PROYECTO 2 - SIMULADOR DE TEATRO DE OPERACIONES PARA DRONES

Autores

Alan Argotte, 19-10664.

David Díaz, 20-10019.

Profesor

Guillermo Palma

Sartenejas, 11 de marzo de 2025.

Descripción del sistema

Los autores (referidos como los *programadores*) proponen dos soluciones generales al problema (referido como el *sistema*) que es adaptada según el tamaño de la instancia del problema. Es decir, la solución basada en hilos y la solución basada en procesos usan la misma plantilla algorítmica para resolver el sistema.

La figura 1 adjunta a continuación describe gráficamente la primera idea que esta plantilla algorítmica usa para implementar la solución de detectar *cuando* un objetivo militar o un objetivo civil es alcanzado por un dron a través de coordenadas cartesianas sobre una matriz $N \times M$. Adicionalmente, cabe destacar que susodicha matriz no existe *per se* en la memoria del sistema computarizado, sino que más bien plantea el modelo conceptual a través del cual las ecuaciones propuestas reflejan la realidad que se quiere representar en el sistema.

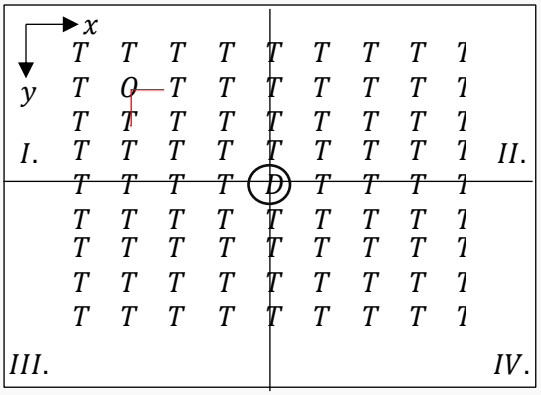

Figura. 1: Eje coordenado con origen en la esquina superior izquierda de la matriz con un dron que explota con radio 3, afectando al objetivo O

Sea el punto (x_0, y_0) la posición de un dron cualquiera. Adicionalmente, (x, y) la posición de un objetivo cualquiera. Se cumple la relación, que $x < x_0 \wedge y < y_0$. Asimismo, dado el radio de explosión r del dron, tenemos que

$$x_0 - r < x \wedge y_0 - r < y$$

Donde $(x_0 - r, y_0 - r)$ representa la esquina superior derecha del radio de explosión con origen en (x_0, y_0). Si se cumple esta condición, tenemos que T es alcanzado por el dron en el cuadrante I. De forma similar, tenemos que para los cuadrantes II., III. Y IV. Se cumple,

$$(II)\ x_0 + r < x \wedge y_0 - r < y\,, \qquad (III)\ x_0 - r < x \wedge y_0 + r > y,$$
$$(IV)\ x_0 + r > x \wedge y_0 + r > y$$

Donde los cálculos en los que participan x_0 y y_0 representan, para el cuadrante II., la esquina superior izquierda; para el cuadrante III., la esquina inferior izquierda; para el cuadrante IV., la esquina inferior derecha. Con estas condiciones es posible determinar si un objetivo es alcanzado por un dron o no. Este cálculo se hace para cada par de dron y objetivo; por lo tanto, su complejidad viene dada por $o * d$, siendo o la cantidad de objetivos y d la cantidad de drones.

La segunda idea que utiliza la plantilla de solución sí consiste en implementar una matriz de tamaño $N \times M$ de apuntadores a objetivos militares o civiles. Luego, dado un dron cualquiera en la posición (x, y) y de radio de explosión r, son buscados objetivos en la subcuadrícula con esquinas superiores e inferiores izquierda y derecha dadas por los puntos $(x - r, y - r)$, $(x + r, y - r)$, $(x - r, y + r)$ y $(x + r, y + r)$, respectivamente; en caso de que exista un objetivo en la subcuadrícula, su resistencia es debidamente actualizada.

Note de la figura 2 adjunta que el dron D es capaz de alcanzar al objetivo O; por ende, se le restaría al objetivo O todo el daño que ocasiona el dron D.

Es importante destacar que esta solución se ejecuta para cada uno de los drones, luego, no es difícil demostrar que la complejidad del algoritmo viene descrita por: $(2r_0 + 1)^2 + (2r_1 + 1)^2 + \cdots + (2r_n + 1)^2$

Siendo r_i el radio de alcance del i-ésimo dron, y n la cantidad de drones.

