Teoría de Algoritmos
(75.29) Curso Buchwald - Genender

Trabajo Práctico 1
Algoritmos Greedy

19 de septiembre de 2023

Stephanie Izquierdo	Tomás Macía	Gino Oggero
104196	99248	106675

1. Análisis del problema

1.1. Problema

Lionel Scaloni es un técnico perfeccionista y ostenta en su haber varios títulos a lo largo de sus 5 años en la Selección Argentina, entre los que se destaca la Copa del Mundo 2022. Para poder lograr esto, previo a sus encuentros con otras selecciones, hace un análisis exhaustivo de cada uno de los rivales. Esto implica ver compilados de videos, pensar estrategias acorde a cada partido y entender la mejor forma de afrontar dicho encuentro.

Sabemos que la selección tiene n rivales a los que debe enfrentar, y además que para que el análisis sea lo más detallado posible, cada rival también es analizado por un ayudante, y disponemos de tantos ayudantes como rivales a enfrentar. Esto quiere decir que cada ayudante puede ver los videos en paralelo. Los ayudantes deben ver los videos de cada rival después de que Scaloni complete su análisis. El tiempo que demora Scaloni en analizar al rival i lo llamaremos s_i y el tiempo que demora un ayudante cualquiera en analizar al mismo rival a_i.

1.2. Entendiendo el problema

Habiendo explicado el problema, el principal cuello de botella es que los ayudantes analizan al rival después de Scaloni, es decir que los ayudantes se encuentran atrasados 1 rival. Dicho de otra forma, cada vez que Scaloni empiece a analizar al rival i, habrá un ayudante empezando a analizar al rival $i-1$. De aquí se deduce que cuando el DT analice al primer rival, no habrá ningún ayudante revisando rivales, como así también cuando el técnico campeón del mundo termine su último análisis, recién en ese instante empezará el último ayudante a estudiar dicho rival.

Si no tuviéramos esta restricción y los ayudantes comenzaran al mismo tiempo que Scaloni, el cuello de botella estaría en el DT, dado que por más perfeccionista que sea, es una única persona y tiene que analizar a todos los rivales, uno por uno, por ende el menor tiempo en el que podría finalizar su estudio es la suma de los n s_i, asumiendo que el último ayudante demora menos que el mismo Scaloni, y que ninguno de los n ayudantes demora más que la suma de los todos los análisis del técnico menos el tiempo de inicio de dicho ayudante.

Volviendo a las restricciones definidas en el marco del presente Trabajo Práctico, dicho análisis puede ser extendido a los casos donde los ayudantes empiezan antes o después que Scaloni termina con un rival específico. No ahondaremos mucho en el caso que empiezan antes, pero si aprovecháramos que tenemos la misma cantidad de ayudantes que rivales, y si todos pudieran empezar antes, cada ayudante empezaría en el instante 0 y el menor tiempo en el que finalizarían todos los análisis sería el máximo entre la suma de todos los s_i o el máximo de los a_i. Es decir, el cuello de botella es Scaloni porque es uno solo estudiando los n rivales, o bien algún ayudante muy lento que se demora más que el campeón del mundo analizando a todas las selecciones rivales. En el caso donde los ayudantes empiezan después, el tiempo óptimo de finalización no puede ser nunca la suma de los s_i ya que cuando termine de observar al último rival, recién ahí comenzará un ayudante a realizar ese análisis. Y también tenemos que tener en cuenta que existe la posibilidad de que algún ayudante lento nos retrase la finalización: si un ayudante demora más que el tiempo que le resta a Scaloni terminar de estudiar a sus rivales, e incluso más si a eso le sumamos el tiempo que demoraría el último ayudante en observar al rival restante, el tiempo óptimo estará definido por el tiempo de inicio de ese ayudante. Caso opuesto, el tiempo óptimo estará definido por el tiempo de inicio del último ayudante (equivalente a la suma de los s_i) sumado al tiempo que demore ese último ayudante.

Veamos dos ejemplos para entender mejor estos casos planteados:

	Rival A	Rival B
Scaloni	10	3
Ayudantes	30	5

Cuadro 1: Ejemplo ayudante lento

En este caso, existen 2 rivales para analizar. Un primer rival cuyo tiempo es más alto comparado al otro, tanto para Scaloni como para sus ayudantes. El DT terminaría el análisis en el instante 13 (ve todos los videos de forma serial). En el instante 10 empezaría el estudio de un ayudante para el rival A, el cual terminará en el instante 40. En el instante 13, empezaría el otro ayudante, el cual finalizará en el instante 18. Para este ejemplo, el tiempo óptimo es 40. Si el orden de rivales fuera al revés, el primer ayudante empezaría al instante 3 y termina a los 8, mientras que el segundo ayudante comenzaría a los 13 y terminaría a los 43.

