
UNIVERSIDAD
DE LA SERENA
CHILE

# MODELO DE SOLUCIÓN
# “InventarisPro” v1.1

**Equipo de desarrollo:**
Antony Rodriguez.
Roberto Contreras.
Sebastian Morgado.

**Profesor:**
Guillermo Leyton.

**Asignatura:**
Programación Avanzada

# Índice

# Características del problema

A continuación se describirán las características del problema de optimización del inventario de un negocio.

- **¿El problema es lineal o no lineal?**
  - El problema es **no lineal**, ya que no todas las variables del problema tienen el mismo peso, para cada negocio es diferente el peso de estas, por lo tanto cambia al momento de optimizar un stock de un negocio a otro.

    **Ejemplo:**

    Para los negocios de Abarrotes A y B, los dos venden jugo y pan. En el negocio A se vende más pan que jugo, mientras que en B ocurre lo contrario. Por lo tanto la optimización del stock de los productos del negocio A sería diferente a la del negocio B, haciendo este problema no lineal.
    Este caso se repetiría en los distintos tipos de negocio que nuestro modelo abarca.[1]

- **¿El problema es estático o dinámico?**
  - El problema es **dinámico**, ya que optimizar el stock se ve afectado con el paso del tiempo.
    **Ejemplo:**
    En el caso de una tienda de ropa, importa la estación del año en que se encuentren, ya que dependiendo de está los clientes comprarán una determinada ropa y por ende la optimización del stock cambia.

- **¿El problema es sincrónico o asincrónico?**
  - El problema es **asíncrono**, porque el hecho de que factores como el agotamiento del stock de un producto, la insuficiencia de stock ante una alta demanda de un producto o que el dueño del establecimiento no pueda atender el negocio por razones personales, no necesariamente hace que falle el proceso de optimización. El proceso se puede retomar después de solucionar estos inconvenientes.

- **¿El problema es determinista?**
  - El problema NO es determinista, puesto que no siempre se obtendrá la misma ganancia(salida) al optimizar el inventario con los mismos productos, ya que el que se vendan los productos está rodeado de incertidumbre y además depende de los gustos o necesidades que tenga el cliente en el momento o también la temporada del año en que se encuentre.

  - Ejemplo: Si se optimiza un producto, como un helado, en verano este tendrá una salida, en donde dirá que es mejor vender estos productos en la temporada de verano, pero, si es invierno, el mismo producto no tendrá la

[1] Ver Alcance en Plan General

misma salida, ya que la gente no comprará con la misma frecuencia el producto. Por ende, ante una misma entrada, se obtienen diferentes salidas.

- **¿El problema es estable?**
  - El problema **NO** es estable, por que si existe perturbación alguna que modifique el stock del inventario (robo o merma), la optimización se ve afectada negativamente dependiendo de la importancia del producto.

- **¿El problema es isomórfico?**
  - El problema de optimizar el stock NO es isomórfico, porque para serlo, este debe ser lineal y biyectivo, cosa con la que no cumple nuestro sistema, basta con leer que no es lineal.

- **¿El problema es uni-sistémico?**
  - No, ya que participan dos sistemas, el vendedor(sistema biológico), la base de datos y el software (como sistema informático). El vendedor realiza las ventas siempre y cuando hayan existencias registradas en la base de datos, de esta manera el software puede recomendar como optimizar el stock.

- **¿El problema es azaroso?**
  - El problema no es azaroso ya que es más propio de la incertidumbre.
    - Ejemplo: No existe la probabilidad de saber cuando una persona pasará delante de la tienda y querrá comprar algún producto. Si bien la probabilidad de que una persona pase delante del negocio no es 0, no se puede calcular con herramientas propias de la estadística.

- **¿El problema es espontáneo?**
  - El problema no es espontáneo, ya que un objetivo fundamental de los jefes/administradores de un negocio es encontrar la cantidad justa y necesaria para satisfacer las necesidades de sus compradores, y de esa forma obtener rentabilidad de las ventas.

