Buque de carga - Greedy

Algotimos y demostraciones

Maximizar número de contenedores

- **Candidatos**: contenedores no escogidos todavía.
- **Seleccionados**: contenedores escogidos hasta el momento.
- **Función objetivo**: maximizar el número de contenedores (maximizar cardinalidad del conjunto de seleccionados).
- **Función selección**: escoger, de entre los candidatos, el de menor peso, p_i
- **Función de factibilidad**: al incluir el contenedor c_i de peso $p_{i,}$ la suma de los pesos de los contenedores seleccionados no será mayor que la capacidad del barco.
- **Función solución**: no se puede incluir ningún contenedor de los candidatos sin que se exceda la capacidad del barco.

Candidato *c*

c.peso == min(candidatos)
peso_actual + c.peso <= k

Seleccionado *s*

Pseudo-código

```
s[] = null;
peso_superado=false;
sort(c);        //ordenamos candidatos según su peso tal que c[i].peso <= c[i+1].peso
while( !c.empty() && peso_superado==false){
        x=c[0]; //Función selección: escogemos el contenedor de menor peso
        c.pop_front(); //Eliminamos el contenedor que acabamos de escoger de la lista de candidatos.
        if (peso_actual + x.peso <= k) //Función factibilidad
        {
                s.push_back(x); //Función solución
                peso_actual += x.peso;
        }else
                peso_superado=true;
}
return s;
```

Explicación

1) Ordenamos los contenedores por orden no decreciente de pesos.

2) Vamos escogiendo el primer contenedor (el menos pesado) y probamos si se puede incluir en el buque; si es así se añade a la lista de seleccionados y repetimos la operación.

3) Sino el algoritmo finaliza pues los siguientes contenedores tendrán mayor o igual peso y no podrán caber.

Optimalidad

$S = c_1 , c_2 ,..., c_n \}$ → solución greedy según un orden de peso no decreciente.

S → *supongamos* solución no óptima.

S' → solución óptima que más elementos coincidentes tiene con S. → Son los primeros k elementos.

Sabemos que los primeros k componentes en S' están en orden no decreciente.

Por ello: $c_{k+1} \neq c'_{k+1}$ *y* $p_{k+1} \leqslant p'_{k+1}$.

c_{k+1} no existe en S' → si sustituimos c'_{k+1} por c_{k+1} seguiríamos obteniendo una solución óptima; no excedemos el límite de peso.

c_{k+1} existe en S' → si cambiamos de posición c'_{k+1} por c_{k+1} seguiríamos obteniendo una solución óptima; no excedemos el límite de peso.

En ambos casos S' pasa a tener los primeros $k+1$ elementos iguales a S → ***contradicción***.

S, la solución greedy, es siempre una solución óptima.

Maximizar beneficio

- **Candidatos**: contenedores no escogidos todavía.
- **Seleccionados**: contenedores escogidos hasta el momento.
- **Función objetivo**: maximizar el beneficio obtenido.
- **Función selección**: escoger, de entre los candidatos, el que mayor razón de beneficio/peso tiene, c_i.
- **Función de factibilidad**: al incluir el contenedor c_i de peso p_i , la suma de los pesos de los contenedores seleccionados no sea mayor que *k*, la capacidad del barco.
- **Función solución**: no se puede incluir ningún contenedor sin que se exceda *k.*

Candidato *c*

(c.beneficio/c.peso) == máx(candidatos)
peso_actual + c.peso <= k

Seleccionado *s*

Pseudo-código

```
Contenedores C[N] ; // Aquí se encuentran todos los Contenedores con sus Pesos y beneficios
Ordenar(C) ; // Ordenamos los contenedores siendo el ultimo el de mayor relación Beneficio/peso
n=N-1 //Función selección
While lleno is False do //Función factibilidad
        If (PesoCargado + C[n].peso) <= CapacidadBuque do
                Agregar(ListaCargamento, C[n]) ;
                PesoCargado += Peso C[n];
                Eliminar(C,n);
                if PesoCargado == CapacidadBuque do //Función objetivo
                        lleno = true; // en este caso se alcanzaría el máximo beneficio
                done
        done
        n-- ; //Función selección
        if n <=0 do
                lleno = true ; // no hemos podido cargar más el Buque y queda espacio libre
        done
done
```

Explicación

1) Ordenamos los contenedores de forma no decreciente según su cociente beneficio/peso.

