## Información de entrada

Nomenclatura

|  |  |
| --- | --- |
| Índices | |
|  | Índice conjunto de productos |
|  | Índice conjunto de periodos de planeación |
|  | Índice conjunto de recursos |
|  | Índice conjunto de strokes |
| Parámetros | |
|  | Número de unidades de que generan un stroke |
|  | Número de unidades de que el stroke consume |
|  | Lead time del stroke |
|  | Costo de almacenar una unidad de producto |
|  | Costo de setup del stroke |
|  | Costo de operación del stroke |
|  | Demanda del product para el period |
|  | Capacidad del recurso requerida para ejecutar una unidad del stroke (en unidades de tiempo) |
|  | Capacidad requerida del recurso para el setup del stroke (en unidades de tiempo) |
|  | Capacidad disponible del resource en el periodo (en unidades de tiempo) |
|  | Inventario inicial del producto |
|  | Recepciones programadas en unidades del producto a recibir en el periodo |
|  | Cantidad de orden fija a aplicar al producto |
| Variables | |
|  | = 1 si el stroke es ejecutado en el periodo (0 de lo contrario) |
|  | Nivel de stock del producto a la mano al final del periodo |
|  | Cantidad de strokes a ser ejecutados en el periodo |

Para que nuestra heurística funcione, es necesario contar con cierta información inicial. Sean:

Relacionados con los strokes

|  |  |
| --- | --- |
|  | Matriz de inputs. Es decir, el número de unidades del producto que requiere el stroke para ser ejecutado. |
|  | Matriz de outputs. Es decir, el número de unidades de producto que genera el stroke luego de ser ejecutado. En nuestro problema representamos un sistema en el que cada stroke puede generar sólo un producto, de no ser así estaríamos hablando de coproducción. |
|  | Vector de lead-time del stroke . |
|  | Vector de costo de preparar el stroke . |
|  | Vector de costo de operación del stroke . |

Relacionados con los productos

|  |  |
| --- | --- |
|  | Matriz de proyección de la demanda. Unidades del producto requeridas en el tiempo . |
|  | Matriz de recepciones programadas. Unidades del producto que llegarán en el tiempo . |
|  | Vector de inventario inicial del producto . |
|  | Horizonte de planeación, medido en unidades de tiempo. |
|  | Vector de cantidad fija a ordenar del producto . |
|  | Vector de costo de almacenar una unidad del producto en inventario. |
|  | Vector de costo por falta en inventario de una unidad del producto . |

## Modelo matemático

|  |  |  |
| --- | --- | --- |
|  | | **(1****)** |
| S.a. | |  |
|  |  | **(2)** |
|  |  | **(3)** |
|  | | (4) |

## Descripción de la heurística

Sea la matriz que representa las relaciones de input y output entre strokes, y que está determinada por

|  |  |
| --- | --- |
|  | () |

Sea también el grafo que muestra las relaciones de input y output strokes, construida a partir de con la propiedad adicional de modelar de manera clara la relación AND u OR entre un conjunto de strokes y el producto que generan (Línea 2, Algoritmo 1).

La heurística resuelve el problema de determinar cuáles productos serán generados a partir de cuáles strokes y en qué proporciones. Para esto es necesario conocer de cuántas formas puede ser generado el mismo producto y estimar cuál es el costo acumulado de cada opción (Líneas 4 y 5, Algoritmo 1), la forma en que se realiza este cálculo se explicará más adelante. Además se requiere ordenar la lista de productos en caso de que si algún producto contiene entre sus ‘hijos’ a otro producto en el mismo nivel, se ejecute este primero (Línea 7, Algoritmo 1). Para definir de qué forma se atenderán los requerimientos de los productos – lo que determinaría la forma en que se distribuirán las llamadas a los strokes – se calcula el MRP de cada producto en la lista utilizando alguna regla de lotificación que se puede definir como dato de entrada del sistema (Línea 10, Algoritmo 1).

Luego de determinar el mejor stroke para generar un producto en el periodo y haber decidido el momento en el que será ejecutado y cuántas veces, se agrega a la lista de productos los requerimientos del stroke elegido y se iguala a cero el requerimiento actual para hacerle entender al programa que se ha cumplido con el requerimiento de ese producto en el periodo (Líneas 12,13 y 14, Algoritmo 1). A continuación se muestra en pseudocódigo el algoritmo que rige la heurística.

