SIMULACIÓN DE SISTEMAS

Práctica 3:

Modelos de Simulación Dinámicos y Discretos

Curso 2019/2020

Capítulo 1

Mi Segundo Modelo de Simulación Discreto

1.1. Planteamiento del sistema a estudiar

Consideremos un sistema consistente en una entidad (servidor) que presta determinado servicio a una serie de entidades (clientes) que periódicamente llegan a solicitar dicho servicio. Por ejemplo, los clientes pueden ser máquinas o piezas que necesitan ser reparadas y el servidor un técnico reparador, o personas que van a ser atendidas por un cajero de un banco o de un supermercado, etcétera. Disponemos de información sobre los tiempos entre llegadas sucesivas de clientes, A_1, A_2, A_3,... (que son variables aleatorias independientes e idénticamente distribuidas; un generador de datos se encargará de proporcionar esos valores en tiempo de simulación). Un cliente que llega y encuentra al servidor libre pasa a ser atendido inmediatamente. También disponemos de información sobre los tiempos de servicio, S_1, S_2, S_3,... de los clientes sucesivos (que son variables aleatorias independientes e idénticamente distribuidas, e independientes de los tiempos entre llegadas; otro generador de datos proporcionará esos valores). Un cliente que llega y encuentra ocupado al servidor debe esperar turno. Cuando se completa un servicio, el servidor elige a un cliente de los que esperan, si lo hay, en una forma FIFO (primero en llegar, primero en ser servido, o sea, los clientes esperan formando una cola). Queremos simular este sistema hasta que n clientes hayan sido atendidos, y estimar el número medio de clientes en cola, $Q(n)$, y el porcentaje de tiempo de ocio del servidor, $PTO(n)$.

Asumiremos que al empezar la simulación no hay clientes esperando y el servidor está libre. Utilizaremos un generador de datos de tiempos entre llegadas y uno de tiempos de servicio ambos exponenciales de medias `tlleg` y `tserv`, respectivamente, expresadas, por ejemplo, en horas. Así, `tlleg`=0.15 (9 minutos) y `tserv`=0.1 (6 minutos).

1.2. Gestión del tiempo. Métodos de incremento fijo e incremento variable de tiempo

Vamos a construir primero un sencillo modelo de simulación basado en lo que se llama método de *incremento fijo de tiempo*, y sin la estructuración que conviene utilizar (que ya se verá más adelante). El tiempo simulado se va a mover de unidad en unidad (como un reloj

convencional), aunque esa unidad de tiempo puede ser la que mejor nos parezca: minutos, medias horas, horas, días,... Posteriormente construiremos otro modelo con el método de *incremento variable de tiempo*.

1.2.1. Pseudocódigo del programa de simulación con incremento fijo

A continuación aparece el pseudocódigo para la programación del modelo de simulación.

```
inicializar generadores de datos //imprescindible (por ejemplo, srand(time(NULL)))
infinito = 10e30; //tiempo en que ocurren cosas que sabemos no pueden ocurrir
atendidos = 0; //al principio no hay nadie ya atendido
inicio_ocio = 0.0; //marcará el momento en que el servidor empieza a estar ocioso
acum_cola = 0.0; //acumulador de número de clientes en cola por el tiempo en que
       //están en dicha cola. El cliente que está siendo atendido NO está en cola
reloj = 0; //marca el valor del tiempo simulado, inicialmente cero
servidor = libre; //inicialmente el servidor no está atendiendo a nadie
encola = 0; //no hay nadie en cola todavía
tiempo_llegada = reloj + generallegada(tlleg); //llegará el primer cliente
tiempo_salida = infinito; //nadie puede salir si nadie ha entrado aun
while (atendidos < total_a_atender) //simularemos hasta que hayamos atendido a
                                    //cierto número de clientes, total_a_atender
     {
      if (reloj == tiempo_llegada) //si estamos en el instante en que llega alguien
       {
        tiempo_llegada = reloj + generallegada(tlleg); //determinamos cuando llegará
                                                       //el siguiente cliente
        if (servidor == libre) //si el servidor está ocioso
             {
              servidor = ocupado; //deja de estarlo
              tiempo_salida = reloj + generaservicio(tserv); //determinamos cuando
                                          //saldrá ese cliente que acaba de llegar
              ocio += reloj - inicio_ocio; //acumulamos el ocio hasta este momento
             }
        else { //el servidor está ocupado
              acum_cola += (reloj - tultsuc) * encola; //acumulamos el número de
                      //clientes en cola por el tiempo en que han estado en cola
              tultsuc = reloj; //para saber en qué momento cambió el tamaño de la
                               //cola, en este caso aumentó en uno más
              encola ++;  //hay un cliente más en cola
             }
       }
      if (reloj == tiempo_salida) //si estamos en el instante en que se va alguien
       {
        atendidos ++; //se ha atendido a un cliente más
        if (encola > 0) //si quedan clientes en cola
```

```
            {
             acum_cola += (reloj - tultsuc) * encola; // acumulamos el número de
                       //clientes en cola por el tiempo en que han estado en cola
             tultsuc = reloj; //para saber en qué momento cambió el tamaño de la
                              //cola, en este caso disminuyó en uno
             encola  --; //hay un cliente menos en cola
             tiempo_salida = reloj + generaservicio(tserv); //determinamos cuando
                                        //saldrá ese cliente que acaba de entrar
            }
         else { //no quedan clientes en cola
               servidor = libre; //el servidor se queda ocioso por falta de clientes
               inicio_ocio = reloj; //marcamos cuando empieza a estar ocioso
               tiempo_salida = infinito; //nadie puede salir puesto que nadie hay
              }
        }
        reloj ++;   //el tiempo se incrementa en una unidad
       }
porcent_ocio = ocio*100/reloj; //calculamos el % de tiempo de ocio del servidor
printf(porcent_ocio);
media_encola = acum_cola/reloj; //calculamos el número medio de clientes en cola
printf(media_encola);
```

Los generadores de datos mencionados, `generallegada` y `generaservicio`, se pueden implementar del siguiente modo (método de inversión):

float generallegada(`tlleg`)

```
u = (float) random(); // o también rand() en lugar de random()
u = (float) (u/(RAND_MAX+1.0); //RAND_MAX es una constante del sistema
return (-tlleg*log(1-u));
```

float generaservicio: es igual, cambiando `tlleg` por `tserv`.

En nuestro caso, y dado que vamos a mover el tiempo en unidades enteras, esos generadores hay que modificarlos ligeramente. En primer lugar, hay que redondear la salida (que es un float) al entero más próximo; también, para evitar problemas con el manejo del tiempo (y no dejarnos cosas perdidas en el pasado), si el valor devuelto por el generador es 0, debe cambiarse por un 1 (pensad por qué); por último, hay que decidir la unidad de tiempo que vamos a emplear: si se trata de horas, entonces `tlleg`=0.15 y `tserv`=0.1; si se trata de minutos, entonces `tlleg`=9 (0.15*60) y `tserv`=6 (0.1*60), y así sucesivamente.

1.2.2. Tareas a realizar

Construid un programa que implemente este modelo. Ejecutadlo, para un número de clientes a atender elevado (p.e. 10000), y empleando diferentes unidades de medida de tiempo (p.e. horas, medias horas, cuartos de horas, minutos, segundos, décimas de segundo, décimas de hora,

centésimas de hora, décimas de minuto,...). Repetid varias veces ¿Qué observais en relación a los valores devueltos por las diferentes simulaciones? ¿Son los resultados obtenidos empleando diferentes unidades de tiempo coherentes entre sí?