2) Seleccionamos el último contenedor (mayor cociente) y vemos si, añadiéndolo al barco, sobrepasamos la carga máxima o no. Si no la sobrepasa lo añadimos a seleccionados; si la sobrepasa continuamos con el siguiente.

3) Este proceso acaba al igualar el peso de todos los contenedores a k o cuando se acaban los contenedores candidatos.

No optimalidad - Ejemplo

Supongamos que en el buque podemos incluir contenedores hasta llegar a un peso k=10. Y tenemos los siguientes contenedores, con sus respectivos pesos, sus respectivos beneficios y sus cocientes beneficio/peso

C = {1, 2, 3] P = {6, 5, 5} B = {12, 7, 6} G = {2, 1.4, 1.2}

Ordenando los contenedores según su cociente peso/beneficio de forma no creciente y representados de la

forma: identificador (beneficio/peso) nos quedaría así: C = {1(2), 2(1.4), 3(1.2)}

El algoritmo greedy escogería el primero, con cociente beneficio/peso = 2, de forma que no podría añadir más contenedores pues es de peso 6 y cualquiera de los otros dos al ser de peso 5 superarían k=10. Así el beneficio obtenido con el algoritmo greedy en este caso sería el beneficio del único contenedor que incluiríamos cuyo valor es 12.

La solución óptima al problema sería elegir los otros 2 contenedores, que igualan la capacidad máxima del buque, cuya suma de sus beneficios sería 13, superando el resultado que nos daba el algoritmo greedy.

Solución óptima

Llenar la carga del barco al completo siempre es una solución óptima. Consideramos elementos dispuestos en orden no decreciente según su cociente beneficio/peso.

S = {c 1 , c 2 ,..., c n } → solución greedy. Suma total del peso == k. → supongamos que no es óptima.

S' → solución óptima (máx beneficio). Sus primeros k elementos coinciden con los k primeros de *S*.

Por todo lo anterior, sabemos que $c_{k+1} \neq c'_{k+1}$ y que $b_{k+1} / p_{k+1} \geq b'_{k+1} / p'_{k+1}$.

Todos los elementos a partir de k tendrán mayor densidad en S que en S' a no ser que no se haya podido escoger alguno en la solución greedy debido a que su peso sobrepasaría la carga máxima.

Aunque haya introducido contenedores de menor densidad, a partir del valor k, la media ponderada de la densidad beneficio/peso por peso será mayor en la solución greedy que en cualquier otra solución pues los contenedores que ha tenido que meter de menor densidad serán de menor peso e influirán menos al cálculo de la media ponderada.

media ponderada = $\sum$ k (densidad $*$ peso) $/$ peso total → Media_ponderada_greedy >= Media_ponderada_s'

Peso total_greedy ≥ Peso total _ s

benefecio total = media ponderada $*$ peso total → beneficio total _ greedy ≥ beneficio total_s′

Cualquier transformación de n bloques de la solución greedy por otros m bloques, estos m bloques supondrán, como mucho, tanto beneficio como suponían los n bloques a sustituir. Llegamos a la contradicción de la existencia de una S' distinta de S con más beneficio quedando demostrada que la solución greedy es óptima siempre que llena el barco.

$$\frac{beneficio\,total_{greedy}}{beneficio\,total_{s'}} = \frac{\frac{media\,ponderada_{greedy}}{media\,ponderada_{s'}} * peso\,total_{greedy}}{peso\,total_{s'}}$$