SIMULACIÓN DE SISTEMAS

Práctica 3:

Modelos de Simulación Dinámicos y Discretos

Curso 2020/2021

Capítulo 1

Mi Segundo Modelo de Simulación Discreto

1.1. Planteamiento del Sistema a Estudiar: Máquinas y Reparadores

Un sistema necesita que `n` máquinas estén funcionando. Para evitar fallos, se dispone de algunas máquinas como repuestos. Siempre que una máquina se estropea, de inmediato se reemplaza por una de repuesto, si la hay, y se envía al taller, en el cual `m` reparadores (de características idénticas, trabajando en paralelo) reparan las máquinas. Una vez reparada una máquina, queda disponible como repuesto para cuando surja la necesidad si las `n` máquinas están funcionando. Todos los tiempos de reparación son variables aleatorias independientes, con distribución exponencial de media `trepar`. Cada vez que una máquina comienza a trabajar, el tiempo que funciona hasta que se estropea es una variable aleatoria independiente de las anteriores, con distribución exponencial de media `tfallo`. El sistema "falla" cuando una máquina se descompone y no hay repuestos disponibles. Cuando ocurre esta situación, las restantes máquinas continuan funcionando, y cuando una máquina es reparada se pone a funcionar inmediatamente. Al principio hay `n+s` máquinas en buen estado, de las cuales `n` se ponen a trabajar y `s` quedan como repuestos. Se pretende obtener información sobre el tiempo medio entre fallos del sistema, la duración media de los fallos, el número medio de máquinas en reparación, el porcentaje de tiempo de ocio de los reparadores y el porcentaje de duración total de los fallos. Las condiciones iniciales son que todas las máquinas están operativas.

1.2. Gestión del Tiempo. Métodos de Incremento Fijo e Incremento Variable de Tiempo

Consideremos el sistema anterior en una versión simplificada, en la que $n = m = 1$ y $s = 0$ o $s = 1$, es decir, una sola máquina debe funcionar, hay un sólo reparador y no hay repuestos o hay un único repuesto. Además, sólo estamos interesados en obtener información sobre la duración media de los fallos del sistema.

1.2.1. Incremento Variable de Tiempo

```
inicializar generador de números pseudoaleatorios;
reloj = 0.0;
hayrepuesto = true (si s=1) o false (si s=0);
fallo = false;
reparadorlibre = true;
maquinaesperando = false;
numfallos = 0;
durfallos = 0.0;
tiempofallomaquina = reloj + generafallo(tfallo);
tiempofinreparacion = 10e30;
while (reloj <= tiempodeparar) do {
     reloj = min(tiempofallomaquina,tiempofinreparacion);
     if (reloj == tiempofallomaquina)
       {
        if (reparadorlibre == true) {
          reparadorlibre = false;
          tiempofinreparacion = reloj + generareparacion(trepar); }
        else maquinaesperando = true;
        if (hayrepuesto == true) {
          hayrepuesto = false;
          tiempofallomaquina = reloj + generafallo(tfallo); }
        else  if (fallo == false) {
                fallo = true;
                comienzofallo = reloj;
                numfallos ++;
                tiempofallomaquina = 10e30; }
       }
     if (reloj == tiempofinreparacion)
       {
        if (maquinaesperando == true) {
          maquinaesperando = false;
          tiempofinreparacion = reloj + generareparacion(trepar); }
        else {
              reparadorlibre = true;
              tiempofinreparacion = 10e30; }
        if (fallo == false) hayrepuesto = true;
        else {
              tiempofallomaquina = reloj + generafallo(tfallo);
              durfallos += reloj - comienzofallo;
              fallo = false; }
        }
   }
if (fallo == true) durfallos += reloj - comienzofallo;
```

```
printf("\nDuracion media de los fallos = %f",durfallos/numfallos);
```

