Memoria de divide y vencerás

Rubén Morales Pérez Francisco Javier Morales Piqueras
Bruno Santindrian Manzanedo Ignacio de Loyola Barragan Lozano
Francisco Leopoldo Gallego Salido

30 de abril de 2016

Índice

1. Ejercicio 2

1.1. Enunciado del problema

Minimizando el número de visitas al proveedor:

Un granjero necesita disponer siempre de un determinado fertilizante. La cantidad máxima que puede almacenar la consume en r días, y antes de que eso ocurra necesita acudir a una tienda del pueblo para abastecerse. El problema es que dicha tienda tiene un horario de apertura muy irregular (solo abre determinados días). El granjero conoce los días en que abre la tienda, y desea minimizar el número de desplazamientos al pueblo para abastecerse.

- Diseñar un algoritmo greedy que determine en qué días debe acudir al pueblo a comprar fertilizante durante un periodo de tiempo determinado (por ejemplo durante el siguiente mes).

- Demostrar que el algoritmo encuentra siempre la solución óptima.

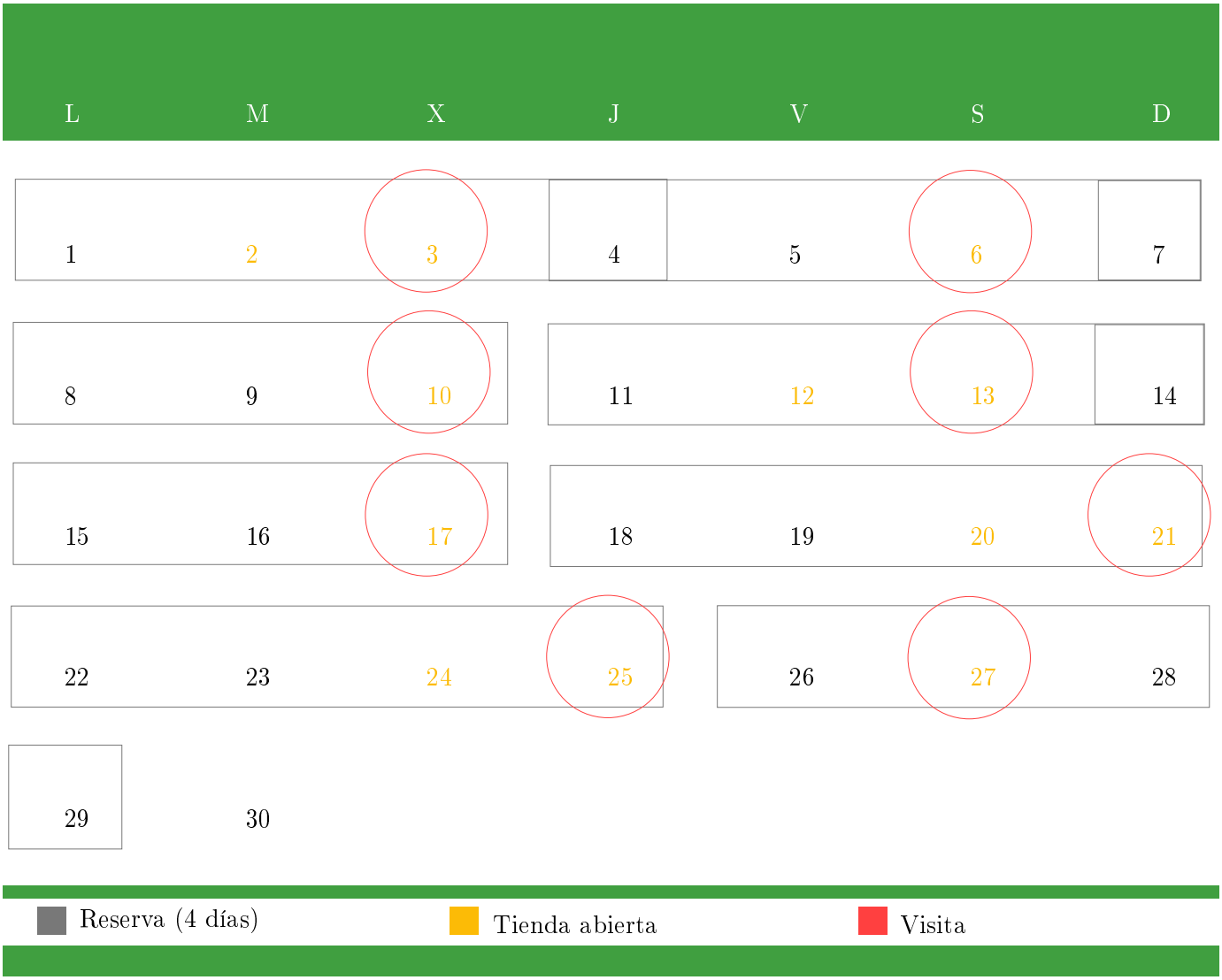
1.2. Diseño del algoritmo

El algoritmo, al basarse en la filosofía greedy, intenta escoger la mejor solución en cada fase. En nuestro caso, la mejor solución pasa por ir a comprar el día más lejano posible que cumpla estas dos condiciones.

1. Que diste menos de r días de la ultima reposición.

2. Que la tienda esté abierta.

Ejemplo:



1.2.1. Representación del algoritmo

Para programar el problema lo hemos modelado de la siguiente manera:

- Un vector de booleanos representa los días que abre la tienda.
