

Backtracking

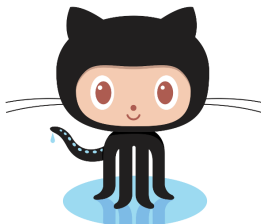
Problema 5 - Problema de la ITV

A. Herrera, A. Moya, I. Sevillano, J.L. Suarez

20 de mayo de 2015

Introducción

- En esta presentación se proporciona una solución para el ejercicio 5.
- El código, los resultados de las ejecuciones, las gráficas y los pdf asociados se puede encontrar en [GitHub](#).



Explicación del problema

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Resumen del enunciado

Una estación de ITV consta de m líneas de inspección de vehículos iguales. Hay un total de n vehículos que necesitan inspección. En función de sus características, cada vehículo tardará en ser inspeccionado un tiempo t_i , $i = 1, \dots, n$. Se desea encontrar la manera de atender a los n vehículos y acabar en el menor tiempo posible. Diseñar e implementar un algoritmo vuelta atrás que determine cómo asignar los vehículos a las líneas. Mejorarlo usando alguna técnica de poda. Realizar un estudio empírico de la eficiencia de los algoritmos.

Análisis del problema

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Información sobre el problema

- Estructura similar al **problema de los electricistas**. Conocido como **scheduling problem**.
- Cambia la función objetivo a optimizar: tiempo medio de espera vs tiempo total de los trabajos.
- Es un problema **NP-Completo**.

Análisis del problema

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Observaciones previas

- Aspiramos a una buena **solución exponencial**.
- Si $n \leq m$, podemos asignar un trabajo a cada máquina. El tiempo es el máximo de los t_i .
- Podemos suponer entonces $n > m$.
- **No importa el orden** en el que una máquina realice sus trabajos asignados.

Análisis del problema

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Proposición 1: El número de formas en las que se pueden asignar los trabajos a las máquinas es m^n .

Demostración.

- A cada trabajo se asigna una máquina, pudiendo repetirse.
- Para cada trabajo tenemos m posibilidades disponibles.
Total: m^n .

Análisis del problema

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Primer algoritmo ($\theta(m^n)$)

- Recorrer todas las soluciones posibles con backtracking.

```
def algoritmo1(k,tiempos, solucion_actual,
max_tiempo):
    if k < len(tiempos):
        sol = Inf
        for i in range(0,len(solucion_actual)):
```

Análisis del problema

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Primer algoritmo ($\theta(m^n)$)

```
solucion_actual[i] += tiempos[k]
sol = min(sol, algoritmo1(k+1,tiempos,\
solucion_actual, max(max_tiempo, \
solucion_actual[i])))
solucion_actual[i] -= tiempos[k]
return sol
else:
    return max_tiempo
```


Análisis del problema

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Mejorando el primer algoritmo

- No nos importa cuál sea la máquina que realiza un determinado conjunto de trabajos. Todas son igual de eficientes.
- **Problema:** el algoritmo anterior calcula varias veces soluciones equivalentes, una vez por cada permutación posible para las máquinas.
- **Solución:** si quedan máquinas sin tareas asignadas solo llamamos al algoritmo recursivo para la primera.

Análisis del problema

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Segundo algoritmo

```
def algoritmo2(k, tiempos, solucion_actual,
max_tiempo):
    if k < len(tiempos):
        sol = Inf
        for i in range(0, len(solucion_actual)):
            # Si la máquina anterior no tiene
            # asignado entonces no se asigna
            # trabajo a la actual (sería la
            # misma rama que la anterior).
```

Análisis del problema

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Segundo algoritmo

```
if i == 0 or solucion_actual[i-1] > 0:
    solucion_actual[i] += tiempos[k]
    sol = min(sol, algoritmo2(k+1,tiempos, solucion_actual,
                             max(max_tiempo, solucion_actual[i])))
    solucion_actual[i] -= tiempos[k]
else:
    break
return sol
else:
    return max_tiempo
```

Análisis del problema

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Mejorando el segundo algoritmo

- No queremos que una máquina esté sin realizar trabajo alguno ya que estaríamos perdiendo tiempo de trabajo.
- **Condición de poda:** si hay k máquinas libres y solo quedan por asignar k trabajos, un trabajo va a cada máquina libre.