1.2.2. Incremento Fijo de Tiempo

```
inicializar generador de números pseudoaleatorios;
reloj = 0.0;
hayrepuesto = true (si s=1) o false (si s=0);
fallo = false;
reparadorlibre = true;
maquinaesperando = false;
numfallos = 0;
durfallos = 0.0;
tiempofallomaquina = reloj + generafallo(tfallo);
tiempofinreparacion = 10e30;
while (reloj <= tiempodeparar) do {
     if (reloj == tiempofallomaquina)
       {
        if (reparadorlibre == true) {
          reparadorlibre = false;
          tiempofinreparacion = reloj + generareparacion(trepar); }
        else maquinaesperando = true;
        if (hayrepuesto == true) {
          hayrepuesto = false;
          tiempofallomaquina = reloj + generafallo(tfallo); }
        else  if (fallo == false) {
                fallo = true;
                comienzofallo = reloj;
                numfallos ++;
                tiempofallomaquina = 10e30; }
       }
     if (reloj == tiempofinreparacion)
       {
        if (maquinaesperando == true) {
          maquinaesperando = false;
          tiempofinreparacion = reloj + generareparacion(trepar); }
        else {
              reparadorlibre = true;
              tiempofinreparacion = 10e30; }
        if (fallo == false) hayrepuesto = true;
        else {
              tiempofallomaquina = reloj + generafallo(tfallo);
              durfallos += reloj - comienzofallo;
              fallo = false; }
        }
     reloj ++
```

```
   }
if (fallo == true) durfallos += reloj - comienzofallo;
printf("\nDuracion media de los fallos = %f",durfallos/numfallos);
```

1.2.3. Generadores de Datos

- Para el caso de incremento variable, los generadores de datos exponenciales, generareparacion(`trepar`) y generafallo(`tfallo`), se pueden implementar fácilmente por el método de inversión ($-media * \log(1-u)$, donde u es un número uniformemente distribuido entre 0 y 1):

  float generaloquesea(`media`)
  ```
  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 (-media*log(1-u));
  ```

  Los valores de `trepar` y `tfallo` pueden expresarse en la unidad de tiempo que se desee, por ejemplo en días u horas (sin que esto afecte realmente al resultado aunque, lógicamente, el resultado vendrá expresado en esa unidad de tiempo).

- En el caso de incremento fijo, los generadores de datos tienen que devolver valores discretos (enteros) (el incremento de tiempo es de una unidad, pudiendo obviamente esa unidad de tiempo ser lo que uno desee, a nivel lógico). Por ejemplo, si la unidad de medida del reloj es el día, se puede simplemente redondear el valor de los generadores empleados anteriormente al entero más próximo[1]. Si se desea otra unidad de medida para el reloj (horas, minutos,...), entonces hay que hacer la conversión correspondiente. Por ejemplo, si vamos a usar horas, y los parámetros son `trepar` = 2 días y `tfallo` = 1 día, entonces tendríamos que usar `trepar` = 48 horas y `tfallo` = 24 horas (en este caso se verá que la elección de la unidad de tiempo puede afectar a la precisión del resultado y a la eficiencia del proceso).

1.2.4. Tareas a realizar

1. Construid un modelo de simulación usando la técnica de incremento fijo, de acuerdo a las especificaciones antes indicadas.

2. Construid un modelo de simulación usando la técnica de incremento variable, de acuerdo a las especificaciones antes indicadas.

3. Probad los dos modelos para diferentes configuraciones de los parámetros, por ejemplo `trepar` = 2 días y `tfallo` = 1 día, y `trepar` = 0.5 días y `tfallo` = 1 día. Comparad la eficiencia de los métodos de incremento fijo y variable midiendo los tiempos de ejecución de ambos modelos, para un número de días elevado. Probad a utilizar diferentes unidades de incremento de tiempo en ambos modelos (por ejemplo días, medios días, horas, minutos). Comparad también la precisión de los resultados obtenida con ambos modelos[2].

[1] Siempre que ese valor no resulte ser cero, pues en ese caso, como el reloj avanza siempre una unidad, el suceso correspondiente se queda "atrapado en el pasado".

[2] Para el caso en que $s = 0$, es posible determinar de forma muy sencilla cuál es la verdadera duración media

1.3. Sucesos y Grafos de Sucesos. Estructura de un Programa de Simulación Dinámico y Discreto

1.3.1. Sucesos y Grafo de Sucesos

Inicialmente podemos considerar los siguientes sucesos:

- Fallo de máquina
- Fallo del sistema
- Fin de fallo del sistema
- Sustitución de máquina
- Llegada al taller de reparación de una máquina estropeada
- Comienzo de reparación
- Fin de reparación

El grafo de sucesos podría ser el representado en la figura 1.1.