1.2.3. Pseudocódigo del programa de simulación con incremento variable de tiempo

Ahora vamos a construir otra versión del modelo, que emplea otra técnica de control del tiempo, denominada *incremento variable de tiempo*, más eficiente y precisa (pero todavía seguimos sin emplear la estructuración apropiada).

En este caso la variable tiempo no tiene que ser entera, puede ser float, y los generadores de datos pueden emplearse sin ninguna modificación. Unicamente hay que tener en cuenta que si modificamos la unidad de medida del tiempo (p.e. minutos en vez de horas), los parámetros `tlleg` y `tserv` deben modificarse en consecuencia.

```
inicializar generadores de datos
infinito = 10e30;
atendidos = 0;
inicio_ocio = 0.0;
acum_cola = 0.0;
reloj = 0.0;
servidor = libre;
encola = 0;
tiempo_llegada = reloj + generallegada(tlleg);
tiempo_salida = infinito;
while (atendidos < total_a_atender) do
     {
      reloj = min(tiempo_llegada,tiempo_salida); //una función que calcula el mínimo
      if (reloj == tiempo_llegada)
       {
        tiempo_llegada = reloj + generallegada(tlleg);
        if (servidor == libre)
             {
              servidor = ocupado;
              tiempo_salida = reloj + generaservicio(tserv);
              ocio += reloj - inicio_ocio;
             }
        else {
              acum_cola += (reloj - tultsuc) * encola;
              tultsuc = reloj;
              encola ++;
             }
       }
      if (reloj == tiempo_salida)
       {
```

```
        atendidos ++;
        if (encola > 0)
             {
              acum_cola += (reloj - tultsuc) * encola;
              tultsuc = reloj;
              encola  --;
              tiempo_salida = reloj + generaservicio(tserv);
             }
        else {
              servidor = libre;
              inicio_ocio = reloj;
              tiempo_salida = infinito;
             }
       }
      }
porcent_ocio = ocio*100/reloj;
printf(porcent_ocio);
media_encola = acum_cola / reloj;
printf(media_encola);
```

Se puede observar que el cambio con respecto al modelo desarrollado anteriormente es muy pequeño, en este caso afecta únicamente a dos líneas de código (pero esto representa una diferencia importantísima).

1.2.4. Tareas a realizar

Construid un programa que implemente este modelo. Comparad la eficiencia de los métodos de incremento fijo y variable midiendo los tiempos de ejecución de ambos modelos de simulación, para un número de clientes atendidos elevado, usando diferentes unidades de tiempo (horas, minutos, segundos, décimas de segundo). ¿Qué conclusiones se pueden extraer?

Comparad también los resultados obtenidos mediante incremento fijo y variable en cuanto a precisión, para las diferentes elecciones de la unidad de tiempo del reloj. Para ello, realizar simulaciones de cada sistema, con un número elevado de clientes atendidos. Se pueden comparar los resultados obtenidos con valores teóricos (siempre que `tserv`<`tlleg`). Concretamente, si $\rho = \frac{\texttt{tserv}}{\texttt{tlleg}}$, entonces $Q(n) \to \frac{\rho^2}{1-\rho}$, y $PTO(n) \to 100 * (1 - \rho)$, cuando $n \to +\infty$. ¿Se obtiene alguna conclusión sobre la precisión de los dos métodos?

1.3. Estructura de un programa de simulación dinámico y discreto

Ahora vamos a modelizar un sistema en el que hay m servidores idénticos en vez de uno solo, trabajando en paralelo (sigue habiendo una sola cola; en este caso cuando un servidor termina de atender a un cliente, el primer cliente que haya en la cola pasa a ser atendido por ese servidor), utilizando el método de incremento variable de tiempo y la estructuración típica de un modelo de simulación. Esto implica manejar una *lista de sucesos* para el control del tiempo, y disponer de procedimientos de *inicialización*, *temporización*, generación de *informes* y de gestión de cada tipo de *suceso*.

Además, ampliaremos el modelo para incluir también el cálculo de otras medidas de rendimiento de interés. Concretamente, además del *número medio de clientes en cola* y del *porcentaje medio de tiempo de ocio de los servidores*, calcularemos también: el *tiempo medio de espera en cola*, el *tiempo medio de estancia en el sistema*, el *número medio de personas en el sistema*, la *longitud media de las colas no vacias* y la *longitud máxima de la cola*. Por último, la forma de detener la simulación también será diferente: en vez de parar cuando se haya atendido cierto número de clientes, pararemos cuando haya transcurrido cierta cantidad de tiempo.

1.3.1. Pseudocódigo del programa correctamente estructurado

Programa principal

```
inicializacion;
while not (parar) do
 {
  suc_sig = temporizacion;
  suceso(suc_sig);
 }
```

Procedimiento inicialización

```
inicializar generador de numeros pseudoaleatorios;
reloj = 0.0;
libres = m;
encola = 0;
ensistema = 0;
atendidos = 0;
acum_cola = 0.0;
acum_sistema = 0.0;
acum_ocio = 0.0;
acum_retraso = 0.0;
tultsuc_cola = reloj;
tultsuc_ocio = reloj;
tultsuc_sistema = reloj;
acum_sincola = 0.0;
```

```
init_sincola = reloj;
maximacola = 0;
crear(cola); //cola es una estructura de datos que almacena los tiempos de llegada
             //de los clientes que deben esperar; sirve para poder calcular los
             //tiempos de espera; inicialmente estará vacía
crear(lsuc); // crea la lista de sucesos, inicialmente vacía. Contiene un registro
             //(nodo) por cada suceso pendiente, indicando su tipo (nodo.suceso) y
             //su tiempo de ocurrencia (nodo.tiempo); también incluirá, para los
             //sucesos de salida, el tiempo en cola del cliente (nodo.retraso).
             //la lista está ordenada de forma creciente en función del tiempo.
insertar(lsuc,suceso_llegada,reloj+generallegada);
insertar(lsuc,suceso_finsimulacion,reloj+tparada); //tparada es un parámetro de
                                                   //entrada al programa
parar = false;
```

Procedimiento temporizacion;

```
recuperar(lsuc,nodo);//extrae y borra de la lista de sucesos el nodo con menor tiempo
suc_sig = nodo.suceso;
reloj = nodo.tiempo;
return (suc_sig);
```

Procedimiento suceso(`suc_sig`)

```
switch(suc_sig) {
                 case suceso_llegada: llegada; break;
                 case suceso_salida: salida; break;
                 case suceso_finsimulacion: fin; break;
                }
```

Procedimiento llegada

```
acum_sistema += (reloj - tultsuc_sistema) * ensistema;
tultsuc_sistema = reloj;
ensistema ++;
insertar(lsuc,suceso_llegada,reloj+generallegada);
if (libres > 0)
     {
      acum_ocio += (reloj - tultsuc_ocio) * libres;
      tultsuc_ocio = reloj;
      libres --;
      insertar(lsuc,suceso_salida,reloj+generaservicio,0.0);
     }
else {
      if (encola == 0) acum_sincola += reloj - init_sincola;
      acum_cola += (reloj - tultsuc_cola) * encola;
      tultsuc_cola = reloj;
      encola ++;
      if (encola > maximacola) maximacola = encola;
      insertar(cola,reloj);
      }
```

