DESARROLLO DE DOS MODELOS MATEMÁTICOS; CON RECURSO SIMPLE Y CON RECURSO COMPLETO; PARA LA PLANEACIÓN DE LA PRODUCCIÓN DE SISTEMAS DISCRETOS DE MANUFACTURA UTILIZANDO PROGRAMACIÓN ESTOCÁSTICA MIXTA ENTERA

JUAN GABRIEL HIGUERA FLÓREZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FISICO-MECANICAS

ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES

BUCARAMANGA

2004

DESARROLLO DE DOS MODELOS MATEMÁTICOS; CON RECURSO SIMPLE Y CON RECURSO COMPLETO; PARA LA PLANEACIÓN DE LA PRODUCCIÓN DE SISTEMAS DISCRETOS DE MANUFACTURA UTILIZANDO PROGRAMACIÓN ESTOCÁSTICA MIXTA ENTERA

JUAN GABRIEL HIGUERA FLÓREZ

Tesis de grado presentada como requisito final

para optar el título de Ingeniero Industrial

Directores

Jaime Alberto Camacho P., PhD

Suleyman Karabuk, PhD

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FISICO-MECANICAS

ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES

BUCARAMANGA

2004

Esta tesis esta dedicada a mis Padres; Clara y Gabriel; por su amor y cuidado,

A mis Hermanas; Adriana y Silvia; por consentirme y ayudarme,

A mi Novia; Beatriz; por su apoyo y estar conmigo en mi Auto-Exilio,

Y a mis Amigos; del Colegio, de la UIS y de OU; porque sé que cuento con ellos siempre.

AGRADECIMIENTOS

El autor expresa sus agradecimientos a:

Dr. Suleyman Karabuk, quien me enseño Programación Estocástica y me ayudo a llevar a cabo este proyecto en Estados Unidos.

Dr. Jaime A. Camacho P., por su colaboración para la presentación de este trabajo cuando estuve y no estuve presente en Colombia.

A la Universidad Industrial de Santander y a todos sus docentes, por todas sus enseñanzas durante mas de 4 años de estudio.

A la Universidad de Oklahoma y a todos sus docentes, por el apoyo durante el tiempo de mi estadía allí y por brindarme ahora la oportunidad de continuar estudiando.

A mi familia, por su ayuda incondicional y por sacrificarse por mi.

CONTENIDO

Pág

INTRODUCCIÓN 1

1. EL PROBLEMA DE LA PLANEACIÓN DE LA PRODUCCIÓN 4

1.1 DEFINICIÓN DE PLANEACIÓN DE LA PRODUCCIÓN 4

1.2 SISTEMAS EXISTENTES DEDICADOS A LA PLANEACIÓN DE LA PRODUCCIÓN 5

1.2.1. Sistema WIP 5

1.2.2. Sistema MRP 5

1.2.3. Sistema MRP de Ciclo Cerrado 9

1.2.4. Sistema MRP II 10

1.2.5. Sistemas de Tipo MRP Avanzado 10

1.2.6. Sistema ERP 11

1.3 PAPEL DE LA INVESTIGACIÓN DE OPERACIONES EN LA PLANEACIÓN DE LA PRODUCCIÓN 12

2. PROGRAMACIÓN ESTOCÁSTICA 16

2.1 INTRODUCCIÓN A LA PROGRAMACIÓN ESTOCÁSTICA 16

2.2 PROBLEMAS CON RECURSO SIMPLE 17

2.3 PROBLEMAS CON RECURSO COMPLETO 18

2.4 PROBLEMAS CON RECURSO PARCIAL 18

2.5 INCORPORACIÓN DE LA INCERTIDUMBRE EN LA MODELACIÓN Y PROGRAMACIÓN 19

3. MODELO GENERAL DE PLANEACIÓN DE LA PRODUCCIÓN UTILIZANDO PROGRAMACIÓN ESTOCÁSTICA 23

3.1 INTRODUCCIÓN 23

3.2 MODELO 23

3.2.1 Elementos Probabilísticos 25

3.2.2 Índices 26

3.2.3 Parámetros de Entrada 27

3.2.4 Variables de Decisión 27

3.2.5 Función Objetivo 28

3.2.6 Restricciones de Balanceo de Inventarios 29

3.2.7 Restricciones de Capacidad 29

3.2.8 Restricciones de Tope Máximo de Producción 30

3.2.9 Otras Restricciones 30

4. MODELOS DE EJEMPLOS NUMÉRICOS ALEATORIOS PROGRAMADOS Y RESUELTOS 31

4.1 INTRODUCCIÓN 31

4.2 PRIMER EJEMPLO ALEATORIO CON RECURSO SIMPLE 31

4.3 SEGUNDO EJEMPLO ALEATORIO CON RECURSO SIMPLE 34

4.4 TERCER EJEMPLO ALEATORIO CON RECURSO SIMPLE 36

4.5 EJEMPLOS ALEATORIOS CON RECURSO COMPLETO 38

4.6 RESULTADOS DE LOS EJEMPLOS 39

5. MODELOS DE EJEMPLOS NO ALEATORIOS PROGRAMADOS Y RESUELTOS.....41

5.1 INTRODUCCIÓN 41

5.2 PRIMER EJEMPLO NO ALEATORIO 41

5.3 SEGUNDO EJEMPLO NO ALEATORIO 43

6. COMPARACIÓN DE RESULTADOS CON LOS OBTENIDOS UTILIZANDO PROGRAMACIÓN LINEAL 45

6.1 INTRODUCCIÓN..... 45

6.2 MODELO DE PROGRAMACIÓN LINEAL 45

6.2.1 Función Objetivo 46

6.2.2 Variables de Decisión 46

6.2.3 Restricciones de Demanda 46

6.2.4 Restricciones de Capacidad 47

6.2.5 Restricciones de Tope Máximo de Producción 47

6.2.6 Restricciones de No Negatividad 47

6.2.7 Datos del Modelo 47

6.3 RESULTADOS DE LOS MODELOS DE PROGRAMACIÓN LINEAL 48

6.4 METODOLOGÍA DE LA COMPARACIÓN DE RESULTADOS 49

6.5 RESULTADOS OBTENIDOS EN LA COMPARACIÓN..... 51

7. CONCLUSIONES.....55

8. RECOMENDACIONES 58

9. BIBLIOGRAFÍA.....60

10. ANEXOS 63

LISTA DE TABLAS

Pag.

Tabla 1. Parámetros del Primer Ejemplo 33

LISTA DE FIGURAS

Pag.

Figura 1. Árbol de Escenarios 20

Figura 2. Típica Estructura de Producto 25

Figura 3. Estructura de Producto del Primer Ejemplo 32

Figura 4. Árbol de Escenarios para el Primer Ejemplo 33

Figura 5. Estructura de Producto del Segundo Ejemplo 20

Figura 6. Árbol de Escenarios para el Segundo Ejemplo 25

Figura 7. Árbol de Escenarios para el Tercer Ejemplo 32

Figura 8. Diferencias entre los Costos del Primer Ejemplo 51

Figura 9. Comparación con cada Escenario, Primer Ejemplo 52

Figura 10. Costos Totales del Primer Ejemplo 52

Figura 11. Diferencias entre los Costos del Segundo Ejemplo 53

Figura 12. Comparación con cada Escenario, Segundo Ejemplo 54

Figura 13. Costos Totales del Segundo Ejemplo 54

LISTA DE CUADROS

Pag.

Cuadro 1. Muestra de los Horizontes Usados en la Comparación 33

LISTA DE ANEXOS

Pag.

Anexo A. Resumen de los Modelos de Programación Estocástica 63

Anexo B. Códigos para Resolver los Ejemplos Aleatorios Con Programación Estocástica 65

Anexo C. Soluciones dadas para los Ejemplos Aleatorios por los Códigos Desarrollados 93

Anexo D. Condiciones de Optimalidad de Karush-Kuhn-Tucker 128

Anexo E. Códigos para Resolver los Ejemplos No Aleatorios con Programación Estocástica 129

Anexo F. Soluciones dadas para los Ejemplos No Aleatorios por los Códigos Desarrollados 151

Anexo G. Códigos Desarrollados Utilizando Programación Lineal 169

Anexo H. Soluciones Encontradas para los Modelos de Programación Lineal 179

RESUMEN

TITULO: DESARROLLO DE DOS MODELOS MATEMÁTICOS; CON RECURSO SIMPLE Y CON RECURSO COMPLETO; PARA LA PLANEACIÓN DE LA PRODUCCIÓN DE SISTEMAS DISCRETOS DE MANUFACTURA UTILIZANDO PROGRAMACIÓN ESTOCÁSTICA MIXTA ENTERA [*]

AUTOR: Juan Gabriel Higuera F. [**]

PALABRAS CLAVE: Programación Estocástica, Escenarios, Investigación de Operaciones, Planeación de la Producción.

En este trabajo se desarrollan dos modelos matemáticos de Programación Estocástica para planear la producción de un sistema productivo, intentando minimizar los costos de producción. Estos modelos no serán implementados en alguna empresa en particular, son modelos teóricos que lo pueden ser con una investigación posterior. De esta forma, se intento crearlos para que puedan ser fácilmente integrados con sistemas de planeación ampliamente reconocidos y aceptados como el MRP, MRP II y el ERP. Los modelos desarrollados cuentan con un elemento probabilístico que es la demanda, la cual, se modeló por medio de árboles de escenarios.

Se crearon seis ejemplos numéricos con parámetros aleatorios, los cuales se codificaron y se resolvieron en el software Xpress-MP de Dash Optimization. Seguidamente, se ampliaron estos códigos a dos ejemplos numéricos no aleatorios que igualmente se resolvieron por medio del software anteriormente nombrado. Estos dos últimos ejemplos fueron igualmente desarrollados, codificados y resueltos por medio de Programación Lineal.

Los resultados de ambas técnicas se compararon utilizando horizontes de tiempo y observando el comportamiento de los costos al aplicar los planes obtenidos. La Programación Estocástica mostró ser superior al ser comparada con la Programación Lineal en cuanto a costos. En cada uno de los dos ejemplos se obtuvieron un ahorro del 1% y del 3% en los costos de producción. Se recomienda ampliar el estudio transformando otros modelos de planeación de la producción que utilizan Programación Lineal a modelos que utilicen Programación Estocástica y de esta forma llevar a cabo una comparación mas completa.

[*] Tesis de Grado

[**] Facultad de Ingenierías Físico-Mecánicas. Programa de Ingeniería Industrial. Director, Dr. Jaime Alberto Camacho P.

ABSTRACT

TITLE: DEVELOPMENT OF TWO MATHEMATICAL MODELS; WITH SIMPLE RECOURSE AND COMPLETE RECOURSE; FOR DISCRETE MANUFACTURING SYSTEMS PRODUCTION PLANNING USING MIXED INTEGER STOCHASTIC PROGRAMMING [*]

AUTHOR: Juan Gabriel Higuera F. [**]

KEY WORDS: Stochastic Programming, Scenarios, Operations Research, Production Planning.

In this project, two stochastic programming models are developed for the production planning of a productive system, trying to minimize the production costs. These models are not going to be used in a particular company, are theoretical models that can be applied with a future work. In this way, the developed models are conceptually and practically easy to integrate with wide spread planning systems like MRP, MRP II and ERP. The models have a probabilistic element that is the demand, which is modeled using scenario trees.

Six numerical examples are created with random parameters. These ones were codified and solved with the software Xpress-MP of Dash Optimization. Then, these codes were modified to use them with two numerical no-randomized examples, which were also solved using the same software. These two last examples were also developed, coded and solved using Linear Programming.

The results provided by both techniques were compared using time horizons and checking the cost behavior when the given plans were applied. Stochastic Programming showed being superior when was compared with Linear Programming using the production costs. In each one of the two examples a reduction of about 3% and 1% in the production costs were obtained when Stochastic Programming was used. Using Stochastic Programming in other production planning models that are developed using Linear Programming is recommended to make a better comparison.

[*] Thesis

[**] College of Fisical-Mecanical Engineering. Industrial Engineering Program . Advisor, Jaime Alberto Camacho, PhD

INTRODUCCIÓN

La Investigación de Operaciones es una ciencia que se creó durante la Segunda Guerra Mundial y desde entonces su evolución no ha cesado. Una de las técnicas desarrolladas en este campo es la Programación Estocástica, la cual fue una ampliación de la técnica conocida como Programación Lineal. En programación Lineal todos los coeficientes envueltos tanto en la Función Objetivo como en las restricciones son de carácter determinístico, mientras que en la Programación Estocástica por lo menos uno de ellos es de carácter probabilístico.

En el Proyecto aquí propuesto se pretende desarrollar modelos encaminados a la Planeación de la Producción que serán modelados por medio de la Programación Estocástica y serán programados y resueltos por medio del software Xpress-MP de Dash Optimization (www.dashoptimization.com). Una vez programados y resueltos estos modelos se procederá a desarrollar otros códigos para dos ejemplo propuesto a la Universidad de Oklahoma, cuyas soluciones serán comparadas con la soluciones que se obtendrían utilizando Programación Lineal.

Los objetivos de este trabajo son:

- Desarrollar dos modelos matemáticos para planear la producción de sistemas discretos de manufactura utilizando Programación Estocástica Mixta.
- Modelar sistemas de producción lo más parecidos a la realidad teniendo en cuenta las limitaciones para su posterior solución.
- Solucionar los modelos propuestos para la planeación de la producción utilizando software reconocido en la industria para resolver modelos de Programación Estocástica como lo es Xpress-MP de Dash Optimization.

- Comparar los resultados obtenidos mediante Programación Estocástica con los obtenidos utilizando Programación Lineal.

El proyecto pretende crear modelos para planear la producción de un sistema con múltiples periodos, plantas, productos y por supuesto escenarios. Sin embargo, la cantidad de éstos no es ilimitada y es una limitante al momento de resolver los modelos para obtener resultados. El software utilizado para esto es Xpress-MP versión 2003 de Dash Optimization. Se espera poder resolver modelos hasta con 20 diferentes índices (teniendo en cuenta que los principales son 4: Plantas, periodos, productos y escenarios). De todas formas ésto depende de la manera en que sea modelado el ejemplo y de si éste es con recurso simple o recurso completo; el ultimo con una mayor complejidad a la hora de ser resuelto. La modelación no tiene ningún tipo de limitante en cuanto al tamaño, ésta se presenta al momento de ser resuelto por limitaciones de las técnicas de solución y del software existente.

La cantidad de coeficientes variables o dependientes de una distribución de probabilidad puede ser mayor; sin embargo, en estos ejemplos sólo se tiene en cuenta la demanda de cada producto y periodo variable. Esto se hace porque al tomar en cuenta mayor número de coeficientes variables, éstos se tendrían que combinar haciendo mas complicado el árbol de escenarios y poco a poco se llega a una explosión de escenarios. En otras palabras el número de escenarios en el modelo crecería exponencialmente y su solución se complicaría (Wu y Sen, 2000).

En este proyecto no se estudian a fondo las diferentes técnicas o algoritmos de solución al no creerse conveniente por la ya existencia de un software y su posible programación.

El proyecto no es para ser implementado en alguna empresa en especial, sino para desarrollar modelos teóricos que podrán ser utilizados en ciertas industrias dependiendo de sus sistemas. El proyecto también envuelve dos ejemplos dados a la Universidad de Oklahoma pero éstos sólo sirven como un medidor para comparar los resultados con otros obtenidos mediante Programación Lineal. Los

resultados no son, ni serán, implementados en la empresa de la cual provienen los datos. Es de tener en cuenta que este proyecto es de tipo teórico y no constituye una práctica empresarial o la implementación de resultados en una industria.

Este proyecto fue escogido y se realiza por varios motivos. El primero de ellos es por la poca difusión que tiene la Programación Estocástica en la Universidad Industrial de Santander, por lo cual trabajar sobre ésto parece interesante a primera vista. Otro motivo es la gran ventaja que tiene, al parecer, la Programación Estocástica sobre la Programación Lineal, ventaja que aquí se quiere comprobar para ciertos modelos. De esta forma este trabajo podría tener conclusiones importantes sobre la aplicación de esta técnica en la Planeación de la Producción.

Actualmente para planear la producción teniendo en cuenta la estructura del producto se utiliza el MRP o el ERP y la Programación Estocástica podría significar un avance significativo en cuanto a la utilización de una técnica matemáticamente mas fundamentada. Los resultados de este proyecto podrían ser aprovechados por industrias como la del automóvil, manufactura de naves aéreas, o la de computadores y aparatos electrónicos. También este proyecto puede servir como ejemplo para ampliar y utilizar la Programación Estocástica en un número significativo de modelos de manufactura de partes discretas.

1. EL PROBLEMA DE LA PLANEACIÓN DE LA PRODUCCIÓN

1.1 DEFINICIÓN DE PLANEACIÓN DE LA PRODUCCIÓN

La planeación de la producción es la función que sistematiza por anticipado los factores de mano de obra, materias primas, maquinaria y equipo en un proceso productivo. Esta hace parte de la dirección de la empresa y es importante para realizar la fabricación que este determinada por anticipado, con relación a:

- Utilidades que deseen lograr.
- Demanda del mercado.
- Capacidad y facilidades de la planta.
- Puestos laborales que se crean.

Es la actividad de decidir acerca de los medios que la empresa industrial necesitará para sus futuras operaciones manufactureras y para distribuir esos medios de tal forma que se fabrique el producto en las cantidades deseadas, al menor costo posible. Su finalidad es vigilar que se logre:

- Disponer de materias primas y demás elementos de fabricación, en el momento oportuno y en el lugar requerido.
- Reducir en lo posible, los periodos muertos de la maquinaria y de los operarios.
- Asegurar que los operarios no trabajan en exceso, ni que estén inactivos.

1.2 SISTEMAS EXISTENTES DEDICADOS A LA PLANEACIÓN DE LA PRODUCCIÓN

Desde un comienzo, a partir de la revolución industrial, las empresas han tratado de llevar a cabo sus trabajos de una manera ordenada y eficiente. Es así como se desarrollaron métodos de planeación de la producción con el pasar de los años y la experiencia. En la actualidad el ERP (Enterprise Resource Planning) es considerado el mas avanzado método para llevar a cabo esta importante función.
El proceso de evolución de los sistemas de planeación se sustenta en 4 grandes etapas que marcaron el proceso de entrada a las ERP, ellos son: WIP, MRP, Ciclo Cerrado MRP, MRP II (Puerto, 2000).
A continuación se desarrollan estos métodos o sistemas.

1.2.1. Sistema WIP

Este proceso de gestación inicia en la revolución industrial con Schonberger, que dio lugar a la necesidad de manejar inventarios de proceso de trabajo (Work in Progress), que controlase materia prima e inventario final. Ya para este siglo se dio inicio a la Administración Científica ideada por Taylor y a la medición de tiempos y movimientos de Frank y Lillian Gilbreth.

1.2.2. Sistema MRP

Los sistemas MRP y MRP II se asociaron en un inicio solamente a las grandes computadoras, computadoras centrales y minicomputadores. Esta perspectiva ha ido cambiando porque en la actualidad se encuentra una gama amplia de software que incluye varios sistemas operativos, redes con todas sus topologías y plataformas que permitieron a los pequeños fabricantes adquirir este tipo de sistemas, situación que en un inicio fue demasiado compleja y casi imposible de implantar debido a lo alto de sus costos.

Las personas que se encargaban de planear la producción se encontraban bastante limitados porque la velocidad de computo hace varios años duraba horas y en algunos casos días y estos planeaban su semana con base en los resultados ofrecidos por el MRP. Era bastante complicado poder realizar una toma de decisiones eficiente. En la actualidad lo que tardaba horas o días, se realiza en pocos minutos.

Este cambio fue revolucionado por la arquitectura Cliente/Servidor que permitió que parte de la aplicación se refiriera al cliente y parte al servidor. Dada esta división la transmisión de los datos disminuyó notoriamente porque solamente se transfería la información necesaria, mejorando el desempeño de las redes y en últimas el desempeño de la aplicación.

Los sistemas de requerimientos de materiales (MRP) se han instalado casi universalmente en las empresas del sector manufacturero, incluso en aquellas que se consideraban pequeñas. El objetivo o la razón de las MRP, fue utilizar un enfoque lógico y de fácil comprensión del problema, que ayudo a determinar el número de partes, componentes y materiales necesarios para producir cualquier producto. Asimismo, los programas que utilizaban MRP fueron capaces de proveer los tiempos de cuando se debía ordenar o producir cada uno de los materiales o materia prima (Chase y Aquilano, 1995).

La MRP original solo planeaba los materiales, sin embargo en la medida en que fue creciendo el poderío de las máquinas computacionales también fue posible para las MRP tener en cuenta mas recursos. Pronto se consideraron los recursos al igual que los materiales y el crecimiento computacional cambio el MRP a MRP II que significa Planeación de Recursos de Manufactura, aspecto que se tocara mas adelante.

Actualmente, la MRP tiene un impacto sobre la totalidad del sistema e incluye el JIT, el Kanban y las manufacturas integradas por computador o CIM.
La MRP logro extenderse hasta los archivos que manejaban la lista de materiales y el archivo de registro del inventario, para crear una programación del tiempo y del número de unidades necesarias en cada etapa del proceso.
Este programa podía incluir 20 o mas módulos para controlar todo el sistema,

desde la entrada del pedido hasta el manejo de la finanzas, contabilidad y cuentas por pagar, entre otros. Las MRP están basadas en la demanda dependiente, que es aquella causada por la demanda de un articulo de más alto nivel .

Las MRP están utilizándose en una variedad de industrias con un ambiente de trabajo basado en la fabricación por lotes utilizando el mismo equipó de producción. Las MRP son muy valiosas para aquellas compañías que involucran operaciones de ensamblaje y menos valiosa para las compañías con procesos de transformación de materias primas en productos. Por otra parte, las MRP no funcionan bien en compañías que producen un bajo número de unidades por año. Especialmente en compañías que fabrican productos complejos y costosos que requieren investigación y diseños avanzados (Chase y Aquilano, 1995).

Con base en el plan de producción, un sistema de Planeación de Requerimiento de Materiales crea programas que identifican partes y materiales específicos requeridos para producir artículos finales.

Los sistemas de MRP utilizan un programa de computador para llevar a cabo estas operaciones. La mayoría de empresas utilizaban sistemas de inventarios computarizados durante años, pero estos eran independientes del sistema de programación; las MRP lograron enlazarlos.

Los principales propósitos de una MRP son controlar los niveles de inventario, asignar prioridades operativas a los artículos y planear la capacidad para cargar el sistema de producción, así:

Inventarios: Ordenar las partes correctas, ordenar la cantidad correcta y ordenar en el momento correcto.

Prioridades: Ordenar con la fecha de vencimiento correcta y mantener válida la fecha de vencimiento.

Capacidad: Planear una carga completa, planear una carga exacta y planear un momento adecuado para mirar la carga futura (Chase y Aquilano, 1995).

Los objetivos del manejo del inventario bajo un sistemas de MRP son los mismos que bajo cualquier sistema del manejo del inventario: Mejorar el servicio al cliente, minimizar la inversión en el inventario y maximizar la eficiencia operativa de la producción.

La filosofía de la Planeación de Requerimientos de Materiales es que estos deben enviarse de prisa y este envío debe efectuarse cuando la falta de ellos pueda retrazar el programa de producción general y demorarse cuando el programa se atrasa, posponiéndose hasta cuando se necesite.

Aparte de utilizar, tal vez, una escasa capacidad, es preferible no tener materias primas ni trabajo en proceso antes de que aparezca la necesidad real por cuanto los inventarios paralizan las finanzas, trastornan los depósitos, prohíben los cambios de diseño e impiden la cancelación o el aplazamiento de pedidos.

Entre las ventajas o beneficios de un sistema MRP se pueden considerar los siguientes ítems:

- Reducción de los costos de preparación y desmonte.
- Reducción del tiempo de inactividad.
- Capacidad para fijar los precios de una manera más competente.
- Mejor servicio al cliente.
- Posibilidad de cambiar las cantidades de los pedidos.
- Mejor respuesta a las demandas del mercado.
- Reducción de los precios de venta.
- Reducción del inventario.
- Capacidad para cambiar el programa maestro.
- Suministrar información por anticipado, de manera que los gerentes puedan ver el programa planeado antes de la expedición real de los pedidos.
- Indicar cuando demorar y cuando agilizar.
- Demorar o cancelar pedidos. Agilizar o retardar la fecha de los pedidos.
- Ayudar en la capacidad de planeación.
- Reducción hasta el 40% en las inversiones de inventario

(Chase y Aquilano, 1995).

Los principales problemas de las MRP se encuentran basados en las fallas del proceso de instalación. Los principales factores son a nivel organizacional y de comportamiento. Se han identificado tres causas principales: La falta de compromiso de la alta gerencia, el hecho de no reconocer que la MRP es solo una herramienta de software que no genera toma de decisiones y la integración de la MRP y el JIT.

1.2.3. Sistema MRP de Ciclo Cerrado

El MRP se difundía a una escala tal y con tantos contratiempos, que dio como resultado la necesidad de obtener más beneficios y mejorar en la técnica. Gran parte de este desarrollo se dio mediante Prueba y Error. Es así como en los años setentas se procedió a ampliar el concepto de MRP. Cuando el sistema de Planeación de Requerimientos de Materiales (MRP) tiene retroalimentación de la información proveniente de los resultados de su módulo, esto se denominan MRP de ciclo cerrado. Esta se podría definir como “Un sistema creado alrededor de los requerimientos de materiales que incluye funciones adicionales como la planeación de ventas y operaciones (planeación de la producción, programación maestra de la producción y planeación de los requerimientos de capacidad), genera una vez completada la fase de planeación y aceptado los planes como realistas y asequibles, las funciones de ejecución. Estas incluyen las funciones de control de fabricación, medición de insumo producto (capacidad), la programación y despachos detallados, al igual que los informes anticipados sobre retraso tanto de la planta como de proveedores. El término "ciclo cerrado" implica que no solo se incluye cada uno de estos elementos en el sistema global sino también las funciones de ejecución que proveen una retroalimentación de manera tal que se puede mantener valida la planeación en todo momento” (APICS Dictionary, 1984).
En resumen, el ciclo cerrado significa que las cuestiones y los datos resultantes se ingresan de nuevo al sistema para su verificación y, de ser necesario, su modificación.

1.2.4. Sistema MRP II

Una expansión del sistema de Planeación de Requerimientos de Materiales para incluir otras porciones del sistema productivo era natural y se preveía. Uno de los primeros elementos en incluirse era la función de compras, al mismo tiempo, había una inclusión mas detallada del sistema productivo mismo, es decir, la planta, el despacho y el control detallado de la programación. La MRP había incluido ya las limitaciones de capacidad con respecto al centro de trabajo, así que era obvio que el término de Planeación de Requerimiento de Materiales, ya no era adecuado para describir el sistema expandido. Por tal motivo, en 1980 se introdujo el término de planeación de recursos de manufactura (MRP II), para reflejar la idea de que una mayor parte de la empresa se esta involucrando en el programa. El intento inicial para la MRP II fue planear y monitorear todos los recursos de una firma manufacturera, entre los que se incluía el mercadeo, la manufactura, las finanzas e ingeniería de procesos, a través de un sistema de ciclo cerrado que generaba cifras financieras. El segundo intento importante del concepto de MRP II fue que este simulará el sistema de fabricación.

1.2.5. Sistemas de Tipo MRP Avanzado

Durante más de dos décadas, los sistemas MRP fueron la primera elección para las empresas enfocadas en el nivel de producción de la planta. Durante este tiempo el mundo fue cambiando con nueva competencia, multiplantas en lugares internacionales, amplia demanda mundial de productos, subcontratación internacional, mercados monetarios variados y otros nuevos aspectos que fueron los causales de que los programas de software de MRP existentes no cumplieran de forma estandarizada. Así tampoco podrían estas nuevas aplicaciones manejar las características anteriormente descritas.

En el medio actual, los usuarios de la MRP desean acceso instantáneo a la información sobre las necesidades de los clientes y sobre los niveles de inventario de toda la compañía, como también tener la capacidad suficiente de suministro. De esta forma, para satisfacer estas necesidades las empresas que ofrecen

soluciones MRP se acomodaron a los nuevos requerimientos y generaron un desarrollo de nuevos sistemas avanzados basados en la lógica de la MRP. A esta nueva generación de MRP se le han dado diferentes nombres entre ellos se les llamo la nueva MRP como Planeación de Recursos Empresariales (ERP), todo con el propósito de operar totalmente en sentido de la empresa.

1.2.6. Sistema ERP

La Planeación de Recursos Empresariales (ERP), son mucho más que un Software por que permite la integración y optimización de todos los procesos y recursos de la organización.

Las ERP se inician en los años 70, cuando Gartner Group y AMR dan el primer paso para definir sus bases. En un comienzo las organizaciones estaban orientadas a las funciones y solo se cubría las áreas de manufactura y finanzas. Posteriormente, en los años 80, se cambio el paradigma de estar orientados a funciones y se orientó hacia los procesos, por medio del cual se identificaron los procesos críticos de las empresas y su integración entre sí. La información comenzó a utilizar el modelo Cliente/Servidor (Puerto, 2000). Una de las mayores fortalezas de las soluciones ERP es su poder integrador de diferentes procesos como son: Logística, Inventarios, Compras, Ventas, Distribución, Producción, Recursos Humanos, entre otros. Así mismo, existen varios autores que explican que una ERP no es un proceso evolutivo de dicha metodología, tal es el caso de Don Ralston que explica: "El ERP no es un paso mayor, pero si una expansión del MRP II, con nuevas aplicaciones a través de la incorporación de tecnología moderna a la empresa, como el concepto Cliente-Servidor y EDI (Intercambio de Información Electrónica), que han provisto de mayor alcance al uso de la misma filosofía." (Ralston, citado por Camacho, 1997). A su vez David Turbide comenta: "Es un intento por renombrar al MRP II sin ningún cambio real en su naturaleza, mas bien, es una extensión que incorpora tecnología moderna." (Turbide, citado por Camacho, 1997).

Ahora, es entendible como las empresas grandes que manipulan un gran volumen de datos, usuarios y complejidad de transacciones escogen este tipo de soluciones porque establecen que el aumento en el rendimiento de las actividades corporativas y mejora en el servicio al cliente y calidad de la producción, provocan una mejora en la imagen corporativa, manejo de la integración de procesos y calidad de los mismos.

1.3 PAPEL DE LA INVESTIGACIÓN DE OPERACIONES EN LA PLANEACIÓN DE LA PRODUCCIÓN

En la planeación de la producción la Investigación de Operaciones ha jugado un papel importante al momento de optimizar los costos y/o los beneficios. Actualmente, muchas empresas han reconocido el potencial de los sistemas existentes para la planeación, pero también se han dado cuenta de las fallas que estas tienen. Una de estas es que la optimización de los costos no se hace de una manera matemática; se asume que esta se da porque el momento de pedir se cree que ha llegado o es el indicado y que de este modo ni se incurre en costo de mantenimiento de inventario ni en costo de no tener los materiales al momento requerido. Es así como se han desarrollado modelos que pretenden planear la producción de sistemas discretos de manufactura explotando esta desventaja de los sistemas anteriormente mencionados como el MRP (Bahl, Ritzman y Gupta, 1987).

Es necesario tener en cuenta que en la mayoría de los modelos matemáticos que se han desarrollado e implementado, su modelación tiene similitudes a los sistemas utilizados para la planeación como el MRP. Se han agregado variables y restricciones que hacen posible no solo la aplicación del modelo por si solo, sino una integración entre el modelo y los sistemas como el MRP, MRP II o el ERP (Shapiro, 1993). Debido a la gran acogida de estos sistemas, los modelos propuestos deben ser conceptual y prácticamente posibles de integrar a estos sistemas si quieren tener un futuro significativo.

La implementación de los sistemas de apoyo a la gestión basados en optimización ha recibido poca atención en cuanto a las publicaciones académicas. Sin embargo, últimamente esta tendencia ha ido bajando debido a que un gran número de estas aplicaciones han fracasado al momento de ser implementadas. Es por esto que grandes investigadores en el tema de Investigación de Operaciones se han preocupado, ya que en la medida que no se apliquen las ideas desarrolladas en esta área; la cual partió con un enfoque plenamente aplicado; la Investigación de Operaciones irá perdiendo relevancia. Algunos de los factores que limitan el uso de la Investigación de Operaciones son la baja productividad de los modeladores y la apatía por parte de los tomadores de decisiones, pero a su vez existen diferentes oportunidades para la implementación como los progresos en tecnologías de bases de datos, la revolución de los computadores personales, progresos en los fundamentos de la modelación y el gran número de usuarios de hojas de cálculo (Geoffrion, 1987).