Si reducimos el grafo al mínimo, tenemos los siguientes sucesos, y el grafo correspondiente se representa en la figura 1.2.

- Fallo de máquina
- Fin de reparación
- Fin de simulación (suceso "ficticio" para detener la simulación exactamente en el tiempo deseado)

1.3.2. Parametros de Entrada

- `n`: número de máquinas que tienen que trabajar para que no se produzca un fallo.
- `s`: número inicial de máquinas de repuesto (el sistema tiene por tanto `n`+`s` máquinas).
- `trepar`: media del tiempo de reparación (exponencial) de una máquina (en días).
- `tfallo`: media del tiempo de trabajo (exponencial) de una máquina hasta que falla (en días).
- `tparada`: tiempo de duración de la simulación (en días).
- `reparadores`: número de reparadores.

de los fallos (deducidla), por lo que puede usarse esta información para contrastar los resultados obtenidos por simulación con los verdaderos, y observar así la precisión obtenida con ambos tipos de simulación. Para el caso $s = 1$ también se puede calcular la verdadera duración media de los fallos, aunque no es tan simple (intentad calcularla).

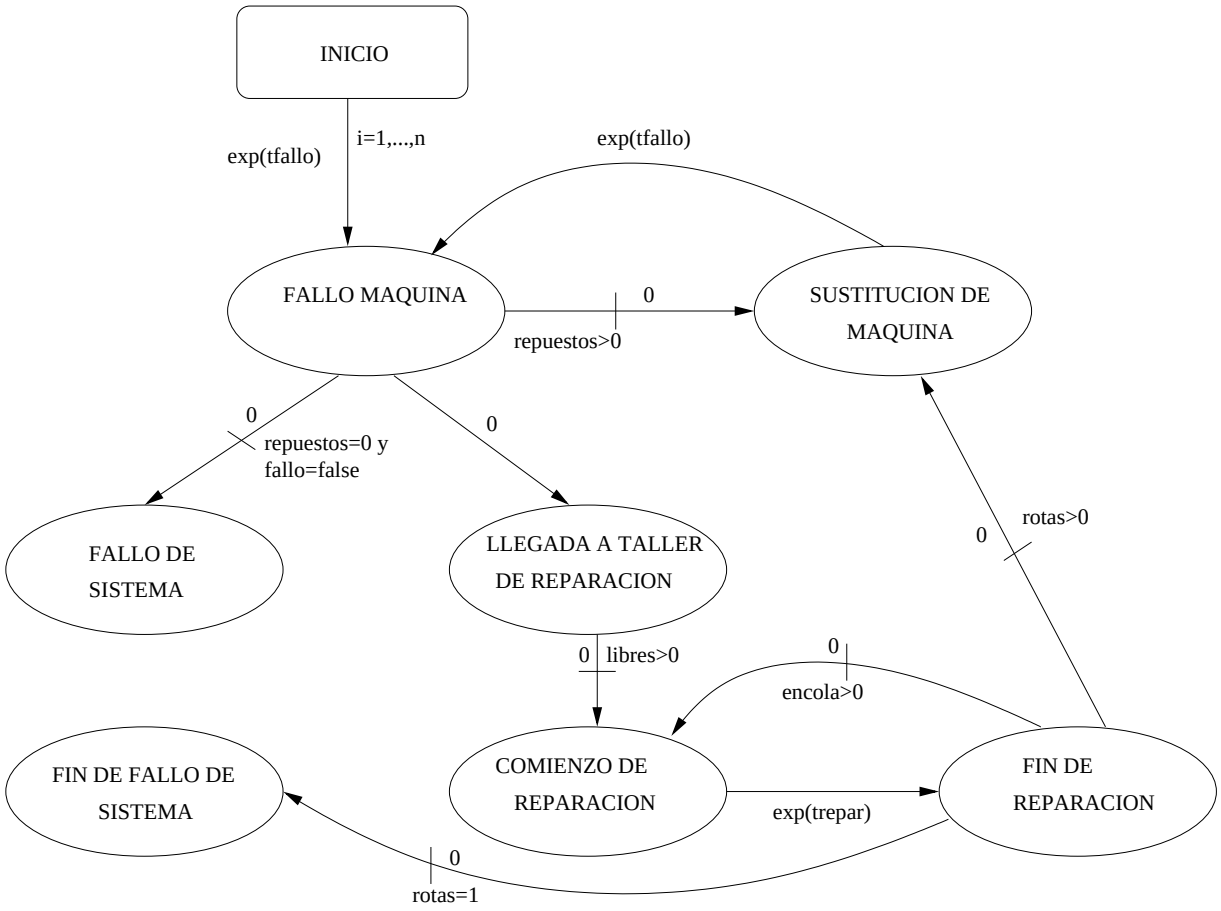

Figura 1.1: Grafo de sucesos original.