Procedimiento salida

```
acum_sistema += (reloj - tultsuc_sistema) * ensistema;
tultsuc_sistema = reloj;
ensistema --;
atendidos ++;
acum_retraso += nodo.retraso;
if (encola > 0)
     {
      acum_cola += (reloj - tultsuc_cola) * encola;
      tultsuc_cola = reloj;
      encola --;
      if (encola == 0) init_sincola = reloj;
      recuperar(cola,valor); //extrae y elimina el primer elemento de la cola
      insertar(lsuc,suceso_salida,reloj+generaservicio,reloj-valor);
     }
else {
      acum_ocio += (reloj - tultsuc_ocio) * libres;
      tultsuc_ocio = reloj;
      libres ++;
     }
```

Procedimiento fin (equivale al generador de informes)

```
parar = true; //para detener la simulación
              //habrá que hacer las últimas actualizaciones de algunas variables
retrasomedio = acum_retraso/atendidos;
printf("Tiempo medio de espera en cola",retrasomedio);
estanciamedia = retrasomedio + tserv;
printf("Tiempo medio de estancia en el sistema",estanciamedia);
acum_cola += (reloj - tultsuc_cola) * encola;
encolamedio = acum_cola/reloj;
printf("Numero medio de personas en cola",encolamedio);
acum_sistema += (reloj - tultsuc_sistema) * ensistema;
ensistemamedio = acum_sistema/reloj;
printf("Numero medio de personas en el sistema",ensistemamedio);
if (encola == 0) acum_sincola += reloj - init_sincola;
colasnovaciasmedio = acum_cola/(reloj - acum_sincola);
printf("Longitud media de colas no vacias",colasnovaciasmedio);
acum_ocio += (reloj - tultsuc_ocio) * libres;
porcentajemedioocio = 100*acum_ocio/(m*reloj);
printf("porcentaje medio de tiempo de ocio por servidor",porcentajemedioocio);
printf("Longitud máxima de la cola",maximacola);
```

1.3.2. Tareas a realizar

El programa *colammk* implementa el modelo de cola con m servidores descrito anteriormente. Para el caso en que $m = 1$, es posible obtener valores teóricos (cuando el tiempo tiende a infinito) de casi todas las medidas de rendimiento calculadas, lo que es útil a la hora de verificar el programa. Concretamente, los valores teóricos son los siguientes:

- Tiempo medio de espera en cola: $\frac{tserv^2}{tlleg-tserv}$
- Tiempo medio de estancia en el sistema: $\frac{tserv*tlleg}{tlleg-tserv}$
- Número medio de clientes en cola: $\frac{tserv^2}{tlleg*(tlleg-tserv)}$
- Número medio de clientes en el sistema: $\frac{tserv}{tlleg-tserv}$
- Longitud media de colas no vacías: $\frac{tlleg}{tlleg-tserv}$
- Porcentaje de tiempo de ocio del servidor: $(1 - \frac{tserv}{tlleg}) * 100$

Probad el programa varias veces, para un tiempo de simulación progresivamente mayor, y (para el caso $m = 1$) contrastad los resultados obtenidos con los valores teóricos antes expuestos. ¿Qué conclusiones se obtienen?

Una vez convencidos de que el programa funciona correctamente, investigad empíricamente los efectos de aumentar el número de servidores pero haciendo que sean más lentos (de forma que se mantenga un equilibrio, es decir, que el tiempo de servicio dividido por el número de

servidores permanezca constante; es decir, un único servidor con tiempo de servicio `tserv` y m servidores cada uno con tiempo de servicio `m*tserv`).

Modificad el programa para que pueda repetir varias veces la simulación, y calcule las medias y las desviaciones típicas de las medidas de rendimiento. Para ello, téngase en cuenta que si la media y la desviación típica se expresan como

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

una expresión equivalente para σ, más cómoda para el cálculo, es

$$\sigma = \sqrt{\frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2\right)}$$

Tened especial cuidado en que las simulaciones sean independientes entre sí (aseguraos de que cuando se empieza una simulación, las variables y estructuras de datos se inicializan correctamente). La inicialización del generador de datos básico (el rand o random) debe de hacerse una sola vez (de forma que esta instrucción no debe aparecer en la rutina de inicialización, sino en el programa principal).

Estudiad qué ocurre si reemplazais todos los generadores de datos exponenciales por generadores determinísticos que siempre devuelven los correspondientes valores medios, o por generadores uniformes (con la misma media que los anteriores)[1]

[1]El objetivo es comprobar que no basta con tener información sobre comportamiento medio, la existencia de aleatoriedad y el tipo de distribución también pueden ser muy importantes. Los resultados pueden ser diferentes manteniendo los valores medios pero cambiando las distribuciones.

Capítulo 2

Mi Tercer Modelo de Simulación Discreto

2.1. Planteamiento del sistema a estudiar: Remolcador de un puerto

Un puerto está dedicado a la carga de petroleros para el transporte marítimo de crudo. El puerto tiene capacidad para cargar simultáneamente hasta 3 petroleros (hay tres puntos de atraque). Los petroleros, que llegan al puerto cada 11 ± 7 horas, son de diferentes tipos (todos los tiempos que se expresan como rangos se suponen uniformemente distribuidos en ese rango). La frecuencia relativa de los diferentes tipos, así como sus tiempos de carga son los siguientes:

Tipo	Frec. relativa	Tiempo de carga (horas)
1	0.25	18 ± 2
2	0.25	24 ± 3
3	0.50	36 ± 4

Hay un único remolcador en el puerto. Todos los petroleros necesitan los servicios del remolcador para moverse desde que llegan a la bocana del puerto hasta un punto de atraque, y después para desatracar y salir del puerto. Cuando el remolcador está disponible, toda actividad de atraque o desatraque dura $1 \pm 0{,}25$ horas. El remolcador tarda 0.25 horas en moverse desde la bocana del puerto hasta los puntos de atraque o viceversa, siempre que no esté remolcando a un petrolero. Cuando el remolcador termina una actividad de atraque, desatracará el primer petrolero que haya terminado ya de ser cargado, si existe alguno. En caso contrario, si hay barcos esperando en la bocana del puerto para atracar, entonces el remolcador se desplaza hasta la bocana y comienza el proceso de atraque del primer petrolero que estuviese esperando. Cuando el remolcador finaliza una actividad de desatraque, comenzará a atracar al primer petrolero de la cola del puerto, si esta cola no está vacía. En caso contrario, el remolcador se desplaza hasta los puntos de atraque y si existen barcos ya cargados, comenzará a desatracar el primero de ellos. Si tampoco existen barcos cargados, entonces el remolcador permanecerá desocupado junto a los puntos de atraque.

La situación se complica porque en esta zona se producen tormentas frecuentes, que duran 4 ± 2 horas. El tiempo desde que termina una tormenta hasta que empieza la siguiente se distri-

buye exponencialmente con media de 48 horas. El remolcador no comenzará ninguna actividad cuando haya una tormenta, pero siempre terminará la actividad que estuviese realizando, con una excepción. Si el remolcador está viajando desde los puntos de atraque hasta la bocana del puerto sin remolcar un petrolero cuando empieza una tormenta, volverá a los puntos de atraque hasta que pase la tormenta (invirtiendo en volver el mismo tiempo que tardó en ir).

Se pretende construir un modelo de simulación de este sistema para un año (8760 horas), para estimar:

(a) las proporciones de tiempo que el remolcador está desocupado (amarrado), desplazándose sin remolcar un petrolero (en cualquier sentido), y remolcando un petrolero (atracando o desatracando),

(b) las proporciones de tiempo que los puntos de atraque están desocupados, ocupados pero sin estar cargando, y cargando,

(c) el número medio de petroleros en las "colas" de atraque y de desatraque,

(d) el tiempo medio de estancia en el puerto de cada tipo de barco (desde que llega a la bocana del puerto hasta que se va del puerto cargado).

