Rotative Workforce Scheduling Problem

Presentación Final

Jonathan Antognini C.
Luis Casanova S.

Universidad Técnica Federico Santa María

25 de julio de 2012

Introducción

Rotating Workforce Scheduling, es un problema que busca realizar una planificación de turnos de trabajo para un conjunto de empleados. Esta planificación debe ser cíclica, es decir, que al final de cada período de planificación se realiza una rotación entre los turnos asignados a cada empleado, esto sujeto a restricciones específicas.

Lu	Ma	Mi	Ju
A	A	D	-
D	D	N	N
-	-	A	N

Estructura de datos

Los datos se obtuvieron de los input (Example*.txt), archivos que contenían los datos necesarios para poder definir el problema. Los datos son:

- w: largo de la planificación. Todos los input tenían un $w = 7$, lo que corresponde a planificar de lunes a domingo.
- n: número de empleados.
- m: número de turnos más día libre. En la mayoría de los input m valía 4 (3 turnos + 1 día libre).
- A: vector de largo m donde se indican los diferentes tipos de turnos. En la mayoría de los casos v contenía $D, A, N, -$.

- R: matriz de requerimientos de turnos por día, ya que hay w días, y $m-1$ día son turnos, la matriz tiene dimensión $R_{(m-1)xw}$
- $MAXS$: Vector de largo m donde por cada turno se indica el máximo de turnos o días libres consecutivos permitidos.
- $MINS$: Vector de largo m donde por cada turno se indica el mínimo de turnos o días libres consecutivos permitidos.
- $MAXW$: número máximo de días consecutivos trabajados.
- $MINW$: número mínimo de días consecutivos trabajados.
- $C2$: matriz con secuencias de turnos no permitidas de largo 2.
- $C3$: matriz con secuencias de turnos no permitidas de largo 3.

Representación

Para implementar el algoritmo se crea el arreglo *solucion* largo $w \times n$.

0	1	2	3	4	5	6	...

Para una mejor comprensión se puede ver el arreglo como matriz, en donde las filas son los empleados y los dias las columnas, los valores al interior son a los índices correspondientes al arreglo *solucion*, y en el orden en que son instanciados.

0	6	12
1	7	13
2	8	14
3	9	15
4	10	16
5	11	17

MFC y GBJ

Funciones *MFC*:
Se verifican las restricciones,basta encontrar basta encontrar un camino factible, se eliminan los dominios futuros incompatibles.

Funciones restricción para el *GBJ*:
Se verifican las restricciones, en caso de no cumplirse se retorna 1 para realizar el salto al nodo más recientemente instanciado y conectado, este se encuentra en la lista padres de cada nodo.

MFC+GBJ

Item a grandes rasgos los pasos realizados son:

1. Se guardan los padres de cada nodo, en una lista.
2. Realiza la asignación del dominio.
3. Realiza función *MFC* para filtrar dominios.
4. Realiza función *consistente* que realiza los chequeos de restricciones para el GBJ.
5. Analiza si debe realizar un salto, si es así es al mayor valor correspondiente a la lista padre del nodo actual.
6. De ser así realiza el salto y se reestablecen los dominios y la solución.
7. Si no se vuelve al punto 2, hasta llegar a la solución final.

Greedy + HC: Representación

Considerando $w = 4$ y $n = 3$:

Lu	Ma	Mi	Ju
A	A	D	-
D	D	N	N
-	-	A	N

Para calcular efecto de comprobación de restricciones se representó de la siguiente forma:

Lu	Ma	Mi	Ju	Lu	Ma	Mi	Ju	Lu	Ma	Mi	Ju
A	A	D	-	D	D	N	N	-	-	A	N

Greedy

Para poder definir un algoritmo Greedy correctamente es necesario especificar:

- Función de evaluación: esta función es la misma definida en la sección anterior.
- Punto de partida: el día en donde se quiera empezar a planificar. Es decir se le entregará un día entre 1 y W.
- Función miope: para el día i, se le asigna al primer trabajador disponible el turno requerido, de tal forma que la diferencia entre la cantidad de empleados necesarios en dicho turno se minimize.

Hill Climbing

- Número de restart: definido como constante (se cambiaba por cada instancia).
- Solución inicial: solución obtenida mediante greedy.
- Función objetivo: la función objetivo corresponde a minimizar la cantidad de penalizaciones hechas debido a restricciones blandas insatisfechas.

- Movimiento: swaps de turnos entre turnos de un día. Por ejemplo:

Lu	Ma	Mi	Ju
A	A	D	-
D	D	N	N
-	-	A	N

Al aplicar el movimiento y generar el primer vecino, se hace un swap de la casilla 1,1, con la casilla 2,1, quedando:

Lu	Ma	Mi	Ju
D	A	D	-
A	D	N	N
-	-	A	N

Greedy + HC

- Se inicializa una solución vacía.
- Se le pasas esa solución vacía al greedy, y además un día de comienzo. El resultado de esta operación genera una solución que respeta las restricciones duras (R).
- La solución del greedy es la entrada ahora para el hill climbing.
- Se realiza el movimiento sobre la solución actual.
- Si el algoritmo no encuentra un mejor vecino, entonces se hace un restart.
- Cuando se termina la cantidad de restart, el algoritmo acaba y entrega la mejor solución encontrada.

Resultados

Los resultados encontrados mediante greedy+hc fueron:

Instancia	Valor función objetivo
Example1.txt	35
Example2.txt	52
Example3.txt	79
Example4.txt	32
Example5.txt	46
Example6.txt	22
Example7.txt	289
Example8.txt	71

Resultados

Mediante MFC+GBJ no se encontraron resultados.

Esto debido a la gran complejidad del algoritmo y que además cuenta con 2 técnicas, esto complico la realización del debugueo.

Al tomar todas las restricciones como duras se reduce de manera considerable la cantidad de soluciones finales aceptadas, por lo que complica aún más el encontrar soluciones.

Las instancias de los problemas son bastante grandes, se realizan instancias más pequeñas, pero no se puede comprobar fácilmente que estas tengan solución.

Conclusiones

- Aprendizaje
- Técnica completa
- Técnica incompleta
- Generales