1.3.3. Variables de Estado

- `reloj`: reloj de simulación.
- `repuestos`: número de repuestos disponibles.
- `enreparacion`: número de máquinas en el taller de reparación (siendo reparadas o en espera de serlo).
- `rotas`: número de máquinas rotas para las que no existen repuestos (se podría calcular en cada instante mediante max(`enreparacion` – `s`,0), pero se ha optado, por claridad, por mantenerla expresamente).
- `fallo`: si se ha producido o no un fallo (esto ocurre cuando `rotas`>0).
- `libres`: número de reparadores libres.
- `encola`: número de máquinas en espera de reparación. La siguiente relación: `encola` + (`reparadores` – `libres`) = `enreparacion` debe verificarse siempre (las máquinas en cola

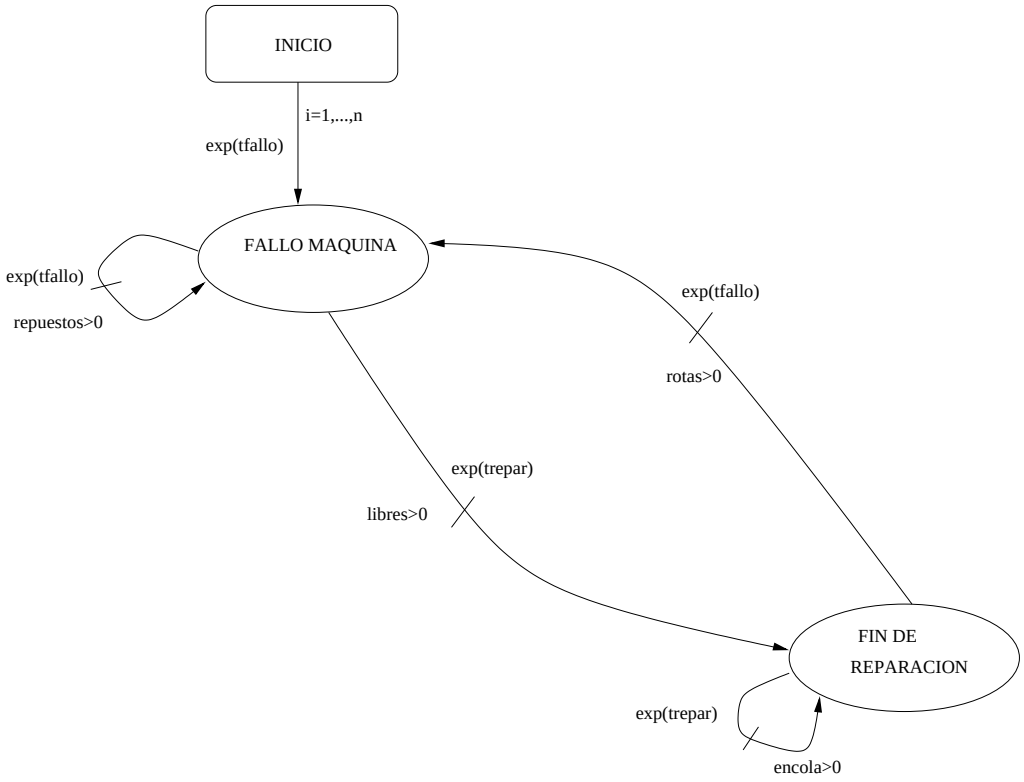

Figura 1.2: Grafo de sucesos reducido.

más las que están siendo reparadas suman el total de máquinas en el taller). Aquí hay otra redundancia entre las variables que se mantiene por claridad.

