UNIVERSIDAD DE GRANADA

Simulación de Sistemas

Ejercicio de Modelos Discretos

Alejandro Manzanares Lemus

alexmnzlms@correo.ugr.es

1 de febrero de 2021

Índice general

políticapolítica

Ejercicio 1:
Modelos discretos

1.1: Simulador base

El código del simulador de esta compañía esta disponible en **src/compania.cpp** e **include/compania.h**

1.1.1: Variables de interés

- **tam**: Representa el tamaño de la demanda del producto **D**.
- **nivel**: Representa el nivel de inventario **I(t)**.
- **pedido**: Representa el volumen de producto que se va a pedir **Z**.
- **tultsuc**: Representa el tiempo ente el suceso actual y el suceso anterior.
- **spequena**: Representa **s**.
- **sgrande**: Representa **S**.

1.1.2: Lista de sucesos

La lista de sucesos se representa de la siguiente manera:

```
typedef struct {
        int suceso;
        float tiempo;
        registro reg_cola;
} suc;

suc nodo;

std::list<suc> lsuc;
```

Simplemente se trata de una lista de objetos del tipo suc, que almacenan el tipo de suceso y el tiempo en el que está planificado. Existe un suceso nodo, que se utiliza para almacenar el suceso actual en cada momento.

1.1.3: Grafo de sucesos

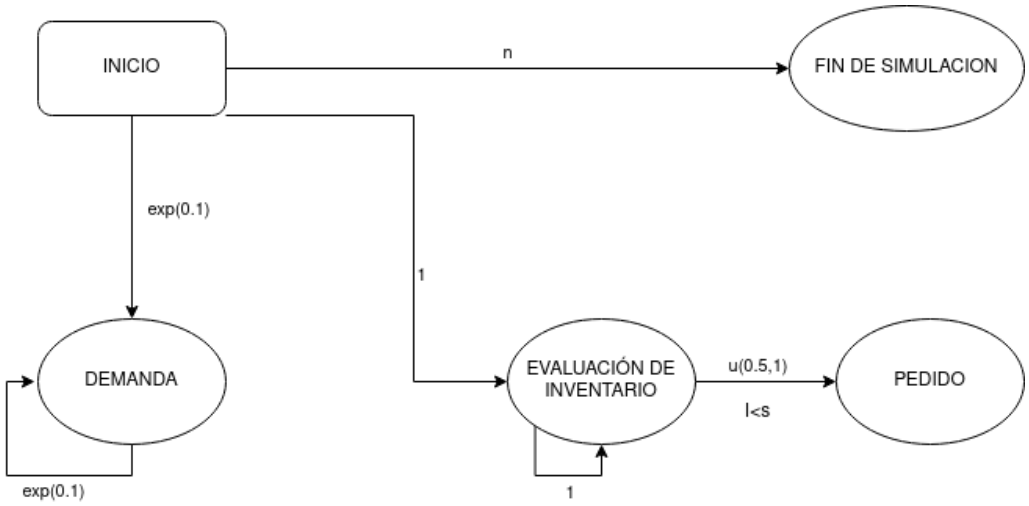

Figura 1.1: Grafo de sucesos

1.1.4: Sucesos

A continuación se muestran los sucesos que se han considerado:

Suceso demanda

Simula la llegada de una demanda del producto a la empresa.

```
if(nivel > 0){
        acummas += (reloj-tultsuc)*nivel;
} else {
        acummenos += (reloj-tultsuc)*(-nivel);
}
tultsuc = reloj;
tam = genera_tamano();
nivel -= tam;
nodo.suceso = SUCESO_DEMANDA;
nodo.tiempo = reloj+gendem(0.1);
insertar_lsuc(nodo);
```

El nivel de inventario disminuye tanto como tamaño tenga la demanda del producto.

Suceso evaluación de inventario

Simula la revisión mensual del inventario por parte de la empresa.

```
if(nivel < spequena && pedido == 0){
        pedido = sgrande - nivel;
        acumpedido += K+i*pedido;
        nodo.suceso = SUCESO_PEDIDO;
        nodo.tiempo = reloj+genpedido(0.5, 1.0);
        insertar_lsuc(nodo);
}
nodo.suceso = SUCESO_EVAL;
nodo.tiempo = reloj+1.0;
insertar_lsuc(nodo);
```

Si $I < s$ y no hay ninguna cantidad de pedido, entonces se hace un pedido de tamaño $S - I$.