# Marco Teórico

Para representar nuestra solución, se realizó una búsqueda de varios modelos matemáticos, de entre los cuales se escogieron los siguientes como candidatos:

- Aprendizaje por Refuerzo (RL).
- Algoritmos Genéticos (AG).

| Modelo | Características |
|---|---|
| RL | Es el entrenamiento de modelos de aprendizaje automático para tomar una secuencia de decisiones. El agente aprende a lograr un objetivo en un entorno incierto y potencialmente complejo. La computadora emplea prueba y error para encontrar una solución al problema. Su objetivo es maximizar la recompensa total. |
| AG | Los Algoritmos Genéticos (AG) son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proceso genético de los organismos vivos. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza de acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin (1859). Por imitación de este proceso, los Algoritmos Genéticos son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas. |

El grupo, ve que para el desarrollo de este proyecto, teniendo como base a la meta planteada, la cual es acerca de optimizar el stock de un negocio para generar ganancias, que el **posible** modelo es Algoritmos Geneticos debido a la caracterización del problema.

## Justificación

En base de la investigación que se realizó, además de las características del problema planteadas anteriormente, se consideró el uso del modelo de Algoritmos Genéticos, puesto que el objetivo del proyecto es optimizar, y este modelo se adecua a nuestra meta, además de que su estructura es siempre la misma, independientemente del problema, lo que facilita su uso.

# Algoritmos Genéticos

Los Algoritmos Genéticos (AGs) son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proceso genético de los organismos vivos. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza de acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin (1859). Por imitación de este proceso, los Algoritmos Genéticos son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas.

Los Algoritmos Genéticos usan una analogía directa con el comportamiento natural. Trabajan con una población de individuos, cada uno de los cuales representa una solución factible a un problema dado. A cada individuo se le asigna un valor o puntuación, relacionado con la bondad de dicha solución. En la naturaleza esto equivaldría al grado de efectividad de un organismo para competir por unos determinados recursos. Cuanto mayor sea la adaptación de un individuo al problema, mayor será la probabilidad de que el mismo sea seleccionado para reproducirse, cruzando su material genético con otro individuo seleccionado de igual forma. Este cruce producirá nuevos individuos – descendientes de los anteriores – los cuales comparten algunas de las características de sus padres. Cuanto menor sea la adaptación de un individuo, menor será la probabilidad de que dicho individuo sea seleccionado para la reproducción, y por tanto de que su material genético se propague en sucesivas generaciones.

Un algoritmo genético usualmente contiene:

1. Una población de posibles soluciones al problema.
2. Una forma de evaluar que tan buena es cada posible solución.
3. Un método para combinar las soluciones y crear nuevas soluciones.
4. Un operador de mutación para evitar la pérdida de diversidad en las soluciones.

## Selección

El operador de selección es el encargado de escoger un conjunto de padres según su calificación, para que transmitan su información genética a la siguiente generación con el objetivo que los individuos o soluciones sean cada vez mejores. En otras palabras, el operador de selección es el encargado de hacer cumplir la ley natural de supervivencia del más fuerte.
Los métodos de selección más utilizados son la selección por ruleta y la selección por torneo.

### Selección por ruleta

Con este método la probabilidad de selección es proporcional a la adaptación de cada individuo. La idea básica de este método es tomar la suma de las calificaciones de todos los individuos de la población como el 100% de una circunferencia, y asignar a cada miembro de la población una porción de la circunferencia, donde el tamaño de dicha porción es proporcional a su adaptación.

### selección por torneo

La idea del mecanismo de selección por torneo es tomar dos individuos de forma aleatoria, donde cada individuo tiene la misma probabilidad de ser elegido. Posteriormente se realiza una competencia o torneo entre ambos, y el individuo con mejor fitness es el vencedor y se selecciona para continuar como padre de la siguiente generación. Este método es bastante sencillo y permite a todos los individuos de la población tener una cierta probabilidad de ser seleccionados.

