ALGORITMO MULTIPARTE BASADO EN GRÁFICOS PARA LA PROGRAMACIÓN DEL PERSONAL EN EMPRESAS DE TRANSPORTE

Ademir Aparecido Constantino
Universidad Estatal de Maringá
ademir@din.uem.br
Roger Calvi
Universidad Estatal de Maringá
calvi@din.uem.br
Silvio Alexandre de Araujo
Universidade Estadual Paulista
saraujo@ibilce.unesp.br Cândido
Ferreira Xavier de Mendonça Neto
Universidad de San Pablo
cfxavier@usp.br

RESUMEN

El problema de la programación de la tripulación es encontrar un conjunto de viajes que cubran la parada de autobús diaria de una empresa de transporte público, minimizando los costos y cumpliendo con un conjunto de leyes y regulaciones laborales de la empresa. En este trabajo, se propone un nuevo algoritmo heurístico de dos fases para resolver el problema de programación de la tripulación utilizando problemas de asignación múltiple que surgen de un modelo basado en un gráfico multiparte. La primera fase del algoritmo crea una solución inicial. En la segunda fase, el algoritmo intenta mejorar la solución encontrada en la primera fase modificando la matriz de costes del problema de asignación. En esta etapa, se proponen dos modelos diferentes. Las pruebas computacionales se realizan utilizando datos del mundo real de dos empresas de transporte por carretera urbana e instancias aleatorias extraídas de datos reales. Los resultados obtenidos se comparan con los resultados de un modelo de cobertura establecido y con límites inferiores calculados para las instancias. El algoritmo mostró buenos resultados comparando la calidad de las soluciones y el tiempo computacional.

Palabras clave: problema de programación de la tripulación, gráfico multiparte, problema de asignación.

RESUMEN

El Problema de Programación de Tripulaciones consiste en encontrar un conjunto de deberes que cubra el horario diario de una empresa de transporte público, con costos mínimos y cumpliendo un conjunto de leyes laborales y regulaciones de la empresa. En este trabajo se propone un nuevo algoritmo heurístico de dos fases para resolver el problema de programación de la tripulación utilizando problemas de asignación múltiple que surgen del modelo basado en grafo multipartito. En la primera fase del algoritmo se construye una solución inicial. En la segunda fase, el algoritmo intenta mejorar la solución inicial modificando la matriz del costo del problema de asignación. En esta fase se proponen dos modelos diferentes. Las pruebas computacionales se realizan utilizando datos del mundo real de dos empresas de bus urbano e instancias aleatorias extraídas de datos reales. Los resultados alcanzados se comparan con los resultados de una resolución utilizando el modelo de cobertura de conjuntos y con los enlaces más bajos calculados para las instancias. El algoritmo alcanzó buenos resultados comparando las soluciones de calidad y el tiempo computacional

Palabras clave: problema de programación de la tripulación, gráfico de múltiples partes, problema de asignación.

1 Problema de programación de la tripulación

El Problema de Programación de Tripulaciones (PET) consiste en generar un horario de trabajo para la cuadrilla, es decir, un conjunto de jornadas laborales de tal manera que se cubra todo un horario de bus predefinido, minimizando así los costos. Además, es necesario que se cumplan las restricciones legales que establece la ley y los contratos con el sindicato de la categoría, así como la normativa de la empresa, lo que aumenta aún más la complejidad del problema.

Algunas definiciones básicas relacionadas con la PET son: ***Cuadra:*** una secuencia de viajes que un vehículo tiene que hacer en un día, generalmente comenzando y terminando en el garaje. El conjunto de bloques forman el*escala de autobús;* ***Oportunidad de intercambio:*** es un lugar y un momento en el que se permite a la tripulación abandonar o hacerse cargo de un vehículo; ***Asignación:*** la parte de trabajo entre dos oportunidades de intercambio consecutivas. Por tanto, durante la realización de una tarea no es posible cambiar de tripulación. Las tareas son los datos de entrada para resolver el PET;***Fracción de trabajo:*** o simplemente ***fracción,*** contiene una serie de tareas consecutivas; ***Viaje:*** el trabajo diario que debe realizar una tripulación y comprende una o más fracciones de trabajo; ***Tripulación:*** generalmente involucra un controlador y un recolector, pero en algunos casos puede referirse a un solo controlador;

Entre los modelos más utilizados para resolver el problema de programación de la tripulación se encuentran el Problema de cobertura de conjuntos (PCC) y el Problema de partición de conjuntos (PPC), que se expresan como problemas de programación lineal de enteros (PLI).

ser *No* el número de viajes potenciales (columnas), *metro* el número de fracciones del día (líneas) a cubrir y $C_j$el costo asociado con el viaje *j.* También considerando las constantes entero La_{ij}asumiendo valor 1 si el viaje *j* cubrir la fracción Iy, en caso contrario, el valor 0, entonces se puede formular el siguiente modelo PCC:

$$\text{Min} \sum_{j=1}^{No} C_j X_j \quad (1)$$

$$\text{Sujeto a: } \sum_{j=1}^{No} La_{ij} X_j \, 1, \; I \in \{1,2,,metro\} \quad (dos)$$

$$X_j \in \{0, 1\}, \; j \in \{1,2,, \; norte\} \quad (3)$$

La función objetivo (1) minimiza el costo total. La restricción (2) asegura que cada fila (fracción de trabajo) esté cubierta por al menos una columna (día laborable), y la restricción de completitud (3) asegura que solo se considerarán las horas completas.

