Métodos de Monte Carlo

Entrega 1

(Unidad 1, Sesiones 1 y 2) Ejercicio 2.1

Integrantes

Nombre	CI	Correo
Germán Ouviña	4.823.566-1	german.ouvina@fing.edu.uy

Índice

1. Problema

Supongamos que para construir una casa debemos efectuar la siguiente lista de tareas:

- T1 - cimientos - tiempo aleatorio uniforme entre 40 y 56 hs.
- T2 - contrapiso - tiempo aleatorio uniforme entre 24 y 32 hs.
- T3 - paredes - tiempo aleatorio uniforme entre 20 y 40 hs.
- T4 - techo - tiempo aleatorio uniforme entre 16 y 48 hs.
- T5 - instalación sanitaria - tiempo aleatorio uniforme entre 10 y 30 hs.
- T6 - instalación eléctrica - tiempo aleatorio uniforme entre 15 y 30 hs.
- T7 - cerramientos - tiempo aleatorio uniforme entre 20 y 25 hs.
- T8 - pintura - tiempo aleatorio uniforme entre 30 y 50 hs.
- T9 - revestimientos sanitarios - tiempo aleatorio uniforme entre 40 y 60 hs.
- T10 - limpieza final - tiempo aleatorio uniforme entre 8 y 16 hs.

Hay ciertas dependencias que implican que una tarea no puede comenzar hasta haberse terminado otra previa:

- T2, T3 dependen de 1.
- T4 depende de 2 y 3
- T5 depende de 2 y 3
- T6 depende de 3
- T7 depende de 3
- T8 depende de 4, 5, 6 y 7
- T9 depende de 5
- T10 depende de 7, 8 y 9

Entrega 1: Ejercicio 2.1 (INDIVIDUAL):

1. Implementar un programa que reciba como parámetros de línea de comando (o pregunte en pantalla) la cantidad de replicaciones n a realizar, y emplee Monte Carlo para calcular (e imprimir) la estimación del tiempo total promedio desde que se comienza la obra hasta que se finaliza la misma, y la desviación estándar de este estimador.

2. Incluir código para calcular el tiempo de cálculo empleado por el programa.

3. Utilizar el programa con $n = 10, 10^2, 10^3, 10^4, 10^5, 10^6$ y mostrar en una tabla las estimaciones de media y desviación estándar, así como los tiempos de cálculo. Discutir estos resultados. (en caso que el tiempo de ejecución para 10^6 replicaciones sea menor que 60 segundos, agregar experimentos con mayor número de iteraciones, siempre multiplicando por 10, hasta que uno de los experimentos supere esa duración).

2. Solución

2.1. Descripción

Al tratarse de un problema de optimización, el cual se busca resolver con un método de monte carlo sencillo, el desafío se encuentra en modelar el problema.

Respetando las dependencias establecidas en la consigna, se asume que cada tarea puede realizarse en paralelo a menos que se indique lo contrario. Cualquier tarea T_a que dependa de una tarea T_b puede comenzar a realizarse ni bien la tarea T_b haya sido completada. De esta manera, cuando una tarea T_a depende de múltiples tareas (por ejemplo, T_b, T_c, T_d), se asume que la tarea T_a comienza a realizarse inmediatamente las tareas T_b, T_c y T_d son completadas, lo cual implica que el tiempo de espera para T_a, si las tareas que la bloquean son realizadas en paralelo, será igual al tiempo más largo entre todas las anteriores.