|  |  |
| --- | --- |
| Algoritmo 1. Visión global de la heurística | |
| 1 | HeuristicaSimple() |
| 2 | = producto matricial entre y |
| 3 | Sea el grafo que representa las relaciones de input y output entre productos y strokes |
| 4 | //estimación del costo acumulado de operación |
| 5 | //estimación del costo acumulado de preparación |
| 6 | function Recursive (listaProductos, strokeMatrix) |
| 7 | ordenarListadeProductos(listaProductos) |
| 8 | For(int i←0;i<listaProductos.size();i++) |
| 9 | For(int t←0;t<longitudPeriodo;t++) |
| 10 | listaProductos(i).mps ← obtenerMPSde() |
| 11 | If(listaProductos(i).mps(t) > 0) Then |
| 12 | stroke ← seleccionarStroke(i,t, listaProductos(i).mps(t), ) |
| 13 | strokeMatrix(stroke.key,stroke.momentoEjecutar) ← strokeMatrix(stroke.key,stroke.momentoEjecutar)+ stroke.vecesQueEjecuta |
| 14 | listaProductos(i).mps(t) ← 0 |
| 15 | End If |
| 16 | End For |
| 17 | For(int i←0;i<listaProductos.size();i++) |
| 18 | If(listaProductos(i).noTieneRequerimientos() and listaProductos.size() > 1 ) Then |
| 19 | listaProductos.remover(i) |
| 20 | i←-1; |
| 21 | End If |
| 22 | Elsif (listaProductos.(i).noTieneRequerimientos() and listaProductos.size()=1) |
| 23 | return strokeMatrix |
| 24 | End Elsif |
| 25 | End For |
| 26 | return Recursive(listaProducto, strokeMatrix) |
| 27 | End For |
| 28 | return strokeMatrix |
| 29 | End Recursive |
| 30 | End HeuristicaSimple |

Como se había mencionado, existen dos funciones en el Algoritmo 1 que merecen atención: accumOC() y accumSC() las cuales realizan la estimación de los costos acumulados de Operación y Setup, respectivamente, de cada opción disponible, refiriéndonos por opción a los caminos que posiblemente se puedan encontrar hacia abajo en el grafo de tomar un stroke de los disponibles. El algoritmo inicia con los strokes que generan un producto final y de ahí hacia abajo a sus componentes. De igual forma se calculan los dos costos: Setup y Operación, como se observa en la línea siguiente se ilustra el algoritmo, que es recursivo:

|  |  |  |
| --- | --- | --- |
|  |  | () |

Para el conjunto de ‘hijos’ en condición AND del stroke , el conjunto de los ‘hijos’ en condición de OR y .

En la línea 10 de Algoritmo 1 se llama una función, obtenerMPSde, que calcula el MPS de cada componente, los requerimientos que se le pasan como parámetros a la función pueden provenir de la demanda o de la explosión de su correspondiente stroke ‘padre’. Aparte de esto es una función típica de cálculo de MPS dentro de la que el usuario tiene la posibilidad de embeber distintas reglas de lotificación. Este MPS se calcula para cada producto, de forma independiente a los strokes.

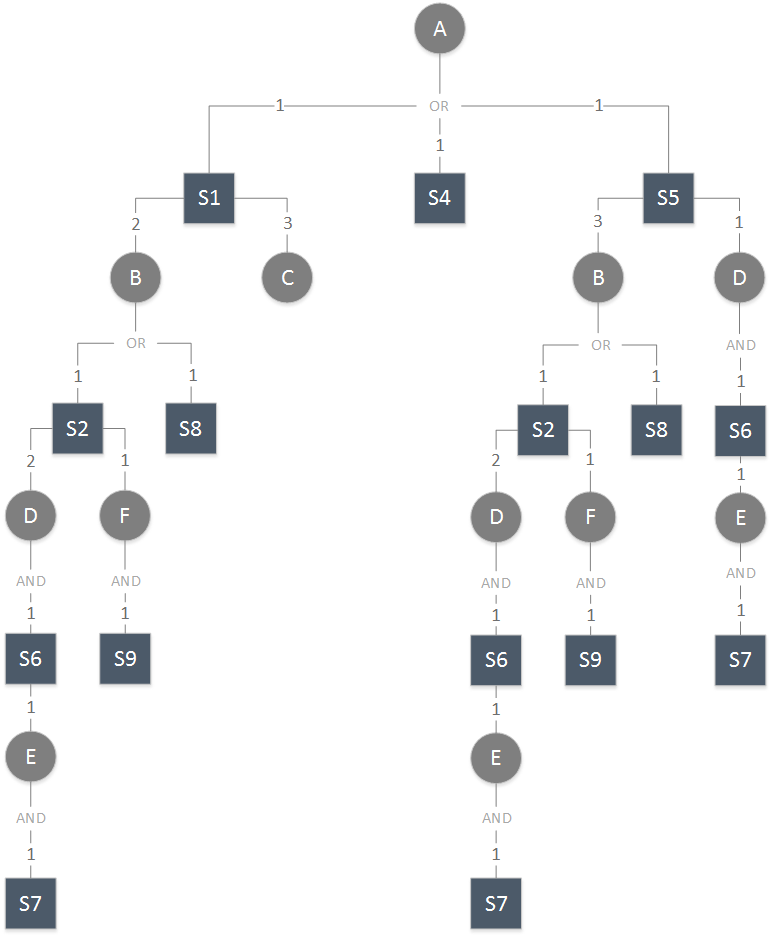
***Seleccionar Stroke:***

El mejor stroke se selecciona dentro del conjunto de strokes relacionados con el producto i tal que el lead time de estos strokes sea menor que el tiempo actual t. A este conjunto que se ha llamado C, se le calculan los costos totales como la suma de accumSC() y accumOC() multiplicado por el requerimiento de i en t. El pseudocódigo para el proceso de seleccionar el mejor stroke se muestra en Algoritmo 2.