Para demostrar los nuevos éxitos que se tienen en la modelación y la implementación se han publicado casos donde se analizan algunos aspectos relacionados con la implementación de un sistema de apoyo a la gestión para la planificación de la producción en una empresa manufacturera (Gazmuri y Arrate, 1995). También, se han analizado los aspectos relacionados con la interfaz con el usuario de este mismo sistema, que es uno de los aspectos que se ha visto que es más relevante para el éxito en la implementación de este tipo de sistemas (Maturana y Eterovic, 1995) y se ha descrito el proceso de este sistema de apoyo a la toma de decisiones (Maturana *et al.*, 1996).

Uno de los usos más importantes de la programación matemática, desde sus mismos comienzos, fue para resolver problemas de planificación, en particular, los problemas de planificación de la producción. Uno de los avances importantes que se produjeron en el tema de la planificación de la producción fue el concepto de planificación jerárquica de la producción. Recientemente la planificación jerárquica de la producción ha vuelto a concentrar bastante actividad de investigación, principalmente en la forma de agregar variables con incertidumbre. Se ha discutido

el uso de la planificación jerárquica para un modelo de múltiples períodos con un nivel de planificación agregada y otro de nivel detallado, con una demanda incierta (Gfrerer y Zaepfel, 1995). También se ha propuesto el uso de la programación lineal estocástica para problemas de planificación jerárquica de la producción. Los autores agregan un término en la función objetivo para penalizar la infactibilidad y una variable aleatoria para la demanda (Kira, et al., 1997). Otros trabajos desarrollados proponen distintos modelos de programación lineal y entera para los problemas de planificación de la producción y de la capacidad, suponiendo una demanda incierta. Los autores usan escenarios para representar la incertidumbre. El primer modelo de programación lineal representa un problema de planificación de la producción para múltiples productos, múltiples períodos que minimice el costo de mantener inventario y la demanda insatisfecha. El segundo modelo de programación entera mixta, también para múltiples productos, múltiples períodos, donde lo que se busca decidir es donde comprar la materia prima. A pesar que estos modelos conducen a problemas de gran tamaño, se deja como conclusión que se pueden llegar a resolver en tiempos razonables (Escudero et al. 1993).

También se ha presentado un sistema de apoyo a la gestión para la planificación de la producción que usa una base de conocimiento obtenida de simulaciones del sistema de producción bajo condiciones que se varían en forma sistemática. Finalmente un modelo de simulación visual interactivo permite comparar distintos resultados de acuerdo a tres criterios de desempeño. En este caso también la interfaz gráfica resulta ser muy importante tanto para captar como para transmitir información del usuario (Gravel *et al.* 1994).

Una aplicación interesante es el desarrollo de un modelo para ayudar en la toma de la decisión de las capacidades de cuatro de la líneas de producción de la General Motors. El modelo se basó en escenarios de demandas con ciertas probabilidades lo que llevó a un problema de programación entera estocástica con recurso. Además se incorporó un análisis de riesgo para tomar en cuenta la probabilidad de una utilidad negativa (Eppen *et al.*, 1989).

En la actualidad, el futuro para la aplicación de nuevos modelos matemáticos es prometedor porque muchos gerentes de producción han notado las limitaciones de sus sistemas MRP. Estos son, de hecho, no mas que programas de información que proveen requerimientos de materiales a partir de planes predeterminados. Además, aunque algunos MRP calculan la cantidad de productos que se van a fabricar que resulta de un plan detallado, los sistemas no determinan un plan de producción efectivo por medio de una locación de recursos de capacidad de una manera optima o efectiva.

2. PROGRAMACIÓN ESTOCÁSTICA

2.1 INTRODUCCIÓN A LA PROGRAMACIÓN ESTOCÁSTICA

Programación Estocástica, como su nombre lo indica, es programación matemática (bien sea lineal, entera, mixta entera o no lineal) pero con un elemento estocástico presente en los datos. Esto significa que al contrario de los modelos determinísticos donde se conoce (o se supone) por anticipado los datos o coeficientes, en Programación Estocástica estos datos son inciertos pero podemos tener una distribución de probabilidad asociada.

De esta manera Programación Estocástica maneja situaciones donde se tiene incertidumbre presente.

Cabe anotar que el termino Programación Estocástica es comúnmente usado para cubrir diferentes tipos de problemas. Todos estos contienen elementos probabilísticos en él, pero en algunos casos la similitud es muy poca.

Se consideran dos tipos de problemas de Programación Estocástica:

- Restricciones Probabilísticas
- Problemas con Recurso.

En este trabajo cuando se refiera a Programación Estocástica, solo se tendrán en cuenta los Problemas con recurso. Estos problemas son, en los cuales, algunas decisiones o acciones recursivas pueden ser tomadas después de conocer el elemento probabilístico. Una descripción probabilística y aproximada se asume disponible a la hora de la modelación, bajo la forma de distribuciones de probabilidad, densidades o cualquier otra medida de probabilidad.

De esta forma el conjunto de decisiones es dividido en dos grupos:

- Un número de decisiones que tienen que ser tomadas antes de conocer el valor del elemento probabilístico. Todas estas decisiones son llamadas

Decisiones de Primera Etapa y el periodo cuando estas son tomadas es conocido como la Primera Etapa.

- Un número de decisiones que pueden ser tomadas después de conocer el valor del elemento probabilístico. Estas son llamadas Decisiones de Segunda Etapa. El correspondiente periodo es conocido como la Segunda Etapa.

Es de resaltar que las definiciones de Primera y Segunda Etapa están únicamente ligadas al momento en que se toma la decisión, dependiendo el conocimiento o no del valor del elemento probabilístico. A su vez, estas etapas pueden tener secuencias de eventos o decisiones.

Dentro de esta clase de problemas, existen tres tipos diferentes de modelos:

- Modelos con Recurso Simple
- Modelos con Recurso Completo
- Modelos con Recurso Parcial

2.2 PROBLEMAS CON RECURSO SIMPLE

Es el mas simple modelo de esta clase de problemas y consta de dos etapas:

- En la primera etapa se toman las decisiones.
- En la segunda etapa se ve el comportamiento de los elementos probabilísticos del problema, pero no se es capaz de tomar decisiones a través del problema. Solo una variable será dependiente del elemento probabilístico para lograr que la restricciones sean respetadas.

En otras palabras, en la segunda etapa existe una variable que da un mayor nivel de flexibilidad, con el fin de preservar la factibilidad del problema, pero con un costo por esto. Estas decisiones que se tomen en la segunda etapa dependerán del comportamiento observado en los elementos probabilísticos.

Dada la solución de la primera etapa, es trivial el calculo de la variable recurso, ya que esta simplemente toma valores para preservar la factibilidad.

Tomando el ejemplo de planeación de producción de una empresa las variables de primera etapa podrían ser las cantidades a producir de cada producto en cada periodo y las variables de segunda etapa podrían ser las cantidades para tener en inventario de cada periodo dependiendo de los valores que tomen los diferentes elementos probabilísticos tenidos en cuenta a la hora de la modelación.

2.3 PROBLEMAS CON RECURSO COMPLETO

Es el mas completo de los problemas con recurso debido a que todas las decisiones incorporadas en el modelo pueden ser tomadas después de que se aclare la incertidumbre. Todas las decisiones después del primer periodo son decisiones recursivas y dependerán de los valores que tomen en un futuro los elementos probabilísticos.

Tomando el ejemplo de planeación de la producción tanto la cantidad a producir de cada producto como el inventario a tener, cambiaran dependiendo de los elementos estocásticos modelados.

2.4 PROBLEMAS CON RECURSO PARCIAL

Es una mezcla entre los dos anteriores casos, ya que en cualquier periodo, algunas de las decisiones son fijas y tomadas en la primera etapa (Independientes de los elementos estocásticos modelados), y otras son tomadas después de aclararse los elementos probabilísticos y son decisiones recursivas.

Tomando el ejemplo de la planeación de la producción se puede suponer que existe un proveedor que suple los mismos productos y que puede ser utilizado. Un problema con recurso parcial puede ser uno en el cual la decisión de las cantidades a producir de cada producto sea tomada en la primera etapa, mientras que las decisiones de cuanto comprar al proveedor y cuanto mantener en

inventario en cada periodo y por cada producto sean tomadas a medida que se aclaran los elementos estocásticos incorporados en el modelo.

2.5 INCORPORACIÓN DE LA INCERTIDUMBRE EN LA MODELACIÓN Y PROGRAMACIÓN

La existencia de la incertidumbre requiere que se modele la disponibilidad de la información con relación al tiempo y dejar claro que tipo de decisiones se pueden hacer o tomar en cada etapa de las que se vayan a modelar. Cabe anotar que cada modelo debe tener su propia forma de incorporar la incertidumbre, ya que al momento de escoger si una variable es de primera etapa o es recursiva, se debe hacer tomando en cuenta diferentes aspectos como si es útil tomar la decisión en una etapa posterior o también si es factible. No todas las variables se podrían tomar como variables recursivas si estas decisiones se deben tomar con demasiada anterioridad al momento de poder conocer el valor de los elementos probabilísticos modelados. De este modo en una decisión, en un problema multiperiodo, se deben tomar en cuenta todas las futuras incertidumbres, así como el acceso a la información. La forma especifica de futuras decisiones depende de todos los elementos asumidos, relativos a la información que esta disponible para el tomador de decisiones, cuando (respecto al tiempo) esta disponible y que ajustes (recursivos) están disponibles. La variación de estos elementos asumidos conllevan a diferentes modelos de Programación Estocástica del mismo problema. Previos trabajos han desarrollados modelos para la planeación de la producción y la capacidad bajo demanda incierta. Estas decisiones se expanden a múltiples productos, múltiples periodos y considera producción “hecha en casa” y outsorcing. Además, también considera mas decisiones de tipo táctico que de tipo estratégico (Escudero et al. 1993).

La Programación Estocástica puede incorporar desde un elemento probabilístico en el modelo, hasta gran numero de estos. A medida que se incorporan mas de

estos elementos el problema se convierte poco a poco en uno mucho mas complejo y complicado tanto al momento de la modelación como a la hora de ser resuelto.

Una de las maneras de involucrar la incertidumbre en modelos de Programación Estocástica es describir esta como un grupo de escenarios de los elementos probabilísticos.

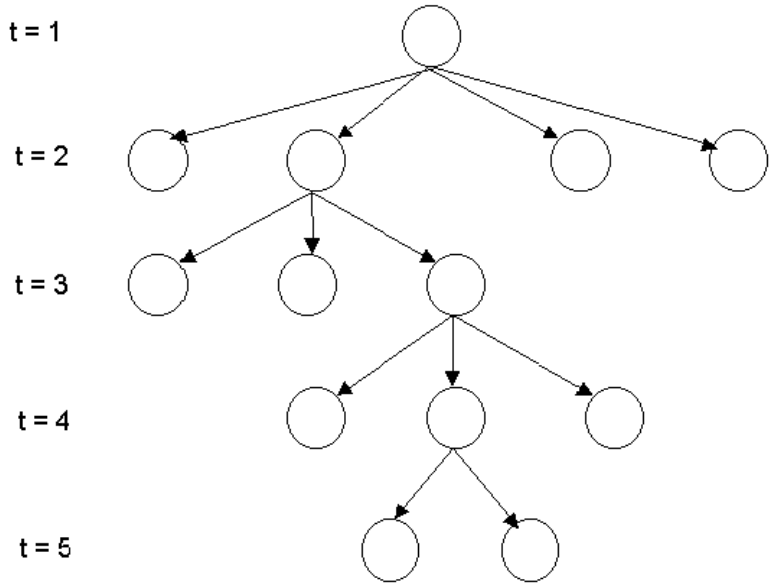

Figura 1. Árbol de Escenarios

La evolución de uno o múltiples elementos probabilísticos en la Programación Estocástica es mejor visualizada mediante el uso de Árboles de Escenarios. Como se muestra en la Figura 1, este tiene ramas y nodos. En cada nodo hay varias ramas para indicar posible futuros valores que se puedan producir a partir de ese nodo. Un escenario es sencillamente un recorrido que se pueda producir a partir del nodo raíz en el periodo 0 hasta un nodo hoja en el periodo T. De esta manera, un escenario incluye un nodo en cada periodo exactamente una vez. Diferentes escenarios poseen al menos un nodo diferente en su grupo de nodos. Los escenarios se crean basados en todos los elementos probabilísticos tomados en cuenta por el modelo. Cabe anotar que si se combinan todos esos procesos estocásticos al tiempo, se obtiene un masivo árbol de escenarios que representa

el incierto futuro de un proceso dinámico. A medida que se incluyen mas elementos probabilísticos en el modelo, el número de escenarios crece exponencialmente, convirtiendo la optimización del problema imposible de trabajar para cualquier algoritmo determinista. Esto es conocido como la *maldición de la dimensionalidad* (Frauendorfer y Schurle, 1996). Este problema se ha sorteado con el desarrollo de múltiples algoritmos basados en muestreo, los cuales permiten incorporar procedimientos del tipo de simulación junto con algoritmos de optimización (King y Wets, 1996 y Higle y Sen, 1989). Uno de los algoritmos frecuentemente utilizados, usa muestreo con simulación, que tiene la ventaja de ser aplicable no solo a problemas con procesos estocásticos conocidos, sino también a problemas en los que se tiene gran cantidad de datos históricos pero no se puede determinar el proceso estocástico del cual provienen. Esto se puede hacer mediante un algoritmo de agregación de escenarios mediante muestreo (Rockafellar y Wets, 1991).

Una parte importante ocurre al momento de implementar estos árboles de escenarios en los softwares utilizados para resolver los problemas. Un aspecto que sólo muy recientemente ha recibido atención, es la de apoyar el uso de la programación lineal estocástica. Para esto se han descubierto cuáles son las características que se requieren agregar a los sistemas de modelación existentes para poder incorporar este tipo de tecnología (Gassmann y Ireland, 1996). Existen dos reglas básicas al momento de modelar problemas de Programación Estocástica en los softwares utilizados para resolver problemas determinísticos:

1. Mantener una explicita estructura durante todo el código utilizado, sin importar cual sea el software.
2. Separar los escenarios e imponer restricciones de igualdad donde un grupo de escenarios deben tener el mismo valor.

Este tipo de restricciones son conocidas como las restricciones de no anticipo, las cuales son necesarias para tener políticas implementables.

Cabe anotar que implementar un árbol de escenarios como una estructura de un árbol (un grupo de variables de decisión por cada nodo en el árbol) es mas

eficiente. Sin embargo, separar el árbol y añadir las restricciones de no anticipo, agrega una estructura especial con la cual algunos métodos de solución explotan (Gassmann y Ireland, 1995).

3. MODELO GENERAL DE PLANEACIÓN DE LA PRODUCCIÓN UTILIZANDO PROGRAMACIÓN ESTOCÁSTICA

3.1 INTRODUCCIÓN

El modelo que se desarrollara en este trabajo corresponde a una generalización del conocido problema dinámico de tamaño de lote simple. Este modela sistemas en los cuales los productos son producidos intermitentemente con costos asociados de instalación y/o montaje (costos fijos incurridos cada vez que se va a producir al menos una unidad del producto) y almacenados en inventario con costos asociados de mantenimiento en inventario. El modelo toma en cuenta la capacidad de producción que debe ser compartida entre los diferentes productos a fabricar y entrega planes para coordinar el tiempo y el tamaño de las corridas de producción de cada uno de los productos en cada una de las etapas de manufactura en un sistema multiproducto. El modelo general aquí descrito puede ser aplicable para problemas de planeación de la producción y la programación de horarios de industrias como la de la producción de automóviles, aeronaves, computadores y aparatos electrónicos, por nombrar algunas.

3.2 MODELO

El modelo aquí desarrollado se trata de una generalización de Programación Mixta Entera de un sistema multiproducto, el cual es aplicable a un amplio número de problemas de Manufactura de Partes Discretas. En el pasado modelos como este tuvieron muy poca aplicabilidad en el mundo real debido al tamaño y la complejidad de estos, que los hacían difíciles de construir y optimizar, pero esta

dificultad técnica ha ido desapareciendo con el crecimiento de la tecnología de los computadores. Además, el uso practico de esta clase de modelos se ve beneficiados por diferentes investigaciones realizadas en cuanto a desarrollar algoritmos encaminados a la solución de modelos de Programación Estocástica.
De todas formas, la barrera para el uso de estos modelos es mas que una simple limitante matemática o técnica. Sistemas de Planeación de Requerimientos de Materiales han encontrado un amplio uso en la industria durante los últimos tiempos. Es por eso que este modelo fue pensado para que pudiera ser integrado a los otros aspectos que manejan los sistemas de Planeación de la Producción mas utilizados y anteriormente vistos.

La esencia de cualquier Sistema de Planeación de la Producción y de cualquier modelo matemático para problemas de manufactura de partes discretas, es la estructura del producto. Una típica estructura de producto se puede observar en la Figura 2. Cada caja significa un producto a ser fabricado y el número en esta es el índice de tal producto. Una flecha o cadena une cada producto a todos aquellos otros que lo requieren. Por ejemplo, el producto 17 esta unido a los productos 12, 8, 5 y 2. Cada producto esta asociado con exactamente un nivel en la estructura del producto, los cuales están indexados de 0 hasta L-1, donde los productos en el nivel 0 son llamados bienes o productos terminados y en el nivel L-1 son llamados materias primas. Con esto en mente, se puede establecer un modelo matemático de Programación Estocástica para optimizar sobre estructuras de producto generales.

Cabe anotar que en el modelo se decidió incluir ciertos parámetros rara vez incluidos en otros modelos. El primero de estos parámetros es el Factor de Producción, el cual se utiliza para los casos en los que se tiene estimada la fracción del total de productos que pasan el control de calidad y pueden ser utilizados para la fabricación de otros productos o para ser vendidos. Esto es muy útil para industrias como la de las impresiones, en la cual se tiene un número

estimado de producción extra para realizar pruebas y porque se sabe que las primeras impresiones no sirven para continuar con el proceso.

Otro de los parámetros incluidos es la Utilización de Alistamiento de la Capacidad, la cual convierte el modelo en uno mas realista, ya que el cambio de productos consume tiempo y en ocasiones este no es tenido en cuenta a la hora de la planeación.

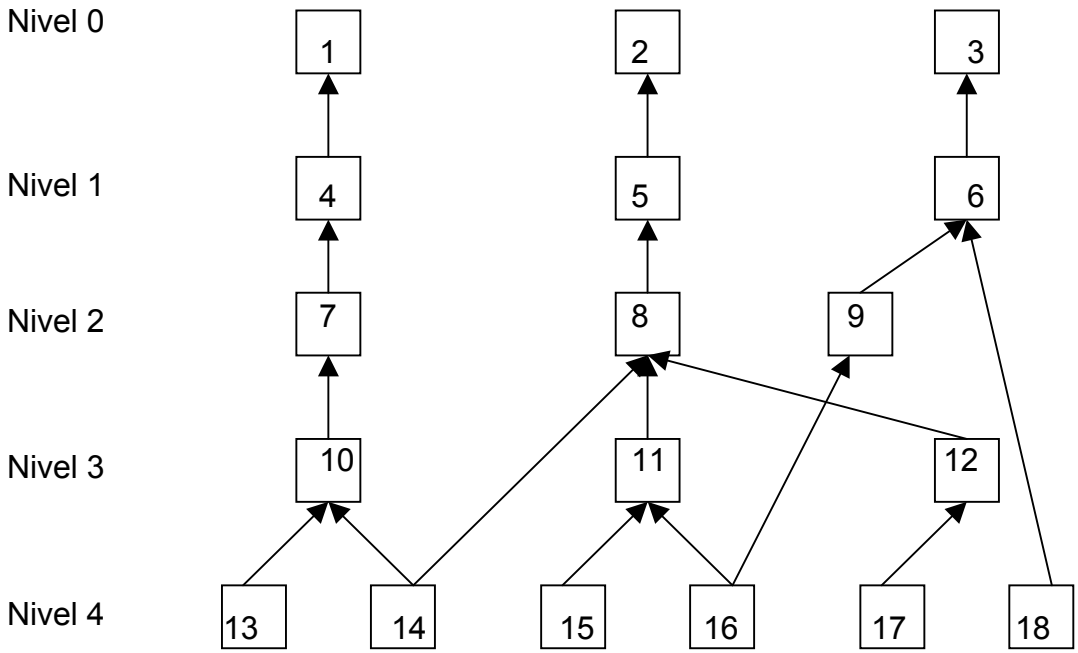

Figura 2. Típica Estructura de Producto

3.2.1 Elementos Probabilísticos

Es esencial anotar que en un Sistema de Manufactura de Partes Discretas como el que se pretende modelar, muchos pueden ser los factores o elementos probabilísticos envueltos. Sin embargo, al momento de pretender modelar el sistema y mas aún, de resolver el modelo propuesto, el tomar varios elementos probabilísticos conlleva a un problema de dimensionalidad a la hora de desarrollar y codificar los escenarios propuestos. Es por esto que la primera decisión tomada fue la de incorporar tan solo un elemento estocástico en el modelo.

El paso siguiente fue el de evaluar y decidir cual de los diferentes elementos probabilísticos podría ser el fundamental y el mas influyente al momento de la planeación de la producción. Se consideraron los casos de el Factor de Producción de cada Producto, el Costo de Mantenimiento en Inventario de cada Producto y la Demanda de cada Producto en cada Periodo de tiempo. Por un simple análisis de estudios anteriormente realizados y con base en la opinión de algunos expertos de la Universidad de Oklahoma, se llego a la conclusión de que el elemento probabilístico mas importante en un sistema como este, era la Demanda de cada Producto en cada periodo de tiempo, la cual es el parámetro mas variable y desconocido. Por esto tomarla como el elemento estocástico es la decisión lógica. Los escenarios y por supuesto el árbol de escenarios dependerá de esta forma única y exclusivamente de un elemento probabilístico, en este caso, la Demanda.

3.2.2 Índices

Lo primero en establecer son los índices que se usaran en el modelo.. Se han establecido cinco índices diferentes:

- i = 1,..., N. Productos
- j = 1,..., J. Productos
- t = 1,…, T. Periodos
- k = 1,…, K. Plantas
- s = 1,…, S. Escenarios

Estos se establecieron de tal forma que pudieran ser utilizados tanto a la hora de modelar como de programar. El índice Productos se encuentra dos veces porque la inclusión de este segundo índice facilita el trabajo a la hora de la codificación en el software.

3.2.3 Parámetros de Entrada

El modelo incluye o necesita los siguientes parámetros de entrada:

- h_i = Costo de Mantenimiento en Inventario ($ por unidad de producto i)
- c_i = Costo de Alistamiento y/o Montaje ($ por alistamiento del producto i)
- co_{kt} = Costo de Tiempo Extra ($ por unidad de capacidad en el periodo t en la planta k)
- L_i = Tiempo de Reorden para el Producto i (Este tiempo en el modelo es en realidad de producción)
- f_i = Factor de Producción del Producto i (fracción)
- a_{ij} = Número de unidades del Producto i requeridos para la fabricación del Producto j.
- r_{its} = Demanda del Producto i, en el Periodo t, bajo el Escenario s
- b_{ik} = Rata de Utilización de Capacidad del Producto i en la Planta k (unidades de capacidad por unidades de producto)
- su_{ik} = Utilización de Alistamiento de la Capacidad de la Planta k por el Producto i (unidades de capacidad)
- CAP_{kt} = Capacidad de la Planta k, en el periodo t (unidades de capacidad)
- q_{it} = Tope Máximo de la Producción del Producto i, en el periodo t (unidades de producto)
- p_s = Probabilidad del Escenario s

3.2.4 Variables de Decisión

Existen cuatro variables de decisión en el modelo desarrollado. Dependiendo de si se desea utilizar el modelo con Recurso Simple o el modelo con Recurso Completo, estas variables dependen o no del elemento probabilístico, es decir, dependerán o serán indexados con los escenarios.

Para el caso del modelo con Recurso Simple las Variables de Decisión serán:

- y_{its} = Inventario de Producto i, al final del Periodo t, bajo el Escenario s.
- d_{it} = 1 si el producto se fabricará en el periodo t y 0 de lo contrario.
- o_{kt} = Capacidad en Tiempo Extra a Utilizar en la Planta k en el periodo t.
- x_{it} = Producción del Producto i comenzado en el periodo t

Para el caso del modelo con Recurso Completo las Variables de Decisión serán:

- y_{its} = Inventario de Producto i, al final del Periodo t, bajo el Escenario s.
- d_{its} = 1 si el Producto i se fabricará en el periodo t, bajo el Escenario s y 0 de lo contrario.
- o_{kts} = Capacidad en Tiempo Extra a Utilizar en la Planta k en el periodo t, bajo el Escenario s.
- x_{its} = Producción del Producto i comenzado en el periodo t, bajo el Escenario s.

En el modelo se decidió incluir la variable binaria d para poder tener en cuenta el parámetro de la utilización consumida al momento del alistamiento. Otra variable de decisión fue la cantidad de cada producto a mantener en inventario en cada periodo de tiempo porque es una de las principales decisiones al momento de planear la producción de un sistema, al igual que la decisión de cuanto producir de cada producto en cada periodo de tiempo.

3.2.5 Función Objetivo

La Función Objetivo desarrollada representa los Costos de Mantenimiento en Inventario de cada Producto en cada Periodo, de Alistamiento de Producción de cada Producto en cada Periodo y de Tiempo extra de Producción de cada Planta en cada Periodo que se busca Minimizar. Los costos de producción como tales no se incluyeron porque no se creyó necesario ni conveniente, ya que el modelo no pretende crear un Mix de Producto que minimice esto, sino un plan de Producción para cumplir con las demandas estipuladas.

La Función Objetivo esta definida de la siguiente manera:

$$MINIMIZAR \sum_{s=1}^{S} p_s \left[\sum_{i=1}^{N} \sum_{t=1}^{T} \left(h_i * y_{its} + c_i * d_{it} \right) \right] + \sum_{k=1}^{K} \sum_{t=1}^{T} \left(co_{kt} * o_{kt} \right)$$

Cabe anotar que al igual que en las siguientes definiciones, el modelo será presentado con Recurso Simple, es decir solo la Variable Inventario (Y) será indexada con los Escenarios.

3.2.6 Restricciones de Balanceo de Inventarios

Estas restricciones representan que cada inventario final de cada Producto i en cada Periodo t, es mayor o igual al inventario inicial mas la Producción Neta (teniendo en cuenta el Factor de Producción) del Producto i en el Periodo t – L_i menos la demanda interna y externa del producto en ese periodo. Aquí el tiempo de Reorden L_i debe igualar el mínimo tiempo de reorden necesario para producir o de lo contrario adquirir el Producto.
Las restricciones son de la siguiente manera:

$$y_{i,t-1,s} + f_i * x_{i,t-Li} - y_{it} - \sum_{j=1}^{J} a_{ij} * x_{jt} \geq r_{its}$$

Para todo i = 1,..., N, t = 1,...,T s = 1,...,S

3.2.7 Restricciones de Capacidad

Este tipo de restricciones consiste en que la capacidad consumida para la producción de cada uno de los productos a fabricar en cada periodo de tiempo y en cada planta, mas la capacidad consumida al momento del alistamiento de la producción, sea menor o igual a la capacidad mas el tiempo extra de cada planta en cada periodo. La importancia de estas restricciones radica en que si se viola, se estaría planeando fabricación de productos sin tener los recursos (Capacidad) necesarios para hacerlo. Las restricciones quedan así definidas:

$$\sum_{i=1}^{N}(b_{ik} * x_{it} + su_{ik} * d_{it}) - o_{kt} \leq CAP_{kt}$$

Para todo k = 1,..., K, t = 1,...,T

3.2.8 Restricciones de Tope Máximo de Producción

El objetivo de estas restricciones es que para cada producto y bajo cada periodo, no se exceda el tope máximo de producción estipulado. Las restricciones quedan definidas de la siguiente manera:

$$x_{it} - q_{it} * d_{it} \leq 0$$

Para todo i = 1,..., N, t = 1,...,T

3.2.9 Otras Restricciones

Las restricciones restantes son las de no negatividad, la de la variable binaria y las de no anticipo. Las de no negatividad hacen que las variables solo puedan tomar valores mayores o iguales a 0. Las de la variable binaria, que estas solo puedan tomar valores 0 o 1. Las de no anticipo hacen que las variables no tomen valores diferentes cuando el árbol de escenarios muestre que deben tener valores iguales en determinado nodo. Estas ultimas no tienen manera de ser escritas generalmente, ya que dependen de cada árbol de escenarios propuesto. Las restricciones quedan definidas de esta manera:

d_{it} = 0 ó 1 $y_{its} \geq 0$ $x_{it} \geq 0$ $o_{kt} \geq 0$

Para todo i = 1,..., N, t = 1,...,T k = 1,...,K s = 1,...,S

Un resumen de los modelos matemático desarrollados con Recurso Simple y con Recurso Completo se muestra en el Anexo A.

4. MODELOS DE EJEMPLOS NUMÉRICOS ALEATORIOS PROGRAMADOS Y RESUELTOS

4.1 INTRODUCCIÓN

En esta parte del proyecto se propusieron tres diferentes clases de ejemplos numéricos. En estos ejemplos se decidió cierto numero de Plantas, Productos, Periodos y Escenarios, pero los parámetros que necesita el modelo para ser corrido fueron generados aleatoriamente en el mismo programa que se resuelven los ejemplos (Xpress-MP de Dash Optimization). Estos fueron programados para los modelos con Recurso Simple y con Recurso Completo.

En total se crearon seis códigos para el software Xpress-MP de Dash Optimization; tres para el modelo con Recurso Simple y tres para el modelo con Recurso Completo. Estos modelos por generar en cada corrida diferentes parámetros, generan igualmente diferentes soluciones.

4.2 PRIMER EJEMPLO ALEATORIO CON RECURSO SIMPLE

El primer modelo aleatorio desarrollado fue pensado para tres escenarios. A partir de esto se decidió el número de plantas, productos y periodos que se tendrían en cuenta. Esto se hizo de una manera casi aleatoria y solo por desarrollar un primer ejemplo fácil de resolver, por lo cual se tuvieron en cuenta pocos escenarios. La mayoría de los datos de este ejemplo fueron tomados de igual manera. El resultado para este primer ejemplo fue de:

- 4 Productos
- 4 Periodos
- 3 Plantas
- 3 Escenarios

Teniendo esta información se procedió a diseñar una estructura del producto para estos cuatro productos. Se decidió dejar un solo producto final como tal y tres subproductos. El producto final necesitaría de los otros tres productos y uno de estos tres necesitaría de los otros dos. La estructura del producto resultante con la cantidad necesaria de cada uno de estos se muestra en la Figura 3.

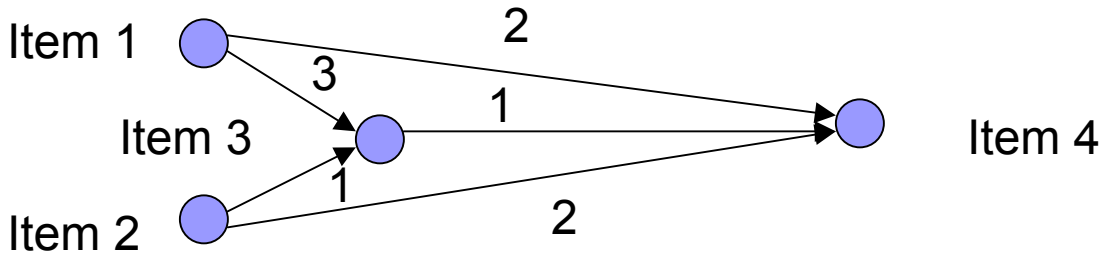

Figura 3. Estructura de Producto del Primer Ejemplo

Una vez tenida la estructura del producto se procedió a poner tiempos de reorden para cada uno de estos productos. Se decidió tomar como tiempo de reorden igual a un periodo para los productos 1, 2 y 3 y de dos periodos para el producto 4.
Para la demanda, se decidió crear un árbol de escenarios en donde cada uno de estos tuviera la misma probabilidad. Este solo tendría un nodo común para los tres escenarios el cual seria el nodo raíz. El árbol de escenarios resultante se muestra en la Figura 4.

En este árbol se puede observar que los tres escenario en el primer periodo tendrán la misma demanda, pero en periodos siguientes serán diferentes. Este ejemplo fue codificado en el software Xpress-MP de Dash Optimizaction. Para cada uno de los parámetros de entrada se procedió a generar estos de manera

aleatoria utilizando la función random. Los máximos y mínimos para cada parámetro generados se encuentran en la Tabla 1.

