	327	306		85/06	277	279
21/90	*			65/07	488	667		75/07	410	205		85/07	290	305
36/92	*			65/08	436	182		75/08	341	233		85/08	289	1146
41/66	125	240		65/09	472	198		75/09	342	216		85/09	230	3452
26/82	*			65/10	477	134		75/10	362	295		85/10	323	1041
60/01	538	230		70/01	439	200		80/01	324	356		90/01	212	9361
60/02	554	112		70/02	426	239		80/02	302	322		90/02	212	9374
60/03	503	137		70/03	419	223		80/03	402	652		90/03	212	9334
60/04	554	128		70/04	426	224		80/04	289	276		90/04	212	9817
60/05	567	150		70/05	427	242		80/05	351	1301		90/05	212	9525
60/06	506	197		70/06	391	663		80/06	280	597		90/06	432	315
60/07	520	122		70/07	445	226		80/07	360	959		90/07	196	1640
60/08	477	143		70/08	385	340		80/08	298	348		90/08	225	2718
60/09	541	285		70/09	401	161		80/09	308	1271		90/09	368	3625
60/10	533	155		70/10	426	214		80/10	329	284		90/10	244	1326
65/01	498	445		75/01	384	264		85/01	283	1580		*INFACTIBLE		

Tabla 1: Valores de la holgura total máxima y del tiempo de ejecución (s.) relativos al experimento 1. Se emplea PC - microprocesador Pentium II a 450 MHz., 128 Mb. de RAM, W98-.

Ejemplar	Descarga	Tiempo(seg)		Ejemplar	Descarga	Tiempo(seg)		Ejemplar	Descarga	Tiempo(seg)		Ejemplar	Descarga	Tiempo(seg)
4/72	76	64		65/02	495	121		75/02	357	153		85/02	320	223
6/76	*	*		65/03	453	120		75/03	359	152		85/03	551	141
10/93				65/04	490	63		75/04	362	187		85/04	272	1537
16/81	54	410		65/05	486	104		75/05	374	412		85/05	240	979
19/71	*	*		65/06	448	156		75/06	327	423		85/06	277	744
21/90	*	*		65/07	488	305		75/07	410	162		85/07	293	243
36/92	*	*		65/08	436	103		75/08	341	158		85/08	289	858
41/66	125	110		65/09	472	295		75/09	342	462		85/09	230	931
26/82	*	*		65/10	476	129		75/10	363	125		85/10	320	566
60/01	539	431		70/01	439	422		80/01	325	173		90/01	212	1678
60/02	554	68		70/02	426	119		80/02	304	672		90/02	212	1678
60/03	503	211		70/03	419	428		80/03	402	159		90/03	212	1682
60/04	554	204		70/04	426	151		80/04	288	222		90/04	212	1684
60/05	567	22		70/05	427	158		80/05	351	509		90/05	212	1682
60/06	506	113		70/06	391	174		80/06	283	204		90/06	434	100
60/07	520	117		70/07	445	134		80/07	360	407		90/07	196	836
60/08	477	89		70/08	388	138		80/08	298	217		90/08	225	1658
60/09	541	77		70/09	401	168		80/09	308	1169		90/09	368	781
60/10	531	214		70/10	426	148		80/10	329	160		90/10	244	2914
65/01	498	320		75/01	386	618		85/01	283	996		* INFACTIBLE		

Tabla 2: Valores de la holgura total máxima y del tiempo de ejecución (s.) relativos al experimento 2. Se emplea Sun Ultra Enterprise 450 - equipada con 4 procesadores Sparc a 400 MHz. (se utiliza uno), memoria conjunta de 1 Gb. de RAM bajo sistema operativo Solaris 7.