2.2. Sucesos y grafo de sucesos

Inicialmente podemos considerar los siguientes sucesos:

- Llegada de barcos (*llegada_barco*): llega un nuevo barco a la bocana del puerto.
- Comienzo de la operación de atraque (*comienzo_atraque*): el remolcador comienza a remolcar a un barco hacia los puntos de atraque.
- Fin de la operación de atraque (*fin-atraque*): el remolcador llega con el barco hasta los puntos de atraque.
- Comienzo de la operación de carga (*comienzo_carga*): el barco comienza a ser cargado en algún punto de atraque.
- Fin de la operación de carga (*fin_carga*): termina de cargarse el barco, permaneciendo en el punto de atraque.
- Comienzo de la operación de desatraque (*comienzo_desatraque*): el remolcador comienza a remolcar el barco hacia la bocana del puerto.
- Fin de la operación de desatraque (*fin_desatraque*): el remolcador llega con el barco a la bocana del puerto y el barco se va.
- Comienzo del desplazamiento del remolcador desde los puntos de atraque hasta la bocana del puerto (*comienzo_viaje_at_bo*): el remolcador empieza a moverse vacío hacia la bocana del puerto para ir a remolcar un barco.

- Fin del desplazamiento del remolcador desde los puntos de atraque a la bocana del puerto (*fin_viaje_at_bo*): el remolcador llega vacío a la bocana del puerto.
- Comienzo del desplazamiento del remolcador desde la bocana del puerto hasta los puntos de atraque (*comienzo_viaje_bo_at*): el remolcador empieza a moverse vacío hacia los puntos de atraque, para desatracar un barco o quedar amarrado.
- Fin del desplazamiento del remolcador desde la bocana del puerto hasta los puntos de atraque (*fin_viaje_bo_at*): el remolcador llega vacío a los puntos de atraque.
- Comienzo de tormenta (*comienzo_tormenta*): se desencadena una tormenta.
- Fin de tormenta (*fin_tormenta*): termina la tormenta y se pueden reanudar las actividades del remolcador.
- Fin de simulación (*fin-simulación*): suceso ficticio.

El grafo de sucesos sería el representado en la figura 2.1, donde T_ll es el tiempo entre llegadas sucesivas de barcos, T_v es el tiempo de atraque o desatraque, T_c es el tiempo de carga, T_t es el tiempo de duración de la tormenta, T_s es el tiempo para la siguiente tormenta y T_m es el tiempo que tarda en volver el remolcador a los puntos de atraque cuando iba en sentido contrario. Las condiciones de los arcos condicionales son las siguientes:

(1) `tormenta = false, remolcador = libre, atraques_libres >0`

(2) `tormenta = false, encola_sal >0`

(3) `tormenta = false, encola_sal = 0, encola_lleg >0, atraques_libres >0`

(4) `tormenta = false, remolcador = libre`

(5) `tormenta = false, encola_lleg >0`

(6) `tormenta = true OR encola_lleg = 0`

(7) `tormenta = false, encola_sal >0`

(8) `remolcador = libre, encola_sal >0`

(9) `remolcador = libre, encola_sal = 0, encola_lleg >0, atraques_libres >0`

(10) existe un suceso `fin_viaje_at_bo` en `lsuc`

Nota importante: para modelizar este sistema es necesario eliminar sucesos previamente insertados de la lista de sucesos (cuando se produce una tormenta y el remolcador se encontraba viajando vacío hacia la bocana, debe dar la vuelta, por lo que es necesario "desprogramar" el suceso de `fin_viaje_at_bo` e insertar en su lugar uno de `fin_viaje_bo_at`. En el grafo de sucesos eso se representa trazando un arco de diferente significado (eliminar en vez de insertar, dibujado con trazo discontinuo) desde el suceso que ocasiona la eliminación hasta el suceso que se debe eliminar.

Si reducimos el grafo al mínimo, tenemos los siguientes sucesos:

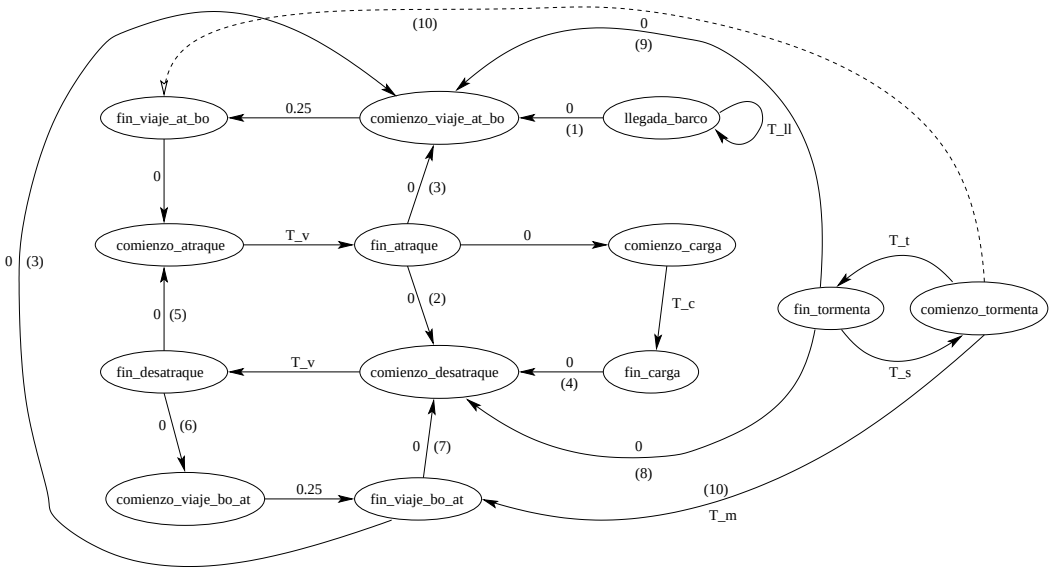

Figura 2.1: Grafo de sucesos inicial

- Llegada de barcos (*llegada_barco*).
- Fin de la operación de atraque (*fin_atraque*).
- Fin de la operación de carga (*fin_carga*).
- Fin de la operación de desatraque (*fin_desatraque*).
- Fin del desplazamiento del remolcador desde los puntos de atraque a la bocana del puerto (*fin_viaje_at_bo*).
- Fin del desplazamiento del remolcador desde la bocana del puerto hasta los puntos de atraque (*fin_viaje_bo_at*).
- Comienzo de tormenta (*comienzo_tormenta*).
- Fin de tormenta (*fin_tormenta*).
- Fin de simulación (*fin_simulación*).

El grafo correspondiente se muestra en la figura 2.2. Las condiciones de los arcos son las mismas de antes. Ya se ha incluido aquí el suceso de fin de simulación y el "suceso" de inicio.

2.3. Variables y medidas de rendimiento

2.3.1. Parámetros de entrada

- `num_remolcadores = 1`: número de remolcadores existentes.