## Elitismo

La selección por ruleta o por torneo no garantizan la selección de ningún individuo en particular, ni siquiera del mejor. En ocasiones este mejor individuo puede no ser elegido y perderse para la siguiente población, arrojando a la basura la mejor solución al problema que se tenía. Para evitar perder la mejor solución o incluso un grupo de las mejores soluciones se puede implementar un operador de elitismo.

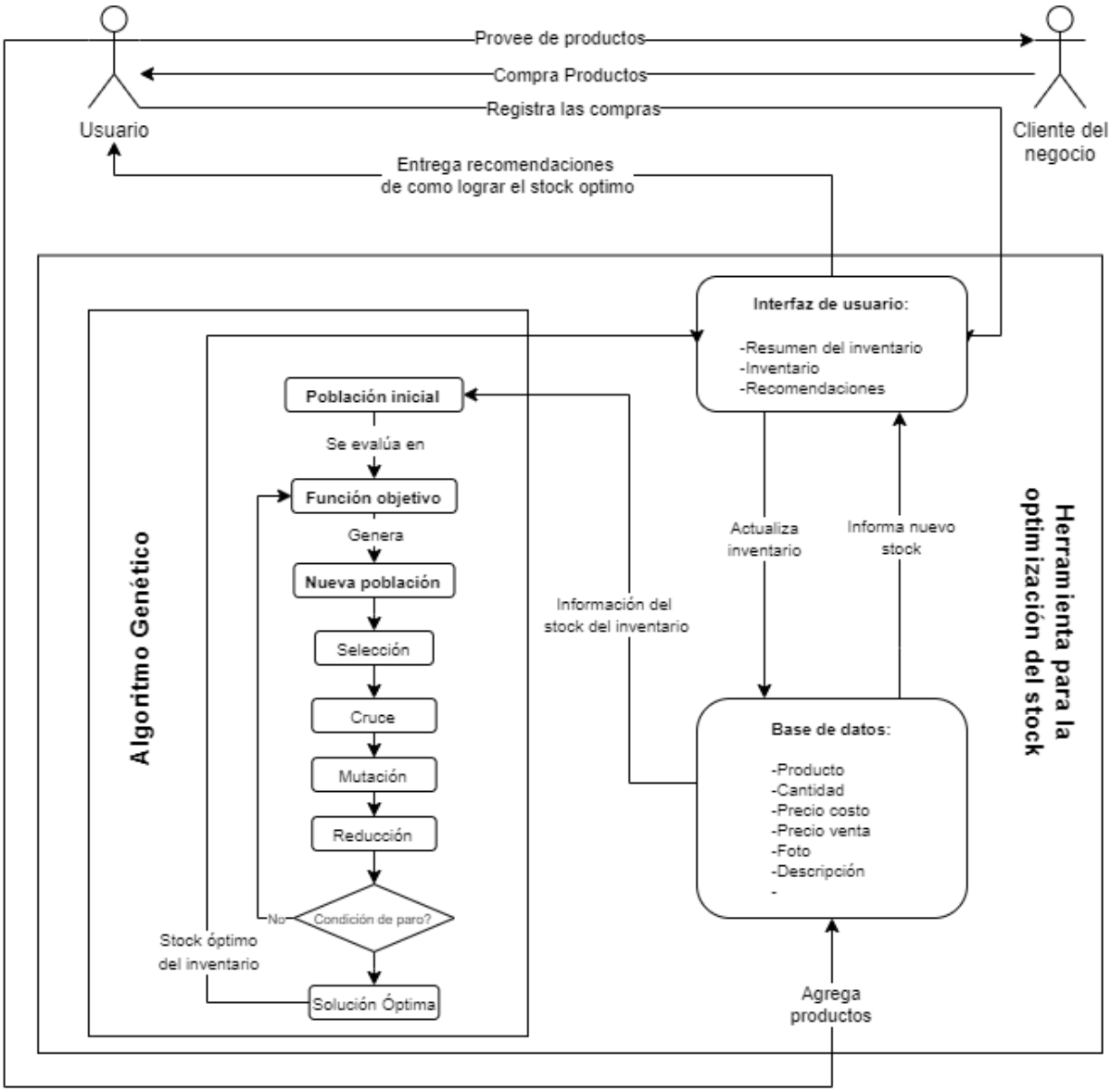
## Cruce

El operador de cruce se encarga de combinar soluciones para obtener nuevos individuos que compartan información de dos individuos seleccionados de la generación anterior. A los individuos generados se les llama comúnmente hijos y a los individuos a partir de los cuales fueron generados se les llama padres.