  Por ejemplo [1, 0, 0, 1, 0, 1] significa que la tienda abre los días 1, 4 y 6.

- Una lista de enteros almacena los días que el granjero tiene que ir a comprar.

1.2.2. Implementación del algoritmo

La implementación se encuentra en *minimizar_visitas.cpp*.
La función clave es **minimize_visits()**, en la que, para buscar el día mas lejano posible, se accede

a la posición i+r del vector y se recorre hacia atrás hasta que se encuentra un día en el que la tienda esté abierta.

También hemos implementado otro algoritmo con el fin de hacer comparaciones, en el que el día de visita se elige aleatoriamente en el rango [i, i+r-1]. Esta implementación se encuentra en *visitas_ aleatorias.cpp*

1.3. Demostración de la optimalidad

Sean:

- **p** la duración del periodo (ej: 30 días)
- **r** la duración de la reserva
- **a** vector de tamaño p que representa los días que abre la tienda
  La tienda abre el día n si $a_n = 1$
  La tienda no abre el día n si $a_n = 0$
- **d** el vector que almacena los días de visita
  $d_{i+1} = d_i + k_i \ / \ k_i \in \{x \in \mathbb{N} : 0 < x < r \ y \ a_{d_i+x} = 1\}$

Formulamos dos ecuaciones:

$d^h_{i+1} = d^h_i + h_i$ donde $h_i = max\{x \in \mathbb{N} : 0 < x < r \ y \ a_{d_i+x} = 1\}$
Podemos definir $n \in \mathbb{N} \ / \ d^h_n = min\{d^h_i : d^h_i + r > p\}$ como el índice del último día que necesitamos ir a comprar.

$d^t_{i+1} = d^t_i + t_i$ donde $t_i \in \{x \in \mathbb{N} : 0 < x < r \ y \ a_{d_i+x} = 1\} \ / \ t_i < h_i$ para algún $i < n$

d^h representa la aproximación greedy, mientras que d^t representa cualquier otro algoritmo.