Suceso pedido

Simula la llegada de un pedido encargado por la empresa.

```
if(nivel > 0){
        acummas += (reloj-tultsuc)*nivel;
} else {
        acummenos += (reloj-tultsuc)*(-nivel);
}
tultsuc = reloj;
nivel += pedido;
pedido = 0;
```

El nivel de inventario aumenta según el tamaño del pedido.

1.1.5: Generador de informes

Las contadores estadísticos que se utilizan para la generación de informes son:

- **acummas**: Contador estadístico que representa el coste de mantenimiento $I(t)^+$.
- **acummenos**: Contador estadístico que representa el coste de déficit $I(t)^-$.
- **acumpedido**: Contador estadístico que representa el coste de pedido **K+iZ**.

Los informes se almacenan en una matriz de datos, en los que cada fila es un linea del tipo:

```
{(s,S), acumpedido + acummas + acummenos, acumpedido, acummas, acummenos};
```

Además finalmente se muestra la combinación de (s,S) que obtiene un valor de coste total más pequeño.

Los valores aportados son la media de todas las simulaciones realizadas.

1.1.6: Resultados de la simulación

Simulaciones realizadas: 100000

Política	Costo Total	Costo de pedido	Costo de mantenimiento	Costo de déficit
(0,40)	146.184	87.9343	5.80992	52.4401
(0,60)	135.859	84.3331	12.9841	38.5419
(0,80)	134.091	82.4005	21.2844	30.4064
(0,100)	136.499	81.2058	30.1149	25.1786
(20,40)	124.947	97.6004	9.12237	18.2241
(20,60)	118.934	88.5082	17.5176	12.9078
(20,80)	121.297	84.9072	26.8582	9.53105
(20,100)	126.9	82.9457	36.4555	7.4985
(40,60)	125.658	98.2922	25.5252	1.841
(40,80)	125.449	89.1253	34.9559	1.36733
(40,100)	131.483	85.4697	44.9903	1.02278
(60,80)	143.825	98.8523	44.9009	0.0722284
(60,100)	144.159	89.6499	54.455	0.0538317

El mínimo es 118.934 y se obtiene para la configuración (20,60).

1.2: Primera modificación

1.2.1: Variables de interés

- **vendido_mes**: Representa la cantidad de producto vendida en el mes anterior al actual.

1.2.2: Grafo de sucesos

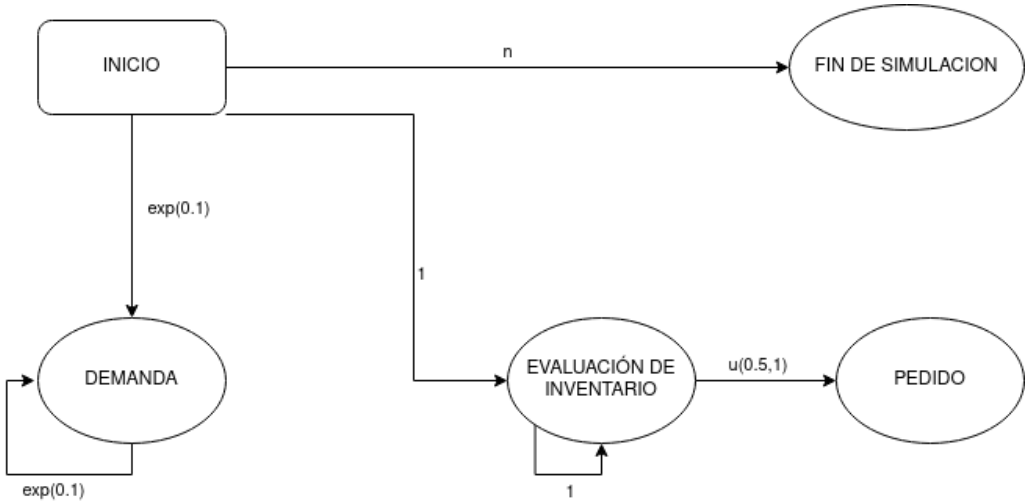

Figura 1.2: Grafo de sucesos

1.2.3: Sucesos

Suceso demanda

```
if(nivel > 0){
        acummas += (reloj-tultsuc)*nivel;
} else {
        acummenos += (reloj-tultsuc)*(-nivel);
}
tultsuc = reloj;
tam = genera_tamano();
nivel -= tam;
vendido_mes += tam;
nodo.suceso = SUCESO_DEMANDA;
nodo.tiempo = reloj+gendem(0.1);
insertar_lsuc(nodo);
```

Se guarda la cantidad de producto vendido durante el mes actual.

