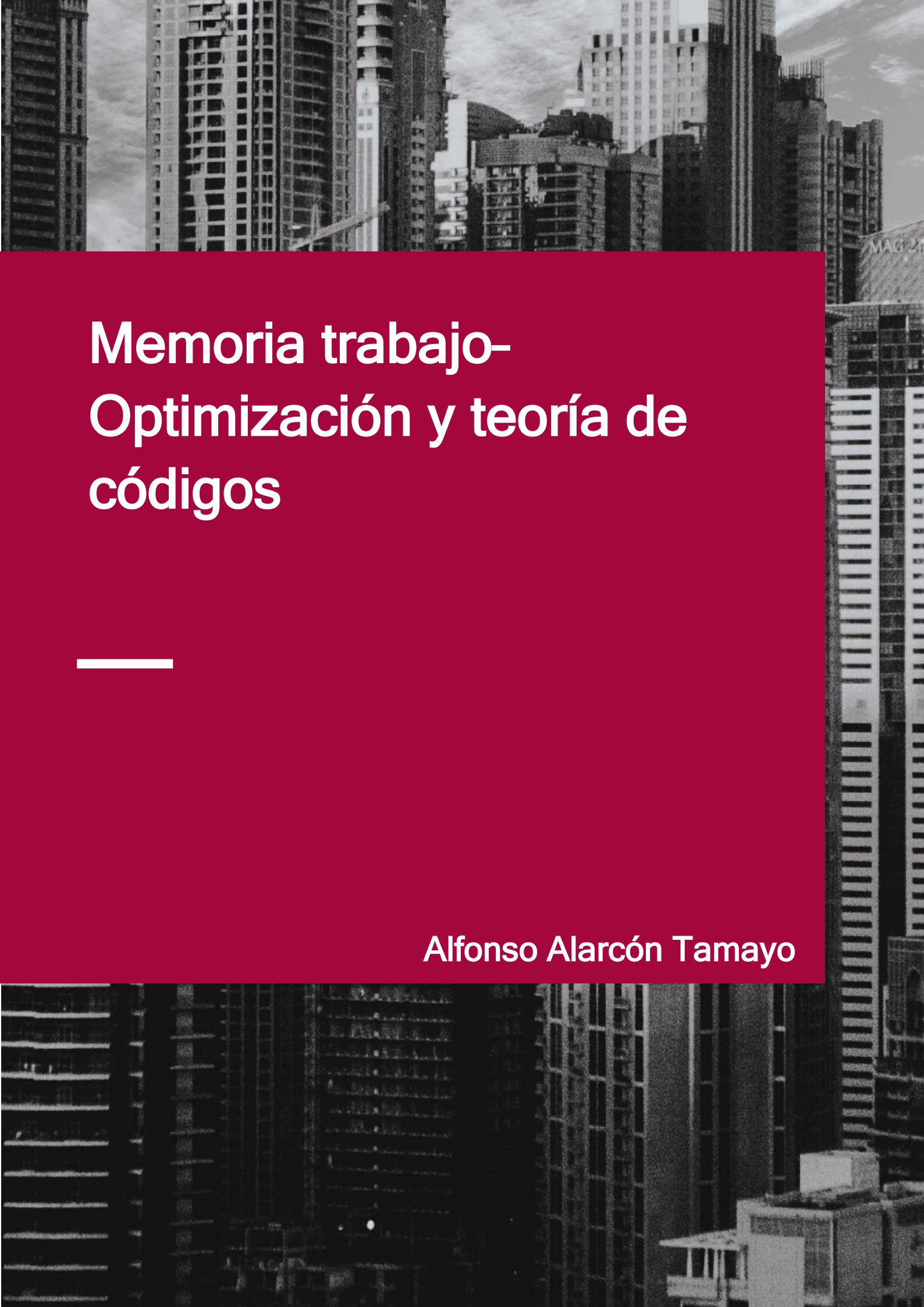

Memoria trabajo- Optimización y teoría de códigos

Alfonso Alarcón Tamayo

Una explicación del algoritmo que se ha implementado

Para obtener una solución del problema de las quinielas me he creado 5 métodos:

- Generar_S
- Generar_S_con_distancia_mas1
- Generar_C
- Calcular_N
- Enfriamiento_acero

Generar_S

En este primer apartado lo que he definido ha sido una función, que teniendo en cuenta:

las opciones posibles ([0,1,2])
el número de triples
Las restricciones correspondientes a cada apartado (número de equis, número de doses...)

Devuelve un conjunto de apuestas que son todas las combinaciones posibles y después los filtros según las restricciones del apartado (en mi caso he usado el apartado a).

Generar_S_con_distancia_mas1

Este apartado en un principio no existía, ya que, para calcular C, directamente me iba a por subconjunto aleatorio de S, pero me di cuenta al final que el profesor comentaba que la posibilidad de seleccionar bolas que no tuvieses el núcleo en el espacio S, me cree este método que a partir del conjunto S, calculo aquellas que estén a una distancia y las añado a un nuevo conjunto S que le paso al método Generar_C

Generar_C

En este método lo que he hecho ha sido una función, pasándole un el conjunto S anterior y un valor **n** (número de apuestas iniciales) y me devuelve un conjunto C, con n apuestas aleatorias.

Calcular_N

Este método calcula a partir de un espacio **S** y un conjunto de apuestas **C** si con una distancia mínima, si los centros con los radios especificados cubren la totalidad del espacio S, devolviendo una tupla que contiene, por una parte el número de apuestas sin cubrir y las propias apuestas que no están cubiertas.