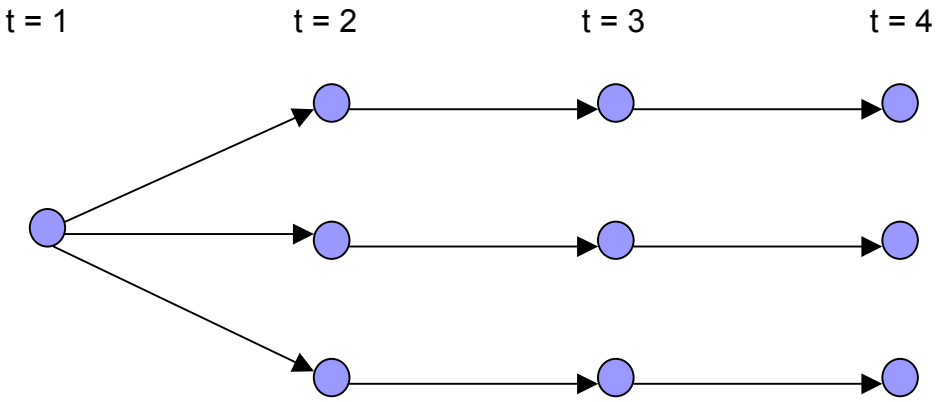

Figura 4. Árbol de Escenarios para el Primer Ejemplo

Tabla 1. Parámetros del Primer Ejemplo

Parámetro	Mínimo	Máximo
Demandas Primer Periodo	100	200
Demandas Otros Periodos	150	200
Costo de Alistamiento	500	550
Costo de Tiempo Extra	30	40
Utilización de Alistamiento de la Cap.	3	4
Rata de Utilización de Capacidad	0.15	0.2
Costo de Mantenimiento en Inventario	2	3
Inventario Inicial	600	630
Capacidad de las Plantas	30	37
Tope Máximo de Producción	500	520
Factor de Producción	0.8	1

El código para este ejemplo, al igual que para los demás ejemplos aleatorios, puede ser visto en el Anexo B.

Una vez se corre el código, este suministra la información de cuanto debe ser producido en cada periodo y por cada producto, cuanta capacidad extra debe ser utilizada en cada periodo y en cada planta y cuanto debe ser la cantidad a almacenar en inventario de cada producto en cada periodo y bajo cada escenario. Una solución corrida con los correspondientes parámetros de entrada para este primer ejemplo, al igual que para los demás ejemplos aleatorios, puede ser vista en el Anexo C.

4.3 SEGUNDO EJEMPLO ALEATORIO CON RECURSO SIMPLE

El segundo modelo aleatorio desarrollado fue pensado para cuatro escenarios. A partir de esto se decidió el número de plantas, productos y periodos que se tendrían en cuenta. Esto se hizo pensando en utilizar este ejemplo como base para resolver los futuros ejemplos no aleatorios, así que estos fueron los mismos que el primer ejemplo no aleatorio del cual ya se tenían los datos. El resultado para este primer ejemplo fue de:

- 3 Productos
- 5 Periodos
- 2 Plantas
- 4 Escenarios

Teniendo esta información se procedió a diseñar una estructura del producto para los tres productos. Se decidió dejar un solo producto final al igual que el anterior ejemplo y dos subproductos porque esta era la estructura dada para los ejemplos no aleatorios. En esta, el producto final necesita de los otros dos productos. La estructura del producto resultante con la cantidad necesaria de cada uno de estos se muestra en la Figura 5.

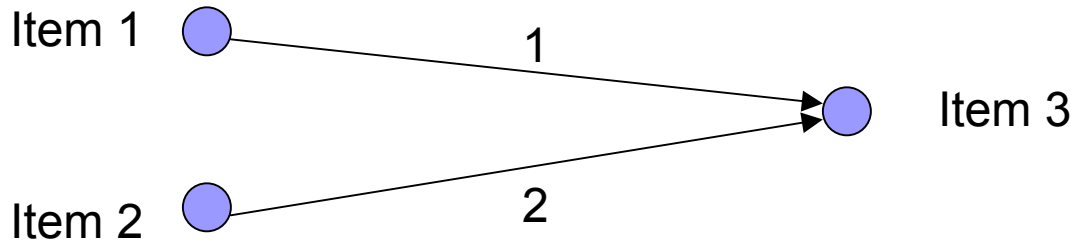

Figura 5. Estructura de Producto del Segundo Ejemplo

Una vez tenida la estructura del producto se procedió a poner tiempos de reorden para cada uno de estos productos. Se decidió tomar como tiempo de reorden igual a un periodo para todos los productos al igual que en los ejemplos no aleatorios dados.

Para la demanda, se decidió crear un árbol de escenarios en donde cada uno de estos tuviera la misma probabilidad. Este tendría un nodo común para los cuatro escenarios el cual seria el nodo raíz y otros dos comunes para dos escenarios. Esta vez la demanda será igual para todos los escenarios en el periodo uno e igual para escenarios 1 y 2 en el periodo dos y 3 y 4 en el periodo dos. El árbol de escenarios resultante se muestra en la figura 6.

Este ejemplo fue codificado en el software Xpress-MP de Dash Optimizaction. Para cada uno de los parámetros de entrada se procedió a generar estos de manera aleatoria utilizando la función random. Los máximos y mínimos para cada parámetro generados son los mismos del ejemplo anterior y se encuentran en la Tabla 1.

El código para este ejemplo puede ser visto en el Anexo B al igual que el anterior ejemplo.

Una vez se corre el código, este suministra la información de cuanto debe ser producido en cada periodo y por cada producto, cuanta capacidad extra debe ser utilizada en cada periodo y en cada planta y cuanto debe ser la cantidad a almacenar en inventario de cada producto en cada periodo y bajo cada escenario. Una solución corrida con los correspondientes parámetros de entrada puede ser vista en el Anexo C al igual que el anterior ejemplo.

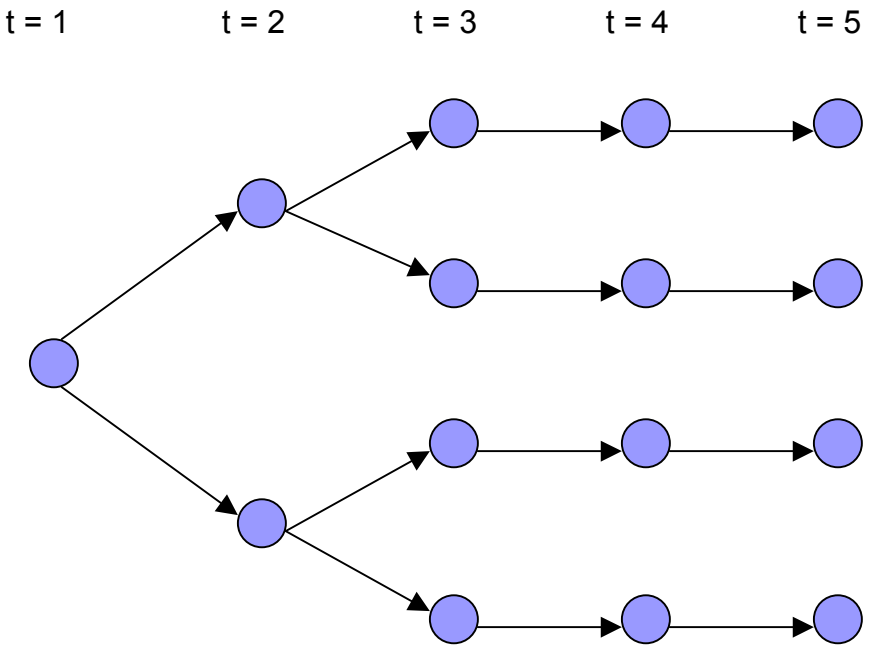

Figura 6. Árbol de Escenarios para el Segundo Ejemplo

4.4 TERCER EJEMPLO ALEATORIO CON RECURSO SIMPLE

El tercer modelo aleatorio desarrollado fue pensado para seis escenarios. A partir de esto se decidió el número de plantas, productos y periodos que se tendrían en cuenta. Estos fueron, pensando a futuro, los mismos que para el segundo ejemplo no aleatorio del cual se tenían los datos. El resultado para este primer ejemplo fue de:

- 3 Productos
- 5 Periodos
- 2 Plantas
- 6 Escenarios

El siguiente paso era crear una estructura de producto, pero se decidió dejar la misma del ejemplo dos, la cual es la misma del segundo ejemplo no aleatorio. La estructura del producto se muestra en la Figura 5.

Los tiempos de reorden se dejaron iguales, es decir, un periodo para cada uno de los tres productos a fabricar como en los ejemplos no aleatorios.

Para la demanda, se decidió crear un árbol de escenarios en donde cada uno de estos tuviera la misma probabilidad. Este tendría un nodo común para los seis escenarios el cual seria el nodo raíz, dos nodos, cada uno compartido por tres escenarios y otros dos nodos compartido entre dos escenarios. Esta vez la demanda será igual para todos los escenarios en el periodo uno, igual para escenarios 1, 2 y 3 en el periodo dos y 4, 5 y 6 en el periodo dos y para escenarios 1 y 2 en el periodo tres y escenarios 4 y 5 en el mismo periodo. El árbol de escenarios resultante se muestra en la Figura 7.

Este ejemplo fue codificado en el software Xpress-MP de Dash Optimizaction. Para cada uno de los parámetros de entrada se procedió a generar estos de manera aleatoria utilizando la función random. Los máximos y mínimos para cada parámetro generados son los mismos de los ejemplos anteriores y se encuentran en la Tabla 1.

El código para este ejemplo puede ser visto en el Anexo B al igual que los anteriores ejemplos.

Una vez se corre el código, este suministra la información de cuanto debe ser producido en cada periodo y por cada producto, cuanta capacidad extra debe ser utilizada en cada periodo y en cada planta y cuanto debe ser la cantidad a almacenar en inventario de cada producto en cada periodo y bajo cada escenario. Una solución corrida con los correspondientes parámetros de entrada puede ser vista en el Anexo C al igual que los anteriores ejemplos.

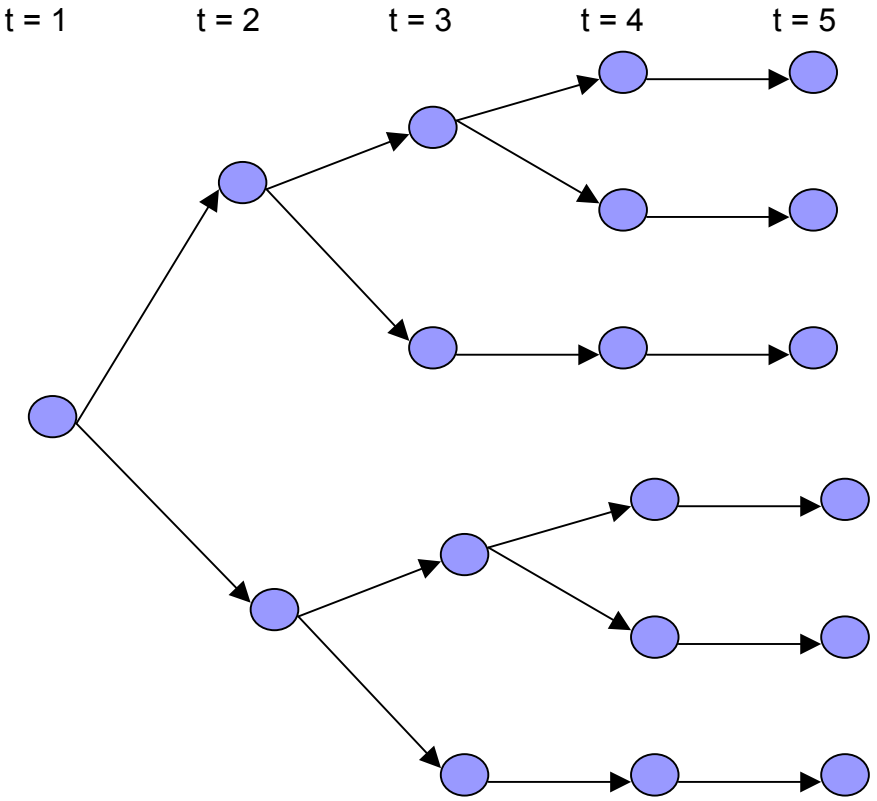

Figura 7. Árbol de Escenarios para el Tercer Ejemplo

4.5 EJEMPLOS ALEATORIOS CON RECURSO COMPLETO

Para los ejemplos aleatorios con recurso completo se utilizaron los mismos ejemplos básicos que se usaron para los modelos con recurso simple. Los códigos hechos en el software Xpress-MP de Dash Optimization fueron modificados para resolver los ejemplos con recurso completo. De este modo fueron obtenidos tres

nuevos códigos e igualmente se procedió a resolver estos ejemplos en el software antes mencionado.

Con estos nuevos resultados se obtiene, igualmente, un programa de las cantidades a producir de cada producto y cuanto tiempo extra utilizar en cada planta a lo largo de los periodos, pero ahora estos valores también dependerán e irán cambiando según se comporte la demanda. Los códigos para los tres ejemplos pueden ser vistos en el Anexo B y los respectivos resultados en el Anexo C.

4.6 RESULTADOS DE LOS EJEMPLOS

Cada código desarrollado utiliza parámetros de entrada de tipo aleatorio, con lo cual cada corrida conlleva a resultados diferentes. Los códigos se corrieron varias veces cada uno y mostraron un buen desempeño. La mayoría de corridas entregaron resultados factibles y óptimos. Esto se concluyó mediante la comprobación de las condiciones de Karush-Kuhn-Tucker para optimalidad (véase Anexo D).

En el Anexo C se muestra tan solo un ejemplo de los resultados obtenidos por cada código. Otros resultados pueden ser vistos corriendo el código en el software. Cabe anotar que algunas corridas para los códigos con recurso completo produjeron ejemplos en los cuales la solución divergía. Esto fue a causa de la formulación aleatoria de los parámetros de entrada como las demandas de los artículos, que hacían que en algunos casos no existiera una región factible para el problema. En otros casos sencillamente el modelo fue demasiado complejo para el software y sus algoritmos de solución, por lo cual los resultados entregados no fueron factibles.

Todas las soluciones obtenidas que no divergían se consideraron implementables ya que no producían valores mas allá de los normalmente trabajados. Estas

soluciones sirvieron como referencia para comprobar que los códigos trabajaban bien y poder pasar a la siguiente etapa de obtener resultados para un ejemplo numérico no aleatorio, los cuales se compararan con los obtenidos para el mismo utilizando Programación Lineal.

5. MODELOS DE EJEMPLOS NO ALEATORIOS PROGRAMADOS Y RESUELTOS

5.1 INTRODUCCIÓN

En este capitulo se mostraran dos ejemplos numéricos no aleatorios. Estos datos fueron presentados a la Universidad de Oklahoma y fueron utilizados en este trabajo con el fin de comparar los resultados obtenidos con los que se obtendrían utilizando Programación Lineal.

Estos ejemplos tienen como base algunos de los códigos que se utilizaron en el capitulo anterior, dado que fueron desarrollados pensando en crear códigos que diesen buenos resultados para los ejemplos no aleatorios de este capitulo.

Los códigos desarrollados en este capitulo dan solución a cada uno de los ejemplos numéricos no aleatorios originando un plan de producción que consta de las cantidades a producir y de las cantidades a mantener en inventario de cada producto y en cada periodo.

5.2 PRIMER EJEMPLO NO ALEATORIO

El primer ejemplo es parecido en estructura al segundo ejemplo aleatorio, ya que consta de tres productos que tienen la misma estructura del producto que el ejemplo aleatorio. Son cinco los periodos para los cuales se va a planear la producción y la demanda esta dada en un árbol de escenarios con cuatro

escenarios. Al igual que en los códigos desarrollados también se cuenta con dos plantas para la producción.

La demanda de los productos para el ejemplo se encuentra entre 180 y 230 para el producto número uno, entre 170 y 210 para el producto número dos y entre 200 y 250 para el producto tres. Esta demanda sigue un árbol de escenarios como el del ejemplo aleatorio número dos.
Otros parámetros del modelo como el costo de alistamiento es de 2000 para los dos primeros productos y de 3000 para el tercero. El costo de mantenimiento en inventarios es de 15 para los dos primeros y de 20 para el tercer producto. El tiempo de reorden es de un periodo para los tres productos. El ejemplo tiene en total 135 parámetros de entrada.
Para el modelo con recurso simple existen 115 variables (15 de cantidades a producir, 15 binarias para producción, 10 de tiempo extra, 60 de inventario y 15 mas para hacer fácil la codificación) y 115 restricciones (60 de demanda, 10 de capacidad, 15 de tope máximo de producción, 15 de no anticipo y 15 auxiliares).
Para el modelo con recurso completo existen 280 variables (60 de cantidades a producir, 60 binarias para producción, 40 de tiempo extra, 60 de inventario y 60 mas para hacer fácil la codificación) y 280 restricciones (60 de demanda, 40 de capacidad, 60 de tope máximo de producción, 60 de no anticipo y 60 auxiliares).

El código desarrollado para el modelo con recurso simple para resolver el ejemplo en el software Xpress-MP de Dash Optimization se encuentra en el Anexo E. Este da el plan de producción teniendo en cuenta que solo la variable inventario depende de cada escenario. El resultado del código desarrollado para el modelo con recurso simple da un completo detalle de los datos o parámetros de entrada utilizados y puede ser visto en el Anexo F.
El código desarrollado para el modelo con recurso completo se encuentra, igualmente, en el Anexo E.

5.3 SEGUNDO EJEMPLO NO ALEATORIO

El segundo ejemplo no aleatorio es parecido en los parámetros al primero, pero la demanda no es dada por cuatro escenarios, sino por seis, haciéndolo mas complejo al momento de su solución. Este árbol de escenarios es igual en estructura al desarrollado en el tercer ejemplo aleatorio y los valores que toma son parecidos pero no iguales al ejemplo anteriormente desarrollado.
Para el primer producto los valores de la demanda están entre 140 y 220 para el segundo producto están entre 120 y 180 y para el tercer producto estos están entre 230 y 300. Un detalle de las parámetros de entrada pueden ser visto en el anexo 23.El ejemplo tiene en total 167 parámetros de entrada.
Para el modelo con recurso simple existen 145 variables (15 de cantidades a producir, 15 binarias para producción, 10 de tiempo extra, 90 de inventario y 15 mas para hacer fácil la codificación) y 163 restricciones (90 de demanda, 10 de capacidad, 15 de tope máximo de producción, 33 de no anticipo y 15 auxiliares).
Para el modelo con recurso completo existen 420 variables (90 de cantidades a producir, 90 binarias para producción, 60 de tiempo extra, 90 de inventario y 90 mas para hacer fácil la codificación) y 462 restricciones (90 de demanda, 60 de capacidad, 90 de tope máximo de producción, 132 de no anticipo y 90 auxiliares).

Igualmente se desarrollaron dos códigos, uno para el modelo con recurso simple y otro para el modelo con recurso completo que se encuentran en el Anexo E. Los resultados del primer código se muestran en el Anexo F.

Aunque en este capitulo no se tocaran a fondo los resultados obtenidos, cabe anotar que para los códigos con recurso completo no se obtuvieron resultados porque el ejemplo fue complejo para el software. Se llego a la conclusión de que la estructura y los parámetros del ejemplo hicieron que los métodos de solución del software explotaran. Los códigos desarrollados para los modelos con recurso completo produjeron resultados que no eran factibles, así que se procedió a buscar posibles causas. Una podría ser que la región factible fuera nula, pero esta

fue descartada porque para los modelos con recurso simple si se obtuvieron resultados factibles y óptimos. La segunda podría ser que el código estuviera mal desarrollado, pero también fue descartado ya que para algunos ejemplos aleatorios se obtuvieron resultados. La tercera podría ser que los métodos de solución que utiliza el software explotan para la estructura de este ejemplo, la cual fue asumida como la verdadera causa por descarte.
Como estos códigos no produjeron resultados satisfactorios, no se pudieron comparar estos con los de Programación Lineal.

Para poder obtener resultados de este modelo se debe entrar a desarrollar o codificar un algoritmo de solución en otro lenguaje computacional o software. Esto no se hizo en este proyecto porque se considero por fuera de los márgenes que este tenia. Desde un comienzo se decidió no involucrarse con los algoritmos de solución debido a la complejidad que estos podrían traer.
De todas formas se deja como recomendación probar los modelos con recurso completo desarrollados en el software GAMS, el cual es otro software reconocido para la optimización de modelos matemáticos.

6. COMPARACIÓN DE RESULTADOS CON LOS OBTENIDOS UTILIZANDO PROGRAMACIÓN LINEAL

6.1 INTRODUCCIÓN

En este capitulo se compararán los resultados obtenidos utilizando los modelos aquí desarrollados de Programación Estocástica, con los obtenidos utilizando modelos de Programación Lineal.

La manera de comparar estos resultados será con una serie de datos obtenidos aleatoriamente de la demanda de cada uno de los productos y de esta manera comparar los costos que acarrearía una empresa al utilizar los dos sistemas de planeación de la producción.

6.2 MODELO DE PROGRAMACIÓN LINEAL

Este modelo se desarrolla para comparar los resultados obtenidos mediante Programación Estocástica. Es parecido en forma a los desarrollados en este trabajo, pero en este la demanda se toma como un parámetro determinístico. El modelo tiene las mismas variables de decisión, pero estas no dependen del comportamiento de la demanda a lo largo del tiempo, sino que la decisión se toma con anterioridad.

6.2.1 Función Objetivo

La función objetivo es muy parecida a la anteriormente desarrollada y mide exactamente los mismos costos. Esta función queda definida de la siguiente forma:

$$MINIMIZAR \sum_{i=1}^{N} \sum_{t=1}^{T} (h_i * y_{it} + c_i * d_{it}) + \sum_{k=1}^{K} \sum_{t=1}^{T} (co_{kt} * o_{kt})$$

6.2.2 Variables de Decisión

Las variables de decisión quedan definidas de la misma forma que en los anteriores modelos, pero ninguna depende de la demanda. Estas son:

- y_{it} = Inventario de Producto i, al final del Periodo t.
- d_{it} = 1 si el producto se fabricará en el periodo t y 0 de lo contrario.
- o_{kt} = Capacidad en Tiempo Extra a Utilizar en la Planta k en el periodo t.
- x_{it} = Producción del Producto i comenzado en el periodo t

6.2.3 Restricciones de Demanda

Esta restricción es para hacer que lo que se produzca y tenga sea mayor que la demanda. Esta queda definida de la siguiente manera:

$$y_{i,t-1} + f_i * x_{i,t-Li} - y_{it} - \sum_{j=1}^{J} a_{ij} * x_{jt} \geq r_{it}$$

Para todo i = 1,..., N, t = 1,...,T

6.2.4 Restricciones de Capacidad

Esta se crea para que no se pueda producir mas cuando no existe capacidad en las plantas. Se define de la siguiente manera

$$\sum_{i=1}^{N}(b_{ik} * x_{it} + su_{ik} * d_{it}) - o_{kt} \leq CAP_{kt}$$

Para todo t = 1,...,T k = 1,...,K

6.2.5 Restricciones de Tope Máximo de Producción

Esta se crea con el fin de que no se produzca mas de cada tipo de producto que cierta cantidad tope. Las restricciones quedan definidas así:

$$x_{it} - q_{it} * d_{it} \leq 0$$

Para todo i = 1,..., N, t = 1,...,T

6.2.6 Restricciones de No Negatividad

Estas hacen que las variables no tomen valores por debajo de cero. Están definidas así:

d_{it} = 0 ó 1 $y_{it} \geq 0$ $x_{it} \geq 0$ $o_{kt} \geq 0$

Para todo i = 1,..., N, t = 1,...,T k = 1,...,K

6.2.7 Datos del Modelo

Para el modelo se utilizaron los mismo datos que se usaron en los dos ejemplos numéricos no aleatorios. De esta manera se crearon dos ejemplos determinísticos cuyos resultados se compararán con los obtenidos mediante Programación

Estocástica. La demanda se halló mediante una simple operación que consistió en hallar la demanda media ponderada para cada producto, en cada periodo y para cada ejemplo. De esta manera los datos de demanda se redujeron considerablemente al igual que el numero de variables y restricciones.
En total los modelos de Programación Lineal están conformados por 86 parámetros de entrada, 70 variables de decisión (15 de cantidades a producir, 15 binarias para producción, 10 de tiempo extra, 15 de inventario y 15 mas para hacer fácil la codificación) y 55 restricciones (15 de demanda, 10 de capacidad, 15 de tope máximo de producción y 15 auxiliares).

6.3 RESULTADOS DE LOS MODELOS DE PROGRAMACIÓN LINEAL

Para estos modelos, igualmente se crearon dos códigos que se resolvieran en el software Xpress-MP de Dash Optimization. Estos códigos son parecidos a los anteriormente desarrollados pero se debe tener en cuenta que debido a su simplicidad y a su menor tamaño, la solución es encontrada mas fácilmente.

Para resolver estos ejemplos de forma determinística, se expandieron los códigos desarrollados para los modelos con recurso simple. En este código se añadió un índice mas para los escenarios en donde se resolvería el ejemplo de forma determinística. En este índice número cinco o siete (dependiendo del ejemplo), las demandas para cada producto, en cada periodo y en cada ejemplo, son las medias ponderadas halladas anteriormente. A su vez, se tiene una función objetivo nueva que se piensa minimizar, la cual es independiente de los escenarios.

Los códigos para resolver los dos ejemplos pueden ser vistos en el Anexo G. Las soluciones dadas por estos dos códigos pueden ser vistas a su vez en el Anexo H.

Estos resultados dan un plan de producción que consta de las cantidades a producir y a mantener en inventario para cada producto y en cada periodo de tiempo. También se muestra la cantidad de tiempo extra a utilizar durante cada periodo y en cada planta.

6.4 METODOLOGÍA DE LA COMPARACIÓN DE RESULTADOS

La comparación de los resultados obtenidos mediante las dos técnicas, se hará utilizando una serie de horizontes de tiempo cada uno con la cantidad demandada para cada producto. Con esto se mirará cual es la cantidad que queda en inventario de cada producto. Como es posible que en algunos casos la cantidad demandada no sea alcanzada debido a los valores que tome esta, se impondrá una multa o castigo por cada producto faltante. En total se usarán 30 horizontes para cada ejemplo.

Tras un debate con uno de los directores del proyecto, el Dr. Suleyman Karabuk, se decidió que este castigo fuera igual al costo de cada producto de mantener en inventario durante un periodo de tiempo, es decir de 15 para los productos uno y dos y de 20 para el producto tres. De esta manera a lo largo de un horizonte de tiempo se incurre en cuatro diferentes costos, los cuales son, el de utilización de tiempo extra y el de producción en determinado periodo (binario) que ya están dados en la función objetivo, además de un costo de mantenimiento en inventario (diferente al de la función objetivo por su dependencia de las demandas) y un costo de incumplimiento de la demanda (el cual no estaba incluido en la función objetivo). La suma de estos costos para cada tipo de técnica (Programación Lineal y Estocástica) se compararán para concluir cual de las dos clases de modelos tiene un mejor comportamiento.

Todo este proceso se hizo mediante una hoja de calculo de Excel, con la cual se hallaba cuanto era el inventario en cada periodo y en caso de faltante, cuanto era este. Esta se utilizo para el ejemplo de Programación Lineal y para cada escenario en cada ejemplo. Con estos resultados se procedió a hallar el verdadero

costo para cada modelo y posteriormente se procedió a hallar la diferencia entre estos.

Una muestra de los horizontes utilizados para el primer ejemplo se encuentra en el Cuadro 1.

Cuadro 1. Muestra de los Horizontes Usados en la Comparación

		Periodos				
Horizonte	**Producto**	**1**	**2**	**3**	**4**	**5**
1	**1**	209	214	206	201	207
	2	174	181	177	180	178
	3	233	241	247	243	239
2	**1**	196	190	193	187	189
	2	189	193	206	213	207
	3	227	215	213	216	221
3	**1**	207	213	221	223	220
	2	183	190	195	199	203
	3	227	241	137	236	241
4	**1**	195	187	183	180	184
	2	176	175	173	169	172
	3	229	220	207	209	205

La comparación se hará de dos maneras diferentes. La primera como se dijo anteriormente y la segunda comparando no solo con los costos esperados de Programación Estocástica, sino con cada escenario utilizado.

6.5 RESULTADOS OBTENIDOS EN LA COMPARACIÓN

Se comenzó con el procedimiento para el primer ejemplo no aleatorio. Una vez hallados los verdaderos costos en los que se incurriría utilizando los dos tipos de modelos, para cada horizonte de demandas se procedió a hallar la diferencia. Estas diferencias fueron graficadas en Excel y se pueden observar en la Figura 8. Se hallaron 30 diferencias y estas produjeron un promedio de 5098 unidades monetarias (en este caso dólares) de ahorro al utilizar Programación Estocástica en la planeación de la producción. Igualmente, al comprarse el resultado obtenido con Programación Lineal con el resultado de cada uno de los escenarios, también se ve una marcada superioridad de la Programación Estocástica. Esta ultima comparación es importante porque serán estos los verdaderos costos al utilizarse esta técnica para la planeación, no el costo esperado. La gráfica del costo para cada escenario contra el de Programación Lineal puede ser observada en la Figura 9. Los costos totales también se graficaron y pueden ser vistos en la Figura 10.

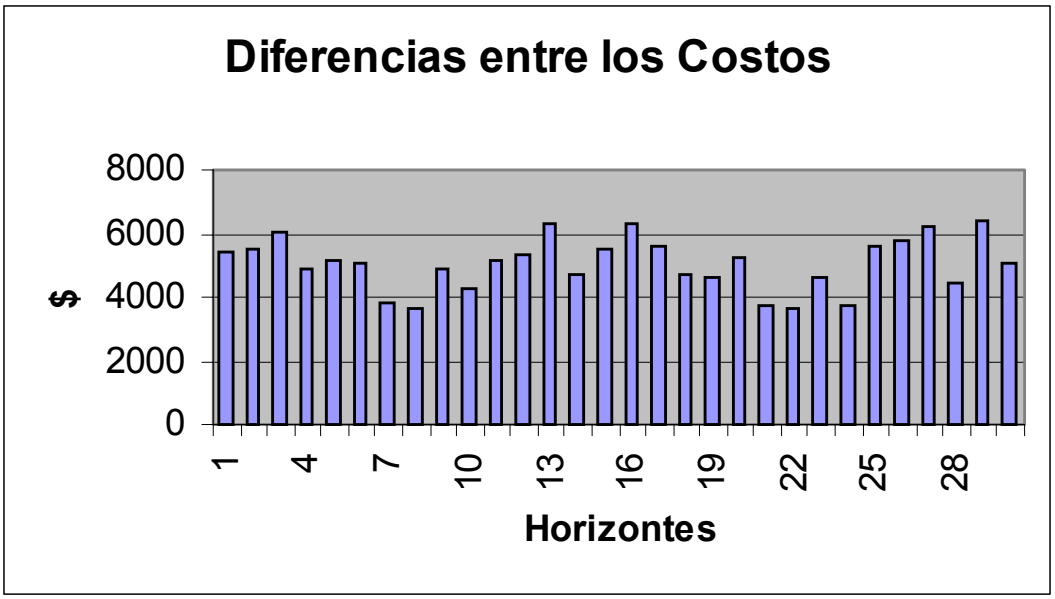

Figura 8. Diferencias entre los Costos del Primer Ejemplo

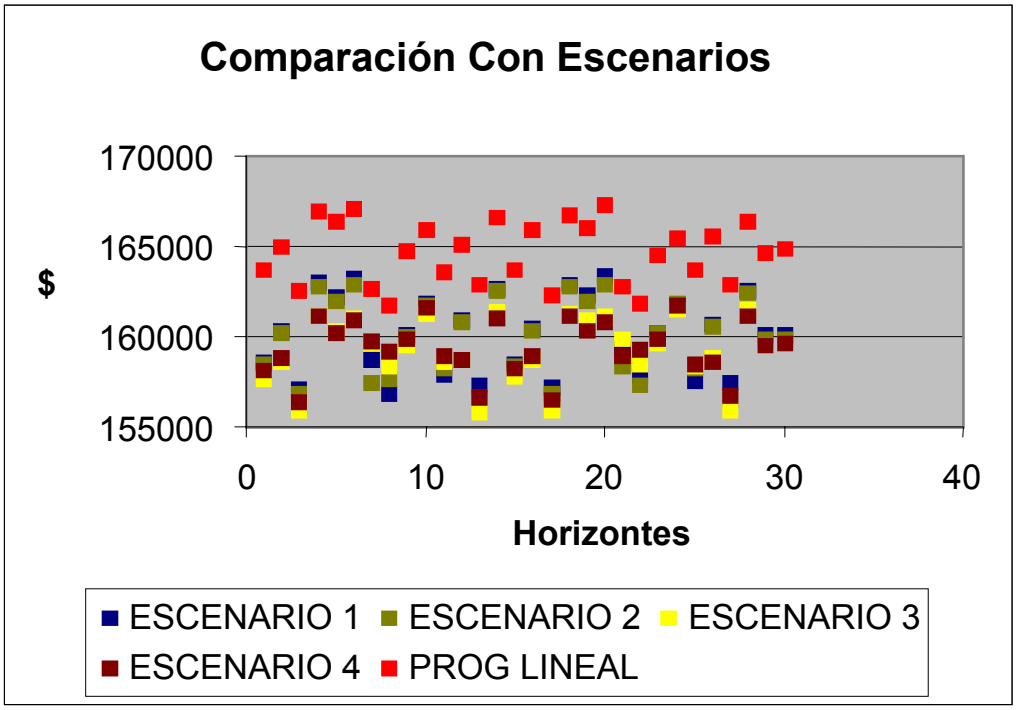

Figura 9. Comparación con Cada Escenario, Primer Ejemplo

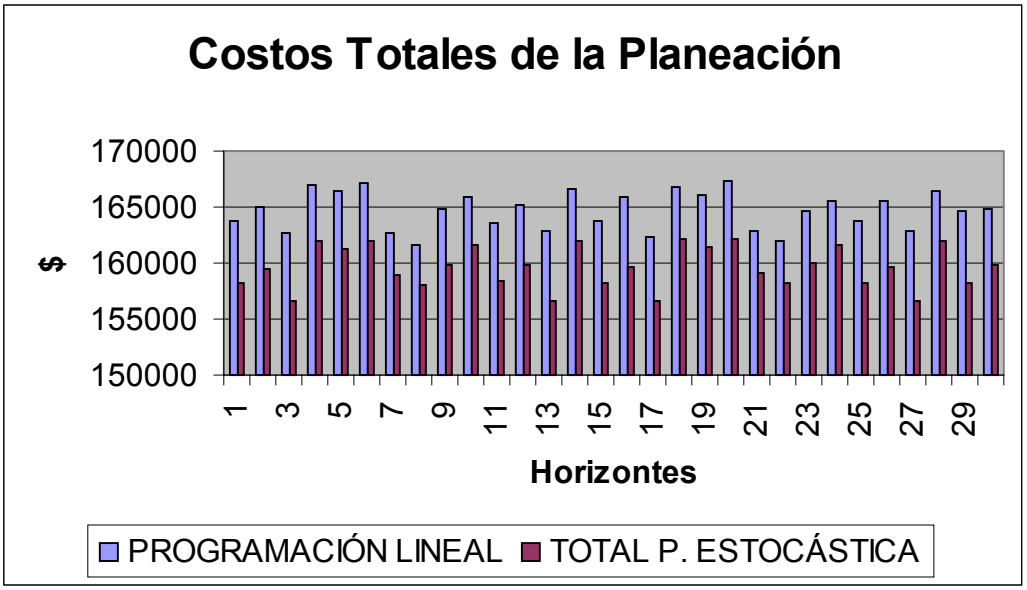

Figura 10. Costos Totales del Primer Ejemplo

Para el segundo ejemplo no aleatorio se procedió a hacer lo mismo que con el primer ejemplo. Se hallaron las diferencias entre los costos y se hallo una media de 1089 unidades monetarias. Las diferencias entre los costos pueden ser vistas en la Figura 11. La gráfica de los costos para cada escenario y los costos de Programación Lineal puede ser vista en la Figura 12 y la de los costos totales para las dos técnicas de Programación puede ser vista en la Figura 13.