	Rival A	Rival B
Scaloni	10	6
Ayudantes	12	8

Cuadro 2: Ejemplo promedio

Para este ejemplo, el primer ayudante empezaría al instante 10 y terminaría en el instante 22, y el segundo ayudante empezaría a los 16 y acabaría al instante 24. El tiempo óptimo sería 24, dado que si el orden fuese invertido, el primer ayudante empezaría a los 6 y terminaría a los 14, mientras que el segundo ayudante comenzaría a los 16 y acabaría al instante 28.

A continuación se muestran 2 gráficos que ilustran de manera visual dichos ejemplos.

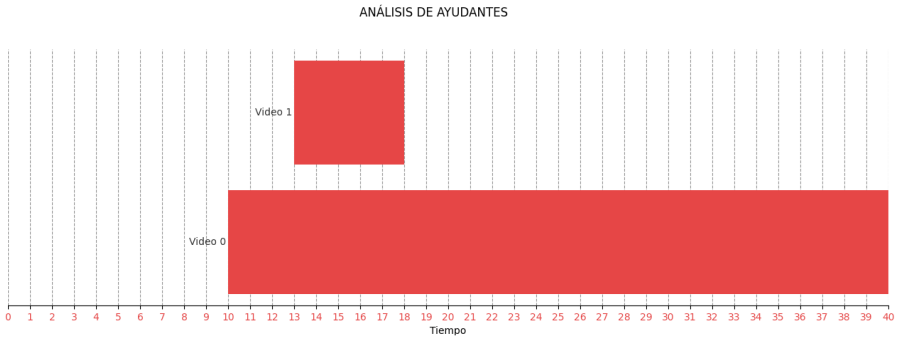


Figura 1: Representación visual Cuadro 1

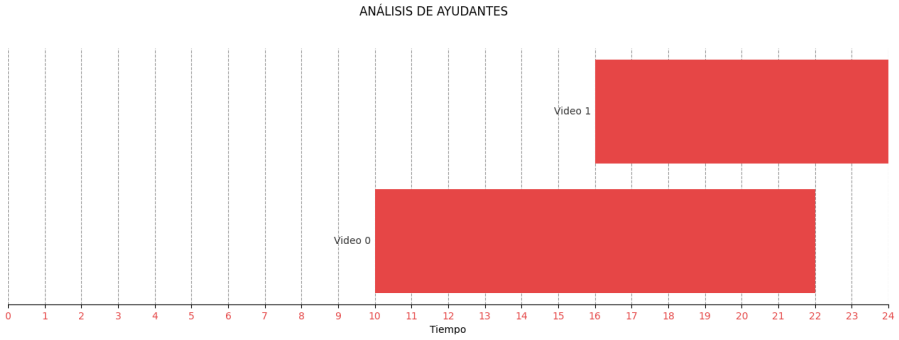


Figura 2: Representación visual Cuadro 2

1.3. Búsqueda de solución

Nuestra primera aproximación al problema consistió en minimizar el tiempo ocioso de los ayudantes para el primer rival, es decir que los ayudantes empiecen cuanto antes a analizar los rivales. Pero a la vez también nos interesa que el tiempo que demora el estudio del último rival sea el menor posible. Como son dos dimensiones analizar, decidimos calcular para cada rival la siguiente relación a_i/s_i y ordenar a los rivales según esa proporción, de mayor a menor. Nuestra idea era que los ayudantes que más demoraran en conjunto con los rivales que menos demora Scaloni, sean los primeros en ser vistos, y que el último rival a observar sea el que menor tiempo le lleva a algún ayudante.

Esta solución no funcionó. Justamente al tener en cuenta 2 dimensiones, se podría cumplir la condición pero no el objetivo final. La menor proporción se podía dar con un tiempo muy grande de Scaloni pero un tiempo no tan pequeño de algún ayudante (ya sea por la naturaleza perfeccionista de Scaloni o por la subestimación de los ayudantes). Y el menor de los tiempos de los ayudantes se podría ver contrarestado por un tiempo similar de Scaloni, o incluso mayor.