Al cambiar la restricción (2) a una restricción de igualdad, ahora tenemos el modelo conocido como el problema de la partición de conjuntos.

Los dos modelos son bastante similares, excepto que la desigualdad en el modelo de cobertura de conjuntos se reemplaza por la igualdad en el modelo de partición. Sin embargo, el número total de viajes posibles (columnas de la matriz) suele ser muy grande. Esto ha dado lugar a algunas técnicas heurísticas para reducir el número de columnas (viajes) o, alternativamente, el problema se ha descompuesto en varios subproblemas y se ha resuelto por separado. Tanto el PCC como el PPC son problemas NP-difíciles (KARP, 1975 y KARP, 1976*apud* GOLDBARG y LUNA, 2000).

Para la resolución de estos modelos, se pueden encontrar varios trabajos en la literatura, tales como: Beasley (1987) quien propuso un algoritmo exacto para el PCC; Beasley y Chu (1996) presentaron un algoritmo genético; Caprara, Fischetti y Toth (1999) propusieron una heurística basada en la relajación lagrangiana; Wren y Wren (1995) utilizaron un algoritmo genético con un nuevo operador de cruce para resolver la programación diaria de conductores; Mayerle

(1996) propuso un algoritmo genético; Fortsyth y Wren (1997) aplicaron la metaheurística *sistema de hormigas* para resolver la programación de la tripulación; Ohlsson, Peterson y Söderberg (2001) desarrollaron una red neuronal artificial con retroalimentación de campo medio para resolver el PCC; Li y Kwan (2000) presentaron un algoritmo *evolución simulada difusa* similar a los algoritmos genéticos); Mauri y Lorena (2004) describieron una metodología interactiva basada en la aplicación del algoritmo de entrenamiento poblacional (ATP) junto con la programación lineal (PL).

Aunque la mayoría de los trabajos encontrados en la literatura tratan sobre métodos que utilizan la cubierta o modelos de partición de conjuntos para resolver el problema de programación de la tripulación (PET), hay algunos trabajos que utilizan métodos heurísticos para resolver PET sin utilizar tales modelos. Shen y Kwan (2000) desarrollaron el algoritmo HACS que utiliza una búsqueda tabú para resolver el problema de programación de la tripulación; Souza, Cardoso y Silva (2003) abordaron el problema de la programación de la tripulación utilizando metaheurísticas de extinción simuladas, búsqueda tabú y una hibridación de estas dos técnicas; Siqueira (1999) resuelve el PET mediante el problema de emparejamiento en un gráfico no bipartito, donde los recorridos están formados por, como máximo, dos fracciones.

2 Algoritmo propuesto

El algoritmo propuesto se basa en la resolución consecutiva de problemas de asignación. El problema de asignación es equivalente al costo mínimo de emparejamiento perfecto en un gráfico. bipartito. Dada una matriz de costos [C_{ij} de dimensión No X No, el problema de asignación consiste en asociar cada fila con una columna diferente de tal manera que la suma de los costos sea mínima. Usando una variable binaria $X_{ij} = 1$ si la linea I está asociado con la columna j, el problema puede ser formulado de la siguiente manera:

$$\text{Min} \sum_{yo=1}^{No} \sum_{j=1}^{No} C_{ij} X_{ij} \tag{8}$$

Sujeto a:

$$\sum_{j=1}^{No} X_{ij} = 1 \qquad yo = 1,2,, No \tag{9}$$

$$\sum_{yo=1}^{No} X_{ij} = 1 \qquad j = 1,2,, No \tag{10}$$

$$X_{ij} \in \{0, 1\} \qquad yo, j = 1,2,, No \tag{11}$$

El algoritmo se divide en dos fases: 1) construcción de una solución inicial; y 2) mejora de la solución. En ambas fases se utiliza el modelo de designación anterior, como se describe a continuación.

2.1 Construcción de la solución inicial

El algoritmo comienza con la construcción de un gráfico multipartito. considera el gráfico $G = (T, A)$, Dónde T es el conjunto de vértices que representan las tareas a las que tripulaciones y LA es el conjunto de aristas La_{ij} que indican la posibilidad de que la misma tripulación realice la tarea j después de la tarea I. También considere que los vértices están organizados en capas y que cada una de estas capas está compuesta por vértices que representan tareas que se pueden realizar como una secuencia de alguna tarea en la capa inmediatamente anterior, y que no se pueden realizar después de tareas ubicadas en la misma capa o capas posteriores.

Por lo tanto, el conjunto de vértices T está dividido en subconjuntos, $T_1, T_{dos}, ..., T_K$, que representan los vértices (tareas) de cada capa, así, $T_{dos} \cup T_{dos} \cup ... \cup T_K = T$, ser K el número de capas (o particiones).

