**Pasos para formular un modelo de PL:**

1. Comprender la situación problemática.
2. Describir el objetivo en palabras. (¿Qué hacer? ¿Cuándo? ¿Para qué?)
3. Describir c/u de las restricciones en palabras.
4. Definir las variables de decisión (controlables).
5. Expresar el objetivo en función de las variables de decisión.
6. Expresar cada restricción en función de las variables de decisión.

**Hipótesis básicas:**

* No hay inflación, no varían los precios o costos.
* No hay fallas en la producción.
* No hay desperdicios de recursos en la producción.
* Todo lo que se produce se vende.
* No hay tiempos muertos.
* Se cuenta con capital suficiente …
* Todos los precios y costos se encuentran expresados en la misma moneda.

**Variables Slack**: Sirven para convertir las restricciones en igualdades. Si una variable Slack vale 0, significa que esa restricción es limitante.

**Problemas de centros de producción.**

En estos problemas la producción se divide en distintos lugares físicos y en c/u de los cuales se realizan distintas partes del proceso.

**Problemas de mezcla.**

Situaciones en los cuales ciertos insumos se deben mezclar en cierta proporción para producir bienes para la venta.

**Problemas de armado.**

Situaciones en las cuales se deben fabricar un producto utilizando determinada cantidad de otros productos (cantidad y no porcentajes).

**Programación de metas.**

¿Cómo hacemos para poner una restricción que nos indique cuanto estamos por encima de un valor y cuanto por debajo? Comparamos lo que “transportamos” con la meta y una resta de variables que lo indican:

TRANSPORTADO – META = EXCESO – DEFECTO

EXCESO, DEFECTO >= 0

Si EXCESO ≠ 0 🡪 DEFECTO = 0 (y viceversa).

**Problemas con varios periodos.**

Ventas(i) + Stock Final(i) = Producción(i) + Stock Final(i-1)

(si sobra plata la pongo en un banco con 0.5% de interés y si falta debo pedir prestado pagando 1% mensual)

Con interés vencido:

CAJA\_INICIAL + INGRESOS – EGRESOS – CAJA\_A\_ALCANZAR = EXCESO\_CAJA – DEFECTO\_CAJA

Z(MAX) = EXC – DEF + 0.005 EXC – 0.01 DEF

Adelantado:

CAJA\_INICIAL + INGRESOS – EGRESOS – CAJA\_A\_ALCANZAR + 0.005 EXC – 0.01 DEF

= EXCESO\_CAJA – DEFECTO\_CAJA

Z(MAX) = EXC – DEF

Si no se aclara, es preferible ir por el vencido.

**Programación lineal entera.**

**Variables:**

* Discretas: productos/recursos enteros.
* Bivalentes o binarias:
  + De decisión: señalan alternativas posibles.
  + Indicativas: marcan el estado de una variable asociada.

Cuando se agregan al modelo las condiciones de producción, las variables Yi pasan a indicar si el producto se fabrica o no y hay que vincularlas con las de producción.

Si no agregamos las restricciones que vinculan a a las binarias Yi con las variables de producción Xi da cualquier cosa. Porque las variables indicativas NECESITAN RESTRICCIONES PARA PODER TOMAR EL VALOR QUE INDICA SU DEFINICIÓN.

ATENCIÓN: NO se puede definir una bivalente que tome valor 1 para indicar que sucede algo y pretender que mágicamente tome ese valor. Hay que poner restricciones.