	Rival A	Rival B	Rival C
Scaloni	2	9	1
Ayudantes	4	3	1
Relación	2	0,3	1

Cuadro 3: Solución inicial no óptima

Con nuestro algoritmo inicial, nuestro primer rival a analizar sería el Rival 1. El DT empieza en el instante 0 y termina su estudio en el instante 2, empezando el análisis el ayudante en el instante 2 y terminando en el instante 6. El segundo rival sería el Rival C, comenzando el instante 2 y acabando en el 3. El último rival Scaloni termina al instante 12 y por ende el ayudante empieza apenas termina y finaliza al instante 15. El tiempo óptimo en este caso NO es 15, debería ser 14 si el orden hubiese sido Rival A, Rival B, Rival C. Lo que falla en este algoritmo es tener en cuenta al Rival C antes que el Rival B, dado que nos conviene evaluarlo a lo último. Esto sucede por tener en cuenta la proporción, es decir las 2 dimensiones, y perder de vista la dimensión que realmente nos interesa, que es el tiempo que demoran los ayudantes.

1.4. Solución

La solución óptima consiste en tener en cuenta únicamente la dimensión de los tiempos que demoran los ayudantes. Para el ejemplo de arriba, el orden sería Rival A, Rival B y Rival C, porque los tiempos de los ayudantes son 4, 3 y 1 respectivamente. Al instante 2 empezaría el primer ayudante, el cual termina a los 6. Al instante 11 empieza otro ayudante, el cual termina a los 14. Y el DT termina a los 12, mientras que el último ayudante a los 13. Por ende, el tiempo óptimo es 14. Puede observarse que si por ejemplo invirtiéramos el orden de manera que sea Rival B, Rival A y Rival C, el tiempo resultante sería 15, el cual es mayor que el óptimo.

Ahora bien, hagamos un pequeño cambio al ejemplo del Cuadro 3:

	Rival A	Rival B	Rival C
Scaloni	2	9	1
Ayudantes	4	4	1

Cuadro 4: Ejemplo desempate por igualdad

Tenemos una igualdad en tiempos de los ayudantes para el Rival A y el Rival B, ¿cuál deberíamos considerar analizar primero? ¿cambia el óptimo según cuál elijamos?. Para poder responder estas preguntas, primero hagamos cuentas con las 2 inversiones posibles:

- Orden Rival A, Rival B, Rival C: en este orden, priorizamos el que menor tiempo ocioso deja a los ayudantes (es decir, no tiene que esperar los 9 sino que espera 2 instantes). Al instante

2 empieza el primer ayudante hasta el instante 6, luego a instante 11 termina Scaloni su segundo análisis, y a los 15 termina el ayudante con el mismo video. Al instante 11 Scaloni empieza con el rival C y terimna a los 12, con lo cual el último ayudante termina al instante 13. El tiempo óptimo es de 15.

- Orden Rival B, Rival A, Rival C: al instante 9 un ayudante empezaría a analizar al rival B, y termina a los 13. Scaloni termina con el video del Rival A a los 11, y el ayudante empieza en ese instante con su estudio hasta el instante 15. Luego Scaloni termina en el instante 12 con el último Rival, mientras que el ayudante restante finaliza a los 13. El tiempo óptimo entonces es a los 15.

El tiempo óptimo de los dos casos es el mismo, y si bien este simple ejemplo no nos alcanza para demostrar que es lo mismo cuál rival analizamos primero, nos permite sacar algunas conclusiones. En el ejemplo son 2 los rivales que coinciden en los a_i, pero podrían ser más. Como el a_i es el mismo, lo que varía es el s_i y el instante inicial donde comienza cada ayudante a ver el video que le corresponde. A la larga, todos los ayudantes deben ver si o si todos los videos, y esto por ende significa que para la suma de todos los s_i cuyo a_i es el mismo, el último ayudante terminará de ver el video en el mismo momento, sea cual sea el orden que elijamos. Si es Rival A, Rival B, al instante 11 un ayudante empezará su estudio del rival, y lo mismo pasaría si eligiéramos el orden Rival B, Rival A. El tiempo máximo será siempre el mismo, independientemente del orden que escojamos.