La construcción de la solución inicial consiste en resolver un problema de asignación para cada capa. Cada problema de asignación definido por la matriz cuadrada $C = [c_{ij}]$, de orden $|T_K| + |J|$, ser J el conjunto de viajes. El tamaño del viaje establecido. J debería ser suficiente para que ninguna tarea quede sin asignar. La matriz $C = [c_{ij}]$ se puede dividir en cuatro bloques como se muestra en la Figura 2, y sus elementos se definen como sigue.
molde:

- **I.** Para $yo = 1, ..., |J|$ y $j = 1, ..., |T_K|$, si es posible asignar la tarea j el viaje I, tenemos $C_{ij} = f(i, j) + CTL$, Dónde $j \in T_{OK}$, la función F establece el costo de asignarte a ti mismo a la tarea j el viaje I, y CTL toma un valor que penaliza el cambio de línea de viaje en el viaje si la línea de viaje relacionada con la tarea j es diferente de la línea de la última tarea del viaje I y de lo contrario valor cero. Si no puede asignar la tarea j el viaje I luego $C_{ij} = \infty$;
- **II.** Para $yo = 1, ..., |J|$ y $j = |T_K| + 1, ..., |T_K| + |J|$, c_{ij} es el costo del viaje I sin recibir tarea adicional;
- **III.** Para $i = |J| + 1, ..., |T_K| + |J|$ y $j = 1, 2, ..., |T_K|$, C_{ij} coincide con un valor lo suficientemente alto como para que ninguna tarea quede sin asignar;
- **IV.** Para $i = |J| + 1, ..., |T_K| + |J|$ y $j = |T_K| + 1, ..., |T_K| + |J|$, $c_{ij} = 0$;

	Tareas	tareas ficticias
viajes	**I** $c_{ij} = f(i, j) + CTL$ o $c_{ij} = \infty$	**II** c_{ij} = costo de viaje I sin tarea adicional
viajes ficticio	**III** $c_{ij} = \infty$	**IV** $c_{ij} = 0$

Figura 1 - Estructura de la matriz de costos C.

Al comienzo de la aplicación del algoritmo, todos los viajes están vacíos y se le asignan tareas en cada iteración de acuerdo con la resolución del problema de asignación de capa actual. Al finalizar el procedimiento, se eliminan los días que quedan vacíos.

Para formar el Bloque I, una de las tres funciones siguientes se puede utilizar para $f(i, j)$:

I) $F_1(yo, j)$ = costo del viaje I en minutos pagados, considerando la inclusión de la tarea j en el viaje I;

ii) o, $F_{dos}(yo, j)$ = minutos de inactividad en el viaje I, considerando la inclusión de la tarea j en el viaje I;

iii) o, $F_3(yo, j)$ = minutos de inactividad en el viaje yo + minutos extra en el viaje $i * AdHE$ considerando la inclusión de la tarea j en el viaje I, Dónde $AdHE$ son las horas extra extras definidos por la legislación como 50%.

En resumen, el procedimiento para construir la solución inicial puede describirse mediante el algoritmo que se muestra en la Figura 2 a continuación.

Aporte: Conjunto de tareas;
Paso 1: Cuchillo $k = 1$;
Paso 2: formar la capa k que contiene las tareas que no se pueden realizar en la secuencia de tareas del mismo nivel o niveles posteriores;

Paso 3: Construya la matriz de costos C;
Paso 4: Resuelve el problema de asignación descrito por la matriz. C y asignar tareas a los viajes según el resultado obtenido.
Paso 5: Cuchillo $k = k + 1$ y vuelva al paso 2 hasta que se asignen todas las tareas.

Figura 2 - Procedimiento para construir la solución inicial.

2.2 Mejora de la solución

La fase de mejora consiste en la aplicación consecutiva de dos procedimientos denominados M1 y M2. En primer lugar, se aplica el procedimiento M1, que divide los viajes construidos en dos partes, formando los viajes parciales. Cada división (o corte) separa las tareas de un viaje formando dos viajes parciales, uno que contiene las tareas a la izquierda del corte y el otro contiene las tareas a la derecha del corte. Estos viajes parciales luego se recombinan resolviendo un problema de asignación. La figura 3 ilustra esta operación.

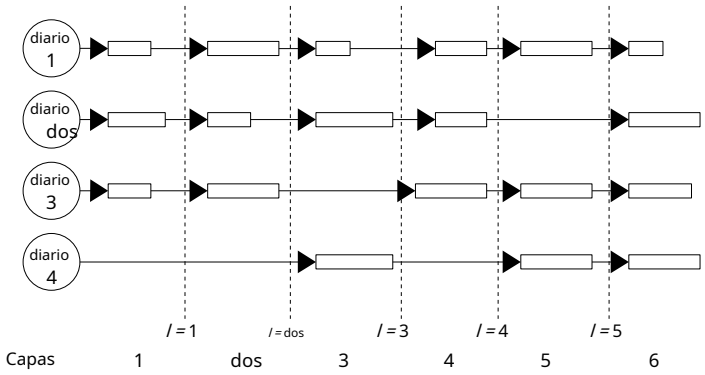

Figura 3 - *La solución obtenida tras la fase constructiva y sus posibles recortes. Los rectángulos representan las tareas y las flechas la secuencia de ellas en un viaje. los cortes posibles están representados por las líneas discontinuas.*

Una iteración del procedimiento M1 consiste en realizar K-1 cortes y resuelva un problema de designación para cada corte. Luego, los costos se calculan para recombinar los días parciales a la izquierda del corte con los días parciales a la derecha del corte. estos costos formar la matriz cuadrada $D = [d_{ij}]$ lo que representa un problema de designación. La resolución de este problema determina cómo se deben recombinar los viajes parciales. Figura 4 ejemplifica este procedimiento.