Por las definiciones anteriores sabemos que $\exists m \in \mathbb{N} \ / \ d^t_i < d^h_i \ \forall i, \ m < i < n \Rightarrow d^t_n < d^h_n$
Distinguimos dos casos:

1. Si $p - r < d^t_n < d^h_n$ las dos opciones son igual de buenas
2. Si $d^t_n < p - r < d^h_n$ la opción greedy es mejor

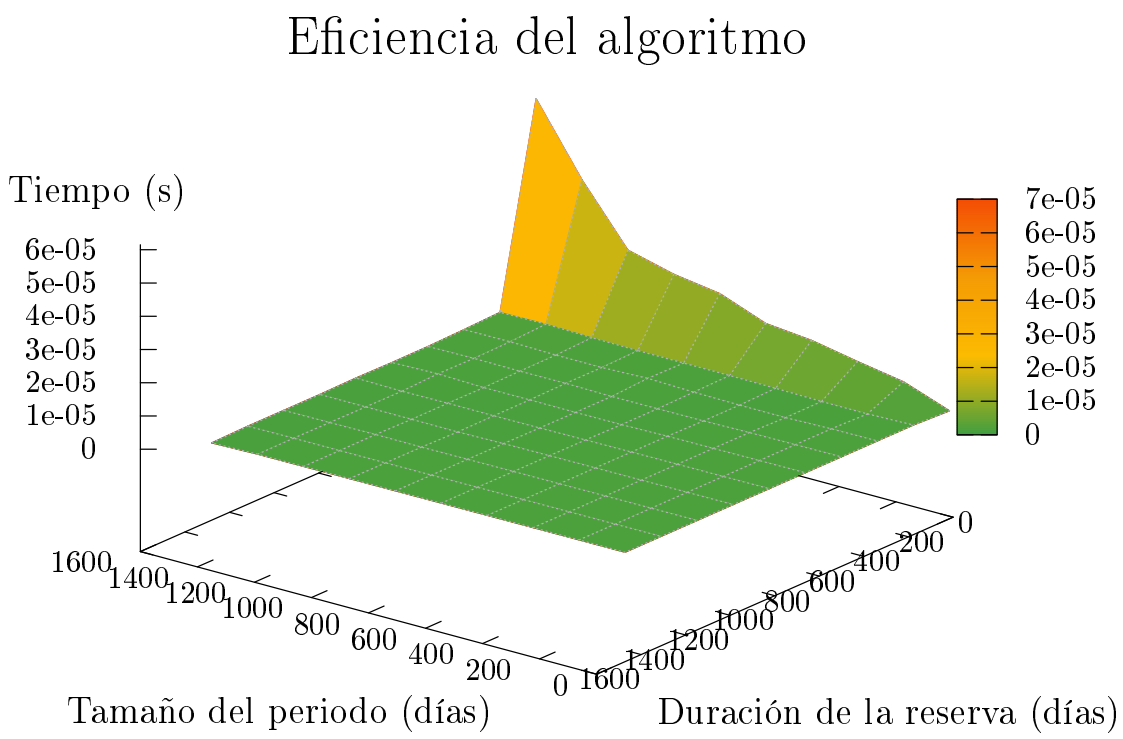
De aquí deducimos que utilizar el enfoque greedy en este caso siempre nos da el mejor resultado.

1.4. Resultados empíricos

1.4.1. Eficiencia

Aunque en esta ocasión la eficiencia no es especialmente importante, está claro que el tiempo aumenta linealmente en relación al tamaño del periodo de tiempo O(p).