Suceso evaluación de inventario

```
pedido = vendido_mes;
vendido_mes = 0;
acumpedido += K+i*pedido;
nodo.suceso = SUCESO_PEDIDO;
nodo.tiempo = reloj+genpedido(0.5, 1.0);
insertar_lsuc(nodo);
nodo.suceso = SUCESO_EVAL;
nodo.tiempo = reloj+1.0;
insertar_lsuc(nodo);
```

El pedido del mes es la cantidad de producto vendida el mes anterior.

Suceso pedido

No ha sido necesario realizar cambios en este suceso.

1.2.4: Resultados de la simulación

Simulaciones realizadas: 100000

Política	Costo Total	Costo de pedido	Costo de mantenimiento	Costo de déficit
0,40)	135.501	105.974	29.0322	0.495184
(0,60)	135.51	105.988	29.027	0.494915
(0,80)	135.518	106.002	29.0191	0.496816
(0,100)	135.514	105.994	29.0242	0.495795
(20,40)	135.512	105.992	29.0239	0.496409
(20,60)	135.513	105.991	29.0245	0.496941
(20,80)	135.502	105.976	29.0317	0.494341
(20,100)	135.52	106.003	29.0191	0.497386
(40,60)	135.512	105.987	29.0295	0.494933
(40,80)	135.52	106.001	29.0199	0.499131
(40,100)	135.508	105.986	29.0272	0.495344
(60,80)	135.513	105.99	29.0259	0.496773
(60,100)	135.513	105.992	29.025	0.495176

El mínimo es 135.501 y se alcanza en la configuración (0,40).

1.3: Segunda modificación

1.3.1: Variables de interés

- **pedido_encargado**: Representa si hay encargado un pedido o no.