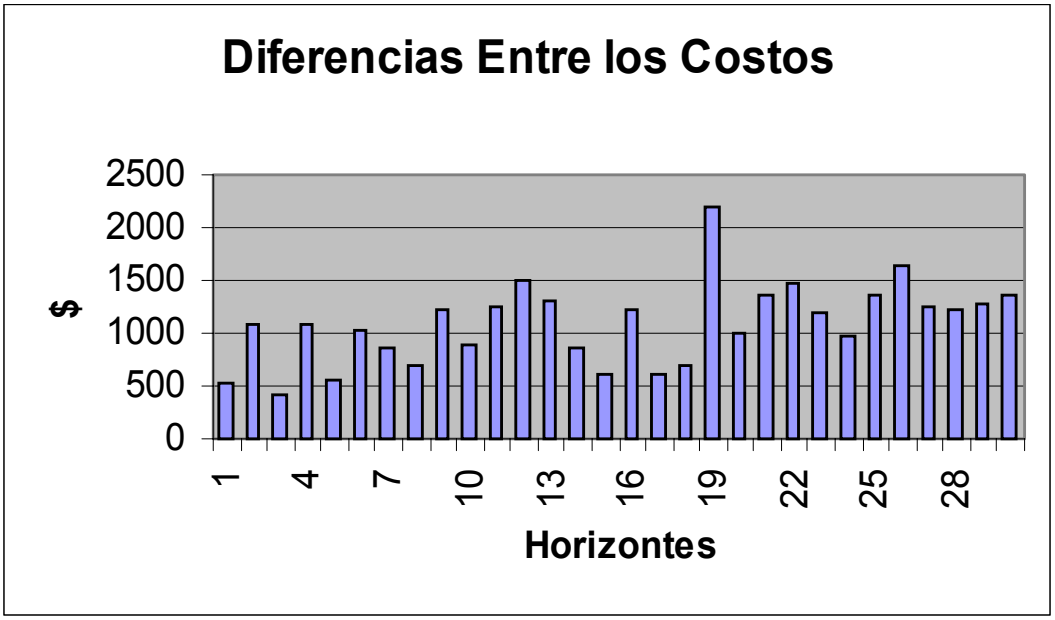

Figura 11. Diferencias entre los Costos del Segundo Ejemplo

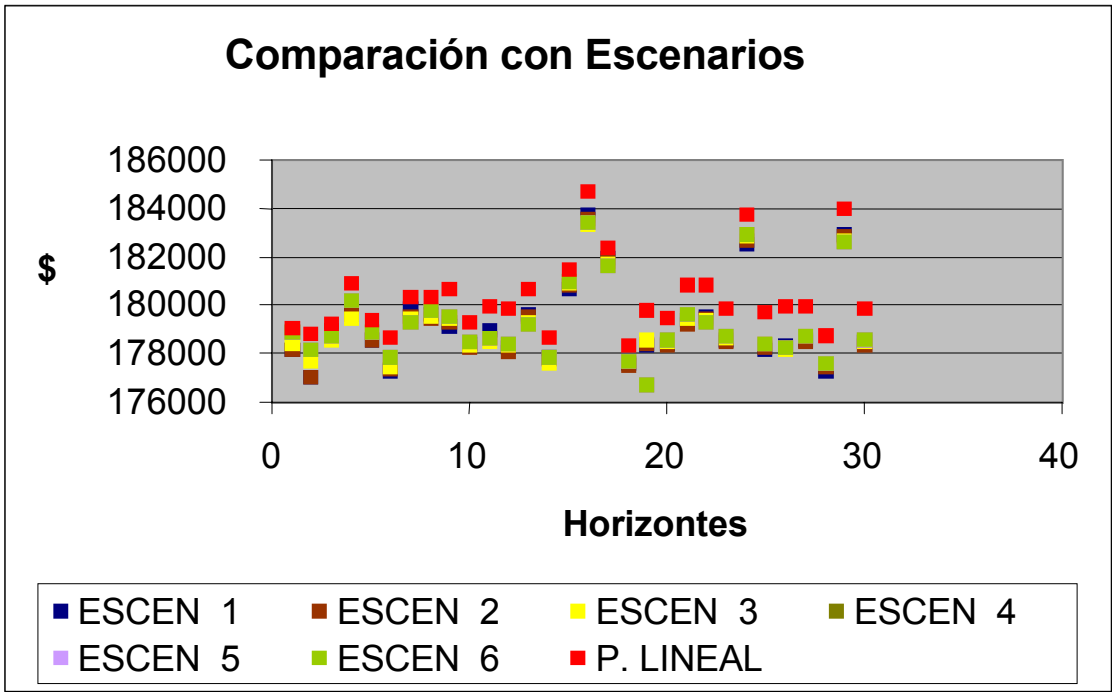

Figura 12. Comparación con Cada Escenario, Segundo Ejemplo

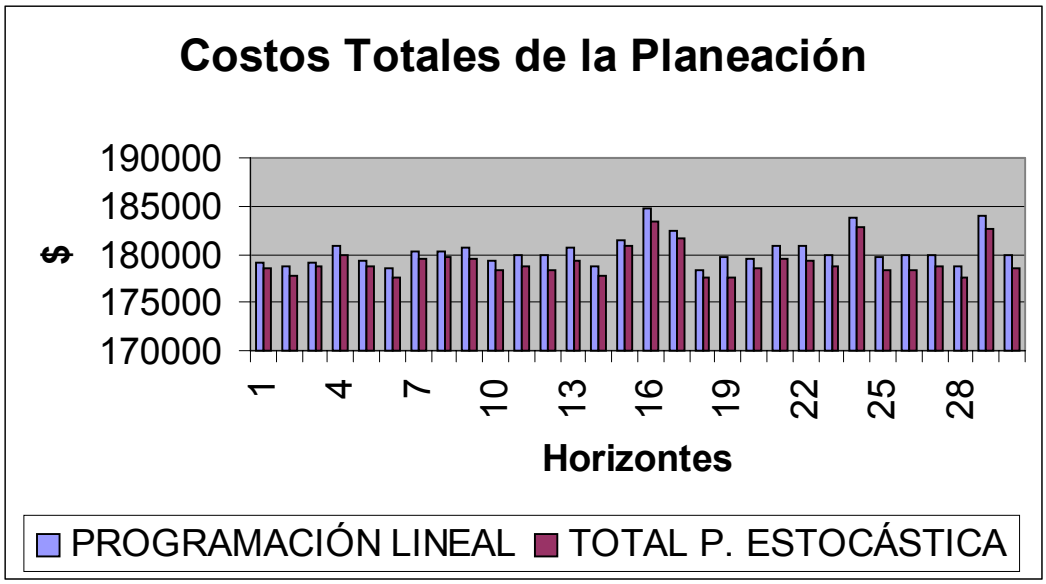

Figura 13. Costos Totales del Segundo Ejemplo

7. CONCLUSIONES

Sistemas de planeación de la producción como el MRP, MRP II o el ERP, están ampliamente difundidos en las empresas actualmente. La mayoría de estas son compañías dedicadas a el ensamblaje y no a la transformación de materias primas en productos, ya que estos sistemas de planeación se adaptan mas fácilmente a las primeras.

Para que modelos matemáticos para la planeación de la producción sean mas fácilmente implementados y acogidos por las empresas, estos deben ser conceptual y prácticamente posibles de integrar a otros sistemas como los anteriormente mencionados.

Uno de los tipos de problemas de Programación Estocástica se trata de los problemas con recurso. Estos a su vez se dividen en tres tipos los cuales son con Recurso Simple, Parcial y Completo, siendo estos últimos los de mas complejidad y por su puesto los mas difíciles de desarrollar a la hora de la codificación y de resolver.

Una de las formas de incorporar la incertidumbre en la modelación es describir esta como un grupo de escenarios. Estos a su vez se implementan eficientemente mediante el uso de árboles de escenarios, en los cuales, se desarrollan los diferentes elementos probabilísticos. Sin embargo, la codificación de estos árboles necesita la separación de los escenarios y la inclusión de nuevas restricciones, lo que agrega una estructura especial con la cual algunos métodos de solución explotan.

Al desarrollar y correr los códigos para algunos ejemplos no aleatorios, los modelos con recurso simple mostraron ser mas fáciles de resolver, debido al menor número de variables y restricciones. La región factible de estos modelos es mas amplia y hallar la solución toma menor tiempo, aunque la diferencia no es significativa. Por el contrario, los modelos con recurso completo son mas complejos y la región factible se disminuye. Para algunos ejemplos aleatorios no se obtuvo solución porque no existía región factible con los parámetros de entrada aleatorios dados o porque sencillamente la estructura era compleja y el software explotaba. Sin embargo, los códigos desarrollados dieron soluciones optimas e implementables para todos los casos corridos con recurso simple, y para algunos casos con recurso completo.

Para los dos ejemplos no aleatorios a los que se les desarrollaron códigos, los modelos con recurso simple dieron soluciones factibles. Por el contrario, los modelos con recurso completo produjeron resultados no factibles, debido a la estructura compleja de este. Los métodos de solución del software explotaron y no se pudieron obtener resultados para compararlos con Programación Lineal.

Para los modelos de Programación Lineal se desarrollaron igualmente códigos para los dos ejemplos no aleatorios. Estos al ser corridos produjeron soluciones factibles. Estas se compararon con las obtenidas para los modelos con recurso simple.

Las gráficas de las Figuras 8 a la 13 muestran los comportamiento de los costos de la producción al ser planeada con Programación Lineal y Programación Estocástica. En estas se puede observar una notoria superioridad de la Programación Estocástica, al ser inferiores los costos en todos y cada uno de los horizontes usados para la comparación en ambos ejemplos. Para el primer ejemplo no aleatorio se ahorra en promedio 5098 unidades monetarias por horizonte de 5 periodos de tiempo al utilizarse esta técnica. Esto equivale a aproximadamente un ahorro del 3% en los costos de producción. A su vez, todos

los escenarios muestran unos costos mas bajos que los obtenidos al usar Programación Lineal y este al final es el costo que cuenta, ya que a medida que pasa el tiempo y los horizontes se van produciendo, solo uno de los escenario se llevara a cabo.

Para el segundo ejemplo no aleatorio, las diferencias en costos no fueron tan grandes pero siguieron siendo significativas. En promedio se puede ahorrar 1089 unidades monetarias por horizonte de 5 periodos de tiempo, que equivale a un ahorro del 1% en los costos de producción. Igualmente, todos los escenarios corridos llevarían a unos costos inferiores que los que se obtendrían al utilizarse Programación Lineal.

Cabe anotar igualmente, que una parte clave de los modelos de Programación Estocástica es el desarrollo de los escenarios mediante árboles de escenarios que se aproximen a la realidad. El elemento probabilístico debe ser el estudio mas importante previo al desarrollo del modelo. La escogencia de este debe ser mediante un análisis con expertos para de esta manera obtener mejores resultados. Cuando se encuentren mas de un elemento probabilístico que debería ser desarrollado, el número de escenarios crece exponencialmente. Para disminuir el número de estos se puede utilizar muestreo al momento de desarrollar los escenarios, combinando los elementos.

Si los escenarios no son parecidos a la realidad, la Programación Estocástica puede traer mayores costos que la Programación Lineal.

8. RECOMENDACIONES

Ya se mostró que para dos ejemplos los resultados obtenidos con Programación Estocástica fueron superiores que los obtenidos con Programación Lineal. De esta manera se recomienda ampliar los modelos existentes de planeación de la producción que utilizan Programación Lineal a modelos de Programación Estocástica. Con esto, el estudio podría quedar mas completo y demostrar que para otros modelos la Programación Estocástica también puede dar mejores resultados.

Algunos escritores han desarrollado modelos de planeación de la producción que han mostrado ser implementables (Bahl et al., 1987 y Saphiro, J., 1993). Sin embargo, estos modelos son de Programación Lineal y pueden ser ampliados para ser resueltos mediante Programación Estocástica. De esta manera un comparación mas completa puede ser llevada a cabo.

Igualmente se recomienda desarrollar un modelo adicional que sería con recurso parcial. En este las decisiones de Inventario y Tiempo Extra serían decisiones recursivas y dependerían del elemento probabilístico. Este modelo puede traer mayores beneficios que el modelo con recurso simple aquí solucionado y su solución no sería tan complicada como el modelo con recurso completo que no pudo ser resuelto.

Otra recomendación para tomar en cuenta es transformar el código desarrollado aquí para el software Xpress-MP de Dash Optimization para que pueda ser utilizado en otro software para la optimización de modelos matemáticos como lo es GAMS. Igualmente, se podría entrar a revisar algoritmos de solución de modelos de Programación Estocástica como el Integer L-Shaped Method, el cual trabaja

con el método de Ramificación y Poda integrado con el L-Shaped Decomposition. Este método sería el idóneo de solución de este modelo debido a la estructura con variables enteras tanto de primera etapa como recursivas. Para la ampliación de este trabajo se podría desarrollar un código que utilice este algoritmo de solución.

9. BIBLIOGRAFÍA

APICS Dictionary. American Production and Inventory Control Society. (1984).

Bahl, H.C., Ritzman, L.P., y Gupta, J.N.D. (1987). Determining lot sizes and resource requirement: A Review. Operations Research, 35(3), 329-345.

Berman, O., Ganz, Z. y Wagner, J.M. (1994). A Stochastic Optimization model for planning capacity expansion in a service industry under ucertain demand. Naval Research Logistics, 41, 545-564.

Birge, J.R. y Louveaux, F. (1997). Introduction to Stochastic Programming. Springer Series in Operations Research.

Camacho, Salvador. (1997). Planeación de Recursos Empresariales.

Chase R.B. y Aquilano, N.J. (1995). Dirección y Administración de la Producción y de las Operaciones. Irwin.

Eppen, G.D., Martin, R.K. y Schrage, L. (1989). A scenario approach to capacity planning. Operations Research, 37(4), 517-527.

Escudero, L.F., Kamesam, P.V., King, A.J. y Wets, R.J. (1993). Production Planning via Scenario modeling. Annals of Operation Research, 43, 311-355.

Frauendorfer, K. y Schurle, M. (1996). The curse of dimensionality in stochastic multistage linear programming. Technical report, Workshop on Stochastic Programming, Tucson, AZ.

Gassmann, H. y Ireland, A. (1995). Scenario Formulation in an Algebraic Modelling Language. Annals of Operations Research, 59, 45-75.

Gassmann, H. y Ireland, A. (1996). On the Automatic Formulation of Stochastic Linear Programs. Annals of Operations Research, 64, 83-112.

Gazmuri, P. y Arrate, I. (1995). Modeling and Visualization for a Production Planning Decision Support System. International Transactions in Operational Research, 2(3), 249-258.

Geoffrion, A. (1987). Introduction to Structured Modeling. Management Science, 33(5), 547-548.

Gfrerer, H. y Zaepfel, G. (1995). Hierarchical Model for Production Planning in the Case of Uncertain Demand. European Journal of Operational Research, 86, 142-161.

Gravel, M., Kiss, L., Martel, J.M. y Price, W. (1994). A DSS for Production Planning. International Transactions in Operational Research, 1(3), 363-373.

Higle, J.L. y Sen, S. (1996). Stochastic Decomposition: A Statistical Method for Large Scale Linear Programming. Kluwer Academic Publishers.

Karabuk, S. y Wu S.D. (1999). Strategic capacity planning in the semiconductor industry: a stochastic programming approach. Report No 99T-12. Department of Industrial and Manufacturing Systems Engineering, Lehigh University.

King, A.J. y Wets, R.J.B. (1989). Epi-consistency of convex stochastic programs, Stochastics and Statistics Reports, 34, 83-92.

Kira, D., Kusy, M. y Rakita, I. (1997). A Stochastic Linear Programming Approach to Hierarchical Production Planning. Journal of the Operational Research Society, 48, 207-211.

Maturana, S. y Eterovic, Y. (1995). Vehicle Routing and Production Planning Decision Support Systems: Designing Graphical User Interfaces. International Transactions in Operational Research, 2(3), 233-247.

Maturana, S., Gazmuri, P. y Villena, C. (1996). Development and Implementation of a Production Planning DSS for a Chilean Manufacturing Firm, Proceedings of the 20th International Conference on Computers & Industrial Engineering, ICC&IE 96, Kyongju, Korea, 757-760.

Puerto, Luis Alfredo. (2000). Impacto de las soluciones de Planeación de Recursos Empresariales (ERP), en empresas de Santa Fé de Bogotá. Universidad Javeriana, Santa Fé de Bogotá (Colombia), Junio de 2000.

Rockafellar, R.T. y Wets, R. (1991). Scenarios and policy aggregation in optimization under uncertainty. Mathematics of Operations Research 16, 119-147.

Shapiro, J. F. (1993). Mathematical Programming Models for Production Planning and Scheduling. Chapter 8 in handbooks in OR & MS, Vol 4, pp. 371-443.

Wu, J. y Sen, S. (2000). A Stochastic Programming Model for Currency Option Hedging. Annals of Operation Research, 100, 227-249.

ANEXO A. Resumen de los Modelos de Programación Estocástica

Modelo General de Planeación de la Producción con Recurso Simple

$$MINIMIZAR \sum_{s=1}^{S} p_s \left[\sum_{i=1}^{N} \sum_{t=1}^{T} \left(h_i * y_{its} + c_i * d_{it} \right) \right] + \sum_{k=1}^{K} \sum_{t=1}^{T} \left(co_{kt} * o_{kt} \right)$$

Sujeto a:

$$y_{i,t-1,s} + f_i * x_{i,t-Li} - y_{it} - \sum_{j=1}^{J} a_{ij} * x_{jt} \geq r_{its}$$

Para todo i = 1,..., N, t = 1,...,T s = 1,...,S

$$\sum_{i=1}^{N} \left(b_{ik} * x_{it} + su_{ik} * d_{it} \right) - o_{kt} \leq CAP_{kt}$$

Para todo t = 1,...,T k = 1,...,K

$$x_{it} - q_{it} * d_{it} \leq 0$$

Para todo i = 1,..., N, t = 1,...,T

$d_{it} = 0$ ó 1 $y_{its} \geq 0$ $x_{it} \geq 0$ $o_{kt} \geq 0$

Para todo i = 1,..., N, t = 1,...,T k = 1,...,K s = 1,...,S

Restricciones de No Anticipo

Modelo General de Planeación de la Producción con Recurso Completo

$$MINIMIZAR \sum_{s=1}^{S} p_s \left[\sum_{i=1}^{N} \sum_{t=1}^{T} (h_i * y_{its} + c_i * d_{it}) + \sum_{k=1}^{K} \sum_{t=1}^{T} (co_{kt} * o_{kts}) \right]$$

Sujeto a:

$$y_{i,t-1,s} + f_i * x_{i,t-Li,s} - y_{its} - \sum_{j=1}^{J} a_{ij} * x_{jts} \geq r_{its}$$

Para todo i = 1,..., N, t = 1,...,T s = 1,...,S

$$\sum_{i=1}^{N} (b_{ik} * x_{its} + su_{ik} * d_{its}) - o_{kts} \leq CAP_{kt}$$

Para todo t = 1,...,T k = 1,...,K s = 1,...,S

$$x_{its} - q_{it} * d_{its} \leq 0$$

Para todo i = 1,..., N, t = 1,...,T s = 1,...,S

d_{its} = 0 ó 1 $y_{its} \geq 0$ $x_{its} \geq 0$ $o_{kts} \geq 0$

Para todo i = 1,..., N, t = 1,...,T k = 1,...,K s = 1,...,S

Restricciones de No Anticipo

ANEXO B. Códigos para resolver los Ejemplos Aleatorios con Programación Estocástica

Código del Primer Ejemplo Numérico Aleatorio con Recurso Simple

```
model ProAle1
  uses "mmxprs"

  declarations
    ! indicies
    ITEMS       = 1..4
    ITEMS2      = 1..4
    PERIODS     = 1..4
    FACILITIES  = 1..3
    SCENARIOS   = 1..3
    ! parameters
    c     :  array(ITEMS) of real
    su    :  array(ITEMS, FACILITIES) of real
    b     :  array(ITEMS, FACILITIES) of real
    h     :  array(ITEMS) of real
    r     :  array(ITEMS, PERIODS, SCENARIOS) of real
    CAP   :  array(FACILITIES, PERIODS) of real
    q     :  array(ITEMS, PERIODS) of real
    p     :  array(SCENARIOS) of real
    co    :  array(FACILITIES, PERIODS) of real
    a     :  array(ITEMS, ITEMS2) of real

    inv0  :  array(ITEMS) of real
    L     :  array(ITEMS) of real
    f     :  array(ITEMS) of real


    ! decision variables
    x     :  array(ITEMS, PERIODS) of mpvar
    v     :  array(ITEMS2, PERIODS) of mpvar
    d     :  array(ITEMS, PERIODS) of mpvar
    o     :  array(FACILITIES, PERIODS) of mpvar
    y     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar
```

```
   dem11,dem12,dem13,dem14 : real

end-declarations

! declare d(i,t) binary
forall(i in ITEMS, t in PERIODS)  d(i,t) is_binary
forall(i in ITEMS, t in PERIODS)  x(i,t) is_integer
forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  y(i,t,s) is_integer


! generate random data from Unifrom distributions
dem11 := 100 + (100*random)
dem12 := 100 + (100*random)
dem13 := 100 + (100*random)
dem14 := 100 + (100*random)
forall(s in SCENARIOS) r(1,1,s):= dem11
forall(s in SCENARIOS) r(2,1,s):= dem12
forall(s in SCENARIOS) r(3,1,s):= dem13
forall(s in SCENARIOS) r(4,1,s):= dem14
forall(i in ITEMS, t in 2..4, s in SCENARIOS)  r(i,t,s):= 150 + (50*random)
forall(i in ITEMS)  c(i):= 500 + (50*random)
forall(k in FACILITIES, t in PERIODS)  co(k,t):= 30 + (10*random)
forall(i in ITEMS, k in FACILITIES)  su(i,k):= 3 + (1*random)
forall(i in ITEMS, k in FACILITIES)  b(i,k):= 0.15 + (0.05*random)
forall(i in ITEMS)  h(i):= 2 + (1*random)
forall(i in ITEMS)  inv0(i):= 600 + (30*random)
forall(k in FACILITIES,t in PERIODS)  CAP(k,t):= 30 + (7*random)
forall(i in ITEMS, t in PERIODS)  q(i,t):= 500 + (20*random)
forall(i in ITEMS)  f(i):= 0.8 + (0.2*random)
forall(i in 1..3) L(i):= 1
L(4) := 2
a(1,3) := 3
a(1,4) := 2
a(2,3) := 1
a(2,4) := 2
a(3,4) := 1

forall(s in SCENARIOS) p(s):=1/3


! Objective function
Total_Expected_Cost:=
   sum(s in SCENARIOS) ( p(s) * (sum(i in ITEMS) sum(t in PERIODS)( c(i)* d(i,t)
+ h(i) * y(i,t,s) ))) + sum(k in FACILITIES) sum(t in PERIODS) (co(k,t)* o(k,t))

! demand constraint for every item i, period t under every scenario s
forall(i in 1..3,t in PERIODS, s in SCENARIOS)
```

```
    demand_constraints1(i,t,s) :=   f(i)* if((t-1)<1,0,x(i,t-1)) + if (t=1,inv0(i),y(i,t-1,s))
- y(i,t,s) - sum(j in ITEMS2) a(i,j)* v(j,t) - r(i,t,s)  >= 0
  forall(t in PERIODS, s in SCENARIOS)
    demand_constraints2(4,t,s) :=   f(4)* if((t-2)<1,0,x(4,t-2)) + if (t=1,inv0(4),y(4,t-
1,s)) - y(4,t,s) - sum(j in ITEMS2) a(4,j)* v(j,t) - r(4,t,s)  >= 0

  ! capacity constraint for every period and facility
  forall(k in FACILITIES, t in PERIODS)
    capacity_constraints(k,t) := sum(i in ITEMS) (su(i,k)*d(i,t) + b(i,k)*x(i,t)) - o(k,t)
<= CAP(k,t)

  ! production bound for every item i and period t
  forall(i in ITEMS, t in PERIODS)
    production_bound(i,t):= x(i,t) - q(i,t) * d(i,t) <= 0

  ! Equality constraint for new variables
  forall(t in PERIODS)do
    x(1,t) = v(1,t)
    x(2,t) = v(2,t)
    x(3,t) = v(3,t)
    x(4,t) = v(4,t)
  end-do

  forall(i in ITEMS) do
    y(i,1,1) = y(i,1,2)
    y(i,1,1) = y(i,1,3)
  end-do


  ! model description complete

  setparam("XPRS_verbose",false)   ! Enable message printing in mmxosl

  ! solve the model
  minimize(Total_Expected_Cost)

  writeln(" INPUT ")
  writeln
  forall(s in SCENARIOS) do
    writeln("Scenario = ", s)
    forall(t in PERIODS) do
    writeln("Period = ", t)
        forall(i in ITEMS) writeln("Demand Item ", i, "=", r(i,t,s))
        end-do
        writeln
     end-do
```

```
  forall(i in ITEMS)writeln("Set-up Cost Item ", i, " = ", c(i))
  forall(i in ITEMS)writeln("Unit Holding Cost Item ", i, " = ", h(i))
  forall(i in ITEMS)writeln("Inventory 0 Item ", i, " = ", inv0(i))
  forall(i in ITEMS)writeln("Lead Time Item ", i, " = ", L(i))
  forall(i in ITEMS)writeln("Yield Item ", i, " = ", f(i))
  writeln

  forall(k in FACILITIES) do
    writeln("Facility = ", k)
    forall(i in ITEMS)writeln("Set-up Resource Utilization Item ", i, " = ", su(i,k))
    forall(i in ITEMS)writeln("Capacity Utilization Rate Item ", i, " = ", b(i,k))
    writeln
    end-do

  forall(t in PERIODS) do
    writeln("Period = ", t)
    forall(k in FACILITIES)writeln("Capacity at Facility ", k, " = ", CAP(k,t))
    forall(i in ITEMS)writeln("Upper Bound Item ", i, " = ", q(i,t))
    writeln
  end-do
  writeln
  writeln(" SOLUTION ")
  writeln("Total Expected Costs are:  ", getobjval)
  writeln
  forall(t in PERIODS) do
  writeln("Period = ", t)
  forall(i in ITEMS)writeln("Produced Item ", i, " = ", getsol(x(i,t)) )
  forall(k in FACILITIES)writeln("Overtime Capacity Facility ", k, " = ", getsol(o(k,t)))
  writeln
  end-do
  writeln
  forall(s in SCENARIOS) do
    writeln("Scenario = ", s)
    forall(t in PERIODS) do
        forall(i in ITEMS) write(  strfmt(getsol(y(i,t,s)),7,1) , "  ")
        writeln
    end-do
    writeln
  end-do

end-model
```