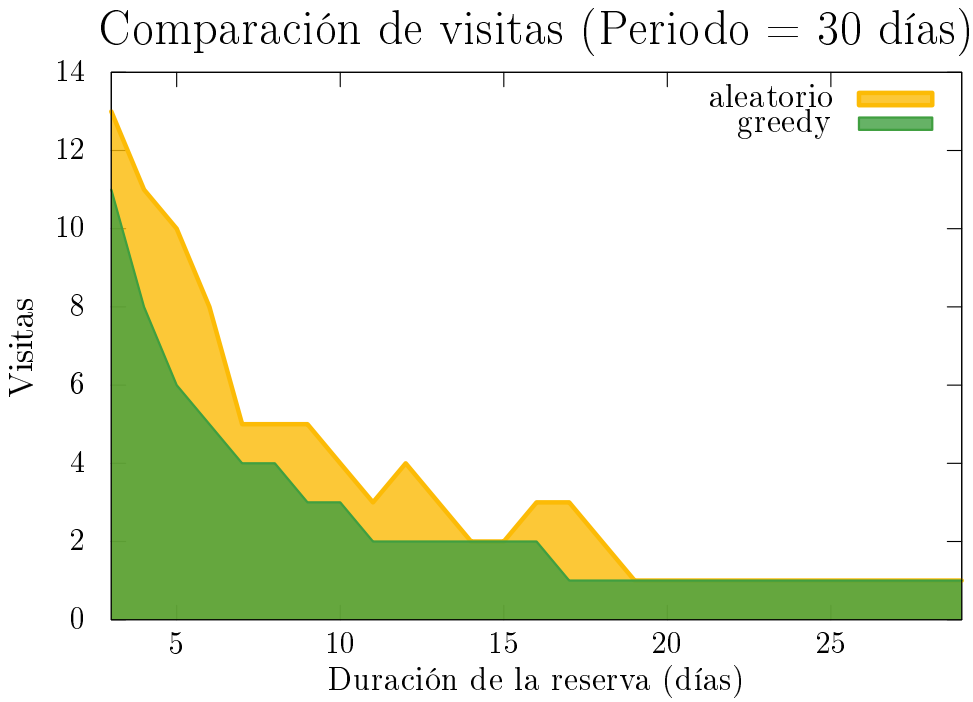
Además también aumenta al disminuir la duración de la reserva, tardando más con un periodo grande y una reserva pequeña.



1.4.2. Comparación

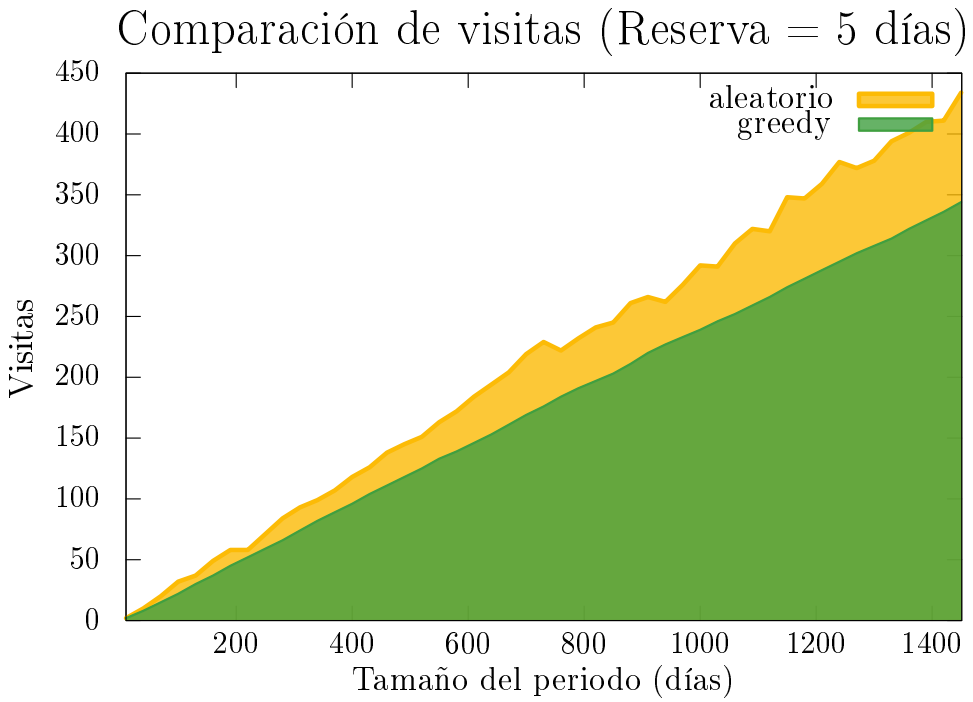
Es más interesante ver como se comporta el algoritmo respecto al número de visitas. Para esto lo compararemos con otro algoritmo que escoge los días de forma aleatoria.