Ser NJE el número de días parciales a la izquierda del corte y NJD el número de viajes parciales a la derecha del corte. Como regla general, para $yo = 1, ..., NJE$ y $j = 1, ..., NJD$, d_{ij} será igual al costo en minutos pagados del viaje que combina los viajes I y j (si es posible combinarlos) + CTL, Dónde CTL toma un pequeño valor que penaliza el cambio de línea de viaje en el viaje si la línea de viaje de la última tarea del viaje I es diferente de la línea de la primera tarea del

viaje *j* y de lo contrario valor cero. Si no es posible combinar los viajes *I* y *j*, $d_{ij} = \infty$.

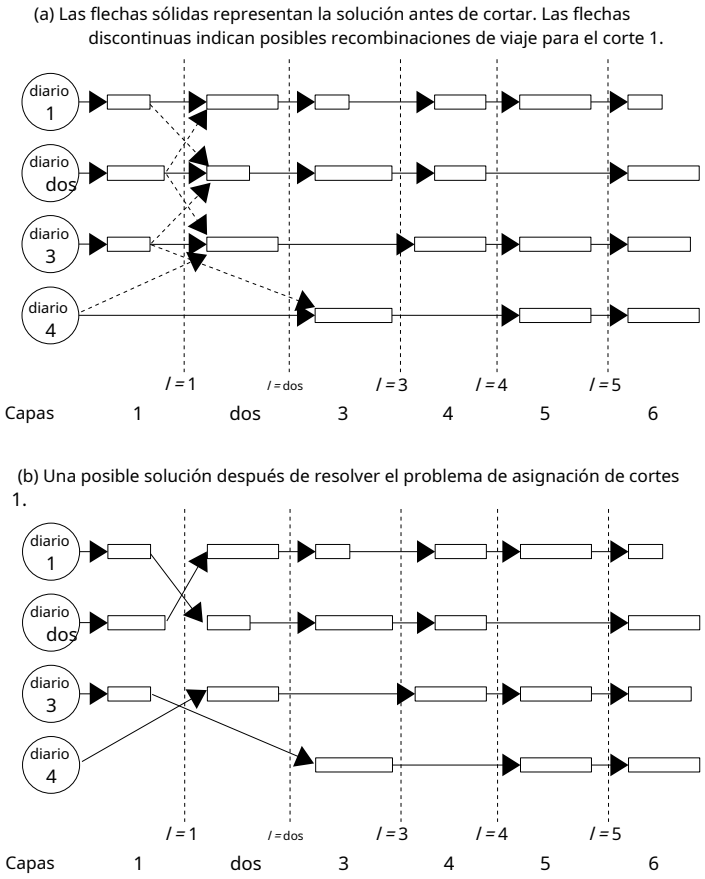

Figura 4 - Ejemplo de ejecución del procedimiento M1.

Puede suceder que *NJE* ser diferente de *NJD,* así que para hacer la matriz *D* Se agregarán viajes ficticios como se ilustra en la Figura 5. El bloque I sigue la definición original de $D_{ij,}$ mientras que el Bloque II se define como el caso:

- Caso 1. Si *NJE> NJD,* El bloque II se define como: para *yo* = 1, ..., *NJE,* ej = *NJD* + 1, ..., *NJE,* $de_{ij} =$ costo en minutos pagados del viaje *I;*
- Caso 2. Si *NJE <NJD,* el bloque II se define como: para i = *NJE* + 1, ..., *NJD,* y *j* = 1, ..., *NJD,* $d_{ij} =$ costo en minutos pagados del viaje *j;*

El procedimiento M2 hace un corte en la solución actual, que consiste en separar la última tarea de las horas con horas extras y solucionar el problema de asignación a este corte. La idea de este procedimiento es reducir el costo de la solución eliminando tareas de los días laborables con horas extra y reasignándolas en los días laborables que tienen tiempo de inactividad. Después del corte, los costos se calculan para recombinar los días parciales a la izquierda del corte con los días parciales a la derecha del corte.
Estos costos forman la matriz cuadrada $D = [d_{ij}]$ que representa un problema de designación y que tiene la misma estructura ya definida para el procedimiento M2. La Figura 6 ejemplifica esto procedimiento.