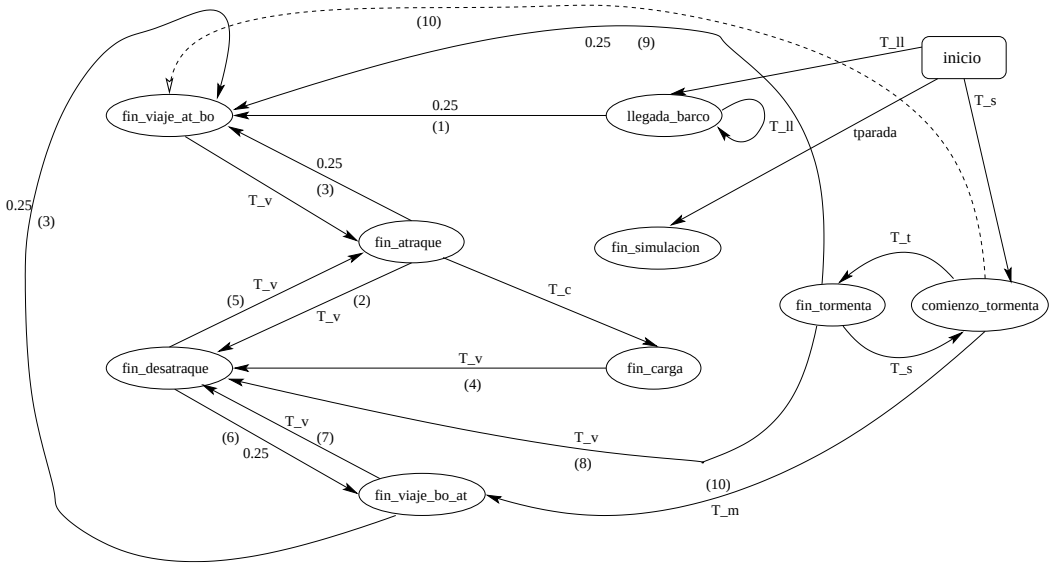

Figura 2.2: Grafo de sucesos reducido

- `num_atraques = 3`: número de puntos de atraque.
- `tllegmin, tllegmax = 4, 18`: tiempo mínimo y máximo entre llegadas consecutivas de barcos al puerto (distribución uniforme).
- `tviajevacio = 0.25`: tiempo de desplazamiento del remolcador de la bocana a los atraques y viceversa, cuando no está remolcando.
- `tviajellenomin, tviajellenomax = 0.75, 1.25`: tiempo mínimo y máximo de desplazamiento del remolcador cuando está remolcando (tiempos de atraque y desatraque, distribución uniforme).
- `dur_tormentamin, dur_tormentamax = 2, 6`: duración mínima y máxima de las tormentas (distribución uniforme).
- `tentre_tormentas = 48`: tiempo medio entre que termina una tormenta y comienza otra (distribución exponencial).
- `num_tiposbarco = 3`: Número de clases de petroleros diferentes.
- `frec1, frec2, frec3 = 0.25, 0.25, 0.50`: probabilidades de los diferentes tipos de barco.
- `tiempo_carga1min, tiempo_carga2min, tiempo_carga3min = 16, 21, 32`: tiempos mínimos de carga de cada tipo de barco.
- `tiempo_carga1max, tiempo_carga2max, tiempo_carga3max = 20, 27, 40`: tiempos máximos de carga de cada tipo de barco.
- `tparada = 8760`: duración de la simulación.

2.3.2. Variables de estado

- `reloj`: reloj de simulación.
- `tormenta`: indica si hay una tormenta (`true`) o no (`false`).
- `remolcador`: indica si el remolcador está amarrado sin hacer nada (`libre`) o haciendo algo (`ocupado`).
- `atraques_libres`: número de puntos de atraque libres.
- `encola_lleg`: número de barcos en la bocana del puerto esperando ser remolcados a los puntos de atraque.
- `cola_llegadas`: estructura donde se almacena el tiempo de llegada de cada barco a puerto y su tipo.
- `encola_sal`: número de barcos en los puntos de atraque, ya cargados, esperando ser remolcados a la bocana del puerto.
- `cola_salidas`: estructura que almacena el tiempo de llegada a puerto de cada barco y su tipo, para los barcos que están atracados y ya han terminado de cargarse.
- `lsuc`: lista de sucesos, que contiene registros con cuatro campos: tipo de suceso y tiempo de ocurrencia; los sucesos que involucran a un barco (fin de atraque, fin de carga y fin de desatraque) incluyen también el tiempo de llegada del barco a puerto, y el tipo de barco. Las rutinas de sucesos están pensadas para una implementación dinámica de esta lista, donde al extraer un suceso de la lista, éste se elimina de la misma. Si se utiliza otra clase de implementación, hay que modificar ligeramente las rutinas de sucesos.
- `tdus_at`: tiempo del último suceso en que algún punto de atraque modifica su situación (que puede ser libre, ocupado cargando un barco u ocupado por un barco ya cargado). Obsérvese que no es necesario mantener de forma explícita la situación de cada punto de atraque, basta con conocer cuántos están libres (`atraques_libres`) y cuántos están ocupados con barcos ya cargados (`encola_sal`), y el resto hasta `num_atraques` serán los que estén ocupados cargando.
- `tdus_lleg`: tiempo del último suceso que modifica el número de barcos en la cola de llegadas.
- `tdus_sal`: tiempo del último suceso que modifica el número de barcos en la cola de salidas.
- `tdus_rem`: tiempo del último suceso que cambia el estado del remolcador (que puede ser estar libre, viajando sin remolcar ningún barco y remolcando un barco).

2.3.3. Contadores estadísticos

- `acum_lleg`: acumulador de tiempos de espera en cola de llegadas por número de barcos que esperan.
- `acum_sal`: acumulador de tiempos de espera en cola de salidas por número de barcos que esperan.
- `acum_estancia[i]`: acumulador de tiempos de estancia, para cada tipo de barco.
- `num_barcos[i]`: número de barcos atendidos (que se han ido ya del puerto), para cada tipo.
- `acum_rem_amarrado`: acumulador de tiempo que el remolcador está desocupado.
- `acum_rem_viajando`: acumulador de tiempo que el remolcador está viajando sin remolcar.
- `acum_rem_remolcando`: acumulador de tiempo que el remolcador está remolcando un barco.
- `acum_at_desocupado`: acumulador de tiempo que los puntos de atraque están desocupados.
- `acum_at_cargando`: acumulador de tiempo que los puntos de atraque están ocupados y cargando un barco.
- `acum_at_yacargado`: acumulador de tiempo que los puntos de atraque están ocupados por barcos ya cargados.

2.3.4. Cálculo de medidas de rendimiento

- Número medio de petroleros en la cola de atraque:

  Cada vez que se modifique el número de barcos en la cola de llegadas, `encola_lleg` (esto ocurre con los sucesos de `llegada_barco` y `comienzo_atraque`, y como este último no se emplea, puede ocurrir con los sucesos `fin_viaje_at_bo` y `fin_desatraque`), se suma al acumulador `acum_lleg` la diferencia entre el tiempo actual (`reloj`) y el tiempo del último cambio (`tdus_lleg`). Al terminar la simulación, y hacer una última actualización del acumulador, se divide su valor por el tiempo total simulado (que lo marca el `reloj` en ese instante).

- Número medio de petroleros en la cola de desatraque:

  Cuando se modifique el número de barcos en cola de salida (los que están en los puntos de atraque pero ya han sido cargados), `encola_sal` (esto ocurre con los sucesos `fin_carga` y `comienzo_desatraque`, y como este último no se emplea, puede ocurrir con los sucesos `fin_atraque`, `fin_viaje_bo_at` y `fin_tormenta`), se suma al acumulador `acum_sal` la diferencia entre el tiempo actual (`reloj`) y el tiempo del último cambio (`tdus_sal`). Al terminar la simulación, y hacer una última actualización del acumulador, se divide su valor por el tiempo total simulado (que lo marca el `reloj` en ese instante).