Análisis del problema

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Tercer algoritmo

```
def algoritmo3(k, tiempos, solucion_actual,
max_tiempo, maquinas_libres):
    if k < len(tiempos):
        # Comprobamos que hay más trabajos
        # libres que máquinas libres
        if maquinas_libres < len(tiempos)-k:
            sol = Inf
```

Análisis del problema

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Tercer algoritmo

```
for i in range(0, len(solucion_actual)):
    if i == 0 or solucion_actual[i-1] > 0:
        solucion_actual[i] += tiempos[k]
        sol = min(sol, algoritmo2(k+1, tiempos, \
            solucion_actual, max(max_tiempo, \
            solucion_actual[i]), maquinas_libres +
            0 if solucion_actual[i] != 0 else 1))
        solucion_actual[i] -= tiempos[k]
```

Análisis del problema

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Tercer algoritmo

```
        else:
            break
        return sol
# Si la comprobación devuelve falso a cada
# máquina se le asigna un trabajo
else:
    return max(max_tiempo, max(tiempos[k:]))
else:
    return max_tiempo
```

Análisis del problema

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Cálculo teórico de las soluciones recorridas por el tercer algoritmo

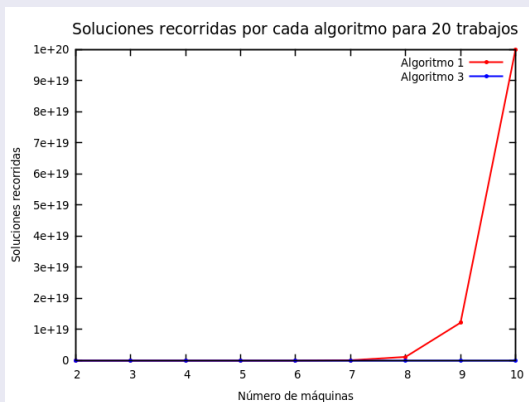
- $a_i :=$ número de soluciones con todas las máquinas ocupadas para i máquinas con $i = 1, \dots, m$.
- $a_1 = 1$
- $a_i = i^n - \sum_{j=1}^{i-1} \binom{i}{j} a_j \quad \forall i = 1, \dots, m$

Análisis del problema

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Comparación del número de soluciones recorridas



Análisis del problema

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Mejorando el tercer algoritmo

Criterio de poda habitual en backtracking: una vez se ha conseguido una solución se comprueba en cada momento si la rama actual puede conseguir una solución mejor que la mejor obtenida o no. En caso negativo se deja la rama sin terminar de visitar.

Análisis del problema

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Algoritmo final

```
def algoritmo4(k, tiempos, solucion_actual,  
    max_tiempo, maquinas_libres, mejor_solucion):  
  
    if k < len(tiempos) and  
        max_tiempo < mejor_solucion:  
        # Comprobamos que hay más trabajos  
        # libres que máquinas libres  
        if maquinas_libres < len(tiempos)-k:
```

Análisis del problema

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Algoritmo final

```
for i in range(0, len(solucion_actual)):
    if i == 0 or solucion_actual[i-1] > 0:
        solucion_actual[i] += tiempos[k]
        mejor_solucion = min(mejor_solucion,
                               algoritmo2(k+1, tiempos, solucion_actual,
                                             max(max_tiempo, solucion_actual[i]),
                                             maquinas_libres + \
                                             0 if solucion_actual[i] != 0 else 1, \
                                             mejor_solucion))
        solucion_actual[i] -= tiempos[k]
    else:
        break
return mejor_solucion
```