- `comienzofallo`: instante de comienzo de una situación de fallo.
- `finfalloanterior`: tiempo de ocurrencia del fin de fallo anterior al actual.
- `tusocio`: tiempo del último suceso que cambia `libres`.
- `tusrep`: tiempo del último suceso que modifica `enreparacion`.
- `lsuc`: lista de sucesos, que contiene registros con dos campos: tipo y tiempo (de ocurrencia de cada suceso). Las rutinas de sucesos estan 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 (de modo que se reemplace el tiempo del suceso "gastado" por "infinito" en caso de que no se genere otro suceso del mismo tipo).

1.3.4. Contadores Estadísticos

- `numfallos`: número de fallos producidos.
- `durfallos`: duración acumulada de los fallos.

- `entrefallos`: acumulador de tiempo entre fallos sucesivos.
- `ocio`: acumulador de tiempo de ocio de los reparadores (área bajo la función número de reparadores libres).
- `maqport`: acumulador de máquinas en reparación por tiempo (área bajo la función número de máquinas en reparación).

1.3.5. Calculo de Medidas de Rendimiento

- Duración media de los fallos DMF:

$$DMD = \frac{\texttt{durfallos}}{\texttt{numfallos}}$$

  Cuando se produce un fallo del sistema (ocurre el suceso fallo de máquina y `repuestos`=0), se anota el tiempo en que esto ocurre (en `comienzofallo`) y cuando se termina la situación de fallo (`rotas` vuelve a ser 0), se calcula el tiempo que ha durado el fallo (diferencia entre el valor de `reloj` y `comienzofallo`) y se acumula en `durfallos`. Al final se divide por el número de fallos ocurridos para calcular el promedio.

- Tiempo medio entre fallos del sistema $TMEFS$:

$$TMEFS = \frac{\texttt{entrefallos}}{\texttt{numfallos} - 1}$$

  Cuando se produce un fallo, se calcula la diferencia entre el tiempo en que ocurre (el valor actual de `reloj`) y el tiempo en que terminó el fallo anterior `finfalloanterior`, y se acumula en `entrefallos`. Al final se divide por el número de periodos entre fallos (que es el número de fallos menos uno).

- Número medio de máquinas en reparación $NMMR$:

$$NMMR = \frac{\texttt{maqport}}{\texttt{reloj}}$$

  Se va calculando el área bajo la función número de máquinas en reparación (que es escalonada) acumulando en `maqport` el número de máquinas en reparación (que es `enreparacion`) por el tiempo que efectivamente ese número de máquinas están en reparación (que es `reloj - tusrep`, donde el reloj marca el tiempo de un suceso que modifica el número de máquinas en reparación y `tusrep` marca el tiempo del suceso anterior que modificó dicho número).

- Porcentaje de tiempo de ocio de los reparadores TOR:

$$TOR = \frac{\texttt{ocio}}{\texttt{reloj} * \texttt{reparadores}} * 100$$

  Se va calculando el área bajo la función número de reparadores libres, acumulando en `ocio` el número de reparadores libres (que es `libres`) por el tiempo que esos reparadores

están libres (que es `reloj - tusocio`, donde el reloj marca el tiempo de un suceso que modifica el número de reparadores libres y `tusocio` marca el tiempo del suceso anterior que modificó dicho número). Al final se calcula el porcentaje respecto del tiempo total de trabajo de los reparadores (que es `reloj*reparadores`).

- Porcentaje de duración total de los fallos DTF:

$$DTF = \frac{\texttt{durfallos}}{\texttt{reloj}} * 100$$

Simplemente se calcula el porcentaje del tiempo total que representa la duración acumulada de los fallos.

1.3.6. Rutinas del Modelo de Simulación

Programa principal

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

Rutina de Inicialización

```
inicializar generador de números pseudoaleatorios;
parar = false;
reloj = 0.0;
rotas = 0;
repuestos = s;
fallo = false;
libres = reparadores;
encola = 0;
enreparacion = 0;
numfallos = 0;
durfallos = 0.0;
entrefallos = 0.0;
maqport = 0.0;
ocio = 0.0;
tusocio = reloj;
tusrep = reloj;
crear(lsuc); /* crea la lista de sucesos, inicialmente vacía */
for (i=0; i<n; i++)
   insertar(lsuc,suceso_fallomaquina,reloj+generafallo(tfallo));
insertar(lsuc,suceso_finsimulacion,reloj+tparada);
```