Enfriamiento_acero

Este método usa todos los anteriores explicados anteriormente para calcular una solución con los parámetros dados, como en el ejemplo del enfriamiento de acero visto en las píldoras de teoría. Partimos de:

Un conjunto de atributos **C,S,T_inicial(temperatura inicial), factor de enfriamiento**
Empiezo definiendo algunas variables para el control del método, como un tiempo máximo aceptable que le doy al algoritmo para que encuentre una solución, además de un contador para ir calculando el tiempo transcurrido. Después defino una restricción para comprobar si he alcanzado una solución, en mi caso, ha diferencia de lo visto en el ejemplo de clase, mi **N** es una tupla porque me quedaré con el primer elemento de esta, que contiene el número de apuestas que faltan por cubrir.

Después recorro los valores de C y pruebo a escoger aleatoriamente un vecino de unos de los centros y comprobamos si cumplen las restricciones de que no empeora el valor de N o que cumpla con la ecuación matemática, si pasa alguna de las dos sustituimos ese núcleo por el nuevo y comprobamos si es la solución, si es ahí, el método se detiene devolviendo la solución C y el tiempo de ejecución.

En caso de que no lo encuentre, sigue iterando durante todo el tiempo establecido y va reduciendo la temperatura (T) en función del factor de enfriamiento.

Para más información, por favor consulte el anexo con el código

Listado explícito del mejor recubrimiento encontrado

El mejor recubrimiento que he encontrado ha sido con un valor de **n = 42** y he tardado unos 58.38 segundos.

El valor de C que me ha devuelto esta solución es:

C42 = [(0, 0, 2, 1, 1, 0), (2, 1, 0, 1, 0, 1), (1, 0, 2, 0, 1, 0), (0, 1, 1, 0, 0, 2), (1, 0, 0, 0, 2, 2), (1, 0, 1, 0, 0, 2), (1, 0, 0, 2, 0, 0), (2, 2, 1, 2, 0, 0), (0, 0, 1, 2, 2, 0), (0, 0, 2, 2, 0, 1), (0, 0, 1, 0, 1, 2), (0, 0, 1, 1, 0, 2), (0, 2, 0, 0, 1, 1), (0, 0, 0, 2, 1, 2), (2, 2, 0, 2, 1, 0), (0, 1, 0, 1, 1, 2), (2, 2, 1, 2, 1, 0), (2, 0, 0, 1, 0, 0), (1, 0, 2, 1, 0, 0), (0, 2, 0, 1, 0, 1), (1, 2, 0, 0, 0, 1), (2, 1, 0, 0, 0, 0), (0, 2, 1, 0, 0, 2), (2, 0, 0, 0, 1, 2), (1, 1, 0, 1, 0, 2), (0, 1, 2, 0, 1, 0), (2, 0, 0, 2, 2, 1), (2, 2, 1, 2, 0, 1), (0, 1, 0, 0, 2, 1), (0, 0, 2, 0, 2, 1), (1, 1, 0, 1, 2, 0), (1, 1, 2, 0, 0, 0), (0, 2, 1, 0, 0, 0), (2, 0, 1, 0, 2, 0), (0, 2, 0, 1, 0, 0), (2, 0, 2, 0, 0, 1), (1, 2, 0, 0, 1, 0), (0, 0, 0, 1, 2, 2), (0, 1, 0, 2, 0, 0), (0, 1, 2, 1, 0, 0), (0, 2, 1, 0, 2, 0), (0, 0, 2, 0, 0, 1)]

Tabla de ensayos

En este apartado he ido indicando algunos ejemplos para un número de C cada vez menor y el tiempo de ejecución que ha llevado. El profesor comenta que él, optimizando el código ha obtenido una solución de n=40, en un tiempo aceptable (alrededor de unos 60 segundos). Por lo que he tratado de acercarme a su solución, aunque para tiempo total aceptable he contemplado que sea inferior a 80 segundos.

Nº ensayo	Tiempo (en segundos)	Tamaño (n) del conjunto	Modificaciones a los valores
1	21.33	100	T=1.9 f.enfriamiento=0.9
2	28.05	80	
3	25.17	60	T=1.9 f.enfriamiento=0.9
4	36.07	50	
5	45.76	45	T=1.9 f.enfriamiento=0.95
6	45	44	
7	59.93	43	
8	48.84	43	T = 1.9
9	76.74	42	f.enfriamiento= 0.99
10	58.38	42	T=1.9 f.enfriamiento=0.9

Comparativa en la cota

Partiendo de la solución que nos dan los profesores para un espacio sin acotar de 729 casos posibles , obtiene una solución con un total de apuestas de 73. Pero este valor no nos interesa para nuestro subespacio, ya que aquellas que su radio de 1 no englobe apuestas del subconjunto no nos interesan, por lo que podemos reducir mucho este número.

En nuestro caso al disponer de un subespacio acotado el número de apuestas posibles se reduce a 270, por lo que el número de centros que necesitamos para disponer de una solución que cubra el espacio total se reduce drásticamente, por lo que debemos refinar el método para optimizar el problema, yo en mi caso he llegado en un tiempo aceptable a una solución con C=42.

Podemos ver que ambos cubren el espacio acotado, pero en nuestro caso lo hemos conseguido con un valor de n menor.