Análisis del problema

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Algoritmo final

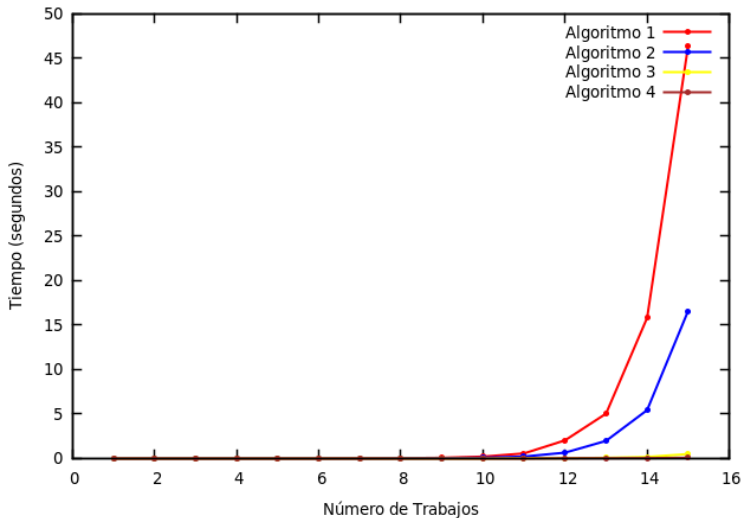
```
# Si la comprobación devuelve falso a cada  
# máquina se le asigna un trabajo  
else:  
    return max(max_tiempo, max(tiempos[k:]))  
else:  
    return mejor_solucion
```

Análisis empírico

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Comportamiento de los algoritmos para 3 trabajadores

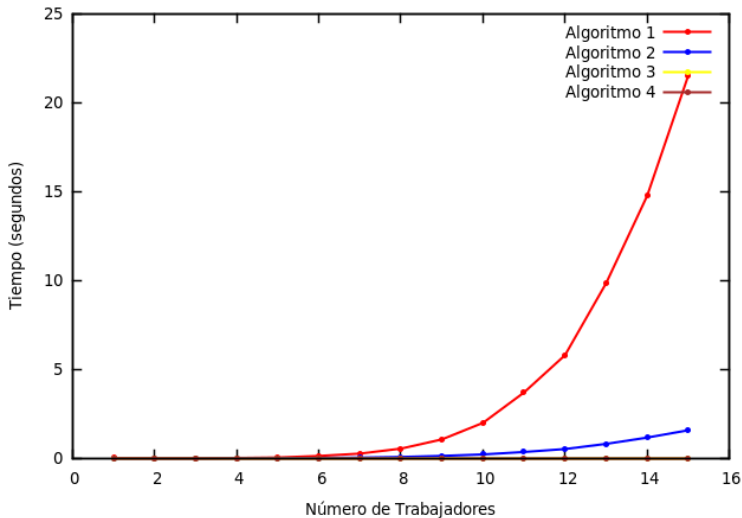


Análisis empírico

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Comportamientos de los algoritmos con 6 Trabajos

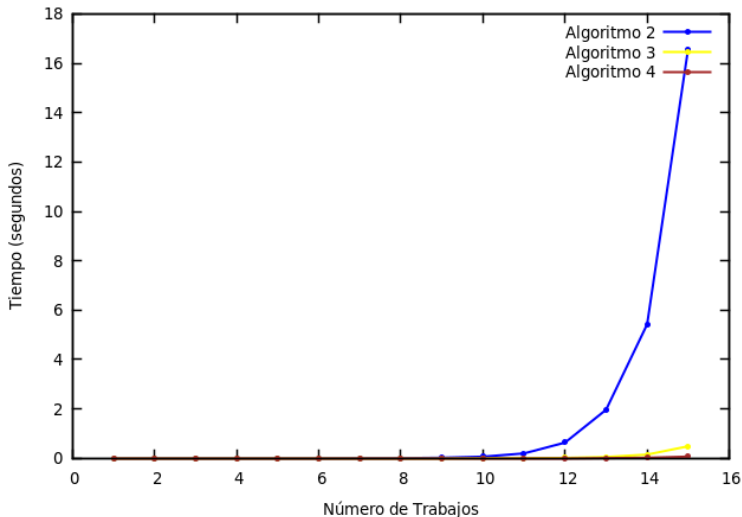


Análisis empírico

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Comportamiento de los algoritmos 2, 3 y 4 con 3 trabajadores