- Tiempo medio de estancia en el puerto de cada tipo de barco:

  Cuando llega un barco (ocurre el suceso `llegada_barco`) se almacena en la cola de llegadas (`cola_llegadas`) el tiempo en que llega y el tipo de barco, y cuando el barco se va (suceso `fin_desatraque`) se calcula la diferencia entre el tiempo actual (`reloj`) y el tiempo de llegada, se acumula en `acum_estancia[tipo]`, y se aumenta en uno el número de barcos que han salido (`num_barcos[tipo]`). Al terminar la simulación `acum_estancia[tipo]` se divide entre `num_barcos[tipo]`. La información sobre el tiempo de llegada debe conservarse hasta la salida del barco, por lo que cuando se extrae de la cola de llegadas se almacena en la lista de sucesos (sucesivamente en los sucesos `fin_atraque` y `fin_carga`), luego en la cola de salidas (`cola_salidas`) y finalmente en el suceso `fin_desatraque`.

- Proporciones de tiempo que el remolcador está desocupado, desplazándose sin remolcar un petrolero, y remolcando un petrolero:

  Cada una de esas tres medidas de rendimiento tiene una variable acumulador asociada `acum_rem_amarrado`, `acum_rem_viajando` y `acum_rem_remolcando`, que se actualizan cada vez que el remolcador cambia de estado (sumándole al acumulador la diferencia entre el tiempo actual (`reloj`) y el tiempo del cambio anterior (`tdus_rem`)). Al finalizar la simulación, tras hacer una última actualización de los acumuladores, se dividen sus valores por el tiempo total, y se multiplican por 100 (para expresarlos en porcentaje).

- Proporciones de tiempo que los puntos de atraque están desocupados, ocupados pero sin estar cargando, y cargando:

  Igual que en el caso anterior, los acumuladores de tiempo en que los puntos de atraque están en esas tres situaciones se actualizan cuando se produce un cambio en alguno de ellos (sumándoles la diferencia entre el tiempo actual (`reloj`) y el tiempo del último cambio (`tdus_at`), multiplicada por el número de puntos de atraque en esa situación). Al final de la simulación y de la última actualización, se dividen sus valores por el tiempo total y por el número de puntos de atraque, y se expresan en porcentaje.

2.4. Rutinas del modelo de simulación.

Programa principal

```
inicializacion;
while not (parar) do
 {
  nodo = temporizacion;
  suceso(nodo);
 }
```

Rutina de Inicializacion

```
inicializar generador de números pseudoaleatorios;
parar = false;
// inicialización de variables de estado
```

```
reloj = 0.0;
tormenta = false;
remolcador = libre;
atraques_libres = num_atraques;
encola_lleg = 0;
crear(cola_llegadas);
encola_sal = 0;
crear(cola_salidas);
tdus_at = 0.0;
tdus_lleg = 0.0;
tdus_sal = 0.0;
tdus_rem = 0.0;
// inicialización de contadores estadísticos
acum_lleg = 0.0;
acum_sal = 0.0;
for (i=0; i<num_tiposbarco; i++) {
     acum_estancia[i] = 0.0;
     num_barcos[i] = 0; }
acum_rem_amarrado = 0.0;
acum_rem_viajando = 0.0;
acum_rem_remolcando = 0.0;
acum_at_desocupado = 0.0;
acum_at_cargando = 0.0;
acum_at_yacargado = 0.0;
// inicialización de la lista de sucesos
crear(lsuc); // crea la lista de sucesos, inicialmente vacía
reg_cola_null.tiempo = 0.0;
reg_cola_null.tipo = num_tiposbarco;//para rellenar los datos inútiles de lsuc
nodo.reg_cola = reg_cola_null;
nodo.suceso = suc_fin_simulacion;
nodo.tiempo = reloj+tparada;
insertar(lsuc,nodo);
nodo.suceso = suc_llegada_barco;
nodo.tiempo = reloj+genera_barco(tllegmin,tllegmax));
insertar(lsuc,nodo);
nodo.suceso = suc_comienzo_tormenta;
nodo.tiempo = reloj+genera_tormenta(tentre_tormentas);
insertar(lsuc,nodo);
```

Rutina de temporizacion

```
recuperar(lsuc,nodo); //extrae (y borra) de la lista el nodo con menor tiempo
reloj = nodo.tiempo;
return(nodo);
```

Rutina de suceso(`nodo`)

```
switch(nodo.suceso) {
                  case suc_llegada_barco: llegada_barco; break;
                  case suc_fin_atraque: fin_atraque; break;
                  case suc_fin_carga: fin_carga; break;
                  case suc_fin_desatraque: fin_desatraque; break;
                  case suc_fin_viaje_at_bo: fin_viaje_at_bo; break;
                  case suc_fin_viaje_bo_at: fin_viaje_bo_at; break;
                  case suc_comienzo_tormenta: comienzo_tormenta; break;
                  case suc_fin_tormenta: fin_tormenta; break;
                  case suc_fin_simulacion: fin_simulacion; break;
                 }
```

Rutina del suceso `llegada_barco`

```
nodo.suceso = suc_llegada_barco;
nodo.tiempo = reloj+genera_barco(tllegmin,tllegmax);
nodo.reg_cola = reg_cola_null;
insertar(lsuc,nodo); //programa la próxima llegada
acum_lleg += (reloj-tdus_lleg)*encola_lleg;
tdus_lleg = reloj;
encola_lleg ++;
tipobarco = genera_tipobarco(); //determina qué tipo de barco acaba de llegar;
reg_cola.tiempo = reloj;
reg_cola.tipo = tipobarco;
insertar(cola_llegadas,reg_cola);
if ((tormenta == false) && (remolcador == libre) && (atraques_libres > 0))
   //el remolcador va por el barco
  {
   acum_rem_amarrado += reloj-tdus_rem;
   tdus_rem = reloj;
   remolcador = ocupado;
   nodo.suceso = suc_fin_viaje_at_bo;
   nodo.tiempo = reloj+tviajevacio;
   nodo.reg_cola = reg_cola_null;
   insertar(lsuc,nodo); //programa la llegada del remolcador a la bocana
  }
```

Rutina del suceso `fin_viaje_at_bo`

```
acum_rem_viajando += reloj-tdus_rem;
tdus_rem = reloj;
acum_lleg += (reloj-tdus_lleg)*encola_lleg;
tdus_lleg = reloj;
encola_lleg --;
recuperar(cola_llegadas,reg_cola);
nodo.suceso = suc_fin_atraque;
```

```
nodo.tiempo = reloj+genera_viajelleno(tviajellenomin,tviajellenomax);
nodo.reg_cola = reg_cola;
insertar(lsuc,nodo); //programa la llegada del remolcador a los puntos de atraque
                     //con un barco
```