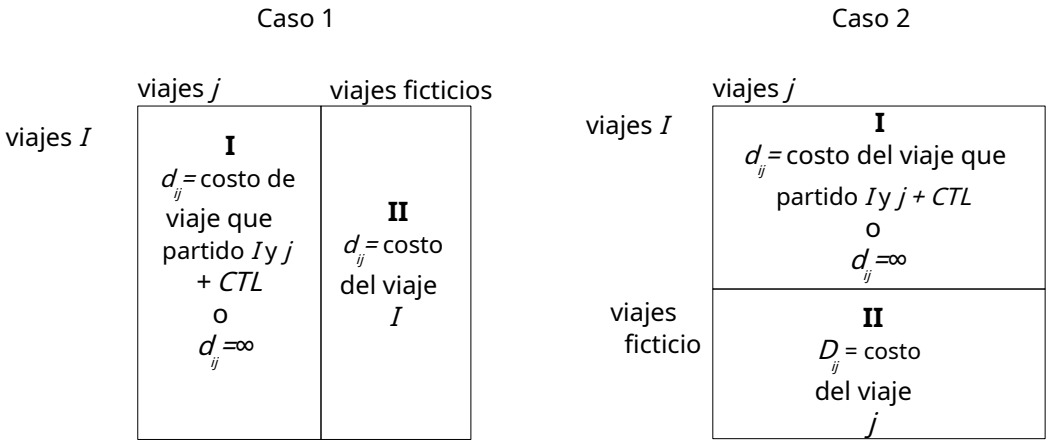

Figura 5 - Estructura de la matriz D.

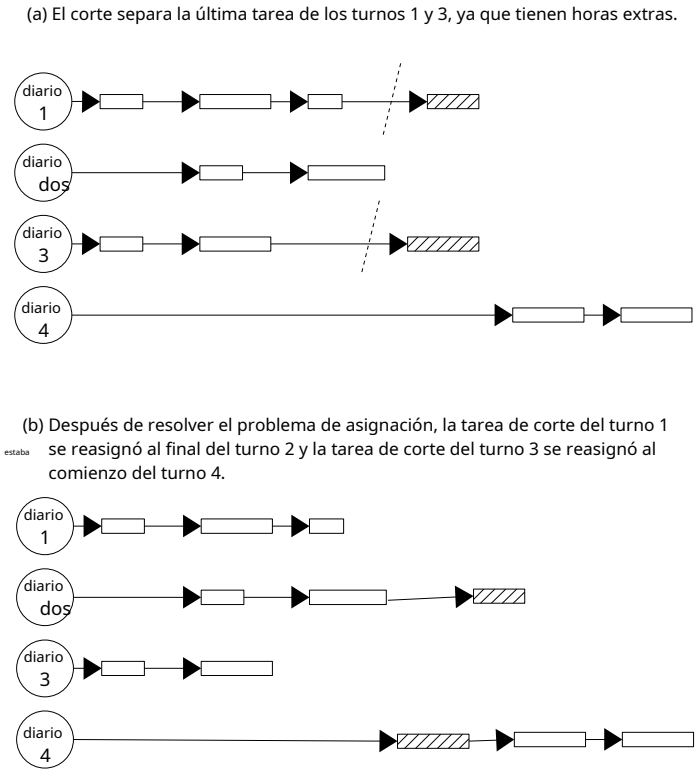

Figura 6 - Ejemplo de ejecución del procedimiento M2.

Los procedimientos M1 y M2 se realizan repetidamente hasta que no hay mejora en el costo de la solución por *NumIter* iteraciones. La Figura 7 presenta el esquema general de la fase de mejora.

3 Resultados y análisis de resultados

Para resolver el problema de designación se implementó el algoritmo propuesto por Carpaneto y Toth (1987), que combina el método húngaro y el método Shortest Augmenting Path. Para resolver los modelos PLI o PL, se utilizó el solucionador *lp-solver* por Markus Weidenauer (2003). Todas las pruebas de computadora se realizaron en un Pentium III

900 Mhz con 512 MB de RAM y funcionando con el sistema operativo Windows 2000.

```
Aporte: Solución inicial y NumInter;
Comienzo
   repetir
      repetir
         Ejecute M1;
      Hasta que no hay mejora en la solución para NumIter iteraciones.
      repetir
         Ejecute M2;
      Hasta que no hay mejora en la solución para NumIter iteraciones.
   Hasta que no hay mejora en la solución para NumIter iteraciones.
Final.
```

Figura 7 - Procedimiento para mejorar la solución

3.1 Características de las instancias de prueba

Dos de las instancias utilizadas en las pruebas son instancias reales y las otras son instancias aleatorias basadas en una instancia real. Por lo tanto, a pesar de las instancias generadas aleatoriamente, tienen características de instancias reales. Las instancias se describen a continuación, con la parte numérica del nombre que indica el número de tareas (*p.ej* CV412 contiene 412 Tareas):

- CV412: corresponde a datos de una empresa local que realiza transporte colectivo metropolitano entre ciudades;
- CC2313: corresponde a datos de una empresa local que realiza transporte urbano;
- CC130, CC251, CC512, CC761, CC1000, CC1253, CC1517 y CC2010: son instancias obtenidas de la instancia CC2313 por sorteo *bloques* de la misma.