**Costo diferencial por intervalo.**

HORAS = H1 + H2 + H3

X1 YH1 <= H1 <= X2 YH1

(X2+m) YH2 <= H2 <= X3 YH2

(X3+m) YH3 <= H3 <= M YH3

YH1 + YH2 + YH3 = 1

**Imagen que contiene Escala de tiempo

Descripción generada automáticamente**

**Función cóncava seccionalmente lineal.**

X = XA + XB + XC

GANANCIA X = $A \* XA + $B \* XB + $C \* XC

UA \* YB <= XA <= UA

UB \* YC<= XB <= UB \* YB

XC <= M \* YC

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

**Texto

Descripción generada automáticamente**

**CASOS:**

1. **Si Y2 vale 0, entonces Y1 no puede valer 1:** Y1 <= Y2
2. **Y1 vale 1 si MES = 12, sino vale 0:** 12 \* Y1 <= MES <= 11 + Y1
3. **OR:**  Yor <= Sum\_i (1,n) Yi <= n \* Yor
4. **AND**: n \* Yand <= Sum\_i (1,n) Yi <= (n – 1) + Yand
5. **Y1 distinto de Y2:** Y1 + Y2 = 1
6. **E1 solo puede tomar valores: 1, 2, 3, 5, 6, 7.**

E1 = 1\*Y1+2\*Y2+3\*Y3+5\*Y5+6\*Y6+7\*Y7

Sumatoria(Yi) = 1

1. **C1 sea mayor a 10**: C1 >= 10 + m
2. **E1 tome únicamente valores impares:** E1 = 2\*E2 + 1
3. **C1 mayor o igual a 50 si Y1=1 o 75 si Y1=0:** C1 >= 50\*Y1 + 75(1-Y1)
4. **E1 sea mayor a 100 o sino menor que 80.**

(100 + m) \* Y <= E1 <= (80 – m) + M\*Y

O

100 + m – M \* (1-Y) <= E1 <= 80 – m + M \* Y

1. **Si C1 > 0 entonces C1>= 22**

Dibujo de una persona

Descripción generada automáticamente con confianza baja

1. **E1 tome el máximo valor entre E2, E3 Y E4**

Imagen que contiene Word

Descripción generada automáticamente

1. **C1 tome el segundo menor valor entre C2, C3, C4 y C5.**

Texto

Descripción generada automáticamente

ES: C5 – M(1-Y’5) <= C1 <= C5 + M (1-Y5)

1. **Si C2 = 0 entonces C1=0.**



1. **C1 distinto de 13.**



1. **E1 tome el valor de C1 redondeado.**

C1 – 0,5 + m <= E1 <= C1 + 0,5

1. **E1 tome un valor igual a la cantidad de variables (E2, E3, E4 y E5) cuyo valor es mayor que 5.**

(5 + m) \* Y1 <= E2 <= 5\*(1-Y1) + M\*Y1



**Problema del viajante.**

Un viajante tiene que partir de su casa y visitar una serie de clientes antes de retornar finalmente a su casa. No puede dejar de visitar ningún cliente. Se conocen las distancias entre cada par de clientes y entre cada cliente y la casa del viajante. (Xij: distancia de i a j)

Viajante simétrico: no importa la dirección. (Xij = Xji).

Viajante asimétrico: importa la dirección.

Salgo a un solo lugar)

Llegó desde un solo lugar)

No realizo subtours)

**Ui – Uj + n \* Xij <= n – 1**

Hay 2 medios de transporte y mínimo se usan Z veces)

Xij = YAij + YBij. Ɐ i,j. i≠j.

Sum\_i (Sum\_j (XAij)) >= Z

Sum\_i (Sum\_j (Xbij)) >= Z

Variables para orden)

* No se puede visitar al cliente de la ciudad D si antes no se visitó al de C:

UD >= UG

* No se puede visitar al cliente de la ciudad F si antes no se visito al de E o B.

UF >= UE – M \* Y

UF >= UB – M\* (1-Y)

**Problemas combinatorios**

Son aquellos en los cuales se desea determinar combinaciones optimas.

Problemas de Distribución o Transporte.

* Un conjunto de lugares donde c/u tiene disponible una cantidad de unidades de un producto (orígenes o suministros).
* Un conjunto de lugares donde c/u demanda una cantidad de unidades de un producto (destinos o demandas)
* Objetivo: determinar la cantidad de unidades de producto que c/origen envía a c/destino, para minimizar los costos de transporte totales en un cierto periodo de tiempo.
* Hipótesis principales: Producto homogéneo, costo lineal.