Código del Segundo Ejemplo Numérico Aleatorio con Recurso Simple

```
model ProAle2
  uses "mmxprs"

  declarations
    ! indicies
    ITEMS       = 1..3
    ITEMS2      = 1..3
    PERIODS     = 1..5
    FACILITIES  = 1..2
    SCENARIOS   = 1..4
    ! parameters
    c     :  array(ITEMS) of real
    su    :  array(ITEMS, FACILITIES) of real
    b     :  array(ITEMS, FACILITIES) of real
    h     :  array(ITEMS) of real
    r     :  array(ITEMS, PERIODS, SCENARIOS) of real
    CAP   :  array(FACILITIES, PERIODS) of real
    q     :  array(ITEMS, PERIODS) of real
    p     :  array(SCENARIOS) of real
    co    :  array(FACILITIES, PERIODS) of real
    a     :  array(ITEMS, ITEMS2) of real

    inv0  :  array(ITEMS) of real
    L     :  array(ITEMS) of real
    f     :  array(ITEMS) of real


    ! decision variables
    x     :  array(ITEMS, PERIODS) of mpvar
    v     :  array(ITEMS2, PERIODS) of mpvar
    d     :  array(ITEMS, PERIODS) of mpvar
    o     :  array(FACILITIES, PERIODS) of mpvar
    y     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar


  end-declarations

  ! declare d(i,t) binary
  forall(i in ITEMS, t in PERIODS)  d(i,t) is_binary
  forall(i in ITEMS, t in PERIODS)  x(i,t) is_integer
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  y(i,t,s) is_integer


  ! generate random data from Unifrom distributions
```

```
  dem11 := 100 + (100*random)
  dem12 := 100 + (100*random)
  dem13 := 100 + (100*random)
  dem211:= 100 + (100*random)
  dem212:= 100 + (100*random)
  dem213:= 100 + (100*random)
  dem221:= 100 + (100*random)
  dem222:= 100 + (100*random)
  dem223:= 100 + (100*random)
  forall(s in SCENARIOS) r(1,1,s):= dem11
  forall(s in SCENARIOS) r(2,1,s):= dem12
  forall(s in SCENARIOS) r(3,1,s):= dem13
  forall(s in 1..2)  r(1,2,s):= dem211
  forall(s in 3..4)  r(1,2,s):= dem221
  forall(s in 1..2)  r(2,2,s):= dem212
  forall(s in 3..4)  r(2,2,s):= dem222
  forall(s in 1..2)  r(3,2,s):= dem213
  forall(s in 3..4)  r(3,2,s):= dem223
  forall(i in ITEMS, t in 3..5, s in SCENARIOS)  r(i,t,s):= 150 + (50*random)
  forall(i in ITEMS)  c(i):= 500 + (50*random)
  forall(k in FACILITIES, t in PERIODS)  co(k,t):= 30 + (10*random)
  forall(i in ITEMS, k in FACILITIES)  su(i,k):= 3 + (1*random)
  forall(i in ITEMS, k in FACILITIES)  b(i,k):= 0.15 + (0.05*random)
  forall(i in ITEMS)  h(i):= 2 + (1*random)
  forall(i in ITEMS)  inv0(i):= 600 + (30*random)
  forall(k in FACILITIES,t in PERIODS)  CAP(k,t):= 30 + (7*random)
  forall(i in ITEMS, t in PERIODS)  q(i,t):= 500 + (20*random)
  forall(i in ITEMS)  f(i):= 0.8 + (0.2*random)
  forall(i in ITEMS) L(i):= 1
  a(1,3) := 2
  a(2,3) := 1


  forall(s in SCENARIOS) p(s):=1/4


  ! Objective function
  Total_Expected_Cost:=
     sum(s in SCENARIOS) ( p(s) * (sum(i in ITEMS) sum(t in PERIODS)( c(i)* d(i,t)
+ h(i) * y(i,t,s) ))) + sum(k in FACILITIES) sum(t in PERIODS) (co(k,t)* o(k,t))

  ! demand constraint for every item i, period t under every scenario s
  forall(i in ITEMS,t in PERIODS, s in SCENARIOS)
     demand_constraints1(i,t,s) :=   f(i)* if((t-1)<1,0,x(i,t-1)) + if (t=1,inv0(i),y(i,t-1,s))
- y(i,t,s) - sum(j in ITEMS2) a(i,j)* v(j,t) - r(i,t,s)  >= 0

  ! capacity constraint for every period and facility
```

```
  forall(k in FACILITIES, t in PERIODS)
    capacity_constraints(k,t) := sum(i in ITEMS) (su(i,k)*d(i,t) + b(i,k)*x(i,t)) - o(k,t)
<= CAP(k,t)

  ! production bound for every item i and period t
  forall(i in ITEMS, t in PERIODS)
    production_bound(i,t):= x(i,t) - q(i,t) * d(i,t) <= 0

  ! Equality constraint for new variables
  forall(t in PERIODS)do
    x(1,t) = v(1,t)
    x(2,t) = v(2,t)
    x(3,t) = v(3,t)
  end-do

  forall(i in ITEMS) do
    y(i,1,1) = y(i,1,2)
    y(i,1,2) = y(i,1,3)
    y(i,1,3) = y(i,1,4)
  end-do

  forall(i in ITEMS) do
    y(i,2,1) = y(i,2,2)
    y(i,2,3) = y(i,2,4)
  end-do


  ! model description complete

  setparam("XPRS_verbose",false)   ! Enable message printing in mmxosl

  ! solve the model
  minimize(Total_Expected_Cost)

  writeln(" INPUT ")
  writeln
  forall(s in SCENARIOS) do
    writeln("Scenario = ", s)
    forall(t in PERIODS) do
    writeln("Period = ", t)
        forall(i in ITEMS) writeln("Demand Item ", i, "=", r(i,t,s))
        end-do
        writeln
     end-do

  forall(i in ITEMS)writeln("Set-up Cost Item ", i, " = ", c(i))
  forall(i in ITEMS)writeln("Unit Holding Cost Item ", i, " = ", h(i))
```

```
  forall(i in ITEMS)writeln("Inventory 0 Item ", i, " = ", inv0(i))
  forall(i in ITEMS)writeln("Lead Time Item ", i, " = ", L(i))
  forall(i in ITEMS)writeln("Yield Item ", i, " = ", f(i))
  writeln

  forall(k in FACILITIES) do
    writeln("Facility = ", k)
    forall(i in ITEMS)writeln("Set-up Resource Utilization Item ", i, " = ", su(i,k))
    forall(i in ITEMS)writeln("Capacity Utilization Rate Item ", i, " = ", b(i,k))
    writeln
    end-do

  forall(t in PERIODS) do
    writeln("Period = ", t)
    forall(k in FACILITIES)writeln("Capacity at Facility ", k, " = ", CAP(k,t))
    forall(i in ITEMS)writeln("Upper Bound Item ", i, " = ", q(i,t))
    writeln
  end-do
  writeln
  writeln(" SOLUTION ")
  writeln("Total Expected Costs are:  ", getobjval)
  writeln
  forall(t in PERIODS) do
  writeln("Period = ", t)
  forall(i in ITEMS)writeln("Produced Item ", i, " = ", getsol(x(i,t)) )
  forall(k in FACILITIES)writeln("Overtime Capacity Facility ", k, " = ", getsol(o(k,t)))
  writeln
  end-do
  writeln
  forall(s in SCENARIOS) do
    writeln("Scenario = ", s)
    forall(t in PERIODS) do
        forall(i in ITEMS) write(  strfmt(getsol(y(i,t,s)),7,1) , "  ")
        writeln
    end-do
    writeln
  end-do

end-model
```

Código del Tercer Ejemplo Numérico Aleatorio con Recurso Simple

```
model ProAle3
  uses "mmxprs"

  declarations
    ! indicies
    ITEMS       = 1..3
    ITEMS2      = 1..3
    PERIODS     = 1..5
    FACILITIES  = 1..2
    SCENARIOS   = 1..6
    ! parameters
    c     :  array(ITEMS) of real
    su    :  array(ITEMS, FACILITIES) of real
    b     :  array(ITEMS, FACILITIES) of real
    h     :  array(ITEMS) of real
    r     :  array(ITEMS, PERIODS, SCENARIOS) of real
    CAP   :  array(FACILITIES, PERIODS) of real
    q     :  array(ITEMS, PERIODS) of real
    p     :  array(SCENARIOS) of real
    co    :  array(FACILITIES, PERIODS) of real
    a     :  array(ITEMS, ITEMS2) of real

    inv0  :  array(ITEMS) of real
    L     :  array(ITEMS) of real
    f     :  array(ITEMS) of real


    ! decision variables
    x     :  array(ITEMS, PERIODS) of mpvar
    v     :  array(ITEMS2, PERIODS) of mpvar
    d     :  array(ITEMS, PERIODS) of mpvar
    o     :  array(FACILITIES, PERIODS) of mpvar
    y     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar


  end-declarations

  ! declare d(i,t) binary
  forall(i in ITEMS, t in PERIODS)  d(i,t) is_binary
  forall(i in ITEMS, t in PERIODS)  x(i,t) is_integer
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  y(i,t,s) is_integer


  ! generate random data from Unifrom distributions
```

```
dem11 := 100 + (100*random)
dem12 := 100 + (100*random)
dem13 := 100 + (100*random)
dem211:= 100 + (100*random)
dem212:= 100 + (100*random)
dem213:= 100 + (100*random)
dem221:= 100 + (100*random)
dem222:= 100 + (100*random)
dem223:= 100 + (100*random)
dem311:= 100 + (100*random)
dem312:= 100 + (100*random)
dem313:= 100 + (100*random)
dem321:= 100 + (100*random)
dem322:= 100 + (100*random)
dem323:= 100 + (100*random)
forall(s in SCENARIOS) r(1,1,s):= dem11
forall(s in SCENARIOS) r(2,1,s):= dem12
forall(s in SCENARIOS) r(3,1,s):= dem13
forall(s in 1..3)  r(1,2,s):= dem211
forall(s in 4..6)  r(1,2,s):= dem221
forall(s in 1..3)  r(2,2,s):= dem212
forall(s in 4..6)  r(2,2,s):= dem222
forall(s in 1..3)  r(3,2,s):= dem213
forall(s in 4..6)  r(3,2,s):= dem223
forall(s in 1..2)  r(1,3,s):= dem311
forall(s in 4..5)  r(1,3,s):= dem321
forall(s in 1..2)  r(2,3,s):= dem312
forall(s in 4..5)  r(2,3,s):= dem322
forall(s in 1..2)  r(3,3,s):= dem313
forall(s in 4..5)  r(3,3,s):= dem323
forall(i in ITEMS) r(i,3,3):= 100 + (100*random)
forall(i in ITEMS) r(i,3,6):= 100 + (100*random)
forall(i in ITEMS, t in 4..5, s in SCENARIOS)  r(i,t,s):= 150 + (50*random)
forall(i in ITEMS)  c(i):= 500 + (50*random)
forall(k in FACILITIES, t in PERIODS)  co(k,t):= 30 + (10*random)
forall(i in ITEMS, k in FACILITIES)  su(i,k):= 3 + (1*random)
forall(i in ITEMS, k in FACILITIES)  b(i,k):= 0.15 + (0.05*random)
forall(i in ITEMS)  h(i):= 2 + (1*random)
forall(i in ITEMS)  inv0(i):= 600 + (30*random)
forall(k in FACILITIES,t in PERIODS)  CAP(k,t):= 30 + (7*random)
forall(i in ITEMS, t in PERIODS)  q(i,t):= 500 + (20*random)
forall(i in ITEMS)  f(i):= 0.8 + (0.2*random)
forall(i in ITEMS) L(i):= 1
a(1,3) := 2
a(2,3) := 1
```

```
  forall(s in SCENARIOS) p(s):=1/6


  ! Objective function
  Total_Expected_Cost:=
    sum(s in SCENARIOS) ( p(s) * (sum(i in ITEMS) sum(t in PERIODS)( c(i)* d(i,t)
+ h(i) * y(i,t,s) ))) + sum(k in FACILITIES) sum(t in PERIODS) (co(k,t)* o(k,t))

  ! demand constraint for every item i, period t under every scenario s
  forall(i in ITEMS,t in PERIODS, s in SCENARIOS)
    demand_constraints1(i,t,s) :=   f(i)* if((t-1)<1,0,x(i,t-1)) + if (t=1,inv0(i),y(i,t-1,s))
- y(i,t,s) - sum(j in ITEMS2) a(i,j)* v(j,t) - r(i,t,s)  >= 0

  ! capacity constraint for every period and facility
  forall(k in FACILITIES, t in PERIODS)
    capacity_constraints(k,t) := sum(i in ITEMS) (su(i,k)*d(i,t) + b(i,k)*x(i,t)) - o(k,t)
<= CAP(k,t)

  ! production bound for every item i and period t
  forall(i in ITEMS, t in PERIODS)
    production_bound(i,t):= x(i,t) - q(i,t) * d(i,t) <= 0

  ! Equality constraint for new variables
  forall(t in PERIODS)do
    x(1,t) = v(1,t)
    x(2,t) = v(2,t)
    x(3,t) = v(3,t)
  end-do

  forall(i in ITEMS) do
    y(i,1,1) = y(i,1,2)
    y(i,1,2) = y(i,1,3)
    y(i,1,3) = y(i,1,4)
    y(i,1,4) = y(i,1,5)
    y(i,1,5) = y(i,1,6)
  end-do

  forall(i in ITEMS) do
    y(i,2,1) = y(i,2,2)
    y(i,2,2) = y(i,2,3)
    y(i,2,4) = y(i,2,5)
    y(i,2,5) = y(i,2,6)
  end-do

  forall(i in ITEMS) do
    y(i,3,1) = y(i,3,2)
    y(i,3,4) = y(i,3,5)
```

```
end-do



! model description complete

setparam("XPRS_verbose",false)   ! Enable message printing in mmxosl

! solve the model
minimize(Total_Expected_Cost)

writeln(" INPUT ")
writeln
forall(s in SCENARIOS) do
  writeln("Scenario = ", s)
  forall(t in PERIODS) do
  writeln("Period = ", t)
      forall(i in ITEMS) writeln("Demand Item ", i, "=", r(i,t,s))
      end-do
      writeln
   end-do

forall(i in ITEMS)writeln("Set-up Cost Item ", i, " = ", c(i))
forall(i in ITEMS)writeln("Unit Holding Cost Item ", i, " = ", h(i))
forall(i in ITEMS)writeln("Inventory 0 Item ", i, " = ", inv0(i))
forall(i in ITEMS)writeln("Lead Time Item ", i, " = ", L(i))
forall(i in ITEMS)writeln("Yield Item ", i, " = ", f(i))
writeln

forall(k in FACILITIES) do
  writeln("Facility = ", k)
  forall(i in ITEMS)writeln("Set-up Resource Utilization Item ", i, " = ", su(i,k))
  forall(i in ITEMS)writeln("Capacity Utilization Rate Item ", i, " = ", b(i,k))
  writeln
  end-do

forall(t in PERIODS) do
  writeln("Period = ", t)
  forall(k in FACILITIES)writeln("Capacity at Facility ", k, " = ", CAP(k,t))
  forall(i in ITEMS)writeln("Upper Bound Item ", i, " = ", q(i,t))
  writeln
end-do
writeln
writeln(" SOLUTION ")
writeln("Total Expected Costs are:  ", getobjval)
writeln
forall(t in PERIODS) do
```

```
  writeln("Period = ", t)
  forall(i in ITEMS)writeln("Produced Item ", i, " = ", getsol(x(i,t)) )
  forall(k in FACILITIES)writeln("Overtime Capacity Facility ", k, " = ", getsol(o(k,t)))
  writeln
  end-do
  writeln
  forall(s in SCENARIOS) do
    writeln("Scenario = ", s)
    forall(t in PERIODS) do
        forall(i in ITEMS) write(  strfmt(getsol(y(i,t,s)),7,1) , "  ")
        writeln
    end-do
    writeln
  end-do

end-model
```

Código del Primer Ejemplo Numérico Aleatorio con Recurso Completo

```
model ProAle4
  uses "mmxprs"

  declarations
    ! indicies
    ITEMS       = 1..4
    ITEMS2      = 1..4
    PERIODS     = 1..4
    FACILITIES  = 1..3
    SCENARIOS   = 1..3
    ! parameters
    c     :  array(ITEMS) of real
    su    :  array(ITEMS, FACILITIES) of real
    b     :  array(ITEMS, FACILITIES) of real
    h     :  array(ITEMS) of real
    r     :  array(ITEMS, PERIODS, SCENARIOS) of real
    CAP   :  array(FACILITIES, PERIODS) of real
    q     :  array(ITEMS, PERIODS) of real
    p     :  array(SCENARIOS) of real
    co    :  array(FACILITIES, PERIODS) of real
    a     :  array(ITEMS, ITEMS2) of real

    inv0  :  array(ITEMS) of real
    L     :  array(ITEMS) of real
    f     :  array(ITEMS) of real


    ! decision variables
    x     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar
    v     :  array(ITEMS2, PERIODS, SCENARIOS) of mpvar
    d     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar
    o     :  array(FACILITIES, PERIODS, SCENARIOS) of mpvar
    y     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar


  end-declarations

  ! declare d(i,t) binary
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  d(i,t,s) is_binary
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  x(i,t,s) is_integer
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  y(i,t,s) is_integer


  ! generate random data from Unifrom distributions
```

```
  dem11 := 100 + (100*random)
  dem12 := 100 + (100*random)
  dem13 := 100 + (100*random)
  dem14 := 100 + (100*random)
  forall(s in SCENARIOS) r(1,1,s):= dem11
  forall(s in SCENARIOS) r(2,1,s):= dem12
  forall(s in SCENARIOS) r(3,1,s):= dem13
  forall(s in SCENARIOS) r(4,1,s):= dem14
  forall(i in ITEMS, t in 2..4, s in SCENARIOS)  r(i,t,s):= 150 + (50*random)
  forall(i in ITEMS)  c(i):= 500 + (50*random)
  forall(k in FACILITIES, t in PERIODS)  co(k,t):= 30 + (10*random)
  forall(i in ITEMS, k in FACILITIES)  su(i,k):= 3 + (1*random)
  forall(i in ITEMS, k in FACILITIES)  b(i,k):= 0.15 + (0.05*random)
  forall(i in ITEMS)  h(i):= 2 + (1*random)
  forall(i in ITEMS)  inv0(i):= 600 + (30*random)
  forall(k in FACILITIES,t in PERIODS)  CAP(k,t):= 30 + (7*random)
  forall(i in ITEMS, t in PERIODS)  q(i,t):= 500 + (20*random)
  forall(i in ITEMS)  f(i):= 0.8 + (0.2*random)
  forall(i in 1..3) L(i):= 1
  L(4) := 2
  a(1,3) := 3
  a(1,4) := 2
  a(2,3) := 1
  a(2,4) := 2
  a(3,4) := 1

  forall(s in SCENARIOS) p(s):=1/3


  ! Objective function
  Total_Expected_Cost:=
    sum(s in SCENARIOS) ( p(s) * (sum(i in ITEMS) sum(t in PERIODS)( c(i)*
d(i,t,s) + h(i) * y(i,t,s) ) + sum(k in FACILITIES) sum(t in PERIODS) (co(k,t)*
o(k,t,s))))

  ! demand constraint for every item i, period t under every scenario s
  forall(i in 1..3,t in PERIODS, s in SCENARIOS)
    demand_constraints1(i,t,s) :=   f(i)* if((t-1)<1,0,x(i,t-1,s)) + if (t=1,inv0(i),y(i,t-
1,s)) - y(i,t,s) - sum(j in ITEMS2) a(i,j)* v(j,t,s) - r(i,t,s)  >= 0
  forall(t in PERIODS, s in SCENARIOS)
    demand_constraints2(4,t,s) :=   f(4)* if((t-2)<1,0,x(4,t-2,s)) + if (t=1,inv0(4),y(4,t-
1,s)) - y(4,t,s) - sum(j in ITEMS2) a(4,j)* v(j,t,s) - r(4,t,s)  >= 0

  ! capacity constraint for every period and facility
  forall(k in FACILITIES, t in PERIODS, s in SCENARIOS)
    capacity_constraints(k,t) := sum(i in ITEMS) (su(i,k)*d(i,t,s) + b(i,k)*x(i,t,s)) -
o(k,t,s) <= CAP(k,t)
```

```
! production bound for every item i and period t
forall(i in ITEMS, t in PERIODS, s in SCENARIOS)
   production_bound(i,t):= x(i,t,s) - q(i,t) * d(i,t,s) <= 0

! Equality constraint for new variables
forall(t in PERIODS, s in SCENARIOS)do
   x(1,t,s) = v(1,t,s)
   x(2,t,s) = v(2,t,s)
   x(3,t,s) = v(3,t,s)
   x(4,t,s) = v(4,t,s)
end-do

forall(i in ITEMS) do
   y(i,1,1) = y(i,1,2)
   y(i,1,1) = y(i,1,3)
end-do

forall(i in ITEMS) do
   x(i,1,1) = x(i,1,2)
   x(i,1,1) = x(i,1,3)
end-do

forall(k in FACILITIES) do
   o(k,1,1) = o(k,1,2)
   o(k,1,1) = o(k,1,3)
end-do

forall(i in ITEMS) do
   d(i,1,1) = d(i,1,2)
   d(i,1,1) = d(i,1,3)
end-do


! model description complete

setparam("XPRS_verbose",false)   ! Enable message printing in mmxosl

! solve the model
minimize(Total_Expected_Cost)


writeln(" SOLUTION ")
writeln("Total Expected Costs are:  ", getobjval)
writeln

writeln("Inventory")
```

```
  forall(s in SCENARIOS) do
    writeln("Scenario = ", s)
    forall(t in PERIODS) do
        forall(i in ITEMS) write(  strfmt(getsol(y(i,t,s)),7,1) , "  ")
        writeln
    end-do
    writeln
  end-do

  writeln("Production")
  forall(s in SCENARIOS) do
    writeln("Scenario = ", s)
    forall(t in PERIODS) do
        forall(i in ITEMS) write(  strfmt(getsol(x(i,t,s)),7,1) , "  ")
        writeln
    end-do
    writeln
  end-do

  writeln("Overtime")
  forall(s in SCENARIOS) do
    writeln("Scenario = ", s)
    forall(t in PERIODS) do
        forall(k in FACILITIES) write(  strfmt(getsol(o(k,t,s)),7,1) , "  ")
        writeln
    end-do
    writeln
  end-do

end-model
```

Código del Segundo Ejemplo Numérico Aleatorio con Recurso Completo

```
model ProAle5
  uses "mmxprs"

  declarations
    ! indicies
    ITEMS       = 1..3
    ITEMS2      = 1..3
    PERIODS     = 1..5
    FACILITIES  = 1..2
    SCENARIOS   = 1..4
    ! parameters
    c     :  array(ITEMS) of real
    su    :  array(ITEMS, FACILITIES) of real
    b     :  array(ITEMS, FACILITIES) of real
    h     :  array(ITEMS) of real
    r     :  array(ITEMS, PERIODS, SCENARIOS) of real
    CAP   :  array(FACILITIES, PERIODS) of real
    q     :  array(ITEMS, PERIODS) of real
    p     :  array(SCENARIOS) of real
    co    :  array(FACILITIES, PERIODS) of real
    a     :  array(ITEMS, ITEMS2) of real

    inv0  :  array(ITEMS) of real
    L     :  array(ITEMS) of real
    f     :  array(ITEMS) of real


    ! decision variables
    x     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar
    v     :  array(ITEMS2, PERIODS, SCENARIOS) of mpvar
    d     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar
    o     :  array(FACILITIES, PERIODS, SCENARIOS) of mpvar
    y     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar


  end-declarations

  ! declare d(i,t) binary
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  d(i,t,s) is_binary
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  x(i,t,s) is_integer
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  y(i,t,s) is_integer


  ! generate random data from Unifrom distributions
```

```
  dem11 := 100 + (100*random)
  dem12 := 100 + (100*random)
  dem13 := 100 + (100*random)
  dem211:= 100 + (100*random)
  dem212:= 100 + (100*random)
  dem213:= 100 + (100*random)
  dem221:= 100 + (100*random)
  dem222:= 100 + (100*random)
  dem223:= 100 + (100*random)
  forall(s in SCENARIOS) r(1,1,s):= dem11
  forall(s in SCENARIOS) r(2,1,s):= dem12
  forall(s in SCENARIOS) r(3,1,s):= dem13
  forall(s in 1..2)  r(1,2,s):= dem211
  forall(s in 3..4)  r(1,2,s):= dem221
  forall(s in 1..2)  r(2,2,s):= dem212
  forall(s in 3..4)  r(2,2,s):= dem222
  forall(s in 1..2)  r(3,2,s):= dem213
  forall(s in 3..4)  r(3,2,s):= dem223
  forall(i in ITEMS, t in 3..5, s in SCENARIOS)  r(i,t,s):= 150 + (50*random)
  forall(i in ITEMS)  c(i):= 500 + (50*random)
  forall(k in FACILITIES, t in PERIODS)  co(k,t):= 30 + (10*random)
  forall(i in ITEMS, k in FACILITIES)  su(i,k):= 3 + (1*random)
  forall(i in ITEMS, k in FACILITIES)  b(i,k):= 0.15 + (0.05*random)
  forall(i in ITEMS)  h(i):= 2 + (1*random)
  forall(i in ITEMS)  inv0(i):= 600 + (30*random)
  forall(k in FACILITIES,t in PERIODS)  CAP(k,t):= 30 + (7*random)
  forall(i in ITEMS, t in PERIODS)  q(i,t):= 500 + (20*random)
  forall(i in ITEMS)  f(i):= 0.8 + (0.2*random)
  forall(i in ITEMS) L(i):= 1
  a(1,3) := 2
  a(2,3) := 1


  forall(s in SCENARIOS) p(s):=1/4


  ! Objective function
  Total_Expected_Cost:=
    sum(s in SCENARIOS) ( p(s) * (sum(i in ITEMS) sum(t in PERIODS)( c(i)*
d(i,t,s) + h(i) * y(i,t,s) ) + sum(k in FACILITIES) sum(t in PERIODS) (co(k,t)*
o(k,t,s))))

  ! demand constraint for every item i, period t under every scenario s
  forall(i in ITEMS,t in PERIODS, s in SCENARIOS)
    demand_constraints1(i,t,s) :=   f(i)* if((t-1)<1,0,x(i,t-1,s)) + if (t=1,inv0(i),y(i,t-
1,s)) - y(i,t,s) - sum(j in ITEMS2) a(i,j)* v(j,t,s) - r(i,t,s)  >= 0
```

```
  ! capacity constraint for every period and facility
  forall(k in FACILITIES, t in PERIODS, s in SCENARIOS)
    capacity_constraints(k,t) := sum(i in ITEMS) (su(i,k)*d(i,t,s) + b(i,k)*x(i,t,s)) -
o(k,t,s) <= CAP(k,t)

  ! production bound for every item i and period t
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)
    production_bound(i,t):= x(i,t,s) - q(i,t) * d(i,t,s) <= 0

  ! Equality constraint for new variables
  forall(t in PERIODS, s in SCENARIOS)do
    x(1,t,s) = v(1,t,s)
    x(2,t,s) = v(2,t,s)
    x(3,t,s) = v(3,t,s)
  end-do

  forall(i in ITEMS) do
    y(i,1,1) = y(i,1,2)
    y(i,1,2) = y(i,1,3)
    y(i,1,3) = y(i,1,4)
  end-do

  forall(i in ITEMS) do
    y(i,2,1) = y(i,2,2)
    y(i,2,3) = y(i,2,4)
  end-do

  forall(i in ITEMS) do
    x(i,1,1) = x(i,1,2)
    x(i,1,2) = x(i,1,3)
    x(i,1,3) = x(i,1,4)
  end-do

  forall(i in ITEMS) do
    x(i,2,1) = x(i,2,2)
    x(i,2,3) = x(i,2,4)
  end-do

  forall(k in FACILITIES) do
    o(k,1,1) = o(k,1,2)
    o(k,1,2) = o(k,1,3)
    o(k,1,3) = o(k,1,4)
  end-do

  forall(k in FACILITIES) do
    o(k,2,1) = o(k,2,2)
    o(k,2,3) = o(k,2,4)
```

```
end-do

forall(i in ITEMS) do
  d(i,1,1) = d(i,1,2)
  d(i,1,2) = d(i,1,3)
  d(i,1,3) = d(i,1,4)
end-do

forall(i in ITEMS) do
  d(i,2,1) = d(i,2,2)
  d(i,2,3) = d(i,2,4)
end-do

! model description complete

setparam("XPRS_verbose",false)   ! Enable message printing in mmxosl

! solve the model
minimize(Total_Expected_Cost)


writeln(" SOLUTION ")
writeln("Total Expected Costs are:  ", getobjval)
writeln

writeln("Inventory")
forall(s in SCENARIOS) do
  writeln("Scenario = ", s)
  forall(t in PERIODS) do
      forall(i in ITEMS) write(  strfmt(getsol(y(i,t,s)),7,1) , "  ")
      writeln
  end-do
  writeln
end-do

writeln("Production")
forall(s in SCENARIOS) do
  writeln("Scenario = ", s)
  forall(t in PERIODS) do
      forall(i in ITEMS) write(  strfmt(getsol(x(i,t,s)),7,1) , "  ")
      writeln
  end-do
  writeln
end-do

writeln("Overtime")
forall(s in SCENARIOS) do
```

```
      writeln("Scenario = ", s)
      forall(t in PERIODS) do
          forall(k in FACILITIES) write(  strfmt(getsol(o(k,t,s)),7,1) , "  ")
          writeln
      end-do
      writeln
    end-do

end-model
```

Código del Tercer Ejemplo Numérico Aleatorio con Recurso Completo

```
model ProAle6
  uses "mmxprs"

  declarations
    ! indicies
    ITEMS       = 1..3
    ITEMS2      = 1..3
    PERIODS     = 1..5
    FACILITIES  = 1..2
    SCENARIOS   = 1..6
    ! parameters
    c     :  array(ITEMS) of real
    su    :  array(ITEMS, FACILITIES) of real
    b     :  array(ITEMS, FACILITIES) of real
    h     :  array(ITEMS) of real
    r     :  array(ITEMS, PERIODS, SCENARIOS) of real
    CAP   :  array(FACILITIES, PERIODS) of real
    q     :  array(ITEMS, PERIODS) of real
    p     :  array(SCENARIOS) of real
    co    :  array(FACILITIES, PERIODS) of real
    a     :  array(ITEMS, ITEMS2) of real

    inv0  :  array(ITEMS) of real
    L     :  array(ITEMS) of real
    f     :  array(ITEMS) of real


    ! decision variables
    x     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar
    v     :  array(ITEMS2, PERIODS, SCENARIOS) of mpvar
    d     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar
    o     :  array(FACILITIES, PERIODS, SCENARIOS) of mpvar
    y     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar


  end-declarations

  ! declare d(i,t) binary
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  d(i,t,s) is_binary
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  x(i,t,s) is_integer
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  y(i,t,s) is_integer


  ! generate random data from Unifrom distributions
```

```
dem11 := 100 + (100*random)
dem12 := 100 + (100*random)
dem13 := 100 + (100*random)
dem211:= 100 + (100*random)
dem212:= 100 + (100*random)
dem213:= 100 + (100*random)
dem221:= 100 + (100*random)
dem222:= 100 + (100*random)
dem223:= 100 + (100*random)
dem311:= 100 + (100*random)
dem312:= 100 + (100*random)
dem313:= 100 + (100*random)
dem321:= 100 + (100*random)
dem322:= 100 + (100*random)
dem323:= 100 + (100*random)
forall(s in SCENARIOS) r(1,1,s):= dem11
forall(s in SCENARIOS) r(2,1,s):= dem12
forall(s in SCENARIOS) r(3,1,s):= dem13
forall(s in 1..3)  r(1,2,s):= dem211
forall(s in 4..6)  r(1,2,s):= dem221
forall(s in 1..3)  r(2,2,s):= dem212
forall(s in 4..6)  r(2,2,s):= dem222
forall(s in 1..3)  r(3,2,s):= dem213
forall(s in 4..6)  r(3,2,s):= dem223
forall(s in 1..2)  r(1,3,s):= dem311
forall(s in 4..5)  r(1,3,s):= dem321
forall(s in 1..2)  r(2,3,s):= dem312
forall(s in 4..5)  r(2,3,s):= dem322
forall(s in 1..2)  r(3,3,s):= dem313
forall(s in 4..5)  r(3,3,s):= dem323
forall(i in ITEMS) r(i,3,3):= 100 + (100*random)
forall(i in ITEMS) r(i,3,6):= 100 + (100*random)
forall(i in ITEMS, t in 4..5, s in SCENARIOS)  r(i,t,s):= 150 + (50*random)
forall(i in ITEMS)  c(i):= 500 + (50*random)
forall(k in FACILITIES, t in PERIODS)  co(k,t):= 30 + (10*random)
forall(i in ITEMS, k in FACILITIES)  su(i,k):= 3 + (1*random)
forall(i in ITEMS, k in FACILITIES)  b(i,k):= 0.15 + (0.05*random)
forall(i in ITEMS)  h(i):= 2 + (1*random)
forall(i in ITEMS)  inv0(i):= 600 + (30*random)
forall(k in FACILITIES,t in PERIODS)  CAP(k,t):= 30 + (7*random)
forall(i in ITEMS, t in PERIODS)  q(i,t):= 500 + (20*random)
forall(i in ITEMS)  f(i):= 0.8 + (0.2*random)
forall(i in ITEMS) L(i):= 1
a(1,3) := 2
a(2,3) := 1
```

```
  forall(s in SCENARIOS) p(s):=1/6


  ! Objective function
  Total_Expected_Cost:=
    sum(s in SCENARIOS) ( p(s) * (sum(i in ITEMS) sum(t in PERIODS)( c(i)*
d(i,t,s) + h(i) * y(i,t,s) ) + sum(k in FACILITIES) sum(t in PERIODS) (co(k,t)*
o(k,t,s))))

  ! demand constraint for every item i, period t under every scenario s
  forall(i in ITEMS,t in PERIODS, s in SCENARIOS)
    demand_constraints1(i,t,s) :=   f(i)* if((t-1)<1,0,x(i,t-1,s)) + if (t=1,inv0(i),y(i,t-
1,s)) - y(i,t,s) - sum(j in ITEMS2) a(i,j)* v(j,t,s) - r(i,t,s)  >= 0

  ! capacity constraint for every period and facility
  forall(k in FACILITIES, t in PERIODS, s in SCENARIOS)
    capacity_constraints(k,t) := sum(i in ITEMS) (su(i,k)*d(i,t,s) + b(i,k)*x(i,t,s)) -
o(k,t,s) <= CAP(k,t)

  ! production bound for every item i and period t
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)
    production_bound(i,t):= x(i,t,s) - q(i,t) * d(i,t,s) <= 0

  ! Equality constraint for new variables
  forall(t in PERIODS, s in SCENARIOS)do
    x(1,t,s) = v(1,t,s)
    x(2,t,s) = v(2,t,s)
    x(3,t,s) = v(3,t,s)
  end-do

  forall(i in ITEMS) do
    y(i,1,1) = y(i,1,2)
    y(i,1,2) = y(i,1,3)
    y(i,1,3) = y(i,1,4)
    y(i,1,4) = y(i,1,5)
    y(i,1,5) = y(i,1,6)
  end-do

  forall(i in ITEMS) do
    y(i,2,1) = y(i,2,2)
    y(i,2,2) = y(i,2,3)
    y(i,2,4) = y(i,2,5)
    y(i,2,5) = y(i,2,6)
  end-do

  forall(i in ITEMS) do
    y(i,3,1) = y(i,3,2)
```

```
   y(i,3,4) = y(i,3,5)
end-do

forall(i in ITEMS) do
   x(i,1,1) = x(i,1,2)
   x(i,1,2) = x(i,1,3)
   x(i,1,3) = x(i,1,4)
   x(i,1,4) = x(i,1,5)
   x(i,1,5) = x(i,1,6)
end-do

forall(i in ITEMS) do
   x(i,2,1) = x(i,2,2)
   x(i,2,2) = x(i,2,3)
   x(i,2,4) = x(i,2,5)
   x(i,2,5) = x(i,2,6)
end-do

forall(i in ITEMS) do
   x(i,3,1) = x(i,3,2)
   x(i,3,4) = x(i,3,5)
end-do

forall(k in FACILITIES) do
   o(k,1,1) = o(k,1,2)
   o(k,1,2) = o(k,1,3)
   o(k,1,3) = o(k,1,4)
   o(k,1,4) = o(k,1,5)
   o(k,1,5) = o(k,1,6)
end-do

forall(k in FACILITIES) do
   o(k,2,1) = o(k,2,2)
   o(k,2,2) = o(k,2,3)
   o(k,2,4) = o(k,2,5)
   o(k,2,5) = o(k,2,6)
end-do

forall(k in FACILITIES) do
   o(k,3,1) = o(k,3,2)
   o(k,3,4) = o(k,3,5)
end-do

forall(i in ITEMS) do
   d(i,1,1) = d(i,1,2)
   d(i,1,2) = d(i,1,3)
   d(i,1,3) = d(i,1,4)
```

```
  d(i,1,4) = d(i,1,5)
  d(i,1,5) = d(i,1,6)
end-do

forall(i in ITEMS) do
  d(i,2,1) = d(i,2,2)
  d(i,2,2) = d(i,2,3)
  d(i,2,4) = d(i,2,5)
  d(i,2,5) = d(i,2,6)
end-do

forall(i in ITEMS) do
  d(i,3,1) = d(i,3,2)
  d(i,3,4) = d(i,3,5)
end-do


! model description complete

setparam("XPRS_verbose",false)   ! Enable message printing in mmxosl

! solve the model
minimize(Total_Expected_Cost)


writeln(" SOLUTION ")
writeln("Total Expected Costs are:  ", getobjval)
writeln

writeln("Inventory")
forall(s in SCENARIOS) do
  writeln("Scenario = ", s)
  forall(t in PERIODS) do
      forall(i in ITEMS) write(  strfmt(getsol(y(i,t,s)),7,1) , "  ")
      writeln
  end-do
  writeln
end-do

writeln("Production")
forall(s in SCENARIOS) do
  writeln("Scenario = ", s)
  forall(t in PERIODS) do
      forall(i in ITEMS) write(  strfmt(getsol(x(i,t,s)),7,1) , "  ")
      writeln
  end-do
  writeln
```

```
  end-do

  writeln("Overtime")
  forall(s in SCENARIOS) do
    writeln("Scenario = ", s)
    forall(t in PERIODS) do
        forall(k in FACILITIES) write(  strfmt(getsol(o(k,t,s)),7,1) , "  ")
        writeln
    end-do
    writeln
  end-do

end-model
```