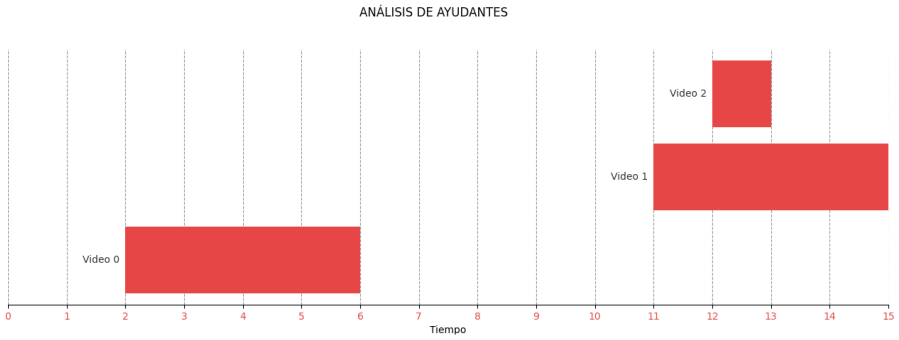


Figura 3: Representación visual Cuadro 4 - Orden A-B-C

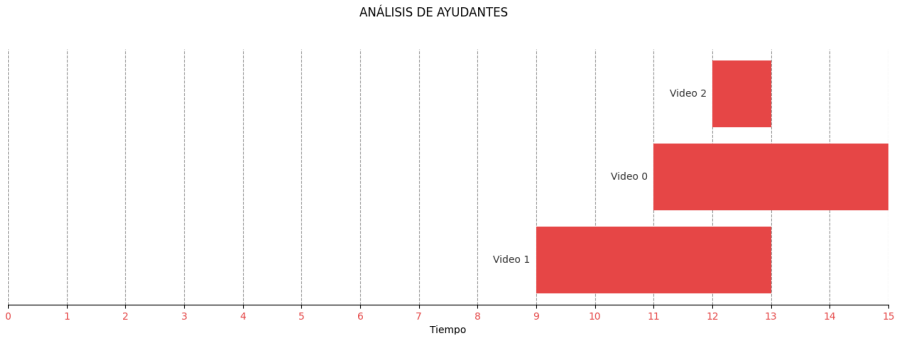


Figura 4: Representación visual Cuadro 4 - Orden B-A-C

Por lo tanto, en nuestro algoritmo no importa el orden que escojamos cuando coincidan los a_i.

Nuestro algoritmo greedy entonces consistirá en seguir los siguientes pasos:

1. Ordenar los n rivales según los a_i de mayor a menor
2. Recorrer los rivales ordenados (es decir, tomar en cada iteración el rival cuyo ayudante más demore en analizar al rival)
3. Por cada iteración, llevar un registro de:
   a) la suma de tiempo que Scaloni tardó en ver desde el video 0 hasta el video i (suma de los s_i)
   b) el tiempo que un ayudante terminará de ver el video i (suma acumulada de s_i y a_i)
   c) el ayudante que más lejano en el tiempo termine
4. Devolver el valor del ayudante que más lejano en el tiempo termine

Es un algoritmo greedy porque en cada paso elegimos la opción óptima, con el objetivo de llegar a la solución óptima general. En cada paso elegimos el ayudante que más nos retrasa el tiempo final, para tener la mayor cantidad de tiempo disponible sin que nos penalice si dicho ayudante destinara ese tiempo más adelante.

2. Algoritmo greedy

En la sección anterior fue descrito el algoritmo en términos de pseudocódigo y los pasos a seguir para ir obteniendo los óptimos. A continuación se presenta el código del algoritmo desarrollado.

```
def scaloni_greedy(times):
  times_sorted = sorted(times, key=lambda x: x[1], reverse=True)
  s_time = 0
  a_time = 0
  max_time = 0
  for scaloni, ayudante in times_sorted:
    s_time += scaloni
    new_a_time = s_time + ayudante
    max_time = max(a_time, new_a_time)
    a_time = new_a_time

  return max_time
```

La complejidad del algoritmo propuesto para encontrar el tiempo óptimo para ver los videos es $\mathcal{O}(n \log n)$, debido a que necesitamos ordenar el arreglo por los a_i de mayor a menor, y luego para cada elemento del arreglo se realizan operaciones $\mathcal{O}(1)$ (son todas operaciones de comparaciones o sumas). El algoritmo de ordenamiento elegido es el que provee el lenguaje (Python), y se trata de un algoritmo llamado Timsort, cuya complejidad en el peor escenario es $\mathcal{O}(n \log n)$

3. Ejemplos de ejecución

3.1. Ejemplos cátedra

```
files = ['3 elem.txt', '10 elem.txt', '100 elem.txt', '10000 elem.txt']

for f in files:
  sample = read_sample(f)

  print(scaloni_greedy(sample))
```

El bloque de código descripto ejecuta todos los ejemplos de la cátedra y devuelve los siguientes resultados:

- 10
- 29
- 5223
- 497886735

Dichos resultados coinciden con los resultados esperados.