Xij: cantidad de unidades que el origen i envía al destino j.

Asumiendo que: Sum (Si) = Sum (Dj)

Z(Min) = Sum\_i Sum\_j (Cij Xij)

Sum(Xij) = Si (i = 1 a m)

Sum(Xij) = Dj (j = 1 a n).

Xij >= 0, Ɐi y Ɐj.

**MUY IMPORTANTE**: Existe un teorema que demuestra que, si todas las ofertas son números enteros y todas las demandas son números enteros, siendo todas las restricciones IGUALDADES, el problema de distribución o transporte tendrá como resultado que todas las variables tomaran valor entero, aunque no se les ponga la condición de que las variables tienen que tomar valor entero.

Por lo que el problema de distribución o transporte se resuelve como un problema con variables continuas. Es muy importante que la oferta total sea igual a la demanda total para que se pueda verificar que resolviéndolo como continua da resultado entero.

**Problema de transbordo.**

En este problema las unidades no son enviadas directamente desde los orígenes hacia los destinos, sino que van desde los orígenes hasta alguno de los centros de transbordo y desde este a alguno de los destinos.

* XOiTj: cantidad de unidades que el origen i envía al transbordo j.
* XTiDj: cantidad de unidades que el transbordo i envía al destino j.
* Se agrega una ecuación para c/transbordo que indica que todo lo que entro debe salir del mismo.

**Problema de asignación.**

Dados dos conjuntos A y B, ambos con n elementos. Encontrar el conjunto P donde c/elemento es un par (a,b) (a e A, b e B) tal que minimice una función de costo ∑ C(a,b).

* Restricciones: cada elemento de A y B deben aparecer en P exactamente una vez.
* Xij: 1 si i es asignado a j, 0 si no.
* Z = ∑ ∑ Cij Xij.
* ∑ Xij = 1 (Ɐi = 1 a m). ∑ Xij = 1 (Ɐj = 1 a m). Xij >= 0 Ɐi, Ɐj.
* Es un caso particular del problema de transporte, donde todas las ofertas y demandas son iguales a 1.

**Problemas de asignación cuadrática.**

Objetivo: colocar elementos en un lugar minimizando los costos 🡪 los costos son 2 y se multiplican.

* Yijkl = 1 si Xij = Xkl = 1, 0 si no.
* 2 Yijkl <= Xij + Xkl <= 1 + Yijkl
* Y en el funcional ponemos las Yijkl en lugar del producto Xij \* Xkl.

**Uncapacitated Facility Location (UFL).**

Se debe decidir donde abrir los depósitos y que proporción de la demanda de los clientes satisface cada deposito abierto.

* Xij: fracción de la demanda de la zona j que satisface el deposito ubicado en i.
* Yi: 1 si se establece el deposito i, 0 si no.
* Fi: costo anual fijo de establecer un deposito en el lugar i.
* Cij: costo de producción y distribución si el deposito que esta ubicado en i le proporciona al cliente j todo lo que este en demanda.

Minimizar: ∑ ∑ Cij Xij + ∑ fi Yi.

∑(i) Xij = 1, para todo j.

Xij <= Yi para todo i,j.

Xij >= 0, Yi e {0, 1}, para todo i,j.

**Problema de la mochila / Knapsack.**

¿Qué llevo y que dejo?

* Wi: peso del objeto i.
* Pi: aporte del objeto i a la mochila.
* C: capacidad de la mochila
* Xi: 1 si el objeto i esta en la mochila, 0 si no.
* Sum\_i(1 a n) Wi \* Xi <= C
* Z(MAX) = Sum\_i(1 a n) Pi \* Xi.