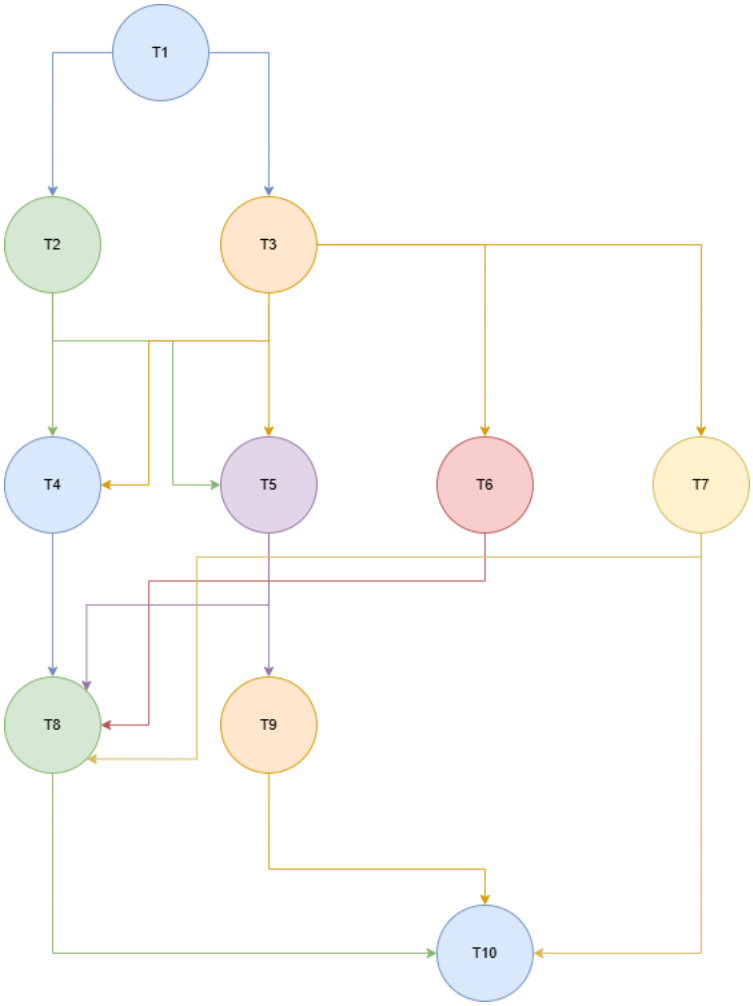

Figura 2.1: Grafo de interdependencia de tareas

Siguiendo los lineamientos de la consigna, las interdependencias entre tareas pueden modelarse con los siguientes niveles. De esta manera, el tiempo total va a ser el tiempo acumulado entre los mayores de cada nivel del grafo.

2.2. Pseudocódigo

Procedimiento EstimacionMonteCarlo (int n, real $\hat{X}$, real $\hat{V}$)
Parámetros de entrada: n (tamaño de la muestra)
Parámetros de salida: $\hat{X}$, $\hat{V}$ (estimadores de esperanza y varianza)

1. $\hat{X} = 0$
2. $\hat{V} = 0$
3. For $i = 1, ..., n$ do
   a) Sortear valores para $T_1, ..., T_{10}$ siguiendo la distribución $U(a, b)$ especifica para cada tarea
   b) Acumular valores según dependencias, generando valores $t_1, ..., t_{10}$ auxiliares
   c) $\hat{X} = \hat{X} + t_{10}$
   d) $\hat{V} = \hat{V} + (t_{10})^2$
4. $\hat{X} = \hat{X}/n$
5. $\hat{V} = \hat{V}/(n * (n-1)) - \hat{X}^2/(n-1)$

El anterior pseudocódigo bosqueja un método de monte carlo sencillo, siguiendo las pautas del pseudocódigo presentado en clase, agregando un paso extra luego del sorteo de variables: la acumulación condicional en base a las dependencias. Las variables auxiliares $t_1, ..., t_{10}$ buscan representar el tiempo total entre el comienzo de la obra y la finalización de la tarea t_i, en contraposición con T_i que representa únicamente el tiempo que dicha tarea lleva en realizarse. De esta manera, la variable t_{10} contiene la cantidad de tiempo total que se busca modelar, siendo el valor final acumulado en $\hat{X}$.

Sin entrar en detalles para cada tarea (el código tiene comentarios con respecto a la acumulación de cada tarea), un ejemplo del proceso de acumulación según dependencias funciona con las tareas T_1, T_2, T_3 y T_4. Tanto T_2 como T_3 dependen de T_1, por lo que los tiempos desde el inicio hasta el final de cada tarea se calculan como $t_2 = T_2 + T_1$ y $t_3 = T_3 + T_1$. Al momento de calcular T_4, se comprueba cual de las dos dependencias es mayor, acumulando dicho valor. En resumen, $t_4 = T_4 + max(t_2, t_3)$. El mismo proceso se repite para el resto de tareas, lo cual lleva a eventualmente acumular la totalidad de la duración de la obra en t_{10}.