3.2 Resultados obtenidos

La Tabla 1 presenta los resultados obtenidos por el algoritmo propuesto (AP) utilizando el construcción de la solución inicial las funciones F_1, F_{dos} y F_3, siendo eso para el parámetro *CTL* que penaliza los cambios de línea, se utilizó el valor 1 y para el parámetro *NumIter* que define el número de iteraciones sin mejora de la solución, se utilizó el valor 4. Para cada instancia se muestra el número de viajes (NJor) en la solución, el costo de la solución en minutos pagados y el tiempo de ejecución en el formato minutos: segundos. La columna LIP muestra el valor límite inferior para PET calculado según el modelo para escalonar personal en una ubicación fija (Bodin*es*allí , 1983) y Gap% es el porcentaje de la mejor solución AP (resaltada en negrita) para cada instancia en relación con LIP, es decir, *brecha = Costo-LABIO / LABIO*·100.

El tiempo de cálculo fue pequeño, siendo el más largo igual a 8min8s para el instancia CC2313 usando la función F_1. Los mejores resultados se obtuvieron utilizando Las funciones F_{dos} y F_3 que obtuvo el menor costo en 4 y 6 instancias respectivamente. Los porcentajes de brecha son aparentemente altos, sin embargo, el valor de LIP es muy bajo ya que que el problema del apartado 3.3 es una propuesta muy simplificada en relación con el PET.

3.3 Comparación entre procedimientos de mejora

La Tabla 2 muestra cuánto se puede reducir el costo de la solución inicial usando

individualmente cada uno de los procedimientos y con la combinación de ambos procedimientos de propuesta de mejora junto con la función F_3.

Tabla 1 - Resultados obtenidos por el algoritmo propuesto

	AP usando F_1		AP usando F_{dos}		AP usando F_3		
	costo de		costo de		costo de		
Ejemplo	NJor	Solución de tiempo	NJor	Solución de tiempo	NJor	Solución de tiempo	% De brecha de labios
CC130	19	8451.3 00:38	19	8472.5 01:09	19	**8389.4** 00:45	8057 4.13
CC251	40	17667.5 00:36	40	17690.0 00:46	40	**17600.0** 00:35	15655 12,42
CV412	69	30810.0 01:05	69	30795.0 00:36	66	**29512.5** 00:48	25427 16.07
CC512	80	35612.5 01:08	79	**35105.0** 00:40	80	35312.5 01:00	30858 13,76
CC761	109	48395.0 02:22	110	48632.5 01:47	107	**47532,9** 01:50	43010 10,52
CC1000	152	67090.0 01:39	147	65019.4 02:10	146	**64873.6** 05:00	57000 13,81
CC1253	191	84580.0 02:29	188	83261.1 02:40	187	**82842,9** 03:36	72261 14,64
CC1517	232	102729.5 02:02	225	**99852,8** 05:29	227	100507.0 02:45	89191 11,95
CC2010	297	131482.5 05:21	290	**128964.2** 07:00	292	129637.8 03:43	116019 11.16
CC2313	339	150649.7 08:08	331	**147215.0** 04:37	340	150522.5 04:04	131800 11.70

Tabla 2 - Comparación entre procedimientos de mejora

Ejemplo	inicial	M1	Reducción	M2	Reducción	M1 + M2	Reducción
CV412	33394,8	29512.5	11,63%	30897.3	7,48%	**29512.5**	11,63%
CC1000	74637.0	65442.0	12,32%	68550.5	8,15%	**64873.6**	13,08%
CC1253	95622.0	82955.4	13,25%	88486,9	7,46%	**82842,9**	13,36%
CC2313	172660.4	150635.0	12,76%	158088.0	8,44%	**150522.5**	12,82%

En la Tabla 2 también se observa que las soluciones obtenidas usando solo M1 fueron mejores que usando solo M2. Sin embargo, la combinación de los dos procedimientos generó mejores soluciones para las instancias CC1000, CC1253 y CC2313. La mayor ganancia al usar M1 y M2 juntos fue para la instancia CC1000 que logró una reducción adicional del 0,76% en comparación con la reducción lograda usando solo M1. Aunque esta ganancia es pequeña en porcentaje, conviene recordar que su valor económico puede ser significativo.

3.4 Resultados con el modelo de cobertura establecida

En esta sección, se presentan los resultados para PET modelado como un PCC. Para instancias más pequeñas, fue posible obtener el valor óptimo usando programación lineal entera, mientras que para instancias más grandes se obtuvieron límites inferiores usando relajación lineal o relajación lagrangeana dependiendo del tamaño de la instancia.

En la Tabla 3 se presentan los resultados obtenidos de la resolución PET utilizando el modelo PCC. La columna *MaxPed* muestra el número máximo de fracciones por columna, con un asterisco antes del número que indica que se han generado todas las columnas posibles. Los valores sin asterisco indican que la instancia podría tener columnas con más de *MaxPed* fracciones que no se generaron para limitar el tamaño del problema en un intento de hacer posible su resolución.