ANEXO C. Soluciones dadas para los Ejemplos Aleatorios por los Códigos Desarrollados

Solución del Primer Ejemplo Numérico Aleatorio con Recurso Simple

INPUT

Scenario = 1
Period = 1
Demand Item 1=153.456
Demand Item 2=165.166
Demand Item 3=107.102
Demand Item 4=163.741
Period = 2
Demand Item 1=152.17
Demand Item 2=166.266
Demand Item 3=197.603
Demand Item 4=175.226
Period = 3
Demand Item 1=183.259
Demand Item 2=192.035
Demand Item 3=151.832
Demand Item 4=181.517
Period = 4
Demand Item 1=188.999
Demand Item 2=178.141
Demand Item 3=156.67
Demand Item 4=164.574

Scenario = 2

Period = 1

Demand Item 1=153.456

Demand Item 2=165.166

Demand Item 3=107.102

Demand Item 4=163.741

Period = 2

Demand Item 1=194.91

Demand Item 2=164.781

Demand Item 3=158.771

Demand Item 4=171.74

Period = 3

Demand Item 1=187.613

Demand Item 2=173.652

Demand Item 3=174.13

Demand Item 4=162.955

Period = 4

Demand Item 1=192.981

Demand Item 2=167.394

Demand Item 3=196.225

Demand Item 4=178.369

Scenario = 3

Period = 1

Demand Item 1=153.456

Demand Item 2=165.166

Demand Item 3=107.102

Demand Item 4=163.741

Period = 2

Demand Item 1=152.687

Demand Item 2=173.558

Demand Item 3=182.913

Demand Item 4=161.046

Period = 3

Demand Item 1=153.11

Demand Item 2=165.559

Demand Item 3=158.451

Demand Item 4=189.947

Period = 4

Demand Item 1=156.047

Demand Item 2=157.614

Demand Item 3=166.699

Demand Item 4=190.155

Set-up Cost Item 1 = 516.719

Set-up Cost Item 2 = 537.23

Set-up Cost Item 3 = 525.396

Set-up Cost Item 4 = 526.796

Unit Holding Cost Item 1 = 2.65677

Unit Holding Cost Item 2 = 2.02252

Unit Holding Cost Item 3 = 2.12703

Unit Holding Cost Item 4 = 2.7704

Inventory 0 Item 1 = 606.483

Inventory 0 Item 2 = 609.885

Inventory 0 Item 3 = 621.922

Inventory 0 Item 4 = 628.002

Lead Time Item 1 = 1

Lead Time Item 2 = 1

Lead Time Item 3 = 1

Lead Time Item 4 = 2

Yield Item 1 = 0.867805

Yield Item 2 = 0.859392
Yield Item 3 = 0.812569
Yield Item 4 = 0.832641

Facility = 1
Set-up Resource Utilization Item 1 = 3.23386
Set-up Resource Utilization Item 2 = 3.43978
Set-up Resource Utilization Item 3 = 3.78062
Set-up Resource Utilization Item 4 = 3.40631
Capacity Utilization Rate Item 1 = 0.159693
Capacity Utilization Rate Item 2 = 0.19422
Capacity Utilization Rate Item 3 = 0.199503
Capacity Utilization Rate Item 4 = 0.154903

Facility = 2
Set-up Resource Utilization Item 1 = 3.19398
Set-up Resource Utilization Item 2 = 3.87416
Set-up Resource Utilization Item 3 = 3.1757
Set-up Resource Utilization Item 4 = 3.08033
Capacity Utilization Rate Item 1 = 0.163605
Capacity Utilization Rate Item 2 = 0.178443
Capacity Utilization Rate Item 3 = 0.168302
Capacity Utilization Rate Item 4 = 0.192811

Facility = 3
Set-up Resource Utilization Item 1 = 3.41888
Set-up Resource Utilization Item 2 = 3.0702
Set-up Resource Utilization Item 3 = 3.99564
Set-up Resource Utilization Item 4 = 3.08236
Capacity Utilization Rate Item 1 = 0.177612
Capacity Utilization Rate Item 2 = 0.159919

Capacity Utilization Rate Item 3 = 0.163787
Capacity Utilization Rate Item 4 = 0.180142

Period = 1
Capacity at Facility 1 = 30.8241
Capacity at Facility 2 = 32.8449
Capacity at Facility 3 = 31.7819
Upper Bound Item 1 = 510.505
Upper Bound Item 2 = 504.01
Upper Bound Item 3 = 506.336
Upper Bound Item 4 = 516.883

Period = 2
Capacity at Facility 1 = 32.5246
Capacity at Facility 2 = 30.0426
Capacity at Facility 3 = 32.6782
Upper Bound Item 1 = 506.112
Upper Bound Item 2 = 504.822
Upper Bound Item 3 = 511.897
Upper Bound Item 4 = 510.458

Period = 3
Capacity at Facility 1 = 35.6203
Capacity at Facility 2 = 35.0722
Capacity at Facility 3 = 30.7906
Upper Bound Item 1 = 504.167
Upper Bound Item 2 = 507.396
Upper Bound Item 3 = 518.626
Upper Bound Item 4 = 502.137

Period = 4

Capacity at Facility 1 = 32.219

Capacity at Facility 2 = 34.4598

Capacity at Facility 3 = 35.936

Upper Bound Item 1 = 517.08

Upper Bound Item 2 = 505.559

Upper Bound Item 3 = 500.648

Upper Bound Item 4 = 517.42

SOLUTION

Total Expected Costs are: 28548.2

Period = 1

Produced Item 1 = 393

Produced Item 2 = 282

Produced Item 3 = 137

Produced Item 4 = 0

Overtime Capacity Facility 1 = 124.492

Overtime Capacity Facility 2 = 115.074

Overtime Capacity Facility 3 = 116.04

Period = 2

Produced Item 1 = 439

Produced Item 2 = 0

Produced Item 3 = 0

Produced Item 4 = 94

Overtime Capacity Facility 1 = 58.7816

Overtime Capacity Facility 2 = 66.1785

Overtime Capacity Facility 3 = 68.7279

Period = 3

Produced Item 1 = 0

Produced Item 2 = 205

Produced Item 3 = 0

Produced Item 4 = 0

Overtime Capacity Facility 1 = 7.63462

Overtime Capacity Facility 2 = 5.38275

Overtime Capacity Facility 3 = 5.06312

Period = 4

Produced Item 1 = 0

Produced Item 2 = 0

Produced Item 3 = 0

Produced Item 4 = 0

Overtime Capacity Facility 1 = 0

Overtime Capacity Facility 2 = 0

Overtime Capacity Facility 3 = 0

Scenario = 1

42.0	307.0	514.0	464.0
0.0	195.0	309.0	269.0
189.0	2.0	157.0	87.0
0.0	0.0	0.0	0.0

Scenario = 2

42.0	307.0	514.0	464.0
0.0	174.0	372.0	264.0
193.0	0.0	197.0	101.0
0.0	0.0	0.0	0.0

Scenario = 3

42.0	307.0	514.0	464.0
0.0	166.0	326.0	302.0
157.0	0.0	167.0	112.0
0.0	0.0	0.0	0.0

Solución del Segundo Ejemplo Numérico Aleatorio con Recurso Simple

INPUT

Scenario = 1
Period = 1
Demand Item 1=122.093
Demand Item 2=129.562
Demand Item 3=111.294
Period = 2
Demand Item 1=122.953
Demand Item 2=189.82
Demand Item 3=172.203
Period = 3
Demand Item 1=153.11
Demand Item 2=192.981
Demand Item 3=156.67
Period = 4
Demand Item 1=178.443
Demand Item 2=165.559
Demand Item 3=158.785
Period = 5
Demand Item 1=193.86
Demand Item 2=182.839
Demand Item 3=173.558

Scenario = 2
Period = 1
Demand Item 1=122.093
Demand Item 2=129.562
Demand Item 3=111.294

Period = 2
Demand Item 1=122.953
Demand Item 2=189.82
Demand Item 3=172.203
Period = 3
Demand Item 1=176.728
Demand Item 2=158.771
Demand Item 3=155.887
Period = 4
Demand Item 1=187.613
Demand Item 2=156.351
Demand Item 3=190.155
Period = 5
Demand Item 1=152.17
Demand Item 2=174.13
Demand Item 3=199.782

Scenario = 3
Period = 1
Demand Item 1=122.093
Demand Item 2=129.562
Demand Item 3=111.294
Period = 2
Demand Item 1=147.303
Demand Item 2=134.787
Demand Item 3=166.063
Period = 3
Demand Item 1=153.551
Demand Item 2=189.947
Demand Item 3=192.811

Period = 4

Demand Item 1=156.047

Demand Item 2=181.517

Demand Item 3=181.4

Period = 5

Demand Item 1=188.999

Demand Item 2=162.955

Demand Item 3=191.702

Scenario = 4

Period = 1

Demand Item 1=122.093

Demand Item 2=129.562

Demand Item 3=111.294

Period = 2

Demand Item 1=147.303

Demand Item 2=134.787

Demand Item 3=166.063

Period = 3

Demand Item 1=182.913

Demand Item 2=152.687

Demand Item 3=178.369

Period = 4

Demand Item 1=197.603

Demand Item 2=175.396

Demand Item 3=159.847

Period = 5

Demand Item 1=153.51

Demand Item 2=157.614

Demand Item 3=166.699

Set-up Cost Item 1 = 531.87
Set-up Cost Item 2 = 516.719
Set-up Cost Item 3 = 521.84
Unit Holding Cost Item 1 = 2.02252
Unit Holding Cost Item 2 = 2.32532
Unit Holding Cost Item 3 = 2.3056
Inventory 0 Item 1 = 611.094
Inventory 0 Item 2 = 619.955
Inventory 0 Item 3 = 618.562
Lead Time Item 1 = 1
Lead Time Item 2 = 1
Lead Time Item 3 = 1
Yield Item 1 = 0.863358
Yield Item 2 = 0.985999
Yield Item 3 = 0.869091

Facility = 1
Set-up Resource Utilization Item 1 = 3.23386
Set-up Resource Utilization Item 2 = 3.91784
Set-up Resource Utilization Item 3 = 3.19398
Capacity Utilization Rate Item 1 = 0.196669
Capacity Utilization Rate Item 2 = 0.18852
Capacity Utilization Rate Item 3 = 0.196225

Facility = 2
Set-up Resource Utilization Item 1 = 3.29708
Set-up Resource Utilization Item 2 = 3.98975
Set-up Resource Utilization Item 3 = 3.87416
Capacity Utilization Rate Item 1 = 0.163605
Capacity Utilization Rate Item 2 = 0.153014
Capacity Utilization Rate Item 3 = 0.154118

Period = 1
Capacity at Facility 1 = 35.311
Capacity at Facility 2 = 34.5616
Upper Bound Item 1 = 500.122
Upper Bound Item 2 = 519.801
Upper Bound Item 3 = 512.057

Period = 2
Capacity at Facility 1 = 31.4987
Capacity at Facility 2 = 32.4456
Upper Bound Item 1 = 503.264
Upper Bound Item 2 = 501.607
Upper Bound Item 3 = 506.663

Period = 3
Capacity at Facility 1 = 31.0544
Capacity at Facility 2 = 30.4239
Upper Bound Item 1 = 508.378
Upper Bound Item 2 = 503.877
Upper Bound Item 3 = 500.648

Period = 4
Capacity at Facility 1 = 35.2122
Capacity at Facility 2 = 32.5623
Upper Bound Item 1 = 515.612
Upper Bound Item 2 = 503.313
Upper Bound Item 3 = 508.2

Period = 5
Capacity at Facility 1 = 33.0859
Capacity at Facility 2 = 36.5191

Upper Bound Item 1 = 506.59

Upper Bound Item 2 = 517.08

Upper Bound Item 3 = 508.128

SOLUTION

Total Expected Costs are: 20163.2

Period = 1

Produced Item 1 = 384

Produced Item 2 = 0

Produced Item 3 = 243

Overtime Capacity Facility 1 = 94.3204

Overtime Capacity Facility 2 = 72.8845

Period = 2

Produced Item 1 = 0

Produced Item 2 = 324

Produced Item 3 = 0

Overtime Capacity Facility 1 = 33.4996

Overtime Capacity Facility 2 = 21.1207

Period = 3

Produced Item 1 = 442

Produced Item 2 = 0

Produced Item 3 = 0

Overtime Capacity Facility 1 = 59.1073

Overtime Capacity Facility 2 = 45.1865

Period = 4

Produced Item 1 = 0

Produced Item 2 = 186

Produced Item 3 = 17

Overtime Capacity Facility 1 = 10.3001

Overtime Capacity Facility 2 = 6.38228

Period = 5

Produced Item 1 = 0

Produced Item 2 = 0

Produced Item 3 = 0

Overtime Capacity Facility 1 = 0

Overtime Capacity Facility 2 = 0

Scenario = 1

3.0	247.0	507.0
179.0	57.0	533.0
25.0	183.0	318.0
194.0	0.0	159.0
0.0	0.0	0.0

Scenario = 2

3.0	247.0	507.0
179.0	57.0	533.0
0.0	174.0	377.0
153.0	0.0	186.0
0.0	0.0	0.0

Scenario = 3

3.0	247.0	507.0
187.0	70.0	552.0
0.0	199.0	359.0
189.0	0.0	177.0
0.0	0.0	0.0

Scenario = 4

3.0	247.0	507.0
187.0	70.0	552.0
4.0	193.0	312.0
154.0	0.0	152.0
0.0	0.0	0.0

Solución del Tercer Ejemplo Numérico Aleatorio con Recurso Simple

INPUT

Scenario = 1
Period = 1
Demand Item 1=153.456
Demand Item 2=122.953
Demand Item 3=144.084
Period = 2
Demand Item 1=132.532
Demand Item 2=195.206
Demand Item 3=165.677
Period = 3
Demand Item 1=166.517
Demand Item 2=106.028
Demand Item 3=185.961
Period = 4
Demand Item 1=152.17
Demand Item 2=182.913
Demand Item 3=186.537
Period = 5
Demand Item 1=155.887
Demand Item 2=171.84
Demand Item 3=153.551

Scenario = 2
Period = 1
Demand Item 1=153.456
Demand Item 2=122.953
Demand Item 3=144.084

Period = 2

Demand Item 1=132.532

Demand Item 2=195.206

Demand Item 3=165.677

Period = 3

Demand Item 1=166.517

Demand Item 2=106.028

Demand Item 3=185.961

Period = 4

Demand Item 1=196.225

Demand Item 2=188.52

Demand Item 3=155.647

Period = 5

Demand Item 1=188.999

Demand Item 2=156.67

Demand Item 3=170.316

Scenario = 3

Period = 1

Demand Item 1=153.456

Demand Item 2=122.953

Demand Item 3=144.084

Period = 2

Demand Item 1=132.532

Demand Item 2=195.206

Demand Item 3=165.677

Period = 3

Demand Item 1=117.57

Demand Item 2=180.31

Demand Item 3=166.063

Period = 4

Demand Item 1=167.394

Demand Item 2=165.85

Demand Item 3=159.919

Period = 5

Demand Item 1=161.046

Demand Item 2=177.612

Demand Item 3=165.28

Scenario = 4

Period = 1

Demand Item 1=153.456

Demand Item 2=122.953

Demand Item 3=144.084

Period = 2

Demand Item 1=129.562

Demand Item 2=129.708

Demand Item 3=116.901

Period = 3

Demand Item 1=143.481

Demand Item 2=163.741

Demand Item 3=147.116

Period = 4

Demand Item 1=156.351

Demand Item 2=178.141

Demand Item 3=187.23

Period = 5

Demand Item 1=192.811

Demand Item 2=153.51

Demand Item 3=178.369

Scenario = 5

Period = 1

Demand Item 1=153.456

Demand Item 2=122.953

Demand Item 3=144.084

Period = 2

Demand Item 1=129.562

Demand Item 2=129.708

Demand Item 3=116.901

Period = 3

Demand Item 1=143.481

Demand Item 2=163.741

Demand Item 3=147.116

Period = 4

Demand Item 1=178.443

Demand Item 2=166.719

Demand Item 3=168.49

Period = 5

Demand Item 1=195.892

Demand Item 2=151.832

Demand Item 3=151.126

Scenario = 6

Period = 1

Demand Item 1=153.456

Demand Item 2=122.953

Demand Item 3=144.084

Period = 2

Demand Item 1=129.562

Demand Item 2=129.708

Demand Item 3=116.901

Period = 3

Demand Item 1=162.801

Demand Item 2=119.693

Demand Item 3=125.91

Period = 4

Demand Item 1=181.517

Demand Item 2=193.708

Demand Item 3=198.499

Period = 5

Demand Item 1=175.123

Demand Item 2=161.693

Demand Item 3=174.13

Set-up Cost Item 1 = 542.207

Set-up Cost Item 2 = 509.699

Set-up Cost Item 3 = 504.017

Unit Holding Cost Item 1 = 2.36065

Unit Holding Cost Item 2 = 2.71301

Unit Holding Cost Item 3 = 2.9313

Inventory 0 Item 1 = 619.114

Inventory 0 Item 2 = 602.235

Inventory 0 Item 3 = 625.62

Lead Time Item 1 = 1

Lead Time Item 2 = 1

Lead Time Item 3 = 1

Yield Item 1 = 0.982977

Yield Item 2 = 0.889062

Yield Item 3 = 0.900905

Facility = 1
Set-up Resource Utilization Item 1 = 3.27573
Set-up Resource Utilization Item 2 = 3.99564
Set-up Resource Utilization Item 3 = 3.06055
Capacity Utilization Rate Item 1 = 0.167469
Capacity Utilization Rate Item 2 = 0.189947
Capacity Utilization Rate Item 3 = 0.151621

Facility = 2
Set-up Resource Utilization Item 1 = 3.8029
Set-up Resource Utilization Item 2 = 3.08236
Set-up Resource Utilization Item 3 = 3.93339
Capacity Utilization Rate Item 1 = 0.180142
Capacity Utilization Rate Item 2 = 0.167273
Capacity Utilization Rate Item 3 = 0.170321

Period = 1
Capacity at Facility 1 = 36.9305
Capacity at Facility 2 = 32.8699
Upper Bound Item 1 = 506.663
Upper Bound Item 2 = 500.233
Upper Bound Item 3 = 500.122

Period = 2
Capacity at Facility 1 = 30.6322
Capacity at Facility 2 = 35.5468
Upper Bound Item 1 = 507.652
Upper Bound Item 2 = 503.877
Upper Bound Item 3 = 513.229

Period = 3

Capacity at Facility 1 = 31.9287

Capacity at Facility 2 = 31.9457

Upper Bound Item 1 = 517.964

Upper Bound Item 2 = 510.505

Upper Bound Item 3 = 504.167

Period = 4

Capacity at Facility 1 = 35.0722

Capacity at Facility 2 = 30.4399

Upper Bound Item 1 = 513.033

Upper Bound Item 2 = 517.688

Upper Bound Item 3 = 519.44

Period = 5

Capacity at Facility 1 = 31.0544

Capacity at Facility 2 = 30.748

Upper Bound Item 1 = 506.336

Upper Bound Item 2 = 511.676

Upper Bound Item 3 = 504.01

SOLUTION

Total Expected Costs are: 19080.7

Period = 1

Produced Item 1 = 310

Produced Item 2 = 0

Produced Item 3 = 232

Overtime Capacity Facility 1 = 56.4971

Overtime Capacity Facility 2 = 70.2249

Period = 2
Produced Item 1 = 0
Produced Item 2 = 353
Produced Item 3 = 0
Overtime Capacity Facility 1 = 40.4146
Overtime Capacity Facility 2 = 26.5829

Period = 3
Produced Item 1 = 392
Produced Item 2 = 0
Produced Item 3 = 3
Overtime Capacity Facility 1 = 40.5101
Overtime Capacity Facility 2 = 46.9174

Period = 4
Produced Item 1 = 0
Produced Item 2 = 183
Produced Item 3 = 0
Overtime Capacity Facility 1 = 3.68366
Overtime Capacity Facility 2 = 3.25337

Period = 5
Produced Item 1 = 0
Produced Item 2 = 0
Produced Item 3 = 0
Overtime Capacity Facility 1 = 0
Overtime Capacity Facility 2 = 0

Scenario = 1

1.0	247.0	481.0
173.0	51.0	524.0
0.0	193.0	338.0
156.0	10.0	154.0
0.0	0.0	0.0

Scenario = 2

1.0	247.0	481.0
173.0	51.0	524.0
0.0	193.0	338.0
189.0	0.0	171.0
0.0	0.0	0.0

Scenario = 3

1.0	247.0	481.0
173.0	51.0	524.0
0.0	181.0	324.0
162.0	15.0	166.0
0.0	0.0	0.0

Scenario = 4

1.0	247.0	481.0
169.0	32.0	512.0
0.0	179.0	364.0
193.0	0.0	179.0
0.0	0.0	0.0

Scenario = 5

1.0	247.0	481.0
169.0	32.0	512.0
0.0	179.0	364.0
196.0	0.0	152.0
0.0	0.0	0.0

Scenario = 6

1.0	247.0	481.0
169.0	32.0	512.0
0.0	194.0	371.0
176.0	0.0	175.0
0.0	0.0	0.0

Solución del Primer Ejemplo Numérico Aleatorio con Recurso Completo

SOLUTION

Total Expected Costs are: 17950.4

Inventory

Scenario = 1

450.0	451.0	472.0	442.0
0.0	0.0	113.0	250.0
0.0	0.0	0.0	94.0
0.0	0.0	0.0	0.0

Scenario = 2

450.0	451.0	472.0	442.0
0.0	0.0	123.0	269.0
0.0	0.0	0.0	108.0
0.0	0.0	0.0	0.0

Scenario = 3

450.0	451.0	472.0	442.0
343.0	66.0	198.0	284.0
0.0	0.0	13.0	91.0
0.0	0.0	0.0	0.0

Production

Scenario = 1

253.0	0.0	0.0	0.0
201.0	95.0	104.0	94.0
0.0	157.0	210.0	0.0
0.0	0.0	0.0	0.0

Scenario = 2

253.0	0.0	0.0	0.0
435.0	173.0	66.0	99.0
137.0	203.0	240.0	0.0

0.0 0.0 0.0 0.0

Scenario = 3

253.0 0.0 0.0 0.0

508.0 349.0 0.0 96.0

177.0 213.0 219.0 0.0

0.0 0.0 0.0 0.0

Overtime

Scenario = 1

11.4 11.4 21.4

0.0 0.0 0.0

0.0 0.0 0.0

0.0 0.0 0.0

Scenario = 2

11.4 11.4 21.4

0.0 0.0 0.0

0.0 0.0 0.0

0.0 0.0 0.0

Scenario = 3

11.4 11.4 21.4

138.0 152.6 149.4

84.5 84.7 85.5

0.0 0.0 0.0

Solución del Segundo Ejemplo Numérico Aleatorio con Recurso Completo

SOLUTION

Total Expected Costs are: 7934.31

Inventory

Scenario = 1

191.0	300.0	469.0
0.0	0.0	140.0
0.0	0.0	0.0
0.0	0.0	0.0
0.0	0.0	0.0

Scenario = 2

191.0	300.0	469.0
0.0	0.0	140.0
0.0	0.0	0.0
0.0	0.0	0.0
0.0	0.0	0.0

Scenario = 3

191.0	300.0	469.0
8.0	182.0	456.0
0.0	0.0	176.0
0.0	0.0	0.0
0.0	0.0	0.0

Scenario = 4

191.0	300.0	469.0
8.0	182.0	456.0
20.0	0.0	282.0
159.0	0.0	119.0
0.0	0.0	0.0

Production

Scenario = 1

0.0 0.0 153.0

189.0 65.0 28.0

90.0 0.0 212.0

0.0 56 .0 209.0

0.0 0.0 0.0

Scenario = 2

0.0 0.0 153.0

189.0 65.0 28.0

0.0 107.0 214.0

0.0 187.0 191.0

0.0 0.0 0.0

Scenario = 3

0.0 0.0 153.0

219.0 0.0 0.0

487.0 214.0 0.0

0.0 151.0 202.0

0.0 0.0 0.0

Scenario = 4

0.0 0.0 153.0

219.0 0.0 0.0

505.0 227.0 0.0

0.0 169.0 58.0

0.0 0.0 0.0

Overtime

Scenario = 1

0.0 0.0

0.0 0.0

0.0 0.0

0.0 0.0

0.0 0.0

Scenario = 2

0.0 0.0

0.0 0.0

0.0 0.0

0.0 0.0

0.0 0.0

Scenario = 3

0.0 0.0

5.4 11.6

0.0 0.0

0.0 0.0

0.0 0.0

Scenario = 4

0.0 0.0

5.4 11.6

95.5 108.7

11.3 9.7

0.0 0.0

Solución del Tercer Ejemplo Numérico Aleatorio con Recurso Completo

SOLUTION

Total Expected Costs are: 6286.48

Inventory

Scenario = 1

211.0	306.0	474.0
0.0	0.0	97.0
0.0	0.0	0.0
0.0	0.0	0.0
0.0	0.0	0.0

Scenario = 2

211.0	306.0	474.0
0.0	0.0	97.0
0.0	0.0	0.0
0.0	0.0	0.0
0.0	0.0	0.0

Scenario = 3

211.0	306.0	474.0
0.0	0.0	97.0
0.0	0.0	0.0
0.0	0.0	0.0
0.0	0.0	0.0

Scenario = 4

211.0	306.0	474.0
36.0	167.0	399.0
0.0	0.0	170.0
0.0	0.0	0.0
0.0	0.0	0.0

Scenario = 5

211.0	306.0	474.0

36.0 167.0 399.0
0.0 0.0 170.0
0.0 0.0 0.0
0.0 0.0 0.0

Scenario = 6

211.0 306.0 474.0
36.0 167.0 399.0
69.0 32.0 252.0
0.0 180.0 70.0
0.0 0.0 0.0

Production

Scenario = 1

0.0 0.0 136.0
159.0 0.0 18.0
395.0 432.0 214.0
187.0 0.0 217.0
0.0 0.0 0.0

Scenario = 2

0.0 0.0 136.0
159.0 0.0 18.0
395.0 432.0 214.0
239.0 0.0 228.0
0.0 0.0 0.0

Scenario = 3

0.0 0.0 136.0
159.0 0.0 18.0
354.0 307.0 223.0
253.0 38.0 196.0
0.0 0.0 0.0

Scenario = 4

0.0 0.0 136.0

192.0 0.0 0.0

382.0 405.0 0.0

202.0 0.0 197.0

0.0 0.0 0.0

Scenario = 5

0.0 0.0 136.0

192.0 0.0 0.0

382.0 405.0 0.0

233.0 0.0 211.0

0.0 0.0 0.0

Scenario = 6

0.0 0.0 136.0

192.0 0.0 0.0

401.0 444.0 0.0

203.0 0.0 105.0

0.0 0.0 0.0

Overtime

Scenario = 1

0.0 0.0

0.0 0.0

0.0 0.0

0.0 0.0

0.0 0.0

Scenario = 2

0.0 0.0

0.0 0.0

0.0 0.0

0.0 0.0

0.0 0.0

Scenario = 3

0.0 0.0

0.0 0.0

0.0 0.0

0.0 0.0

0.0 0.0

Scenario = 4

0.0 0.0

2.2 1.8

0.0 0.0

0.0 0.0

0.0 0.0

Scenario = 5

0.0 0.0

2.2 1.8

0.0 0.0

0.0 0.0

0.0 0.0

Scenario = 6

0.0 0.0

2.2 1.8

130.3 124.7

28.9 28.0

0.0 0.0

ANEXO D. Condiciones de Optimalidad de Karush-Kuhn-Tucker

Dado un problema de la forma

Min f(x)

Sujeto a $g_1(x) \leq 0$

.....