Para los primeros 2 casos, vamos a ver su representación visual:

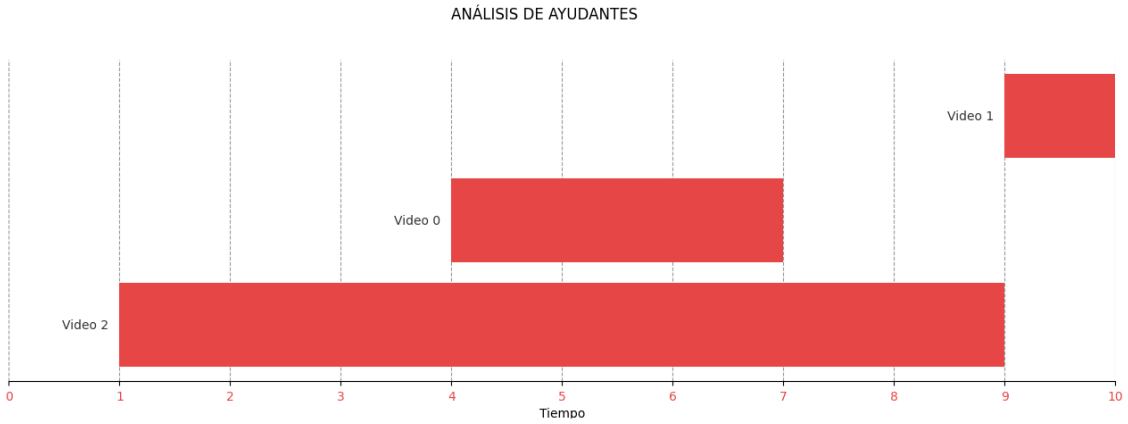


Figura 5: Representación visual ejemplo 3 elementos

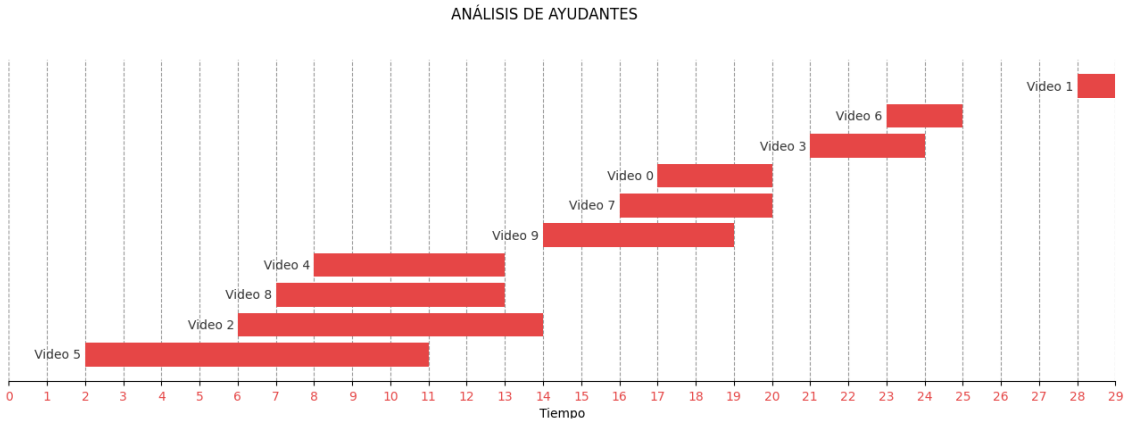


Figura 6: Representación visual ejemplo 10 elementos

3.2. Otros ejemplos

En el siguiente ejemplo vemos como hay un ayudante que nos relentiza la salida. En este caso, el óptimo no está delimitado por el último ayudante, sino un ayudante lento que termina después que el resto