1.3.2: Grafo de sucesos

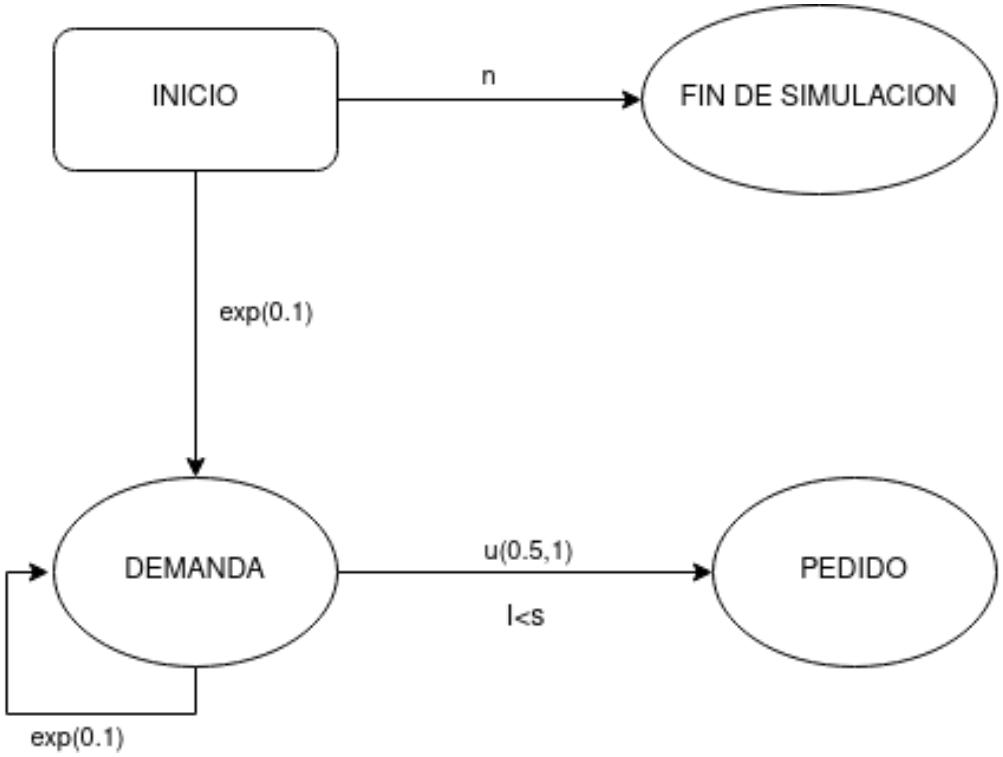

Figura 1.3: Grafo de sucesos

1.3.3: Sucesos

Suceso demanda

```
if(nivel > 0){
        acummas += (reloj-tultsuc)*nivel;
} else {
        acummenos += (reloj-tultsuc)*(-nivel);
}
tultsuc = reloj;
tam = genera_tamano();
nivel -= tam;
if(nivel < spequena && !pedido_encargado){
        pedido = sgrande - nivel;
        acumpedido += K+i*pedido;
        nodo.suceso = SUCESO_PEDIDO;
        nodo.tiempo = reloj+genpedido(0.5, 1.0);
        insertar_lsuc(nodo);
        pedido_encargado = true;
}
nodo.suceso = SUCESO_DEMANDA;
```

```
nodo.tiempo = reloj+gendem(0.1);
insertar_lsuc(nodo);
```

Siempre que no haya un pedido encargado e $I < s$, se hace un pedido de tamaño $S - I$.

Suceso pedido

```
if(nivel > 0){
        acummas += (reloj-tultsuc)*nivel;
} else {
        acummenos += (reloj-tultsuc)*(-nivel);
}
tultsuc = reloj;
nivel += pedido;
pedido = 0;
pedido_encargado = false
```

Una vez llega el pedido, se puede volver a realizar otro.

1.3.4: Resultados de la simulación

Simulaciones realizadas: 100000

Política	Costo Total	Costo de pedido	Costo de mantenimiento	Costo de déficit
(0,40)	125.901	92.8309	7.37193	25.6985
(0,60)	119.952	87.0459	15.5264	17.3799
(0,80)	121.883	84.2107	24.4896	13.1828
(0,100)	127.023	82.5862	33.8031	10.6338
(20,40)	124.722	106.049	11.5772	7.09565
(20,60)	117.737	93.4654	22.2841	1.9878
(20,80)	121.003	87.6403	32.0333	1.32936
(20,100)	127.627	84.7803	41.8365	1.00968
(40,60)	136.957	106.777	29.9181	0.262181
(40,80)	135.786	94.0802	41.6825	0.022782
(40,100)	139.859	88.2189	51.6256	0.0145748
(60,80)	157.185	107.478	49.7034	0.00345746
(60,100)	156.292	94.699	61.5931	2.88242e-05

El mínimo es 117.737 y se alcanza en la configuración (20,60).

1.4: Conclusión

Como se puede observar en los apartados correspondientes a los resultados de las simulaciones, la versión base y la segunda versión alcanzan el mínimo en la configuración (20,60) mientras que la primera versión lo hace en la (0,40). Esto significa:

- En la versión base, cada vez que se programa una evaluación, si el nivel del inventario es inferior a 20, se hace un pedido de 60 menos el nivel de inventario actual (se suma si existe déficit de existencias).
- La primera modificación depende de la demanda, no de la configuración.
- En la segunda modificación, implica que cada vez que el nivel del inventario es inferior a 20, sin necesidad de una evaluación y siempre que no haya otro pedido en curso, se hace un pedido de 60 menos el nivel de inventario actual (se suma si existe déficit de existencias).

Podemos descartar a la primera versión como una política que merezca la pena adoptar, puesto que es la que peores resultados aporta. No es conveniente dejarse llevar sólo por la demanda del mes anterior, sin realizar un análisis previo, a la hora de planificar las ventas del mes actual.

Por tanto, compararemos la versión base y la segunda versión. La diferencia entre estas versiones, no es más que el momento en el que se realiza un pedido. Para la versión base, cada vez se realiza una revisión del estado del inventario, se hace un pedido en consecuencia con el resultado de la evaluación. Sin embargo, la segunda versión no necesita esperar a realizar una evaluación y puede, de manera más dinámica, realizar un pedido siempre que el nivel de inventario rebaje un umbral de peligro (con la restricción de no poder hacer varios pedidos simultáneos).

Según los datos, la política más ventajosa es la que implementa la segunda versión, por lo que podemos inferir que una política más dinámica es más favorable. Esto, si bien es cierto, es una afirmación un tanto pobre, ya que la mejora que obtiene la segunda versión respecto a la versión base es de 1.197€ de media en 100000 simulaciones. Esta diferencia es lo suficientemente pequeña como para no poder asegurar que en todos los casos, una política dinámica sea mejor que una política estática.