2.3. Código

El código para solucionar el problema se encuentra adjunto en un archivo *.zip* junto al informe, y se encuentra disponible en el repositorio https://github.com/gouvina-fing/fing-mmc2025.

El mismo fue desarrollado en *Python*, cuenta con instrucciones de como instalarlo y correrlo, y ofrece la posibilidad de ejecutar el método siguiendo un enfoque iterativo (como el discutido en el pseudocódigo anterior) o un enfoque vectorial, aprovechando las ventajas del lenguaje y las bibliotecas disponibles para operaciones vectoriales. Este último enfoque ofrece los mismos resultados exactamente, bajo tiempos de ejecución menores pero mayor coste de almacenamiento en memoria. Se agregó la opción como simple experimentación, y en la siguiente sección se contrastan los tiempos de ejecución entre ambos métodos de manera anecdótica.

A su vez, el código también ofrece la posibilidad de generar tablas y gráficas comparativas siguiendo la consigna de la parte 3 del ejercicio planteado, automáticamente ejecutando el método para los distintos valores de n mencionados y cortando la ejecución cuando el tiempo supera los 60 segundos.

3. Experimentación

3.1. Especificaciones

A continuación, se desglosan las especificaciones técnicas utilizadas durante la ejecución:

- **Procesador:** 11th Gen Intel(R) Core(TM) i5-11400F @ 2.60GHz
- **Memoria RAM:** 16.0 GB
- **Sistema Operativo:** Windows 10
- **Semilla:** 42
- **Tamaños de muestra:** 10, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7

3.2. Resultados

A continuación, se muestra una tabla comparativa de los resultados obtenidos al ejecutar el método para cada cantidad de iteraciones pedida:

N° de iteraciones	**Valor de $\hat{X}$ (h)**	**Valor de $\hat{V}$ (h)**	**Tiempo iterativo (s)**	**Tiempo vectorial (s)**
10	165.994424	10.280895	0.000000	0.000000
10^2	167.130305	1.213965	0.002684	0.000000
10^3	168.279518	0.095294	0.029005	0.000000
10^4	168.573983	0.010683	0.281049	0.001000
10^5	168.590124	0.001054	2.678553	0.015029
10^6	168.566021	0.000106	26.986812	0.129291
10^7	168.563704	0.000011	267.537041	1.303841

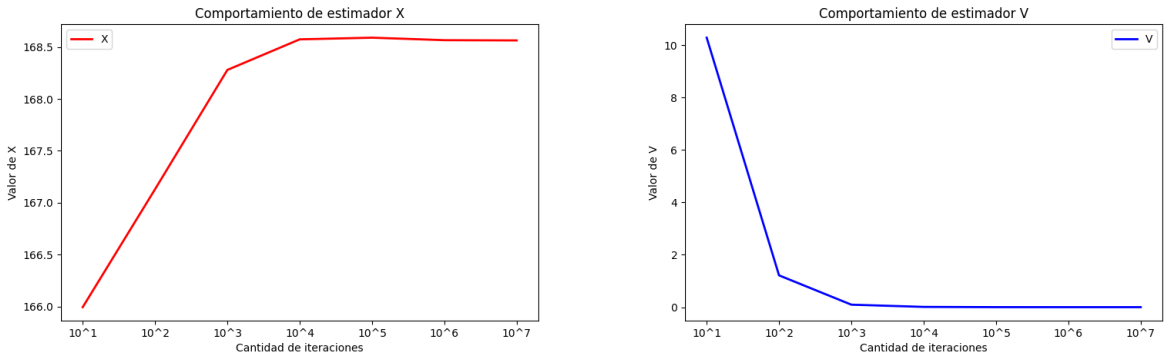

Figura 3.1: Grafo de interdependencia de tareas

Tanto en las gráficas como en la tabla, se puede observar que a partir de las 10^4 iteraciones el valor del estimador $\hat{X}$ se mantiene alrededor de 168,5. La varianza $\hat{V}$ a su vez parece tener un comportamiento asintótico hacia 0, a medida que el valor de iteraciones aumenta.