	Rival 0	Rival 1	Rival 2	Rival 3	Rival 4
Scaloni	8	7	7	2	4
Ayudantes	5	9	5	1	6

Cuadro 5: Ejemplo con ayudante lento

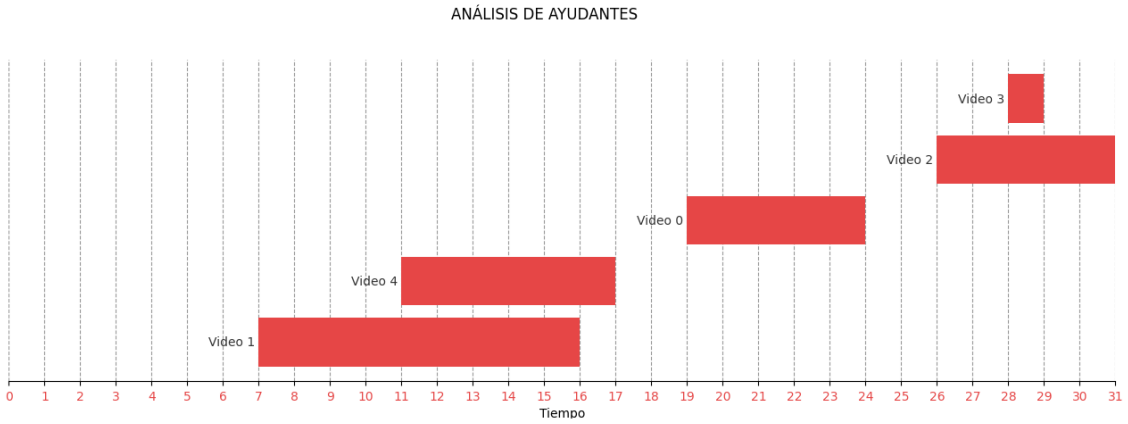


Figura 7: Representación visual Cuadro 5

Por el contrario, en este otro ejemplo vemos como el ayudante más rápido es el que dejamos para el final, dado que como siempre empieza después que Scaloni, y el DT termina siempre en el mismo instante, si dejáramos otro ayudante empezar en este mismo instante, ninguno terminaría más temprano que el ayudante que revisa al Rival 2