Generador de informes

```
parar = true;
/* ultimas actualizaciones de contadores estadísticos */
maqport += (reloj - tusrep) * enreparacion;
ocio += (reloj - tusocio) * libres;
if (fallo) durfallo += reloj - comienzofallo;
calcular TMEFS, DMF, NMMR, TOR y DTF como se indicó anteriormente e imprimir;
```

Rutina de Temporización

Supuesto que `lsuc` es una estructura de datos con los registros ordenados en orden creciente del campo tiempo, simplemente extrae el registro con menor valor del tiempo, actualiza el valor de `reloj` a ese valor y determina el tipo de suceso de que se trata, para llamar después a la rutina de suceso correspondiente.

```
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);
```

Rutina de suceso(suc_sig)

```
switch(suc_sig) {
                 case suceso_fallomaquina: fallomaquina; break;
                 case suceso_finreparacion: finreparacion; break;
                 case suceso_finsimulacion: generador_de_informes; break;
                }
```

Rutina del suceso fallomaquina

```
maqport += (reloj - tusrep) * enreparacion;
tusrep = reloj;
enreparacion ++;
if (libres > 0)
  {
   ocio += (reloj - tusocio) * libres;
   tusocio = reloj;
   libres --;
   insertar(lsuc,suceso_finreparacion,reloj+generareparacion(trepar));
  }
else encola ++;
if repuestos > 0
  {
   repuestos --;
   insertar(lsuc,suceso_fallomaquina,reloj+generafallo(tfallo));
  }
else {
      rotas ++;
```

```
      if (!fallo)
        {
         fallo = true;
         comienzofallo = reloj;
         numfallos ++;
         if (numfallos > 1) entrefallos += reloj - finfalloanterior;
        }
      }
```

Rutina del suceso finreparacion

```
maqport += (reloj - tusrep) * enreparacion;
tusrep = reloj;
enreparacion --;
if (encola} > 0)
  {
   encola --;
   insertar(lsuc,suceso_finreparacion,reloj+generareparacion(trepar));
  }
else {
      ocio += (reloj - tusocio) * libres;
      tusocio = reloj;
      libres ++;
     }
if (rotas == 0) repuestos ++;
else {
      rotas --;
      insertar(lsuc,suceso_fallomaquina,reloj+generafallo(tfallo));
      if (rotas == 0)
        {
         durfallos += reloj - comienzofallo;
         fallo = false;
         finfalloanterior = reloj;
        }
     }
```

1.3.7. Tareas a realizar

1. El programa `reparadores.cpp` implementa un modelo de simulación para el sistema descrito en su versión general, de acuerdo a las especificaciones anteriores (con el método de incremento variable). La condición de parada a usar en este caso será simular exactamente 365 días.

2. Investigad la diferencia entre un sistema con m reparadores cuya eficiencia es `trepar` = x días (cada uno) y un sistema con un único reparador ($m = 1$) con eficiencia `trepar` = $\frac{x}{m}$ días (es decir, m veces más rápido). Probad también configuraciones intermedias, por

ejemplo $\frac{m}{2}$ reparadores con eficiencia `trepar` $= \frac{x}{2}$ días (la mitad de reparadores pero el doble de rápidos). ¿Resultan sistemas equivalentes?

3. Investigad qué configuración para el sistema, con el menor número posible de repuestos (s) y reparadores (m), consigue reducir la duración total de los fallos del sistema a menos de un $k\,\%$ (por ejemplo $k = 10$) del tiempo total de simulación. Por ejemplo, usad como parámetros $n = 4$, `trepar`=2 días y `tfallo`=1.5 días.

1.4. Modificando el Modelo Original

Supongamos que los responsables del sistema que estamos estudiando se plantean la posibilidad de modificar su funcionamiento (con vistas a mejorar el rendimiento) de la siguiente forma:

Periódicamente, cada `trevision` tiempo, se someterá al sistema a una operación de mantenimiento preventivo. Si hay en ese momento reparadores que se encuentren disponibles y hay máquinas funcionando, cada reparador se encargará de revisar una de esas máquinas (el número de máquinas que se revisarán es el mínimo entre el numero de máquinas funcionando y el número de reparadores disponibles). Si hay más máquinas funcionando que reparadores libres, las máquinas a revisar se elijen al azar (no se sabe cuáles fallarán antes); si hay más reparadores libres que máquinas funcionando, cada reparador se encarga de una máquina y el resto de reparadores continua estando disponible. La operación de mantenimiento de cada máquina tiene una duración distribuida de manera uniforme, entre `tmantmin` y `tmantmax`, y puede realizarse sin afectar al funcionamiento normal de la máquina (se asume que mientras la máquina está siendo revisada, no puede estropearse). Cuando un reparador termina de realizar una operación de mantenimiento, vuelve a quedar disponible para reparar máquinas, y la máquina afectada queda "como nueva" (es decir, su tiempo de próximo fallo se distribuye exponencialmente con media `tfallo`).