El sistema calculará la cantidad de iteraciones que tiene que ejecutar para decidir qué solución utilizará, se decidirá por la primera solución (solución aritmética) solo si $o * d \leq (2r_0 + 1)^2 + (2r_1 + 1)^2 + \cdots + (2r_n + 1)^2$; en caso contrario, preferirá la segunda solución (solución matricial).

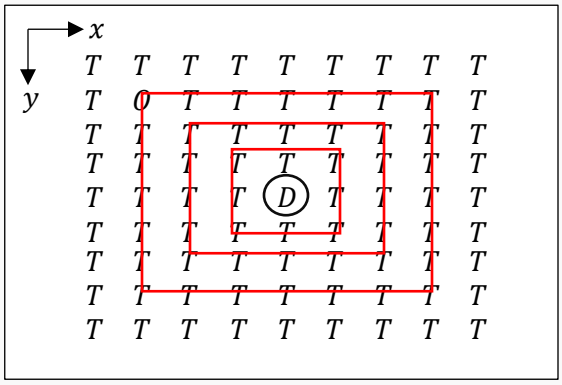

Figura. 2: Eje coordenado con origen en la esquina superior izquierda de la matriz, donde un dron D explota con radio 3 afectando al objetivo O

Con estas dos soluciones, los programadores pueden distinguir dos recursos esenciales que deben ser compartidos por los procesos e hilos: un arreglo de drones y un arreglo de objetivos. Cierto número de drones deben ser compartidos por cada proceso e hilo y deben actualizar la resistencia resultante de un objetivo, siendo posible que sea afectado por más de un proceso o hilo a la vez, convirtiéndose así en una sección crítica.

Solución basada en hilos:

La solución consiste en que, dados k drones y n hilos, un hilo se encarga de computar el daño provocado a los objetivos por, aproximadamente, $\frac{k}{n}$ drones y actualizar el arreglo de objetivos tal que reflejen el daño causado por los drones. Con este propósito, son implementados arreglos locales a cada hilo cuyo i-ésimo elemento corresponde al daño acumulado de los $\frac{k}{n}$ drones en cada objetivo y que es posteriormente sumado al i-ésimo elemento del arreglo de objetivos si el objetivo no ha sido ya destruido por otro dron. En la función $compute\_damage()$, son implementados mutexs tales que aseguran la exclusión mutua entre aquellos hilos que quieren actualizar el arreglo de objetivos con la resistencia sobrante después de un ataque de dron.

De forma similar, la segunda solución consiste en verificar el daño provocado por cada uno de los drones en las subcuadrículas con centro en la posición (x, y) donde cae un dron en la matriz $N \times M$ y actualizar aquellos objetivos afectados en el arreglo de objetivos.

La primera o segunda solución son implementadas en función de cuál deba realizar menos operaciones.

Solución basada en procesos:

La solución basada en procesos es análoga a la solución basada en hilos. Dados k drones, n procesos y k objetivos, un proceso computa el daño provocado a los k objetivos por los $\frac{k}{n}$ drones asignados; dada la matriz $N \times M$, verifica si los $\frac{k}{n}$ drones impactan algún objetivo en la subcuadrícula con centro en la posición (x, y) donde cae cada uno de los drones.

Para implementar este par de soluciones en el contexto de procesos, un proceso padre genera a través de la llamada al sistema $fork()$ n procesos hijos que se comunican a través de memoria compartida en una dirección que pertenece al espacio de memoria del proceso padre. Los procesos hijos actualizan los objetivos en el arreglo de objetivos en este espacio de memoria y se sincronizan entre sí a través de mutexs para evitar condiciones de carrera en la función $compute\_damage()$. El proceso padre espera a los n procesos hijos y termina el programa retornando la información actualizada en el arreglo de objetivos.

Referencias bibliográficas

Delorie, DJ. Malloc Internals (2022). Encontrando en:
https://sourceware.org/glibc/wiki/MallocInternals

Ippolito, G. POSIX thread (pthread) libraries (2004). Encontrado en:
https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html#CREATIONTERMINATION

Palma, G. Sistemas de Operación I. (2025). Clases de laboratorio del curso Sistemas de Operación I. (CI-3825) impartidas durante el trimestre Enero-Marzo 2025.