Rutina del suceso `fin_atraque`

```
acum_rem_remolcando += reloj-tdus_rem;
tdus_rem = reloj;
acum_at_desocupado += (reloj-tdus_at)*atraques_libres;
acum_at_yacargado += (reloj-tdus_at)*encola_sal;
acum_at_cargando += (reloj-tdus_at)*(num_atraques-atraques_libres-encola_sal);
tdus_at = reloj;
atraques_libres --;
nodo.suceso = suc_fin_carga;
nodo.tiempo = reloj+genera_tiempocarga(nodo.reg_cola.tipo);
insertar(lsuc,nodo);//programa la finalización de la operación de carga
if (tormenta == false)
   if (encola_sal > 0) //se desatracará un barco
      {
       acum_sal += (reloj-tdus_sal)*encola_sal;
       tdus_sal = reloj;
       encola_sal --;
       atraques_libres ++;
       recuperar(cola_salidas,reg_cola);
       nodo.suceso = suc_fin_desatraque;
       nodo.tiempo = reloj+genera_viajelleno(tviajellenomin,tviajellenomax);
       nodo.reg_cola = reg_cola;
       insertar(lsuc,nodo);
      }
   else if ((atraques_libres > 0) && (encola_lleg > 0))
           { //el remolcador va por un barco a la bocana
            nodo.suceso = suc_fin_viaje_at_bo;
            nodo.tiempo = reloj+tviajevacio;
            nodo.reg_cola = reg_cola_null;
            insertar(lsuc,nodo);
           }
        else remolcador = libre;
else remolcador = libre;
```

Rutina del suceso `fin_carga`

```
acum_sal += (reloj-tdus_sal)*encola_sal;
tdus_sal = reloj;
acum_at_desocupado += (reloj-tdus_at)*atraques_libres;
acum_at_yacargado += (reloj-tdus_at)*encola_sal;
```

```
acum_at_cargando += (reloj-tdus_at)*(num_atraques-atraques_libres-encola_sal);
tdus_at = reloj;
if ((remolcador == libre) & (tormenta == false)) //comienza a desatracar el barco
   {
    acum_rem_amarrado += reloj-tdus_rem;
    tdus_rem = reloj;
    remolcador = ocupado;
    atraques_libres ++;
    nodo.suceso = suc_fin_desatraque;
    nodo.tiempo = reloj+genera_viajelleno(tviajellenomin,tviajellenomax);
    insertar(lsuc,nodo);
   }
else {
      insertar(cola_salidas,nodo.reg_cola);
      encola_sal ++;
     }
```

Rutina del suceso `fin_desatraque`

```
acum_rem_remolcando += reloj-tdus_rem;
tdus_rem = reloj;
acum_estancia[nodo.reg_cola.tipo] += reloj-nodo.reg_cola.tiempo;
num_barcos[nodo.reg_cola.tipo] ++;
if ((encola_lleg > 0) && (tormenta == false)) //comienza a atracar un barco
   {
    acum_lleg += (reloj-tdus_lleg)*encola_lleg;
    tdus_lleg = reloj;
    encola_lleg --;
    recuperar(cola_llegadas,reg_cola);
    nodo.suceso = suc_fin_atraque;
    nodo.tiempo = reloj+genera_viajelleno(tviajellenomin,tviajellenomax);
    nodo.reg_cola = reg_cola;
    insertar(lsuc,nodo);
   }
else { //hay tormenta o no hay barcos esperando: el remolcador se vuelve a los
       //puntos de atraque
      nodo.suceso = suc_fin_viaje_bo_at;
      nodo.tiempo = reloj+tviajevacio;
      nodo.reg_cola = reg_cola_null;
      insertar(lsuc,nodo);
     }
```

Rutina del suceso `fin_viaje_bo_at`

```
acum_rem_viajando += reloj-tdus_rem;
tdus_rem = reloj;
```

```
if (tormenta == false)
   if (encola_sal > 0) //comienza a desatracar un barco
      {
       acum_sal += (reloj-tdus_sal)*encola_sal;
       tdus_sal = reloj;
       acum_at_desocupado += (reloj-tdus_at)*atraques_libres;
       acum_at_yacargado += (reloj-tdus_at)*encola_sal;
       acum_at_cargando += (reloj-tdus_at)*
                           (num_atraques-atraques_libres-encola_sal);
       tdus_at = reloj;
       encola_sal --;
       atraques_libres ++;
       recuperar(cola_salidas,reg_cola);
       nodo.suceso = suc_fin_desatraque;
       nodo.tiempo = reloj+genera_viajelleno(tviajellenomin,tviajellenomax);
       nodo.reg_cola = reg_cola;
       insertar(lsuc,nodo);
      }
   else if ((atraques_libres > 0) && (encola_lleg > 0))
           { //el remolcador va por un barco
            nodo.suceso = suc_fin_viaje_at_bo;
            nodo.tiempo = reloj+tviajevacio;
            nodo.reg_cola = reg_cola_null;
            insertar(lsuc,nodo);
           }
        else remolcador = libre;
else remolcador = libre;
```

Rutina del suceso `comienzo_tormenta`

```
tormenta = true;
nodo.suceso = suc_fin_tormenta;
nodo.tiempo = reloj+genera_durtormenta(dur_tormentamin,dur_tormentamax);
nodo.reg_cola = reg_cola_null;
insertar(lsuc,nodo);
busca_suceso(lsuc,suc_fin_viaje_at_bo,nodo);//busca si existe un suceso de ese tipo
                        // si existe lo extrae y si no devuelve null
if (nodo != null)
   {
    tmediavuelta = tviajevacio-nodo.tiempo+reloj;
    nodo.suceso = suc_fin_viaje_bo_at;
    nodo.tiempo = reloj+tmediavuelta;
    nodo.reg_cola = reg_cola_null;
    insertar(lsuc,nodo);
```

```
   }
```

Rutina del suceso `fin_tormenta`

```
tormenta = false;
nodo.suceso = suc_comienzo_tormenta;
nodo.tiempo = reloj+genera_tormenta(tentre_tormentas);
nodo.reg_cola = reg_cola_null;
insertar(lsuc,nodo);
if (remolcador == libre) //con los datos actuales esto siempre ocurrirá
   if (encola_sal > 0) //comienza a desatracar un barco
      {
       acum_sal += (reloj-tdus_sal)*encola_sal;
       tdus_sal = reloj;
       acum_at_desocupado += (reloj-tdus_at)*atraques_libres;
       acum_at_yacargado += (reloj-tdus_at)*encola_sal;
       acum_at_cargando += (reloj-tdus_at)*
                              (num_atraques-atraques_libres-encola_sal);
       tdus_at = reloj;
       encola_sal --;
       atraques_libres ++;
       acum_rem_amarrado += reloj-tdus_rem;
       tdus_rem = reloj;
       remolcador = ocupado;
       recuperar(cola_salidas,reg_cola);
       nodo.suceso = suc_fin_desatraque;
       nodo.tiempo = reloj+genera_viajelleno(tviajellenomin,tviajellenomax);
       nodo.reg_cola = reg_cola;
       insertar(lsuc,nodo);
      }
   else if ((atraques_libres > 0) && (encola_lleg > 0))
           { //el remolcador va por un barco
            acum_rem_amarrado += reloj-tdus_rem;
            tdus_rem = reloj;
            remolcador = ocupado;
            nodo.suceso = suc_fin_viaje_at_bo;
            nodo.tiempo = reloj+tviajevacio;
            nodo.reg_cola = reg_cola_null;
            insertar(lsuc,nodo);
           }
```