1.4.1. Tareas a realizar

1. Modificad el modelo de simulación construido anteriormente para adaptarlo a la nueva situación. El nuevo grafo de sucesos (ya reducido) se representa en la figura 1.3. Especificad las variables de estado, así como las nuevas rutinas de sucesos que sean necesarias. Construid el programa correspondiente.

2. Investigad si este modelo alternativo funciona mejor que el original, desde el punto de vista de la duración total de los fallos del sistema (si se consigue una duración total de los fallos inferior al $k\,\%$ del tiempo total de simulación empleando *menos* recursos, es decir menos repuestos y/o reparadores).

Nota importante: para modelizar este sistema alternativo es necesario eliminar sucesos previamente insertados de la lista de sucesos (cuando un reparador va a mantener una máquina que está funcionando, el suceso de fallo de esa máquina ya no se va a producir en el momento en que estaba previsto, por lo que hay que eliminarlo de la lista). En el grafo de sucesos eso

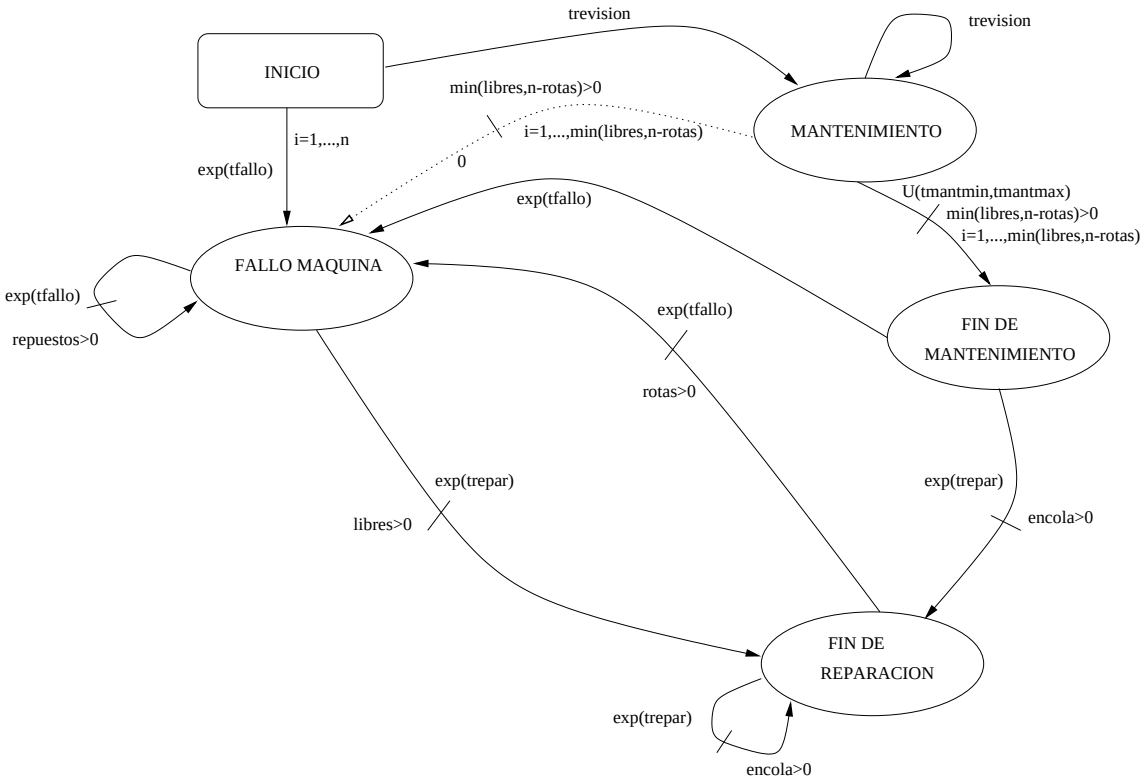

Figura 1.3: Grafo de sucesos reducido del nuevo modelo.