## Mutación

El operador de mutación le da a los algoritmos genéticos la posibilidad de que en ocasiones no comunes y con cierta probabilidad *“Pm”* se produzcan cambios o alteraciones en los individuos de la población.

# Diagrama del modelo




# Flujo del modelo

1. El usuario primeramente deberá de proveer información al sistema, esto es agregando uno a uno los productos que su negocio posea.
2. El usuario provee al cliente de su negocio, esto es que el negocio ponga a disposición los productos para que sean adquiridos.
3. El cliente del negocio podrá realizar la compra de los productos que requiera.
4. El usuario registra la venta en el sistema y se actualiza el stock anterior, reflejando una disminución de tanta cantidad que el cliente haya comprado.
5. Posteriormente según el periodo de tiempo que el usuario haya escogido en el sistema se realizará el análisis de ventas.
6. El sistema entrega la recomendación de productos que le generan más ganancia y cuanto podría llegar a ganar si invierte en ellos.

# Modelo solución

A continuación se presentarán los componentes de la solución propuesta. En ella están los actores y el sistema de conocimiento propuesto

## Actores

En el modelo propuesto, existen 2 tipos de actores, los cuales son:

- Usuario:
  - Es quien provee los productos de su negocio a los clientes y es quien administra el inventario con los datos que posee, ya sea el registro de las ventas, stock, inversión.
- Cliente del negocio:
  - Es la persona quien realiza compras en el negocio del usuario afectando de manera directa al stock del inventario.

# Metodología

La representación que hemos definido para los individuos, es un vector X = (x1, x2, . . . , xm) , donde cada xj representa el precio de compra de cada producto del inventario. La población inicial se genera de manera aleatoria. Luego se crea la población inicial con valores aleatorios entre 1 y el tamaño del cromosoma. Se toma desde uno porque se espera que al menos haya un producto en el inventario.
La función fitness debe incluir las variables del producto, como el precio de compra y cantidad vendida. Además, existe una restricción la cual es el capital del negocio. En una primera aproximación se utiliza la función fitness que muestra la ecuación también llamada función de adaptación.

Sean:

$$\sum_{i=0}^{n}\sum_{j=0}^{k} PrecioCompra_{i,j} * CantidadVendida_{j}$$

El esquema general del algoritmo genético utilizado se muestra en el siguiente pseudocódigo.

***Entrada:*** *productos(IdProducto,CantidadVendida,PrecioCompra,PrecioVenta)*
***Entrada:*** *Generaciones,Individuos, ProbMutacion, PorcElitismo*

1. *n = count(productos)*
2. *Nelite = ceil(Individuos*PorcElitismo/100)*
3. *generar T y HQ (* rutina para generar las matrices necesarias *)*
4. *CromTam = count(T)*
5. *PobIni = rand(1, Individuos)*
6. *PobOld = PobIni*

```
7.  call Fitness()
8.  gen = 0
9.  mientras gen < Generaciones hacer
10.         Pobnew = PobOld
11.         call Cruce()
12.         call Mutacion()
13.         call Fitness()
14.         sort(PobNew)
15.         gen = gen+1
16. fin mientras
17. devolver PobNew[0]
```

El algoritmo recibe como entrada datos, como la lista de productos del inventario, además de los parámetros básicos del algoritmo genético, como el número máximo de generaciones para el criterio de parada (Generaciones), el número de individuos en la población (Individuos), la probabilidad de mutación (Prob Mutación) y el porcentaje de elitismo (PorcElitismo). Luego de eso, se generan los vectores T y HQ.
El vector HQ corresponde al vector del lado derecho de las restricciones. En este caso, cada restricción *"i"* se refiere a la cantidad vendida en el intervalo de fecha que decida el usuario. El vector T corresponde al vector de coeficientes de la función objetivo, en este caso la utilidad que genera el $producto_i$.