$g_m(x) \leq 0$

que tiene un mínimo local en el punto y. Si los vectores gradientes de las restricciones saturadas en y son linealmente independientes en y entonces cada restricción saturada tiene asociado un número positivo $l_i \geq 0$ (conocido como multiplicador de Kuhn-Tucker) de tal forma que:

$$\text{Gradiente } f(y) + l_1 * \text{Gradiente } g_1(y) + ... + l_m * \text{Gradiente } g_m(y) = 0$$

Para que en la expresión anterior intervengan todas las restricciones se asocian multiplicadores nulos a las restricciones no saturadas, para ello se imponen las condiciones:

$$l_i * g_i(y) = 0 \qquad \text{para } i=1,...,m$$

ANEXO E. Códigos para resolver los Ejemplos No Aleatorios con Programación Estocástica

Código del Primer Ejemplo Numérico No Aleatorio con Recurso Simple

```
model ProFijo1
  uses "mmxprs"

  declarations
    ! indicies
    ITEMS       = 1..3
    ITEMS2      = 1..3
    PERIODS     = 1..5
    FACILITIES  = 1..2
    SCENARIOS   = 1..4

    ! parameters
    c     :  array(ITEMS) of real
    su    :  array(ITEMS, FACILITIES) of real
    b     :  array(ITEMS, FACILITIES) of real
    h     :  array(ITEMS) of real
    r     :  array(ITEMS, PERIODS, SCENARIOS) of real
    CAP   :  array(FACILITIES, PERIODS) of real
    q     :  array(ITEMS, PERIODS) of real
    p     :  array(SCENARIOS) of real
    co    :  array(FACILITIES, PERIODS) of real
    a     :  array(ITEMS, ITEMS2) of real

    inv0  :  array(ITEMS) of real
    L     :  array(ITEMS) of real
    f     :  array(ITEMS) of real

    ! decision variables
    x     :  array(ITEMS, PERIODS) of mpvar
    v     :  array(ITEMS2, PERIODS) of mpvar
    d     :  array(ITEMS, PERIODS) of mpvar
    o     :  array(FACILITIES, PERIODS) of mpvar
    y     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar
```

```
end-declarations

! declare d(i,t) binary
forall(i in ITEMS, t in PERIODS)  d(i,t) is_binary
forall(i in ITEMS, t in PERIODS)  x(i,t) is_integer
forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  y(i,t,s) is_integer


! generate random data from Unifrom distributions
forall(s in SCENARIOS) r(1,1,s):= 200
forall(s in SCENARIOS) r(2,1,s):= 180
forall(s in SCENARIOS) r(3,1,s):= 230
forall(s in 1..2)  r(1,2,s):= 220
forall(s in 3..4)  r(1,2,s):= 180
forall(s in 1..2)  r(2,2,s):= 200
forall(s in 3..4)  r(2,2,s):= 170
forall(s in 1..2)  r(3,2,s):= 250
forall(s in 3..4)  r(3,2,s):= 200
r(1,3,1):= 230
r(1,3,2):= 210
r(1,3,3):= 190
r(1,3,4):= 180
r(2,3,1):= 210
r(2,3,2):= 200
r(2,3,3):= 180
r(2,3,4):= 170
r(3,3,1):= 250
r(3,3,2):= 240
r(3,3,3):= 220
r(3,3,4):= 200
r(1,4,1):= 230
r(1,4,2):= 210
r(1,4,3):= 190
r(1,4,4):= 180
r(1,5,1):= 230
r(1,5,2):= 110
r(1,5,3):= 190
r(1,5,4):= 180
r(2,4,1):= 210
r(2,4,2):= 200
r(2,4,3):= 180
r(2,4,4):= 170
r(2,5,1):= 210
r(2,5,2):= 200
r(2,5,3):= 180
r(2,5,4):= 170
```

```
  r(3,4,1):= 250
  r(3,4,2):= 240
  r(3,4,3):= 220
  r(3,4,4):= 200
  r(3,5,1):= 250
  r(3,5,2):= 240
  r(3,5,3):= 220
  r(3,5,4):= 200
  forall(i in 1..2)  c(i):= 2000
  c(3) := 3000
  forall(k in FACILITIES, t in 1..3)  co(k,t):= 100
  forall(k in FACILITIES, t in 4..5)  co(k,t):= 110
  forall(i in 1..2, k in FACILITIES)  su(i,k):= 5
  forall(k in FACILITIES)  su(3,k):= 8
  forall(k in FACILITIES)  b(1,k):= 0.25
  forall(k in FACILITIES)  b(2,k):= 0.28
  forall(k in FACILITIES)  b(3,k):= 0.33
  forall(i in 1..2)  h(i):= 15
  h(3):= 20
  inv0(1):= 300
  inv0(2):= 380
  inv0(3):= 450
  forall(t in PERIODS)  CAP(1,t):= 140
  forall(t in PERIODS)  CAP(2,t):= 280
  forall(i in ITEMS, t in PERIODS)  q(i,t):= 3000
  forall(i in 1..2) f(i):= 0.95
  f(3):= 0.98
  forall(i in ITEMS) L(i):= 1
  a(1,3) := 1
  a(2,3) := 2


  forall(s in SCENARIOS) p(s):=1/4


  ! Objective function
  Total_Expected_Cost:=
     sum(s in SCENARIOS) ( p(s) * (sum(i in ITEMS) sum(t in PERIODS)( c(i)* d(i,t)
+ h(i) * y(i,t,s) ))) + sum(k in FACILITIES) sum(t in PERIODS) (co(k,t)* o(k,t))

  ! demand constraint for every item i, period t under every scenario s
  forall(i in ITEMS,t in PERIODS, s in SCENARIOS)
     demand_constraints1(i,t,s) :=   f(i)* if((t-1)<1,0,x(i,t-1)) + if (t=1,inv0(i),y(i,t-1,s))
- y(i,t,s) - sum(j in ITEMS2) a(i,j)* v(j,t) - r(i,t,s)  >= 0

  ! capacity constraint for every period and facility
  forall(k in FACILITIES, t in PERIODS)
```

```
    capacity_constraints(k,t) := sum(i in ITEMS) (su(i,k)*d(i,t) + b(i,k)*x(i,t)) - o(k,t)
<= CAP(k,t)

  ! production bound for every item i and period t
  forall(i in ITEMS, t in PERIODS)
    production_bound(i,t):= x(i,t) - q(i,t) * d(i,t) <= 0

  ! Equality constraint for new variables
  forall(t in PERIODS)do
    x(1,t) = v(1,t)
    x(2,t) = v(2,t)
    x(3,t) = v(3,t)
  end-do

  forall(i in ITEMS) do
    y(i,1,1) = y(i,1,2)
    y(i,1,2) = y(i,1,3)
    y(i,1,3) = y(i,1,4)
  end-do

  forall(i in ITEMS) do
    y(i,2,1) = y(i,2,2)
    y(i,2,3) = y(i,2,4)
  end-do


  ! model description complete

  setparam("XPRS_verbose",false)   ! Enable message printing in mmxosl

  ! solve the model
  minimize(Total_Expected_Cost)

  writeln(" INPUT ")
  writeln
  forall(s in SCENARIOS) do
    writeln("Scenario = ", s)
    forall(t in PERIODS) do
    writeln("Period = ", t)
        forall(i in ITEMS) writeln("Demand Item ", i, "=", r(i,t,s))
        end-do
        writeln
     end-do

  forall(i in ITEMS)writeln("Set-up Cost Item ", i, " = ", c(i))
  forall(i in ITEMS)writeln("Unit Holding Cost Item ", i, " = ", h(i))
  forall(i in ITEMS)writeln("Inventory 0 Item ", i, " = ", inv0(i))
```

```
  forall(i in ITEMS)writeln("Lead Time Item ", i, " = ", L(i))
  forall(i in ITEMS)writeln("Yield Item ", i, " = ", f(i))
  writeln

  forall(k in FACILITIES) do
    writeln("Facility = ", k)
    forall(i in ITEMS)writeln("Set-up Resource Utilization Item ", i, " = ", su(i,k))
    forall(i in ITEMS)writeln("Capacity Utilization Rate Item ", i, " = ", b(i,k))
    writeln
    end-do

  forall(t in PERIODS) do
    writeln("Period = ", t)
    forall(k in FACILITIES)writeln("Capacity at Facility ", k, " = ", CAP(k,t))
    forall(i in ITEMS)writeln("Upper Bound Item ", i, " = ", q(i,t))
    writeln
  end-do
  writeln
  writeln(" SOLUTION ")
  writeln("Total Expected Costs are:  ", getobjval)
  writeln
  forall(t in PERIODS) do
  writeln("Period = ", t)
  forall(i in ITEMS)writeln("Produced Item ", i, " = ", getsol(x(i,t)) )
  forall(k in FACILITIES)writeln("Overtime Capacity Facility ", k, " = ", getsol(o(k,t)))
  writeln
  end-do
  writeln
  forall(s in SCENARIOS) do
    writeln("Scenario = ", s)
    forall(t in PERIODS) do
        forall(i in ITEMS) write(  strfmt(getsol(y(i,t,s)),7,1) , "  ")
        writeln
    end-do
    writeln
  end-do

end-model
```

Código del Primer Ejemplo Numérico No Aleatorio con Recurso Completo

```
model ProFijo2
  uses "mmxprs"

  declarations
    ! indicies
    ITEMS       = 1..3
    ITEMS2      = 1..3
    PERIODS     = 1..5
    FACILITIES  = 1..2
    SCENARIOS   = 1..4

    ! parameters
    c     :  array(ITEMS) of real
    su    :  array(ITEMS, FACILITIES) of real
    b     :  array(ITEMS, FACILITIES) of real
    h     :  array(ITEMS) of real
    r     :  array(ITEMS, PERIODS, SCENARIOS) of real
    CAP   :  array(FACILITIES, PERIODS) of real
    q     :  array(ITEMS, PERIODS) of real
    p     :  array(SCENARIOS) of real
    co    :  array(FACILITIES, PERIODS) of real
    a     :  array(ITEMS, ITEMS2) of real

    inv0  :  array(ITEMS) of real
    L     :  array(ITEMS) of real
    f     :  array(ITEMS) of real
    RP, WS, EV, EEV : real

    ! decision variables
    x     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar
    v     :  array(ITEMS2, PERIODS, SCENARIOS) of mpvar
    d     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar
    o     :  array(FACILITIES, PERIODS, SCENARIOS) of mpvar
    y     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar


  end-declarations

  ! declare d(i,t) binary
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  d(i,t,s) is_binary
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  x(i,t,s) is_integer
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  y(i,t,s) is_integer
```

```
! generate random data from Unifrom distributions
forall(s in SCENARIOS) r(1,1,s):= 200
forall(s in SCENARIOS) r(2,1,s):= 180
forall(s in SCENARIOS) r(3,1,s):= 230
forall(s in 1..2)  r(1,2,s):= 220
forall(s in 3..4)  r(1,2,s):= 180
forall(s in 1..2)  r(2,2,s):= 200
forall(s in 3..4)  r(2,2,s):= 170
forall(s in 1..2)  r(3,2,s):= 250
forall(s in 3..4)  r(3,2,s):= 200
r(1,3,1):= 230
r(1,3,2):= 210
r(1,3,3):= 190
r(1,3,4):= 180
r(2,3,1):= 210
r(2,3,2):= 200
r(2,3,3):= 180
r(2,3,4):= 170
r(3,3,1):= 250
r(3,3,2):= 240
r(3,3,3):= 220
r(3,3,4):= 200
r(1,4,1):= 230
r(1,4,2):= 210
r(1,4,3):= 190
r(1,4,4):= 180
r(1,5,1):= 230
r(1,5,2):= 110
r(1,5,3):= 190
r(1,5,4):= 180
r(2,4,1):= 210
r(2,4,2):= 200
r(2,4,3):= 180
r(2,4,4):= 170
r(2,5,1):= 210
r(2,5,2):= 200
r(2,5,3):= 180
r(2,5,4):= 170
r(3,4,1):= 250
r(3,4,2):= 240
r(3,4,3):= 220
r(3,4,4):= 200
r(3,5,1):= 250
r(3,5,2):= 240
r(3,5,3):= 220
r(3,5,4):= 200
forall(i in 1..2)  c(i):= 2000
```

```
  c(3) := 3000
  forall(k in FACILITIES, t in 1..3)  co(k,t):= 100
  forall(k in FACILITIES, t in 4..5)  co(k,t):= 110
  forall(i in 1..2, k in FACILITIES)  su(i,k):= 5
  forall(k in FACILITIES)  su(3,k):= 8
  forall(k in FACILITIES)  b(1,k):= 0.25
  forall(k in FACILITIES)  b(2,k):= 0.28
  forall(k in FACILITIES)  b(3,k):= 0.33
  forall(i in 1..2)  h(i):= 15
  h(3):= 20
  inv0(1):= 300
  inv0(2):= 380
  inv0(3):= 450
  forall(t in PERIODS)  CAP(1,t):= 140
  forall(t in PERIODS)  CAP(2,t):= 280
  forall(i in ITEMS, t in PERIODS)  q(i,t):= 3000
  forall(i in 1..2) f(i):= 0.95
  f(3):= 0.98
  forall(i in ITEMS) L(i):= 1
  a(1,3) := 1
  a(2,3) := 2

 forall(s in SCENARIOS) p(s):=1/4

  ! Objective function
  Total_Expected_Cost:=
     sum(s in SCENARIOS) ( p(s) * (sum(i in ITEMS) sum(t in PERIODS)( c(i)*
d(i,t,s) + h(i) * y(i,t,s) ) + sum(k in FACILITIES) sum(t in PERIODS) (co(k,t)*
o(k,t,s))))

  ! Objective Function for individual scenario problems
  forall(s in SCENARIOS + {5})  Total_Cost(s):= sum(i in ITEMS) sum(t in
PERIODS)( c(i)* d(i,t,s) +
      + h(i) * y(i,t,s) ) + sum(k in FACILITIES) sum(t in PERIODS) (co(k,t)* o(k,t,s))

  ! demand constraint for every item i, period t under every scenario s
  forall(i in ITEMS,t in PERIODS, s in SCENARIOS + {5})
     demand_constraints1(i,t,s) :=   f(i)* if((t-1)<1,0,x(i,t-1,s)) + if (t=1,inv0(i),y(i,t-
1,s)) - y(i,t,s) - sum(j in ITEMS2) a(i,j)* v(j,t,s) - r(i,t,s)  >= 0

  ! capacity constraint for every period and facility
  forall(k in FACILITIES, t in PERIODS, s in SCENARIOS + {5})
     capacity_constraints(k,t) := sum(i in ITEMS) (su(i,k)*d(i,t,s) + b(i,k)*x(i,t,s)) -
o(k,t,s) <= CAP(k,t)

  ! production bound for every item i and period t
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS + {5})
```

```
   production_bound(i,t):= x(i,t,s) - q(i,t) * d(i,t,s) <= 0

! Equality constraint for new variables
forall(t in PERIODS, s in SCENARIOS + {5})do
   x(1,t,s) = v(1,t,s)
   x(2,t,s) = v(2,t,s)
   x(3,t,s) = v(3,t,s)
end-do

forall(i in ITEMS) do
   y(i,1,1) = y(i,1,2)
   y(i,1,2) = y(i,1,3)
   y(i,1,3) = y(i,1,4)
end-do

forall(i in ITEMS) do
   y(i,2,1) = y(i,2,2)
   y(i,2,3) = y(i,2,4)
end-do

forall(i in ITEMS) do
   x(i,1,1) = x(i,1,2)
   x(i,1,2) = x(i,1,3)
   x(i,1,3) = x(i,1,4)
end-do

forall(i in ITEMS) do
   x(i,2,1) = x(i,2,2)
   x(i,2,3) = x(i,2,4)
end-do

forall(k in FACILITIES) do
   o(k,1,1) = o(k,1,2)
   o(k,1,2) = o(k,1,3)
   o(k,1,3) = o(k,1,4)
end-do

forall(k in FACILITIES) do
   o(k,2,1) = o(k,2,2)
   o(k,2,3) = o(k,2,4)
end-do

forall(i in ITEMS) do
   d(i,1,1) = d(i,1,2)
   d(i,1,2) = d(i,1,3)
   d(i,1,3) = d(i,1,4)
end-do
```

```
  forall(i in ITEMS) do
    d(i,2,1) = d(i,2,2)
    d(i,2,3) = d(i,2,4)
  end-do

  ! model description complete
    setparam("XPRS_verbose",false)   ! Enable message printing in mmxosl

  ! solve the model
  minimize(Total_Expected_Cost)

  writeln(" SOLUTION ")
  writeln("Total Expected Costs are:  ", getobjval)
  writeln

  writeln("Inventory")
  forall(s in SCENARIOS) do
    writeln("Scenario = ", s)
    forall(t in PERIODS) do
        forall(i in ITEMS) write(  strfmt(getsol(y(i,t,s)),7,1) , "  ")
        writeln
    end-do
    writeln
  end-do

  writeln("Production")
  forall(s in SCENARIOS) do
    writeln("Scenario = ", s)
    forall(t in PERIODS) do
        forall(i in ITEMS) write(  strfmt(getsol(x(i,t,s)),7,1) , "  ")
        writeln
    end-do
    writeln
  end-do

  writeln("Overtime")
  forall(s in SCENARIOS) do
    writeln("Scenario = ", s)
    forall(t in PERIODS) do
        forall(k in FACILITIES) write(  strfmt(getsol(o(k,t,s)),7,1) , "  ")
        writeln
    end-do
    writeln
  end-do

end-model
```

Código del Segundo Ejemplo Numérico No Aleatorio con Recurso Simple

```
model ProFijo3
  uses "mmxprs"

  declarations
    ! indicies
    ITEMS       = 1..3
    ITEMS2      = 1..3
    PERIODS     = 1..5
    FACILITIES  = 1..2
    SCENARIOS   = 1..6

    ! parameters
    c     :  array(ITEMS) of real
    su    :  array(ITEMS, FACILITIES) of real
    b     :  array(ITEMS, FACILITIES) of real
    h     :  array(ITEMS) of real
    r     :  array(ITEMS, PERIODS, SCENARIOS) of real
    CAP   :  array(FACILITIES, PERIODS) of real
    q     :  array(ITEMS, PERIODS) of real
    p     :  array(SCENARIOS) of real
    co    :  array(FACILITIES, PERIODS) of real
    a     :  array(ITEMS, ITEMS2) of real

    inv0  :  array(ITEMS) of real
    L     :  array(ITEMS) of real
    f     :  array(ITEMS) of real


    ! decision variables
    x     :  array(ITEMS, PERIODS) of mpvar
    v     :  array(ITEMS2, PERIODS) of mpvar
    d     :  array(ITEMS, PERIODS) of mpvar
    o     :  array(FACILITIES, PERIODS) of mpvar
    y     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar


  end-declarations

  ! declare d(i,t) binary
  forall(i in ITEMS, t in PERIODS)  d(i,t) is_binary
  forall(i in ITEMS, t in PERIODS)  x(i,t) is_integer
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  y(i,t,s) is_integer
```

```
! generate data

forall(s in SCENARIOS) r(1,1,s):= 150
forall(s in SCENARIOS) r(2,1,s):= 130
forall(s in SCENARIOS) r(3,1,s):= 250
forall(s in 1..3)  r(1,2,s):= 190
forall(s in 4..6)  r(1,2,s):= 150
forall(s in 1..3)  r(2,2,s):= 170
forall(s in 4..6)  r(2,2,s):= 120
forall(s in 1..3)  r(3,2,s):= 270
forall(s in 4..6)  r(3,2,s):= 230
forall(s in 1..2)  r(1,3,s):= 210
forall(s in 4..5)  r(1,3,s):= 170
forall(s in 1..2)  r(2,3,s):= 180
forall(s in 4..5)  r(2,3,s):= 140
forall(s in 1..2)  r(3,3,s):= 280
forall(s in 4..5)  r(3,3,s):= 240
r(1,3,3):= 180
r(1,3,6):= 140
r(1,4,1):= 220
r(1,4,2):= 200
r(1,4,3):= 180
r(1,4,4):= 180
r(1,4,5):= 170
r(1,4,6):= 150
r(1,5,1):= 220
r(1,5,2):= 200
r(1,5,3):= 180
r(1,5,4):= 180
r(1,5,5):= 170
r(1,5,6):= 150
r(2,3,3):= 160
r(2,3,6):= 130
r(2,4,1):= 180
r(2,4,2):= 170
r(2,4,3):= 160
r(2,4,4):= 155
r(2,4,5):= 140
r(2,4,6):= 125
r(2,5,1):= 180
r(2,5,2):= 170
r(2,5,3):= 160
r(2,5,4):= 155
r(2,5,5):= 140
r(2,5,6):= 130
r(3,3,3):= 260
r(3,3,6):= 230
```

```
  r(3,4,1):= 300
  r(3,4,2):= 280
  r(3,4,3):= 260
  r(3,4,4):= 255
  r(3,4,5):= 250
  r(3,4,6):= 235
  r(3,5,1):= 290
  r(3,5,2):= 280
  r(3,5,3):= 260
  r(3,5,4):= 250
  r(3,5,5):= 250
  r(3,5,6):= 240
  forall(i in 1..2)  c(i):= 2000
  c(3) := 3000
  forall(k in FACILITIES, t in 1..3)  co(k,t):= 100
  forall(k in FACILITIES, t in 4..5)  co(k,t):= 110
  forall(i in 1..2, k in FACILITIES)  su(i,k):= 5
  forall(k in FACILITIES)  su(3,k):= 8
  forall(k in FACILITIES)  b(1,k):= 0.25
  forall(k in FACILITIES)  b(2,k):= 0.28
  forall(k in FACILITIES)  b(3,k):= 0.33
  forall(i in 1..2)  h(i):= 15
  h(3):= 20
  inv0(1):= 300
  inv0(2):= 380
  inv0(3):= 450
  forall(t in PERIODS)  CAP(1,t):= 140
  forall(t in PERIODS)  CAP(2,t):= 280
  forall(i in ITEMS, t in PERIODS)  q(i,t):= 3000
  forall(i in 1..2) f(i):= 0.95
  f(3):= 0.98
  forall(i in ITEMS) L(i):= 1
  a(1,3) := 1
  a(2,3) := 2


  forall(s in SCENARIOS) p(s):=1/6


  ! Objective function
  Total_Expected_Cost:=
    sum(s in SCENARIOS) ( p(s) * (sum(i in ITEMS) sum(t in PERIODS)( c(i)* d(i,t)
+ h(i) * y(i,t,s) ))) + sum(k in FACILITIES) sum(t in PERIODS) (co(k,t)* o(k,t))

  ! demand constraint for every item i, period t under every scenario s
  forall(i in ITEMS,t in PERIODS, s in SCENARIOS)
```

```
    demand_constraints1(i,t,s) :=   f(i)* if((t-1)<1,0,x(i,t-1)) + if (t=1,inv0(i),y(i,t-1,s))
- y(i,t,s) - sum(j in ITEMS2) a(i,j)* v(j,t) - r(i,t,s)  >= 0

  ! capacity constraint for every period and facility
  forall(k in FACILITIES, t in PERIODS)
    capacity_constraints(k,t) := sum(i in ITEMS) (su(i,k)*d(i,t) + b(i,k)*x(i,t)) - o(k,t)
<= CAP(k,t)

  ! production bound for every item i and period t
  forall(i in ITEMS, t in PERIODS)
    production_bound(i,t):= x(i,t) - q(i,t) * d(i,t) <= 0

  ! Equality constraint for new variables
  forall(t in PERIODS)do
    x(1,t) = v(1,t)
    x(2,t) = v(2,t)
    x(3,t) = v(3,t)
  end-do

  forall(i in ITEMS) do
    y(i,1,1) = y(i,1,2)
    y(i,1,2) = y(i,1,3)
    y(i,1,3) = y(i,1,4)
    y(i,1,4) = y(i,1,5)
    y(i,1,5) = y(i,1,6)
  end-do

  forall(i in ITEMS) do
    y(i,2,1) = y(i,2,2)
    y(i,2,2) = y(i,2,3)
    y(i,2,4) = y(i,2,5)
    y(i,2,5) = y(i,2,6)
  end-do

  forall(i in ITEMS) do
    y(i,3,1) = y(i,3,2)
    y(i,3,4) = y(i,3,5)
  end-do



  ! model description complete

  setparam("XPRS_verbose",false)   ! Enable message printing in mmxosl

  ! solve the model
  minimize(Total_Expected_Cost)
```

```
writeln(" INPUT ")
writeln
forall(s in SCENARIOS) do
  writeln("Scenario = ", s)
  forall(t in PERIODS) do
  writeln("Period = ", t)
      forall(i in ITEMS) writeln("Demand Item ", i, "=", r(i,t,s))
      end-do
      writeln
   end-do

forall(i in ITEMS)writeln("Set-up Cost Item ", i, " = ", c(i))
forall(i in ITEMS)writeln("Unit Holding Cost Item ", i, " = ", h(i))
forall(i in ITEMS)writeln("Inventory 0 Item ", i, " = ", inv0(i))
forall(i in ITEMS)writeln("Lead Time Item ", i, " = ", L(i))
forall(i in ITEMS)writeln("Yield Item ", i, " = ", f(i))
writeln

forall(k in FACILITIES) do
  writeln("Facility = ", k)
  forall(i in ITEMS)writeln("Set-up Resource Utilization Item ", i, " = ", su(i,k))
  forall(i in ITEMS)writeln("Capacity Utilization Rate Item ", i, " = ", b(i,k))
  writeln
  end-do

forall(t in PERIODS) do
  writeln("Period = ", t)
  forall(k in FACILITIES)writeln("Capacity at Facility ", k, " = ", CAP(k,t))
  forall(i in ITEMS)writeln("Upper Bound Item ", i, " = ", q(i,t))
  writeln
end-do
writeln
writeln(" SOLUTION ")
writeln("Total Expected Costs are:  ", getobjval)
writeln
forall(t in PERIODS) do
writeln("Period = ", t)
forall(i in ITEMS)writeln("Produced Item ", i, " = ", getsol(x(i,t)) )
forall(k in FACILITIES)writeln("Overtime Capacity Facility ", k, " = ", getsol(o(k,t)))
writeln
end-do
writeln
forall(s in SCENARIOS) do
  writeln("Scenario = ", s)
  forall(t in PERIODS) do
      forall(i in ITEMS) write(  strfmt(getsol(y(i,t,s)),7,1) , "  ")
```

```
            writeln
        end-do
        writeln
    end-do

end-model
```

Código del Segundo Ejemplo Numérico No Aleatorio con Recurso Completo

```
model ProFijo4
  uses "mmxprs"

  declarations
    ! indicies
    ITEMS       = 1..3
    ITEMS2      = 1..3
    PERIODS     = 1..5
    FACILITIES  = 1..2
    SCENARIOS   = 1..6

    ! parameters
    c     :  array(ITEMS) of real
    su    :  array(ITEMS, FACILITIES) of real
    b     :  array(ITEMS, FACILITIES) of real
    h     :  array(ITEMS) of real
    r     :  array(ITEMS, PERIODS, SCENARIOS) of real
    CAP   :  array(FACILITIES, PERIODS) of real
    q     :  array(ITEMS, PERIODS) of real
    p     :  array(SCENARIOS) of real
    co    :  array(FACILITIES, PERIODS) of real
    a     :  array(ITEMS, ITEMS2) of real

    inv0  :  array(ITEMS) of real
    L     :  array(ITEMS) of real
    f     :  array(ITEMS) of real


    ! decision variables
    x     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar
    v     :  array(ITEMS2, PERIODS, SCENARIOS) of mpvar
    d     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar
    o     :  array(FACILITIES, PERIODS, SCENARIOS) of mpvar
    y     :  array(ITEMS, PERIODS, SCENARIOS) of mpvar


  end-declarations

  ! declare d(i,t) binary
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  d(i,t,s) is_binary
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  x(i,t,s) is_integer
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)  y(i,t,s) is_integer
```

```
! generate random data from Unifrom distributions

forall(s in SCENARIOS) r(1,1,s):= 150
forall(s in SCENARIOS) r(2,1,s):= 130
forall(s in SCENARIOS) r(3,1,s):= 250
forall(s in 1..3)  r(1,2,s):= 190
forall(s in 4..6)  r(1,2,s):= 150
forall(s in 1..3)  r(2,2,s):= 170
forall(s in 4..6)  r(2,2,s):= 120
forall(s in 1..3)  r(3,2,s):= 270
forall(s in 4..6)  r(3,2,s):= 230
forall(s in 1..2)  r(1,3,s):= 210
forall(s in 4..5)  r(1,3,s):= 170
forall(s in 1..2)  r(2,3,s):= 180
forall(s in 4..5)  r(2,3,s):= 140
forall(s in 1..2)  r(3,3,s):= 280
forall(s in 4..5)  r(3,3,s):= 240
r(1,3,3):= 180
r(1,3,6):= 140
r(1,4,1):= 220
r(1,4,2):= 200
r(1,4,3):= 180
r(1,4,4):= 180
r(1,4,5):= 170
r(1,4,6):= 150
r(1,5,1):= 220
r(1,5,2):= 200
r(1,5,3):= 180
r(1,5,4):= 180
r(1,5,5):= 170
r(1,5,6):= 150
r(2,3,3):= 160
r(2,3,6):= 130
r(2,4,1):= 180
r(2,4,2):= 170
r(2,4,3):= 160
r(2,4,4):= 155
r(2,4,5):= 140
r(2,4,6):= 125
r(2,5,1):= 180
r(2,5,2):= 170
r(2,5,3):= 160
r(2,5,4):= 155
r(2,5,5):= 140
r(2,5,6):= 130
r(3,3,3):= 260
r(3,3,6):= 230
```

```
  r(3,4,1):= 300
  r(3,4,2):= 280
  r(3,4,3):= 260
  r(3,4,4):= 255
  r(3,4,5):= 250
  r(3,4,6):= 235
  r(3,5,1):= 290
  r(3,5,2):= 280
  r(3,5,3):= 260
  r(3,5,4):= 250
  r(3,5,5):= 250
  r(3,5,6):= 240
  forall(i in 1..2)  c(i):= 2000
  c(3) := 3000
  forall(k in FACILITIES, t in 1..3)  co(k,t):= 100
  forall(k in FACILITIES, t in 4..5)  co(k,t):= 110
  forall(i in 1..2, k in FACILITIES)  su(i,k):= 5
  forall(k in FACILITIES)  su(3,k):= 8
  forall(k in FACILITIES)  b(1,k):= 0.25
  forall(k in FACILITIES)  b(2,k):= 0.28
  forall(k in FACILITIES)  b(3,k):= 0.33
  forall(i in 1..2)  h(i):= 15
  h(3):= 20
  inv0(1):= 300
  inv0(2):= 380
  inv0(3):= 450
  forall(t in PERIODS)  CAP(1,t):= 140
  forall(t in PERIODS)  CAP(2,t):= 280
  forall(i in ITEMS, t in PERIODS)  q(i,t):= 3000
  forall(i in 1..2) f(i):= 0.95
  f(3):= 0.98
  forall(i in ITEMS) L(i):= 1
  a(1,3) := 1
  a(2,3) := 2


  forall(s in SCENARIOS) p(s):=1/6


  ! Objective function
  Total_Expected_Cost:=
     sum(s in SCENARIOS) ( p(s) * (sum(i in ITEMS) sum(t in PERIODS)( c(i)*
d(i,t,s) + h(i) * y(i,t,s) ) + sum(k in FACILITIES) sum(t in PERIODS) (co(k,t)*
o(k,t,s))))

  ! demand constraint for every item i, period t under every scenario s
  forall(i in ITEMS,t in PERIODS, s in SCENARIOS)
```

```
    demand_constraints1(i,t,s) :=   f(i)* if((t-1)<1,0,x(i,t-1,s)) + if (t=1,inv0(i),y(i,t-
1,s)) - y(i,t,s) - sum(j in ITEMS2) a(i,j)* v(j,t,s) - r(i,t,s)  >= 0

  ! capacity constraint for every period and facility
  forall(k in FACILITIES, t in PERIODS, s in SCENARIOS)
    capacity_constraints(k,t,s) := sum(i in ITEMS) (su(i,k)*d(i,t,s) + b(i,k)*x(i,t,s)) -
o(k,t,s) <= CAP(k,t)

  ! production bound for every item i and period t
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS)
    production_bound(i,t,s):= x(i,t,s) - q(i,t) * d(i,t,s) <= 0

  ! Equality constraint for new variables
  forall(t in PERIODS, s in SCENARIOS)do
    x(1,t,s) = v(1,t,s)
    x(2,t,s) = v(2,t,s)
    x(3,t,s) = v(3,t,s)
  end-do

  forall(i in ITEMS) do
    y(i,1,1) = y(i,1,2)
    y(i,1,2) = y(i,1,3)
    y(i,1,3) = y(i,1,4)
    y(i,1,4) = y(i,1,5)
    y(i,1,5) = y(i,1,6)
  end-do

  forall(i in ITEMS) do
    y(i,2,1) = y(i,2,2)
    y(i,2,2) = y(i,2,3)
    y(i,2,4) = y(i,2,5)
    y(i,2,5) = y(i,2,6)
  end-do

  forall(i in ITEMS) do
    y(i,3,1) = y(i,3,2)
    y(i,3,4) = y(i,3,5)
  end-do

  forall(i in ITEMS) do
    x(i,1,1) = x(i,1,2)
    x(i,1,2) = x(i,1,3)
    x(i,1,3) = x(i,1,4)
    x(i,1,4) = x(i,1,5)
    x(i,1,5) = x(i,1,6)
  end-do
```

```
forall(i in ITEMS) do
  x(i,2,1) = x(i,2,2)
  x(i,2,2) = x(i,2,3)
  x(i,2,4) = x(i,2,5)
  x(i,2,5) = x(i,2,6)
end-do

forall(i in ITEMS) do
  x(i,3,1) = x(i,3,2)
  x(i,3,4) = x(i,3,5)
end-do

forall(k in FACILITIES) do
  o(k,1,1) = o(k,1,2)
  o(k,1,2) = o(k,1,3)
  o(k,1,3) = o(k,1,4)
  o(k,1,4) = o(k,1,5)
  o(k,1,5) = o(k,1,6)
end-do

forall(k in FACILITIES) do
  o(k,2,1) = o(k,2,2)
  o(k,2,2) = o(k,2,3)
  o(k,2,4) = o(k,2,5)
  o(k,2,5) = o(k,2,6)
end-do

forall(k in FACILITIES) do
  o(k,3,1) = o(k,3,2)
  o(k,3,4) = o(k,3,5)
end-do

forall(i in ITEMS) do
  d(i,1,1) = d(i,1,2)
  d(i,1,2) = d(i,1,3)
  d(i,1,3) = d(i,1,4)
  d(i,1,4) = d(i,1,5)
  d(i,1,5) = d(i,1,6)
end-do

forall(i in ITEMS) do
  d(i,2,1) = d(i,2,2)
  d(i,2,2) = d(i,2,3)
  d(i,2,4) = d(i,2,5)
  d(i,2,5) = d(i,2,6)
end-do
```

```
  forall(i in ITEMS) do
    d(i,3,1) = d(i,3,2)
    d(i,3,4) = d(i,3,5)
  end-do

  ! model description complete

  setparam("XPRS_verbose",false)   ! Enable message printing in mmxosl

  ! solve the model
  minimize(Total_Expected_Cost)

  writeln(" SOLUTION ")
  writeln("Total Expected Costs are:  ", getobjval)
  writeln

  writeln("Inventory")
  forall(s in SCENARIOS) do
    writeln("Scenario = ", s)
    forall(t in PERIODS) do
        forall(i in ITEMS) write(  strfmt(getsol(y(i,t,s)),7,1) , "  ")
        writeln
    end-do
    writeln
  end-do

  writeln("Production")
  forall(s in SCENARIOS) do
    writeln("Scenario = ", s)
    forall(t in PERIODS) do
        forall(i in ITEMS) write(  strfmt(getsol(x(i,t,s)),7,1) , "  ")
        writeln
    end-do
    writeln
  end-do

  writeln("Overtime")
  forall(s in SCENARIOS) do
    writeln("Scenario = ", s)
    forall(t in PERIODS) do
        forall(k in FACILITIES) write(  strfmt(getsol(o(k,t,s)),7,1) , "  ")
        writeln
    end-do
    writeln
  end-do

end-model
```