Rutina del suceso `fin_simulacion`

```
parar = true;
acum_at_desocupado += (reloj-tdus_at)*atraques_libres;
acum_at_yacargado += (reloj-tdus_at)*encola_sal;
```

```
acum_at_cargando += (reloj-tdus_at)*(num_atraques-atraques_libres-encola_sal);
acum_lleg += (reloj-tdus_lleg)*encola_lleg;
acum_sal += (reloj-tdus_sal)*encola_sal;
if (remolcador == libre) acum_rem_amarrado += reloj-tdus_rem;
else {
      busca_suceso(lsuc,suc_fin_viaje_at_bo,nodo);
      if (nodo != null) acum_rem_viajando += reloj-tdus_rem;
      else {
            busca_suceso(lsuc,suc_fin_viaje_bo_at,nodo);
            if (nodo != null) acum_rem_viajando += reloj-tdus_rem;
            else acum_rem_remolcando += reloj-tdus_rem;
           }
     }
print("Número medio de barcos en cola de llegadas =",acum_lleg/reloj;
print("Número medio de barcos en cola de salidas =",acum_sal/reloj;
for (i=0; i<num_tiposbarco; i++)
   print("Tiempo medio de estancia en puerto =",acum_estancia[i]/num_barcos[i]);
print("Porcentaje de tiempo remolcador desocupado =",
       100*acum_rem_amarrado/reloj);
print("Porcentaje de tiempo remolcador viajando vacío=",
       100*acum_rem_viajando/reloj);
print("Porcentaje de tiempo remolcador remolcando barcos=",
       100*acum_rem_remolcando/reloj);
print("Porcentaje de tiempo puntos de atraque libres=",
       100*acum_at_desocupado/(reloj*num_atraques));
print("Porcentaje de tiempo puntos de atraque ocupados sin cargar=",
       100*acum_at_yacargado/(reloj*num_atraques));
print("Porcentaje de tiempo puntos de atraque ocupados cargando=",
       100*acum_at_cargando/(reloj*num_atraques));
```

Generadores de datos

- Generador de tiempos entre llegadas de barcos al puerto (uniforme)

  `genera_barco(tllegmin,tllegmax)`

- Generador de tipos de barco (discreto)

  `genera_tipobarco()`

- Generador de tormentas (tiempo entre fin de una y comienzo de otra) (exponencial)

  `genera_tormenta(tentre_tormentas)`

- Generador de duraciones de tormentas (uniforme)

  `genera_durtormenta(dur_tormentamin,dur_tormentamax)`

- Generador de tiempos de viaje del remolcador remolcando (uniforme)

  `genera_viajelleno(tviajellenomin,tviajellenomax)`

- Generador de tiempos de carga (uniforme)

  `genera_tiempocarga(tipo)`

generador uniforme(min,max)

```
u = (float) random() // o también rand() en lugar de random()
u = (float) (u/(RAND_MAX+1.0) // o sin sumar 1
return(min+(max-min)*u)
```

Generador exponencial(media)

```
u = (float) random() // o también rand() en lugar de random()
u = (float) (u/(RAND_MAX+1.0) // o sin sumar 1
return(-media*log(1-u))
```

Generador discreto (devuelve el tipo de barco, que es 0, 1 o 2)

```
u = (float) random() // o también rand() en lugar de random()
u = (float) (u/(RAND_MAX+1.0) // o sin sumar 1
if (u < frec1) return(0)
else if (u < (frec1+frec2)) return(1)
     else return(2)
```

2.5. Tareas a realizar

El programa *puerto* implementa el modelo de simulación del sistema del puerto con remolcador antes descrito. Investigad las prestaciones del sistema en su configuración actual. Realizad varias simulaciones y calculad valores medios y desviaciones típicas de todas las medidas de rendimiento.

Imaginemos que los responsables del puerto pretenden mejorar el sistema, y se plantean las siguientes alternativas:

- Aumentar los puntos de atraque, pasando de 3 a 4 o a 5 puntos de atraque.
- Cambiar el remolcador por otro de iguales características, salvo que no le afectan las tormentas, y puede continuar operando incluso cuando haya tormenta.
- Cambiar el remolcador por uno algo más rápido, que puede viajar (sin remolcar un barco) de la bocana a los atraques y viceversa tardando 0.15 horas en lugar de 0.25.

Realizad las modificaciones necesarias para poder experimentar con el sistema original y con sus alternativas con objeto de recomendar las mejores opciones.

Supongamos ahora que el número de toneladas que cargan los petroleros es 1000, 2000 y 3000 para los tipos 1, 2 y 3 respectivamente. Incluid en el modelo esta información y una nueva medida de rendimiento que sea el total de toneladas cargadas. Investigad cuál de las alternativas anteriores es preferible desde el punto de vista de esta nueva medida de rendimiento.

Capítulo 3

Análisis de Salidas y Experimentación

Para el modelo de simulación del puerto con remolcador, desarrollado en el capítulo anterior, vamos a realizar diversos tipos de análisis de salidas y experimentación (la condición de parada es simular 365 días).

3.1. ¿Cuánto hay que simular?

Se pretende analizar y comparar las siguientes dos configuraciones alternativas del sistema, desde el punto de vista del número medio de barcos en la cola de atraque, $NBCA$:

(A) Modelo original.

(B) Modelo en el que el remolcador puede operar incluso cuando hay tormentas.

Utilizad el programa del capítulo anterior para hacer lo siguiente:

- Haced una simulación de cada sistema, obtened los valores estimados, $NBCA_A$ y $NBCA_B$ para cada sistema, y señalad como preferible el sistema que produzca un valor de $NBCA$ menor. Repetid 100 veces este proceso, obteniendo 100 valores de $NBCA_A$ y $NBCA_B$, y el porcentaje de veces en que el sistema (A) es preferible al (B) y viceversa. Esos porcentajes representan un estimador de la probabilidad de, en cada caso, tomar la decisión equivocada al decidir que un sistema es mejor que el otro, sobre la base de hacer *una única* simulación.

- Haced ahora cinco simulaciones de cada sistema, y obtened los valores estimados de $NBCA_A$ y $NBCA_B$ como la media de los resultados de las cinco simulaciones de cada sistema. Repetid este proceso 100 veces (o sea, haciendo un total de 500 simulaciones de cada sistema), obteniendo de nuevo 100 valores de $NBCA_A$ y $NBCA_B$, y el porcentaje de veces en que el sistema (A) es preferible al (B) y viceversa. Los porcentajes ahora representan un estimador de la probabilidad de, en cada caso, tomar la decisión equivocada al decidir que un sistema es mejor que el otro, sobre la base de hacer *cinco* simulaciones.

- Reiterad este proceso, con 10, 25, 50 simulaciones (y, si es factible, con 100, 500,... simulaciones). ¿Cómo varían las probabilidades de equivocarse en función del número de simulaciones?

Repetid todo el proceso anterior, pero ahora compararemos el sistema (A) con el sistema (C):

(C) Modelo con el remolcador algo más rápido.

¿Qué conclusiones se pueden extraer?

3.2. Intervalos de confianza

- Utilizad alguna de las técnicas de comparación de dos sistemas alternativos mediante intervalos de confianza para decidir si resulta preferible un remolcador más rápido o uno al que no le afecten las tormentas (desde el punto de vista del número medio de barcos en la cola de atraque).

3.3. Comparación de más de dos sistemas

Utilizad una técnica de elección del mejor de entre un conjunto de $k = 4$ sistemas para decidir cuál de las cuatro configuraciones siguientes es preferible:

- Tener cuatro puntos de atraque.
- Tener cinco puntos de atraque.
- Tener un remolcador más rápido.
- Tener un remolcador al que no le afectan las tormentas

La medida de rendimiento para realizar esta comparación sigue siendo el número medio de barcos en la cola de atraque.