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.

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:

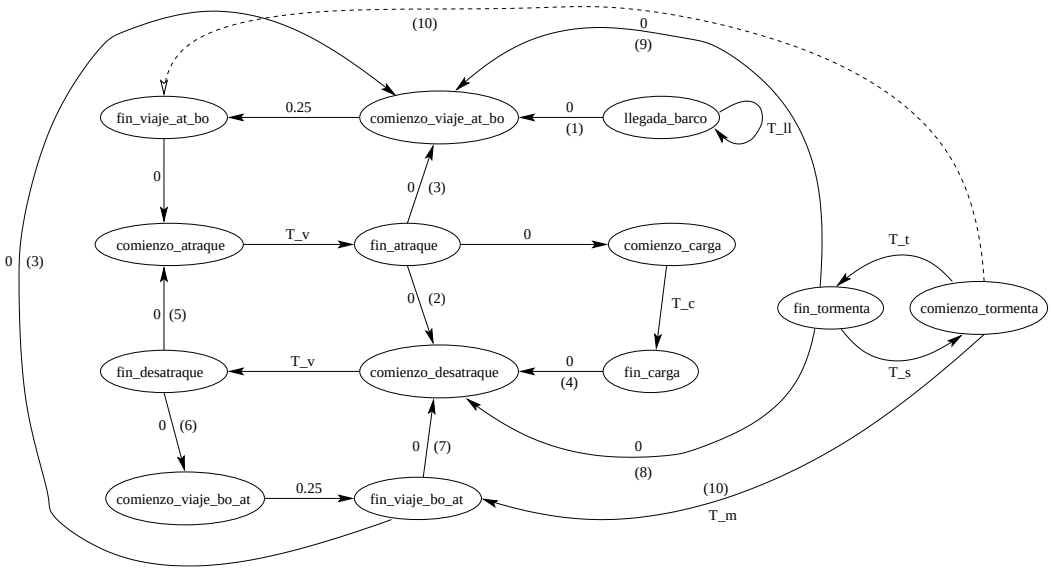

Figura 2.1: Grafo de sucesos inicial

- 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.

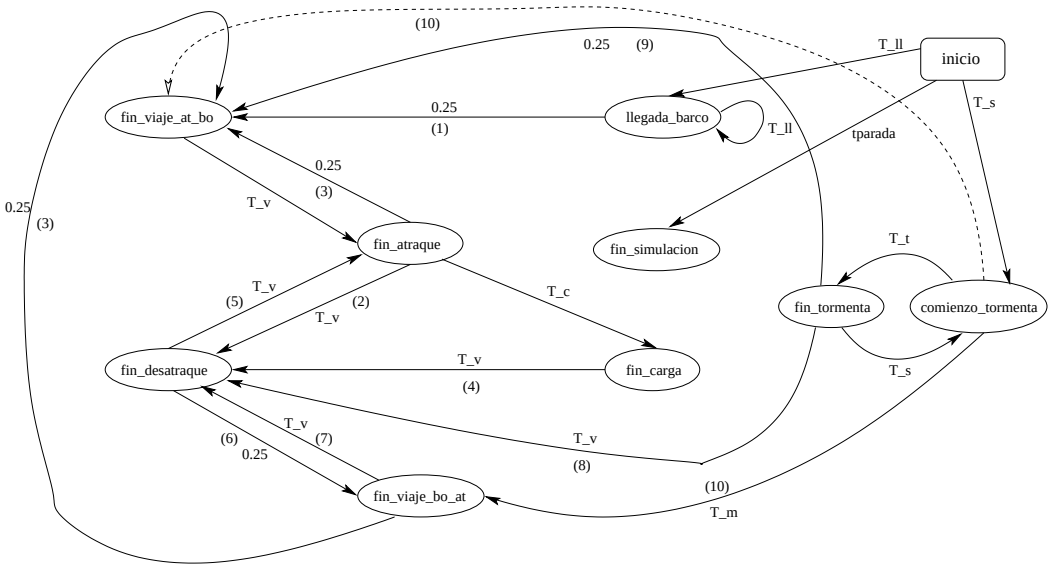

Figura 2.2: Grafo de sucesos reducido

- `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.