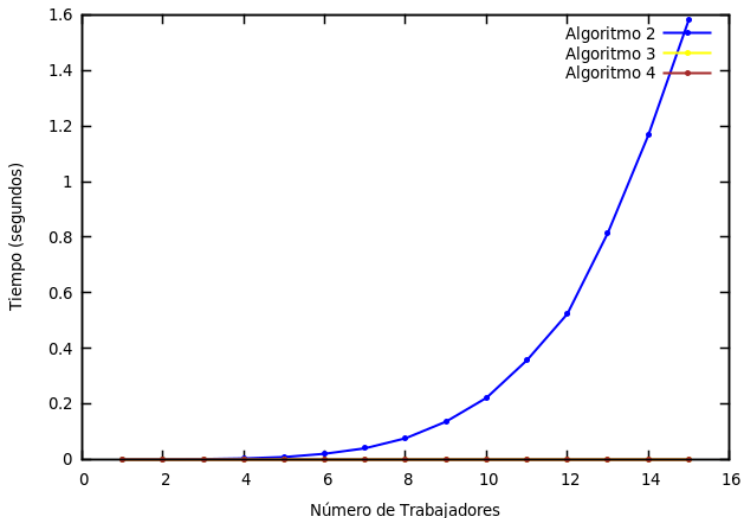


Análisis empírico

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Comportamiento de los algoritmos 2,3 y 4 con 6 trabajos

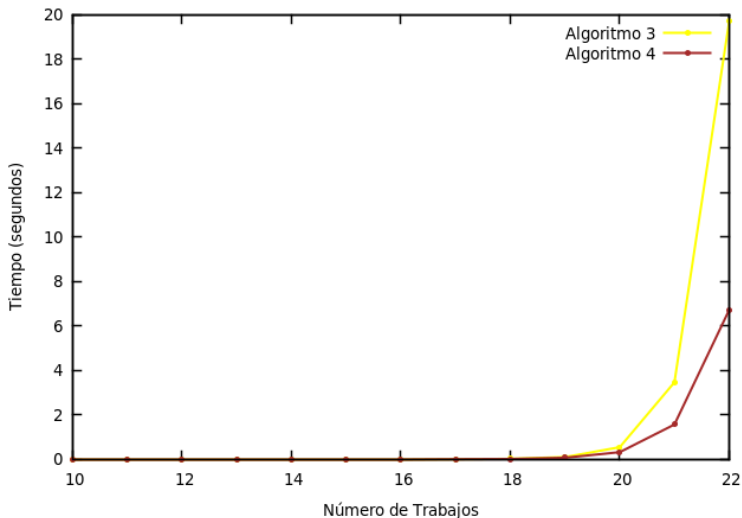


Análisis empírico

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Comportamiento de los algoritmos 3 y 4 con 10 trabajadores

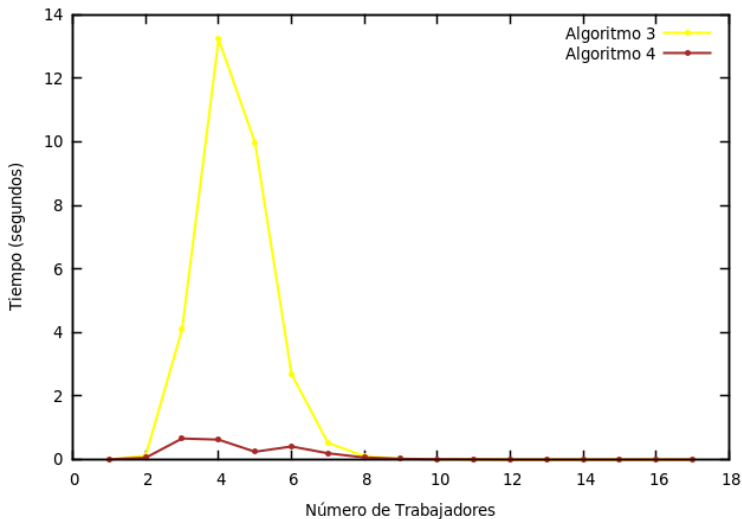


Análisis empírico

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

Comportamiento de los algoritmos 3 y 4 con 17 trabajos



Fin de la presentación

Backtracking

A. Herrera, A.
Moya, I.
Sevillano, J.L.
Suarez

¡Gracias por su atención!