La Tabla 3 también muestra el número de Filas y Columnas obtenido al combinar las fracciones. La columna ZPL trae el valor del problema de programación lineal relajado (PL), ZPLG es el valor del límite inferior lagrangiano ((BEASLEY; 1987 y REEVES; 1993) y ZPLI es el costo de la solución completa obtenida al resolver el modelo PLI Para obtener la ZPL / ZPLI se adoptó un límite de tiempo igual a 24 horas. Para el cálculo de la ZPLG no hubo límite de tiempo. El valor de Gap% se obtiene mediante la fórmula $brecha = Costo\text{-}LABIO \ / \ LABIO \cdot 100$, donde *Costo* es el valor de ZPL para las instancias donde fue posible obtenerlo y para las demás es el valor de ZPLG.

Tabla 3 - Resultados obtenidos para el PCC

Ejemplo	Líneas MaxPed	columnas	ZPLI	ZPL	ZPLG	LABIO	brecha%
CC130	* 3 36	172	9530.60	9530.60	9528.90	8057	18.29
CC251	* 4 77	1483	18691.20	18691.20	18587.40	15655	19.39
CV412	* 4 136	6730	30455,00	30455,00	30228.10	25427	19,77
CC512	* 5 162	16875	37112.50	37112.50	36468.69	30858	20.27
CC761	* 5 234	47552	51752.50	51752.50	50242.08	43010	20,33
CC1000	* 6 315	176734	67971.50	67971.50	65821.81	57000	19.25
CC1253	* 6 398	314149	86349,00	86349,00	84052.23	72261	19,50
CC1517	* 6.480	579666	105178.50	105178.50	102315.24	89191	17,93
CC2010	4.623	1206504	138035.40	138035.40	133055.62	116019	1610242 18,98
CC2313	4.715	-		-	149648.34	131800	13.54

Se observa en la Tabla 3 que fue posible obtener el valor de ZPLI para casi todas las instancias, excepto por ejemplo CC2313 debido al alto número de columnas y al alto tiempo computacional sin obtener la solución (más de 100 horas de espera). Los valores de la solución completa (ZPLI) para estas instancias son excelentes, ya que son iguales al valor del costo de resolver el problema relajado (ZPL). Recordando que el valor obtenido para *ZPLG* es como máximo igual al valor obtenido por el modelo PL, por lo que *ZPLG* solo debe usarse si la resolución del modelo PL no es práctica

3.5 Comparación de resultados

Para una comparación directa de los resultados obtenidos por el algoritmo propuesto (AP) y por la resolución PET utilizando el modelo PCC, la Tabla 4 muestra el Desplazamiento porcentual relativo (APR) del costo de la mejor solución obtenida por AP tomado de la Tabla 1, en relación con los valores de ZPL (cuando estén disponibles) obtenidos por el PCC de acuerdo con las Tablas 3. Además, para todos los casos el valor obtenido por AP se compara con LIP. La TAE viene dada por la fórmula

$APR = Costo1\text{-}Costo2 \ / Costo2 \cdot 100$ donde *Costo1* es el costo de la solución AP y *Costo2* es el valor de ZPL o LIP, según la columna de la tabla.

La Tabla 4 muestra que los costos de las soluciones AP en todos los casos son menores que los costos de las soluciones PCC. Las soluciones de AP estuvieron por encima del límite inferior del problema (LIP) entre 4.13% y 16.07%.

Tabla 4 - Comparación de los resultados de AP con los resultados de PCC

Ejemplo	ZPL	LABIO
CC130	- 11,97%	4,13%
CC251	- 5,84%	12,42%
CV412	- 3,09%	16,07%
CC512	- 5,41%	13,76%
CC761	- 8,15%	10,52%
CC1000	- 4,56%	13,81%
CC1253	- 4,06%	14,64%
CC1517	- 5,06%	11,95%
CC2010	- 6,57%	11,16%
CC2313	- 1,63%	11,70%

4. Conclusiones

El diseño del algoritmo fue el resultado de una combinación de conocimientos de diseño y análisis de algoritmos, investigación operativa, programación matemática, gráficos y heurística. Sin esta integración, sería muy difícil lograr los resultados indicados. Se puede demostrar que este algoritmo tiene una complejidad asintótica O ($No_{4)}$, ser *No* el número de tareas. Por tanto, un algoritmo de complejidad polinomial.

El rendimiento computacional del algoritmo propuesto (AP) fue muy satisfactorio tanto en términos de la calidad de las soluciones obtenidas como del tiempo computacional. Superando los resultados obtenidos por el modelo clásico de PCC.

El algoritmo propuesto resuelve problemas tanto pequeños como grandes en un tiempo computacional muy reducido y con una calidad aceptable para instancias grandes, ya que sería difícil llegar a mejores soluciones para estos casos, utilizando programación matemática, en un tiempo computacional aceptable.

Una característica muy importante de este algoritmo es que se puede aplicar tanto para resolver problemas de programación estáticos (las tareas no cambian durante el día) como dinámicos (cuando las tareas se alteran por imprevistos).

Reconocimiento: los autores desean agradecer al CNPq por el apoyo parcial de acuerdo con el proceso 305534 / 2004-1.

5 referencias bibliográficas

BEASLEY, JE Un algoritmo para el problema de cobertura de conjuntos. *Revista europea de investigación operativa,* v. 31, no. 1, pág. 85-93, julio. 1987.