ANEXO F. Soluciones dadas para los Ejemplos No Aleatorios por los Códigos Desarrollados

Solución del Primer Ejemplo Numérico No Aleatorio con Recurso Simple

INPUT
Scenario = 1
Period = 1
Demand Item 1=200
Demand Item 2=180
Demand Item 3=230
Period = 2
Demand Item 1=220
Demand Item 2=200
Demand Item 3=250
Period = 3
Demand Item 1=230
Demand Item 2=210
Demand Item 3=250
Period = 4
Demand Item 1=230
Demand Item 2=210
Demand Item 3=250
Period = 5
Demand Item 1=230
Demand Item 2=210
Demand Item 3=250

Scenario = 2

Period = 1

Demand Item 1=200

Demand Item 2=180

Demand Item 3=230

Period = 2

Demand Item 1=220

Demand Item 2=200

Demand Item 3=250

Period = 3

Demand Item 1=210

Demand Item 2=200

Demand Item 3=240

Period = 4

Demand Item 1=210

Demand Item 2=200

Demand Item 3=240

Period = 5

Demand Item 1=110

Demand Item 2=200

Demand Item 3=240

Scenario = 3

Period = 1

Demand Item 1=200

Demand Item 2=180

Demand Item 3=230

Period = 2

Demand Item 1=180

Demand Item 2=170

Demand Item 3=200

Period = 3
Demand Item 1=190
Demand Item 2=180
Demand Item 3=220
Period = 4
Demand Item 1=190
Demand Item 2=180
Demand Item 3=220
Period = 5
Demand Item 1=190
Demand Item 2=180
Demand Item 3=220

Scenario = 4
Period = 1
Demand Item 1=200
Demand Item 2=180
Demand Item 3=230
Period = 2
Demand Item 1=180
Demand Item 2=170
Demand Item 3=200
Period = 3
Demand Item 1=180
Demand Item 2=170
Demand Item 3=200
Period = 4
Demand Item 1=180
Demand Item 2=170
Demand Item 3=200
Period = 5

Demand Item 1=180

Demand Item 2=170

Demand Item 3=200

Set-up Cost Item 1 = 2000

Set-up Cost Item 2 = 2000

Set-up Cost Item 3 = 3000

Unit Holding Cost Item 1 = 15

Unit Holding Cost Item 2 = 15

Unit Holding Cost Item 3 = 20

Inventory 0 Item 1 = 300

Inventory 0 Item 2 = 380

Inventory 0 Item 3 = 450

Lead Time Item 1 = 1

Lead Time Item 2 = 1

Lead Time Item 3 = 1

Yield Item 1 = 0.95

Yield Item 2 = 0.95

Yield Item 3 = 0.98

Facility = 1

Set-up Resource Utilization Item 1 = 5

Set-up Resource Utilization Item 2 = 5

Set-up Resource Utilization Item 3 = 8

Capacity Utilization Rate Item 1 = 0.25

Capacity Utilization Rate Item 2 = 0.28

Capacity Utilization Rate Item 3 = 0.33

Facility = 2

Set-up Resource Utilization Item 1 = 5

Set-up Resource Utilization Item 2 = 5

Set-up Resource Utilization Item 3 = 8
Capacity Utilization Rate Item 1 = 0.25
Capacity Utilization Rate Item 2 = 0.28
Capacity Utilization Rate Item 3 = 0.33

Period = 1
Capacity at Facility 1 = 140
Capacity at Facility 2 = 280
Upper Bound Item 1 = 3000
Upper Bound Item 2 = 3000
Upper Bound Item 3 = 3000

Period = 2
Capacity at Facility 1 = 140
Capacity at Facility 2 = 280
Upper Bound Item 1 = 3000
Upper Bound Item 2 = 3000
Upper Bound Item 3 = 3000

Period = 3
Capacity at Facility 1 = 140
Capacity at Facility 2 = 280
Upper Bound Item 1 = 3000
Upper Bound Item 2 = 3000
Upper Bound Item 3 = 3000

Period = 4
Capacity at Facility 1 = 140
Capacity at Facility 2 = 280
Upper Bound Item 1 = 3000
Upper Bound Item 2 = 3000

Upper Bound Item 3 = 3000

Period = 5
Capacity at Facility 1 = 140
Capacity at Facility 2 = 280
Upper Bound Item 1 = 3000
Upper Bound Item 2 = 3000
Upper Bound Item 3 = 3000

SOLUTION
Total Expected Costs are: 154251

Period = 1
Produced Item 1 = 698
Produced Item 2 = 1140
Produced Item 3 = 100
Overtime Capacity Facility 1 = 404.7
Overtime Capacity Facility 2 = 264.7

Period = 2
Produced Item 1 = 240
Produced Item 2 = 220
Produced Item 3 = 441
Overtime Capacity Facility 1 = 145.13
Overtime Capacity Facility 2 = 5.13

Period = 3
Produced Item 1 = 512
Produced Item 2 = 760
Produced Item 3 = 0

Overtime Capacity Facility 1 = 210.8
Overtime Capacity Facility 2 = 70.8

Period = 4
Produced Item 1 = 243
Produced Item 2 = 222
Produced Item 3 = 256
Overtime Capacity Facility 1 = 85.39
Overtime Capacity Facility 2 = 0

Period = 5
Produced Item 1 = 0
Produced Item 2 = 0
Produced Item 3 = 0
Overtime Capacity Facility 1 = 0
Overtime Capacity Facility 2 = 0

Scenario = 1

0.0	0.0	220.0
2.0	1.0	68.0
0.0	0.0	250.0
0.0	0.0	0.0
0.0	0.0	0.0

Scenario = 2

0.0	0.0	220.0
2.0	1.0	68.0
0.0	0.0	240.0
0.0	0.0	0.0

0.0 0.0 0.0

Scenario = 3

0.0 0.0 220.0
0.0 0.0 8.0
0.0 0.0 220.0
0.0 0.0 0.0
0.0 0.0 0.0

Scenario = 4

0.0 0.0 220.0
0.0 0.0 8.0
0.0 0.0 200.0
0.0 0.0 0.0
0.0 0.0 0.0

Solución del Segundo Ejemplo Numérico No Aleatorio con Recurso Simple

INPUT

Scenario = 1
Period = 1
Demand Item 1=150
Demand Item 2=130
Demand Item 3=250
Period = 2
Demand Item 1=190
Demand Item 2=170
Demand Item 3=270
Period = 3
Demand Item 1=210
Demand Item 2=180
Demand Item 3=280
Period = 4
Demand Item 1=220
Demand Item 2=180
Demand Item 3=300
Period = 5
Demand Item 1=220
Demand Item 2=180
Demand Item 3=290

Scenario = 2
Period = 1
Demand Item 1=150
Demand Item 2=130
Demand Item 3=250

Period = 2

Demand Item 1=190

Demand Item 2=170

Demand Item 3=270

Period = 3

Demand Item 1=210

Demand Item 2=180

Demand Item 3=280

Period = 4

Demand Item 1=200

Demand Item 2=170

Demand Item 3=280

Period = 5

Demand Item 1=200

Demand Item 2=170

Demand Item 3=280

Scenario = 3

Period = 1

Demand Item 1=150

Demand Item 2=130

Demand Item 3=250

Period = 2

Demand Item 1=190

Demand Item 2=170

Demand Item 3=270

Period = 3

Demand Item 1=180

Demand Item 2=160

Demand Item 3=260

Period = 4

Demand Item 1=180

Demand Item 2=160

Demand Item 3=260

Period = 5

Demand Item 1=180

Demand Item 2=160

Demand Item 3=260

Scenario = 4

Period = 1

Demand Item 1=150

Demand Item 2=130

Demand Item 3=250

Period = 2

Demand Item 1=150

Demand Item 2=120

Demand Item 3=230

Period = 3

Demand Item 1=170

Demand Item 2=140

Demand Item 3=240

Period = 4

Demand Item 1=180

Demand Item 2=155

Demand Item 3=255

Period = 5

Demand Item 1=180

Demand Item 2=155

Demand Item 3=250

Scenario = 5
Period = 1
Demand Item 1=150
Demand Item 2=130
Demand Item 3=250
Period = 2
Demand Item 1=150
Demand Item 2=120
Demand Item 3=230
Period = 3
Demand Item 1=170
Demand Item 2=140
Demand Item 3=240
Period = 4
Demand Item 1=170
Demand Item 2=140
Demand Item 3=250
Period = 5
Demand Item 1=170
Demand Item 2=140
Demand Item 3=250

Scenario = 6
Period = 1
Demand Item 1=150
Demand Item 2=130
Demand Item 3=250
Period = 2
Demand Item 1=150
Demand Item 2=120
Demand Item 3=230

Period = 3

Demand Item 1=140

Demand Item 2=130

Demand Item 3=230

Period = 4

Demand Item 1=150

Demand Item 2=125

Demand Item 3=235

Period = 5

Demand Item 1=150

Demand Item 2=130

Demand Item 3=240

Set-up Cost Item 1 = 2000

Set-up Cost Item 2 = 2000

Set-up Cost Item 3 = 3000

Unit Holding Cost Item 1 = 15

Unit Holding Cost Item 2 = 15

Unit Holding Cost Item 3 = 20

Inventory 0 Item 1 = 300

Inventory 0 Item 2 = 380

Inventory 0 Item 3 = 450

Lead Time Item 1 = 1

Lead Time Item 2 = 1

Lead Time Item 3 = 1

Yield Item 1 = 0.95

Yield Item 2 = 0.95

Yield Item 3 = 0.98

Facility = 1

Set-up Resource Utilization Item 1 = 5

Set-up Resource Utilization Item 2 = 5
Set-up Resource Utilization Item 3 = 8
Capacity Utilization Rate Item 1 = 0.25
Capacity Utilization Rate Item 2 = 0.28
Capacity Utilization Rate Item 3 = 0.33

Facility = 2
Set-up Resource Utilization Item 1 = 5
Set-up Resource Utilization Item 2 = 5
Set-up Resource Utilization Item 3 = 8
Capacity Utilization Rate Item 1 = 0.25
Capacity Utilization Rate Item 2 = 0.28
Capacity Utilization Rate Item 3 = 0.33

Period = 1
Capacity at Facility 1 = 140
Capacity at Facility 2 = 280
Upper Bound Item 1 = 3000
Upper Bound Item 2 = 3000
Upper Bound Item 3 = 3000

Period = 2
Capacity at Facility 1 = 140
Capacity at Facility 2 = 280
Upper Bound Item 1 = 3000
Upper Bound Item 2 = 3000
Upper Bound Item 3 = 3000

Period = 3
Capacity at Facility 1 = 140
Capacity at Facility 2 = 280

Upper Bound Item 1 = 3000
Upper Bound Item 2 = 3000
Upper Bound Item 3 = 3000

Period = 4
Capacity at Facility 1 = 140
Capacity at Facility 2 = 280
Upper Bound Item 1 = 3000
Upper Bound Item 2 = 3000
Upper Bound Item 3 = 3000

Period = 5
Capacity at Facility 1 = 140
Capacity at Facility 2 = 280
Upper Bound Item 1 = 3000
Upper Bound Item 2 = 3000
Upper Bound Item 3 = 3000

SOLUTION
Total Expected Costs are: 176207

Period = 1
Produced Item 1 = 434
Produced Item 2 = 699
Produced Item 3 = 125
Overtime Capacity Facility 1 = 223.47
Overtime Capacity Facility 2 = 83.47

Period = 2
Produced Item 1 = 534

Produced Item 2 = 804
Produced Item 3 = 247
Overtime Capacity Facility 1 = 318.13
Overtime Capacity Facility 2 = 178.13

Period = 3
Produced Item 1 = 538
Produced Item 2 = 814
Produced Item 3 = 292
Overtime Capacity Facility 1 = 336.726
Overtime Capacity Facility 2 = 196.726

Period = 4
Produced Item 1 = 232
Produced Item 2 = 189
Produced Item 3 = 296
Overtime Capacity Facility 1 = 86.6
Overtime Capacity Facility 2 = 0

Period = 5
Produced Item 1 = 0
Produced Item 2 = 0
Produced Item 3 = 0
Overtime Capacity Facility 1 = 0
Overtime Capacity Facility 2 = 0

Scenario = 1

25.0	0.0	200.0
0.0	0.0	52.0
5.0	0.0	14.0

0.0 1.0 0.0
0.0 0.0 0.0

Scenario = 2
25.0 0.0 200.0
0.0 0.0 52.0
5.0 0.0 14.0
0.0 0.0 0.0
0.0 0.0 0.0

Scenario = 3
25.0 0.0 200.0
0.0 0.0 52.0
0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0

Scenario = 4
25.0 0.0 200.0
0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0

Scenario = 5
25.0 0.0 200.0
0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0

Scenario = 6

25.0	0.0	200.0
0.0	0.0	0.0
0.0	0.0	0.0
0.0	0.0	0.0
0.0	0.0	0.0

ANEXO G. Códigos Desarrollados utilizando Programación Lineal

Código del Primer Ejemplo Numérico No Aleatorio Utilizando Programación Lineal

```
model ProLin1
  uses "mmxprs"

  declarations
    ! indicies
    ITEMS       = 1..3
    ITEMS2      = 1..3
    PERIODS     = 1..5
    FACILITIES  = 1..2
    SCENARIOS   = 1..4

    ! parameters
    c     :  array(ITEMS) of real
    su    :  array(ITEMS, FACILITIES) of real
    b     :  array(ITEMS, FACILITIES) of real
    h     :  array(ITEMS) of real
    r     :  array(ITEMS, PERIODS, SCENARIOS + {5}) of real
    CAP   :  array(FACILITIES, PERIODS) of real
    q     :  array(ITEMS, PERIODS) of real
    p     :  array(SCENARIOS) of real
    co    :  array(FACILITIES, PERIODS) of real
    a     :  array(ITEMS, ITEMS2) of real

    inv0  :  array(ITEMS) of real
    L     :  array(ITEMS) of real
    f     :  array(ITEMS) of real

    ! decision variables
    x     :  array(ITEMS, PERIODS) of mpvar
    v     :  array(ITEMS2, PERIODS) of mpvar
    d     :  array(ITEMS, PERIODS) of mpvar
    o     :  array(FACILITIES, PERIODS) of mpvar
    y     :  array(ITEMS, PERIODS, SCENARIOS + {5}) of mpvar


  end-declarations
```

```
! declare d(i,t) binary
forall(i in ITEMS, t in PERIODS)  d(i,t) is_binary
forall(i in ITEMS, t in PERIODS)  x(i,t) is_integer
forall(i in ITEMS, t in PERIODS, s in SCENARIOS + {5})  y(i,t,s) is_integer


! generate random data from Unifrom distributions
forall(s in SCENARIOS) r(1,1,s):= 200
forall(s in SCENARIOS) r(2,1,s):= 180
forall(s in SCENARIOS) r(3,1,s):= 230
forall(s in 1..2)  r(1,2,s):= 220
forall(s in 3..4)  r(1,2,s):= 180
forall(s in 1..2)  r(2,2,s):= 200
forall(s in 3..4)  r(2,2,s):= 170
forall(s in 1..2)  r(3,2,s):= 250
forall(s in 3..4)  r(3,2,s):= 200
r(1,3,1):= 230
r(1,3,2):= 210
r(1,3,3):= 190
r(1,3,4):= 180
r(2,3,1):= 210
r(2,3,2):= 200
r(2,3,3):= 180
r(2,3,4):= 170
r(3,3,1):= 250
r(3,3,2):= 240
r(3,3,3):= 220
r(3,3,4):= 200
r(1,4,1):= 230
r(1,4,2):= 210
r(1,4,3):= 190
r(1,4,4):= 180
r(1,5,1):= 230
r(1,5,2):= 110
r(1,5,3):= 190
r(1,5,4):= 180
r(2,4,1):= 210
r(2,4,2):= 200
r(2,4,3):= 180
r(2,4,4):= 170
r(2,5,1):= 210
r(2,5,2):= 200
r(2,5,3):= 180
r(2,5,4):= 170
r(3,4,1):= 250
r(3,4,2):= 240
```

```
  r(3,4,3):= 220
  r(3,4,4):= 200
  r(3,5,1):= 250
  r(3,5,2):= 240
  r(3,5,3):= 220
  r(3,5,4):= 200
  forall(i in 1..2)  c(i):= 2000
  c(3) := 3000
  forall(k in FACILITIES, t in 1..3)  co(k,t):= 100
  forall(k in FACILITIES, t in 4..5)  co(k,t):= 110
  forall(i in 1..2, k in FACILITIES)  su(i,k):= 5
  forall(k in FACILITIES)  su(3,k):= 8
  forall(k in FACILITIES)  b(1,k):= 0.25
  forall(k in FACILITIES)  b(2,k):= 0.28
  forall(k in FACILITIES)  b(3,k):= 0.33
  forall(i in 1..2)  h(i):= 15
  h(3):= 20
  inv0(1):= 300
  inv0(2):= 380
  inv0(3):= 450
  forall(t in PERIODS)  CAP(1,t):= 140
  forall(t in PERIODS)  CAP(2,t):= 280
  forall(i in ITEMS, t in PERIODS)  q(i,t):= 3000
  forall(i in 1..2) f(i):= 0.95
  f(3):= 0.98
  forall(i in ITEMS) L(i):= 1
  a(1,3) := 1
  a(2,3) := 2


  forall(s in SCENARIOS) p(s):=1/4

  ! Calculate expected demand from the set of scenarios

  forall(i in ITEMS,t in PERIODS) do
                r(i,t,5) := 0
                forall(s in SCENARIOS)  r(i,t,5) := r(i,t,5) + p(s)*r(i,t,s)
  end-do

  ! Objective function
  Total_Expected_Cost:=
    sum(s in SCENARIOS) ( p(s) * (sum(i in ITEMS) sum(t in PERIODS)( c(i)* d(i,t)
+ h(i) * y(i,t,s) ))) + sum(k in FACILITIES) sum(t in PERIODS) (co(k,t)* o(k,t))

  ! Objective Function for individual scenario problems
```

```
  forall(s in SCENARIOS + {5})  Total_Cost(s):= sum(i in ITEMS) sum(t in
PERIODS)( c(i)* d(i,t) + h(i) * y(i,t,s) ) + sum(k in FACILITIES) sum(t in PERIODS)
(co(k,t)* o(k,t))

  ! demand constraint for every item i, period t under every scenario s
  forall(i in ITEMS,t in PERIODS, s in SCENARIOS + {5})
    demand_constraints1(i,t,s) :=   f(i)* if((t-1)<1,0,x(i,t-1)) + if (t=1,inv0(i),y(i,t-1,s))
- y(i,t,s) - sum(j in ITEMS2) a(i,j)* v(j,t) - r(i,t,s)  >= 0

  ! capacity constraint for every period and facility
  forall(k in FACILITIES, t in PERIODS)
    capacity_constraints(k,t) := sum(i in ITEMS) (su(i,k)*d(i,t) + b(i,k)*x(i,t)) - o(k,t)
<= CAP(k,t)

  ! production bound for every item i and period t
  forall(i in ITEMS, t in PERIODS)
    production_bound(i,t):= x(i,t) - q(i,t) * d(i,t) <= 0

  ! Equality constraint for new variables
  forall(t in PERIODS)do
    x(1,t) = v(1,t)
    x(2,t) = v(2,t)
    x(3,t) = v(3,t)
  end-do

  forall(i in ITEMS) do
    y(i,1,1) = y(i,1,2)
    y(i,1,2) = y(i,1,3)
    y(i,1,3) = y(i,1,4)
  end-do

  forall(i in ITEMS) do
    y(i,2,1) = y(i,2,2)
    y(i,2,3) = y(i,2,4)
  end-do


  ! model description complete

  setparam("XPRS_verbose",false)   ! Enable message printing in mmxosl

  ! solve the model
  minimize(Total_Cost(5))

  writeln
  writeln(" SOLUTION ")
  writeln("Total Expected Costs are:  ", getobjval)
```

```
  writeln
  forall(t in PERIODS) do
  writeln("Period = ", t)
  forall(i in ITEMS)writeln("Produced Item ", i, " = ", getsol(x(i,t)) )
  forall(k in FACILITIES)writeln("Overtime Capacity Facility ", k, " = ", getsol(o(k,t)))
  writeln
  end-do

  writeln
  forall(s = 5) do
    writeln("Linear Programing ")
    forall(t in PERIODS) do
        forall(i in ITEMS) write(  strfmt(getsol(y(i,t,s)),7,1) , "  ")
        writeln
    end-do
    writeln
  end-do

end-model
```

Código del Segundo Ejemplo Numérico No Aleatorio Utilizando Programación Lineal

```
model ProLin2
  uses "mmxprs"

  declarations
    ! indicies
    ITEMS       = 1..3
    ITEMS2      = 1..3
    PERIODS     = 1..5
    FACILITIES  = 1..2
    SCENARIOS   = 1..6

    ! parameters
    c     :  array(ITEMS) of real
    su    :  array(ITEMS, FACILITIES) of real
    b     :  array(ITEMS, FACILITIES) of real
    h     :  array(ITEMS) of real
    r     :  array(ITEMS, PERIODS, SCENARIOS + {7}) of real
    CAP   :  array(FACILITIES, PERIODS) of real
    q     :  array(ITEMS, PERIODS) of real
    p     :  array(SCENARIOS) of real
    co    :  array(FACILITIES, PERIODS) of real
    a     :  array(ITEMS, ITEMS2) of real

    inv0  :  array(ITEMS) of real
    L     :  array(ITEMS) of real
    f     :  array(ITEMS) of real

    ! decision variables
    x     :  array(ITEMS, PERIODS) of mpvar
    v     :  array(ITEMS2, PERIODS) of mpvar
    d     :  array(ITEMS, PERIODS) of mpvar
    o     :  array(FACILITIES, PERIODS) of mpvar
    y     :  array(ITEMS, PERIODS, SCENARIOS + {7}) of mpvar

  end-declarations

  ! declare d(i,t) binary
  forall(i in ITEMS, t in PERIODS)  d(i,t) is_binary
  forall(i in ITEMS, t in PERIODS)  x(i,t) is_integer
  forall(i in ITEMS, t in PERIODS, s in SCENARIOS + {7})  y(i,t,s) is_integer
```

```
! generate data

forall(s in SCENARIOS) r(1,1,s):= 150
forall(s in SCENARIOS) r(2,1,s):= 130
forall(s in SCENARIOS) r(3,1,s):= 250
forall(s in 1..3)  r(1,2,s):= 190
forall(s in 4..6)  r(1,2,s):= 150
forall(s in 1..3)  r(2,2,s):= 170
forall(s in 4..6)  r(2,2,s):= 120
forall(s in 1..3)  r(3,2,s):= 270
forall(s in 4..6)  r(3,2,s):= 230
forall(s in 1..2)  r(1,3,s):= 210
forall(s in 4..5)  r(1,3,s):= 170
forall(s in 1..2)  r(2,3,s):= 180
forall(s in 4..5)  r(2,3,s):= 140
forall(s in 1..2)  r(3,3,s):= 280
forall(s in 4..5)  r(3,3,s):= 240
r(1,3,3):= 180
r(1,3,6):= 140
r(1,4,1):= 220
r(1,4,2):= 200
r(1,4,3):= 180
r(1,4,4):= 180
r(1,4,5):= 170
r(1,4,6):= 150
r(1,5,1):= 220
r(1,5,2):= 200
r(1,5,3):= 180
r(1,5,4):= 180
r(1,5,5):= 170
r(1,5,6):= 150
r(2,3,3):= 160
r(2,3,6):= 130
r(2,4,1):= 180
r(2,4,2):= 170
r(2,4,3):= 160
r(2,4,4):= 155
r(2,4,5):= 140
r(2,4,6):= 125
r(2,5,1):= 180
r(2,5,2):= 170
r(2,5,3):= 160
r(2,5,4):= 155
r(2,5,5):= 140
r(2,5,6):= 130
r(3,3,3):= 260
r(3,3,6):= 230
```

```
r(3,4,1):= 300
r(3,4,2):= 280
r(3,4,3):= 260
r(3,4,4):= 255
r(3,4,5):= 250
r(3,4,6):= 235
r(3,5,1):= 290
r(3,5,2):= 280
r(3,5,3):= 260
r(3,5,4):= 250
r(3,5,5):= 250
r(3,5,6):= 240
forall(i in 1..2)  c(i):= 2000
c(3) := 3000
forall(k in FACILITIES, t in 1..3)  co(k,t):= 100
forall(k in FACILITIES, t in 4..5)  co(k,t):= 110
forall(i in 1..2, k in FACILITIES)  su(i,k):= 5
forall(k in FACILITIES)  su(3,k):= 8
forall(k in FACILITIES)  b(1,k):= 0.25
forall(k in FACILITIES)  b(2,k):= 0.28
forall(k in FACILITIES)  b(3,k):= 0.33
forall(i in 1..2)  h(i):= 15
h(3):= 20
inv0(1):= 300
inv0(2):= 380
inv0(3):= 450
forall(t in PERIODS)  CAP(1,t):= 140
forall(t in PERIODS)  CAP(2,t):= 280
forall(i in ITEMS, t in PERIODS)  q(i,t):= 3000
forall(i in 1..2) f(i):= 0.95
f(3):= 0.98
forall(i in ITEMS) L(i):= 1
a(1,3) := 1
a(2,3) := 2


forall(s in SCENARIOS) p(s):=1/6

! Calculate expected demand from the set of scenarios

forall(i in ITEMS,t in PERIODS) do
                r(i,t,7) := 0
                forall(s in SCENARIOS)  r(i,t,7) := r(i,t,7) + p(s)*r(i,t,s)
end-do

! Objective function
Total_Expected_Cost:=
```

```
    sum(s in SCENARIOS) ( p(s) * (sum(i in ITEMS) sum(t in PERIODS)( c(i)* d(i,t)
+ h(i) * y(i,t,s) ))) + sum(k in FACILITIES) sum(t in PERIODS) (co(k,t)* o(k,t))

  ! Objective Function for individual scenario problems
  forall(s in SCENARIOS + {7})  Total_Cost(s):= sum(i in ITEMS) sum(t in
PERIODS)( c(i)* d(i,t) + h(i) * y(i,t,s) ) + sum(k in FACILITIES) sum(t in PERIODS)
(co(k,t)* o(k,t))


  ! demand constraint for every item i, period t under every scenario s
  forall(i in ITEMS,t in PERIODS, s in SCENARIOS + {7})
    demand_constraints1(i,t,s) :=   f(i)* if((t-1)<1,0,x(i,t-1)) + if (t=1,inv0(i),y(i,t-1,s))
- y(i,t,s) - sum(j in ITEMS2) a(i,j)* v(j,t) - r(i,t,s)  >= 0

  ! capacity constraint for every period and facility
  forall(k in FACILITIES, t in PERIODS)
    capacity_constraints(k,t) := sum(i in ITEMS) (su(i,k)*d(i,t) + b(i,k)*x(i,t)) - o(k,t)
<= CAP(k,t)

  ! production bound for every item i and period t
  forall(i in ITEMS, t in PERIODS)
    production_bound(i,t):= x(i,t) - q(i,t) * d(i,t) <= 0

  ! Equality constraint for new variables
  forall(t in PERIODS)do
    x(1,t) = v(1,t)
    x(2,t) = v(2,t)
    x(3,t) = v(3,t)
  end-do

  forall(i in ITEMS) do
    y(i,1,1) = y(i,1,2)
    y(i,1,2) = y(i,1,3)
    y(i,1,3) = y(i,1,4)
    y(i,1,4) = y(i,1,5)
    y(i,1,5) = y(i,1,6)
  end-do

  forall(i in ITEMS) do
    y(i,2,1) = y(i,2,2)
    y(i,2,2) = y(i,2,3)
    y(i,2,4) = y(i,2,5)
    y(i,2,5) = y(i,2,6)
  end-do

  forall(i in ITEMS) do
    y(i,3,1) = y(i,3,2)
```

```
    y(i,3,4) = y(i,3,5)
  end-do



  ! model description complete

  setparam("XPRS_verbose",false)   ! Enable message printing in mmxosl

  ! solve the model
  minimize(Total_Cost(7))

  writeln
  writeln(" SOLUTION ")
  writeln("Total Expected Costs are:  ", getobjval)
  writeln
  forall(t in PERIODS) do
  writeln("Period = ", t)
  forall(i in ITEMS)writeln("Produced Item ", i, " = ", getsol(x(i,t)) )
  forall(k in FACILITIES)writeln("Overtime Capacity Facility ", k, " = ", getsol(o(k,t)))
  writeln
  end-do

  writeln
  forall(s = 7) do
    writeln("Linear Programing ")
    forall(t in PERIODS) do
        forall(i in ITEMS) write(  strfmt(getsol(y(i,t,s)),7,1) , "  ")
        writeln
    end-do
    writeln
  end-do

end-model
```

ANEXO H. Soluciones Encontradas para los Modelos de Programación Lineal

Solución del Primer Ejemplo Numérico No Aleatorio Utilizando Programación Lineal

SOLUTION
Total Expected Costs are: 152157

Period = 1
Produced Item 1 = 720
Produced Item 2 = 1180
Produced Item 3 = 82
Overtime Capacity Facility 1 = 415.46
Overtime Capacity Facility 2 = 275.46

Period = 2
Produced Item 1 = 251
Produced Item 2 = 200
Produced Item 3 = 465
Overtime Capacity Facility 1 = 150.2
Overtime Capacity Facility 2 = 10.2

Period = 3
Produced Item 1 = 479
Produced Item 2 = 740
Produced Item 3 = 0
Overtime Capacity Facility 1 = 196.95
Overtime Capacity Facility 2 = 56.95

Period = 4

Produced Item 1 = 243

Produced Item 2 = 222

Produced Item 3 = 250

Overtime Capacity Facility 1 = 83.41

Overtime Capacity Facility 2 = 0

Period = 5

Produced Item 1 = 0

Produced Item 2 = 0

Produced Item 3 = 0

Overtime Capacity Facility 1 = 0

Overtime Capacity Facility 2 = 0

Linear Programming

0.0	0.0	145.0
0.0	0.0	0.0
0.0	0.0	228.0
0.0	0.0	0.0
0.0	0.0	0.0

Solución del Segundo Ejemplo Numérico No Aleatorio Utilizando Programación Lineal

SOLUTION
Total Expected Costs are: 173947

Period = 1
Produced Item 1 = 517
Produced Item 2 = 756
Produced Item 3 = 125
Overtime Capacity Facility 1 = 260.18
Overtime Capacity Facility 2 = 120.18

Period = 2
Produced Item 1 = 485
Produced Item 2 = 762
Produced Item 3 = 270
Overtime Capacity Facility 1 = 301.71
Overtime Capacity Facility 2 = 161.71

Period = 3
Produced Item 1 = 505
Produced Item 2 = 800
Produced Item 3 = 270
Overtime Capacity Facility 1 = 317.261
Overtime Capacity Facility 2 = 177.261

Period = 4
Produced Item 1 = 232
Produced Item 2 = 190
Produced Item 3 = 296

Overtime Capacity Facility 1 = 86.88

Overtime Capacity Facility 2 = 0

Period = 5

Produced Item 1 = 0

Produced Item 2 = 0

Produced Item 3 = 0

Overtime Capacity Facility 1 = 0

Overtime Capacity Facility 2 = 0

Linear Programing

0.0	0.0	128.0
0.0	0.0	0.0
0.0	0.0	0.0
0.0	0.0	0.0
0.0	0.0	0.0