Primero fijaremos el periodo de tiempo en 30 días y veremos como se comporta al variar la duración de la reserva.



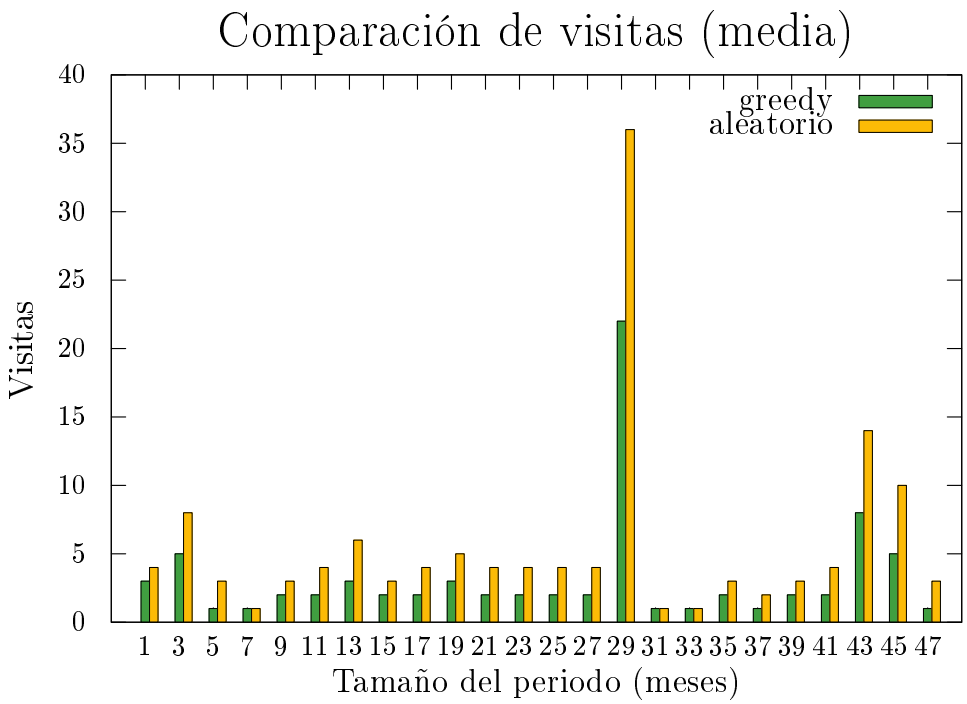
Como ya hemos probado el algoritmo greedy obtiene la mejor solución, por lo que siempre se encuentra por debajo del otro algoritmo. Además genera una gráfica descendiente, mientras que el aleatorio presenta máximos y mínimos locales.

Ahora fijamos la duración de la reserva en 5 días y variamos el tamaño del periodo. De esta manera obtenemos una gráfica creciente.



Las conclusiones son parecidas a las de la gráfica anterior. Cabe mencionar que aunque parece que las visitas aumentan de forma lineal en el algoritmo greedy, no se puede asegurar, ya que la distribución de días en los que abre la tienda es aleatoria.

También es interesante observar que ocurre de forma media. Para ello calculamos el número medio de visitas con una reserva aleatoria y en un periodo determinado.



Podemos observar dos conclusiones muy interesantes.

Primero, parece que hay una proporción entre el número de visitas medio del greedy y del aleatorio, aproximadamente $\frac{2}{3}$.

Segundo, el tamaño del periodo no influye en el número de visitas, ya que lo realmente importante es la proporción entre el periodo y la reserva y al ser esta última la media entre 1 y p las visitas se estabilizan (salvo excepciones de la probabilidad).

2. Bibliografia