BEASLEY, J.E; CHU, PC Un algoritmo genético para el problema de cobertura de conjuntos.*Revista europea de investigación operativa,* v. 94, no. 2, pág. 392-404, oct. 1996.

BODIN, L .; DORADO, B .; ASSAD, A .; BALL, M .. Enrutamiento y programación de vehículos y tripulaciones - La estrella del arte. *Revista Internacional de Investigación en Computación y Operaciones,* v. 10, norte. 2, pág. 63-211, 1983.

CALVI, R. *Un algoritmo para el problema de programación de la tripulación en las compañías aéreas Autobús.* 2005. Disertación (Maestría en Ciencias de la Computación) - Universidad Estadual de Maringá.

CAPRARA, A .; FISCHETTI, M .; TOTH P. Un método heurístico para el problema de cobertura de conjuntos. *La investigación de operaciones,* v. 47, no. 5, pág. 730-743, septiembre / octubre 1999.

CARPANETO, G .; TOTH, P. Algoritmos primarios-duales para el problema de asignación.*Matemáticas aplicadas discretas,* norte. 18, pág. 137-153, Holanda Septentrional, 1987.

CASTRO, CE de; CONSTANTINE, AA; MENDONÇA NETO, CFX de; ARAUJO, SA de. Problema de escala del conductor: una metodología heurística para construir el problema de cobertura del conjunto. En: XXXVII SIMPOSIO BRASILEÑO DE INVESTIGACIÓN OPERACIONAL-SBPO,*Anales...* 2005, césped. . v. 1, págs. 1-5.

CONSTANTINE, AA; REIS, P .; MENDONÇA NETO, CFX de; FIGUEIREDO, MF Aplicación de algoritmos genéticos al problema de cobertura de conjuntos. En: XXXV

SBPO - SIMPOSIO BRASILEÑO DE INVESTIGACIÓN OPERACIONAL, 2003, Natal. Anales .... v. 1, pag. 1-10.

CORMEN, T. LEISERSON, C .; RIVEST, R .; STEIN, C.*Algoritmos: teoría y práctica.* Editorial Campus, 2002.

GOLDBARG, MC; LUNA, HPL Optimización combinatoria y programación lineal: modelos y algoritmos, 4.ed. Río de Janeiro: Campus, 2000.

ÉL ES UN. *Una heurística para todo el problema de corte de inventario unidimensional.* 2006. Propuesta de Disertación (Maestría en Ciencias de la Computación - Universidad Estadual de Maringá.

MAURI, GR; LORENA, LAN Método interactivo para resolver el problema de programación de horarios de la tripulación. En: SBPO, no. 36, 2004, São João Del Rei.*Anales...* São João Del Rei: UFMG. v.1, págs. 1-10.

MARTELOZZI, SR. *Optimización de la distribución / asignación de clases a los estudiantes: un estudio del caso.* 2004. Disertación (Maestría en Ciencias de la Computación) - Universidad Estadual de Maringá.

MAYERLE, SF *Un sistema de apoyo a la toma de decisiones para la planificación operativa de las empresas de transporte urbano de viajeros por carretera.* 1996. Tesis (Doctorado en Ingeniería de Producción) - Universidad Federal de Santa Catarina, Florianópolis.

OHLSSON, M .; PETERSON, C .; SÖDERBERG, B. Un enfoque de campo medio eficiente para el problema de cobertura de conjuntos.*Revista europea de investigación operativa,* v. 133, no. 3, pág. 583-595, sept. 2001.

REEVES, CR Técnicas heurísticas modernas para problemas combinatorios, Oxford: Blackwell Scientific Publications, 1993.

SHEN, Y .; KWAN, RSK*Búsqueda tabú para la programación del conductor,* Report 2000.10, Research Report Series, School of Computer Studies, Universidad de Leeds, 2000. Disponible en: <http://www.comp.leeds.ac.uk/research/pubs/reports/2000/2000_10.ps.gz> . Consultado en: 14 de junio. 2004.

SIQUEIRA, PH *Aplicación del algoritmo de emparejamiento al problema de construir escalas de conductores de autobuses y coleccionistas.* 1999. Disertación (Maestría en Ciencias) - Universidad Federal de Paraná.

NETTO, CA de S. *Problema de programación del personal en el área de servicio Teléfono.* 205. Trabajo de Conclusión del Curso (Informática) - Universidad Estadual de Maringá.

SOUZA, MJF; CARDOSO, LXT; SILVA, GP Programación de la tripulación del autobús urbano: un enfoque heurístico. En: SBPO, no. 35, 2003, Navidad.*Anales...* Natal: UFRN, 2003. p. 1285-1294.

VILELA JUNIOR, A. *Planificación y programación de la producción en una fundición automatizada de gran tamaño.* 2005. Propuesta de Disertación (Maestría en Ciencias de la Computación) - Universidad Estadual de Maringá.

WREN, A .; WREN, D. Un algoritmo genético para la programación de conductores de transporte público. *Computers Ops Res.,* v. 22, no. 1, pág. 101-110, 1995.