Se calcula el tamaño del cromosoma que se utiliza en la función de mutación y al generar la población inicial. La generación de T y HQ se presenta en los siguientes pseudocódigos.

```
Función: Generar T
Entrada: productos
1.  Coeficientes
2.  para i = 0 hasta productos.filas i++
3.          coeficiente.Agregar(productos.utilidad)
4.  fin para
5.  devolver Coeficientes
```

```
Función: Generar HQ
Entrada: productos
1.  HQ[]
2.  para i = 0 hasta HQ.columnas i++
3.          si i = HQ.columnas - 1 hacer
4.                  HQ[i] = presupuesto
5.          si no
6.                  HQ[i,j] = productos[i].CantidadVendida
7.          fin si
8.  fin para
9.
10. devolver HQ
```

La función de cruce utilizada es del tipo uniforme, para lo cual se genera una cadena aleatoria de unos y ceros del tamaño del cromosoma para escoger el padre del cual se toma cada uno de los genes para crear cada hijo. El criterio de selección de los padres es por torneo, donde se seleccionan aleatoriamente dos pares de individuos y se enfrentan entre sí para obtener el mejor de cada par para ser padres de la siguiente generación. Estas funciones se muestran en el siguiente algoritmo.

***Función cruce***
***Entradas:*** *PobOld, PobNew, Fitness, Individuos, Nelite, CromTam*

```
1.  Nijos = 0
2.  mientras Nijos + Nelite < Individuos hacer
3.          r1,r2,r3,r4 = rand(Individuos)
4.          patron[CromTam] = rand(0,1)
5.          si PobOld[r1](Fitness) > PobOld[r2](Fitness) entonces
6.                  padre1 = PobOld[r1]
7.          si no
8.                  padre1 = PobOld[r2]
9.          fin si
10.         si PobOld[r3](Fitness) > PobOld[r4](Fitness) entonces
11.                 padre2 = PobOld[r3]
12.         si no
13.                 padre2 = PobOld[r4]
14.         fin si
15.         hijo1 = patron * padre1 + (1 - patron)* padre2
16.         hijo2 = (1 - patron) * padre1 + patron*padre2
17.         agregar hijo1 e hijo2 a PobNew
18.         Nijos = Nijos + 2
19. fin mientras
20. devolver PobNew
```

La función de mutación se define con una probabilidad de mutación Pm.
Cada gen de cada individuo de la población tiene asociada una probabilidad Pm de sufrir una mutación. Para llevar a cabo dicha mutación se suma o resta una cantidad aleatoria entre 0 y 1.

***Función Mutación***
***Entradas:*** *PobOld, A, PobNew, ProbMutac, Individuos, CromTam*

```
1.  para i = 0 hasta Individuos i++
2.          para j = 0 hasta CromTam j++
3.                  p = rand(1)
4.                  si p < ProbMutac hacer
5.                          x = rand(0,1)
6.                          PobNew[i,j] = Pob[i,j] + x
7.                  si no
8.                          si A[j,0] = 0
9.                                  PobNew[i,j] = rand( 0, |Pob[i,j]|)
10.                         fin si
11.                 si no
12.                         PobNew[i,j] = Pob[i,j]
```

13. ***fin si***
14. ***fin para***
15. ***fin para***
16. ***devolver*** *PobNew*

La función fitness utilizada en esta primera aproximación es la definida en la ecuación.
El algoritmo muestra la función para calcular el fitness de cada individuo, de acuerdo con la función objetivo.

***Función Fitness***
***Entradas:*** *PobOld, T*

1. ***para*** *i = 0* ***hasta*** *Individuos i++*
2. ***para*** *j = 0* ***hasta*** *CromTam j++*
3. *sumatoria = sumatoria + PobOld[i,j] * T[j]*
4. ***fin para***
5. ***fin para***
6. ***devolver*** *PobNew*