Múltiples mochilas)

* Xij: 1 si el objeto i esta en la mochila j.
* Sum\_j(1 a m) Xij <= 1
* Sum\_i(1 a n) Wi \* Xij <= Cj.
* Z(MAX) = Sum\_j Sum\_i Pi\* Xij.

Acotado (más de 1 objeto de cada tipo, bi indica cantidad de i))

* Xi (entera): cantidad de objeto de tipo i que se colocan en la mochila.
* Xi <= bi
* Sum\_i (1 a n) Wi \* Xi <= C
* Z(MAX) = Sum\_i Pi \* Xi.

**Problema de cobertura de conjuntos.**

* Problemas de grupos que se deben cubrir/particionar.
* Problemas de Packing.

Genéricamente:

* S = {1, 2, .., n} conjunto de elementos a cubrir.
* L = { (1,2), (2), … } conjunto formado por subconjuntos de S.
* Elegir elementos de L tales que:
  + Cobertura: se cubran todos los elementos de S con solapamiento
  + Partición: se cubran todos los elementos de S sin solapamiento.
  + Packing: se cubra la máxima cantidad de elementos de S que se pueda sin solapamiento.

Ejemplo:

Cubrir un vuelo a 5 ciudades (1, 2, 3, 4 y 5). Se definieron 6 posibles circuitos: A={1,2}, B={1,3,5}, C={2,4,5}, D={3}, E={1} y F={4,5}.

* Yi: 1 si se realiza el circuito i, 0 si no.

Cobertura) Determinar cuáles circuitos se realizaron, de modo tal que c/u de las 5 ciudades sea cubierta por al menos un circuito.

MIN: YA + YB + YC + YD +YE + YF

C1) YA + YB + Y3 >= 1

C2) YA + YC >= 1

C3) YB + YD >= 1

C4) YC + YF >= 1

C5) YB + YC + YF >= 1

Particionar) Se trata de cubrir todas las ciudades sin solapamiento (muchas veces no tiene solución)

MIN: YA + YB + YC + YD +YE + YF

C1) YA + YB + Y3 = 1

C2) YA + YC = 1

C3) YB + YD = 1

C4) YC + YF = 1

C5) YB + YC + YF = 1

Packing) Cubrir la mayor cantidad de elementos que se pueda sin solapamiento.

MAX: YA + YB + YC + YD +YE + YF

C1) YA + YB + Y3 <= 1

C2) YA + YC <= 1

C3) YB + YD <= 1

C4) YC + YF <= 1

C5) YB + YC + YF <= 1

Otro planteo) Queremos que visite la mayor cantidad de ciudades.

* Vi: 1 si se visito la ciudad i, 0 si no.

MAX: V1 + V2 + V3 + V4 +V5

C1) YA + YB + Y3 = V1

C2) YA + YC = V2

C3) YB + YD = V3

C4) YC + YF = V4

C5) YB + YC + YF = V5

Calendarización.

Existen N tareas a realizar, cada una con un tiempo de procesamiento. Cada una de las tareas se puede realizar en cualquiera de las M maquinas. No existe restricciones en lo que respecta a la procedencia de las tareas. El objetivo sería definir en que momento se debe procesar cada tarea y en que máquina, para poder completar todas las tareas lo antes posible.

* Iij: minuto en que empieza la tarea i en la maquina j.
* Fij: minuto en que finaliza la tarea i en la maquina j.
* FINAL: minuto en el cual finaliza la ultima tarea.

Como las tareas no se interrumpen: Fij = Iij + Tiempo de i en j.

Fi1 = Ii1 + T i en 1 | Fi2 = Ii2 +T i en 2 | Fi1 <= Ii2 (para todas las tareas i).

Fi2 <= FINAL (para todas las tareas i)

Z(MIN) = FINAL.

Para que no se hagan 2 tareas al mismo tiempo en la misma maquina:

Fi1 <= Ik1 + M Yanuloik

Fk1 <= Ii1 + M Yanuloki

Yanuloik + Yanuloki = 1

(para todo par de tareas ik)