|  |  |
| --- | --- |
| Algoritmo 2. Seleccionar mejor Stroke | |
| 1 | seleccionarStroke(i,t, listaProductos(i).mps(t), ) |
| 2 |  |
| 3 | For (int k; k < numeroStrokes; k←k+1) |
| 4 | Conjunto C |
| 5 | c ← Iterador(C) |
| 6 | If(k ↔ i) Then |
| 7 | If( > 0) Then |
| 8 | C.agregar (k) |
| 9 |  |
| 10 | c.siguiente() |
| 11 | End If |
| 12 | End If |
| 13 | End For |
| 14 | minCost ← argMin() |
| 15 | stroke := c\* ∈ C : = min |
| 16 | return stroke |
| 17 | End seleccionarStroke |

## Ejemplo de generación de un grafo:

La finalidad del grafo es permitir tener una visión general de la estructura de los productos y es una herramienta fundamental que permite el cálculo de los costos acumulados de cada stroke, por las funciones accumOC() y accumSC().



Grafo . Estructura del producto

A continuación se presenta un ejemplo de 6 productos () y que son elaborados con 9 strokes () y las siguientes matrices y :

|  |  |
| --- | --- |
|  | () |
|  | () |
|  | () |

Para facilitar la lectura se han enumerado los productos con letras de la ‘A’ a la ‘Z’ y a los strokes se les ha colocado la letra ‘S’ al frente.

El resultado se puede observar en Grafo 1, basado en la matriz se elimina la interacción de stroke-producto con las que se presentan las matrices y para ofrecernos una visión global de los requerimientos stroke-stroke. Datos importantes como el ‘número de opciones’ o formas de generar o llevar a cabo por producto (o por stroke en este caso) se obtiene de la matriz así:

|  |  |
| --- | --- |
|  | () |

Los arcos del grafo guardan información sobre cuántas ejecuciones de un stroke ‘hijo’ se requieren para habilitar un stroke ‘padre’. Por esto en , son los strokes y son las relaciones de requerimiento entre un stroke y sus ‘hijos’ dadas en .

En la matriz se leen los ‘padres’ como las columnas y los ‘hijos’ serán las filas, así, el stroke S2 requiere 2 *ejecuciones* de S6 y 1 de S9.

### Cálculo de los costos acumulados a partir de Grafo 1

Supóngase una situación en la que se requiera hallar el costo de Setup acumulado de S2 dada la siguiente información inicial:

|  |  |  |  |
| --- | --- | --- | --- |
| Stroke |  |  |  |
| k=1, S1 | 1 | 2000 | 2 |
| k=2, S2 | 2 | 1800 | 0.5 |
| k=2, S3 | 1 | 1500 | 0.5 |
| k=2, S4 | 5 | 18000 | 0.5 |
| k=2, S5 | 1 | 7300 | 2.0 |
| k=2, S6 | 2 | 4000 | 1.0 |
| k=2, S7 | 1 | 1800 | 0.5 |
| k=2, S8 | 3 | 3000 | 0.5 |
| k=2, S9 | 1 | 2500 | 0.5 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Producto |  | | | | | | | | | |  |  |
| t=1 | t=2 | t=3 | t=4 | t=5 | t=6 | t=7 | t=8 | t=9 | t=10 |
| i=1, A | 0 | 0 | 0 | 0 | 800 | 300 | 300 | 300 | 100 | 200 | 5000 | 200 |
| i=2, B | 0 | 0 | 100 | 0 | 0 | 0 | 230 | 100 | 347 | 900 | 5000 | 150 |
| i=3, C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5000 | 30 |
| i=4, D | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5000 | 20 |
| i=5, E | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5000 | 90 |
| i=6, F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5000 | 10 |

El conjunto de los ‘hijos’ de S2 en condición de AND y OR:

## Resultados obtenidos

Grafo 2. Producto B

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Calidad de la solución(COP) | | | | | |
| LxL | FOQ | LxL y FOQ | FOQ y LxL | GMOP | Wagner Whitin |
| Costos | $538.013,00 | $13.486.980,00 | $3.107.237,50 | $2.750.665,00 | $533.273,5 | $564.013,00 |
| % | 1% | 2429% | 483% | 416% | - | 6% |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Tiempos de ejecución (nanosegundos) | | | | | |
|  | LxL | FOQ | LxL y FOQ | FOQ y LxL | GMOP | Wagner Whitin |
| Costos | 17745495 | 18618336 | 20926811 | 17535944 | 228000000 | 23123669 |
| % | 92,22% | 91,83% | 90,82% | 92,31% | - | 89,86% |