	Rival 0	Rival 1	Rival 2	Rival 3	Rival 4
Scaloni	4	3	5	8	7
Ayudantes	2	7	1	9	2

Cuadro 6: Ejemplo con ayudante lento pero al inicio

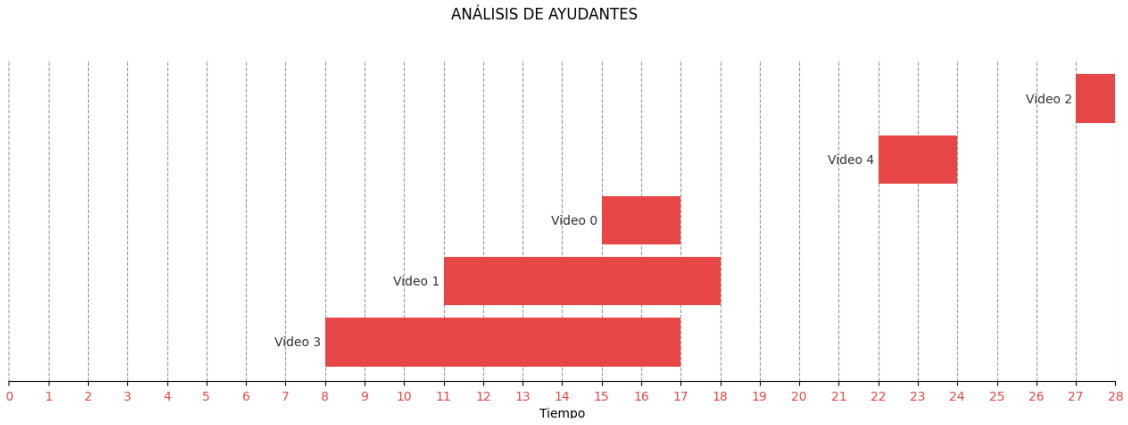
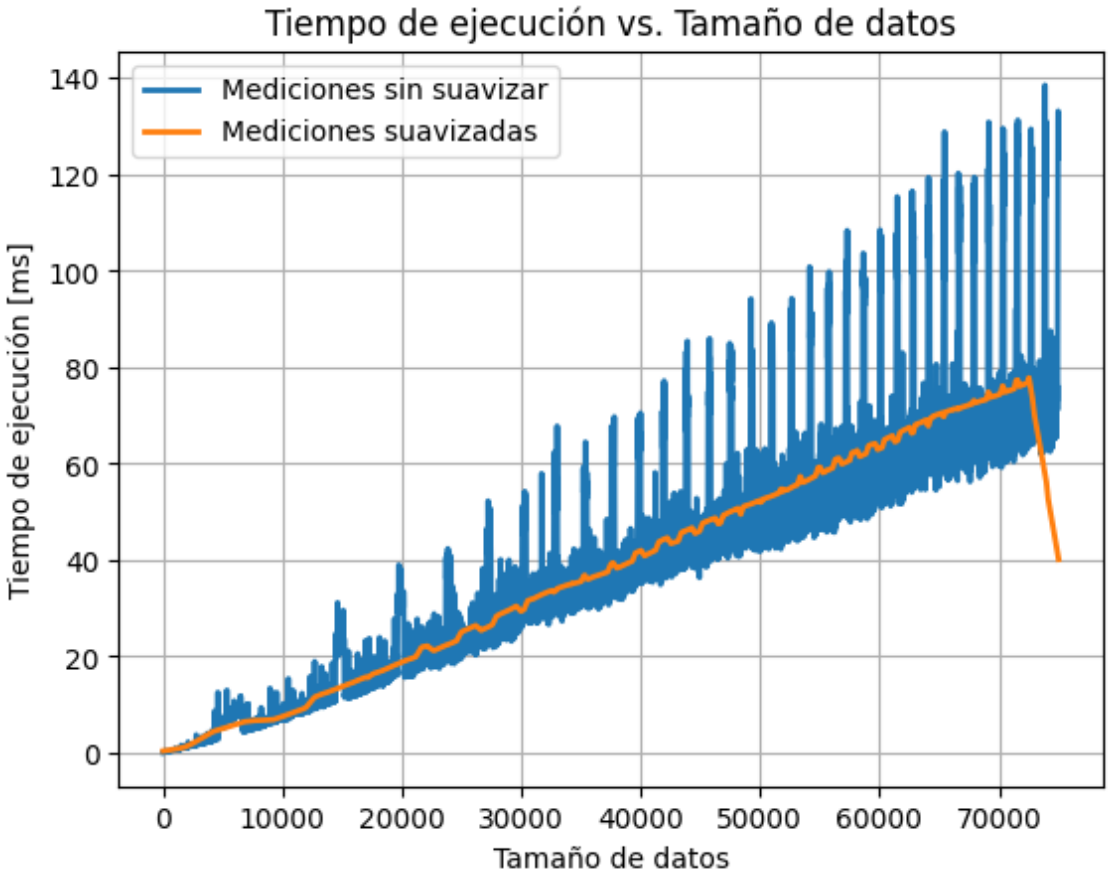


Figura 8: Representación visual Cuadro 6

4. Mediciones

Se realizaron mediciones en base a crear distintas combinaciones de s_i y a_i, cuyos valores se encuentran entre 1 y 20. Se ejecutó el algoritmo con distintos valores de n, es decir el tamaño de los datos. Estos datos van desde 10 elementos hasta 75000 elementos, con un intervalo de 10. Es decir

se calculó con 10, 20, 30.... 74990 y 75000 elementos y se midió el tiempo que tardó únicamente la ejecución del algoritmo (el tiempo de generación del set de pruebas random no fue medido). Debido al ruido presentado por las mediciones originales, decidimos suavizar los valores de la curva, para poder observar su comportamiento



5. Conclusiones

En síntesis, el enfoque desarrollado en este trabajo aborda eficazmente el desafío de optimizar el tiempo de análisis de videos de los rivales por parte de Lionel Scaloni y sus ayudantes. Se ha identificado que el factor crítico es reducir al máximo el tiempo de espera de los ayudantes que más demoran en terminar su estudio del rival, para comenzar cuanto antes su tarea de análisis y así terminar lo antes posible. A través de un algoritmo greedy, se logra asignar los rivales de manera estratégica, minimizando los tiempos de ocio y optimizando el proceso en su totalidad. Esta solución ofrece una forma efectiva y eficiente de abordar el problema, permitiendo a Scaloni y su equipo prepararse de manera más ágil y efectiva para los partidos.